数理解析研究所講究録
第 646 巻 1988 年 52-73

Four New Algorithms for Multivariate Polynomial GCD

佐々木建昭
Tateaki Sasaki

鈴木正幸
Masayuki Suzuki

The Institute of Physical and Chemical Research

2-1, Hirosawa, Wako-shi, Saitama 351-01, Japan

ABSTRACT

Four new algorithms for multivariate polynomial GCD (greatest common divisor) are given. The first is a simple improvement of PRS (polynomial remainder sequence) algorithms. The second is to calculate a Gröebner basis with a certain term ordering. The third is to calculate subresultant by treating the coefficients as truncated power series. The fourth is to calculate PRS by treating the coefficients as truncated power series. The first and second algorithms are not important practically, but the third and fourth ones are quite efficient and seem to be useful practically.

1. Introduction

Study of algorithm for multivariate polynomial GCD (greatest common divisor) has a long history. The idea of generalizing Euclidean algorithm for integer GCD to polynomial GCD has appeared in as early as 16th century. However, Collin's study [1] will be the first modern analysis of the Euclidean algorithm for multivariate polynomial GCD. Collin's algorithm, or the reduced-PRS algorithm, was soon improved to the subresultant-PRS algorithm by Brown and Traub [2, 3]. Another improvement of the Euclidean algorithm is Hearn's trial-division algorithm [4] which is practically efficient. For the GCD computation, modular algorithms are very important. Brown's algorithm [5] will be the first modular GCD algorithm. Subsequently, Moses and Yun proposed the so-called EZGCD algorithm [6, 7]. This algorithm utilizes the generalized Hensel construction and will be the best algorithm for large multivariate polynomials. For very sparse multivariate polynomials, Zippel's sparse modular algorithm is efficient [8]. Yet another modular algorithm has been proposed by Char, Geddes and Gonnet [9]. This algorithm uses the integer GCD computation. Furthermore, an algorithm using Gröbner basis has been proposed by Gianni and Trager [10].

Since the GCD computation is one of the most important operations in computer algebra, we should search for the most efficient algorithm. In this paper, we propose four new algorithms for multivariate GCD. These algorithms are based on simple ideas. The first one is a simple improvement of PRS (polynomial remainder sequence) algorithms. The second one calculates a Gröbner basis with a certain term ordering, but it is different from Gianni-Trager's algorithm. In the third and fourth algorithms, we treat the coefficients of polynomials as truncated power series. This device allows us to develop very efficient GCD algorithms for multivariate polynomials. Since the underlying ideas and the algorithms are very simple, we think that algorithms using truncated power series will quite useful in actual computation.

We use the following notations in this paper.

deg(P) : degree (w.r.t. variable x) of polynomial P;

lc(P) : leading coefficient (w.r.t. variable x) of P;

pp(P)	:	primitive part (w.r.t. variable x) of P;
cont(P)	:	GCD of the coefficients (w.r.t. variable x) of P;
P_1, P_2	:	polynomials in $K[x,y,\ldots,z]$ with K a number field, we assume that P_1 and P_2 are primitive.

2. An improvement of PRS algorithms

Let P_1 and P_2 be primitive polynomials in $K[x, y, \ldots, z]$, with the main-variable x, and represent them as

$$\begin{cases} P_1 = a_m x^m + a_{m-1}x^{m-1} + \cdots + a_0, & a_m \neq 0, \\ P_2 = b_n x^n + b_{n-1}x^{n-1} + \cdots + b_0, & b_n \neq 0, \end{cases} \tag{1}$$

where we assume $m \geq n$. We define the polynomial $S^{(j)}$ as

$$S^{(j)} = D_j^{(j)}x^j + D_{j-1}^{(j)}x^{j-1} + \cdots + D_0^{(j)}, \tag{2}$$

where $D_i^{(j)}$, $i=j,j-1,\ldots,0$, are the following determinants:

$$D_i^{(j)} = \left|\begin{array}{ccccccc} a_m & a_{m-1} & \cdots & \cdots & \cdots & a_{2j+2-n} & a_{i+j+1-n} \\ & a_m & a_{m-1} & \cdots & \cdots & a_{2j+3-n} & a_{i+j+2-n} \\ & & \cdots & \cdots & \cdots & \cdots & \cdots \\ & & & a_m & \cdots & a_{j+1} & a_i \\ b_n & b_{n-1} & \cdots & \cdots & \cdots & b_{2j+2-m} & b_{i+j+1-m} \\ & b_n & b_{n-1} & \cdots & \cdots & b_{2j+3-m} & b_{i+j+2-m} \\ & & \cdots & \cdots & \cdots & \cdots & \cdots \\ & & & b_n & \cdots & b_{j+1} & b_i \end{array}\right| \begin{array}{l} \left.\begin{array}{c} \\ \\ \\ \\ \end{array}\right\} n-j \text{ rows} \\ \left.\begin{array}{c} \\ \\ \\ \\ \end{array}\right\} m-j \text{ rows} \end{array} \tag{3}$$

Here, we mean $a_i = b_i = 0$ if $i<0$. The $S^{(j)}$ is the j-th order subresultant of P_1 and P_2, and deg($S^{(j)}$) is usually j.

Theorem 1. Let deg(GCD(P_1, P_2)) = d and g = GCD(lc(P_1), lc(P_2)). Then,

$$g \mid D_i^{(j)}, \quad i=j-1,\ldots,0, \tag{4}$$

$$[D_d^{(d)} / g] \mid D_i^{(d)}, \quad i=d-1,\ldots,0. \tag{5}$$

Proof: Expanding the determinant in Eq.(3) w.r.t. the first column, we see GCD$(a_m, b_n) | D_i^{(j)}$. This proves (4). Next, we note that $S_d^{(d)}$ is a multiple of G = GCD(P_1, P_2) (see, for example, [2]). Since lc$(G) | g$, $[g$ / lc$(G)] \cdot G$ is a multiple of G and $S^{(d)} / [D_d^{(d)} / g] = [g$ / lc$(G)] \cdot G$. This proves (5). □

Every PRS algorithm for GCD calculates a PRS

$$(P_1, P_2, ..., P_k \neq 0, P_{k+1} = 0). \tag{6}$$

Then, pp(P_k) = GCD(P_1, P_2). The sizes of coefficients of P_k will often very large and computation of pp(P_k) is time-consuming. The algorithm to be given in this section is a device to calculate pp(P_k) efficiently.

When P_k is equal to or a multiple of subresultant $S^{(d)}$, Theorem 1 tells that we can remove the factor lc$(P_k) / g$ from P_k. The resulting polynomial

$$\tilde{P} = P_k / [\mathrm{lc}(P_k) / g] \tag{7}$$

will have the coefficients of much smaller sizes than those of P_k because lc$(\tilde{P}) = g$. This device is applicable to reduced-PRS algorithm [1] and subresultant-PRS algorithm [3]. When P_k is not equal to or multiple of $S^{(d)}$, we can calculate $\tilde{P}$ by the formula

$$\tilde{P} = gP_k / \mathrm{lc}(P_k). \tag{8}$$

This device is applicable, for example, to the trial-division algorithm [4]. Note that the $\tilde{P}$'s defined by Eqs.(7) and (8) are the same. Thus, we have the following algorithm which is applicable to any PRS algorithm.

Algorithm 1 (improvement of PRS algorithms).

```
Step 1.  Calculate a PRS $(P_1, P_2, ..., P_k \neq 0, P_{k+1} = 0)$ and
         if deg$(P_k) = 0$ then return 1;
Step 2.  Calculate $g$ = GCD(lc$(P_1)$, lc$(P_2)$) and
         if $g$ | lc$(P_k)$ then $\tilde{P} \leftarrow P_k / [\mathrm{lc}(P_k) / g]$ else $\tilde{P} \leftarrow gP_k / \mathrm{lc}(P_k)$;
```

Step 3. Return pp($\tilde{P}$). □

Although this algorithm is rather trivial, we have described it because it plays some roles in other algorithms given in this paper.

Let us briefly mention about the computation of pp(P_k), $P_k = C_d x^d + C_{d-1} x^{d-1} + \cdots + C_0$, where C_d, ..., C_0 are large-sized polynomials. Suppose we calculate cont(P_k) as

$$\mathrm{cont}(P_k) = \mathrm{GCD}(C_0, \mathrm{GCD}(\cdots, \mathrm{GCD}(C_{d-1}, C_d)\cdots)).$$

Then, if cont(P_k) = C_d or cont(P_k) = C_{d-1}, we can calculate cont(P_k) easily by performing the trial-division of C by D in the calculation of GCD(C, D). In such cases, our device in Algorithm 1 is not useful. However, if cont(P_k) = $C_d / \tilde{C}_d = C_{d-1} / \tilde{C}_{d-1}$, with $\tilde{C}_d \neq 1$ and $\tilde{C}_{d-1} \neq 1$, then Algorithm 1 improves any PRS algorithms except for the primitive PRS algorithm, so long as the computation of g is not costly. When the PRS is already primitive, the above Steps 2 and 3 are unnecessary and we had better apply our device to construct the primitive PRS. That is, after calculating $\tilde{P}_{i+1}$ as the pseudo-remainder of P_{i-1} and P_i, we make the trial-division of $g\tilde{P}_{i+1}$ by lc($\tilde{P}_{i+1}$) and if the division succeeds then we calculate P_{i+1} as $P_{i+1} = \mathrm{pp}(g\tilde{P}_{i+1} / \mathrm{lc}(\tilde{P}_{i+1}))$.

3. Gröbner basis method

Gianni and Trager proposed a method of using Gröbner basis for multivariate GCD computation [10]. For Gröbner basis, see [11]. We propose another algorithm in this section. Our algorithm calculates a Gröbner basis of the ideal (P_1, P_2) in $K[y,...,z][x]$, i.e., we regard P_1 and P_2 as polynomials in variable x with coefficients in $K[y,...,z]$. (The Gröbner basis in $K[y,...,z][x]$ is equivalent to Gröbner basis in $K[x, y,...,z]$ with the variable ordering $x > y$, ..., z. The term ordering in $K[y,...,z]$ may be arbitrary.)

Theorem 2. Let a Gröbner basis of the ideal (P_1, P_2) in $K[y,...,z][x]$ be $\Gamma = \{P_1, P_2, ..., P_s\}$. Let $\deg(P_i) = d_i$, $i = 1,...,s$, and d_k be the minimum value among $\{d_1, d_2, ..., d_s\}$. Then, there exists a polynomial C, $C \in K[y,...,z]$, such that $P_k = C \cdot \mathrm{GCD}(P_1, P_2)$.

Proof: Put $G = \mathrm{GCD}(P_1, P_2)$. The ideal (P_1, P_2) in $K(y,...,z)[x]$ is a principal ideal (G). Since $[ideal(P_1, P_2) \text{ in } K[y,...,z][x]] \subseteq [\, ideal(P_1, P_2) \text{ in } K(y,...,z)[x]] = (G)$, there exists a polynomial C, $C \in K[y,...,z][x]$, such that $P_k = CG$. On the other hand, there exist polynomials A and B, $A, B \in K(y,...,z)[x]$, such that $G = AP_1 + BP_2$. Multiplying $\tilde{C} = \mathrm{LCM}(\text{denominators of } A \text{ and } B)$ to this equation, we see that $\tilde{C}G \in [ideal(P_1, P_2) \text{ in } K[y,...,z][x]]$. Since $\deg(\tilde{C}G) = \deg(G)$ and $\tilde{C}G$ must be M-reduced to 0 by Γ, $\deg(P_k) \leq \deg(\tilde{C}G) = \deg(G)$. Hence, $C \in K[y,...,z]$. □

The above theorem gives us the following GCD algorithm.

Algorithm 2 (Gröbner basis method).

Step 1. Calculate a Gröbner basis $\Gamma = \{P_1, P_2, ..., P_s\}$ of the ideal (P_1, P_2) in $K[y,...,z][x]$;

Step 2. Let P_k be a minimum degree element, w.r.t. x, of Γ and if $\deg(P_k) = 0$ then return 1 else return $\mathrm{pp}(P_k)$. □

(Note) In the calculation of $\mathrm{pp}(P_k)$ in Step 2, we can use the method given in the previous section.

Example 1.

$$P_1 = (x - y + 2z)\,((2y + z)x - y^2 + 2y - 3z),$$
$$P_2 = (x - y + 2z)\,((y + 2z)x - 3y + 2z^2).$$

The Gröbner basis of (P_1, P_2) in $Q[y, z][x]$, with term-degree order for terms in $Q[y, z]$, is calculated as $\{P_3, P_1 - 2P_2, P_2\}$, where

$$P_3 = (y^3 + 2y^2z - 8y^2 + 4yz^2 - 4yz + 2z^3 + 6z^2)x$$

$$+ (-y^4 + 8y^3 - 12y^2z + 6yz^3 - 14yz^2 + 4z^4 + 12z^3).$$

The lowest degree element of the basis is P_3, and we obtain

$$\mathrm{GCD}(P_1, P_2) = \mathrm{pp}(P_3) = x - y + 2z. \square$$

From the viewpoint of variable elimination, the Gröbner basis method is similar to the PRS method: the former applies the head term elimination, and the latter applies the leading term elimination, see [12]. However, the PRS method causes intermediate expression growth: in the calculation of the pseudo-remainder, we multiply a power of leading coefficient of the divisor and we factor out it later. The Gröbner basis method does not cause this kind of expression growth, but it generate a number of polynomials and the reduction procedure is time-consuming.

The calculation of Gröbner basis in Algorithm 2 is averagically most efficient if we adopt the term-degree order for the terms in $K[y,...,z]$. Furthermore, if we know the value of $d = \deg(\mathrm{GCD}(P_1, P_2))$, then we may stop the Gröbner basis construction when a polynomial of degree d is constructed. These devices will make the Algorithm 2 quite efficient compared with the original version. However, our experience shows that the Gröbner basis method is quite inefficient when the sizes of P_1 and P_2 are large. This low efficiency can be understood by the fact that the lowest degree element P_k in Γ is a polynomial of almost the same size as $S^{(d)}(P_1, P_2)$, the subresultant of the d-th order. Hence, the Gröbner basis method is not much more efficient than the subresultant PRS method.

4. Terminology about truncated power series

We use the truncated power series in our third and fourth algorithms, hence we introduce some terminology about the truncated power series and derive a useful degree bounds. We denote the power series ring in the variables $y, ..., z$ by $K\{y,...,z\}$, where K is a coefficient field. By the term-degree of $T = cy^{e_y} \cdots z^{e_z}$, with $c \in K$, we mean $e_y + \cdots + e_z$. We impose the reverse term-degree order > for the terms in the power series. Hence, $1 > y > \cdots > z > y^2 > \cdots > yz > \cdots > z^2 > y^3 > \cdots$.

The addition and multiplication of truncated power series are the same as those for polynomials, except for the cutoff of terms whose term-degrees and variable exponents are higher than some predetermined values. With the reverse term-degree order >, the division of power series is performed as follows.

Division of power series. Let $A, B, C \in K\{y, \cdots, z\}$. We express A as

$$A = A^{(a)} + A^{(a+1)} + A^{(a+2)} + \cdots ,$$

where $A^{(d)}$ denotes the terms of term-degree d of A. Similarly, we define $B^{(d)}$ and $C^{(d)}$. Let $A = B \cdot C$, then we calculate the quotient $C = C^{(a-b)} + C^{(a-b+1)} + \cdots$ by the power series division of A and B as follows. First, since $A^{(a)} = B^{(b)}C^{(a-b)}$, we calculate $C^{(a-b)}$ by the polynomial division of $A^{(a)}$ by $B^{(b)}$:

$$\begin{aligned} A &= C^{(a-b)}[B^{(b)} + B^{(b+1)} + \cdots] \\ &\quad + [A^{(a+1)} - C^{(a-b)}B^{(b+1)}] + [A^{(a+2)} - C^{(a-b)}B^{(b+2)}] + \cdots . \end{aligned}$$

Second, we calculate $C^{(a-b+1)}$ by the polynomial division of $[A^{(a+1)} - C^{(a-b)}B^{(b+1)}]$ by $B^{(b)}$, and continue this procedure. □

Definition 1 [exponent range and term-degree range]. Let

$$C = \sum_{i=1} c_i v^{e_i} \times (\text{monomial not containing } v), \; c_i \in K.$$

We define the exponent range of the variable v of C, abbreviated to $\mathrm{ran}_v(C)$, as $\mathrm{ran}_v(C) = (e_{\min}, e_{\max})$ where $e_{\min} = \mathrm{MIN}\{e_i \mid i=1,2, \cdots \}$ and $e_{\max} = \mathrm{MAX}\{e_i \mid i=1,2, \cdots \}$. Similarly, we define the term-degree range of C, abbreviated to $\mathrm{tran}(C)$, as $\mathrm{tran}(C) = (E_{\min}, E_{\max})$ where $E_{\min}(E_{\max})$ is the minimum (maximum) term-degree of C. □

Example. For $C = y+2y^2+3yz-y^3-3yz^2+3y^3z-4y^2z^2+5yz^3$, we have $\mathrm{ran}_y(C) = (1, 3)$, $\mathrm{ran}_z(C) = (0, 3)$, and $\mathrm{tran}(C) = (1, 4)$.

Lemma 1. For polynomials/finite power series C_1 and C_2, we have

$$\begin{cases} \mathrm{ran}_v(C_1, C_2) = \mathrm{ran}_v(C_1) + \mathrm{ran}_v(C_2), \\ \mathrm{tran}(C_1, C_2) = \mathrm{tran}(C_1) + \mathrm{tran}(C_2), \end{cases} \tag{9}$$

where we define the addition of numeric lists as $(m_1, n_1)+(m_2, n_2) = (m_1+m_2, n_1+n_2)$.

Proof: Obvious. □

Definition 2 [significant terms]. Let $C \in K\{y,...,z\}$ and $\mathrm{tran}(C) = (E, some)$. We define $(\tilde{E}, \tilde{e}_y, ..., \tilde{e}_z)$ significant terms of C to be the terms of C such that the term-degree is in the range $(E, E+\tilde{E})$ and, for each variable $v \in \{y,..., z\}$, the v-exponent is in the range $(0, E+\tilde{e}_v)$. We use the same terminology for $P \in K\{y, ..., z\}[x]$ also, where tran and ran are defined for the coefficients of P. □

Example. For $C = y+2y^2+3yz-y^3-3yz^2+3y^3z-4y^2z^2+5yz^3$, the (2, 1, 2) significant terms of C are

$$y + 2y^2 + 3yz - 3yz^2. \square$$

(Note) Among the bounds $(\tilde{E}, \tilde{e}_y, ..., \tilde{e}_z)$, the term-degree bound $\tilde{E}$ is the strongest because we assumed the reverse term-degree order. Hence, if $\tilde{e}_v \geq \tilde{E}$ then we may omit the exponent bound for the variable v.

Lemma 2. Let C_1, C_2, D be in $K\{y,...,z\}$ and satisfy $C_1 = C_2 D$. Let $\mathrm{tran}(D) = (E_l, E_h)$ and $\mathrm{ran}_v(D) = (some, e_v)$, $v = y, ..., z$. Then, in order to calculate D from C_1 and C_2 by the power series division, we need only the $(E_h - E_l, e_y - E_l, \cdots, e_z - E_l)$ significant terms of C_i, i=1,2.

Proof: Obvious from the above division operation for power series. □

Lemma 3. For P_1 and P_2 in $K\{y, ..., z\}[x]$, put $G = \mathrm{GCD}(P_1, P_2)$. Let

$$E = \textit{maximum term-degree of the coefficient terms of } G,$$

E_i = *maximum term-degree of the coefficient terms of* P_i,
E_i' = *maximum term-degree of the terms of* $\mathrm{lc}(P_i)$,
E'' = *maximum term-degree of the terms of* $\mathrm{lc}(G)$.

Furthermore, for each variable v in $\{y, ..., z\}$, let

e_v = *maximum* v*-exponent of the coefficient terms of* G,
e_{vi} = *maximum* v*-exponent of the coefficient terms of* P_i,
e_{vi}' = *maximum* v*-exponent of the terms of* $\mathrm{lc}(P_i)$,
e_v'' = *maximum* v*-exponent of the terms of* $\mathrm{lc}(G)$.

Then, we have

$$E \leq \mathrm{MIN}\{E_i - E_i' + E'' \mid i=1,2\}, \qquad (10)$$
$$e_v \leq \mathrm{MIN}\{e_{vi} - e_{vi}' + e_v'' \mid i=1,2\}. \qquad (11)$$

Proof: Let the maximum term-degree of the terms of $\mathrm{lc}(P_i\ /\ G) = \mathrm{lc}(P_i)\ /\ \mathrm{lc}(G)$ be $\tilde{E}_i$, then $\tilde{E}_i = E_i' - E''$. Since

$$\begin{aligned}[\text{term-degree of } G] &= [\text{term-degree of } P_i] - [\text{term-degree of } (P_i\ /\ G)] \\ &\leq E_i - [\text{term-degree of } \mathrm{lc}(P_i\ /\ G)],\end{aligned}$$

we obtain (10). Similarly, we can derive (11). □

Note that Lemma 3 is useless in actual GCD computation because we do not know $\mathrm{lc}(G)$ in advance. However, the following Lemma 4 is useful.

Lemma 4. For P_1 and P_2 in $K\{y,..., z\}[x]$, put $G = \mathrm{GCD}(P_1, P_2)$, $g = \mathrm{GCD}(\mathrm{lc}(P_1), \mathrm{lc}(P_2))$, $\gamma = g\ /\ \mathrm{lc}(G)$, and $\tilde{P} = \gamma G$. Let

E = *maximum term-degree of the coefficient terms of* $\tilde{P}$,
E_i = *maximum term-degree of the coefficient terms of* P_i,
E_i' = *maximum term-degree of the terms of* $\mathrm{lc}(P_i)$,
E'' = *maximum term-degree of the terms of* g.

Furthermore, for each variable v in $\{y, ..., z\}$, let

e_v = *maximum* v*-exponent of the coefficient terms of* $\tilde{P}$,
e_{vi} = *maximum* v*-exponent of the coefficient terms of* P_i,

$e_{vi}' = \textit{maximum } v\textit{-exponent of the terms of } \mathrm{lc}(P_i),$
$e_v'' = \textit{maximum } v\textit{-exponent of the terms of } g.$

Then, we have

$$E \le \mathrm{MIN}\{E_i - E_i' + E'' \mid i=1,2\}, \tag{12}$$
$$e_v \le \mathrm{MIN}\{e_{vi} - e_{vi}' + e_v'' \mid i=1,2\}. \tag{13}$$

Proof: Let the maximum term-degrees of the terms of γ and $\mathrm{lc}(\gamma P_i \,/\, \tilde{P})$ be $\tilde{e}$ and $\tilde{E}_i$, respectively. Then $\tilde{E}_i = \tilde{e} + E_i' - E''$ because $\mathrm{lc}(\gamma P_i \,/\, \tilde{P}) = \gamma\, \mathrm{lc}(P_i) \,/\, g$. Since

$$\begin{aligned}[\text{term–degree of } \gamma P_i] &= [\text{term–degree of } \tilde{P}\,] + [\text{term–degree } of\ \ \gamma P_i \,/\, \tilde{P}\,] \\ &\ge E + [\text{term–degree of } \mathrm{lc}(\gamma P_i \,/\, \tilde{P}\,)] \\ &= E + \tilde{E}_i,\end{aligned}$$

we obtain $\tilde{e} + E_i \ge E + \tilde{E}_i = E + \tilde{e} + E_i' - E''$. This proves (12). Similarly, we can derive (13). □

Our algorithms require only rational arithmetic of power series. We have no problem in the multiplication and division of power series, but the addition and subtraction may cause a problem which we call "accuracy decreasing problem". Suppose we add power series P_1 and P_2 such that $\mathrm{tran}(P_1) = \mathrm{tran}(P_2) = (E_l, E_h)$ and obtain the result $P = P_1 + P_2$ such that $\mathrm{tran}(P) = (E_l + 1, E_h)$. Then, we cannot use P any more for the calculation up to $(E_h - E_l, \ \cdots\ )$ significant terms.

5. Subresultant method with power series coefficient

Suppose we know the value of $d = \deg(\mathrm{GCD}(P_1, P_2))$. We can use a modular algorithm to calculate d cheaply: calculate $\mathrm{GCD}(P_1(x, n_y, ..., n_z), P_2(x, n_y, ..., n_z))$ for several sets of numbers $(n_y, ..., n_z)$ and set d to the lowest degree of the GCD's calculated.) Then, we can construct the subresultant $S^{(d)}$, see Eqs.(2) and (3), by calculating the determinants $D_i^{(d)}$, $i = d, d-1, ..., 0$, defined by Eq.(3), and we can obtain $G = \mathrm{GCD}(P_1, P_2)$ as $G = \mathrm{pp}(S^{(d)})$. This method has been known for long years, but it is practically not efficient.

In 2, we have seen that $g = \mathrm{GCD}(\mathrm{lc}(P_1), \mathrm{lc}(P_2))$ divides $\mathrm{lc}(S^{(d)})$ and

$$\tilde{P} = S^{(d)} / [\mathrm{lc}(S^{(d)}) / g] \tag{14}$$

is a multiple of G, hence $G = \mathrm{pp}(\tilde{P})$. Representing $\tilde{P}$ as

$$\tilde{P} = g_d x^d + g_{d-1} x^{d-1} + \cdots + g_0, \tag{15}$$

we see that (for $D_i^{(j)}$ below, see Eqs.(2) and (3))

$$g_d = g, \quad g_i = D_i^{(d)} / [D_d^{(d)} / g], \quad i=d-1,...,0. \tag{16}$$

Eqs.(15) and (16) show that what we need are g_d and g_i, $i=d-1,...,0$, and not $D_d^{(d)}$ and $D_i^{(d)}$ themselves. The sizes of g_i, $i=d,...,0$, are usually much smaller than those of $D_d^{(d)}$ and $D_i^{(d)}$. The calculation of g_i by Eq.(16) does not require all the terms of $D_d^{(d)}$ and $D_i^{(d)}$ but we need only some higher exponent terms (or lower exponent terms). Exploiting this fact, we may discard the unnecessary terms in the calculation of determinants $D_d^{(d)}$ and $D_i^{(d)}$, which will make the calculation fairly efficient. We discard the unnecessary terms systematically by treating the coefficients of P_1 and P_2 as truncated power series.

Let E, e_y, ..., e_z be defined as in Lemma 4. Lemma 2 and 4 show that, in order to calculate $\tilde{P}$, we have only to calculate the determinants $D_i^{(d)}$, $i=d,d-1,...,0$, up to $(E, e_y, ..., e_z)$ significant terms. Note that, since P_1 and P_2 are primitive, we have

$$\begin{cases} \mathrm{ran}_v(P_i) = (0, \mathit{some}), \quad i=1,2, \\ \mathrm{ran}_v(G) = (0, \mathit{some}), \end{cases} \tag{17}$$

for each variable $v \in \{y, ..., z\}$. Thus, we obtain the following algorithm.

Algorithm 3 (subresultant method with power series coefficient).

```
Step 1.  Estimate $d = \deg(\mathrm{GCD}(P_1, P_2))$ by a modular method (hence, the estimated
         value is an upper bound);
         If $d = 0$ then return 1
            else $g \leftarrow \mathrm{GCD}(\mathrm{lc}(P_1), \mathrm{lc}(P_2))$,
```

$E \leftarrow \mathrm{MIN}\{E_i - E_i' + E'' \mid i=1,2\}$,

$e_v \leftarrow \mathrm{MIN}\{e_{vi} - e_{vi}' + e_v'' \mid i=1,2\}$, $v \in \{y, \ldots, z\}$,

where E_i etc. are defined in Lemma 4;

Step 2. Construct determinants $D_i^{(d)}$, $i=d,d-1,\ldots,0$, of the d-th order subresultant $S^{(d)}$, cf. Eqs.(2) and (3), and calculate the determinants $D_i^{(d)}$ up to the $(E, e_y, \ldots, e_z)$ significant terms for $(y, \ldots, z)$;

Step 3. Calculate $\tilde{P} = \sum_{i=0}^{d} x^i D_i^{(d)} / [D_d^{(d)} / g]$ up to the $(E, e_y, \ldots, e_z)$ significant terms, and $G \leftarrow \mathrm{pp}(\tilde{P})$;

Step 4. If $G \mid P_1$ and $G \mid P_2$ then return G

else $d \leftarrow d-1$, and go to Step 2. □

(Note) In Appendix A, we give a method for calculating the determinants $D_i^{(d)}$.

Example 2.

$$\begin{aligned} P_1 = {} & yx^4 + (-y^2 - yz + 2y + z)x^3 + (-2y^2 - 2yz - z^2 + y + z)x^2 \\ & + (y^3 + y^2z - y^2 - z^2)x + (yz^2 + z^3 - z^2), \end{aligned}$$

$$\begin{aligned} P_2 = {} & zx^4 + (-yz - z^2 + z - 3)x^3 + (4y + 3z - 3)x^2 \\ & + (-y^2 - yz + 2y - z)x + (-y^2 + z^2 + y - z). \end{aligned}$$

We see $g = \mathrm{GCD}(\mathrm{lc}(P_1), \mathrm{lc}(P_2)) = \mathrm{GCD}(y, z) = 1$. Furthermore, $\mathrm{ran}_y(P_1) = (0, 3)$, $\mathrm{ran}_z(P_1) = (0, 3)$, $\mathrm{tran}(P_1) = (1, 3)$, $\mathrm{ran}_y(P_2) = (0, 2)$, $\mathrm{ran}_z(P_2) = (0, 2)$, $\mathrm{tran}(P_2) = (0, 2)$. Hence,

$$\begin{aligned} e_y &= \mathrm{MIN}\{3 - 1 + 0, 2 - 0 + 0\} = 2, \\ e_z &= \mathrm{MIN}\{3 - 0 + 0, 2 - 1 + 0\} = 1, \\ E &= \mathrm{MIN}\{3 - 1 + 0, 2 - 1 + 0\} = 1. \end{aligned}$$

Therefore, we have only to calculate the subresultant up to (1, 1, 1) significant terms. Suppose we found that $\deg(G) = 1$ by a modular method, so we calculate $D_1^{(1)}$ and $D_0^{(1)}$ up to (1, 1, 1) significant terms. The result is

$$\begin{aligned} D_1^{(1)} &= (3y^5 - 24y^3z^2 + 48yz^4) \\ &\quad + (-3y^4z^2 - 8y^2z^4 + 16yz^5 + 16z^6) \\ &\quad + (\text{terms of term–degree} \geq 7), \end{aligned}$$

$$\begin{aligned} D_0^{(1)} &= (3y^5 - 24y^3z^2 + 48yz^4) \\ &\quad + (-3y^6 - 3y^5z + 21y^4z^2 + 24y^3z^3 - 56y^2z^4 - 32yz^5 + 16z^6) \\ &\quad + (\text{terms of term–degree} \geq 7). \end{aligned}$$

Calculating $D_0^{(1)} \,/\, D_1^{(1)}$ up to (1, 1, 1) significant terms, we find

$$D_0^{(1)} \,/\, D_1^{(1)} = (1 - y - z) + (\text{terms of term–degree} \geq 2).$$

Therefore, we have $\tilde{P} = 1 \cdot x + (1 - y - z)$ as a multiple of GCD. Since $\tilde{P}$ is already primitive, we have

$$G = \mathrm{pp}(\tilde{P}) = x + (1 - y - z). \,\square$$

Example 3.

$$\begin{aligned} P_1 &= [y \to y + 1,\ z \to z - 1]\ (P_1 \text{ in Example 1}), \\ P_2 &= [y \to y + 1,\ z \to z - 1]\ (P_2 \text{ in Example 1}). \end{aligned}$$

In this case also, we have only to calculate the subresultant up to (1, 1, 1) significant terms, but the calculation is much simpler than that in Example 1 because each coefficient of P_1 and P_2 contains a constant term. We have

$$\begin{aligned} D_1^{(1)} &= 23 - 38y - 149z + (\text{terms of term–degree} \geq 2), \\ D_0^{(1)} &= 23 - 61y - 172z + (\text{terms of term–degree} \geq 2). \end{aligned}$$

After the power-series division of $D_0^{(1)}$ by $D_1^{(1)}$, we have

$$D_0^{(1)} \,/\, D_1^{(1)} = (1 - y - z) + (\text{terms of term–degree} \geq 2).$$

Therefore, we have

$$\tilde{P} = 1 \cdot x + (1 - y - z). \,\square$$

(Note) If we calculate the determinants $D_1^{(1)}$ and $D_0^{(1)}$ fully, we obtain the polynomials of terms 21 and 38, respectively, for P_1 and P_2 in Example 2 and polynomials of terms 45 and 57, respectively, for P_1 and P_2 in Example 3. Therefore, if we use the subresultant GCD algorithm with x as the main variable, we will face a large expression growth in the above examples.

It is important to note that, although Examples 2 and 3 are essentially the same, the computational efficiency is considerably different in Examples 2 and 3. This is due to a simple fact on power series: the more the term-degree is, the more different terms we have. For example, in $Q\{y, z\}$, we have three different terms y^2, yz, z^2 of term-degree = 2, while we have five different terms y^4, y^3z, y^2z^2, yz^3, z^4 of term-degree = 4. Although we calculate power series up to (1, 1, 1) significant terms in both Examples 2 and 3, we must handle more terms in Example 2 than in Example 3. This shows that, although we may choose higher degree terms as necessary terms in calculating GCD, we had better handle lower degree terms. This is the reason why we utilize truncated power series for discarding unnecessary terms. Furthermore, it indicates the importance of preprocessing which generates many constants in the coefficients of P_1 and P_2.

6. PRS method with power series coefficient

As we have seen in 5, when we calculate a small-sized polynomial by applying rational operations to large polynomials, we can often obtain the answer efficiently by treating the polynomials as power series and cutting off the higher degree terms. We can apply this idea to PRS algorithms also.

Let $E, e_y, ..., e_z$ be defined as in Lemma 4. If we calculate the PRS $(P_1, P_2, ..., P_k{\neq}0, P_{k+1}{=}0)$ up to $(E, e_y, ..., e_z)$ significant terms, and calculate $\tilde{P} = gP_k \,/\, \mathrm{lc}(P_k)$ up to $(E, e_y, ..., e_z)$ significant terms, then Lemma 2 means that $\tilde{P}$ is a polynomial such that $\tilde{P} = \gamma G$ where $G = \mathrm{GCD}(P_1, P_2)$ and $\gamma = \mathrm{GCD}(\mathrm{lc}(P_1), \mathrm{lc}(P_2)) \,/\, \mathrm{lc}(G)$. The PRS can be calculated by the conventional formula:

$$\begin{cases} \beta_i P_{i+1} = \alpha_i P_{i-1} - Q_i P_i, \\ \alpha_i = \mathrm{lc}(P_i)^{\delta+1},\ \delta = \deg(P_{i-1}) - \deg(P_i), \end{cases} \tag{18}$$

where "$\approx$" means the equality up to the $(E, e_y, \ldots, e_z)$ significant terms. One may think that, since the higher degree terms are discarded, we may choose $\beta_i = 1$ without causing the expression growth. However, this is not true as we have mentioned in 5.

In the actual computation of PRS, we must be careful in the "accuracy decreasing", as we have mentioned in 4. Let us clarify the meaning of accuracy decreasing for the case of PRS calculation. First, we note that what we need is P_k (the final element of the PRS). In order to preserve the accuracy of P_k, it is enough that some coefficients of P_i, preserve the accuracy for each i=3,...,k, so long as β_i in Eq.(18) is a number. Next, since we calculate $\tilde{P} = P_k \,/\, [\mathrm{lc}(P_k) \,/\, g]$, we require $\mathrm{lc}(P_k)$ to preserve the accuracy. Finally, we must check the termination of PRS. This check is not trivial because there may happen a case that all the significant terms of P_{i+1}, $i<k$, vanish hence $P_{i+1} \approx 0$. We can discriminate this case by the trial-division of P_1 and P_2 by pp(P_i): if $P_i \neq P_k$ then the trial-division fails because $\deg(P_i) > \deg(G)$. Therefore, for the case $\beta_i = a\ number$, i=2,...,k, we say the accuracy is decreased if either of the following is satisfied:

(1) All the coefficients of P_i lose the accuracy for some i, $3 \le i \le k$;

(2) $\mathrm{lc}(P_k)$ loses the accuracy.

When β_i is also a power series calculated from leading coefficients, we say the accuracy is decreased if the following condition is satisfied:

(3) $\mathrm{lc}(P_i)$ loses the accuracy for some i, $3 \le i \le k$.

Note that the check of accuracy decreasing is quite simple when both P_1 and P_2 contain constant terms: we have only to check the existence of constant terms in the coefficients of P_i.

There are many solutions to the "accuracy decreasing problem", and our solution is as follows:

(1) Continue the PRS construction even if the accuracy decreased;

(2) After calculating the PRS, apply the correctness check.

This method will be reasonable because the degree bounds of, Eqs.(12) and (13), are usually over-estimates and we can often calculate the correct GCD even if the accuracy decreases.

Thus, we obtain the following algorithm.

Algorithm 4 (PRS method with power series coefficient).

```
Step 1. $FLAG \leftarrow NO$;
        $g \leftarrow \mathrm{GCD}(\mathrm{lc}(P_1), \mathrm{lc}(P_2))$;
        $E \leftarrow \mathrm{MIN}\{E_i - E_i' + E'' \mid i=1,2\}$;
        $e_v \leftarrow \mathrm{MIN}\{e_{vi} - e_{vi}' + e_v'' \mid i=1,2\}, v \in \{y, ..., z\}$;
        ( $E_i$ etc. are defined in Lemma 3.)
Step 2. Calculate PRS $(P_1, P_2, ..., P_k \neq 0, P_{k+1}=0)$ up to $(E, e_y, ..., e_z)$ significant
        terms. (This means only formally, that is, we regard the lowest degree terms
        as nonvanished even if their coefficients are 0);
        If the accuracy decreases then $FLAG \leftarrow YES$;
Step 3. If $\deg(P_k)=0$ then return 1 else $G \leftarrow \mathrm{pp}(gP_k \,/\, \mathrm{lc}(P_k))$;
        If $G \mid P_1$ and $G \mid P_2$ then return $G$
          else if $FLAG = YES$ then ( $E \leftarrow E+1$; $e_y \leftarrow e_y+1$; $\cdots$; $e_z \leftarrow e_z+1$ )
          else ( $E \leftarrow 2\times E$; $e_y \leftarrow 2\times e_y$; $\cdots$; $e_z \leftarrow 2\times e_z$ );
        $FLAG \leftarrow NO$; go to Step 2. □
```

As we have mentioned in 5, computation will often be made efficient by preprocessing which generates as many constants as possible in the coefficients of P_1 and P_2. The preprocessing may include the replacement $v^k \rightarrow V$ if both P_1 and P_2 are polynomials in v^k. Furthermore, cheap test of relative primality of P_1 and P_2 is also quite useful. The actual implementation of Algorithm 4 (and 3) will include these devices.

Example 4. P_1 and P_2 given in Example 3, that is

$$\begin{aligned} P_1 = {} & (1+y)x^4 + (1+y-y^2-yz)x^3 + (-1-y+z-2y^2-2yz-z^2)x^2 \\ & + (-2-y+3z+y^2+2yz-z^2+y^3+y^2z)x + (-1+y+3z-2yz-3z^2+yz^2+z^3) \end{aligned}$$

$$\begin{aligned} P_2 = {} & (-1+z)x^4 + (-4+y+2z-yz-z^2)x^3 + (-2+4y+3z)x^2 \\ & + (3+y-2z-y^2-yz)x + (2-y-3z-y^2+z^2). \end{aligned}$$

Lemma 4 gives $(E, e_y, e_z) = (1, 2, 1) \to (1, 1, 1)$. Hence, we calculate a PRS by handling only constants, *constant*×*y* terms, and *constant*×*z* terms. We use the formula (18) with $\beta_i = 1$ for simplicity.

$$\begin{aligned} P_3 &= (-1+z)\cdot P_1 - (1+y)\cdot P_2 \\ &\approx (3+2y-z)x^3 + (3-y-5z)x^2 + (-1-3y-3z)x + (-1-2y-z), \end{aligned}$$

$$\begin{aligned} P_4 &= (3+2y-z)^2\cdot P_2 - Q_3(x)\cdot P_3 \\ &\approx (6+10y-17z)x^2 + (15+4y-60z)x + (9-9y-46z), \end{aligned}$$

$$\begin{aligned} P_5 &= (6+10y-17z)^2\cdot P_3 - Q_4(x)\cdot P_4 \\ &\approx 3\{(-69+22y+631z)x + (-69+91y+700z)\}, \end{aligned}$$

$$P_6 = (-69+22y+631z)^2\cdot P_4 - Q_5(x)\cdot P_5/3 \approx 0.$$

Since $g = \mathrm{GCD}(\mathrm{lc}(P_1), \mathrm{lc}(P_2)) = 1$, we calculate $\tilde{P}$ as

$$\tilde{P} = P_5 \,/\, \mathrm{lc}(P_5) \approx 1\cdot x + (1 - y - z).$$

Since there is no accuracy decreasing, we obtain $G = x + (1 - y - z)$. □

(Note) Since degrees of P_1 and P_2 w.r.t. y or z are smaller than $\deg(P_1)$ and $\deg(P_2)$ in the above example, many GCD programs will calculate $\mathrm{GCD}(P_1, P_2)$ by treating y or z as main variable. The above example shows that our algorithm does not cause expression growth even if the PRS becomes a long sequence.

Compared with the conventional PRS algorithms, say the subresultant PRS algorithm, the improvement in the above computation is drastic. Hence, we think our Algorithms 3 and 4 are quite useful. The computing time analysis and comparison with other algorithms are now going.

Appendix A

In [13], a method of reducing the determinants of the form in Eq.(3) was described. Since the paper has not been published, we describe the method below. The determinant $D_i^{(j)}$ in Eq.(3) is of order $m + n - 2j$, and the method reduces it to a determinant of order $m - j$ (note that $m \geq n$).

At the first step of the reduction, the determinant in Eq.(3) is modified as

$$\begin{array}{|ccccccccc|l}
1 & & & & & & & & & \\
 & \cdot & & & & & & & & \} \ \text{m-n rows} \\
 & & 1 & & & & & & & \\
 & & & a_m & a_{m-1} & \cdots & \cdots & a_{2j+2-n} & a_{i+j+1-n} & \\
 & & & & \cdots & \cdots & \cdots & \cdots & \cdots & \} \ \text{n-j rows} \quad \text{(A1)} \\
 & & & & & a_m & \cdots & a_{j+1} & a_i & \\
0 & \cdot & 0 & b_n & b_{n-1} & \cdots & \cdots & b_{2j+2-m} & b_{i+j+1-m} & \\
 & & & & \cdots & \cdots & \cdots & \cdots & \cdots & \} \ \text{m-j rows} \\
 & & & & & b_n & \cdots & b_{j+1} & b_i &
\end{array}$$

At the second step, we move the bottom $m - n$ rows of (A1) to the middle:

$$\begin{array}{|cccccccccccc|l}
1 & & & & & & & & & & & & \\
 & \cdot & & & & & & & & & & & \} \ \text{m-n} \\
 & & 1 & & & & & & & & & & \\
 & & & a_m & \cdots & a_{m+j+1-n} & a_{m+j-n} & \cdots & \cdots & \cdots & a_{2j+2-n} & a_{i+j+1-n} & \\
 & & & & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \} \ \text{n-j} \\
 & & & & & a_m & a_{m-1} & \cdots & \cdots & \cdots & a_{j+1} & a_i & \\
0 & & & & & & b_n & \cdots & \cdots & \cdots & b_{j+2-m+n} & b_{i+1-m+n} & \text{(A2)} \\
 & \cdot & & & & & & \cdots & \cdots & \cdots & \cdots & \cdots & \} \ \text{m-n} \\
 & & 0 & & & & & & b_n & \cdots & b_{j+1} & b_i & \\
 & & & b_n & \cdots & b_{j+1} & b_j & \cdots & \cdots & \cdots & b_{2j+2-m} & b_{i+j+1-m} & \\
 & & & & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \} \ \text{n-j} \\
 & & & & & b_n & b_{n-1} & \cdots & \cdots & \cdots & b_{j+1-m+n} & b_{i-m+n} &
\end{array}$$

Writing this determinant as

$$\begin{vmatrix} M_{11} & M_{21} \\ M_{12} & M_{22} \end{vmatrix},$$

where $M_{ij}, i=1,2,\ j=1,2,$ are $(m-j)\times(m-j)$ square matrices, we see $M_{11}M_{21} = M_{21}M_{11}$. Hence, Schur's theorem leads us to the last step of the reduction:

$$D_i^{(j)} = (-1)^{(m-n)(n-j)} \mid M_{11}M_{22} - M_{21}M_{12} \mid . \tag{A3}$$

This is our reduction formula. Below, we rewrite (A3) slightly. We define matrices $\tilde{M}_{11}, \tilde{M}_{12}, \tilde{M}_{21}, \tilde{M}_{22}$ as follows:

$$\tilde{M}_{11} = \begin{bmatrix} a_m & a_{m-1} & \cdots & a_{m+j+1-n} \\ & \cdots & \cdots & \cdots \\ & & a_m & a_{m-1} \\ & & & a_m \end{bmatrix},$$

$$\tilde{M}_{21} = \begin{bmatrix} b_n & b_{n-1} & \cdots & b_{j+1} \\ & \cdots & \cdots & \cdots \\ & & b_n & b_{n-1} \\ & & & b_n \end{bmatrix},$$

$$\tilde{M}_{12} = \begin{bmatrix} a_{m+j-n} & \cdots & \cdots & a_{2j+2-n} & a_{i+j+1-n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{m-1} & \cdots & \cdots & a_{j+1} & a_i \end{bmatrix},$$

$$\tilde{M}_{21} = \begin{bmatrix} b_j & \cdots & \cdots & b_{2j+2-m} & b_{j+1-m+n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ b_{n-1} & \cdots & \cdots & b_{i+j+1-m} & b_{i-m+n} \end{bmatrix}.$$

The $\tilde{M}_{11}, \tilde{M}_{12}, \tilde{M}_{21}, \tilde{M}_{22}$ are submatrices of $M_{11}, M_{12}, M_{21}, M_{22}$, respectively. Then, the determinant in (A3) can be written as

$$\begin{vmatrix} b_n \cdots & \cdots\cdots & b_{j+2-m+n} & b_{i+1-m+n} \\ \cdots & \cdots\cdots & \cdots & \cdots \\ & b_n \cdots & b_{j+1} & b_i \\ & & & \\ & \tilde{M}_{11}\tilde{M}_{22} - \tilde{M}_{21}\tilde{M}_{12} & & \\ & & & \end{vmatrix}. \tag{A4}$$

References

1. Collins, G.E., "Subresultants and Reduced Polynomial Remainder Sequences," *J. ACM*, vol. 14, p. 128, 1967.

2. Brown, W.S. and Traub, J.F., "On Euclid's Algorithm and the Theory of Subresultants," *J. ACM*, vol. 18, p. 505, 1971.

3. Brown, W.S., "The Subresultant PRS Algorithm," *ACM Trans. Math. Soft.*, vol. 4, p. 237, 1978.

4. Hearn, A.C., "Non-modular Computation of Polynomial GCDs Using Trial Division," in *Lecture Notes in Comp. Sci. (Proc. of EUROSAM'79)*, vol. 72, p. 227, 1979.

5. Brown, W.S., "On Euclid's Algorithm and the Computation of Polynomial Greatest Common Divisor," *J. ACM*, vol. 18, p. 478, 1971.

6. Moses, J. and Yun, D.Y.Y., "The EZ GCD Algorithm," in *Proceedings of ACM'73*, p. 159, 1973.

7. Wang, P., "The EEZ-GCD Algorithm," *SIGSAM Bulletin*, vol. 14, p. 50, 1980.

8. Zippel, R., "Probabilistic Algorithms for Sparse Polynomials," in *Lecture Notes in Comp. Sci. (Proc. of EUROSAM'79)*, vol. 72, p. 216, 1979.

9. Char, B.W., Geddes, K.O., and Gonnet, G.H., "GCDHEU: Heuristic Polynomial GCD Algorithm Based on Integer GCD Computation," in *Lecture Notes in Comp. Sci. (Proc. of EUROSAM'84)*, vol. 174, p. 285, 1984.

10. Gianni, P. and Trager, B., "GCD's and Factoring Multivariate Polynomials Using Gröbner Bases," in *Lecture Notes in Comp. Sci. (Proc. of EUROCAL'85)*, vol. 204, p. 409, 1985.

11. Buchberger, B., "Gröbner Bases: An Algorithmic Method in Polynomial Idel Theory," in *Multidimensional Systems Theory*, ed. Bose, R., Reidel Publishing, 1985. Ch.6.

12. Sasaki, T., "Some Algebraic Algorithms Based on Head Term Elimination over Polynomial Rings," *paper presented at EUROCAL'87*, Leipzig, June 1987.

13. Sasaki, T., "Extended Euclidean Algorithm and Determinants," *preprint of ICPR (unpublished), 24 pages*, 1982.