Journal of Symbolic Computation 44 (2009) 99–110

Contents lists available at ScienceDirect

Journal of Symbolic Computation

journal homepage: www.elsevier.com/locate/jsc

An improved EZ-GCD algorithm for multivariate polynomials

Kuniaki Tsuji [1]

Amakubo 1-13-1 Sanyo Rojyumanroom 412, Tsukuba city, Japan

ARTICLE INFO

Article history:
Received 8 December 2007
Accepted 16 April 2008
Available online 10 July 2008

Keywords:
EZ-GCD
Bad-zero problem

ABSTRACT

The EZ-GCD algorithm often has the bad-zero problem, which has a remarkable influence on polynomials with higher-degree terms. In this paper, by applying special ideals, the EZ-GCD algorithm for sparse polynomials is improved. This improved algorithm greatly reduces computational complexity because of the sparseness of polynomials. The author expects that the use of these ideals will be useful as a resolution for obtaining a GCD of sparse multivariate polynomials with higher-degree terms.

1. Introduction

GCD computation for multivariate polynomials over **Z** is basic to several important algebraic operations. GCD computation has been done using Euclid's algorithm in the early years of algorithm development and many algorithms were built on this foundation; for example, the reduced-PRS algorithm (Collins, 1967, 1966), the subresultant PRS algorithm (Brown and Traub, 1971), etc. However, these algorithms were insufficient for the GCD computation of multivariate polynomials. Therefore, studies have been conducted from different perspectives to obtain more effective algorithms for multivariate polynomials. One of these is the sparse modular algorithm that was introduced by Zippel (1979, 1993), which is a Las Vegas algorithm of GCD computation. Another is the EZ-GCD algorithm (Moses and Yun, 1973; Yun, 1980), which is widely known as a speedy algorithm. However, the EZ-GCD algorithm has a problem that often leads to a very large intermediate expression growth of terms in polynomials, which is called the "bad-zero problem" (Geddes et al., 1992). In particular, the influence of this problem on polynomials with higher-degree terms is remarkable. To avoid this problem, Wang developed the EEZ-GCD algorithm (Wang, 1980). Since, however, many differentiations are performed to compute the correction coefficients (Wang, 1978), large integer

E-mail address: tuzi@lapis.plala.or.jp.
[1] Tel.: +81 029 853 8166.

0747-7171/$ – see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jsc.2008.04.006

coefficients may often occur. In addition, the efficiency of the EEZ-GCD algorithm depends mainly on its coefficient determination algorithm (Wang, 1980). However, it may not always be applied to all cases of GCD computation. Also, another solution called the sparse EZ-GCD algorithm was developed (Kaltofen, 1985), which makes use of the sparse Hensel lifting by using Newton's iteration. This algorithm can speed up Hensel lifting, although it is based on many assumptions. Furthermore, the PC-PRS algorithm (Sasaki and Suzuki, 1992) was suggested as an algorithm that matches the EZ-GCD algorithm in efficiency. However, it also has the bad-zero problem.

To tackle the bad-zero problem, the author attempted to improve the EZ-GCD algorithm using a special ideal as a new modulus. Through the adoption of this ideal, the improved EZ-GCD algorithm can reduce an intermediate expression growth of terms in polynomials, which are sparse, have higher-degree terms with respect to sub-variables and the degree of the main variable is not very high. As a result, the algorithm can perform at high speed for obtaining a GCD of two multivariate polynomials.

There were several investigations of polynomials with higher-degree terms. For example, an algorithm using the Frobenius automorphism was suggested by Kaltofen et al. (Kaltofen and Lobo, 1994), which deals only with univariate polynomials. Also, a fast factorization method using primitive elements in finite fields was developed by Noro et al. (Noro and Yokoyama, 2002), which only refers to bivariate polynomial factorization over finite fields. However, none of the previous methods used a special modulus for polynomials with higher-degree terms. Recently, an algorithm for resolving the bad-zero problem was suggested (Inaba, 2005), which makes use of a Newton polynomial with special moduli and some assumptions.

In Section 2, some definitions and main theorems are described. Two new algorithms with some examples, their study, and experiments comparing several algorithms are provided in Section 3. The conclusions are presented in Section 4.

2. Definitions and theorems

Let x_1 be the main variable, the remaining $x_2, \ldots, x_s$ be the sub-variables, and s be the number of variables. Let $P = \sum_{i=0}^{t} d_i \prod_{j=1}^{s} x_j^{e_{ji}} \in \mathbf{Z}[x_1, \ldots, x_s]$ with $d_i \in \mathbf{Z}$ or $P = \sum_{i=0}^{t} d_i x_1^{e_i} \in R[x_1]$ with $d_i \in R$, where R is a ring and there are no similar terms in P; then, $\mathrm{lc}(P)$ is the leading coefficient w.r.t. x_1 of P, $\mathrm{pp}_{x_1}(P)$ is the primitive part w.r.t. x_1 of P, for $d_i \in \mathbf{Z}$ the height of P is defined as $\max\{|d_t|, \ldots, |d_0|\}$, and $P|_{A_1=B_1,\ldots,A_r=B_r}$ or $P|_{A_1\to B_1,\ldots,A_r\to B_r}$ shows the substitution $A_1 \to B_1, \ldots, A_r \to B_r$ in P.

Definitions of ideals and names of algorithms. The *"power ideal S"* is an ideal of $\mathbf{Z}[x_1, x_2, \ldots, x_s]$ and is expressed as

$$S = ({x_2}^{l_2} - c_2, {x_3}^{l_3} - c_3, \ldots, {x_s}^{l_s} - c_s),$$
$$c_u, l_u \in \mathbf{Z},\ l_u > 1,\ c_u \neq 0,\ \ (2 \leq u \leq s),$$

where S satisfies the following conditions (a), (b):

(a) The algebraic numbers $\alpha_u = {c_u}^{\frac{1}{l_u}}$, $(2 \leq u \leq s)$ are algebraically independent of each other. (According to this condition, $\mathbf{Q}[x_2, \ldots, x_s]/S$ is an algebraic extension field, where $\mathbf{Q}$ is the field of rational numbers. Therefore, $(\mathbf{Q}[x_2, \ldots, x_s]/S)[x_1]$ becomes a UFD, and we can define a GCD of polynomials in it.)

(b) Every $x_i^{l_i} - c_i$, $(2 \leq i \leq s)$ is irreducible over $\mathbf{Z}$.

For any natural number k, S^k is defined as an ideal of $\mathbf{Z}[x_1, x_2, \ldots, x_s]$ generated by all k-generators, where the k-generator is the product of generators that are arbitrarily selected k times from ${x_i}^{l_i} - c_i$, $(2 \leq i \leq s)$ of the power ideal S. For example, suppose $S = ({x_2}^{l_2} - c_2, {x_3}^{l_3} - c_3)$; then, we have $S^2 = (({x_2}^{l_2} - c_2)^2, ({x_3}^{l_3} - c_3)^2, ({x_2}^{l_2} - c_2)({x_3}^{l_3} - c_3))$.

"PI" is an acronym of "power ideal" and the improved EZ-GCD algorithm using a PI is called the *"PI algorithm."*

In addition, the ideal "$\hat{S}$" of $\mathbf{Z}_p[x_1, x_2, \ldots, x_s]$ is defined such that $l_2 = l_3 = \cdots = l_s = p$ in S, where p is a prime number of the finite fields $\mathbf{Z}_p \cong \mathbf{Z}/(p)$, and $\hat{S}$ satisfies condition (c) (it need not satisfy conditions (a) and (b)):

(c) GCD $(c_u, p) = 1$ for all $u = 2, 3, \ldots, s$.

$\hat{S}$ is called the *"power ideal with a prime number p,"* the term *"PIP"* is its abbreviation and the improved EZ-GCD algorithm using a PIP is called the *"PIP algorithm."*

Furthermore, let

$$\bar{S} = (x_2 - c_2, \ldots, x_s - c_s), \quad c_i \in \mathbf{Z}, (2 \le i \le s)$$

be an ideal of $\mathbf{Z}[x_1, x_2, \ldots, x_s]$.

For convenience, let $K = \tilde{K} = \mathbf{Q}[x_2, \ldots, x_s]/S$, $K' = \mathbf{Z}_p[x_2, \ldots, x_s]/\hat{S}$ and $K'' = \mathbf{Z}_p[x_2, \ldots, x_s]/\bar{S}$.

The following theorems support the PI and the PIP algorithms.

Theorem 1. *Let $F_1, F_2 \in K[x_1]$. Assume that F_1 and F_2 do not have a common factor w.r.t. x_1 in $\tilde{K}[x_1]$, and*

$$\mathrm{lc}(F_1 F_2) \not\equiv 0 \pmod{S}. \tag{2.1}$$

Then, the following equation holds:

$$\hat{A}' F_1 + \hat{B}' F_2 \equiv 1 \pmod{S}, \tag{2.2}$$

where $\deg_{x_1}(F_1) > \deg_{x_1}(\hat{B}')$, $\deg_{x_1}(F_2) > \deg_{x_1}(\hat{A}')$

and only one pair of $\hat{A}'$ and $\hat{B}'$ exists in $K[x_1]$.

Proof. Since $K[x_1]$ is a UFD, if we apply the generalized Euclidean algorithm to F_1 and F_2 in $K[x_1]$ and if the polynomials $\hat{A}'$ and $\hat{B}'$ obtained by applying it are normalized to satisfy the degree conditions $\deg_{x_i} \hat{A}' < l_i$ and $\deg_{x_i} \hat{B}' < l_i$, $(2 \le i \le s)$, then we can obtain Eq. (2.2) which becomes unique. □

From this theorem, the following corollary is also derived.

Corollary of Theorem 1. *Under the conditions that Theorem 1 holds, for two multivariate polynomials F_1, F_2 and any multivariate polynomial $H \in \mathbf{Z}[x_1, \ldots, x_s]$, the following equation holds, and only one pair of $\hat{A}$ and $\hat{B}$ exists in $K[x_1]$:*

$$\hat{A} F_1 + \hat{B} F_2 \equiv H \pmod{S}, \text{ where } \deg_{x_1}(F_2) > \deg_{x_1}(\hat{A}). \tag{2.3}$$

Next, for $F_1, F_2 \in K'[x_1]$, assuming that $F_1|_{x_2=c_2,\ldots,x_s=c_s}$ and $F_2|_{x_2=c_2,\ldots,x_s=c_s}$ do not have a common factor w.r.t. x_1 in $K''[x_1]$ instead of $\tilde{K}[x_1]$, replace K by K' in Theorem 1 and its corollary, replace S by $(\bar{S}, p)$ in Eq. (2.1) and S by $(\hat{S}, p)$ in Eqs. (2.2) and (2.3). Then, Theorem 1 and its corollary hold similarly for the PIP algorithm because of the following proof. In the proof of Theorem 1, if we replace K by K'', then $K''[x_1]$ is also a UFD. Furthermore, by applying the generalized Euclidean algorithm to $F_1|_{x_2=c_2,\ldots,x_s=c_s}$ and $F_2|_{x_2=c_2,\ldots,x_s=c_s}$ in $K''[x_1]$, and by using the usual Hensel construction, we can derive $\hat{A}' F_1 + \hat{B}' F_2 \equiv 1 \pmod{((x_2 - c_2)^p, (x_3 - c_3)^p, \ldots, (x_s - c_s)^p, p)}$. Then, from Fermat's little theorem, we have $x_i^p - c_i \equiv (x_i - c_i)^p \pmod{p}$, $(2 \le i \le s)$. Therefore, we can also derive Eqs. (2.2) and (2.3) for the PIP algorithm.

From this corollary and the fact that the set K is a ring, we can also construct Hensel's lemma for the PI algorithm (this Hensel construction using a PI is called the *"Hensel construction modulo PI"*). This lemma is derived from the same proof as the usual Hensel's lemma and is as follows.

Theorem 2 (*Hensel's Lemma Modulo PI*). *Let $F_1, F_2 \in K[x_1]$, $V \in \mathbf{Z}[x_1, x_2, \ldots, x_s]$. For $S = (X_2, \ldots, X_s)$, where $X_i = x_i^{l_i} - c_i$, the following three conditions hold:*

(A)

$$\mathrm{lc}(F_1) \not\equiv 0 \pmod{S} \quad \textit{and} \quad \mathrm{lc}(F_2) \not\equiv 0 \pmod{S}, \tag{2.4}$$

(B) *F_1 and F_2 do not have a common factor w.r.t. x_1 in $\tilde{K}[x_1]$, and*

(C) $\deg_{x_u}(F_1) < l_u$, $\deg_{x_u}(F_2) < l_u$, $(2 \le u \le s)$.

Put $F_1^{(1)} = F_1$ and $F_2^{(1)} = F_2$. Under these conditions, suppose that the following equation holds for the multivariate polynomial V:

$$V \equiv F_1^{(1)} F_2^{(1)} \pmod{S}. \tag{2.5}$$

Then, for any natural number k, there exist polynomials $F_1^{(k)}$ and $F_2^{(k)}$ such that

$$V \equiv F_1^{(k)} F_2^{(k)} \pmod{S^k}, \tag{2.6}$$

$$F_1^{(k)} \equiv F_1^{(1)} \pmod{S} \quad \textit{and} \quad F_2^{(k)} \equiv F_2^{(1)} \pmod{S}, \tag{2.7}$$

$$F_1^{(k)}, F_2^{(k)} \in \mathbf{Q}[x_1, x_2, \ldots, x_s, X_2, \ldots, X_s].$$

In Theorem 2, replace K by K', replace $\mathbf{Q}$ by $\mathbf{Z}_p$, and let $\hat{S} = (X_2, \ldots, X_s)$ as a PIP by setting $l_i = p$ for all $i \geq 2$. Furthermore, assuming that $F_1|_{x_2=c_2,\ldots,x_s=c_s}$ and $F_2|_{x_2=c_2,\ldots,x_s=c_s}$ do not have a common factor w.r.t. x_1 in $K''[x_1]$ instead of $\tilde{K}[x_1]$ in the condition (B), replace S by $(\hat{S}, p)$ in Eqs. (2.5) and (2.7), replace S^k by $(\hat{S}^k, p)$ in Eq. (2.6), and replace S by $(\bar{S}, p)$ in Eq. (2.4). Then, Theorem 2 holds similarly for the PIP algorithm (this Hensel construction using a PIP is called the *"Hensel construction modulo PIP"*).

3. Two new algorithms and their study

The PI and the PIP algorithms are presented in this section. For the GCD computation of multivariate polynomials P_1 and P_2 in $\mathbf{Z}[x_1, x_2, \ldots, x_s]$, we use the following degree notation:

$$n_i = \max\{\deg_{x_i}(gP_1), \deg_{x_i}(gP_2)\}, \quad (1 \leq i \leq s),$$
if $\sum_{j=2}^{s} \deg_{x_j}(gP_1) < \sum_{j=2}^{s} \deg_{x_j}(gP_2)$ then $\tilde{P} = gP_1$ else $\tilde{P} = gP_2$
and $g = \mathrm{GCD}(\mathrm{lc}(P_1), \mathrm{lc}(P_2))$.

3.1. New algorithms

The PI algorithm is as follows:
<**Input**>**:** Multivariate polynomials $P_1, P_2 \in \mathbf{Z}[x_1, x_2, \ldots, x_s]$, which are primitive w.r.t. x_1 (*1) and sparse, where $\deg_{x_1} P_2 \leq \deg_{x_1} P_1$, and g.
<**Output**>**:** $\mathrm{GCD}(P_1, P_2) \in \mathbf{Z}[x_1, x_2, \ldots, x_s]$.

Step 1. As a PI, choose S satisfying the conditions (a) and (b) in Section 2, where $\mathrm{lc}(P_1P_2) \not\equiv 0 \pmod{S}$, $X_i \leftarrow {x_i}^{l_i} - c_i$, and $\tilde{S} \leftarrow S$. ${P_1}'' \leftarrow P_1|_{{x_2}^{l_2}=c_2,\ldots,{x_s}^{l_s}=c_s}$ and ${P_2}'' \leftarrow P_2|_{{x_2}^{l_2}=c_2,\ldots,{x_s}^{l_s}=c_s}$.

Step 2. Compute D the GCD of ${P_1}''$ and ${P_2}''$ modulo $\tilde{S}$ using the generalized Euclidean algorithm, and if d is known, obtain the relations $P_1 \equiv g_1 D \pmod{S}$ and $P_2 \equiv g_2 D \pmod{S}$ in $K[x_1]$ such that $d = \deg_{x_1} D$; else obtain these relations regardless of the value of d and $d \leftarrow \deg_{x_1} D$ as well as the EZ-GCD algorithm (see (Yun, 1980)).
If $d = 0$ then return 1;
When d is not equal to 0, either of three cases (a), (b), (c) occurs.

(a) Else if $\deg_{x_1} P_2 = d$ then if $P_2 | P_1$ then return P_2; else compute again (*3).

(b) Else if D and g_1 (or g_2) do not have a common factor w.r.t. x_1 in $K[x_1]$ (*4), then adjust D, g_1 (or g_2) modulo S like in the EZ-GCD algorithm (see (Yun, 1980)). Do $F_2^{(1)} \leftarrow g_1$, $F_1^{(1)} \leftarrow D$, $L \leftarrow gP_1$, $L' \leftarrow gP_2$ (or $F_2^{(1)} \leftarrow g_2$, $F_1^{(1)} \leftarrow D$, $L \leftarrow gP_2$, $L' \leftarrow gP_1$), $\hat{n}_i \leftarrow [\frac{\deg_{x_i} L}{l_i}]$ (where [ ] denotes Gauss's symbol), $V \leftarrow L|_{{x_2}^{l_2}=X_2+c_2,\ldots,{x_s}^{l_s}=X_s+c_s}$ and $\hat{n} \leftarrow$ the total degree in X_i's of V (*5), and construct $V \equiv F_1^{(1)} F_2^{(1)} \pmod{S}$. In addition, by applying the Corollary of Theorem 1, obtain $\hat{A}_i F_1^{(1)} + \hat{B}_i F_2^{(1)} \equiv {x_1}^i \pmod{S}$, $(0 \leq i \leq \deg_{x_1} V)$, where $\deg_{x_j} \hat{A}_i$, $\deg_{x_j} \hat{B}_i < l_j$, $(2 \leq j \leq s)$.

(c) Otherwise, deal with the common divisor problem (*2).

Step 3. Apply the Hensel lifting of Theorem 2 to $V \equiv F_1^{(1)} F_2^{(1)} \pmod{S}$; then,

$$V \equiv F_1^{(\hat{n}+1)} F_2^{(\hat{n}+1)} \pmod{S^{\hat{n}+1}}.$$

Step 4. ${x_i}^{l_i} - c_i \leftarrow X_i$ for all $i > 1$ in $F_1^{(\hat{n}+1)}, F_2^{(\hat{n}+1)}$.

Step 5. $\tilde{F_1} \leftarrow {F_1}^{(\hat{n}+1)}$. Verify whether $\tilde{F_1}$ is the factor of V over $\mathbf{Z}$ and whether $\tilde{F_1} | L'$ holds. If both hold, then return $\mathrm{pp}_{x_1}(\tilde{F_1})$; else compute again (*3).

The PIP algorithm is as follows:

> For the PI algorithm as described above, let $p \leftarrow l_i$ for all $i > 1$ in Step 2 and Step 4. In Step 1, choose $(\bar{S}, p)$, and define $\hat{S}$ with generators $X_i \leftarrow {x_i}^p - c_i$ as a PIP from $(\bar{S}, p)$, where $\hat{S}$ satisfies the condition (c) in Section 2 and $\mathrm{lc}(P_1 P_2) \not\equiv 0 \pmod{(\bar{S}, p)}$. In addition, let $\tilde{S} \leftarrow (\bar{S}, p)$, ${P_1}'' \leftarrow P_1|_{x_2=c_2,\ldots,x_s=c_s}$ and ${P_2}'' \leftarrow P_2|_{x_2=c_2,\ldots,x_s=c_s}$ in Step 1. Replace K by K', $\tilde{S}$ by $(\bar{S}, p)$ and S by $(\hat{S}, p)$ in Step 2 (see Remark 2). Replace S by $(\hat{S}, p)$, and $S^{\hat{n}+1}$ by $(\hat{S}^{\hat{n}+1}, p)$ in Step 3. After the transformation in Step 4, if necessary, apply the coefficient determination algorithm (Wang, 1980) or apply p-adic Hensel lifting, then we obtain $V \equiv {F_1^{(\hat{n}+1)}}' {F_2^{(\hat{n}+1)}}' \pmod{(\hat{S}^{\hat{n}+1}, p^{\zeta})}$, $1 < \zeta \in \mathbf{Z}$, and moreover, replace $F_1^{(\hat{n}+1)}$ by ${F_1^{(\hat{n}+1)}}'$ in Step 5.

Remark 1. In Step 1, for each c_i in S or $\bar{S}$, choose a value such that its absolute value becomes small (± 1 is a desirable value because its use does not lead to coefficient increasing). Furthermore, for each l_i in S, choose a relatively small number such that $l_i < \sqrt{{n_i}'}$, ${n_i}' = \deg_{x_i} \tilde{P}$, $i = 2, \ldots, s$, where $\prod_{i=2}^{s} l_i$ is somewhat larger than 2^{s-1}. For p in $\hat{S}$, choose a relatively small prime number such that $2 < p \leq$ a value around $\min\{\sqrt{{n_2}'}, \sqrt{{n_3}'}, \ldots, \sqrt{{n_s}'}\}$. (These results were obtained from some experiments.)

Remark 2. In Step 2, after the relation $P_1 \equiv g_1 D \pmod{(\bar{S}, p)}$ (or $P_2 \equiv g_2 D \pmod{(\bar{S}, p)}$) is obtained using the generalized Euclidean algorithm when we apply the PIP algorithm, $P_1 \equiv g_1 D \pmod{(\hat{S}, p)}$ (or $P_2 \equiv g_2 D \pmod{(\hat{S}, p)}$) is derived from its relation with the aid of the usual Hensel lifting as described in the proof of Theorem 1. Similarly, we can obtain $\hat{A}_i F_1^{(1)} + \hat{B}_i F_2^{(1)} \equiv {x_1}^i \pmod{(\hat{S}, p)}$ in Step 2(b).

Remark 3. In Step 3, immediately before we perform Hensel lifting, we always apply the coefficient determination algorithm (Wang, 1980) to $V \equiv F_1^{(1)} F_2^{(1)} \pmod{S}$ for the PI algorithm, and apply it to $V \equiv F_1^{(1)} F_2^{(1)} \pmod{(\hat{S}, p)}$ for the PIP algorithm. If GCD (P_1, P_2) is obtained, this algorithm terminates.

Remark 4. In Step 3, we can apply the technique of variable-by-variable approach (Wang, 1978) to Hensel lifting of the PI (or PIP) algorithm as well as the EEZ-GCD algorithm. Then, its lifting is performed on only one of the X_i's ($2 \leq i \leq s$) and is repeated from X_2 to X_s. (Note that this method is different from the Hensel construction of Theorem 2 using the total degree in the X_i's.) Moreover, immediately before the Hensel lifting of each variable X_i is carried out, the coefficient determination algorithm is always carried out and it works especially efficiently. This method is called the *"PI (or PIP)(∗) algorithm."*

Remark 5. The PI algorithm has the following characteristic. When performing Hensel lifting in Step 3, if either of two factors $F_1^{(k)}$ and $F_2^{(k)}$ ($1 \leq k \leq \hat{n} + 1$) becomes a polynomial with a denominator (for example, ${x_1}^2 + \frac{3}{2} x_1 x_2 X_2$) during computing, we cannot continuously perform Hensel lifting, and the procedure for obtaining a GCD is interrupted. This greatly avoids wasteful computation.

Some other remarks are as follows.
(*1) If P_1 (or P_2) is a non-monic polynomial, we must remove its content w.r.t. x_1 from it in advance.
(*2) The common divisor problem often occurs when both P_1 and P_2 are not square-free. To resolve the situation, a few strategies have been suggested by Yun (Yun, 1980), by Wang and by Spear (Wang, 1980). These strategies can also be adopted for the PI and the PIP algorithms.
(*3) Set $d \leftarrow d - 1$. If $d = 0$ then return 1. Otherwise, return to Step 1. This feedback is similar to that of the EZ-GCD algorithm (Yun, 1980).

(*4) When D and g_1 do not have a common factor, and moreover, D and g_2 do not have a common factor, if $\sum_{j=2}^{s} \deg_{x_j}(gP_1) < \sum_{j=2}^{s} \deg_{x_j}(gP_2)$ then choose D, g_1; else choose D, g_2.

(*5) Let $V = \sum_{i=0}^{t'} f_i \prod_{j=2}^{s} X_j^{e_{ji}'}$, $e_{ji}' \geq 0$, $f_i \in \mathbf{Z}[x_1, x_2, \ldots, x_s]$, where there are no similar terms in V; then $\hat{n} \leftarrow \max\{e_{20}' + e_{30}' + \cdots + e_{s0}', e_{21}' + e_{31}' + \cdots + e_{s1}', \ldots, e_{2t'}' + e_{3t'}' + \cdots + e_{st'}'\}$. For example, for $V = {x_1}^3 + x_2 x_3 {X_2}^{10} {X_3}^5 {x_1}^2 + (x_2 x_3 {X_2}^{21} {X_3}^2 + x_2^2 x_3^2 {X_2}^{19} {X_3}^6) x_1 + x_2^{13} x_3^9 {X_2}^4 {X_3}^{20} - 8 x_2^3 x_3^3$, $\hat{n}$ is $25 (= 19 + 6)$.

Some examples are shown below (for the sake of simplicity, the degree of each sub-variable in polynomials in the examples is relatively low, and the values of l_u, c_u $(u = 2, 3)$ and p in S or $\hat{S}$ are very small).

Example 1. We compute a GCD of the following P_1 and P_2. Let $X_2 = Y, X_3 = Z, x_1 = x, x_2 = y, x_3 = z$,

$$\begin{aligned} P_1(x,y,z) &= x^5 + (y^2z^3 + y^3z^3)x^4 + (y^4z^2 + y^4z^5)x^3 \\ &\quad + (2y^3z^4 + y^6z^5 + y^7z^5)x^2 + (2y^5z^7 + 2y^6z^7 + y^8z^7)x + 2y^7z^9 \\ &= (x^2 + (y^2z^3 + y^3z^3)x + y^4z^5)(x^3 + y^4z^2x + 2y^3z^4), \\ P_2(x,y,z) &= x^4 + (y^2z^3 + 2y^3z^3)x^3 + (y^5z + y^4z^5 + y^5z^6 + y^6z^6)x^2 \\ &\quad + (y^7z^4 + y^8z^4 + y^7z^8)x + y^9z^6 \\ &= (x^2 + (y^2z^3 + y^3z^3)x + y^4z^5)(x^2 + y^3z^3x + y^5z). \end{aligned}$$

As a PI, let $S = (y^3 - 2, z^2 + 1)$, and apply the Euclidean algorithm modulo S to P_1 and P_2.
Then the common factor D of P_1 and P_2 in $K[x]$ is as follows:

$$P_1 \equiv (x^3 - 2yx + 4)D \pmod{S},$$
$$P_2 \equiv (x^2 - 2zx + 2y^2z)D \pmod{S}, \quad \text{and} \quad D \equiv x^2 + (-2z - y^2z)x + 2yz \pmod{S}.$$

Secondly, put $F_1^{(1)} = x^2 + (-2z - y^2z)x + 2yz$, $F_2^{(1)} = x^2 - 2zx + 2y^2z$ and $L = P_2$.
Furthermore, put $Y = y^3 - 2$ and $Z = z^2 + 1$.
V is rewritten as follows:

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

Therefore, $V \equiv F_1^{(1)} F_2^{(1)} \pmod{(Y, Z)}$.
Applying the Hensel construction modulo PI to this relation,

$$\begin{aligned} V - F_1^{(1)}F_2^{(1)} &\equiv (-2zY + 4zZ + y^2zZ)x^3 + (-4Y - y^2Y + yzY + y^2zY + 8Z + 4y^2Z - 4yzZ)x^2 \\ &\quad + (6yY + 4y^2Y - 16yZ - 4y^2Z)x - 8Y + 16Z \pmod{(Y^2, YZ, Z^2)}, \end{aligned}$$
$$F_1^{(2)} \equiv x^2 + (-2z - y^2z - Yz + 2zZ + y^2zZ)x + 2yz + yzY - 4yzZ \pmod{(Y^2, YZ, Z^2)} \quad \text{and}$$
$$F_2^{(2)} \equiv x^2 + (-2z - Yz + 2zZ)x + 2y^2z + y^2zY \pmod{(Y^2, YZ, Z^2)}.$$

Repeating a similar process, we finally obtain

$$F_1^{(4)} \equiv x^2 + (y^2z^3 + y^3z^3)x + y^4z^5 \pmod{(Y^4, Y^3Z, Y^2Z^2, YZ^3, Z^4)},$$
$$F_2^{(4)} \equiv x^2 + y^3z^3x + y^5z \pmod{(Y^4, Y^3Z, Y^2Z^2, YZ^3, Z^4)}.$$

Put $\tilde{F_1} = x^2 + (y^2z^3 + y^3z^3)x + y^4z^5$ and $\tilde{F_2} = x^2 + y^3z^3x + y^5z$; then $V = \tilde{F_1}\tilde{F_2}$. Verifying whether $P_1 (= L')$ is exactly divisible by $\tilde{F_1}$, we have $\tilde{F_1} | P_1$. Thus, we can obtain $\tilde{F_1}$ as the true GCD of P_1 and P_2.

Example 2. We compute a GCD of the following P_1 and P_2. Let $X_2 = Y, X_3 = Z, x_1 = x, x_2 = y, x_3 = z$,

$$\begin{aligned}
P_1 &= y^4z^2x^4 + (y^2z^2 + y^3z^2 + 2y^2z^4 + y^3z^4 + y^4z^4)x^3 + (2y^2z + y^3z + 2z^4 \\
&\quad + 3yz^4 + 2y^2z^4 + y^3z^4)x^2 + (2z + 2yz + 2yz^3 + y^2z^3 + y^3z^3)x + 2y \\
&= (zy^2x^2 + z^3(y^2 + y + 2)x + 2)(zy^2x^2 + z(y + 1)x + y), \\
P_2 &= y^2zx^5 + (2z^3 + yz^3 + y^2z^3)x^4 + (2 + 2y^7z^6)x^3 + (4y^5z^5 + 4y^5z^8 + 2y^6z^8 \\
&\quad + 2y^7z^8)x^2 + (4y^5z^5 + 8y^3z^7 + 4y^4z^7 + 4y^5z^7)x + 8y^3z^4 \\
&= (zy^2x^2 + z^3(y^2 + y + 2)x + 2)(x^3 + 2z^5y^5x + 4z^4y^3).
\end{aligned}$$

Let $\bar{S} = (y + 1, z + 1)$, and $p = 3$. Then, we have $\hat{S} = (Y, Z)$ as a PIP, $Y = y^3 + 1$ and $Z = z^3 + 1$. Substituting $y = -1$ and $z = -1$ in P_1 and P_2, and applying the Euclidean algorithm modulo $(\bar{S}, p)$ to these polynomials, we obtain

$$\begin{aligned}
&D \equiv \mathrm{GCD}(\mathrm{P}_1, \mathrm{P}_2) \equiv \mathrm{x}^2 + 2\mathrm{x} + 1 \pmod{(\bar{S}, 3)}, \\
&P_1 \equiv (x^2 + 2x + 1)(x^2 + 1) \equiv Dg_1 \pmod{(\bar{S}, 3)}.
\end{aligned}$$

Considering $g = \mathrm{GCD}(\mathrm{lc}(P_1), \mathrm{lc}(P_2)) = y^2z$, and adjusting the leading coefficient of P_1, we have $gP_1 \equiv gg_1D \pmod{(\bar{S}, 3)}$.

Thus, we have

$$\begin{aligned}
&\bar{F}_1^{(1)} = gD \equiv g(x^2 + 2x + 1) \equiv y^2zx^2 + x + 2 \pmod{(\bar{S}, 3)} \quad \text{and} \\
&\bar{F}_2^{(1)} = \mathrm{lc}(P_1)g_1 \equiv \mathrm{lc}(P_1)(x^2 + 1) \equiv \mathrm{y}^4\mathrm{z}^2\mathrm{x}^2 + 1 \pmod{(\bar{S}, 3)}.
\end{aligned}$$

Applying the usual Hensel lifting to these equations, we have

$$\begin{aligned}
&gP_1 = V \equiv F_1^{(1)}F_2^{(1)} \pmod{(\hat{S}, 3)}, \\
&\text{where } F_1^{(1)} \equiv y^2zx^2 + (2y^2 + 2y + 1)x + 2 \pmod{(\hat{S}, 3)} \quad \text{and} \\
&F_2^{(1)} \equiv y^4z^2x^2 + (2z^2 + y^2z^2)x + 2z \pmod{(\hat{S}, 3)}.
\end{aligned}$$

Applying the Hensel construction modulo PIP to these equations as well as Example 1, we have

$$\begin{aligned}
&F_1^{(2)} \equiv y^2zx^2 + (2z^3 + yz^3 + y^2z^3)x + 2 (= \tilde{F_1}) \pmod{(Y^2, YZ, Z^2, 3)} \quad \text{and} \\
&F_2^{(2)} \equiv y^4z^2x^2 + (y^2z^2 + y^3z^2)x + y^3z (= \tilde{F_2}) \pmod{(Y^2, YZ, Z^2, 3)}.
\end{aligned}$$

Since $gP_2 (= L')$ is exactly divisible by $\tilde{F_1}$ and $\mathrm{pp}_x(\tilde{F_1}) = \tilde{F_1}$, $\tilde{F_1}$ is the true GCD of P_1 and P_2.

3.2. Analysis of computational complexity

We compare the computational complexity of the PI algorithm (or PIP algorithm) with that of the EZ-GCD algorithm in cases where the bad-zero problem occurs when a GCD($\neq 1$) of sparse polynomials is obtained using the EZ-GCD algorithm. For the sake of simplicity, assume that the number of variables in polynomials is 3, i.e., x_1, x_2, x_3, $L = gP_1$, $\hat{n}$ is equal to $\sum_{i=2}^{3} \hat{n}_i$ in Step 2 and $\tilde{n} = \sum_{j=2}^{3} \deg_{x_j}(gP_1)$ becomes the (usual) total degree in x_2 and x_3 of gP_1. Computational complexity is mainly estimated for multiplication/division of their terms.

The computational complexities of Steps 2 and 3 in the PI algorithm (or PIP algorithm) are very different from those in the EZ-GCD algorithm. In Step 2, the computational complexity of the generalized Euclidean algorithm (**1) for P_1 and P_2 of the PI algorithm becomes $O(n_1(n_1l_2l_3)^2)$ (set $p = l_2 = l_3$ for that of the PIP algorithm), while that in the EZ-GCD algorithm becomes $O({n_1}^3)$. For Step 3 in the PI algorithm, its computational complexity becomes at most $O(\rho^2{n_1}^2\hat{n}^4)$, $\rho \ll l_2l_3$ (set $p = l_2 = l_3$ for the computational complexity of the PIP algorithm); however, for the EZ-GCD algorithm, the corresponding computational complexity becomes at most $O({n_1}^2\tilde{n}^4)$; therefore,

$$O(\rho^2{n_1}^2\hat{n}^4) < O({n_1}^2\tilde{n}^4). \tag{3.1}$$

The relation (3.1) of computational complexity of Hensel lifting depends upon the difference between linear transformations for an input polynomial P_1 in both algorithms. That is, from the sparseness of an input polynomial, its factors also become sparse in many cases (**2), and so the resulting input polynomial and its factors for Hensel lifting from the linear transformation of sub-variables in the PI (or PIP) algorithm do not become more dense than those in the EZ-GCD algorithm. For example, consider $\tilde{F}_1 = {x_1}^2 + {x_2}^{10}{x_3}^{10}x_1 + x_2x_3 \in \mathbf{Z}[x_1, x_2, x_3]$ as a factor. When the transformation of the EZ-GCD algorithm is applied, we must perform Hensel lifting for obtaining ${x_1}^2 + (x_2 + c_2)^{10}(x_3 + c_3)^{10}x_1 + (x_2 + c_2)(x_3 + c_3)$, $c_2 \neq 0$, $c_3 \neq 0$ and the number of terms in the factor increases from 3 to 126. On the other hand, when that of the PI algorithm is applied, we must perform Hensel lifting for obtaining ${x_1}^2 + x_2(X_2 + c_2)^3(X_3 + c_3)^2x_1 + x_2x_3$, where $X_2 = {x_2}^3 - c_2$ and $X_3 = {x_3}^5 - c_3$, which has only 14 terms. Therefore, the computational complexity of Hensel lifting in the PI (or PIP) algorithm is less than that in the EZ-GCD algorithm in general.

In addition, the computational complexity of some other steps of the PI (or PIP) algorithm is less than that of Hensel lifting of the EZ-GCD algorithm, except for a few cases of the PIP algorithm (**3).

Without loss of generality, we can also extend the above-mentioned description to polynomials such that the number of their variables is not equal to 3 (**4). Consequently, from the point of view of complexity the PI and the PIP algorithms are superior to the EZ-GCD algorithm in general.

Remarks.

(**1) For an input polynomial of the generalized Euclidean algorithm that is performed in the PI algorithm, its height becomes smaller than that in the EZ-GCD algorithm. (This claim also holds for the PIP algorithm because the generalized Euclidean algorithm performs over $\mathbf{Z}_p$.) For example, consider the input polynomial ${x_1}^2 + {x_2}^{10}x_1 + {x_2}^5x_3^2$. When we substitute $c_2 = 2$ for x_2 in the highest term ${x_2}^{10}x_1$ of the polynomial by using the EZ-GCD algorithm, we have $2^{10}x_1$ and its height becomes 1024 (where $c_3 = 1$). On the other hand, when we substitute ${x_2}^3 = 2$ in the term ${x_2}^{10}x_1$ by using the ideal $S = (X_2, X_3), X_2 = {x_2}^3 - 2, X_3 = {x_3}^3 - 3$ of the PI algorithm, we have $x_2 2^3 x_1$ and its height becomes 8.

(**2) Let $I = u_{m1}x_1^{m1} + \cdots + u_0 \in \mathbf{Z}[x_1, x_2, \ldots, x_s]$ and $J = v_{m2}x_1^{m2} + \cdots + v_0 \in \mathbf{Z}[x_1, x_2, \ldots, x_s]$. Then, $V = IJ = u_{m1}v_{m2}{x_1}^{m1+m2} + \cdots + u_0v_0$, $u_{i1}v_{j1} \in \mathbf{Z}[x_2, \ldots, x_s]$, $(0 \leq i1 \leq m1)$, $(0 \leq j1 \leq m2)$. If V is sparse and I is dense, some sums of terms in IJ must vanish and its probability is not high. For example, when $IJ = u_{m1}v_{m2}{x_1}^{m1+m2} + u_0v_0$ holds, we have $u_{m1}v_{m2-1} + u_{m1-1}v_{m2} = 0, \ldots, u_1v_0 + v_1u_0 = 0$, and this phenomenon rarely occurs.

(**3) In some cases, when p-adic Hensel lifting is carried out after the transformation in Step 4, its computational complexity may become large and have a bad influence in efficiency.

(**4) When there are several variables in input polynomials and the PI (or PI(*)) algorithm is forced to choose many $l_j > 1$ in a PI, the computational complexity of the generalized Euclidean algorithm in Step 2 becomes $O(n_1(n_1\prod_{j=2}^{s} l_j)^2)$. Its complexity is exponential in the number of variables s and the rational coefficients that appear in its computation also will be exponential in size in s (coefficients often become dense polynomials in $\hat{A}_i$ and $\hat{B}_i$). Therefore, in cases where there are a few variables in polynomials, we choose each l_j, $(2 \leq j \leq s)$ such that l_j becomes somewhat smaller in advance. (Since the number of iterations of Hensel lifting increases in Step 3 and the computational complexity of Step 4 becomes large when $\prod_{j=2}^{s} l_j$ is very small, it is desirable that $\prod_{j=2}^{s} l_j$ is somewhat larger than 2^{s-1}.) However, when there are many variables in polynomials, the PI (or PI(*)) algorithm will become hopeless in practice, e.g. 20 variables. As a solution for it, if possible, it is desirable that we choose each l_j such that the number of variables in ${P_1}''$ and ${P_2}''$ becomes less than that in P_1 and P_2. For example, let $P_1 = u'_m{x_1}^m + u'_1{x_2}^{\mu_{21}}{x_3}^{\mu_{31}}{x_4}^{\mu_{41}}{x_5}^{\mu_{51}}{x_6}^{\mu_{61}}{x_7}^{\mu_{71}}x_1 + u'_0{x_2}^{\mu_{20}}{x_3}^{\mu_{30}}{x_4}^{\mu_{40}}{x_5}^{\mu_{50}}{x_6}^{\mu_{60}}{x_7}^{\mu_{70}}$, where $l_j = \frac{\mu_{j1}}{\tau_{j1}} = \frac{\mu_{j0}}{\tau_{j0}}$, $(2 \leq j \leq 7)$, $u'_m, u'_0, u'_1 \in \mathbf{Z}$, then we have ${P_1}'' = u'_m{x_1}^m + u'_1{c_2}^{\tau_{21}}{c_3}^{\tau_{31}}{c_4}^{\tau_{41}}{c_5}^{\tau_{51}}{c_6}^{\tau_{61}}{c_7}^{\tau_{71}}x_1 + u'_0{c_2}^{\tau_{20}}{c_3}^{\tau_{30}}{c_4}^{\tau_{40}}{c_5}^{\tau_{50}}{c_6}^{\tau_{60}}{c_7}^{\tau_{70}}$.

3.3. Computer experiments and some results

The benchmarks of the following twelve GCD computation examples were carried out using MATHEMATICA 4.2 on a personal computer running Windows Me. Polynomials P_1 and P_2 in every

example cause the bad-zero problem when we compute their GCD using the EZ-GCD algorithm. In addition, in these examples, the degree of the main variable x is not very high, while the degrees of sub-variables are very high.

Twelve examples

(1) $D' = x^2 + 2y^{23}z^{24}$, $P_1 = D'(x^2 + y^{23}z^{22})$, $P_2 = D'(yzx^3 + 2z^{35}y^{41}x^2 + z^3y^5x + 525)$.

(2) $D' = x^3 + 2(y^{34} + y^{75})x^2 + y^{84}$, $P_1 = D'(x^4 + y^{43}x^3 + yx + y^{92})$, $P_2 = D'(yx^8 + 2y^{47}x^6 + (y^{20} + y^{14})x^4 + y^{69}x^3 + y^{55}x^2 + y^{99} + y^{45}x + 54)$.

(3) $D' = x^2 + 2y^{54}z^{54}$, $P_1 = D'(x^2 + y^{53}z^{62})$, $P_2 = D'(yx^3 + 10y^{61}z^{55}x^2 + 7y^{74}z^{35}x + 2z^{75}y^{51})$.

(4) $D' = (y^7 + 2y)x^3 + 2(3y^{170} + y^{110})x + 4$, $P_1 = D'((y^2 + 2y)x^3 + 3y^{80}x + y^{10})$, $P_2 = D'(x^4 + y^{120}x^3 + 10y^{190}x^2 + 6)$.

(5) $D' = (y^{17} + 2y)x^2 + 2z^{34}x + 4y^{81}z^{14}$, $P_1 = D'(x^2 + 3y^{78}z^4x + y^{69})$, $P_2 = D'(zx^3 + y^{120}z^{45}x + 10y^{51}z^7 + y)$.

(6) $D' = (x + 2y^{80} + 2)(x + 2)(x^2 + 2y^{40} + 1)$, $P_1 = D'(x + 5y^{20} + 2)(x + 3)(x^2 + 3y^{20} + 1)$, $P_2 = D'(x^4 + 5y^{51}x^3 + y^{10}x^2 + 10y^{121})(yx + 2)$.

(7) $D' = y^9z^8x^2 + 2y^{103}z^{35}x + 4y^{141}z^{15} + 2$, $P_1 = D'(x^2 + 3y^{128}z^{54}x + y^{69}z^{105})$, $P_2 = D'(x^3 + y^{220}z^{145}x + 10y^{51}z^7)$.

(8) $D' = (x + 2y^{79}z^{56})(zx^2 + 2y^{55})$, $P_1 = D'(x + 5y^{71}z^2)(x^2 + 3yz^{53})$, $P_2 = D'(yzx^4 + 5y^{81}x^3 + 2 + y^{1000}x^2 + 10y^{1001} + 1)$.

(9) $D' = (x^2 + y^{40}x + y^{80})(x^2 - y^{10}x + y^5)$, $P_1 = D'(x^2 + 6y^7x + 2y^4)(x - y^{40})$, $P_2 = D'(x^3 + 3y^{10}x^2 + 2y^{701} + y)(x + y^{120})$.

(10) $a = y^4z^4$, $b = -y^4$, $D' = (x^2 + a^5x + a^{10})(x - a^5)(x - a^2)$, $P_1 = D'(x^2 + b^5x + b^{10})(x - b^5)(x + b^2)$, $P_2 = D'(x^3 + 3a^{20}x^2 + 2a^{11}x + a)(zx + 3y^{215})$.

(11) $D' = zy^3x^2 + z^{31}(y^3 + y^{51} + 2)x + 2$, $P_1 = D'(zy^3x^2 + z^{54}(y^{51} + 1)x + y^{51})$, $P_2 = D'(yzx^3 + 2z^{55}y^5x^2 - z^{51235}y^5x + 4y^{301} + 2z)$.

(12) $D' = x^3 + y^{52}x^2 + y^{52} + z^{12}$, $P_1 = D'(x^3 + 3z^{100}y^{52}x^2 + z^{40}y^{52}(z - 2) + z^{34})$, $P_2 = D'(zyx^3 + 2z^{51235}y^5x^2 + 4z^4y^1 + y + z)$,

where $D' = \mathrm{GCD}\,(P_1, P_2)$.

For these examples, the following tables show the comparison results among some algorithms, i.e., the PI algorithm, the PIP algorithm, the EZ-GCD algorithm and the EEZ-GCD algorithm. The item "Example (ν-var)" indicates the GCD computation results for ν-variable polynomials.

Table 1
Comparison of speed among some algorithms

Example	EZ-GCD	PI	PI(*)	PIP	PIP(*)	EEZ-GCD
(2-var)						
(2)	173.4	5.44	3.30	11.87	13.18	274.90
	(1, ∗, 7)	(−1, ∗, 4)	(−1, ∗, 4)	(1, ∗, 5)	(1, ∗, 5)	(5)
(4)	345.15	41.69	36.09	35.10	39.49	466.04
	(1, ∗, 7)	(−2, ∗, 7)	(−2, ∗, 7)	(4, ∗, 13)	(4, ∗, 13)	(3)
(6)	450.0	6.92	3.46	55.97	68.98	717.0
	(1, ∗, 7)	(2, ∗, 5)	(2, ∗, 5)	(2, ∗, 11)	(2, ∗, 11)	(4)
(9)	114.03	14.72	10.16	27.35	32.46	285.88
	(−1, ∗, 7)	(3, ∗, 3)	(3, ∗, 3)	(3, ∗, 7)	(3, ∗, 7)	(2)
(3-var)						
(1)	643.56	3.68	1.70	5.71	5.71	29.99
	(1, 1, 3)	(2, 3, 3, 3)	(2, 3, 3, 3)	(2, −1, 7)	(2, −1, 7)	(2, 7)
(3)	–	14.77	1.10	12.36	10.00	14.83
	(1, 1, 11)	(−1, 3, 4, 6)	(−1, 3, 4, 6)	(1, −1, 7)	(1, −1, 7)	(2, 7)

(continued on next page)

Table 1 *(continued)*

Example	EZ-GCD	PI	PI(*)	PIP	PIP(*)	EEZ-GCD
(3-var)						
(5)	–	32.52	13.29	77.56	111.94	180.49
	(3, 1, 7)	(−1, 2, 4, 5)	(−1, 2, 4, 5)	(−1, 1, 5)	(−1, 1, 5)	(3, 7)
(7)	–	1140.53	126.71	953.56	62.07	856.72
	(1, 1, 7)	(2, −1, 5, 4)	(2, −1, 5, 4)	(−1, 1, 5)	(−1, 1, 5)	(3, 5)
(8)	–	396.95	172.90	71.18	47.95	589.68
	(−1, 1, 7)	(−1, 3, 4, 6)	(−1, 3, 4, 6)	(−1, 1, 5)	(−1, 1, 5)	(2, 3)
(10)	–	200.37	3.85	167.03	56.63	120.34
	(1, 1, 7)	(−1, 3, 4, 2)	(−1, 3, 4, 2)	(−1, −1, 5)	(−1, −1, 5)	(2, 3)
(11)	–	453.25	363.89	60.75	62.89	1347.87
	(1, 1, 7)	(3, −1, 5, 4)	(3, −1, 5, 4)	(−1, −1, 5)	(−1, −1, 5)	(2, 3)
(12)	–	383.05	107.65	181.92	246.56	3353.31
	(1, 1, 7)	(2, −1, 5, 4)	(2, −1, 5, 4)	(−1, −1, 7)	(−1, −1, 7)	(2, 3)

Times listed in Table 1 are in seconds. The symbol "−" indicates that it took more than 1 hour or the benchmark program that is based on each algorithm ran out of computer memory. The modulus condition arrays (,,,) in this table were given appropriately by the author and their meaning is as follows:

For the EZ-GCD algorithm,

$(a, b, c) \Longrightarrow y - a, z - b, \bmod c$,

$(a, *, c) \Longrightarrow y - a, \bmod c$.

For example, for (3, 2, 7), the modulus ideal is $y - 3, z - 2, \bmod 7$.

For the PI (or PI(*)) algorithm,

$(a, b, c, d) \Longrightarrow y^c - a, z^d - b$,

$(a, *, c) \Longrightarrow y^c - a$.

For example, for $(3, *, 2)$, the modulus ideal is $y^2 - 3$.

For the PIP (or PIP(*)) algorithm,

$(a, b, c) \Longrightarrow y^c - a, z^c - b, \bmod c$,

$(a, *, c) \Longrightarrow y^c - a, \bmod c$.

For example, for (3, 2, 7), the modulus ideal is $y^7 - 3, z^7 - 2, \bmod 7$.

For the EEZ-GCD algorithm,

$(a, b) \Longrightarrow y - a, z - b$,

$(a) \Longrightarrow y - a$.

For example, for (3, 2), the modulus ideal is $y - 3, z - 2$.

From Table 1, we can infer that the PI (or PIP) algorithm is much faster than the EZ-GCD algorithm. Furthermore, in many cases, the time for the PI (or PIP) algorithm is shorter than or almost equal to that for the EEZ-GCD algorithm. However, there are some exceptional cases where the time for the former is longer than that for the latter. For example, this occurs in No. (7) in Table 1. The author thinks that this is an effect of the use of the coefficient determination algorithm in the latter. Therefore, in such cases, if we instead apply the PI(*) algorithm (or the PIP(*) algorithm) that can perform the coefficient determination algorithm (Wang, 1980) for each X_i, $(2 \leq i \leq s)$, its time becomes shorter than that for the latter.

Table 2
Comparison of the maximum number of terms among some algorithms

Example	EZ-GCD	PI	PI(*)	PIP	PIP(*)	EEZ-GCD
(2-var)						
(2)	[840, 162, 140]	[299, 51, 37]	[299, 51, 37]	[135, 17, 26]	[135, 17, 26]	[717, 151, 95]
(4)	[686, 180, 116]	[399, 45, 35]	[399, 45, 35]	[161, 14, 21]	[161, 14, 21]	[520, 173, 83]
(6)	[1069, 405, 125]	[221, 85, 29]	[221, 85, 29]	[295, 71, 19]	[295, 71, 19]	[889, 321, 85]
(9)	[685, 299, 135]	[329, 102, 48]	[329, 102, 48]	[165, 53, 24]	[165, 53, 24]	[657, 243, 121]
(3-var)						
(1)	[2007, 577, 529]	[393, 73, 65]	[393, 10, 9]	[82, 17, 17]	[82, 5, 5]	[69, 25, 24]
(3)	[−, −, −]	[835, 141, 155]	[56, 15, 15]	[67, 5, 9]	[17, 3, 3]	[159, 55, 54]
(5)	[−, −, −]	[826, 75, 51]	[296, 27, 51]	[607, 30, 53]	[265, 11, 53]	[586, 84, 139]
(7)	[−, −, −]	[7678, 312, 918]	[748, 57, 50]	[1740, 64, 115]	[225, 15, 11]	[745, 207, 139]

Table 2 *(continued)*

Example	EZ-GCD	PI	PI(*)	PIP	PIP(*)	EEZ-GCD
(3-var)						
(8)	[−, −, −]	[4408, 555, 219]	[358, 69, 39]	[289, 55, 4]	[51, 15, 4]	[889, 166, 144]
(10)	[−, −, −]	[5208, 1142, 38]	[162, 38, 38]	[755, 230, 24]	[70, 24, 24)	[548, 122, 122]
(11)	[−, −, −]	[2056, 119, 181]	[2056, 21, 30]	[539, 22, 28]	[539, 14, 14]	[493, 65, 110]
(12)	[−, −, −]	[3164, 27, 538]	[3164, 24, 58]	[185, 7, 36]	[185, 6, 21]	[414, 105, 105]

The modulus condition arrays that are shown in Table 1 are used similarly for each case in Table 2. The symbol "—" indicates that since it took more than one hour, the maximum number of terms was not calculated.

For $[T_A, T_B, T_C]$ in Table 2, T_A indicates the maximum number of terms that appear in any $F_1^{(k)}F_2^{(k)}$ such that $gP_1 = V \equiv F_1^{(k)}F_2^{(k)} \pmod M$ in the Hensel lifting step of each algorithm. Similarly, T_B indicates the maximum number of terms that appear in any $F_1^{(k)}$, and T_C indicates the maximum number of terms that appear in any $F_2^{(k)}$.

Remark. When the program that is based on each algorithm was carried out, $F_1^{(k)}$ and $F_2^{(k)}$ were computed in practice. However, $F_1^{(k)}F_2^{(k)}$ was not used directly. Instead, the author adopted the iteration of the equation $V^{(k+1)} = V^{(k)} - F_1^{(k)}\hat{F}_2^{(k)} - F_2^{(k)}\hat{F}_1^{(k)} - \hat{F}_1^{(k)}\hat{F}_2^{(k)}$, $k \in \mathbf{N}$ such that $F_1^{(k+1)} = F_1^{(k)} + \hat{F}_1^{(k)}$, $F_2^{(k+1)} = F_2^{(k)} + \hat{F}_2^{(k)}$, and $V^{(1)} = V - F_1^{(1)}F_2^{(1)}$ in the Hensel lifting routine of the program. Furthermore, note that the relation $T_A \leq T_B T_C$ does not always hold. For example, let $I = x + y^4$, $J = x + y^2$, and $gP_1 = V = IJ$. Then, by using $S = (Y) = (y^3 - 2)$ in the PI algorithm, $I = x + y(Y+2) \equiv F_1^{(k)} \pmod S$, $J = x + y^2 \equiv F_2^{(k)} \pmod S$, and $V = x^2 + (y^2 + y(Y+2))x + (Y+2)^2 \equiv F_1^{(k)}F_2^{(k)} \pmod S$, and so we have $T_B = 3$, $T_C = 2$ and $T_A = 7$.

From these results in Table 2, we can infer that the maximum number of terms that appear in $F_1^{(k)}F_2^{(k)}$ in the PI (or PIP) algorithm is less than that in the EZ-GCD algorithm (because in the latter case the numbers of terms in $P_1|_{y \to y+1, z \to z+1}$ for Nos. (3), (7), (10), (11), (12), $P_1|_{y \to y+3, z \to z+1}$ for No. (5), and $P_1|_{y \to y-1, z \to z+1}$ for No. (8), respectively, become more than 10000).

4. Conclusions

From the description of the previous section, the comparison between the computational complexity of the PI (or the PIP) algorithm and that of the EZ-GCD algorithm is consistent with the experimental results in Tables 1 and 2. Therefore, the former can greatly reduce the computational complexity and performs much faster than the latter in many cases. Consequently, the use of the former is a solution of the bad-zero problem.

In the future, it is desirable that the PI (or the PIP) algorithm should find an optimum PI (or PIP) automatically for input polynomials. Furthermore, the author thinks that the idea in this paper (especially, that of the PIP algorithm) can be applied to the factorization of a multivariate polynomial with higher-degree terms and the extension of the idea may be useful in finding the solution for several other problems.

Acknowledgements

The author acknowledges the referees of the previous version of the paper for valuable comments and Lester Clowney, Ph.D., of the National Institute of Advanced Industrial Science and Technology, for refinement of the paper.

References

Brown, W.S., Traub, J.F., 1971. On Euclid's algorithm and the theory of subresultants. J. ACM 18, 505–514.
Collins, G.E., 1967. Subresultants and reduced polynomial remainder sequences. J. ACM 14, 128–142.
Collins, G.E., 1966. Polynomial remainder sequences and determinants. Amer. Math. Monthly. 73 (7), 708–712.
Geddes, K.O., Czapor, S.R., Labahn, G., 1992. Algorithms for Computer Algebra. Kluwer Academic Publishers, 260–277.
Inaba, D., 2005. Factorization of multivariate polynomials by extended Hensel construction. SIGSAM Bull. 39, 142–154.
Kaltofen, E., Lobo, A., 1994. Factoring high-degree polynomials by the black box Berlekamp algorithm. In: Proc. ISSAC '94, pp. 90–98.
Kaltofen, E., 1985. Sparse Hensel lifting. In: Proc. EUROCAL 85, pp. 4–17.
Moses, J., Yun, D.Y.Y., 1973. The EZ GCD algorithm. In: Proc. ACM 73, pp. 159–166.

Noro, M., Yokoyama, K., 2002. Yet Another practical implementation of polynomial factorization over finite fields. In: Proc. ISSAC '02, pp. 200–206.
Sasaki, T., Suzuki, M., 1992. Three new algorithms for multivariate polynomial GCD. J. Symbolic Comput. 13, 395–411.
Wang, P.S., 1980. The EEZ-GCD algorithm. SIGSAM Bull. 14, 50–60.
Wang, P.S., 1978. An improved multivariate polynomial factoring algorithm. Math. Comp. 32, 1215–1231.
Yun, D.Y.Y., 1980. The Hensel Lemma in Algebraic Manipulation. Garland Publishing.
Zippel, R., 1993. Effective Polynomial Computation. Kluwer Academic Publishers.
Zippel, R., 1979. Probabilistic algorithms for sparse polynomials. In: Proc. EUROSAM '79, pp. 216–226.