Approximate Polynomial GCD over Integers with Digits-wise Lattice

Kosaku Nagasaka
Kobe University *
E-mail: nagasaka@main.h.kobe-u.ac.jp

Abstract

For the given coprime polynomials over integers, we change their coefficients slightly over integers so that they have a greatest common divisor (GCD) over integers. That is an approximate polynomial GCD over integers. There are only two algorithms known for this problem. One is based on an algorithm for approximate integer GCDs. The other is based on the well-known subresultant mapping and the lattice basis reduction. In this paper, we give an improved algorithm of the latter with a new lattice construction process by which we can restrict the range of perturbations. This helps us for computing approximate polynomial GCD over integers of the input erroneous polynomials having a priori errors on some digits of their coefficients.

Key words: Approximate Polynomial GCD, Lattice Basis Reduction

1 Introduction

Symbolic numeric algorithms for polynomials are very important, especially for practical computations since we have to operate with empirical polynomials having numerical errors on their coefficients. Recently, for those erroneous polynomials, many algorithms have been introduced, approximate univariate GCD and approximate multivariate factorization for example. However, for polynomials over integers having erroneous coefficients (e.g. rounded from empirical data), changing their coefficients over reals does not remain them in the polynomial ring over integers, hence we need algorithms designed over integers. In this paper, we discuss about computing a polynomial GCD of univariate or multivariate polynomials over integers approximately. Here, "approximately" means that we compute a polynomial GCD over integers by changing their coefficients slightly over integers so that the input polynomials still remain over integers. We improve one of known algorithms for computing an approximate polynomial GCD over integers defined below.

*3-11 Tsurukabuto, Nada-ku, Kobe 657-8501 JAPAN.

Definition 1 *(Approximate Polynomial GCD Over Integers)*
*Let $f(\vec{x})$ and $g(\vec{x})$ be polynomials in variables $\vec{x} = x_1, \ldots, x_\ell$ over $\mathbb{Z}$, and let ε be a small positive integer. If they satisfy $f(\vec{x}) = t(\vec{x})h(\vec{x}) + \Delta_f(\vec{x})$, $g(\vec{x}) = s(\vec{x})h(\vec{x}) + \Delta_g(\vec{x})$ and $\varepsilon = \max\{\| \Delta_f \|, \| \Delta_g \|\}$ for some polynomials $\Delta_f, \Delta_g \in \mathbb{Z}[\vec{x}]$, then we say that the above polynomial $h(\vec{x})$ is an **approximate GCD over integers**. We also say that $t(\vec{x})$ and $s(\vec{x})$ are **approximate cofactors over integers**, and we say that their **tolerance** is ε. ($\| p \|$ denotes a suitable norm of $p(\vec{x})$.)* ◁

Example 1 Let $f(x_1, x_2)$ and $g(x_1, x_2)$ be the following polynomials over integers, which are relatively prime and supposed to have numerical errors on their coefficients.

$$\begin{aligned} f(x_1,x_2) &= 1530x_1^2x_2^2 - 3601x_1^2x_2 + 2109x_1^2 - 171x_1x_2^2 \\ &\qquad + 3506x_1x_2 - 3703x_1 - 699x_2^2 + 94x_2 + 1561, \\ g(x_1,x_2) &= 2755x_1^2x_2^2 - 5851x_1^2x_2 + 3110x_1^2 - 5118x_1x_2^2 \\ &\qquad + 5296x_1x_2 + 351x_1 + 2275x_2^2 - 1098x_2 - 3822. \end{aligned}$$

We would find the following approximate GCD over integers, where the underlined figures are slightly changed to make them having a polynomial GCD.

$$\begin{aligned} f(x_1,x_2) &\approx (34x_1x_2 - 37x_1 - 25x_2 + 39) \times (45x_1x_2 - 57x_1 + 28x_2 + 40) \\ &= 1530x_1^2x_2^2 - 360\underline{3}x_1^2x_2 + 2109x_1^2 - 17\underline{3}x_1x_2^2 \\ &\qquad + 350\underline{4}x_1x_2 - 3703x_1 - \underline{700}x_2^2 + 9\underline{2}x_2 + 156\underline{0}, \\ g(x_1,x_2) &\approx (34x_1x_2 - 37x_1 - 25x_2 + 39) \times (81x_1x_2 - 84x_1 - 91x_2 - 98) \\ &= 275\underline{4}x_1^2x_2^2 - 585\underline{3}x_1^2x_2 + 31\underline{08}x_1^2 - 511\underline{9}x_1x_2^2 \\ &\qquad + 529\underline{4}x_1x_2 + 35\underline{0}x_1 + 2275x_2^2 - 109\underline{9}x_2 - 3822. \end{aligned}$$

In this case, $\Delta_f = 2x_1^2x_2 + 2x_1x_2^2 + 2x_1x_2 + x_2^2 + 2x_2 + 1$, $\Delta_g = x_1^2x_2^2 + 2x_1^2x_2 + 2x_1^2 + x_1x_2^2 + 2x_1x_2 + x_1 + x_2$ and $\varepsilon = 2$ in the ∞-norm. ◁

We note that for polynomials over the complex numbers, there are many studies and various algorithms ([9, 4, 2, 12, 27, 26, 3, 28, 19, 30, 29, 21, 10, 18, 6, 20, 5, 13, 17, 22, 23]). Hence one may think that we can compute an approximate GCD over integers by rounding the result by those algorithms since they compute approximate GCDs over complex numbers. However, it is difficult to make them as polynomials over integers since the resulting tolerance easily becomes large and far from the given polynomials (see [15]). Therefore, we need algorithms designed for polynomials over integers.

For computing approximate GCD over integers, there are two known algorithms. One is based on the result from approximate integer common divisors by Howgrave-Graham [8]. The other is based on the well-known subresultant mapping and the lattice basis reduction (the LLL algorithm [11]). The former algorithm is originally proposed by von zur Gathen and Shparlinski [25] at LATIN 2008 and revised by von zur Gathen et al [24]. Their algorithm only works for very tiny tolerances and one of input polynomials $f(\vec{x})$ and $g(\vec{x})$ must be given exactly and can not be perturbed though the algorithm always can compute an approximate GCD over integers if the given polynomials satisfy the certain conditions. The latter algorithm is proposed by the present author ([14]) at ISSAC 2008 and revised ([15]). In contrast with that by von zur Gathen et al., this algorithm works for not only very tiny but also small tolerances and all the given

polynomials can be perturbed (as described in the definition) though any theoretical condition which guarantees that the algorithm can compute an approximate GCD over integers, is not given.

1.1 The problem to be solved

In this paper, we give an improved algorithm with a new lattice construction process by which we can restrict the range of perturbations in some cases. This helps us for computing approximate polynomial GCD over integers of the input erroneous polynomials having a priori errors on some digits of their coefficients. For example, the known methods can not compute any approximate polynomial GCD over integers for the following polynomials.

$$\begin{aligned} f(x) &= -302260x^4 - 174933528x^3 + 45943440x^2 + 231047900996x - 143756712 \\ &\approx (889x^2 + 512701x - 319)\,(-340x^2 - 692x + 450648) - 2 \times 10^3 x^2, \\ g(x) &= 526407460x^4 + 303589900698x^3 - 690875197x^2 - 323202349x + 205289 \\ &\approx (889x^2 + 512701x - 319)\,(592140x^2 - 978x - 631) - 5 \times 10^3 x^4 + 4 \times 10^3. \end{aligned}$$

In this case, the tolerance (the absolute error) is 5×10^3 in the ∞-norm and the relative error is not small in relation to the smallest coefficients hence computing an approximate GCD over integers for this pair of polynomials is not so easy.

One may think that this example seems to be odd. However, this situation possibly occurs in some computations with multi-precision integers (each integer is represented as an array of word size integers). For example, transmission errors on some elements of the array, computing lower and higher digits separately and so on. In fact, the above pair of polynomials has perturbations on the second digit only (as an array of 10^3 integers) hence they are in this case. Moreover, this is also useful for simplifying algebraic expressions (e.g. each simplicity of expression is heavily depending on the number of terms not the magnitude of coefficients in general) as in the following polynomial.

$$\begin{aligned} f(x_1, x_2) &= (286x_2^2 - 54821x_2 - 3907787)\,x_1^2 \\ &\quad + (203830x_2^2 + 11276643x_2 + 35293)\,x_1 - 17930x_2^2 - 990865x_2 + 54765 \\ &= (22x_2 + 1217)((13x_2 - 3211)x_1^2 + (9265x_2 + 29)x_1 - 815x_2 + 45) \\ &\qquad + 5 \times 10^2 x_2 x_1. \end{aligned}$$

For this problem, we review the algorithm given by the author [15] in Section **2**. We give a new lattice construction process in Section **3**, including various numerical examples. In Section **4**, we give some remarks for this extension. We note that the present article is an extended work of the presentation ([16]) with the extended abstract at SNC 2011 (Symbolic-Numeric Computation, June 7-9, 2011, San Jose, California), and the ideal of this paper is based on the preliminary presentation about computing approximate GCD of integers (not polynomials) by the present author in Research Institute for Mathematical Sciences, Kyoto University in 2010.

2 Approximate GCD by Lattice Basis Reduction

We review the known result ([14, 15]) briefly. Let $f(\vec{x})$ and $g(\vec{x})$ have total degrees $n = \mathrm{tdeg}(f)$ and $m = \mathrm{tdeg}(g)$, respectively. We call the following mapping $\mathcal{S}_r(f, g)$ the subresultant mapping

of $f(\vec{x})$ and $g(\vec{x})$ of order r.

$$\mathcal{S}_r(f,g): \begin{array}{ccc} \mathcal{P}_{m-r-1} \times \mathcal{P}_{n-r-1} & \to & \mathcal{P}_{n+m-r-1} \\ (s(\vec{x}), t(\vec{x})) & \mapsto & s(\vec{x})f(\vec{x}) + t(\vec{x})g(\vec{x}) \end{array}$$

where $r = 0, \ldots, \min\{n, m\} - 1$ and $\mathcal{P}_d$ denotes the set of polynomials in variables $x_1, \ldots, x_\ell$, of total degree d or less. We denote the coefficient vector of polynomial $p(\vec{x})$ by $\mathrm{vect}(p)$ w.r.t. the lexicographic ascending order in this article. We note that any term order can be used for representing coefficient vectors since the order is not essential. To see the number of elements of a coefficient vector, we define the notation: $\beta_{d,r} = \binom{d-r+\ell}{\ell}$ hence the number of terms $x_1^{i_1} \cdots x_\ell^{i_\ell}$ satisfying $i_1 + \cdots + i_\ell \leq d$ can be denoted by $\beta_{d,0}$. The k-th convolution matrix $C_k(f)$ is defined to satisfy $C_k(f)\mathrm{vect}(p) = \mathrm{vect}(fp)$ for any polynomial $p(\vec{x})$ of total degree $k-1$ or less, where $\mathrm{vect}(p) \in \mathbb{Z}^{\beta_{k-1,0} \times 1}$ and $C_k(f) \in \mathbb{Z}^{\beta_{n+k-1,0} \times \beta_{k-1,0}}$. We have the matrix representation of the subresultant mapping: $Syl_r(f,g) = (C_{m-r}(f)\ C_{n-r}(g))$ of size $(\beta_{n+m-1,r}) \times (\beta_{m-1,r} + \beta_{n-1,r})$, satisfying

$$\mathcal{S}_r(f,g): \begin{array}{ccc} \mathcal{P}_{m-r-1} \times \mathcal{P}_{n-r-1} & \to & \mathcal{P}_{n+m-r-1} \\ (\ \mathrm{vect}(s)^{\ t}\ \mathrm{vect}(t)^{\ t}\ )^t & \mapsto & \mathrm{vect}(sf+tg) = Syl_r(f,g)(\ \mathrm{vect}(s)^{\ t}\ \mathrm{vect}(t)^{\ t}\ )^t. \end{array}$$

This mapping is the same as in [7], and has the same property that $f(\vec{x})/t(\vec{x})$ and $g(\vec{x})/s(\vec{x})$ is the GCD of $f(\vec{x})$ and $g(\vec{x})$ if r is the greatest integer such that this mapping is not injective. Hence by computing null vectors of $Syl_r(f,g)$ approximately for the given coprime polynomials, we can compute candidate vectors of approximate cofactors over integers. This procedure can be done by finding short vectors by the well-known LLL algorithm ([11]). For this, we construct the lattice generated by the row vectors of $\mathcal{L}(f,g,r,c)$ which is defined as the following matrix where r denotes the order of the subresultant mapping.

$$\mathcal{L}(f,g,r,c) = (E_{\beta_{n-1,r}+\beta_{m-1,r}} \mid c \cdot Syl_r(f,g)^t)$$

where E_i denotes the identity matrix of size $i \times i$ and $c \in \mathbb{Z}$. The size of $\mathcal{L}(f,g,r,c)$ is $(\beta_{n-1,r} + \beta_{m-1,r}) \times (\beta_{n-1,r} + \beta_{m-1,r} + \beta_{n+m-1,r})$. We note that we mark a block matrix with a vertical bar to distinguish the identity matrix representing a collection of linear combinations from the matrix formed by the coefficient vectors.

However, the short vectors found are only candidate cofactors $t(\vec{x})$ and $s(\vec{x}) \in \mathbb{Z}[\vec{x}]$ such that $s(\vec{x})f(\vec{x}) + t(\vec{x})g(\vec{x}) \approx 0$, and $f(\vec{x})$ and $g(\vec{x})$ may not be divisible by $h(\vec{x})$. To compute an approximate GCD from the candidate cofactors, we apply the LLL algorithm again to the lattice generated by the row vectors of the following matrix $\mathcal{H}(f,g,r,c,t,s)$ of size $(\beta_{r+1,0} + 1) \times (\beta_{n,0} + \beta_{m,0} + \beta_{r+1,0} + 1)$.

$$\mathcal{H}(f,g,r,c,t,s) = \left(E_{\beta_{r+1,0}+1} \;\middle|\; \begin{array}{cc} c \cdot \mathrm{vect}(f)^t & c \cdot \mathrm{vect}(g)^t \\ c \cdot C_{r+2}(-t)^t & c \cdot C_{r+2}(s)^t \end{array} \right)$$

We have the following lemmas in [15].

Lemma 1 *Let B be the maximum of coefficients of any factors of $f(\vec{x})$ and $g(\vec{x})$. For the lattice generated by the rows of $\mathcal{L}(f,g,r,c_{\mathcal{L}})$ with $c_{\mathcal{L}} = 2^{(\beta_{n-1,r}+\beta_{m-1,r}-1)/2}\sqrt{\beta_{n-1,r}+\beta_{m-1,r}}B$, the LLL algorithm can find a short vector whose first $\beta_{n-1,r}+\beta_{m-1,r}$ elements are a multiple of the transpose of the coefficient vectors of cofactors of $f(\vec{x})$ and $g(\vec{x})$ by their GCD, if r is the greatest integer such that the subresultant mapping is not injective.* ◁

Lemma 2 *Let B be the maximum of coefficients of any factors of $f(\vec{x})$ and $g(\vec{x})$. For the lattice generated by the row vectors of $\mathcal{H}(f, g, r, c_{\mathcal{H}}, t, s)$ with $c_{\mathcal{H}} = 2^{\beta_{r+1,0}/2}\sqrt{\beta_{r+1,0}+1}B+1$, the LLL algorithm can find a short vector whose 2-nd, ..., $(\beta_{r+1,0}+1)$-th elements are a multiple of the transpose of the coefficient vector of the GCD of $f(\vec{x})$ and $g(\vec{x})$, if r is the greatest integer such that the subresultant mapping is not injective.* $\triangleleft$

For example, we consider the following pair of erroneous polynomials.

$$\begin{aligned} f(x) &= 20x^2 + 1\underline{8}x - 2\underline{7} &&= (4x+7)(5x-4) - x + 1, \\ g(x) &= 2\underline{9}x^2 + 61x + \underline{19} &&= (4x+7)(7x+3) + x^2 - 2. \end{aligned}$$

We construct the following matrix $\mathcal{L}(f, g, r, c)$ with $r = 0$ and $c = 1$, and apply the LLL algorithm to the lattice generated by the row vectors of $\mathcal{L}(f, g, r, c)$.

$$\left(\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 19 & 61 & 29 & 0 \\ 0 & 1 & 0 & 0 & 0 & 19 & 61 & 29 \\ 0 & 0 & 1 & 0 & -27 & 18 & 20 & 0 \\ 0 & 0 & 0 & 1 & 0 & -27 & 18 & 20 \end{array}\right) \rightarrow \left(\begin{array}{cccc|cccc} \underline{-4} & \underline{5} & \underline{-3} & \underline{-7} & 5 & -14 & 3 & 5 \\ -5 & 6 & -3 & -9 & -14 & -2 & -1 & -6 \\ -7 & 9 & -5 & -13 & 2 & 5 & 12 & 1 \\ -4 & 5 & -3 & -8 & 5 & 13 & -15 & -15 \end{array}\right).$$

We take the first row vector as candidate cofactors (we note that we have to seek the candidate through all the short vectors). We construct the following matrix $\mathcal{H}(f, g, r, c, t, s)$ with $c = 1$ and apply the LLL algorithm, to compute an approximate GCD.

$$\left(\begin{array}{ccc|cccccc} 1 & 0 & 0 & -27 & 18 & 20 & 19 & 61 & 29 \\ 0 & 1 & 0 & -4 & 5 & 0 & 3 & 7 & 0 \\ 0 & 0 & 1 & 0 & -4 & 5 & 0 & 3 & 7 \end{array}\right) \rightarrow \left(\begin{array}{ccc|cccccc} 1 & \underline{-7} & \underline{-4} & 1 & -1 & 0 & -2 & 0 & 1 \\ 0 & 1 & 0 & -4 & 5 & 0 & 3 & 7 & 0 \\ 0 & 0 & 1 & 0 & -4 & 5 & 0 & 3 & 7 \end{array}\right).$$

Hence, we get $4x + 7$ as an approximate polynomial GCD over integers and $5x - 4$ and $7x + 3$ as approximate cofactors. We note that there are more complicated examples, some lemmas and techniques for decreasing the computing-time (see [14, 15]) though we do not show them here.

3 Digits-wise Lattice

The algorithms introduced in [14, 15] work well for nearby polynomials having polynomial GCD, according to the numerical experiments therein. However, they can not detect any approximate GCD for the following type of polynomials as noted in the introduction. We note again that this problem is not so special in practice (multi-precision integers, simplifying algebraic expressions and so on) though the word size we use here is 10^1 since this is easy to understand and does not exceed the paper width.

$$\begin{aligned} f(x) &= 32x^3 + \underline{7}6x^2 + \underline{2}2x + 15 &&= (4x+5)(8x^2+4x+3) + 20x^2 - 10x, \\ g(x) &= \underline{1}0x^3 + 53x^2 + 59x + \underline{4}0 &&= (4x+5)(5x^2+7x+6) - 10x^3 + 10. \end{aligned}$$

To extend the algorithms for the above case (all the coefficients have a priori errors on only the limited number of digits), we introduce the following digits-wise lattice instead of

$\mathcal{L}(f, g, r, c)$ by extending the coefficient vector to the digits-wise.

$$\mathcal{L}(f,g,r,c) = \left(\begin{array}{cccccc|cccccc} 1&0&0&0&0&0&40&59&53&10&0&0\\ 0&1&0&0&0&0&0&40&59&53&10&0\\ 0&0&1&0&0&0&0&0&40&59&53&10\\ 0&0&0&1&0&0&15&22&76&32&0&0\\ 0&0&0&0&1&0&0&15&22&76&32&0\\ 0&0&0&0&0&1&0&0&15&22&76&32 \end{array}\right) \Rightarrow \left(\begin{array}{cccccc|cccccccccc} 1&0&0&0&0&0&4&0&5&9&5&3&1&0&0&0&0&0\\ 0&1&0&0&0&0&0&0&4&0&5&9&5&3&1&0&0&0\\ 0&0&1&0&0&0&0&0&0&0&4&0&5&9&5&3&1&0\\ 0&0&0&1&0&0&1&5&2&2&7&6&3&2&0&0&0&0\\ 0&0&0&0&1&0&0&0&1&5&2&2&7&6&3&2&0&0\\ 0&0&0&0&0&1&0&0&0&0&1&5&2&2&7&6&3&2 \end{array}\right).$$

However, the row spaces of the above matrices are not the same and they are essentially different since digit-wise operations can not follow the carrying and borrowing operations for integers. For computing an approximate GCD we need to guarantee that the row space has the coefficient vectors corresponding to their cofactors, hence we have to perform some artificial carrying and borrowing operations on this matrix. To do this, we add some extra row vectors representing carry and borrow digits to the matrix as follows.

$$\left.\begin{array}{cccccc|cccccccccc} 1&0&0&0&0&0&4&0&5&9&5&3&1&0&0&0&0&0\\ 0&1&0&0&0&0&0&0&4&0&5&9&5&3&1&0&0&0\\ 0&0&1&0&0&0&0&0&0&0&4&0&5&9&5&3&1&0\\ 0&0&0&1&0&0&1&5&2&2&7&6&3&2&0&0&0&0\\ 0&0&0&0&1&0&0&0&1&5&2&2&7&6&3&2&0&0\\ 0&0&0&0&0&1&0&0&0&0&1&5&2&2&7&6&3&2\\ \hline 0&0&0&0&0&0&-1&10&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&-1&10&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&-1&10&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&-1&10&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&-1&10&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&-1&10 \end{array}\right\} \text{ extra rows.}$$

Moreover, in this case, we assume that only the second digit has a priori error hence we multiple the columns except ones corresponding to the second digit by 100 as penalty terms. The LLL algorithm gives the following result for the lattice generated by row vectors of this scaled matrix.

$$\left(\begin{array}{cccccc|cccccccccc} 1&-2&6&-2&1&-5&1&0&-5&0&-3&0&16&0&-5&0&-10&0\\ \underline{3}&\underline{4}&\underline{8}&\underline{-6}&\underline{-7}&\underline{-5}&3&0&10&0&3&0&-12&0&-14&0&-8&0\\ 0&-10&0&10&0&0&15&0&-18&0&17&0&-21&0&-10&0&0&0\\ 2&-4&2&-4&2&0&2&0&-10&0&-31&0&-5&0&13&0&2&0\\ 6&-2&-4&-2&6&0&21&0&32&0&2&0&11&0&-4&0&-4&0\\ -1&2&-16&2&9&5&-1&0&20&0&-15&0&1&0&-16&0&0&0\\ 2&4&2&-6&-3&0&-1&0&10&100&-10&0&-7&0&5&0&2&0\\ -3&-1&-4&6&1&0&-3&0&-7&0&10&0&-5&-100&-19&0&-4&0\\ 1&-2&4&-2&-1&0&1&0&-8&0&-8&100&0&0&16&0&4&0\\ 3&-5&-1&4&1&0&18&0&8&0&15&0&-9&0&-7&-100&-1&0\\ -3&-2&-4&6&3&1&-3&0&-8&0&10&0&7&0&-6&0&-1&200\\ 2&-4&2&1&2&0&9&500&1&0&7&0&11&0&13&0&2&0 \end{array}\right).$$

We can see that the resulting matrix has the row vector corresponding to the coefficient vectors of expected approximate cofactors $(8x^2+4x+3,\ 5x^2+7x+6)$ on the second row underlined. In the following subsections, we formalize this process into definitions and an algorithm.

3.1 Definitions of Digits-wise Representation

We denote the canonical form of length w of the base b digits in the integer a as

$$\forall a \in \mathbb{Z},\ \mathrm{digits}_{b,w}(a) = \{a_{w-1}, \ldots, a_1, a_0\} \ \text{ such that } \\ a = \textstyle\sum_{i=0}^{w-1} a_i b^i \ \text{ and } \ \begin{cases} 0 \le \mathrm{sign}(a) a_i < b & (i = 0, \ldots, w-2) \\ \mathrm{sign}(a) = \mathrm{sign}(a_i) & (i = w-1). \end{cases}$$

For example, we have $\mathrm{digits}_{10,2}(123) = \{12, 3\}$, $\mathrm{digits}_{10,3}(123) = \{1, 2, 3\}$, $\mathrm{digits}_{10,4}(123) = \{0, 1, 2, 3\}$ and $\mathrm{digits}_{10,3}(-123) = \{-1, -2, -3\}$. We extend the coefficient vector of polynomial $p(\vec{x})$ to the digits-wise operations and denote it by $\mathrm{vect}_{b,w}(p)$ where b and w are the base number and the length of the list of digits, respectively, such that

$$\mathrm{vect}_{b,w}(p) = \{\mathrm{digits}_{b,w}(p_e)\ \ldots\ \mathrm{digits}_{b,w}(p_0)\}^t \ \text{ where } \ \mathrm{vect}(p) = \{p_e\ \ldots\ p_0\}^t.$$

For example, we have $\mathrm{vect}_{10,2}(32x^3 + 76x^2 + 22x + 15) = \{1, 5, 2, 2, 7, 6, 3, 2\}^t$. We note that the sizes of the coefficient vectors $\mathrm{vect}_{b,w}(f)$ and $\mathrm{vect}_{b,w}(g)$ of $f(\vec{x})$ and $g(\vec{x})$ in the digits-wise form are $w \times \beta_{n,0}$ and $w \times \beta_{m,0}$, respectively. Therefore, their inverse mappings $\mathrm{digits}_{b,w}^{-1}(\cdot)$ and $\mathrm{vect}_{b,w}^{-1}(\cdot)$ can be defined as follows.

$$\mathrm{digits}_{b,w}^{-1}(\vec{a}) = \sum_{i=0}^{w-1} a_i b^i, \ \ \mathrm{vect}_{b,w}^{-1}(\vec{p}) = \mathrm{vect}^{-1}(\mathrm{digits}_{b,w}^{-1}(\vec{p}_{w\times\beta_{n,0}}), \ldots, \mathrm{digits}_{b,w}^{-1}(\vec{p}_0))$$

where $\vec{a} = \{a_{w-1}, \ldots, a_1, a_0\}^t \in \mathbb{Z}^w$ and $\vec{p} = \{\vec{p}^{\,t}_{w\times\beta_{n,0}}, \ldots, \vec{p}^{\,t}_0\}^t \in \mathbb{Z}^{w\times\beta_{n,0}}$, and $\mathrm{vect}^{-1}(\cdot)$ is the conventional mapping from the coefficient vector to the polynomial.

We also extend the k-th convolution matrix and the matrix representation of the subresultant mapping to the digits-wise operations in the same manner and denote them by $C_{k,b,w}(f)$ and $Syl_{r,b,w}(f, g)$, respectively. We note that they do not satisfy $C_{k,b,w}(f)\mathrm{vect}_{b,w}(p) = \mathrm{vect}_{b,w}(fp)$ for any polynomial $p(\vec{x})$ of total degree $k-1$ in general, however this is not the matter in our approach. Moreover, we have $\mathrm{vect}_{b,1}(f) = \mathrm{vect}(f)$, $C_{k,b,1}(f) = C_k(f)$ and $Syl_{r,b,1}(f, g) = Syl_r(f, g)$.

For the digits-wise lattice introduced in the beginning of this section, the carrying and borrowing are important hence we define the following carry-borrow vectors $\vec{z}_{b,w,i}$ $(i = 0, 1, \ldots, w-2)$ and matrix $\mathcal{Z}_{b,w}$, satisfying $\mathrm{digits}_{b,w}^{-1}(\vec{z}_{b,w,i}) = 0$ $(i = 0, 1, \ldots, w-2)$.

$$\vec{z}_{b,w,i} = \{\underbrace{0, \ldots, 0}_{i}, -1, b, \underbrace{0, \ldots, 0}_{w-i-2}\}^t \in \mathbb{Z}^w, \ \mathcal{Z}_{b,w} = \{\vec{z}_{b,w,0}\ \ldots\ \vec{z}_{b,w,w-2}\}^t \in \mathbb{Z}^{(w-1)\times w}.$$

We also extend $\mathcal{L}(f, g, r, c)$ and $\mathcal{H}(f, g, r, c, t, s)$ as follows and denote them by $\mathcal{L}_{b,w}(f, g, r, c)$

and $\mathcal{H}_{b,w}(f,g,r,c,t,s)$, respectively.

$$\mathcal{L}_{b,w}(f,g,r,c) = \left(\begin{array}{c|ccc} E_{\beta_{n-1,r}+\beta_{m-1,r}} & c\cdot Syl_{r,b,w}(f,g)^t & & \\ \hline & c\cdot\mathcal{Z}_{b,w} & & \\ & & c\cdot\mathcal{Z}_{b,w} & \\ & & & \ddots \\ & & & c\cdot\mathcal{Z}_{b,w} \end{array}\right),$$

$$\mathcal{H}_{b,w}(f,g,r,c,t,s) = \left(\begin{array}{c|ccc} E_{\beta_{r+1,0}+1} & \begin{array}{cc} c\cdot \mathrm{vect}_{b,w}(f)^t & c\cdot \mathrm{vect}_{b,w}(g)^t \\ c\cdot C_{r+2,b,w}(-t)^t & c\cdot C_{r+2,b,w}(s)^t \end{array} & & \\ \hline & c\cdot\mathcal{Z}_{b,w} & & \\ & & c\cdot\mathcal{Z}_{b,w} & \\ & & & \ddots \\ & & & c\cdot\mathcal{Z}_{b,w} \end{array}\right).$$

The sizes of $\mathcal{L}_{b,w}(f,g,r,c)$ and $\mathcal{H}_{b,w}(f,g,r,c,t,s)$ are $((\beta_{n-1,r}+\beta_{m-1,r})+(w-1)\beta_{n+m-1,r})\times(\beta_{n-1,r}+\beta_{m-1,r}+w\beta_{n+m-1,r})$ and $(\beta_{r+1,0}+1+(w-1)(\beta_{n,0}+\beta_{m,0}))\times(\beta_{r+1,0}+1+w(\beta_{n,0}+\beta_{m,0}))$, respectively.

Example 2 We show some examples of $\mathcal{L}_{b,w}(f,g,r,c)$ and $\mathcal{H}_{b,w}(f,g,r,c,t,s)$ for

$$\begin{array}{rcl} f(x) &=& 32x^3+56x^2+32x+15 \;=\; (4x+5)(8x^2+4x+3), \quad t(x) = -8x^2-4x-3, \\ g(x) &=& 20x^3+53x^2+59x+30 \;=\; (4x+5)(5x^2+7x+6), \quad s(x) = 5x^2+7x+6. \end{array}$$

We have the following matrices for the base number $b=10$ and length $w=2$ if we assume that the order of subresultant mapping is 0 and $c=1$.

$$\mathcal{L}_{10,2}(f,g,0,1) = \left(\begin{array}{cccccc|cccccccccccc} 1&0&0&0&0&0&3&0&5&9&5&3&2&0&0&0&0&0\\ 0&1&0&0&0&0&0&0&3&0&5&9&5&3&2&0&0&0\\ 0&0&1&0&0&0&0&0&0&0&3&0&5&9&5&3&2&0\\ 0&0&0&1&0&0&1&5&3&2&5&6&3&2&0&0&0&0\\ 0&0&0&0&1&0&0&0&1&5&3&2&5&6&3&2&0&0\\ 0&0&0&0&0&1&0&0&0&0&1&5&3&2&5&6&3&2\\ \hline 0&0&0&0&0&0&-1&10&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&-1&10&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&-1&10&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&-1&10&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&-1&10&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&-1&10 \end{array}\right),$$

$$\mathcal{H}_{10,2}(f,g,0,1,t,s) = \left(\begin{array}{ccc|cccccccccccccccc}
1&0&0& 1&5&3&2&5&6&3&2&3&0&5&9&5&3&2&0\\
0&1&0& 0&3&0&4&0&8&0&0&0&6&0&7&0&5&0&0\\
0&0&1& 0&0&0&3&0&4&0&8&0&0&0&6&0&7&0&5\\
\hline
0&0&0&-1&10&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0& 0&0&-1&10&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0& 0&0&0&0&-1&10&0&0&0&0&0&0&0&0&0&0\\
0&0&0& 0&0&0&0&0&0&-1&10&0&0&0&0&0&0&0&0\\
\hline
0&0&0& 0&0&0&0&0&0&0&0&-1&10&0&0&0&0&0&0\\
0&0&0& 0&0&0&0&0&0&0&0&0&0&-1&10&0&0&0&0\\
0&0&0& 0&0&0&0&0&0&0&0&0&0&0&0&-1&10&0&0\\
0&0&0& 0&0&0&0&0&0&0&0&0&0&0&0&0&0&-1&10
\end{array}\right).$$

◁

For any fixed non-negative integer n, $\mathrm{vect}_{b,w}(\cdot)$ and $\mathrm{vect}^{-1}_{b,w}(\cdot)$ can be thought as linear mappings over $\mathbb{Z}$ between $\mathcal{P}_n$ and $\mathbb{Z}^{w\times\beta_{n,0}}$ where $\mathcal{P}_n$ is a submodule of $\mathbb{Z}[\vec{x}]$ defined in the previous section. However, $\mathcal{P}_n$ and $\mathbb{Z}^{w\times\beta_{n,0}}$ are not isomorphic by these mappings. We define the quotient module of $\mathbb{Z}^{w\times\beta_{n,0}}$ by the equivalence relation "$\vec{f}\equiv\vec{g}$ iff $\mathrm{vect}^{-1}_{b,w}(\vec{f}) = \mathrm{vect}^{-1}_{b,w}(\vec{g})$" or its subspace generated by the row vectors of block diagonal matrix of $\{\mathcal{Z}_{b,w},\ldots,\mathcal{Z}_{b,w}\}$, and we denote this quotient module by $\mathbb{Z}^{w\times\beta_{n,0}}_{b,w}$. By these definitions, $\mathcal{P}_n$ is isomorphic to $\mathbb{Z}^{w\times\beta_{n,0}}_{b,w}$ by $\mathrm{vect}_{b,w}(\cdot)$ and $\mathrm{vect}^{-1}_{b,w}(\cdot)$.

Lemma 3 *Let B be a bound of maximum absolute value of coefficients of any factors of $f(\vec{x})$ and $g(\vec{x})$. For the lattice generated by the row vectors of $\mathcal{L}_{b,w}(f,g,r,c_{\mathcal{L}})$ with $c_{\mathcal{L}} = 2^{(\beta_{n-1,r}+\beta_{m-1,r}+(w-1)\beta_{n+m-1,r}-1)/2}\sqrt{\beta_{n-1,r}+\beta_{m-1,r}}B$, the LLL algorithm can find a short vector whose first $\beta_{n-1,r}+\beta_{m-1,r}$ elements are a multiple of the transpose of the coefficient vectors of cofactors of $f(\vec{x})$ and $g(\vec{x})$ by their GCD, if r is the greatest integer such that the subresultant mapping is not injective.* ◁

Proof There are cofactors $t(\vec{x})$ and $s(\vec{x})$ of $f(\vec{x})$ and $g(\vec{x})$ by their GCD, respectively, if r is the greatest integer such that the subresultant mapping is not injective. Hence, the lattice generated by row vectors of $\mathcal{L}_{b,w}(f,g,r,c_{\mathcal{L}})$ has the following vector $\vec{u}_{min}$ since $\mathbb{Z}^{w\times\beta_{n+m-1,r}}_{b,w}$ is isomorphic to $\mathcal{P}_{n+m-r-1}$ as shown above.

$$\vec{u}_{min} = (\text{the transpose of the coefficient vectors of } s(\vec{x}) \text{ and } t(\vec{x}),\ \underbrace{0\ \cdots\ 0}_{w\times\beta_{n+m-1,r}}).$$

The LLL algorithm can find a short vector $\vec{u}$ satisfying

$$\|\vec{u}\|_2 \leq 2^{(\beta_{n-1,r}+\beta_{m-1,r}+(w-1)\beta_{n+m-1,r}-1)/2}\ \|\vec{u}_{min}\|_2\ .$$

Since all the non-zero elements of right $w\times\beta_{n+m-1,r}$ columns of any row vectors in the lattice generated by the row vectors of $\mathcal{L}_{b,w}(f,g,r,c_{\mathcal{L}})$ must be larger than or equal to $c_{\mathcal{L}} = 2^{(\beta_{n-1,r}+\beta_{m-1,r}+(w-1)\beta_{n+m-1,r}-1)/2}\sqrt{\beta_{n-1,r}+\beta_{m-1,r}}B$ in absolute value, the right $w\times\beta_{n+m-1,r}$ columns of the found short vector $\vec{u}$ must be zeros. This means that the transpose of the vector formed by the first $\beta_{n-1,r}+\beta_{m-1,r}$ elements of $\vec{u}$ is in the null space of $Syl_{r,b,w}(f,g)$ hence in that of $Syl_r(f,g)$ and the lemma is proved. □

Lemma 4 *Let B be the maximum absolute value of coefficients of any factors of $f(\vec{x})$ and $g(\vec{x})$. For the lattice generated by the row vectors of $\mathcal{H}_{b,w}(f, g, r, c_{\mathcal{H}}, t, s)$ with $c_{\mathcal{H}} = 2^{(\beta_{r+1,0}+(w-1)(\beta_{n,0}+\beta_{m,0}))/2}\sqrt{\beta_{r+1,0}+1}B+1$, the LLL algorithm can find a short vector whose 2-nd, ..., $(\beta_{r+1,0}+1)$-th elements are a multiple of the transpose of the coefficient vector of the GCD of $f(\vec{x})$ and $g(\vec{x})$, if r is the greatest integer such that the subresultant mapping is not injective.* ◁

Proof The proof is similar to that of Lemma 3. □

We note that the short vectors corresponding to the GCD must have ± 1 on the first element since this means the number of coefficient vectors of $f(\vec{x})$ and $g(\vec{x})$ reduced by the coefficient vectors of cofactors. Moreover, this can be thought as the closest vector problem (CVP) hence it may be possible to use Babai's nearest plane algorithm ([1]) instead of the method based on the lattice in Lemma 4.

Example 3 For polynomials in Example 2, we have the following matrices with the base number $b = 10$, length $w = 2$, order $r = 0$, $c_{\mathcal{L}} = 9658$ and $c_{\mathcal{H}} = 4829$ if we use the Landau-Mignotte bound of $f(x)$ and $g(x)$.

$$\mathcal{L}_{10,2}(f, g, 0, 9658) =$$
$$\left(\begin{array}{cccccc|cccccccc}
1 & 0 & 0 & 0 & 0 & 0 & 28974 & 0 & 48290 & 86922 & \cdots & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 28974 & 0 & \cdots & 19316 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 48290 & 28974 & 19316 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 9658 & 48290 & 28974 & 19316 & \cdots & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 9658 & 48290 & \cdots & 28974 & 19316 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & \cdots & 48290 & 57948 & 28974 & 19316 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & -9658 & 96580 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -9658 & 96580 & \cdots & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & -9658 & 96580 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & -9658 & 96580
\end{array}\right),$$
$$\mathcal{H}_{10,2}(f, g, 0, 4829, t, s) =$$
$$\left(\begin{array}{ccc|ccccccccccc}
1 & 0 & 0 & 4829 & 24145 & 14487 & 9658 & 24145 & \cdots & 43461 & 24145 & 14487 & 9658 & 0 \\
0 & 1 & 0 & 0 & 14487 & 0 & 19316 & 0 & \cdots & 33803 & 0 & 24145 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 14487 & 0 & \cdots & 28974 & 0 & 33803 & 0 & 24145 \\
\hline
0 & 0 & 0 & -4829 & 48290 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -4829 & 48290 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -4829 & \cdots & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 48290 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & -4829 & 48290 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & -4829 & 48290
\end{array}\right).$$

By the LLL algorithm we found the following short vectors and in fact their first rows are corresponding to the coefficient vectors of cofactors and GCD of $f(x)$ and $g(x)$.

$$\mathcal{L}_{10,2}(f,g,0,9658) \Rightarrow$$
$$\left(\begin{array}{cccccc|cccccccccc} 3 & 4 & 8 & -6 & -7 & -5 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ -1 & 2 & -2 & 0 & 0 & 1 & -28974 & 0 & 0 & 9658 & \cdots & -9658 & 0 & -9658 & 19316 \end{array}\right),$$
$$\mathcal{H}_{10,2}(f,g,0,4829,t,s) \Rightarrow$$
$$\left(\begin{array}{ccc|cccccccccccc} 1 & -5 & -4 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 14487 & 0 & 4829 & 0 & \cdots & 4829 & 0 & -9658 & 0 & -24145 \end{array}\right).$$

Note that 1) we show only the first and second shortest short vectors found though there are more short vectors that are not corresponding to approximate cofactors and GCD, and 2) the LLL algorithm can find the expected short vectors with much smaller c_L and $c_{\mathcal{H}}$ in most cases. In fact, short vectors in this example can be computed from $\mathcal{L}_{10,2}(f,g,0,\underline{10})$ and $\mathcal{H}_{10,2}(f,g,0,\underline{10},t,s)$. ◁

3.2 Algorithm in Digits-wise Representation

We consider the case introduced in the beginning of this section hence we assume that all the coefficients have a priori errors on only the limited number of digits. For such polynomials, the resulting tolerance ε defined in Definition 1 easily becomes large even though the norm of errors in the digits-wise representation is small. We need to adapt the definition to the digits-wise representation. By the following definition, we have digits-wise tolerances $\varepsilon_{10,1} = \varepsilon = 20$, $\varepsilon_{10,2} = 2$ and $\varepsilon_{5,2} = 4$ in the ∞-norm for the pair of $f(x) = (4x+5)(8x^2+4x+3)+20x^2-10x$ and $g(x) = (4x+5)(5x^2+7x+6)-10x^3+10$ for example.

Definition 2 *(Digits-wise Approximate Polynomial GCD Over Integers)*
*Let $f(\vec{x})$ and $g(\vec{x})$ be polynomials in variables $\vec{x} = x_1, \ldots, x_\ell$ over $\mathbb{Z}$, and let ε be a small positive integer. If they satisfy $f(\vec{x}) = t(\vec{x})h(\vec{x}) + \Delta_f(\vec{x})$, $g(\vec{x}) = s(\vec{x})h(\vec{x}) + \Delta_g(\vec{x})$ and $\varepsilon_{b,w} = \max\{\| \mathrm{vect}_{b,w}(\Delta_f) \|, \| \mathrm{vect}_{b,w}(\Delta_g) \|\}$ for some polynomials $\Delta_f, \Delta_g \in \mathbb{Z}[\vec{x}]$, then we say that the above polynomial $h(\vec{x})$ is an **digits-wise approximate GCD over integers** w.r.t. the base number b and length w. We also say that $t(\vec{x})$ and $s(\vec{x})$ are **digits-wise approximate cofactors over integers**, and we say that their **tolerance** is $\boldsymbol{\varepsilon_{b,w}}$. ($\|p\|$ denotes a suitable vector norm.)* ◁

For computing digits-wise approximate GCD over integers, the lemmas introduced above do not guarantee that we can find the coefficient vectors of approximate cofactors and approximate GCD by the LLL algorithm. However, as same as the algorithms in [15], the short vectors found have a possibility that corresponding polynomials $t(\vec{x})$ and $s(\vec{x}) \in \mathbb{Z}[\vec{x}]$ satisfy $s(\vec{x})f(\vec{x}) + t(\vec{x})g(\vec{x}) \approx 0$, and they can be candidate approximate cofactors. Moreover, in the digits-wise representation, we have to distinguish correct digits from erroneous digits in the digits-wise lattice. We define the following diagonal weight matrix $\mathcal{W}_{b,w}(k_{id}, k_{cf}, c, \mathcal{E}, c_{\mathcal{E}})$ to distinguish them.

$$\mathcal{W}_{b,w}(k_{id}, k_{cf}, c, \mathcal{E}, c_{\mathcal{E}}) = \mathrm{diag}(\underbrace{1, \ldots, 1}_{k_{id}} \underbrace{\vec{w}, \ldots, \vec{w}}_{k_{cf}}),\ \vec{w} = \{c_{w-1}, \ldots, c_0\},\ c_i = \begin{cases} c_{\mathcal{E}} & (i \in \mathcal{E}) \\ c & (i \notin \mathcal{E}) \end{cases}$$

where we assume that the coefficients have a priori error on the i-th digits in the base b representation for any $i \in \mathcal{E} \subset \mathbb{Z}_{>0}$, and c and $c_{\mathcal{E}}$ are penalty weights that force the LLL algorithm to reduce more correct digits (columns) than other digits and reduce more erroneous digits than coefficient digits of candidate factors, respectively in the lattice basis. With this diagonal weight matrix, we define the following matrices that are based on $\mathcal{L}_{b,w}(f,g,r,1)$ and $\mathcal{H}_{b,w}(f,g,r,1,t,s)$, respectively.

$$\begin{aligned}\tilde{\mathcal{L}}_{b,w}(f,g,r,c,\mathcal{E},c_{\mathcal{E}}) &= \mathcal{L}_{b,w}(f,g,r,1)\mathcal{W}_{b,w}(\beta_{n-1,r}+\beta_{m-1,r}, w\beta_{n+m-1,r}, c, \mathcal{E}, c_{\mathcal{E}}),\\ \tilde{\mathcal{H}}_{b,w}(f,g,r,c,t,s,\mathcal{E},c_{\mathcal{E}}) &= \mathcal{H}_{b,w}(f,g,r,1,t,s)\mathcal{W}_{b,w}(\beta_{r+1,0}+1, w(\beta_{n,0}+\beta_{m,0}), c, \mathcal{E}, c_{\mathcal{E}}).\end{aligned}$$

Lemma 5 *Let B be the maximum absolute value of coefficients of any factors of $f(\vec{x})$ and $g(\vec{x})$ with perturbations. For the lattice generated by the rows of $\tilde{\mathcal{L}}_{b,w}(f,g,r,c_{\tilde{\mathcal{L}}},\mathcal{E},c_{\mathcal{E}})$ with the following $c_{\tilde{\mathcal{L}}}$, the LLL algorithm can find a short vector whose first $\beta_{n-1,r}+\beta_{m-1,r}$ elements are a multiple of the transpose of the coefficient vectors of candidate approximate cofactors of $f(\vec{x})$ and $g(\vec{x})$.*

$$c_{\tilde{\mathcal{L}}} = 2^{(\beta_{n-1,r}+\beta_{m-1,r}+(w-1)\beta_{n+m-1,r}-1)/2} \times \sqrt{(\beta_{n-1,r}+\beta_{m-1,r})B^2 + (\#\mathcal{E}\times\beta_{n+m-1,r})(b-1)^2c_{\mathcal{E}}^2}$$

where $\#\mathcal{E}$ is the number of elements in $\mathcal{E}$. ◁

Proof Let $t(\vec{x})$ and $s(\vec{x})$ be one of candidate approximate cofactors of $f(\vec{x})$ and $g(\vec{x})$, respectively, satisfying $\|\,\mathrm{vect}_{b,w}(s(\vec{x})f(\vec{x})+t(\vec{x})g(\vec{x}))\,\|\approx 0$. In this case, the lattice generated by rows of $\tilde{\mathcal{L}}_{b,w}(f,g,r,c_{\tilde{\mathcal{L}}},\mathcal{E},c_{\mathcal{E}})$ has the following vector $\vec{u}_{cac}$ for some integer r.

$$\vec{u}_{cac} = (\text{the transpose of the coefficient vectors of } s(\vec{x}) \text{ and } t(\vec{x}),\ \underbrace{*\ \cdots\ *}_{w\times\beta_{n+m-1,r}})$$

where all the correct digits are 0 on the right $w\times\beta_{n+m-1,r}$ elements denoted by $*$. The shortest vector of this lattice must be smaller than or equal to $\vec{u}_{cac}$ hence the LLL algorithm can find a short vector $\vec{u}$ satisfying

$$\begin{aligned}\|\vec{u}\|_2 &\le 2^{(\beta_{n-1,r}+\beta_{m-1,r}+(w-1)\beta_{n+m-1,r}-1)/2}\,\|\vec{u}_{cac}\|_2\\ &\le 2^{(\beta_{n-1,r}+\beta_{m-1,r}+(w-1)\beta_{n+m-1,r}-1)/2}\\ &\qquad\times\sqrt{(\beta_{n-1,r}+\beta_{m-1,r})B^2+(\#\mathcal{E}\times\beta_{n+m-1,r})(b-1)^2c_{\mathcal{E}}^2}\end{aligned}$$

since the left $\beta_{n-1,r}+\beta_{m-1,r}$ elements of $\vec{u}_{cac}$ are bounded by B and the erroneous digits on the right $w\times\beta_{n+m-1,r}$ elements of $\vec{u}_{cac}$ are bounded by $(b-1)c_{\mathcal{E}}$.

Therefore, all the correct digits on the right $w\times\beta_{n+m-1,r}$ elements of the found short vector $\vec{u}$ must be zeros since all the non-zero correct digits on the right $w\times\beta_{n+m-1,r}$ elements of row vectors in the lattice generated by the row vectors of $\tilde{\mathcal{L}}_{b,w}(f,g,r,c_{\tilde{\mathcal{L}}},\mathcal{E},c_{\mathcal{E}})$ are larger than or equal to $c_{\tilde{\mathcal{L}}}$ in absolute value. This means that the polynomials $t(\vec{x})$ and $s(\vec{x})$ whose coefficient vectors are the first $\beta_{n-1,r}+\beta_{m-1,r}$ elements of $\vec{u}$ satisfy

$$\|\text{all the correct digits of } \mathrm{vect}_{b,w}(s(\vec{x})f(\vec{x})+t(\vec{x})g(\vec{x}))\| = 0$$

hence they are candidate approximate cofactors of $f(\vec{x})$ and $g(\vec{x})$ though we may not guarantee $\|\mathrm{vect}_{b,w}(s(\vec{x})f(\vec{x})+t(\vec{x})g(\vec{x}))\|\approx 0$. □

Lemma 6 *Let B be the same maximum in Lemma 5. For the lattice generated by the row vectors of $\tilde{\mathcal{H}}_{b,w}(f,g,r,c_{\tilde{\mathcal{H}}},t,s,\mathcal{E},c_{\mathcal{E}})$ with the following $c_{\tilde{\mathcal{H}}}$, the LLL algorithm can find a short vector whose 2-nd, ..., $(\beta_{r+1,0}+1)$-th elements are a multiple of the transpose of the coefficient vector of a candidate approximate GCD of $f(\vec{x})$ and $g(\vec{x})$.*

$$c_{\tilde{\mathcal{H}}} = 2^{(\beta_{r+1,0}+1+(w-1)(\beta_{n,0}+\beta_{m,0})-1)/2}\sqrt{(\beta_{r+1,0}+1)B^2 + (\#\mathcal{E}\times(\beta_{n,0}+\beta_{m,0}))(b-1)^2c_{\mathcal{E}}^2}$$

where $\#\mathcal{E}$ is the number of elements in $\mathcal{E}$. ◁

Proof The proof is similar to that of Lemma 5. □

In general, there are short vectors that are not corresponding to approximate cofactors nor approximate GCD with small perturbations (small tolerance) hence the above lemmas can not guarantee that our algorithm always can find such a good approximate GCD. However, in most cases, according to our numerical experiment in Section **4**, the following algorithm works well, in which we use $c_{\mathcal{E}} = \sqrt{\beta_{n-1,r}+\beta_{m-1,r}}B$ and $c_{\mathcal{E}} = \sqrt{\beta_{r+1,0}+1}B$ for $\tilde{\mathcal{L}}_{b,w}(f,g,r,c,\mathcal{E},c_{\mathcal{E}})$ and $\tilde{\mathcal{H}}_{b,w}(f,g,r,c,t,s,\mathcal{E},c_{\mathcal{E}})$, respectively. We again note that $c_{\mathcal{E}}$ is a scaling weight to make the LLL algorithm do reducing more erroneous digits than coefficient digits of candidate cofactors and GCD, as in the proofs of Lemma 3 and Lemma 5.

Algorithm 1 (digits-wise approximate GCD over integers)
Input: $f,g\in\mathbb{Z}[\vec{x}], n=\mathrm{tdeg}(f), m=\mathrm{tdeg}(g),\ b,w\in\mathbb{Z}_{>0},\ \mathcal{E}\subset\{0,1,\ldots,w-1\}$.
Output: $h,t,s\in\mathbb{Z}[\vec{x}]$ satisfying $f(\vec{x})\approx t(\vec{x})h(\vec{x})$ and $g(\vec{x})\approx s(\vec{x})h(\vec{x})$, or "not found".
1. $\varepsilon\leftarrow 1$ and while $\varepsilon<\min\{\|\mathrm{vect}_{b,w}(f)\|,\|\mathrm{vect}_{b,w}(g)\|\}$ do **2**–**14**
(or do once for the possible smallest ε)
2. $r\leftarrow\min\{n,m\}-1$ and while $r\geq 0$ do **3**–**13** (or do once for $r=0$)
3. $c\leftarrow\max\{\|f\|,\|g\|\}$ and construct a matrix $\tilde{\mathcal{L}}_{b,w}(f,g,r,c,\mathcal{E},c_{\mathcal{E}})$
4. while $c\leq c_{\tilde{\mathcal{L}}}$ do **5**–**12** (or do once for $c=\max\{\|f\|,\|g\|\}$)
5. apply the LLL algorithm to the lattice generated by the row vectors of $\tilde{\mathcal{L}}_{b,w}(f,g,r,c,\mathcal{E},c_{\mathcal{E}})$
6. for each basis vector sorted by the norm of right $w\beta_{n+m-1,r}$ columns, do **7**–**11**
7. $c'\leftarrow\max\{\|f\|,\|g\|\}$ and construct a matrix $\tilde{\mathcal{H}}_{b,w}(f,g,r,c,t,s,\mathcal{E},c_{\mathcal{E}})$
8. while $c'\leq c_{\tilde{\mathcal{H}}}$ do **9**–**11** (or do once for $c'=\max\{\|f\|,\|g\|\}$)
9. apply the LLL algorithm to the lattice generated by the row vectors of $\tilde{\mathcal{H}}_{b,w}(f,g,r,c,t,s,\mathcal{E},c_{\mathcal{E}})$
10. let $h(\vec{x}),t(\vec{x}),s(\vec{x})$ be candidate approximate GCD and cofactors, and output $h(\vec{x}),t(\vec{x}),s(\vec{x})$ if $\max\{\|\mathrm{vect}_{b,w}(f-th)\|,\|\mathrm{vect}_{b,w}(g-sh)\|\}\leq\varepsilon$
11. $c'\leftarrow c'\times\max\{\|f\|,\|g\|\}$ (or multiply some positive integer)
12. $c\leftarrow c\times\max\{\|f\|,\|g\|\}$ (or multiply some positive integer)
13. $r\leftarrow r-1$
14. $\varepsilon\leftarrow\varepsilon\times 10$ (or multiply/add some positive integer)
15. output "not found".

Example 4 Algorithm 1 works for polynomials $f(x_1, x_2)$ and $g(x_1, x_2)$ below as follows.

$$\begin{aligned} f(x_1, x_2) &= 15336x_1^2 - 3651x_1x_2 - 11673x_1 - 1271x_2^2 + 11618x_2 - 15979, \\ g(x_1, x_2) &= 23184x_1^2 - 15094x_1x_2 + 53046x_1 + 2425x_2^2 - 19493x_2 + 26112. \end{aligned}$$

We assume that these polynomials have a priori errors on their 2nd and 3rd digits of coefficients in the base $b = 4$ presentation (note: $\log_4(\max\{\|f\|_\infty, \|g\|_\infty\}) \approx 7.85$). By the algorithm, we reduce the lattice generated by the row vectors of the following matrix of size 76×86 with $c_{\tilde{\mathcal{L}}} = 6986206386174202099$ and $c_{\mathcal{E}} = 2671636$.

$$\tilde{\mathcal{L}}_{4,8}(f, g, 0, c_{\tilde{\mathcal{L}}}, \{2, 3\}, c_{\mathcal{E}}) = \left(\begin{array}{c|ccc} 1\,0\,0\,0\,0\,0 & 6986206386174202099 & 13972412772348404198 \cdots & 0 \\ 0\,1\,0\,0\,0\,0 & 0 & 0 \cdots & 0 \\ 0\,0\,1\,0\,0\,0 & 0 & 0 \cdots & 0 \\ 0\,0\,0\,1\,0\,0 & 0 & -20958619158522606297 \cdots & 0 \\ 0\,0\,0\,0\,1\,0 & 0 & 0 \cdots & 0 \\ 0\,0\,0\,0\,0\,1 & 0 & 0 \cdots & 0 \\ \hline 0\,0\,0\,0\,0\,0 & -6986206386174202099 & 27944825544696808396 \cdots & 0 \\ 0\,0\,0\,0\,0\,0 & 0 & -6986206386174202099 \cdots & 0 \\ 0\,0\,0\,0\,0\,0 & 0 & 0 \cdots & 0 \\ 0\,0\,0\,0\,0\,0 & 0 & 0 \cdots & 0 \\ \vdots\,\vdots\,\vdots\,\vdots\,\vdots\,\vdots & \vdots & \vdots\ddots & \vdots \\ 0\,0\,0\,0\,0\,0 & 0 & 0 \cdots & 27944825544696808396 \end{array}\right).$$

We found the following short vectors that are sorted by the norm of right columns.

$$\left(\begin{array}{cccccc|ccccc} 313 & -41 & -213 & 512 & -71 & 322 & 0 & 0 & 0 & 0 & \cdots\ 0 \\ 165 & -21 & -113 & 272 & -43 & 170 & 0 & 0 & 0 & 0 & \cdots\ 0 \\ 295 & -39 & -203 & 480 & -73 & 302 & 0 & 0 & 0 & 0 & \cdots\ 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots\ \vdots \end{array}\right) \begin{array}{c} \\ -320596320 \\ -1485429616 \\ 1301086732 \\ \vdots \end{array} \begin{array}{c} -85492352 \\ -371357404 \\ 325939592 \\ \vdots \end{array}.$$

We construct the following matrix of size 88×100 for the first short vector in the step **9** with $c_{\tilde{\mathcal{H}}} = 399729686425627882725$ and $c_{\mathcal{E}} = 2181382$.

$$\tilde{\mathcal{H}}_{4,8}(f, g, 0, c_{\tilde{\mathcal{H}}}, t, s, \{2, 3\}, c_{\mathcal{E}}) = \left(\begin{array}{c|ccc} 1\,0\,0\,0 & 0 & -1199189059276883648175 \cdots & 0 \\ 0\,1\,0\,0 & 0 & 0 \cdots & 0 \\ 0\,0\,1\,0 & 0 & 0 \cdots & 0 \\ 0\,0\,0\,1 & 0 & 0 \cdots & -79459372851255765450 \\ \hline 0\,0\,0\,0 & -399729686425627882725 & 1598918745702511530900 \cdots & 0 \\ 0\,0\,0\,0 & 0 & -399729686425627882725 \cdots & 0 \\ 0\,0\,0\,0 & 0 & 0 \cdots & 0 \\ 0\,0\,0\,0 & 0 & 0 \cdots & 0 \\ \vdots\,\vdots\,\vdots\,\vdots & \vdots & \vdots\ddots & \vdots \\ 0\,0\,0\,0 & 0 & 0 \cdots & 1598918745702511530900 \end{array}\right).$$

We found the following short vectors that are sorted by the norm of right columns. We show only short vectors having ± 1 on their first elements as noted just after Lemma 4.

$$\left(\begin{array}{cccc|cccccccccc} 1 & 51 & -31 & 72 & 0 & 0 & 0 & 0 & 0 & -2181382 & 0 & \cdots & 0 \\ 1 & 51 & -27 & 72 & 0 & 0 & 0 & 0 & 0 & -2181382 & 0 & \cdots & 0 \\ 1 & 51 & -29 & 72 & 0 & 0 & 0 & 0 & 0 & -2181382 & 0 & \cdots & 0 \\ 1 & 49 & -31 & 72 & 0 & 0 & 0 & 0 & -19632438 & -8725528 & -39972968642562788272 5 & \cdots & 0 \end{array}\right).$$

Hence, we get $213x_1+41x_2-313$ and $322x_1-71x_2+512$ as approximate cofactors, $72x_1-31x_2+51$ as an approximate GCD of $f(x_1,x_2)$ and $g(x_1,x_2)$, and $\varepsilon_{4,8}=\sqrt{38}\approx 6.16$ in the Euclidean norm. Moreover, the perturbation polynomials are $(3\times 4^2-4^3)+(-3\times 4^2-2\times 4^3)x_2$ and $(-3\times 4^2-3\times 4^3)x_1+(2\times 4^2+3\times 4^3)x_2^2$. ◁

Example 5 Though the discussions above and Algorithm 1 are only for the case of two polynomials, it is easy to extend them to several polynomials, using the generalized subresultant mapping (see also [21, 15]). We show some example of the case of three polynomials below as follows.

$$\begin{array}{rcl} f(x_1,x_2) & = & 23112x_1^2-6999x_1x_2-6117x_1-1271x_2^2+11730x_2-15963, \\ g(x_1,x_2) & = & 2304x_1^2-6104x_1x_2+38432x_1+2201x_2^2-19493x_2+26224, \\ h(x_1,x_2) & = & -3744x_1^2+24724x_1x_2+6060x_1-9951x_2^2+12700x_2+6139. \end{array}$$

We assume that these polynomials have a priori errors on their 1st digits of coefficients in the base $b=16$ presentation (note: $\log_{16}(\max\{\|f\|_\infty,\|g\|_\infty,\|h\|_\infty\})\approx 3.81$). We construct a matrix of size 69×89 which is similar to $\tilde{\mathcal{L}}_{b,w}(f,g,r,c_{\tilde{\mathcal{L}}},\mathcal{E},c_{\mathcal{E}})$ with $c_{\tilde{\mathcal{L}}}=2816708953910864585$ and $c_{\mathcal{E}}=2443811$ and found the following short vectors that are sorted by the norm of right columns.

$$\left(\begin{array}{ccccccccc|ccccc} -313 & 41 & 321 & -512 & 71 & -32 & -121 & -321 & 52 & 0 & 0 & -5354389901 & \cdots & 0 \\ 496 & -64 & -512 & 816 & -112 & 48 & 192 & 512 & -80 & 0 & 0 & -2856815059 & \cdots & 0 \\ -205 & 29 & 213 & -336 & 51 & -16 & -77 & -213 & 36 & 0 & 0 & -1886622092 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \end{array}\right).$$

For the first short vector found, corresponding to candidate three cofactors, we construct a matrix of size 58×76 which is similar to $\tilde{\mathcal{H}}_{b,w}(f,g,r,c_{\tilde{\mathcal{H}}},t,s,\mathcal{E},c_{\mathcal{E}})$ with $c_{\tilde{\mathcal{H}}}=39365206313183407$ and $c_{\mathcal{E}}=1629208$ and found the following short vectors that are sorted by the norm of right columns.

$$\left(\begin{array}{cccc|ccccccccccccccccccccccccc} -1 & 51 & -31 & 72 & 0 & 0 & 0 & 0 & 0 & 0 & 6516832 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -4887624 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ -1 & 51 & -31 & 72 & 0 & 0 & 0 & 0 & 0 & 0 & 6516832 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -4887624 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ -1 & 51 & -31 & 72 & 0 & 0 & 0 & 0 & 0 & 0 & 6516832 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -4887624 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \end{array}\right).$$

Hence, we get $321x_1+41x_2-313$, $32x_1-71x_2+512$ and $-52x_1+321x_2+121$ as approximate cofactors, $72x_1-31x_2+51$ as an approximate GCD of $f(x_1,x_2)$, $g(x_1,x_2)$ and $h(x_1,x_2)$, and $\varepsilon_{16,4}=\sqrt{65}\approx 8.06$ in the Euclidean norm. Moreover, the perturbation polynomials are $3\times 16^1x_1-4\times 16^1x_2$, $-4\times 16^1x_1+7\times 16^1$ and $5\times 16^1x_2-2\times 16^1$. ◁

4 Remarks

To see the efficiency of Algorithm 1, we have generated several sets of 100 pairs of polynomials: A pair of bivariate polynomials of total degree randomly chosen from $[2, 6]$, having their GCD of total degree randomly chosen from $[1, 3]$, coefficients of their factors randomly chosen from $[100, 100]$ and added noise bivariate polynomials of the same total degree, whose coefficients are randomly chosen from $[-9, 9] \times 10^k$ but 0 at α probability, for randomly chosen erroneous digit k within the coefficient size. For example, the following pair of polynomials is one of them ($\alpha = 0.0$ and $k = 4$).

$$\begin{cases} (-85x_1^3 + 21x_2x_1^2 + 88x_1^2 + 18x_2^2x_1 - 99x_2x_1 + 17x_1 + 95x_2^3 - 49x_2^2 - 89x_2 - 96) \\ \quad \times (46x_1 + 92x_2 + 47) + (8 \times 10^4 x_1^4 + 1 \times 10^4 x_2x_1^3 - 1 \times 10^4 x_1^3 + 4 \times 10^4 x_2x_1^2 \\ \quad -7 \times 10^4 x_1^2 - 6 \times 10^4 x_2^3x_1 + 8 \times 10^4 x_2^2x_1 - 6 \times 10^4 x_2x_1 - 8 \times 10^4 x_1 \\ \quad -5 \times 10^4 x_2^4 - 7 \times 10^4 x_2^3 + 7 \times 10^4 x_2^2 + 7 \times 10^4 x_2 - 6 \times 10^4), \\ (-85x_1^3 + 21x_2x_1^2 + 88x_1^2 + 18x_2^2x_1 - 99x_2x_1 + 17x_1 + 95x_2^3 - 49x_2^2 - 89x_2 - 96) \\ \quad \times (-80x_1 + 83x_2 + 62) + (-8 \times 10^4 x_1^4 + 7 \times 10^4 x_2x_1^3 + 5 \times 10^4 x_1^3 + 9 \times 10^4 x_2^2x_1^2 \\ \quad -6 \times 10^4 x_2x_1^2 - 5 \times 10^4 x_1^2 + 8 \times 10^4 x_2^3x_1 + 4 \times 10^4 x_2^2x_1 - 9 \times 10^4 x_2x_1 \\ \quad -4 \times 10^4 x_1 + 5 \times 10^4 x_2^4 - 2 \times 10^4 x_2^3 + 6 \times 10^4 x_2^2 - 9 \times 10^4 x_2). \end{cases}$$

We have computed their approximate GCDs by the algorithm with $\varepsilon = 10$ in the step **1**, $r = 0$ in the step **2** and $c_{\tilde{\mathcal{L}}} = c_{\tilde{\mathcal{H}}} = 10^{10}$ and $c_{\mathcal{E}} = 10^5$ in the steps **3** and **7**. Note that all the experiments have been computed by our preliminary implementation on Mathematica 8.0, and we use the max norm for polynomials. Table 1 shows the results where "#success" denotes the number of pairs for which we got the expected digits-wise approximate polynomial GCD over integers and "#failure" denotes otherwise. According to the result, our algorithm works well for most of pairs of polynomials. However, the computation time is not good since the time-complexity of the lattice basis reduction is heavily depending on the number of bases that is the number of rows of matrices in our algorithm. Therefore, our algorithm works well but any faster algorithm is required to be used in the practical situation.

probability α	0.75		0.5		0.0	
	1st set	2nd set	1st set	2nd set	1st set	2nd set
#success:#failure	99:1	99:1	93:7	96:4	97:3	91:9

Table 1: The result of our experiments

Although we consider about only polynomials over integers in this paper, the digits-wise representation can be extended to polynomials over reals or complexes. For example, we can construct the Sylvester matrix of the given polynomials over reals in the digits-wise representation: dividing mantissae of coefficients into several elements if the given polynomials do not have both of small and large exponential parts. This may help us to treat erroneous coefficients having errors on only higher bits and should be studied as a further work.

The preliminary implementation on Mathematica 8.0, of our algorithm introduced in this paper with some examples can be found at the following URL: `http://wwwmain.h.kobe-u.ac.jp/~nagasaka/research/snap/snc2011plus.nb`.

Acknowledgments

The author would like to thank Prof. Kaltofen for having the personal conversation on approximate polynomial GCD over integers which is very helpful for the ideal of digit-wise lattice. Moreover, this work was supported in part by Japanese Ministry of Education, Culture, Sports, Science and Technology under Grant-in-Aid for Young Scientists, MEXT KAKENHI (22700011).

References

[1] L. Babai. On lovsz lattice reduction and the nearest lattice point problem. *Combinatorica*, 6:1–13, 1986.

[2] D. Christou and M. Mitrouli. Estimation of the greatest common divisor of many polynomials using hybrid computations performed by the ERES method. *Appl. Numer. Anal. Comput. Math.*, 2(3):293–305, 2005.

[3] R. M. Corless, S. M. Watt, and L. Zhi. QR factoring to compute the GCD of univariate approximate polynomials. *IEEE Trans. Signal Process.*, 52(12):3394–3402, 2004.

[4] G. M. Diaz-Toca and L. Gonzalez-Vega. Computing greatest common divisors and square-free decompositions through matrix methods: the parametric and approximate cases. *Linear Algebra Appl.*, 412(2-3):222–246, 2006.

[5] I. Z. Emiris, A. Galligo, and H. Lombardi. Numerical univariate polynomial GCD. In *The mathematics of numerical analysis (Park City, UT, 1995)*, volume 32 of *Lectures in Appl. Math.*, pages 323–343. Amer. Math. Soc., Providence, RI, 1996.

[6] I. Z. Emiris, A. Galligo, and H. Lombardi. Certified approximate univariate GCDs. *J. Pure Appl. Algebra*, 117/118:229–251, 1997. Algorithms for algebra (Eindhoven, 1996).

[7] S. Gao, E. Kaltofen, J. May, Z. Yang, and L. Zhi. Approximate factorization of multivariate polynomials via differential equations. In *ISSAC 2004*, pages 167–174. ACM, New York, 2004.

[8] N. Howgrave-Graham. Approximate integer common divisors. In *Cryptography and lattices (Providence, RI, 2001)*, volume 2146 of *Lecture Notes in Comput. Sci.*, pages 51–66. Springer, Berlin, 2001.

[9] N. Karcanias, S. Fatouros, M. Mitrouli, and G. H. Halikias. Approximate greatest common divisor of many polynomials, generalised resultants, and strength of approximation. *Comput. Math. Appl.*, 51(12):1817–1830, 2006.

[10] N. K. Karmarkar and Y. N. Lakshman. On approximate GCDs of univariate polynomials. *J. Symbolic Comput.*, 26(6):653–666, 1998. Symbolic numeric algebra for polynomials.

[11] A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261(4):515–534, 1982.

[12] T. Y. Li and Z. Zeng. A rank-revealing method with updating, downdating, and applications. *SIAM J. Matrix Anal. Appl.*, 26(4):918–946 (electronic), 2005.

[13] M. Mitrouli and N. Karcanias. Computation of the GCD of polynomials using Gaussian transformations and shifting. *Internat. J. Control*, 58(1):211–228, 1993.

[14] K. Nagasaka. Approximate polynomial gcd over integers. *ACM Communications in Computer Algebra*, 42(3):124–126, 2008. (ISSAC 2008 poster session).

[15] K. Nagasaka. Approximate polynomial gcd over integers. *J. Symbolic Comput.*, 46(12):1306–1317, 2011.

[16] K. Nagasaka. An improvement in the lattice construction process of approximate polynomial gcd over integers. In *Proceedings of Symbolic-Numeric Computation (SNC2011)*, pages 63–64. 2011. (extended abstract).

[17] M.-a. Ochi, M.-T. Noda, and T. Sasaki. Approximate greatest common divisor of multivariate polynomials and its application to ill-conditioned systems of algebraic equations. *J. Inform. Process.*, 14(3):292–300, 1991.

[18] V. Y. Pan. Approximate polynomial gcds, Padé approximation, polynomial zeros and bipartite graphs. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (San Francisco, CA, 1998)*, pages 68–77, New York, 1998. ACM.

[19] V. Y. Pan. Computation of approximate polynomial GCDs and an extension. *Inform. and Comput.*, 167(2):71–85, 2001.

[20] C. Rössner and J.-P. Seifert. The complexity of approximate optima for greatest common divisor computations. In *Algorithmic number theory (Talence, 1996)*, volume 1122 of *Lecture Notes in Comput. Sci.*, pages 307–322. Springer, Berlin, 1996.

[21] D. Rupprecht. An algorithm for computing certified approximate GCD of n univariate polynomials. *J. Pure Appl. Algebra*, 139(1-3):255–284, 1999. Effective methods in algebraic geometry (Saint-Malo, 1998).

[22] T. Sasaki and M.-T. Noda. Approximate square-free decomposition and root-finding of ill-conditioned algebraic equations. *J. Inform. Process.*, 12(2):159–168, 1989.

[23] A. Schönhage. Quasi-gcd computations. *J. Complexity*, 1(1):118–137, 1985.

[24] J. von zur Gathen, M. Mignotte, and I. E. Shparlinski. Approximate polynomial gcd: Small degree and small height perturbations. *J. Symbolic Comput.*, 45(8):879–886, 2010.

[25] J. von zur Gathen and I. E. Shparlinski. Approximate polynomial gcd: small degree and small height perturbations. In *LATIN 2008: Theoretical informatics*, volume 4957 of *Lecture Notes in Comput. Sci.*, pages 276–283. Springer, Berlin, 2008.

[26] C. J. Zarowski, X. Ma, and F. W. Fairman. QR-factorization method for computing the greatest common divisor of polynomials with inexact coefficients. *IEEE Trans. Signal Process.*, 48(11):3042–3051, 2000.

[27] Z. Zeng and B. H. Dayton. The approximate GCD of inexact polynomials. II. A multivariate algorithm. In *ISSAC 2004*, pages 320–327. ACM, New York, 2004.

[28] L. Zhi. Displacement structure in computing approximate GCD of univariate polynomials. In *Computer mathematics*, volume 10 of *Lecture Notes Ser. Comput.*, pages 288–298. World Sci. Publ., River Edge, NJ, 2003.

[29] L. Zhi and M.-T. Noda. Approximate GCD of multivariate polynomials. *Sūrikaisekikenkyūsho Kōkyūroku*, (1138):64–76, 2000. Research on the theory and applications of computer algebra (Japanese) (Kyoto, 1999).

[30] L. H. Zhi and M.-T. Noda. Approximate GCD of multivariate polynomials. In *Computer mathematics (Chiang Mai, 2000)*, volume 8 of *Lecture Notes Ser. Comput.*, pages 9–18. World Sci. Publ., River Edge, NJ, 2000.