Extended QRGCD Algorithm

Nagasaka, Kosaku

Masui, Takaaki

Extended QRGCD Algorithm*

Kosaku Nagasaka[1] and Takaaki Masui[2]

[1] Kobe University, 3-11 Tsurukabuto, Nada-ku, Kobe 657-8501 JAPAN
[2] Kobe-Takatsuka High School, 9-1 Mikatadai, Nishi-ku, Kobe 651-2277 JAPAN

Abstract. For computing the greatest common divisor of two univariate polynomials with a priori numerical errors on their coefficients, we use several approximate polynomial GCD algorithms: QRGCD, UVGCD, STLN-based, Fastgcd, GPGCD and so on. Among them, QRGCD is the most common algorithm since it has been distributed as a part of Maple and there are many papers including their comparisons of efficiency and effectiveness against QRGCD. In this paper, we give an improved QRGCD algorithm (ExQRGCD) which is unfortunately not faster than the original but more accurate and the resulting perturbation is able to satisfy the given tolerance.

1 Introduction

Computing the greatest common divisor (GCD) of polynomials is one of the most primitive computations in symbolic algebraic computations hence several algorithms are known: the Euclidean algorithm, the half GCD algorithm and so on (see also the recent result[1] and the text book[2]). However, in the practical situations (e.g. control theory, image processing and so on), the input polynomials are represented by the floating-point numbers or derived from the result of numerical computations or experiments hence in general they have a priori errors on their coefficients. For such polynomials, any conventional algorithms cannot compute their GCD since it easily becomes coprime due to a priori errors. This problem is called "approximate polynomial GCD" and there are many known studies (see Boito[3]).

Recently, most of modern algorithms for this problem use some optimization techniques and matrix decompositions. UVGCD[4], STLN-based[5], Fastgcd[6] and GPGCD[7] are typical algorithms using optimization techniques. The Gauss-Newton algorithm is used in UVGCD and Fastgcd to refine a tentative approximate GCD computed by some matrix decomposition and solving a linear system. Structured total least norm (STLN) techniques and the gradient projection algorithm are used in STLN-based and GPGCD, respectively, for seeking the approximate GCD directly.

QRGCD[8] is one of algorithms based on matrix decompositions and is also implemented as a part of the SNAP package of Maple. It uses the QR decomposition of the Sylvester matrix of the input polynomials and constructs approximate

* This work was supported in part by JSPS KAKENHI Grant Number 22700011.

GCD directly from the upper triangular matrix (i.e. QRGCD does not refine the resulting approximate factor). It is notable that QRGCD has been used as the benchmark algorithm for newly proposed algorithms. However, since QRGCD was proposed in the early stage of approximate GCD, its theoretical background is not enough analyzed (e.g. detectability of approximate GCD by QR factoring).

In this paper, we show some relative closeness and property of the upper triangular matrix, which clarify some weak points of QRGCD and for which we give an improved QRGCD algorithm (we call it ExQRGCD). Unfortunately our algorithm is not faster than the original but more accurate and the resulting perturbation is able to satisfy the given tolerance (note that QRGCD is not able to do this). We introduce the notations and the framework of the QRGCD algorithm in the rest of this section, and give our improved framework based on the original QRGCD in the section **2**. We give an improved QRGCD algorithm (ExQRGCD) with theoretical considerations in the section **3**. In the sections **4** and **5**, we give some numerical experiments and concluding remarks, respectively.

1.1 Notations

Let input polynomials $f(x), g(x) \in \mathbb{R}[x]$ of degree m, n be

$$f(x) = \sum_{i=0}^{m} f_i x^i,\ g(x) = \sum_{i=0}^{n} g_i x^i.$$

We assume that the input polynomials have the unit Euclidean norm (2-norm) (i.e. $\|f(x)\|_2 = \|g(x)\|_2 = 1$). Hence we scale the polynomials if they do not have the unit norm at the input. In this paper, we use the following (descending term order and row major) Sylvester matrix of $f(x)$ and $g(x)$.

$$Syl(f,g) = \begin{pmatrix} f_m & f_{m-1} & \cdots & f_1 & f_0 & & \\ & f_m & f_{m-1} & \cdots & f_1 & f_0 & \\ & & \ddots & \ddots & \cdots & \ddots & \ddots \\ & & & f_m & f_{m-1} & \cdots & f_1 & f_0 \\ g_n & g_{n-1} & \cdots & g_1 & g_0 & & \\ & g_n & g_{n-1} & \cdots & g_1 & g_0 & \\ & & \ddots & \ddots & \cdots & \ddots & \ddots \\ & & & g_n & g_{n-1} & \cdots & g_1 & g_0 \end{pmatrix}.$$

For vectors and matrices, $\|\cdot\|_2$ and $\|\cdot\|_F$ denote the Euclidean norm (2-norm) and the Frobenius norm, respectively. A^T and A^{-1} are the transpose[3] and the inverse of matrix A, respectively. To specify the submatrix of matrix A, we use the MATLAB like colon notation: the submatrix consisting of elements in the i_1 through i_2-th rows and j_1 through j_2-th columns is denoted by $A_{(i_1:i_2,j_1:j_2)}$.

[3] For the complex case, we just use the conjugate transpose (Hermitian transpose) instead of the transpose.

The coefficient vector (in the descending term order) of $p(x)$ of degree k is denoted by $\vec{p} \in \mathbb{R}^{k+1}$. We represent the evaluation of $p(x)$ at the point $\omega \in \mathbb{C}$ as $p(\omega) = \vec{p}^{\,T}\vec{\omega}_*$ where $\vec{\omega}_* = (\omega^k \ \cdots \ \omega^1 \ \omega^0)^T$. We denote the reversal polynomial of $p(x)$ by $\mathrm{rev}(p)$ (i.e. $\mathrm{rev}(p) = x^{\deg(p)}p(1/x)$). Moreover, we abbreviate "factor whose roots are outside the unit circle in the complex plane" to "outside-root factor" and also use "inside-root factor" similarly.

We note that approximate polynomial GCD has been defined from the several point of view in the literatures. In this paper, we use the following coefficient-wise definition without any constraint on the Bézout coefficients, which is compatible with most of known results.

Definition 1 (Approximate Polynomial GCD).
For the input polynomials[4] $f(x), g(x) \in \mathbb{R}[x]$, *we call the polynomial* $d(x) \in \mathbb{R}[x]$ *"approximate polynomial GCD" of tolerance* $\varepsilon \in \mathbb{R}_{\geq 0}$ *if it satisfies*

$$f(x) + \Delta_f(x) = f_1(x)d(x), \ g(x) + \Delta_g(x) = g_1(x)d(x)$$

for some polynomials $\Delta_f(x), \Delta_g(x), f_1(x), g_1(x) \in \mathbb{R}[x]$ *such that* $\deg(\Delta_f) \leq \deg(f)$, $\deg(\Delta_g) \leq \deg(g)$, $\|\Delta_f\|_2 < \varepsilon \|f\|_2$ *and* $\|\Delta_g\|_2 < \varepsilon \|g\|_2$ *where* $\|\cdot\|_2$ *denotes the* 2*-norm.* ◁

Remark 1. In the recent studies of approximate GCD, one may consider the extra conditions that $\deg(d)$ is maximized w.r.t. the tolerance ε or the tolerance ε is minimized w.r.t. $\deg(d)$. However, in this paper, we do not consider these conditions since it is difficult that the QRGCD or similar algorithms guarantee these properties without any refinement (optimization) step. ◁

1.2 Framework and Algorithm

The framework of the QRGCD algorithm is as follows.

1. Compute the QR decomposition of $Syl(f, g)$: $Syl(f, g) = QR$.
2. Find the gap between the k-th and $(k+1)$-th row vectors $\vec{r}_k, \vec{r}_{k+1}$ of R and form the polynomial with coefficients $\vec{r}_k$, which is an approximate polynomial GCD (or its factor).
3. Apply the same procedures to the reversal polynomials of cofactors since R may not have the approximate outside-root factor.

This is basically based on the following well known properties of the QR decomposition of the Sylvester matrix.

Lemma 1. *The last non-zero row vector of* R *gives the (mathematically exact) coefficients of the polynomial GCD of* $f(x)$ *and* $g(x)$. *(see e.g. [9])* ◁

Lemma 2. *The QR factoring is numerically backward stable. Thus,* $\|Syl(f, g) - QR\|_2$ *is enough small even if we compute the QR decomposition numerically. (see e.g. [10])* ◁

[4] We focus on polynomials over $\mathbb{R}$ for easy understanding though QRGCD and our algorithm can work also over $\mathbb{C}$.

We briefly review the original QRGCD algorithm below.

Algorithm 1 (QRGCD in the original paper).
Input: $f(x), g(x) \in \mathbb{R}[x]$ and tolerance $\varepsilon \in \mathbb{R}_{\geq 0}$.
Output: $u(x), v(x), d(x) \in \mathbb{R}[x]$ ($d(x)$ is an approx. GCD)

1. Compute the QR decomposition of $Syl(f,g)$: $Syl(f,g) = QR$.
2. Suppose that $R^{(k)}$ is the last $(k+1) \times (k+1)$ submatrix of R such that $\|R^{(k)}\|_2 > \varepsilon$ and $\|R^{(k-1)}\|_2 < \varepsilon$.
   Case 1: $\|R^{(0)}\|_2 > \varepsilon$ (Approximately Coprime)
   (a) $d_1(x) := 1$, $u(x)$ and $v(x)$ formed by rows of Q^T.
   Case 2: $\|R^{(k-1)}\|_2 < 10\varepsilon \, \|R^{(k)}\|_2$ (Big Gap Found)
   (a) $d_1(x) :=$ the last k-th row vector of R.
   Case 3: $\exists k_1$(biggest), $\|R^{(k_1-1)}\|_2 < 10\varepsilon \, \|R^{(k_1)}\|_2$ (Gap Found)
   (a) $d_1(x) :=$ the last k_1-th row vector of R.
   Case 4: Otherwise (Difficult Case)
   (a) find the inside-root factors of $f(x)$ and $g(x)$ by the algorithm "Split" and compute an approximate divisor $d_1(x)$ of the inside-root factors.
3. Apply the above steps to reversal polynomials of cofactors of $f(x)$ and $g(x)$ w.r.t. $d_1(x)$, to obtain $d_2(x)$.
4. Apply the above steps to cofactors $f(x)$ and $g(x)$ w.r.t. $d(x) = d_1(x)d_2(x)$, to obtain $u(x)$ and $v(x)$.
5. Output $u(x)$, $v(x)$ and $d(x)$. ◁

Here, the algorithm "Split" was proposed in the original paper[8] which finds the inside-root factor, and we also use it in our algorithm. This is done by the Graeffe's root-squaring, contour integration, Newton's formula, Newton's iteration and lifting steps. This algorithm is not the purpose of this paper and there are similar methods in the literatures hence we omit this in detail.

2 Improved QRGCD Framework

In this section, we give new lemmas and theorems that guarantee our framework similar to that of QRGCD above. At first, we cite the following lemma given by Stetter[11] (see also [12]), which plays an essential role in the frameworks of QRGCD and our improvement.

Lemma 3 (Corollary 7 in [11]).
For $\varepsilon \in \mathbb{R}_{\geq 0}$, let $\mathcal{P}_\varepsilon$ be the polynomial set s.t. $\mathcal{P}_\varepsilon = \{\tilde{p}(x) \in \mathbb{C}[x] \mid \deg(\tilde{p}) \leq \deg(p), \ \|p(x) - \tilde{p}(x)\|_2 \leq \varepsilon\}$. For $\omega \in \mathbb{C}$, we have

$$\exists \tilde{p}(x) \in \mathcal{P}_\varepsilon(p), \ \tilde{p}(\omega) = 0 \iff |p(\omega)| \leq \varepsilon \, \|\vec{\omega}_*\|_2 \ .$$

(note that the original is intended for the dual norm) ◁

2.1 Detectability of approximate GCD by QR factoring

Let $d(x)$ be the approximate polynomial GCD of $f(x)$ and $g(x)$, of degree k and having the unit 2-norm hence $d(x)$ is the exact GCD of $f(x)+\Delta_f(x)$ and $g(x)+\Delta_g(x)$. Similar to $f(x)$ and $g(x)$, we denote the Sylvester matrix of $f(x)+\Delta_f(x)$ and $g(x)+\Delta_g(x)$ by $Syl(f+\Delta_f, g+\Delta_g)$. For simplicity, we abbreviate $Syl(f,g)$ and $Syl(f+\Delta_f, g+\Delta_g)$ to S and $S+\Delta_S$, respectively, where $\Delta_S = Syl(\Delta_f, \Delta_g)$. For S and $S+\Delta_S$, let their QR decompositions be $S=QR$ and $S+\Delta_S = \hat{Q}\hat{R}$ where $Q, \hat{Q}$ are orthogonal and $R, \hat{R}$ are upper triangular matrices. We note that the notations in this paper are different from the notations used in the original paper[8] as the result of simplifications. Moreover, we denote the matrices whose column vectors generate the null spaces of S and $S+\Delta_S$ by N and $\hat{N}$, respectively, thus we have $SN = \mathbf{0}$ and $(S+\Delta_S)\hat{N} = \mathbf{0}$ and substituting $S = QR$ gives

$$-R\hat{N} = Q^T \Delta_S \hat{N}. \tag{2.1}$$

Lemma 4. *Let $r(x)$ be a polynomial with coefficients $\vec{r}$ which is a row vector of R, and ω be any root of $d(x)$. Then we have $|r(\omega)| \leq \|\Delta_S\|_2 \|\vec{\omega}_*\|_2$.* ◁

Proof. By the equality (2.1), we have $-R\vec{\omega}_* = Q^T \Delta_S \vec{\omega}_*$. Therefore, we have $|r(\omega)| \leq \|R\vec{\omega}_*\|_2 = \|Q^T \Delta_S \vec{\omega}_*\|_2 \leq \|\Delta_S\|_2 \|\vec{\omega}_*\|_2$ since $r(\omega)$ is an element of $R\vec{\omega}_*$. □

Note that any factor of $d(x)$ can be an approximate polynomial GCD by the definition. By $d_r(x)$ we denote such a factor of $d(x)$, and let $\Omega(d_r) = \{\omega_1, \ldots, \omega_k\}$ be the set of distinct roots of $d_r(x)$ and $\Omega_*(d_r)$ be the $k \times (m+n)$ matrix whose i-th row vector is $\vec{\omega}_{i*}^T = (\omega_i^{m+n-1} \ \cdots \ \omega_i^1 \ \omega_i^0)$, the transpose of the evaluation vector of ω_i. We denote the condition number of $\Omega_*(d_r)$ by $\kappa_2(\Omega_*(d_r))$.

Theorem 1. *Let $r(x)$ be a polynomial with coefficients $\vec{r}$ which is a row vector of R. Then, an upper bound of relative distance of $r(x)$ from $d_r(x)$, an approximate polynomial GCD of $f(x)$ and $g(x)$, is given by $\sqrt{k+1}\kappa_2(\Omega_*(d_r))\frac{\|\Delta_S\|_2}{\|r(x)\|_2}$. Moreover, there may exist another approximate polynomial GCD of higher degree other than $r(x)$ if $\frac{\|\Delta_S\|_2}{\|r(x)\|_2} > 1$.* ◁

Proof. First claim. Let $\delta_r(x) = d_r(x) - r(x)$ be the residual polynomial, and all the coefficient vectors are treated as polynomials of degree $m+n-1$ by padding zeros. By Lemma 4, for any $\omega \in \Omega(d_r)$ we have

$$|r(\omega)| \leq \|\Delta_S\|_2 \|\vec{\omega}_*\|_2 = \frac{\| \Delta_S \|_2}{\| r(x) \|_2} \|r(x)\|_2 \|\vec{\omega}_*\|_2,$$

hence the square root of the sum of $|r(\omega)|^2$ over all roots $\omega \in \Omega(d_r)$ gives

$$\|\Omega_*(d_r)\vec{r}\|_2 \leq \frac{\| \Delta_S \|_2}{| r(x) \|_2} \|r(x)\|_2 \|\Omega_*(d_r)\|_F \leq \sqrt{k+1}\frac{\| \Delta_S \|_2}{\| r(x) \|_2} \|r(x)\|_2 \|\Omega_*(d_r)\|_2 .$$

Therefore, we have the following relative upper bound of $\|\delta_r(x)\|_2$ since $\Omega_*(d_r)$ is of full row rank and $\Omega_*(d_r)\vec{\delta_r} = \Omega_*(d_r)(\vec{d_r} - \vec{r}) = -\Omega_*(d_r)\vec{r}$ can be solved by

the least square w.r.t. $\vec{\delta_r}$.

$$\frac{\| \vec{\delta_r} \|_2}{\| r(x) \|_2} \leq \sqrt{k+1}\kappa_2(\Omega_*(d_r))\frac{\| \Delta_S \|_2}{\| r(x) \|_2}.$$

Second claim. We note that Δ_S is the required perturbation of the $Syl(f,g)$ so that $f(x)+\Delta_f(x)$ and $g(x)+\Delta_g(x)$ may have an exact polynomial GCD (i.e. $d(x)$). Therefore, the row vector $\vec{r}$ can become the zero vector within the given tolerance if $\|\Delta_S\|_2 > \|r(x)\|_2$ since Q is orthogonal hence the perturbation of S equals to the perturbation of R in the 2-norm. This means that the degree of $d(x)$ may be larger than that of $r(x)$. □

If $\deg(d(x)) = \mathbf{1}$ then the first claim is proved easily and directly by Lemma 3. Note that Theorem 1 gives us only the necessary condition for that a computed row vector of R is close to $d(x)$. It does not give us any property of each row vector of R, which will be discussed in the next subsection. Moreover, we cannot know the magnitude $\|\Delta_S\|_2$ hence we have to estimate it in advance or in the algorithm. We will discuss this later in Section 3.

Remark 2. The condition number of $\Omega_*(d_r)$ becomes large if the approximate polynomial GCD has close roots. This means that QRGCD and our algorithm may be weak for polynomials having mutually close roots. ◁

Remark 3. A similar theorem is proved by Corless et al. in [8]. Their theorem is only for absolute closeness hence we extend it for the relative closeness. In general, lower row vectors of matrix R have smaller 2-norms. For example, we may have the case that the expected approximate polynomial GCD is $10^{-6}(x-1)(x-2)$ (before normalized) and the detected polynomial is $10^{-6}(x+1)(x+2)$. This result is not suitable but may happen since we compute the QR decomposition numerically. In order to prepare for such cases, we extend their theorem as above, though this is just a theoretical possibility (we didn't find any simple small example which has this behavior explicitly). ◁

2.2 Roots outside the unit circle

As reported by Corless et al. in [8], QR factoring of the Sylvester matrix may not detect any outside-root factor of approximate GCD, which has the (approximately[5]) common roots outside the unit circle. Therefore, the framework and the algorithm compute an inside-root factor and then compute an outside-root factor. In this subsection, we prove this from the different approach.

Let $p_1(x) = f(x), p_2(x) = g(x), p_3(x), \ldots \in \mathbb{R}[x]$ be the polynomial remainder sequence (PRS) of $f(x)$ and $g(x)$. We denote the k-th subresultant of $f(x)$ and $g(x)$ by $\mathrm{res}_k(f,g)$ which is defined by the determinant as follows.

$$\mathrm{res}_k(f,g) = \begin{vmatrix} Syl(f,x^n)_{(1:n-k,1:m+n-2k-1)} & \vec{x}_* f(x) \\ Syl(x^m,g)_{(n+1:m+n-k,1:m+n-2k-1)} & \vec{x}_* g(x) \end{vmatrix}$$

[5] By "approximately common root" we denote the roots of approximate GCD.

where $\vec{x}_* f(x) = \{x^{n-k-1} f(x) \ \cdots \ x^1 f(x) \ x^0 f(x)\}^T$, $\vec{x}_* g(x) = \{x^{m-k-1} g(x) \ \cdots \ x^1 g(x) \ x^0 g(x)\}^T$ and $f_i = 0$ and $g_i = 0$ for any negative index ($i < 0$). It is well known that $p_i(x)$ is a multiple of $\mathrm{res}_{\deg(p_{i-1})-1}(f, g)$ (see [13] for example) and the subresultant is related to the Sylvester's single sum ([14] and references therein) as follows.

Lemma 5 (Sylvester's single sum).
Let A and B be the sets of all the roots of $f(x)$ and $g(x)$, respectively. Then we have

$$\mathrm{res}_k(f, g) = \sum_{A' \subset A, \ \#A' = k} R(x, A') \frac{R(A \setminus A', B)}{R(A \setminus A', A')}$$

where $R(A, B) = \prod_{a \in A, \ b \in B}(a - b)$, $R(x, A) = \prod_{a \in A}(x - a)$ and $\#A$ denotes the cardinality of A. ◁

To describe some quality of PRS, we estimate the norm of each summand in the single sum for the following specific polynomials (similar estimation method used in [15] for clusters of close roots) where ω_{in_i} and $\hat{\omega}_{\mathrm{in}_j}$ are roots inside the unit circle and ω_{out_i} and $\hat{\omega}_{\mathrm{out}_j}$ are roots outside the unit circle. One may think that these polynomials are odd, however, this situation easily happens (see Example 1 for such actual polynomials).

$$\begin{aligned} f(x) &= \textstyle\prod_{i=1}^{m_{\mathrm{in}}}(x - \omega_{\mathrm{in}_i}) \prod_{i=1}^{m_{\mathrm{out}}}(x - \omega_{\mathrm{out}_i}), \ \deg(f) = m, \\ g(x) &= \textstyle\prod_{j=1}^{n_{\mathrm{in}}}(x - \hat{\omega}_{\mathrm{in}_j}) \prod_{j=1}^{n_{\mathrm{out}}}(x - \hat{\omega}_{\mathrm{out}_j}), \ \deg(g) = n \end{aligned}$$

where the magnitudes of roots satisfy $|\omega_{\mathrm{in}_i}| = O(\alpha)$, $|\hat{\omega}_{\mathrm{in}_j}| = O(\alpha)$, $|\omega_{\mathrm{out}_i}| = O(\alpha\beta)$, and $|\hat{\omega}_{\mathrm{out}_j}| = O(\alpha\beta)$ with the common big O notation, for $\alpha, \beta \in \mathbb{R}_{\geq 0}$ satisfying $\alpha < 1$ and $\alpha\beta > 1$. We assume that the roots of $f(x)$ and $g(x)$ are well isolated respectively, the approximate GCD of $f(x)$ and $g(x)$ is of degree k, the corresponding (approximately common) roots satisfy $|\omega_{\mathrm{in}_i} - \hat{\omega}_{\mathrm{in}_j}| = O(\alpha\gamma_{\mathrm{in}})$ and $|\omega_{\mathrm{out}_i} - \hat{\omega}_{\mathrm{out}_j}| = O(\alpha\beta\gamma_{\mathrm{out}})$ for some $\gamma_{\mathrm{in}}, \gamma_{\mathrm{out}} \in \mathbb{R}_{\geq 0}$, and the other pairs of roots satisfy $|\omega_{\mathrm{in}_i} - \hat{\omega}_{\mathrm{in}_j}| = O(\alpha)$ and $|\omega_{\mathrm{out}_i} - \hat{\omega}_{\mathrm{out}_j}| = O(\alpha\beta)$. Moreover, c_{in} and c_{out} denote the numbers of common inside roots of $f(x)$ and $g(x)$, and common outside roots of $f(x)$ and $g(x)$, respectively, hence we have $k = c_{\mathrm{in}} + c_{\mathrm{out}}$.

Lemma 6. *With the above notations and assumptions, for the coefficient of each summand of single sum representation of $\mathrm{res}_k(f, g)$, we have*

$$\frac{R(A \setminus A', B)}{R(A \setminus A', A')} = O(\alpha^{(m-k)(n-k)} \beta^{(m-k)(n-k)-(m_{\mathrm{in}}-k_{\mathrm{in}})(n_{\mathrm{in}}-k_{\mathrm{in}})} \gamma_{\mathrm{in}}^{c_{\mathrm{in}}-k_{\mathrm{in}_c}} \gamma_{\mathrm{out}}^{c_{\mathrm{out}}-k_{\mathrm{out}_c}})$$

where k_{in}, k_{in_c} and k_{out_c} denote the numbers of inside roots included in A', common inside roots included in A' and common outside roots included in A', respectively. ◁

Proof. At first, we estimate the numerator by multiplying each part (inside non-common roots: B_{in_s}, inside common roots: B_{in_c}, outside non-common roots:

B_{out_s} and outside common roots: B_{out_c}) of B as follows.

$$\begin{aligned}
R(A \setminus A', B) &= R(A \setminus A', B_{\mathrm{in}_s}) \times R(A \setminus A', B_{\mathrm{in}_c}) \\
&\quad \times R(A \setminus A', B_{\mathrm{out}_s}) \times R(A \setminus A', B_{\mathrm{out}_c}), \\
R(A\setminus A', B_{\mathrm{in}_s}) &= O(\alpha^{(m_{\mathrm{in}}-k_{\mathrm{in}})(n_{\mathrm{in}}-c_{\mathrm{in}})}(\alpha\beta)^{(m_{\mathrm{out}}-k_{\mathrm{out}})(n_{\mathrm{in}}-c_{\mathrm{in}})}), \\
R(A\setminus A', B_{\mathrm{in}_c}) &= O(\alpha^{(m_{\mathrm{in}}-k_{\mathrm{in}})c_{\mathrm{in}}-(c_{\mathrm{in}}-k_{\mathrm{in}_c})}(\alpha\gamma_{\mathrm{in}})^{c_{\mathrm{in}}-k_{\mathrm{in}_c}}(\alpha\beta)^{(m_{\mathrm{out}}-k_{\mathrm{out}})c_{\mathrm{in}}}), \\
R(A\setminus A', B_{\mathrm{out}_s}) &= O((\alpha\beta)^{(m-k)(n_{\mathrm{out}}-c_{\mathrm{out}})}), \\
R(A\setminus A', B_{\mathrm{out}_c}) &= O((\alpha\beta)^{(m_{\mathrm{in}}-k_{\mathrm{in}})c_{\mathrm{out}}}(\alpha\beta\gamma_{\mathrm{out}})^{c_{\mathrm{out}}-k_{\mathrm{out}_c}} \\
&\quad \times (\alpha\beta)^{(m_{\mathrm{out}}-k_{\mathrm{out}})c_{\mathrm{out}}-(c_{\mathrm{out}}-k_{\mathrm{out}_c})}).
\end{aligned}$$

Therefore, we have

$$R(A\setminus A', B) = O(\alpha^{(m-k)n}\beta^{(m-k)n_{\mathrm{out}}+(m_{\mathrm{out}}-k_{\mathrm{out}})n_{\mathrm{in}}}\gamma_{\mathrm{in}}^{c_{\mathrm{in}}-k_{\mathrm{in}_c}}\gamma_{\mathrm{out}}^{c_{\mathrm{out}}-k_{\mathrm{out}_c}}).$$

Next, we estimate the denominator in the same way.

$$\begin{aligned}
R(A \setminus A', A') &= R(A \setminus A', A'_{\mathrm{in}}) \times R(A \setminus A', A'_{\mathrm{out}}), \\
&= O(\alpha^{(m_{\mathrm{in}}-k_{\mathrm{in}})k_{\mathrm{in}}}(\alpha\beta)^{(m_{\mathrm{out}}-k_{\mathrm{out}})k_{\mathrm{in}}}) \times O((\alpha\beta)^{(m-k)k_{\mathrm{out}}}) \\
&= O(\alpha^{(m-k)k}\beta^{(m-k)k_{\mathrm{out}}+(m_{\mathrm{out}}-k_{\mathrm{out}})k_{\mathrm{in}}}).
\end{aligned}$$

By substituting $m_{\mathrm{out}} - k_{\mathrm{out}} = (m-k) - (m_{\mathrm{in}} - k_{\mathrm{in}})$ for the power of β, the lemma follows from the above directly. □

Theorem 2. *There exists a pair of polynomials $f(x)$ and $g(x)$ such that the QR decomposition of $Syl(f,g)$ cannot detect any outside-root factor of the approximate GCD. Moreover, we need to detect such factors from* $\mathrm{rev}(f)$ *and* $\mathrm{rev}(g)$ *or their cofactors several times and combine them.* ◁

Proof. With the assumptions of Lemma 6 for a fixed $k = \deg(d)$, let A'_0 be the set of all the (approximately) common roots of $f(x)$ and $g(x)$ and A'_s be a set of A'_0 replaced s common outside roots with s (not common) inside roots. We focus on the roots of $r(x)$ with coefficients $\vec{r}$ which is the last $(k+1)$-th row vector of R.

By Lemmas 5 and 6, the coefficient $R(A\setminus A'_s, B)/R(A\setminus A'_s, A'_s)$ of $R(x, A'_s)$, is $O(\beta^{(m_{\mathrm{in}}+n_{\mathrm{in}}-2k_{\mathrm{in}}-s)s}\gamma_{\mathrm{out}}^{s})$ times larger than $R(A \setminus A'_0, B)/R(A \setminus A'_0, A'_0)$ of $R(x, A'_0)$. Therefore, $r(x)$ easily becomes to be without common outside roots, regardless of their closeness, hence the QR decomposition cannot detect any outside-root factor of the approximate GCD. For such factors, we have to compute the QR decomposition of matrix of $\mathrm{rev}(f)$ and $\mathrm{rev}(g)$ (i.e. transform the outside roots into inside). □

One may think that Theorems 1 and 2 seem like a contradiction since Theorem 1 does not restrict roots to the inside. However, there is no contradiction. Theorem 2 states only one of properties of PRS which may not have common outside roots. As the result of the computation of PRS, Theorem 1 states only a necessary condition that the resulting PRS is an approximate GCD (or its approximate factor).

Moreover, the above theorem indicates a concrete example below that we have to compute approximate GCD from the both of input and reversal polynomials. In the rest of paper, by "normal side" and "reversal side" we denote the computations from the input and reversal polynomials (finding inside and outside roots), respectively.

Example 1 (Fake Common Inside Roots).
We compute the QR decomposition of $Syl(f, g)$ of the following $f(x)$ and $g(x)$.

$$\begin{aligned} f(x) &= (x+0.001)(x-0.001)(x+1000)(x-1000), \\ g(x) &= (x+0.0010000001)(x-0.009)(x+1000.0001)(x-2000). \end{aligned}$$

They have the following approximate GCD of tolerance 10^{-12}.

$$\begin{aligned} d(x) &= 9.99998 \times 10^{-4}x^2 + 0.999999x + 9.99998 \times 10^{-4} \\ &\approx 9.99998 \times 10^{-4}(x + 0.00100000)(x + 1000.00). \end{aligned}$$

The 2-nd, 3-rd and 4-th last row vectors of R are as follows.

$$\begin{aligned} \text{4-th last row: } & 0.00143003x^3 + 1.39855x^2 - 0.00559397x - 6.99252 \times 10^{-6} \\ & \approx 0.00143003(x - 0.00499981)(x + 0.001)(x + 977.993), \\ \text{3-rd last row: } & 0.20978x^2 - 0.000839074x - 1.04885 \times 10^{-6} \\ & \approx 0.20978(x - 0.00499978)(x + 0.001), \\ \text{2-nd last row: } & 0.00565685x + 5.65685 \times 10^{-6} \\ & \approx 0.00565685(x + 0.001). \end{aligned}$$

As in the proof of Theorem 2, the 3-rd last row is not related to the above $d(x)$ and has a non-common inside root (i.e. 0.00499978 instead of approximate value of 1000). To detect the correct common outside root (≈ 1000), we have to transform the outside roots into inside. This is a reason that we have to compute with reversal polynomials. In the rest of this paper, these non-common inside roots detected instead of the common outside roots are called "fake common inside roots". ◁

We note that the QRGCD algorithm also works for this example since there is only one fake common root. However, for polynomials having more than one fake common roots, QRGCD may not detect the expected degree of GCD since QRGCD computes from the normal and reversal sides only once. For example, suppose that the row vectors of R consist of roots in this order from bottom to top: common, fake, common, fake, common roots. In this case, we need to compute from the normal, reversal and normal sides (the last normal side is not done by QRGCD). This fact leads us the improved QRGCD algorithm in the next section.

3 Improved QRGCD Algorithm

We propose the following algorithm based on the discussions, which is unfortunately not faster than the original but more accurate and the resulting perturbation is able to satisfy the given tolerance. We will explain the algorithm in subsections 3.1–3.5.

Algorithm 2 (Extended QRGCD).

Input: $f(x), g(x) \in \mathbb{R}[x]$ and tolerance $\varepsilon \in \mathbb{R}_{\geq 0}$.

Output: $f_1(x), g_1(x), d(x) \in \mathbb{R}[x]$ s.t.

$$\|f(x) - d(x)f_1(x)\|_2 < \varepsilon,\ \|g(x) - d(x)g_1(x)\|_2 < \varepsilon.$$

1. Determine the first side (normal or reversal sides), put $i := 1$, $f_1(x) := f(x)$ and $g_1(x) := g(x)$ (or $\mathrm{rev}(f)$ and $\mathrm{rev}(g)$, respectively, if the first side is reversal).
2. Compute the QR decomposition of $Syl(f_1, g_1)$: $Syl(f_1, g_1) = QR$.
3. Suppose that $R^{(k)}$ is the bottom-rightmost $(k+1) \times (k+1)$ submatrix of R and $\vec{r}_k$ is the top row vector of $R^{(k)}$.
   Case 1: $\|R^{(0)}\|_F > \varepsilon\sqrt{m+n}$ (Approx. Coprime)
   (a) $d_i(x) := 1$.
   Case 2: $\|R^{(0)}\|_F \leq \varepsilon\sqrt{m+n}$ (Trial Divisions)
   (a) for all $k \geq 1$ s.t. $\|R^{(k-1)}\|_F \leq \varepsilon\sqrt{m+n}$, compute $er_k := \|R^{(k-1)}\|_F / \|\vec{r}_k\|_2$, and sort er_k in ascending order s.t. $er_{k_1} \leq er_{k_2} \leq er_{k_3} \leq \cdots$.
   (b) for $k = k_1, k_2, k_3$ (i.e. up to the 3-rd smallest er_k at most), do
      (i) $d_i(x) := r_k(x)$ and if $\exists f_1, g_1$, $\|f(x) - d(x)f_1(x)\|_2 < \varepsilon$ and $\|g(x) - d(x)g_1(x)\|_2 < \varepsilon$ for $d(x) := \prod d_i(x)$, then goto step 4.
      (if no factor found in the recursive call by step (e) below, then put $d_i(x) := 1$ and goto step 4.)
   (c) find the last two rows of R whose norm is not less than 1.0 and let $p_1(x)$ and $p_2(x)$ be their polynomial representations s.t. $\deg(p_1) > \deg(p_2)$.
   (d) find the inside-root factors of $p_1(x), p_2(x)$ by the algorithm "Split" and let them be $p_{1_{\mathrm{in}}}(x), p_{2_{\mathrm{in}}}(x)$.
   (e) apply steps 2 and 3 to $p_{1_{\mathrm{in}}}(x)$ and $p_{2_{\mathrm{in}}}(x)$ and form the divisor $d_i(x)$.
4. $i := i + 1$, change the side (normal $\leftrightarrow$ reversal) and apply the above steps 2 and 3 to (if in the reversal side, reversal polynomials of) cofactors $f_1(x)$ and $g_1(x)$ of $f(x)$ and $g(x)$ w.r.t. $d(x) := \prod d_i(x)$, to obtain $d_{i+1}(x)$ until $d_i(x) = d_{i+1}(x) = 1$ for some i.
5. Output $f_1(x)$, $g_1(x)$ and $d(x)$. ◁

3.1 The matrix norm used

In the algorithm, we use the Frobenius norm instead of the 2-norm since it is easy to compute. It should be the 2-norm if we can compute it fastly (but usually not fast).

3.2 Approximate coprime condition

The QRGCD and our algorithms are based on the QR decomposition of $Syl(f, g)$ hence they cannot detect $d(x)$ directly but may be able to detect $r(x)$ close to $d(x)$ as the result of the QR factoring. Therefore, any (approximately) coprime detection must not be against the expected $d(x)$ but be against $r(x)$ with coefficients $\vec{r}$ which is the last $(k+1)$-th row of R. From this point of view, $\|R^{(k)}\|_F$ represents a sufficient "unstructured" magnitude to make $Syl(f, g)$ to

be rank deficient and $f(x)$ and $g(x)$ may have $r(x)$ as an approximate GCD. Note that "unstructured" perturbation does not preserve a Toeplitz-block structure of Sylvester matrix $Syl(f,g)$. Let $f(x)+\delta_f$ and $g(x)+\delta_g$ be polynomials whose (exact) GCD is $r(x)$. In general, $Syl(\delta_f,\delta_g)$ which makes $Syl(f+\delta_f, g+\delta_g)$ to be rank deficient is the "structured" perturbation hence its norm is larger than $\|R^{(k)}\|_F$ of "unstructured" perturbation in most cases. This means that $r(x)$ is not any approximate GCD if we have $\|R^{(k)}\|_F > \varepsilon\sqrt{m+n}$ hence $\|Syl(\delta_f,\delta_g)\|_F > \varepsilon\sqrt{m+n}$.

3.3 Trial divisions

By the same reason of the above subsection and Theorem 1, the QR decomposition might not detect all the inside common roots but detect some $r(x)$ close to an approximate factor of $d(x)$ and its closeness can be estimated by $er_k := \|R^{(k-1)}\|_F / \|\vec{r}_k\|_2$ though this is depending on the condition number $\Omega_*(d_r)$ and is not the sufficient condition. If we take $r(x)$ of the maximal degree, $r(x)$ may have some fake inside roots and it becomes difficult to detect the common outside roots from the reversals of cofactors. Therefore, we try to find $r(x)$ for which er_k is as small as possible (since we cannot estimate the condition number $\Omega_*(d_r)$). According to our experiments, the smallest er_k gives enough close factors in most cases but to make sure we try to do trial divisions up to the 3-rd smallest er_k at most.

3.4 The polynomials to be applied to "Split"

Let $p_i(x)$ and $p_{i+1}(x)$ be successive two elements of PRS of $f(x)$ and $g(x)$, and $r(x)$ be an approximate GCD of $f(x)$ and $g(x)$. We have

$$f(x)+\delta_f(x) = f_1(x)r(x),\ g(x)+\delta_g(x) = g_1(x)r(x),$$
$$u_i(x)f(x)+v_i(x)g(x)=p_i(x),\ u_{i+1}(x)f(x)+v_{i+1}(x)g(x)=p_{i+1}(x)$$

where the coefficients of $u_i(x), v_i(x), u_{i+1}(x), v_{i+1}(x)$ are some column vectors of the orthogonal matrix Q of the QR decomposition of $Syl(f,g)$ hence they have the unit 2-norm by the orthogonality of Q. Then $r(x)$ is an approximate GCD of $p_i(x)$ and $p_{i+1}(x)$ of tolerance $(\|\delta_f\|_2 + \|\delta_g\|_2)/\min\{\|p_i\|_2, \|p_{i+1}\|_2\}$ since we have

$$p_i(x)=(u_i(x)f_1(x)+v_i(x)g_1(x))r(x)-(u_i(x)\delta_f(x)+v_i(x)\delta_g(x)),$$
$$p_{i+1}(x)=(u_{i+1}(x)f_1(x)+v_{i+1}(x)g_1(x))r(x)-(u_{i+1}(x)\delta_f(x)+v_{i+1}(x)\delta_g(x)).$$

Therefore, we should split $p_i(x)$ and $p_{i+1}(x)$ whose norms are not small (e.g. $\|p_i\|_2, \|p_{i+1}\|_2 \geq 1.0$) since there is a chance that $p_i(x)$ and $p_{i+1}(x)$ have (approximately) common roots other than that of $f(x)$ and $g(x)$ if their norms are small (i.e. tolerance for $p_i(x)$ and $p_{i+1}(x)$ becomes large).

3.5 The Fail-Safe Retry Loop

By the same reason of the section 3.3, we should try to detect a piece of approximate factors which is enough close to the ideal factors. Therefore, our algorithm seeks such a factor by computing from the normal and reversal sides several times. At the worst case, this makes the computational cost k (the degree of approximate GCD) times larger than that of the original, hence our algorithm unfortunately is not faster than the original. However, this is an inevitable cost proven by Theorem 2 and ExQRGCD becomes more stable than the original as in the following examples. Note that in our implementation, the first side of this retry loop is the normal side if $\min\{|f_m|, |g_n|\} \geq \min\{|f_0|, |g_0|\}$.

4 Numerical Experiments

All the computations in this section are done by Maple 16 with $Digits := 16$ on Linux (Intel Core i7 3.30GHz and 64GB memory).

4.1 Against SNAP and QRGCD

At first, we compare ExQRGCD with QRGCD: our algorithm (Algorithm 2) and the SNAP implementation of QRGCD (Algorithm 1), respectively. The vertical axes of Figure 1 and 2 denote the sum of detected degrees of approximate GCD (larger is better) and the magnitude of required perturbation in the common logarithmic scale (smaller is better), respectively. We note that the official SNAP implementation increases the precision in case of the difficult case. However in our experiments we omit this functionality[6] to make the condition equal.

Example 2 (Random Polynomials).
For $i = 1, \ldots, 10$, we have generated 100 pairs of (f, g) such that

$$f(x) = f_1(x)d(x),\ g(x) = g_1(x)d(x),\ d(x) = \sum_{j=0}^{5i} d_j x^j,$$
$$f_1(x) = \sum_{j=0}^{5i} f_{1,j} x^j,\ g_1(x) = \sum_{j=0}^{5i} g_{1,j} x^j$$

where $f_{1,j},\ g_{1,j},\ d_j \in [-99, 99] \subset \mathbb{Z}$ is randomly chosen, $f(x), g(x)$ are normalized ($\|f(x)\|_2 = \|g(x)\|_2 = 1$) and rounded with $Digits := 10$. We computed with tolerance 10^{-5}.

Figure 1 shows the result that ExQRGCD and QRGCD are not so different. However, we note that QRGCD outputs "failure" or GCDs that do not satisfy the given tolerance as in Table 1 (ExQRGCD does not have this problem and always can work well). As for computing time, ExQRGCD is 1.59 times slower than QRGCD. The average of resulting perturbations of QRGCD is better than ExQRGCD except failure cases. ◁

[6] With this functionality, the QRGCD and ExQRGCD algorithms become to be much better since the QR decomposition becomes to be more stable.

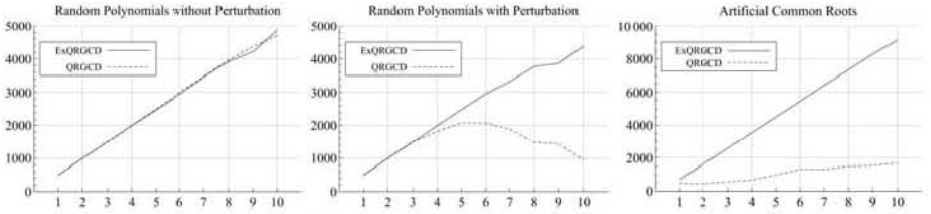

Fig. 1. Sum of Detected Degrees (failure is counted as 0)

Example 3 (Random Poly. with Perturbations).
For $i = 1, \ldots, 10$, we have generated 100 pairs of (f, g) such that

$$f(x) = f_1(x)d(x)/\ \|f_1 d\|_2 + 10^{-8}\Delta_f(x)/\ \|\Delta_f\|_2,$$
$$g(x) = g_1(x)d(x)/\ \|g_1 d\|_2 + 10^{-8}\Delta_g(x)/\ \|\Delta_g\|_2,$$
$$\Delta_f(x) = \textstyle\sum_{j=0}^{10i} \Delta_{f_j} x^j,\ \Delta_g(x) = \sum_{j=0}^{10i} \Delta_{g_j} x^j$$

where $\Delta_{f_j},\ \Delta_{g_j} \in [-99, 99] \subset \mathbb{Z}$ is randomly chosen, $f_1(x)$, $g_1(x)$, $d(x)$ are the polynomials of Example 2 and rounded with $Digits := 10$. We computed with tolerance 10^{-5}.

Figure 1 and Table 1 show the result that ExQRGCD is explicitly better than QRGCD. As for computing time, ExQRGCD is 1.99 times slower than QRGCD. The average of resulting perturbations of QRGCD is better than ExQRGCD except failure cases. ◁

Example 4 (Fake Common Roots).
For $i = 1, \ldots, 10$, we have generated 100 pairs of (f, g) such that

$$f(x) = d(x) \textstyle\prod_{j=1}^{2i}(x - \omega_{f,j}) \prod_{j=1}^{2i}(x - \hat{\omega}_{f,j}),$$
$$g(x) = d(x) \textstyle\prod_{j=1}^{2i}(x - \omega_{g,j}) \prod_{j=1}^{2i}(x - \hat{\omega}_{g,j}),$$
$$d(x) = \textstyle\prod_{j=1}^{3i}(x - \omega_{d,j}) \prod_{j=1}^{3i}(x - \hat{\omega}_{d,j})$$

where $\omega_{\cdot,j} = O(10^{-2}), \hat{\omega}_{\cdot,j} = O(10^2)$ is randomly chosen, $f(x), g(x)$ are normalized (i.e. $\|f(x)\|_2 = \|g(x)\|_2 = 1$) and rounded with $Digits := 10$. We computed with tolerance 10^{-5}. We note that the degree of approximate GCD should be larger than or equal to $6i$.

Figure 1 and Table 1 show the result that ExQRGCD is explicitly better than QRGCD. As for computing time, ExQRGCD is 39.8 times slower than QRGCD (note that QRGCD outputs failure for 62% pairs so computing time is very fast for the rest easy cases). The average of resulting perturbations of ExQRGCD is better than QRGCD. ◁

Other than the above, we have tested with several examples of polynomials of higher degree (up to 1020) which are not listed here due to the page restriction. Our algorithm also works for such polynomials. For some of examples we have tested by the native version (C with ATLAS, LAPACK and LAPACKE) since it is about 100 times faster or more.

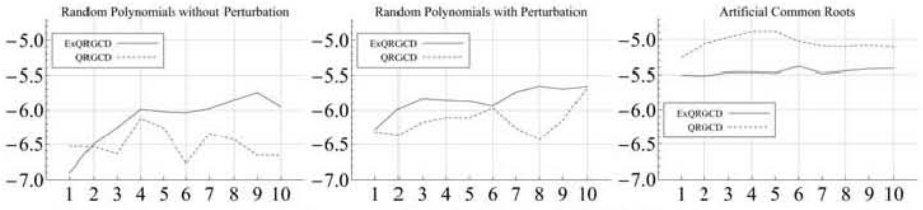

Fig. 2. Resulting Perturbations (i.e. $(\|\Delta_f\|_2 + \|\Delta_g\|_2)/2$) (failure is excepted)

	$i=1$	2	3	4	5	6	7	8	9	10	Δ_f	Δ_g
Random Polys. w/o Perturbation	0	0	0	0	0	0	1	1	3	6	6	6
Random Polys. with Perturbation	0	0	0	9	17	31	46	63	68	81	6	6
Fake Common Roots	29	63	68	71	64	60	66	66	69	68	110	119

Table 1. Number of fail events or wrong perturbations by QRGCD (The numbers by ExQRGCD are 0)

4.2 Against Fastgcd and UVGCD

In this subsection, we compare ExQRGCD with Fastgcd and UVGCD by examples given by Bini and Boito[6, 3]. We note that the results of Fastgcd and UVGCD are quoted from there hence the following comparisons are just subsidiary data since the conditions may not be equal.

Example 5 (Mignotte-like polynomials).
Let the input polynomials be $f(x) = x^{100} + (x - 1/2)^{17}$ and $g(x) = f'(x)$. We compute with tolerance $\varepsilon = 10^{-1}, \ldots, 10^{-12}$. Table 2 shows the detected degrees. ExQRGCD is almost better than others. We note that this is Example 8.2.2 [3] and ExQRGCD is not better than Fastgcd for the polynomials in Example 8.2.1 [3] but same as UVGCD. ◁

Example 6 (An ill-conditioned case).
Let n be an even positive integer and $k = n/2$; define $f(x) = f_1(x)d(x)$ and

ε	ExQRGCD	ε	Fastgcd	ε	UVGCD
10^{-1}	99	10^{-1}	99	$10^{-1} \ldots 10^{-2}$	99
10^{-2}	25	$10^{-2} \ldots 10^{-5}$	17	$10^{-3} \ldots 10^{-5}$	17
10^{-3}	23	$10^{-6} \ldots 10^{-7}$	6	$10^{-6} \ldots 10^{-11}$	16
10^{-4}	20	$10^{-8} \ldots 10^{-9}$	4	10^{-12}	0
$10^{-5} \ldots 10^{-10}$	16	10^{-10}	3		
10^{-11}	0	10^{-11}	0		

Table 2. Detected degrees (Mignotte-like polys.)

n	ExQRGCD	Fastgcd	UVGCD
12	2.86×10^{-12}	1.65×10^{-14}	9.99×10^{-15}
14	2.67×10^{-11}	4.81×10^{-14}	3.66×10^{-14}
16	9.31×10^{-10}	2.27×10^{-13}	1.54×10^{-13}
18	4.38×10^{-9}	1.08×10^{-12}	5.21×10^{-13}
20	1.22×10^{-7}	(detected degree fails)	1.59×10^{-12}

Table 3. Resulting Perturbations (ill-conditioned case)

$g(x) = g_1(x)d(x)$ where

$$d(x) = \prod_{j=1}^{k}((x - r_1\alpha_j)^2 + r_1^2\beta_j^2),\ r_1 = 0.5,\ r_2 = 1.5,$$
$$f_1(x) = \prod_{j=1}^{k}((x - r_2\alpha_j)^2 + r_2^2\beta_j^2),\ \alpha_j = \cos(j\pi/n),$$
$$g_1(x) = \prod_{j=k+1}^{n}((x - r_1\alpha_j)^2 + r_1^2\beta_j^2),\ \beta_j = \sin(j\pi/n).$$

Table 3 shows the resulting perturbations in the 2-norm (i.e. $\sqrt{\|\Delta_f\|_2^2 + \|\Delta_g\|_2^2}$). ExQRGCD is not good though it detected the correct degree of approximate GCD. ◁

Other than the above, we have tested several examples in Boito[3]. Although ExQRGCD is more competitive than the original QRGCD algorithm, it is not better than Fastgcd and UVGCD for most of those examples. Moreover, ExQRGCD requires higher precision (e.g. $Digits := 24$ or 32) to make it competitive against them for those examples. ExQRGCD and QRGCD do not refine the resulting factor hence the resulting perturbation may become larger than those algorithms with refinement steps.

5 Concluding Remarks

In this paper, we improved the QRGCD algorithm from the different approach and our algorithm works as same as the original for polynomials without perturbations and much better than the original for polynomials with perturbations or having fake common roots. We again note that ExQRGCD does not refine the output hence it is notable that ExQRGCD is almost better than Fastgcd and UVGCD for Mignotte-like polynomials in Example 5.

We note that our preliminary implementations on Maple and written in C, and generated polynomial data are available at our website:

`http://wwwmain.h.kobe-u.ac.jp/~nagasaka/research/snap/exqrgcd/`

though some routines derived from the SNAP package are not included.

Acknowledgments

The authors would like to thank an anonymous reviewer for his/her many constructive comments that are very helpful to improve the manuscript.

References

1. Roy, M.F., Sedjelmaci, S.M.: New fast euclidean algorithms. J. Symbolic Comput. **50** (2013) 208–226
2. von zur Gathen, J., Gerhard, J.: Modern computer algebra. Second edn. Cambridge University Press, Cambridge (2003)
3. Boito, P.: Structured Matrix Based Methods for Approximate GCD. Ph.D. Thesis. Department of Mathematics, University of Pisa, Italia (2007)
4. Zeng, Z.: The numerical greatest common divisor of univariate polynomials. In: Randomization, relaxation, and complexity in polynomial equation solving. Volume 556 of Contemp. Math. Amer. Math. Soc., Providence, RI (2011) 187–217
5. Kaltofen, E., Yang, Z., Zhi, L.: Approximate greatest common divisors of several polynomials with linearly constrained coefficients and singular polynomials. In: ISSAC 2006: Proceedings of the 2006 international symposium on Symbolic and algebraic computation. (2006) 169–176
6. Bini, D.A., Boito, P.: A fast algorithm for approximate polynomial GCD based on structured matrix computations. In: Numerical methods for structured matrices and applications. Volume 199 of Oper. Theory Adv. Appl. Birkhäuser Verlag, Basel (2010) 155–173
7. Terui, A.: An iterative method for calculating approximate GCD of univariate polynomials. In: ISSAC 2009: Proceedings of the 2009 international symposium on Symbolic and algebraic computation. (2009) 351–358
8. Corless, R.M., Watt, S.M., Zhi, L.: QR factoring to compute the GCD of univariate approximate polynomials. IEEE Trans. Signal Process. **52**(12) (2004) 3394–3402
9. Laidacker, M.A.: Another theorem relating Sylvester's matrix and the greatest common divisor. Math. Mag. **42** (1969) 126–128
10. Golub, G.H., Van Loan, C.F.: Matrix computations. Third edn. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD (1996)
11. Stetter, H.J.: The nearest polynomial with a given zero, and similar problems. SIGSAM Bull. **33**(4) (December 1999) 2–4
12. Rezvani, N., Corless, R.M.: The nearest polynomial with a given zero, revisited. SIGSAM Bull. **39**(3) (2005) 73–79
13. Brown, W.S., Traub, J.F.: On Euclid's algorithm and the theory of subresultants. J. Assoc. Comput. Mach. **18** (1971) 505–514
14. D'Andrea, C., Hong, H., Krick, T., Szanto, A.: An elementary proof of Sylvester's double sums for subresultants. J. Symbolic Comput. **42**(3) (2007) 290–297
15. Sasaki, T.: The subresultant and clusters of close roots. In: ISSAC 2003: Proceedings of the 2003 international symposium on Symbolic and algebraic computation. (2003) 232–239