Approximate GCD by relaxed NewtonSLRA algorithm

Approximate GCD by relaxed NewtonSLRA algorithm

Kosaku Nagasaka
Kobe University, Japan
`nagasaka@main.h.kobe-u.ac.jp`

Abstract

We propose a better algorithm for approximate greatest common divisor (approximate GCD) of univariate polynomials in terms of robustness and distance, based on the NewtonSLRA algorithm that is a solver for the structured low rank approximation (SLRA) problem. Our algorithm mainly enlarges the tangent space in the NewtonSLRA algorithm and adapts it to a certain weighted Frobenius norm. Moreover, we propose some improvement in computing time.

1 Introduction

Approximate GCD is one of the classical and important problems in symbolic-numeric algorithms, which finds a non-trivial GCD of polynomials in a certain neighborhood of the given polynomials. For example, approximate GCD is used for a nearby non-trivial Smith form of a matrix polynomial[5], applications in signal processing[7], rational function approximation [1] and so on. For further information, see [2, 4, 10] for fundamental references of approximate univariate GCD.

In this short communication, we propose a better algorithm in terms of robustness and distance, based on the NewtonSLRA algorithm by Schost and Spaenlehauer[9] which is a solver for the SLRA problem. Our algorithm mainly enlarges the tangent space (i.e. orthogonality is relaxed) in the NewtonSLRA algorithm and adapts it to a certain column weighted Frobenius norm. By this improvement, we try to prevent a convergence to a local optimum that is possibly far from the global optimum. We also propose some modification using a sparsity on the NewtonSLRA algorithm for the subresultant matrix in terms of computing time (please note that the NewtonSLRA algorithm is not specific to approximate GCD).

1.1 definitions and notations

Let the given polynomials be $f(x) = \sum_{i=0}^{m} f_i x^i,\ g(x) = \sum_{i=0}^{n} g_i x^i \in \mathbb{R}[x]$ with the assumption $m \geq n$. We denote the set of matrices of size $p \times q$ by $\mathbb{R}^{p\times q}$ which is endowed with the inner product $\langle M_1, M_2\rangle = \mathrm{trace}(M_1 M_2^T)$ where M^T is the transpose of matrix M, and the set of matrices of rank $r \in \mathbb{N}$ by $\mathcal{D}_r \subset \mathbb{R}^{p\times q}$. For an affine subspace $\mathcal{S} \subset \mathbb{R}^{p\times q}$, we denote its underlying vector space by $\mathcal{S}^0$, its tangent space at M by $T_M\mathcal{S}$, and the orthogonal projection of M on $\mathcal{S}$ by $\Pi_{\mathcal{S}}(M)$. For vectors and matrices, $\|\cdot\|_2$ and $\|\cdot\|_F$ denote the Euclidean norm (2-norm) and the Frobenius norm (deduced from the inner product above), respectively. By this Frobenius norm, we denote the open and closed balls centered at $M \in \mathbb{R}^{p\times q}$ and of radius ρ by $\mathcal{B}_\rho(M), \overline{\mathcal{B}_\rho(M)} \subset \mathbb{R}^{p\times q}$, respectively. M^{-1} and $M^\dagger$ are the inverse and Moore-Penrose pseudo-inverse of matrix M, respectively. We define the transversally condition as follows.

Definition 1 (transversally condition). *Let $\mathcal{S}$ be an affine subspace of $\mathbb{R}^{p\times q}$. We say that $\mathcal{S}$ and $\mathcal{D}_r$ intersect transversally at $\zeta \in \mathcal{S}\cap\mathcal{D}_r$ if* $\mathrm{codim}(\mathcal{S}^0 \cap T_\zeta\mathcal{D}_r^0) = \mathrm{codim}(\mathcal{S}^0) + \mathrm{codim}(T_\zeta\mathcal{D}_r^0)$ *where* codim *denotes the codimension in $\mathbb{R}^{p\times q}$.*

The coefficient vector (in the descending term order) of $p(x) \in \mathbb{R}[x]$ of degree k is denoted by $\vec{p} \in \mathbb{R}^{k+1}$. We denote the polynomial norm of $p(x)$ by $\|p\|_2$ that is defined as $\|p\|_2 = \|\vec{p}\|_2$. For a pair $(f, g) \in \mathbb{R}[x]\times\mathbb{R}[x]$, we define its norm as $\|(f, g)\|_2 = \sqrt{\|f\|_2^2 + \|g\|_2^2}$.

1.2 problem description and SLRA

Although there are so many definitions for approximate GCD, we focus on the following widely used one, and note that this is often used in a combination with degree search strategies (cf. [10, Section 1][8]).

Definition 2 (approximate GCD)**.** *For $k \in \mathbb{N}$, we compute the polynomial $d(x) \in \mathbb{R}[x]$ called "approximate GCD" of degree k, which minimizes $\|(\Delta_f, \Delta_g)\|_2$ (called* ***perturbation****) and satisfies*

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

for some polynomials $\Delta_f(x), \Delta_g(x), f_1(x), g_1(x) \in \mathbb{R}[x]$ such that $\deg(\Delta_f) \le \deg(f),\ \deg(\Delta_g) \le \deg(g)$.

We denote the subresultant matrix of order r of $f(x)$ and $g(x)$ by $\mathrm{Syl}_r(f, g) \in \mathbb{R}^{(m+n-r)\times(m+n-2r)}$.

Fact 1. *For the largest integer r such that $Syl_r(f, g)$ is not column full rank, let $\vec{v}$ be a non-zero vector in the null space of $Syl_r(f, g)$ and $g_1(x), f_1(x) \in \mathbb{R}[x]$ be polynomials whose coefficient vectors are the first $n - r$ elements and the last $m - r$ elements of $\vec{v}$, respectively. Then, $f(x)/f_1(x)$ and $-g(x)/g_1(x)$ are the polynomial GCD of $f(x)$ and $g(x)$, and their degree is $r + 1$.*

By this well-known property, the approximate GCD can be considered as the following optimization.

$$\min_{\Delta_f, \Delta_g} \|(\Delta_f, \Delta_g)\|_2 \quad \text{subject to} \ \mathrm{rank}(\mathrm{Syl}_{k-1}(f + \Delta_f, g + \Delta_g)) < m + n - 2k + 2. \tag{1.1}$$

This means that we want to compute a nearby low rank matrix of $\mathrm{Syl}_{k-1}(f+\Delta_f, g+\Delta_g)$ that is structured. Therefore, the approximate GCD problem is reduced to the structured low rank approximation (SLRA) problem. We note that computing the global optimum for this problem is not easy since it is non-convex due to the rank constraint (see also [6] for further references in low rank approximation). Hence most of algorithms (including ours) will compute a local optimum or some enough small perturbation. In this short communication, following Schost and Spaenlehauer[9], we define the SLRA problem as follows.

Definition 3 (structured low rank approximation, SLRA)**.** *Let $\mathcal{S} \subset \mathbb{R}^{p\times q}$ be an affine subspace of $\mathbb{R}^{p\times q}$ (i.e. this defines the structure of matrices). For the given $M \in \mathcal{S}$ and $r \in \mathbb{N}$, find a matrix $M^* \in \mathcal{S} \cap \mathcal{D}_r$ such that $\|M - M^*\|_F$ is "small".*

For the SLRA problem, one of the first iterative methods (the convergence ratio is linear) is proposed by Cadzow[3] and is based on alternating projections. The first quadratic convergent method is the NewtonSLRA algorithm[9]. Actually the approximate GCD is one of applications of the algorithm in their paper. Following them, we use the NewtonSLRA algorithm for computing approximate GCD.

Let $\mathcal{S}_{k-1}(f, g)$ be the set of subresultant matrices of order $k-1$ of $f(x)+\Delta_f(x)$ and $g(x)+\Delta_g(x)$ whose degrees are less than or equal to m and n, respectively. This set plays the set of structured matrices and is $\mathcal{S}$ in Definition 3. As a consequence of the discussion above we have Algorithm 1. We note that there are several ways to extract $d(x)$ from the subresultant matrix $\mathrm{Syl}_{k-1}(f + \Delta_f, g + \Delta_g)$ on the line 4. For example, at first we extract cofactors $f_1(x)$ and $g_1(x)$ by computing the null space of $\mathrm{Syl}_{k-1}(f+\Delta_f, g+\Delta_g)$ by Fact 1 and approximate GCD $d(x)$ can be computed by the least squares with the convolution matrices of $f_1(x)$ and $g_1(x)$, unknown vector $\vec{d}$ and the right hand side constant vector $(\vec{f}^T, \vec{g}^T)^T$.

1.3 NewtonSLRA algorithm

We briefly review the NewtonSLRA algorithm as Algorithm 2. It basically follows the alternating projection method by Cadzow[3] that is formed by the following two steps: 1) lifting M_i up to the desired rank matrix in $\mathcal{D}_r$ by the well known Eckart-Young theorem, 2) projecting it back to the structured matrix in $\mathcal{S}$ by the orthogonal projection on $\mathcal{S}$. The NewtonSLRA/1 algorithm also has the following similar two steps: 1) the lifting step is exactly the same as the alternating projection method, 2) projecting it back to the structured matrix in $\mathcal{S} \cap T_{\tilde{M}}\mathcal{D}_r$ by the orthogonal projection on $\mathcal{S} \cap T_{\tilde{M}}\mathcal{D}_r$. This difference makes the NewtonSLRA algorithm being *locally quadratic convergent.*

Algorithm 1 approximate GCD by SLRA

```
Input: $f(x), g(x) \in \mathbb{R}[x]$ and $k \in \mathbb{N}$
Output: $d(x) \in \mathbb{R}[x]$, approximate GCD of degree $k$
1: construct a SLRA problem in Definition 3 with $M = \mathrm{Syl}_{k-1}(f, g)$, $\mathcal{S} = \mathcal{S}_{k-1}(f, g)$ and $r = m+n-2k+1$;
2: solve the SLRA problem and let $M^*$ be the resulting matrix;
3: construct $f(x) + \Delta_f(x)$ and $g(x) + \Delta_g(x)$ back from $M^*$;
4: compute $d(x)$ from $\mathrm{Syl}_{k-1}(f + \Delta_f, g + \Delta_g)$;
5: return $d(x)$;
```

Algorithm 2 one iteration of NewtonSLRA/1[9]

```
Input: $M \in \mathcal{S} \subset \mathbb{R}^{p\times q}$, $r \in \mathbb{N}$ and $\{B_1, \ldots, B_d\}$: an orthonormal basis of $\mathcal{S}^0$
Output: $\Pi_{\mathcal{S}\cap T_{\tilde{M}}\mathcal{D}_r}(M)$ where $\tilde{M} = \Pi_{\mathcal{D}_r}(M)$
1: $U\Sigma V^T :=$ the singular value decomposition of $M$ and let $U = (\overrightarrow{u_1}, \ldots, \overrightarrow{u_p})$ and $V = (\overrightarrow{v_1}, \ldots, \overrightarrow{v_q})$;
2: $\tilde{M} := U_r\Sigma_r V_r^T$ where $U_r, V_r$ are the first $r$ columns of $U, V$, and $\Sigma_r$ is the top-left $r\times r$ sub-matrix of $\Sigma$;
3: for $i \in \{1, \ldots, p-r\}$, $j \in \{1, \ldots, q-r\}$ do $N_{(i-1)(q-r)+j} := \overrightarrow{u_{r+i}}\ \overrightarrow{v_{r+j}}^T$;
4: $A := (a_{k,\ell}) \in \mathbb{R}^{(p-r)(q-r)\times d}$, $a_{k,\ell} = \langle N_k, B_\ell\rangle$, $\vec{b} := (b_k) \in \mathbb{R}^{(p-r)(q-r)}$, $b_k = \langle N_k, \tilde{M} - M\rangle$;
5: return $M + \sum_{\ell=1}^{d}(A^\dagger\vec{b})_\ell B_\ell$;
```

2 Improvements

We consider to compute an approximate GCD of degree k, of $f(x)$ and $g(x)$, by the NewtonSLRA/1 algorithm. In this case, the given orthonormal basis of $\mathcal{S}_{k-1}(f, g)$ will be made from

$$\{\bar{B}_1, \ldots, \bar{B}_d\} = \left\{\overline{\mathrm{Syl}_{k-1}(f,g)}^{f_m}, \ldots, \overline{\mathrm{Syl}_{k-1}(f,g)}^{f_0}, \overline{\mathrm{Syl}_{k-1}(f,g)}^{g_n}, \ldots, \overline{\mathrm{Syl}_{k-1}(f,g)}^{g_0}\right\} \tag{2.1}$$

where $\overline{\mathrm{Syl}_{k-1}(f,g)}^{h}$ denotes $\mathrm{Syl}_{k-1}(f, g)$ of symbolic $f(x)$ and $g(x)$ with the substitution $f_m = \cdots = f_0 = g_n = \cdots = g_0 = 0$ but $h = 1$, such that $\mathrm{Syl}_{k-1}(f, g) = f_m\bar{B}_1 + \cdots + f_0\bar{B}_{m+1} + g_n\bar{B}_{m+2} + \cdots + g_0\bar{B}_d$. Therefore, its orthonormalized basis is given by $\{B_1, \ldots, B_d\} = \{\eta_1^{-1}\bar{B}_1, \ldots, \eta_{m+1}^{-1}\bar{B}_{m+1}, \eta_{m+2}^{-1}\bar{B}_{m+2}, \ldots, \eta_d^{-1}\bar{B}_d\}$ where $\eta_i = \sqrt{n-k+1}$ $(i \le m+1)$ and $\eta_i = \sqrt{m-k+1}$ $(m+2 \le i)$. We note that removing some of B_is fixes a priori corresponding coefficients (e.g. this can be used to keep polynomials monic).

In the NewtonSLRA algorithm, we compute the matrix A with the elements $a_{(i-1)(q-r)+j,\ell} = \overrightarrow{u_i}^T B_\ell \overrightarrow{v_j}$, and the complexity of each element is $O(pq)$ for $\overrightarrow{u_i}^T B_\ell$ and $O(q)$ for $(\overrightarrow{u_i}^T B_\ell)\overrightarrow{v_j}$. However, since we have $B_i = \eta_i^{-1}\bar{B}_i$ and $\bar{B}$ is a sparse matrix (i.e. only $n-k+1$ or $m-k+1$ non-zero elements), $\overrightarrow{u_i}^T B_\ell$ can be computed in $O(m)$ by the following formula where $\overrightarrow{u_i}^T = (u_{i1}\ \cdots\ u_{ip})$.

$$\begin{cases} \overrightarrow{u_i}^T B_\ell = (u_{i,\ell}\eta_\ell^{-1}\ \cdots\ u_{i,\ell+n-k}\eta_\ell^{-1}\ 0\ \cdots\ 0) & (\ell \le m+1) \\ \overrightarrow{u_i}^T B_\ell = (0\ \cdots\ 0\ u_{i,\ell-m-1}\eta_\ell^{-1}\ \cdots\ u_{i,\ell-k-1}\eta_\ell^{-1}) & (m+2 \le \ell) \end{cases}.$$

As a consequence, computing A and $\vec{b}$ is done in $O(m^2k)$, computing $\tilde{M}$ and updating M are done in $O((m+n)(m+n-k)(m+n-2k))$, and computing the Moore-Penrose pseudo-inverse is done in $O((m+n)k^2)$. Therefore, with our modification, one iteration of the NewtonSLRA/1 algorithm for computing approximate GCD requires the singular value decomposition and $O(m^3)$ arithmetic operations that is reduced from $O(m^3k)$ (i.e. our revised version is k **times faster** than the original one).

2.1 relaxed NewtonSLRA algorithm

In this subsection, we assume $p > q$ and $r = q-1$ that mean the case specific to the subresultant matrix for computing non-linear approximate GCD. According to our experiments, approximate GCDs computed

Algorithm 3 one iteration of relaxed NewtonSLRA

```
Input: $M \in \mathcal{S} \subset \mathbb{R}^{p\times q}$, $r \in \mathbb{N}$ (s.t. $p > q$ and $r = q-1$) and $\{B_1, \dots, B_d\}$: an orthonormal basis of $\mathcal{S}^0$
Output: $\Pi_{\mathcal{S}\cap\mathcal{T}_{M,\tilde{M}}\mathcal{D}_r}(M)$ where $\tilde{M} = \Pi_{\mathcal{D}_r}(M)$
1: $U\Sigma V^T :=$ the singular value decomposition of $M$ and let $U = (\overrightarrow{u_1}, \dots, \overrightarrow{u_p})$ and $V = (\overrightarrow{v_1}, \dots, \overrightarrow{v_q})$;
2: $\tilde{M} := U_r\Sigma_r V_r^T$ as in Algorithm 2;
3: $A := (a_{1,\ell}) \in \mathbb{R}^{1\times d}$, $a_{1,\ell} = \langle \overrightarrow{u_{r+1}}\overrightarrow{v_q}^T, B_\ell\rangle$, $\vec{b} := (b_1) \in \mathbb{R}^1$, $b_1 = \langle \overrightarrow{u_{r+1}}\overrightarrow{v_q}^T, \tilde{M} - M\rangle$;
4: return $M + \sum_{\ell=1}^d (A^\dagger \vec{b})_\ell B_\ell$;
```

by the NewtonSLRA algorithm sometimes have $O(1)$ perturbations even if the expected value is 1.0e-2 or so. To overcome this issue, we enlarge the tangent space and tries to prevent a local limit point from being far from the global optimum. Let $U\Sigma V^T$ be the singular value decomposition of M, and let U and V be $(\overrightarrow{u_1} \ \cdots \ \overrightarrow{u_p})$ and $(\overrightarrow{v_1} \ \cdots \ \overrightarrow{v_q})$, respectively. In the NewtonSLRA/1 algorithm, the tangent space of $\mathcal{D}_r^0$ at M and its normal space $N_M\mathcal{D}_r^0$ are given as follows.

$$T_M\mathcal{D}_r^0 = \mathrm{Im}(M) \otimes \mathbb{R}^q + \mathbb{R}^p \otimes \mathrm{Ker}(M)^\perp,\ N_M\mathcal{D}_r^0 = \mathrm{Ker}(M^T) \otimes \mathrm{Ker}(M).$$

Moreover, we have that $\{\overrightarrow{u_1}, \dots, \overrightarrow{u_r}\}$, $\{\overrightarrow{v_1}, \dots, \overrightarrow{v_r}\}$, $\{\overrightarrow{u_{r+1}}, \dots, \overrightarrow{u_p}\}$ and $\{\overrightarrow{v_{r+1}}, \dots, \overrightarrow{v_q}\}$ ($= \{\overrightarrow{v_q}\}$ since $r = q-1$ by the assumption) are bases of $\mathrm{Im}(M)$, $\mathrm{Ker}(M)^\perp$, $\mathrm{Ker}(M^T)$ and $\mathrm{Ker}(M)$, respectively.

Instead of these tangent and normal spaces, we define the following relaxed tangent space $\mathcal{T}_{O,M}\mathcal{D}_r^0$ and relaxed normal space $\mathcal{N}_{O,M}\mathcal{D}_r^0$ at M from O such that $M - O$ is orthogonal to $T_M\mathcal{D}_r^0$ and $M \neq O$.

$$\mathcal{T}_{O,M}\mathcal{D}_r^0 = (\mathcal{N}_{O,M}\mathcal{D}_r^0)^\perp,\ \mathcal{N}_{O,M}\mathcal{D}_r^0 = \mathrm{span}(M - O).$$

For $O \notin \mathcal{D}_r$ and $M = \Pi_{\mathcal{D}_r}(O)$, by the non-linear approximate GCD assumption, we have

$$\mathcal{T}_{O,M}\mathcal{D}_r^0 = \mathrm{span}(\overrightarrow{u_1}, \dots, \overrightarrow{u_r}, \overrightarrow{u_{r+2}}, \dots, \overrightarrow{u_p}) \otimes \mathbb{R}^q + \mathbb{R}^p \otimes \mathrm{Ker}(M)^\perp,\ \mathcal{N}_{O,M}\mathcal{D}_r^0 = \mathrm{span}(\overrightarrow{u_{r+1}}) \otimes \mathrm{Ker}(M).$$

As a consequence of these modifications above, we propose the relaxed NewtonSLRA algorithm as Algorithm 3 and it also has the following **local quadratic convergence** property.

Theorem 1 (convergence property). *Let ζ be in $\mathcal{S} \cap \mathcal{D}_r$ such that $\Pi_{\mathcal{D}_r}$ is C^2 around ζ and $\mathcal{S}$ and $\mathcal{D}_r$ intersect transversally at ζ. There exist $\nu, \gamma, \gamma' > 0$ such that, for all M_0 in $\mathcal{S} \cap \mathcal{B}_\nu(\zeta)$, the sequence (M_i) given by $M_{i+1} = \varphi(M_i)$ is well defined and converges toward a matrix $M_\infty \in W$, and we have $\| M_{i+1} - M_\infty \|_F \leq \gamma \| M_i - M_\infty \|_F^2$ for all $i \geq 0$ and $\| \Pi_W(M_0) - M_\infty \|_F \leq \gamma' \| \Pi_W(M_0) - M_0 \|_F^2$ where φ is Algorithm 3 as a mapping, U is an open neighborhood of $\mathcal{D}_r$ at ζ and $W = \mathcal{S} \cap \mathcal{D}_r \cap U$.*

Moreover, according to our numerical experiments, our relaxed NewtonSLRA algorithm can find approximate GCDs with **smaller perturbations** than the original NewtonSLRA/1 algorithm.

2.2 column weighted Frobenius norm

In the both of original and relaxed NewtonSLRA algorithms, let the solution of least squares on the last line be $\vec{h}$ hence $\vec{h} = A^\dagger \vec{b}$. Consider that this iteration is for computing approximate GCD of $f(x)$ and $g(x)$, we have the perturbed polynomials $f(x) + \Delta_f(x)$ and $g(x) + \Delta_g(x)$ as follows.

$$(\ \overrightarrow{f + \Delta_f} \ \ \overrightarrow{g + \Delta_g}\ )^T = (\ \vec{f} \ \ \vec{g}\ )^T + \mathrm{diag}(\eta_1^{-1}, \dots, \eta_d^{-1})\vec{h}.$$

Each iteration minimizes the Frobenius norm of the correction matrix $\sum_{\ell=1}^d (A^\dagger \vec{b})_\ell B_\ell$ and its value is $\| \vec{h} \|_2$ since $\{B_1, \dots, B_d\}$ is an orthonormal basis of $\mathcal{S}^0$. Actually each iteration computes the nearest intersection point between two subspaces in the Frobenius norm. However, as for approximate GCD, we have $\| (\Delta_f, \Delta_g) \|_2 = \| \mathrm{diag}(\eta_1^{-1}, \dots, \eta_d^{-1})\vec{h} \|_2$ hence this may not be minimized (i.e. not the nearest point) in the polynomial 2-norm if $m \neq n$ (i.e. $\eta_1 \neq \eta_d$). To make this difference smaller, we introduce the following column weighted inner product $\langle M_1, M_2 \rangle_W$ and norm $\| M \|_W$.

$$\langle M_1, M_2 \rangle_W = \langle M_1 W, M_2 W \rangle = \operatorname{tr}(W^T M_1^T M_2 W),\ \| M \|_W = \sqrt{\langle M, M \rangle_W} = \sqrt{\langle MW, MW \rangle} = \| MW \|_F$$

where W is a diagonal invertible matrix and in this paper we use $W = \operatorname{diag}(\overbrace{\eta_1^{-1}, \ldots, \eta_1^{-1}}^{n-k+1}, \overbrace{\eta_{m+2}^{-1}, \ldots, \eta_{m+2}^{-1}}^{m-k+1})$.

Lemma 1. *Let consider $\mathcal{S}_{k-1}(f, g)$ as an affine subspace of inner product space endowed with $\langle M_1, M_2 \rangle_W$. Then, $\{\bar{B}_1, \ldots, \bar{B}_d\}$ in (2.1) is an orthonormal basis of $\mathcal{S}_{k-1}(f, g)$.*

To make the relaxed NewtonSLRA algorithm being compatible with the column weighted inner product and its Frobenius norm, we give the column weighted Frobenius norm version of the Eckart-Young theorem.

Theorem 2. *For $M \in \mathbb{R}^{p \times q}$ and $r \in \mathbb{N}$, let $U \Sigma V^T$ be the singular value decomposition of MW, and let U_r, V_r be the first r columns of U, V, respectively, and Σ_r be the top-left $r \times r$ sub-matrix of Σ. We have*

$$U_r \Sigma_r V_r^T W^{-1} = \operatorname{argmin}_{M_r \in \mathcal{D}_r} \| M - M_r \|_W .$$

With the above modifications, each iteration becomes to be able to find the nearest intersection of subspaces in the polynomial 2-norm instead of the Frobenius norm. Although this is just for each iteration and it does not guarantee any better resulting perturbation in total, sometimes it can compute better results or can have done faster (unfortunately not always).

Acknowledgement

This work was supported by JSPS KAKENHI Grant Number 19K11827.

References

[1] Juan Gerardo Alcázar and Emily Quintero, *Affine equivalences of trigonometric curves*, Acta Appl. Math. **170** (2020), 691–708.

[2] Paola Boito, *Structured matrix based methods for approximate polynomial GCD*, Tesi. Scuola Normale Superiore di Pisa (Nuova Series) [Theses of Scuola Normale Superiore di Pisa (New Series)], vol. 15, Edizioni della Normale, Pisa, 2011.

[3] James A. Cadzow, *Signal enhancement-a composite property mapping algorithm*, IEEE Trans. Acoust. Speech Signal Process. **36** (1988), no. 1, 49–62.

[4] Antonio Fazzi, Nicola Guglielmi, and Ivan Markovsky, *An ODE-based method for computing the approximate greatest common divisor of polynomials*, Numer. Algorithms **81** (2019), no. 2, 719–740.

[5] Mark Giesbrecht, Joseph Haraldson, and George Labahn, *Computing nearby non-trivial Smith forms*, J. Symbolic Comput. **102** (2021), 304–327.

[6] Ivan Markovsky, *Low-rank approximation: Algorithms, implementation, applications. 2nd ed.*, Communications and Control Engineering Series, Springer, Cham, 2019.

[7] Ivan Markovsky, Antonio Fazzi, and Nicola Guglielmi, *Applications of polynomial common factor computation in signal processing*, Latent Variable Analysis and Signal Separation (Yannick Deville, Sharon Gannot, Russell Mason, Mark D. Plumbley, and Dominic Ward, eds.), Springer, Cham, 2018, pp. 99–106.

[8] Kosaku Nagasaka, *Toward the best algorithm for approximate GCD of univariate polynomials*, J. Symbolic Comput. **105** (2021), 4–27.

[9] Éric Schost and Pierre-Jean Spaenlehauer, *A quadratically convergent algorithm for structured low-rank approximation*, Found. Comput. Math. **16** (2016), no. 2, 457–492.

[10] Konstantin Usevich and Ivan Markovsky, *Variable projection methods for approximate (greatest) common divisor computations*, Theoret. Comput. Sci. **681** (2017), 176–198.