A geometrical approach to finding multivariate approximate LCMs and GCDs

Kim Batselier , Philippe Dreesen, Bart De Moor[1]

Department of Electrical Engineering, ESAT-SCD, KU Leuven / IBBT Future Health Department

Abstract

In this article we present a new approach to compute an approximate least common multiple (LCM) and an approximate greatest common divisor (GCD) of two multivariate polynomials. This approach uses the geometrical notion of principal angles whereas the main computational tools are the Implicitly Restarted Arnoldi Method and sparse QR decomposition. Upper and lower bounds are derived for the largest and smallest singular values of the highly structured Macaulay matrix. This leads to an upper bound on its condition number and an upper bound on the 2-norm of the product of two multivariate polynomials. Numerical examples are provided.

Keywords: Approximate LCM, Approximate GCD, Multivariate Polynomials, Principal angles, Singular values, Arnoldi Iteration
2010 MSC: 65F30, 65F35, 65F50, 15A18

1. Introduction

The computation of the greatest common divisor (GCD) of two polynomials is a basic operation in computer algebra with numerous applications

Email address: {kim.batselier}@esat.kuleuven.be ()

[1]Kim Batselier is a research assistant at the Katholieke Universiteit Leuven, Belgium. Philippe Dreesen is supported by the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen). Bart De Moor is a full professor at the Katholieke Universiteit Leuven, Belgium. Research supported by Research Council KUL: GOA/11/05 AMBioRICS, GOA/10/09 MaNet, CoE EF/05/006 Optimization in Engineering (OPTEC), IOF-SCORES4CHEM, several PhD/postdoc & fellow grants; *Flemish Government*: FWO: FWO: PhD/postdoc grants, projects: G0226.06 (cooperative systems and optimization), G0321.06 (Tensors), G.0302.07 (SVM/Kernel), G.0320.08 (convex MPC), G.0558.08 (Robust MHE), G.0557.08 (Glycemia2), G.0588.09 (Brain-machine) research communities (WOG: ICCoS, ANMMM, MLDM); G.0377.09 (Mechatronics MPC); IWT: PhD Grants, Eureka-Flite+, SBO LeCoPro, SBO Climaqs, SBO POM, O&O-Dsquare; *Belgian Federal Science Policy Office*: IUAP P6/04 (DYSCO, Dynamical systems, control and optimization, 2007-2011); *EU*: ERNSI; FP7-HD-MPC (INFSO-ICT-223854), COST intelliCIS, FP7-EMBOCON (ICT-248940); *Contract Research*: AMINAL; *Other*: Helmholtz: viCERP; ACCM.

[1, 2, 3] and has therefore already received a lot of attention. The most well known algorithm for calculating the GCD of two univariate polynomials symbolically is probably the Euclidean algorithm. Even in computer algebra, the use of linear algebra as an alternative to the Euclidean algorithm has been recognized from an early time. Unfortunately, finding an exact GCD numerically is an ill-posed problem which is easily seen from the fact that a tiny perturbation of the coefficients reduces a nontrivial GCD to a constant. This has sparked research into computing quasi- or approximate GCDs with different formulations [4, 5, 6, 7, 8, 9]. Linear algebra plays an even stronger role in this context, mainly since many numerical methods are based on the singular value decomposition (SVD) of either Sylvester matrices [7, 10] or Bézout matrices [11, 12]. This is because the determination of the degree of the (approximate) GCD corresponds with the detection of a rank-deficiency of the Sylvester/Bézout matrix and the SVD is the most reliable way to determine the numerical rank of a matrix. Finding the least common multiple (LCM) has not received as much attention. Although most GCD-methods based on the Sylvester matrix make use of the LCM implicitly, this is never really mentioned.

In contrast with the usual approach of defining an approximate LCM/GCD this article will introduce a geometrical definition. The main contribution of this article is the presentation of a new numerical algorithm to compute approximate LCMs/GCDs and the derivation of bounds on the largest and smallest singular values of the highly structured Macaulay matrix. These bounds lead to upper bounds on both the condition number of the Macaulay matrix and on the 2-norm of the product of two multivariate polynomials. The proposed algorithm uses the geometrical notion of principal angles between vector spaces [13] and the interrelation of the LCM with the GCD. In addition, the method works for multivariate polynomials whereas in most other methods only the univariate case is considered. Existing literature on numerical SVD-based methods to compute multivariate approximate GCD includes [14, 15]. These methods generalize the univariate SVD-based methods to the multivariate case in the sense that they also detect a numerical rank deficiency of a (multivariate) Sylvester matrix.

The article is structured as follows: in Section 2 a brief overview on the notation is presented. Section 3 introduces the key ingredients, the Macaulay matrix and principal angles, and presents the algorithm to compute an approximate LCM. In Section 4 it is explained how the approximate LCM is used to compute an approximate GCD. Upper and lower bounds are derived

in Section 5 for the largest and smallest singular values of the Macaulay matrix which leads to an upper bound on its condition number. Finally, in Section 6 numerical examples are provided. All algorithms are implemented in MATLAB [16] and are freely available on request.

2. Vector Space of Multivariate Polynomials

The ring of n-variate polynomials over the field of real numbers will be denoted by $\mathcal{R}^n$. Polynomials of $\mathcal{R}^n$ up to a certain degree d together with the addition and scalar multiplication operations form a vector space. This vector space will be denoted by $\mathcal{R}^n_d$. A canonical basis for this vector space consists of all monomials from degree 0 up to d. In order to be able to represent a multivariate polynomial by a vector a monomial ordering needs to be used. One simply orders the coefficients in a row vector according to the chosen monomial ordering. For a formal definition of monomial orderings see [17]. Since the results on the computation of the approximate LCM and GCD do not depend on the monomial ordering we assume a admissible monomial ordering is used. By convention a coefficient vector will always be a row vector. Depending on the context we will use a lowercase character for both a polynomial and its coefficient vector. $(.)^T$ will denote the transpose of the matrix or vector. A polynomial $l \in \mathcal{R}^n$ is called an exact LCM of $f_1, f_2 \in \mathcal{R}^n$ if f_1 divides l and f_2 divides l and l divides any polynomial which both f_1 and f_2 divide. A polynomial $g \in \mathcal{R}^n$ is called an exact GCD of f_1 and f_2 if g divides f_1 and f_2 and if p is any polynomial which divides both f_1 and f_2, then p divides g. The following theorem interrelates the exact LCM of multivariate polynomials with the exact GCD and will be of crucial importance for our method.

Theorem 2.1. *Let $f_1, f_2 \in \mathcal{R}^n$ and l, g their exact LCM and GCD respectively then*

$$l\, g \;=\; f_1\, f_2. \tag{1}$$

Proof. See [17, p. 190] □

This theorem provides a way to find the exact LCM or GCD once either of them has already been found. Approximate LCMs and GCDs are usually defined as the exact LCM and GCD of polynomials $\tilde{f}_1, \tilde{f}_2$ that satisfy $||f_1 - \tilde{f}_1||_2 \leq \epsilon, ||f_2 - \tilde{f}_2||_2 \leq \epsilon$ where ϵ is some user-defined tolerance. In this

article we will define the approximate LCM in Section 3 using a geometrical perspective. The connection with the traditional definition will be made in Section 5. The algorithm we propose will first compute an approximate LCM and derive an approximate GCD as a least-squares solution of (1).

3. Computing the LCM

3.1. The Macaulay Matrix

Finding an approximate LCM will be the first step in the proposed algorithm. As mentioned in the previous section, the GCD will then be computed using Theorem 2.1. Given a multivariate polynomial $f_1 \in \mathcal{R}^n$ of degree d_1 then we define its Macaulay matrix of degree d as the matrix containing the coefficients of

$$M_{f_1}(d) \;=\; \begin{pmatrix} f_1 \\ x_1 f_1 \\ \vdots \\ x_1 x_2 f_1 \\ \vdots \\ x_n^{d-d_1} f_1 \end{pmatrix}$$

where the polynomial f_1 is multiplied with all monomials in $\mathcal{R}_d^n$ from degree 0 up to $d - d_1$. Note that the Macaulay matrix is in fact a generalization of the discrete convolution operation to the multivariate case. It depends explicitly on the degree d for which it is defined, hence the notation $M_{f_1}(d)$. The row space of $M_{f_1}(d)$ will be denoted by $\mathcal{M}_{f_1}$ and can be interpreted as the set of all polynomials $p = f_1\, k_1$ with $k_1 \in \mathcal{R}^n$ and $\deg(p) \leq d$. The Macaulay matrix of any nonzero polynomial f_1 will always be of full row rank. The number of rows and columns of this matrix grows polynomially as a function of d. The number of rows for a degree d are given by $\binom{d-d_1+n}{n}$ and the number of columns by $\binom{d+n}{n}$. Fortunately this matrix is extremely sparse. Its density is inversely proportional to its number of columns and therefore a sparse matrix representation can save a lot of storage space. We can now rewrite Theorem (2.1) as

$$l = f_1\; k_1 = f_2\; k_2 \tag{2}$$

where $k_1, k_2 \in \mathcal{R}^n$ and of degrees $\deg(l) - d_1$ and $\deg(l) - d_2$ respectively. From (2) it can be immediately deduced that the exact LCM of f_1 and f_2 lies

in the row space of both $M_{f_1}(d)$ and $M_{f_2}(d)$. We therefore define an approximate LCM as the polynomial that lies in the intersection $\mathcal{M}_{f_1} \cap \mathcal{M}_{f_2}$ for a certain degree d with a certain tolerance τ. We will make this definition more specific in the next section. The algorithm we propose will therefore check the intersection of these two subspaces. Since $\deg(l)$ is not known a priori, iterations over the degree are necessary. An upper bound for $\deg(l) \triangleq d_l$ is given by $\deg(f_1) + \deg(f_2)$ since in this case the approximate GCD $g = 1$. Deciding whether an intersection exists between $\mathcal{M}_{f_1}$ and $\mathcal{M}_{f_2}$ will be done in our algorithm by inspecting the first principal angle between these two vector spaces.

3.2. Principal Angles and Vectors

Let S_1 and S_2 be subspaces of $\mathbb{R}^n$ whose dimensions satisfy

$$\dim(S_1) = s_1 \geq \dim(S_2) = s_2.$$

The principal angles $0 \leq \theta_1 \leq \ldots \leq \theta_{\min(d_1,d_2)} \leq \pi/2$ between S_1 and S_2 and the corresponding principal directions $u_i \in S_1$ and $v_i \in S_2$ are then defined recursively as

$$\cos(\theta_k) \;=\; \max_{u \in S_1, v \in S_2} u^T v \;=\; u_k^T v_k \;\text{ for } k = 1, \ldots, \min(s_1, s_2)$$

subject to

$$||u|| = ||v|| = 1,$$

and for $k > 1$

$$\begin{aligned} u^T u_i &= 0, \quad i = 1, \ldots, k-1, \\ v^T v_i &= 0, \quad i = 1, \ldots, k-1. \end{aligned}$$

Inspecting the principal angles between $\mathcal{M}_{f_1}$ and $\mathcal{M}_{f_2}$ in each iteration of the algorithm will therefore reveal whether an intersection exists. Since the principal angles form an ordered set one simply needs to check whether $\cos(\theta_1) = 1$. If there is an intersection, the first principal vectors u_1 and v_1 are equal and one of them can be chosen as the sought-after approximate LCM. We now have all ingredients to define our approximate LCM.

Definition 3.1. *Let $f_1, f_2 \in \mathcal{R}^n$, then their approximate LCM with tolerance $\tau > 0$ is the first principal vector u_1 of $\mathcal{M}_{f_1}$ and $\mathcal{M}_{f_2}$ such that*

$$|1 - cos(\theta_1)| \leq \tau$$

where θ_1 is the first principal angle between $\mathcal{M}_{f_1}$ and $\mathcal{M}_{f_2}$.

Several numerical algorithms have been proposed for the computation of the principal angles and the corresponding principal directions [18]. The following theorem will be crucial for the implementation of the algorithm.

Theorem 3.1. *Assume that the columns of Q_{f_1} and Q_{f_2} form orthogonal bases for two subspaces of $\mathcal{R}_d^n$. Let*

$$A \;=\; Q_{f_1}^T\, Q_{f_2},$$

and let the SVD of this $p \times q$ matrix be

$$A \;=\; Y\, C\, Z^T, \quad C \;=\; diag(\sigma_1, \ldots, \sigma_q),$$

where $Y^T Y = Z^T Z = I_q$. If we assume that $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_q$, then the principal angles and principal vectors associated with this pair of subspaces are given by

$$cos(\theta_k) = \sigma_k(A), \quad U = Q_{f_1} Y, \quad V = Q_{f_2} Z.$$

Proof. See [18, p. 582]. □

Theorem 3.1 requires orthogonal bases for both $\mathcal{M}_{f_1}$ and $\mathcal{M}_{f_2}$. The QR or singular value decomposition would provide these bases but computing the full Q or singular vectors is too costly in terms of storage. Both $M_{f_1}(d)^T$ and $M_{f_2}(d)^T$ are large sparse matrices of full column rank. The Implicitly Restarted Arnoldi (IRA) method [19] is therefore the method of choice to retrieve the required left singular vectors Q_{f_1} and Q_{f_2} that span $\mathcal{M}_{f_1}$ and $\mathcal{M}_{f_2}$. Similarly, instead of computing the full SVD of $Q_{f_1}^T\, Q_{f_2}$ one can use IRA iterations to compute the largest singular value σ_1 and corresponding singular vectors y and z. Once a tolerance τ is chosen an approximate LCM can be computed as soon as $|1 - \sigma_1| \leq \tau$. The tolerance τ therefore acts as a measure of how well the approximate LCM needs to lie in $\mathcal{M}_{f_1} \cap \mathcal{M}_{f_2}$. Using IRA iterations instead of the QR decomposition for finding the orthogonal bases Q_{f_1} and Q_{f_2} might suffer from numerical instability due to the loss of orthogonality of the Arnoldi vectors. In practice however we have not yet observed any problems due to this possible numerical instability. This possible loss of orthogonality of the basis vectors can be avoided by using a QR decomposition. The complete procedure to compute an approximate LCM

is summarized in pseudo-code in Algorithm 3.1. Note that it is possible that there is a loss of numerical accuracy when computing the cosine of a small principal angle in double precision using Theorem 3.1. This can be resolved by computing the sine of the small principal angle as explained in [18]. One could therefore define the approximate LCM with tolerance τ as the principal vector u_k for which $\sin(\theta_k) \leq \tau$. The computation of the sine-based approximate LCM would then require a small modification of Algorithm 3.1 (one needs to compute the smallest singular value of $Q_{f_2} - Q_{f_1} Q_{f_1}^T Q_{f_2}$). Using the sine-based approximate LCM has no further implications on the computation of the approximate GCD.

Algorithm 3.1. *Computing an approximate LCM*
Input: *polynomials* $f_1, f_2 \in \mathcal{R}_d^n$, *tolerance* τ
Output: *approximate LCM* l *of* f_1, f_2

```
d ← max(deg(f1), deg(f2))
l ← [ ]
while l = [ ] & d ≤ deg(f1) + deg(f2) do
  M_f1(d) ← Macaulay matrix of degree d of f1
  Q_f1 ← orthogonal basis for M_f1 from IRA iterations
  M_f2(d) ← Macaulay matrix of degree d of f2
  Q_f2 ← orthogonal basis for M_f2 from IRA iterations
  σ1, y, z ← 1st singular value/vectors of Q_f1^T Q_f2 from IRA iterations
  if |1 − σ1| < τ then
    l ← Q_f1 y
  else
    d ← d + 1
  end if
end while
```

4. Computing the GCD

Once the approximate LCM l is found an approximate GCD g is easily retrieved from Theorem 2.1. This implies that no extra tolerance needs to be defined anymore. One could compute the vector corresponding with $f_1\, f_2$ and divide this by l. This division is achieved by constructing the Macaulay matrix of l and solving the sparse overdetermined system of equations

$$M_l^T(d_1 + d_2)\, g^T \;=\; M_{f_2}^T(d_1 + d_2)\, f_1^T.$$

This can be done in the least squares sense [7] using a sparse Q-less QR decomposition [20]. The approximate GCD g is defined in this case as the solution of $\min_w ||f_1\, M_{f_2}(d_1+d_2) - w M_l(d_1+d_2)||_2^2$. The 2-norm of the residual $||f_1\, M_{f_2}^T(d_1+d_2) - g M_l(d_1+d_2)||_2$ then provides a measure on how well the computed GCD satisfies Theorem 2.1 with the approximate LCM. The use of a QR factorization guarantees numerical stability. The problem with this approach however is that computing the product $f_1\, f_2$ can be quite costly in terms of storage. The need to do this multiplication can be circumvented by first doing the division $h_2 = l/f_2$. This is also done by solving another sparse overdetermined system

$$M_{f_2}(d_l)^T h_2^T \;=\; l^T \tag{3}$$

where d_l is the degree of l. Note that $M_{f_2}(d_l)$ will typically have much smaller dimensions than $M_l(d_1+d_2)$. From Theorem 2.1 it is easy to see that $h_2 = f_1/g$ and hence an alternative for the computation of the approximate GCD g is solving

$$M_{h_2}(d_1)^T g^T \;=\; f_1^T. \tag{4}$$

We therefore define the approximate GCD as the least squares solution of (4).

Definition 4.1. *Let $f_1, f_2 \in \mathcal{R}^n$ and let l be their approximate LCM as in Definition 3.1 then their approximate GCD g is the solution of*

$$g \;=\; \min_w ||f_1 - w\, M_{h_2}(d_1)||_2^2.$$

where h_2 is the least squares solution of (3).

For each of the divisions described above the 2-norm of the residual provides a natural measure on how well the division succeeded. Since g is defined up to a scalar, one can improve the 2-norm of the residual by normalizing the right-hand side f_1^T. The residual thus improves with a factor $||f_1||_2$ and will typically be of the same order as $||l - w\, M_{f_2}(d_l)||_2$. Algorithm 4.1 summarizes the high-level algorithm of finding the approximate GCD. The computational complexity of the entire method is dominated by the cost of solving the 2 linear systems (3) and (4). These are both $O(qp^2)$ where p and q stand for the number of rows and columns of the matrices involved. The large number of zero elements however make the solving of these systems still feasible.

Algorithm 4.1. *Computing an approximate GCD*
Input*: polynomials $f_1, f_2, l \in \mathcal{R}^n$ with l an approximate LCM of f_1, f_2*
Output*: approximate GCD g of f_1, f_2*

$h_2 \leftarrow$ *solution of* $\min_w ||l - wM_{f_2}(d_l)||_2^2$

$g \leftarrow$ *solution of* $\min_w ||\frac{f_1}{||f_1||_2} - wM_{h_2}(d_1)||_2^2$

5. Choosing τ

In this section we will derive the relationship between the tolerance τ of Algorithm 3.1 and the ϵ which is commonly used in other methods [11]. Note that if there is no information to choose an ϵ or τ then one can simply compute the σ_1's for all degrees in Algorithm 3.1. Plotting these σ_1's on a graph then reveals all possible approximate LCMs and GCDs which can be found. Running the algorithm for all degrees up to $d_1 + d_2$ corresponds with the case that the approximate GCD is a scalar and does not change the computational complexity of the method. Let $e_1 = f_1 - \tilde{f}_1, e_2 = f_2 - \tilde{f}_2$ with $||e_1||_2 \leq \epsilon, ||e_2||_2 \leq \epsilon$. Then both $M_{f_1}(d)$ and $M_{f_2}(d)$ are perturbed by structured matrices E_1 and E_2. Now suppose that

$$||E_1||_2/||M_{f_1}(d)||_2 \leq \epsilon_1, \quad ||E_2||_2/||M_{f_2}(d)||_2 \leq \epsilon_2$$

then in [18, p. 585] the following expression is proved

$$|\Delta \cos\theta_1| \leq \epsilon_1 \, \kappa_1 + \epsilon_2 \, \kappa_2 \tag{5}$$

where κ_1, κ_2 are the condition numbers of $M_{f_1}(d)$ and $M_{f_2}(d)$ respectively. Since in the exact case $\cos\theta_1 = 1$, the left hand side of (5) is actually $|1 - \sigma_1|$ from Algorithm 3.1. Determining how ϵ_1, ϵ_2 and κ_1, κ_2 are related to ϵ allows then to find a lower bound for τ. First, we derive an upper bound on the condition number of the Macaulay matrices. For a $p \times q$ Macaulay matrix $M_{f_1}(d) = (m_{ij})$ we define $r_i = \sum_{1 \leq j \leq q} |m_{ij}|$ and $c_j = \sum_{1 \leq i \leq p} |m_{ij}|$. The following lemma is a result from the particular structure of the Macaulay matrix and will prove to be useful.

Lemma 5.1. *Let $f_1 \in \mathcal{R}_{d_1}^n$ and $M_{f_1}(d) = (m_{ij})$ its corresponding $p \times q$ Macaulay matrix of degree d. Then*

$$\max_{1 \leq j \leq q} c_j \leq ||f_1||_1$$

where equality is guaranteed from $d \geq (n+1)d_1$.

Proof. The structure of the Macaulay matrix ensures that no column can contain the same coefficient more than once. Hence the largest c_j that can be obtained is $||f_1||_1$. This happens when each coefficient of f_1 is shifted to the LCM of all monomials in n unknowns from degree 0 up to d_1. This LCM equals $x_1^{d_1} x_2^{d_1} \dots x_n^{d_1}$. The degree for which each monomial is shifted to its LCM is $d = (n+1)d_1$. □

We now prove the following upper bound on the largest singular value of $M_{f_1}(d)$.

Theorem 5.1. *Let $f_1 \in \mathcal{R}_d^n$ and $M_{f_1}(d)$ its corresponding $p \times q$ Macaulay matrix of degree d. Then its largest singular value σ_1 is bounded from above by*

$$\sigma_1 \leq ||f_1||_1.$$

Proof. Schur [21] provided the following upper bound on the largest singular value

$$\sigma_1^2 \leq \max_{1 \leq i \leq p, 1 \leq j \leq q} r_i \, c_j.$$

Lemma 5.1 ensures that the maximal c_j is $||f_1||_1$. Each row of the Macaulay matrix contains the same coefficients and therefore $r_i = ||f_1||_1$ for any row i. From this it follows that $\sigma_1 \leq ||f_1||_1$. □

Theorem 5.1 also provides an upper bound on the 2-norm of the product of two polynomials which is better in practice then the bound given in [22, p. 222].

Corollary 5.1. *Let $f_1, f_2 \in \mathcal{R}^n$ with degrees d_1, d_2 respectively then the 2-norm of their product is bounded from above by*

$$||f_1 \, f_2||_2 \leq min\left(||f_1||_1 \, ||f_2||_2, ||f_2||_1 \, ||f_1||_2\right).$$

Proof. The product $f_1 \, f_2$ can be computed using the Macaulay matrix as either $f_1 \, M_{f_2}(d_1 + d_2)$ or $f_2 \, M_{f_1}(d_1 + d_2)$. Theorem 5.1 bounds these vectors in 2-norm from above by $||f_1||_2 \, ||f_2||_1$ and $||f_1||_2 \, ||f_1||_1$ respectively. □

Theorem 5.2. *Let $f_1 \in \mathcal{R}_d^n$ and $M_{f_1}(d) = (m_{ij})$ its corresponding $p \times q$ Macaulay matrix of degree d. Then its smallest singular value σ_{min} is bounded from below by*

$$\sigma_{min} \geq 2|m_{00}| - ||f_1||_1.$$

Proof. Johnson [23] provided the following bound for the smallest singular value of a $p \times q$ matrix with $p \leq q$

$$\sigma_{\min} \geq \min_{1 \leq i \leq p} \left\{ |m_{ii}| - \frac{1}{2}(\sum_{j \neq i} |m_{ij}| + \sum_{j \neq i} |m_{ji}|) \right\}.$$

The structure of the Macaulay ensures that any $m_{ii} = m_{00}$ for any row i. The sums $\sum_{j \neq i} |m_{ij}|$ and $\sum_{j \neq i} |m_{ji}|$ are therefore $r_i - |m_{00}|$ and $c_j - |m_{00}|$ respectively since the diagonal element m_{00} cannot be counted. Note that these sums are limited to the leftmost $p \times p$ block of $M_{f_1}(d)$. The upper bound for both these sums is $||f_1||_1 - |m_{00}|$. □

Note that for the case $2|m_{00}| \leq ||f_1||_1$ this lower bound is trivial. We will assume for the remainder of this article that the nontrivial case applies. From Theorem 5.1 and 5.2 the following upper bound for the condition number of $M_{f_1}(d)$ follows.

Corollary 5.2. *Let $f_1 \in \mathcal{R}_d^n$ and $M_{f_1}(d)$ its corresponding $p \times q$ Macaulay matrix of degree d, then its condition number κ_1 is bounded from above for the nontrivial case by*

$$\kappa_1 \leq \frac{||f_1||_1}{2|m_{00}| - ||f_1||_1}. \tag{6}$$

This upper bound on the condition numbers can be used in expression (5). Note that this upper bound can be quite an overestimation. In practice, one sees that the condition number grows very slowly in function of the degree and therefore the least squares problems of Algorithm 4.1 are well-conditioned. First we relate the ϵ of $||f_1 - \tilde{f}_1||_2 \leq \epsilon$ with ϵ_1 from above. It is clear that

$$\begin{array}{rrcl} & ||E_1||_2 & \leq & \epsilon_1 \, ||M_{f_1}(d)||_2 \\ \Leftrightarrow & \sigma_1(E_1) & \leq & \epsilon_1 \, \sigma_1(M_{f_1}(d)) \\ \Leftrightarrow & ||e_1||_1 & \leq & \epsilon_1 \, ||f_1||_1 \\ \Leftrightarrow & ||e_1||_2 & \leq & \epsilon_1 \, ||f_1||_1. \end{array}$$

This allows us to set $\epsilon = \epsilon_1 \, ||f_1||_1$. Using this in (5) the following upper bound on the error of the cosine is obtained

$$|\Delta \cos\theta_1| \leq \epsilon \, (\frac{1}{2|m_{00,1}| - ||f_1||_1} + \frac{1}{2|m_{00,2}| - ||f_2||_1}). \tag{7}$$

The extra subscript in $m_{00,1}$ and $m_{00,2}$ is to make the distinction between f_1 and f_2. This expression also provides an upper bound on τ to find an approximate LCM with a given ϵ.

6. Numerical Experiments

In this section we discuss some numerical examples and compare the results with those obtained from NAClab, the MATLAB counterpart of ApaTools, by Zeng [24]. The 'mvGCD' method from NAClab improves the accuracy of its result by Gauss-Newton iterations and will hence always produce results with lower relative errors. We therefore use these results as a reference. All numerical examples were computed on a 2.66 GHz quad-core desktop computer with 8 GB RAM in MATLAB.

6.1. Example 1

First, consider the following two bivariate polynomials with exact coefficients

$$\begin{aligned} f_1 &= (x_1x_2 + x_2^2 + 200)\,(x_1x_2 + x_2 + 200)\,(100 + 2x_1^3 - 2x_1x_2 + 3x_2^2) \\ f_2 &= (x_1x_2 + x_2^2 + 200)\,(x_1x_2 + x_2 + 200)\,(100 - 2x_1^3 + 2x_1x_2 - 3x_2^2). \end{aligned}$$

Both f_1 and f_2 satisfy the nontrivial case. The relatively large coefficients of the constant terms in the 3 factors have as a consequence that the absolute value of the coefficients of f_1, f_2 vary between 1 and 4000000. Since we assume the coefficients are exact, we set $\tau = 10\,\epsilon_0$ where ϵ_0 is the unit roundoff ($\approx 10^{-16}$). The exact GCD g is obviously the product of the first 2 factors of f_1 and hence the exact LCM l is

$$l = g\,(100 + 2x_1^3 - 2x_1x_2 + 3x_2^2)\,(100 - 2x_1^3 + 2x_1x_2 - 3x_2^2).$$

The absolute difference $|\Delta\cos\theta_1|$ drops from 7.99×10^{-4} to 4.44×10^{-16} when going from $d = 9$ to $d = 10$. The condition numbers of $M_{f_2}(10)$ and $M_{h_2}(10)$ are 1.05 and 1.06 respectively. The relative error between our computed numerical LCM and the exact answer is 8.01×10^{-13}. The 2-norm of the residual for calculating h_2 is 2.12×10^{-14} and for solving (4) 6.11×10^{-15}. For the computed GCD, the relative error is 3.48×10^{-11}. The GCD computed from NAClab has a relative error of 1.24×10^{-20}. The 2-norm of the absolute difference between the approximate GCD of our method compared to one from NAClab is 5.13×10^{-13}. In order to investigate how

our method performs when the polynomials have inexact coefficients we now add perturbations of the order 10^{-3} to f_1, f_2 and obtain

$$\begin{aligned}\tilde{f}_1 &= (x_1x_2 + x_2^2 + 200)(x_1x_2 + x_2 + 200 + 10^{-3}x_1)(100 + 2x_1^3 - 2x_1x_2 + 3x_2^2)\\ \tilde{f}_2 &= (x_1x_2 + x_2^2 + 200)(x_1x_2 + x_2 + 200 - 10^{-3}x_1)(100 - 2x_1^3 + 2x_1x_2 - 3x_2^2).\end{aligned}$$

Again, both $\tilde{f}_1$ and $\tilde{f}_2$ satisfy the nontrivial case. The perturbation of 10^{-3} in one of the factors results in $||f_1 - \tilde{f}_1||_2$ and $||f_2 - \tilde{f}_2||_2$ being equal to 20.0. With $\epsilon \leq 25$ an approximate LCM of degree 10 is found. From (7) $|\Delta\cos\theta_1| \leq 1.35 \times 10^{-05}$. The approximate LCM can then be found from Algorithm 3.1 by setting $\tau \geq 1.35 \times 10^{-05}$. This is in fact a large overestimation of the real error. $|\Delta\cos\theta_1|$ drops from 7.99×10^{-4} to 1.11×10^{-16} when going from $d = 9$ to $d = 10$. Note that a perturbation of 20 on the coefficients resulted in no significant change in $|\Delta\cos\theta_1|$. The same applies to the condition numbers of $M_{f_2}(d_l)$ and $M_{h_2}(d_1)$. The 2-norm of the residual when calculating h_2 and the approximate GCD is 7.60×10^{-9} and 4.22×10^{-9} respectively. The absolute difference between our approximate GCD and the one from NAClab is of the order 10^{-5}. All computations of approximate GCD's took about 0.17 seconds for our algorithm and 0.025 seconds for NAClab. The reason for this difference in execution time can be seen from the next example in which we investigate whether our method can handle a large number of variables.

6.2. Example 2

Suppose we have the following polynomials

$$\begin{aligned}u &= (x_1 + x_2 + x_3 + \ldots + x_{10} + 1)^2\\ v &= (x_1 - x_2 - x_3 - \ldots - x_{10} - 2)^2\\ w &= (x_1 + x_2 + x_3 + \ldots + x_{10} + 2)^2\end{aligned}$$

then $f_1 = u\,v$ and $f_2 = u\,w$ are two 10-variate polynomials of degree four. The 2-norm of the difference between the approximate GCD computed using Algorithm 4.1 with the approximate GCD from NAClab is 2.38×10^{-10}. It took NAClab 4.70 seconds to calculate the result while it took our method 70.14 seconds. Out of these 70.14 seconds, 67.78 were spent constructing the M_{f_1} and M_{f_2} matrices. In effect, the actual computation of the approximate LCM and GCD in our algorithm took 2.36 seconds. Most of the gain in execution time can therefore be made in a more efficient construction and updating of the Macaulay matrices.

6.3. Example 3

In this example the capability of Algorithms 3.1 and 4.1 to handle high degrees is tested. Let

$$\begin{aligned} p &= x_1 - x_2\, x_3 + 1 \\ q &= x_1 - x_2 + 3\, x_3 \\ f_1 &= p^6\, q^{12} \\ f_2 &= p^{12}\, q^6. \end{aligned}$$

Note that for this case the exact GCD is $p^6\, q^6$. This is reminiscent of f_1 and f_2 having multiple common roots for the univariate case. It took NAClab several runs to find a result. In most cases it returned an error message. This is probably somehow related to the high degrees since for the case $f_1 = p^4\, q^6$ and $f_2 = p^6\, q^4$ an approximate GCD could always be computed with NAClab. For this lower degree case, the run times were 0.79 seconds for NAClab and 4.09 seconds for Algorithm 4.1. The 2-norm of the difference between the two computed approximate GCDs was 2.89×10^{-15}. For the high degree case this 2-norm was 4.37×10^{-08} and the total run time was 11.05 seconds for NAClab and 260.22 seconds for our algorithm.

6.4. Example 4

The next example demonstrates the robustness of our algorithm with respect to noisy coefficients. We revisit the polynomials of Example 6.2 and change the number of variables to three. We therefore have

$$\begin{aligned} u &= (x_1 + x_2 + x_3 + 1)^2 \\ v &= (x_1 - x_2 - x_3 - 2)^2 \\ w &= (x_1 + x_2 + x_3 + 2)^2 \end{aligned}$$

and again set $f_1 = u\, v$ and $f_2 = u\, w$. Every nonzero coefficient of f_1, f_2 is then perturbed with noise, uniformly drawn from $\left[0, 10^{-k}\right]$ for $k = 1, 3, 5, 7$. We will denote the exact GCD by g, the approximate GCD found with NAClab by $\tilde{g}_n$ and the approximate GCD found with Algorithm 4.1 by $\tilde{g}_\tau$. Table 1 lists the 2-norms of the differences between the coefficients. As expected, $\tilde{g}_n$ lies slightly closer to the exact result. However, NAClab cannot find an approximate GCD anymore for $k = 1$.

Table 1: Errors Example 6.4

	$k=1$	$k=3$	$k=5$	$k=7$
$\|\|g-\tilde{g}_n\|\|_2$	NA	7.83×10^{-05}	4.45×10^{-07}	3.86×10^{-09}
$\|\|g-\tilde{g}_\tau\|\|_2$	1.17×10^{-2}	1.01×10^{-04}	5.73×10^{-07}	2.23×10^{-08}
$\|\|\tilde{g}_n-\tilde{g}_\tau\|\|_2$	NA	8.42×10^{-05}	8.42×10^{-07}	2.44×10^{-08}

References

[1] S. Barnett, Polynomials and Linear Control Systems, M. Dekker, New York, 1983.

[2] P. Stoica, T. Söderström, Common Factor Detection And Estimation, Automatica 33 (1996) 985–989.

[3] S. Unnikrishna Pillai, B. Liang, Blind Image Deconvolution using a robust GCD approach, IEEE Transactions on Image Processing 8 (2) (1999) 295–301.

[4] N. Karmarkar, Y. N. Lakshman, On Approximate GCDs of Univariate Polynomials, Journal of Symbolic Computation 26 (6) (1998) 653–666.

[5] A. Schönhage, Quasi-GCD computations, Journal of Complexity 1 (1) (1985) 118–137.

[6] I. Z. Emiris, A. Galligo, H. Lombardi, Certified Approximate Univariate GCDs, Journal of Pure and Applied Algebra 117-118 (0) (1997) 229 – 251.

[7] R. M. Corless, P. M. Gianni, B. M. Trager, S. M. Watt, The Singular Value Decomposition for Polynomial Systems, in: ACM International Symposium on Symbolic and Algebraic Computation, 1995, pp. 195–207.

[8] M.-T. Noda, T. Sasaki, Approximate GCD and its application to ill-conditioned equations, Journal of Computational and Applied Mathematics 38 (1-3) (1991) 335 – 351.

[9] Z. Zeng, B. H. Dayton, The approximate GCD of inexact polynomials Part I: a univariate algorithm, preprint (2004).

[10] R. Corless, S. Watt, L. Zhi, QR factoring to compute the GCD of univariate approximate polynomials, IEEE Transactions on Signal Processing 52 (12) (2004) 3394–3402.

[11] G. M. Diaz-Toca, L. Gonzalez-Vega, Computing greatest common divisors and squarefree decompositions through matrix methods: The parametric and approximate cases, Linear Algebra and its Applications 412 (2-3) (2006) 222–246.

[12] L. González-Vega, An elementary proof of Barnett's theorem about the greatest common divisor of several univariate polynomials, Linear Algebra and its Applications 247 (0) (1996) 185–202.

[13] J. R. Winkler, X. Lao, The calculation of the degree of an approximate greatest common divisor of two polynomials, Journal Of Computational And Applied Mathematics 235 (6) (2011) 1587–1603.

[14] E. Kaltofen, J. P. May, Z. Yang, L. Zhi, Approximate factorization of multivariate polynomials using singular value decomposition, Journal of Symbolic Computation 43 (5) (2008) 359–376.

[15] Z. Zeng, The approximate GCD of inexact polynomials part II: a multivariate algorithm, in: Proceedings of the 2004 international symposium on Symbolic and algebraic computation, 2004, pp. 320–327.

[16] MATLAB R2011a, The Mathworks Inc., Natick, Massachusetts (2011).

[17] D. A. Cox, J. B. Little, D. O'Shea, Ideals, Varieties and Algorithms, 3rd Edition, Springer-Verlag, 2007.

[18] R. Björck, G. H. Golub, Numerical Methods for Computing Angles Between Linear Subspaces, Mathematics of Computation 27 (123) (1973) pp. 579–594.

[19] R. B. Lehoucq, D. C. Sorensen, Deflation Techniques for an Implicitly Restarted Arnoldi Iteration, SIAM Journal on Matrix Analysis and Applications 17 (4) (1996) 789–821.

[20] T. A. Davis, Algorithm 915, SuiteSparseQR: Multifrontal multithreaded rank-revealing sparse qr factorization, ACM Trans. Math. Softw. 38 (1) (2011) 1–22.

[21] I. Schur, Bemerkungen zur Theorie der beschränkten Bilinearformen mit unendlich vielen Veränderlichen, Journal für die Reine und Angewandte Mathematik 140 (1911) 1–28.

[22] B. Beauzamy, E. Bombieri, P. Enflo, H. Montgomery, Products Of Polynomials in Many Variables, Journal Of Number Theory 36 (2) (1990) 219–245.

[23] C. R. Johnson, A Gershgorin-type Lower Bound for the Smallest Singular Value, Linear Algebra and its Applications 112 (1989) 1–7.

[24] Z. Zeng, ApaTools: A Software Toolbox for Approximate Polynomial Algebra, in: M. Stillman, J. Verschelde, N. Takayama (Eds.), Software for Algebraic Geometry, Vol. 148 of The IMA Volumes in Mathematics and its Applications, Springer New York, 2008, pp. 149–167.