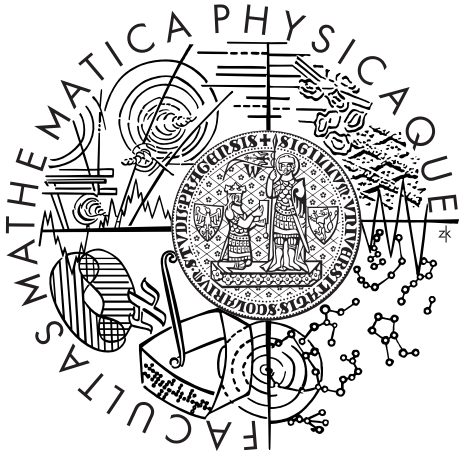
Charles University in Prague

Faculty of Mathematics and Physics

MASTER THESIS



Ján Eliaš

Approximate Polynomial Greatest Common Divisor

Department of Numerical Mathematics

Supervisor: Jan Zítko

Study programme: Numerical and Computational Mathematics

Specialization: Numerical Analysis

Prague 2012

First and foremost, I would like to thank my supervisor, Doc. Jan Zítko. I am grateful for his constant assistance and help for all the time and effort he devoted to the supervision of this thesis. I greatly appreciate the help of Dr. Joab Winkler from the University of Sheffield. I would like to thank them for invaluable discussions on the topics of polynomial GCD and numerical rank estimation.

Finally, I would like to thank my mother and all the people who have supported and helped me throughout these years.

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Charles University in Prague has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 paragraph 1 of the Copyright Act.

In Prague, 2nd August 2012 Ján Eliaš

Název práce: Approximate Polynomial Greatest Common Divisor

Autor: Ján Eliaš

Katedra: Katedra numerické matematiky, MFF UK

Vedoucí diplomové práce: Doc. RNDr. Jan Zítko, CSc., Katedra numerické matematiky, MFF UK

Abstrakt: Výpočet najväčšieho spoločného deliteľa (GCD) dvoch polynómov patrí medzi základné problémy numerickej matematiky. Euklidov algoritmus je najstaršia a bežne používaná metóda na výpočet GCD, avšak táto metóda je značne nestabilná. Výpočet GCD je navyše zle postavená úloha v tom zmysle, že ľubovoľný šum pridaný ku koeficientom polynómov redukuje netriviálny GCD na konštantu.

Jednu skupinu nových metód predstavujú metódy založené na odhade numerickej hodnosti matíc. Operácie s polynómami sa tak redukujú na maticové počty. Ich nevýhodou je, že ani numerická hodnosť nemusí byť spočítaná presne a hodnoverne kvôli citlivosti singulárnych čísel na šume. Cieľom práce je prekonať citlivosť výpočtu GCD na šume.

Klíčová slova: AGCD, Sylvesterova matica, numerická hodnosť, TLS

Title: Approximate Polynomial Greatest Common Divisor

Author: Ján Eliaš

Department: Department of Numerical Mathematics, MFF UK

Supervisor: Doc. RNDr. Jan Zítko, CSc., Department of Numerical Mathematics, MFF UK

Abstract: The computation of polynomial greatest common divisor (GCD) ranks among basic algebraic problems with many applications. The Euclidean algorithm is the oldest and usual technique for computing GCD. However, the GCD computation problem is ill-posed, particularly when some unknown noise is applied to the polynomial coefficients. Since the Euclidean algorithm is unstable, new methods have been extensively studied in recent years.

Methods based on the numerical rank estimation represent one group of current methods. Their disadvantage is that the numerical rank cannot be computed reliably due to the sensitivity of singular values on noise. The aim of the work is to overcome the ill-posed sensitivity of GCD computation in the presence of noise.

Keywords: AGCD, Sylvester matrix, numerical rank, TLS

Introduction

Motivation, problem statement and aims of thesis

The computation of greatest common divisor (GCD) of two polynomials ranks among basic problems of numerical algebra. It has applications in many technical areas such as control theory [18], image (signal) processing [20], *etc.*

The problem is simply stated as finding the coefficients of the GCD of two univariate polynomials $f(x)$ and $g(x)$.

The task as stated is well defined and can be solved symbolically if polynomials with the exact coefficients are considered. However, problems arise whenever the computation is performed in a floating point environment. Polynomials with a non-constant GCD can be very easily changed so that their non-constant GCD becomes a constant. For example, measurement or human errors along with rounding errors appearing in the computation affect the coefficients of polynomials and polynomials that have a non-constant GCD become coprime in presence of such errors. The GCD computation is therefore an ill-posed problem in sense that arbitrarily small changes in the coefficients may reduce a non-constant GCD to a constant.

Many researchers have tried to overcome the ill-posed sensitivity of the GCD computation and some more or less efficient numerical methods have been recently developed. All these methods compute only an approximation of the GCD of polynomials and it is therefore called an approximate greatest common divisor (AGCD).

A preview on some methods is given later, however note that a very important group of methods works in two principal stages: estimation of the degree of an AGCD on one side and the computation of the coefficients performed afterwards. While the second partial problem is relatively easy, the computation of the degree is non-trivial and difficult. We will see later that the problem of the degree computation is equivalent to the estimation of the rank of a Sylvester matrix.[1] This "problem substitution" does not make the problem of the degree calculation easier, since the rank estimation is noise sensitive as well, but it allows us to avoid doing long polynomial divisions (as those in the Euclidean algorithm).

Most techniques for the numerical rank computation calculate numerical rank through the inspection of the singular values and count these singular values that are greater

[1] This part of the thesis is only an introduction to the studied problem, precise definitions of terms are omitted and left to the next sections.

than a specified threshold. However, due to sensitivity of singular values on noise it may happen very often that computed numerical rank differs from the exact rank of a matrix (linear independence of columns or rows can be very easily destroyed only by rounding errors). The computation of an AGCD by using this equivalently stated problem may give therefore unacceptable results.

The motivation behind this thesis is the development of a polynomial AGCD solver that successfully overcomes the noise and hence the threshold dependence.

A new algorithm for the computation of an AGCD is presented. Firstly, we explore in details the relationship between AGCD and Total Least Squares (TLS). Our new method is then based on a method for solving structured TLS problems.

Note that a method of Dr. Winkler [24, 25] is also the extension of an existing method for solving TLS customised for the purposes of the AGCD computation. Author, during his stay at the University of Sheffield, participated in the project led by Dr. Winkler and a modification of their recent results is also presented.

Organisation of thesis

The organization of the thesis is as follows:

Chapter 1 introduces basic preliminaries that are essential for further explanation. Mainly, Sylvester matrices and some preprocessing operations of polynomials are discussed. Sylvester matrices represent algebraic structure on which all computations are performed, however, matrices may be badly conditioned. Hence, the preprocessing operations are not important from the theoretical point of view because GCD and rank are well defined, but they cannot be computed reliably mainly because of high condition numbers of involved matrices.

An AGCD is defined in Chapter 2. This chapter also contains basic relations between the GCD and Sylvester matrices. In particular, the degree of the GCD of two polynomials is equal to the rank loss of the Sylvester matrix formed from the coefficients of the polynomials. Problem is then equivalently formulated as computing the rank of the Sylvester matrix. A new algorithm used to obtain estimates on the numerical rank of the Sylvester matrix, and therefore on the degree of an AGCD is proposed in the same chapter.

Chapter 3 introduces an iterative refinement of an AGCD. The Gauss-Newton method is used as a tool for the computation of the coefficients of an AGCD.

Chapter 4 is devoted to a description of an AGCD relation to TLS problems. Since Sylvester matrices are structured block Toeplitz matrices, then a condition on structure has to be involved in a standard TLS formulation. Two methods for solving structured TLS (STLS) problem are then briefly presented.

Finally, Chapter 5 introduces two methods for an AGCD computation. The first of them is known from the literature as STLN and SNTLN, respectively, [13, 24, 25]. The second method is new and uses Newton method to find the stationary point of a Lagrangian function. Hence, it might be referred to as the Lagrange-Newton STLS

method.

The algorithms presented in the thesis have been implemented in Matlab. A software package also written in Matlab is enclosed.

Notation

In the thesis the following notation is generally used:

$\mathbb{R}$	real numbers
$\mathbb{R}^{m\times n}$	space of m by n matrices with real entries
f, g, p	polynomials as lower-case Latin letters
$\mathbf{x}$, $\mathbf{z}$, $\mathbf{f}$	vectors as bold lower-case Latin letters
$\mathbf{0}$	vector of zeros
A, C, S	matrices as upper-case Latin letters
A^T	transposition of a matrix A
$A^\dagger$	Moore-Penrose inverse of a matrix A
I_n	n by n identity matrix
$\mathbf{e_i}$	ith canonical vector, ith column of I_n
$0_{m\times n}$	m by n zero matrix
$\varepsilon_{\text{mach}}$	machine precision, in double precision $\varepsilon_{\text{mach}} \approx 10^{-16}$
Range(A)	range of A; for $A \in \mathbb{R}^{m\times n}$, Range($A$) $= \{\mathbf{y} \in \mathbb{R}^m \colon \exists\, \mathbf{x} \in \mathbb{R}^n,$ $A\mathbf{x} = \mathbf{y}\}$
Null(A)	null space of A; for $A \in \mathbb{R}^{m\times n}$, Null($A$) $= \{\mathbf{x} \in \mathbb{R}^n \colon A\mathbf{x} = \mathbf{0}\}$
$\sigma_i, \sigma_i(A)$	ith singular value of A
$\mathbf{v}_i, \mathbf{v}_i(A)$	right singular vector associated with $\sigma_i(A)$
$\sigma_{\min}, \sigma_{\min}(A)$	the smallest singular value of A
$\mathbf{v}_{\min}, \mathbf{v}_{\min}(A)$	right singular vector associated with $\sigma_{\min}(A)$

The entries of a vector or a matrix are written as lower-case Latin letters with subscripts. For example, $\mathbf{x} = [x_1, \ldots, x_n]^T \in \mathbb{R}^n$. In case of the column partitioning of a matrix, columns are written in bold lower-case Latin. For example, $\mathbf{a}_i \in \mathbb{R}^m$ is the ith column of $A = [\mathbf{a}_1, \ldots, \mathbf{a}_n] \in \mathbb{R}^{m\times n}$.

For a matrix $A \in \mathbb{R}^{m\times n}$, the rank of A is defined by

$$\text{rank}(A) = \dim(\text{Range}(A)). \tag{1}$$

We say that a matrix $A \in \mathbb{R}^{m\times n}$ is of full rank or A has full rank, respectively, if $\text{rank}(A) = \min\{m, n\}$. A matrix $A \in \mathbb{R}^{m\times n}$ is rank deficient if $\text{rank}(A) < \min\{m, n\}$. A matrix $A \in \mathbb{R}^{m\times n}$ is rank deficient by k if $\text{rank}(A) + k = \min\{m, n\}$, k is the nullity of A.

Note that rank(A) is also referred to as the numerical rank of A. A proper definition of the numerical rank is given in Section .

Abbreviations

AGCD	approximate greatest common divisor
GCD	greatest common divisor
LNSNTLS	Lagrange-Newton structured non-linear total least squares
LNSTLS	Lagrange-Newton structured total least squares
LS	least squares
LSE	equality constrained least squares
SNTLN	structured non-linear total least norm
STLN	structured total least norm
STLS	structured total least squares
SVD	singular value decomposition
TLS	total least squares

Norms

From the most frequent norms, the 2-norm and the Frobenius matrix norm are used in the thesis. In particular, for a vector $\mathbf{x} \in \mathbb{R}^n, \mathbf{x} = [x_1, \ldots, x_n]^T$ and a matrix $A = (a_{ij})_{i,j} \in \mathbb{R}^{m \times n}$, these norms are defined by

$$\|\mathbf{x}\|_2 = \sqrt{(x_1^2 + \cdots + x_n^2)} = \sqrt{\mathbf{x}^T\mathbf{x}},$$

$$\|A\|_2 = \sup_{\mathbf{x} \neq \mathbf{0}} \frac{\|A\mathbf{x}\|_2}{\|\mathbf{x}\|_2},$$

$$\|A\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2}.$$

Rarely, the norms

$$\|\mathbf{x}\|_1 = |x_1| + \cdots + |x_n|,$$

$$\|\mathbf{x}\|_\infty = \max_{i=1,\ldots,n} |x_i|$$

may appear.

Note that the distance between two polynomials is meant to be the distance between the vectors of their coefficients measured by the 2-norm.

When it is not necessary to specify a particular norm, the subscript identifying the norm is omitted and the 2-norm is used, $\|\cdot\| = \|\cdot\|_2$.

Numerical rank

We say that a m by n matrix A has numerical rank r, $r \leq n \leq m$, within a specified threshold θ if

$$r = \inf_B \{\text{rank}(B) : \|A - B\| \leq \theta\}.$$

In other words, A has numerical rank r if it is close to a defective matrix of rank r within the tolerance θ. This definition, that can be found in [9], is rather theoretical and difficult to use. However, in the same paper [9] some other equivalent characteristics are stated:

Let $A = U\Sigma V^T$ be the SVD of $A \in \mathbb{R}^{m\times n}$, $m \geq n$, with the singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$. Then A has numerical rank r within θ if and only if there exist δ such that

$$\sigma_r \geq \delta > \theta \geq \sigma_{r+1}. \tag{2}$$

The constant δ is an upper bound applied to the values of θ for which the numerical rank remains at least r. We will, however, use slightly simplified definition of the numerical rank, since the additional parameter δ does not simplify the numerical rank computation as can be seen from Figure 1.

Definition. Let $A = U\Sigma V^T$ be the SVD of $A \in \mathbb{R}^{m\times n}$, $m \geq n$, with the singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$. Then A is of numerical rank r within the tolerance θ if θ and r satisfy

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > \theta \geq \sigma_{r+1} \geq \cdots \geq \sigma_n.$$

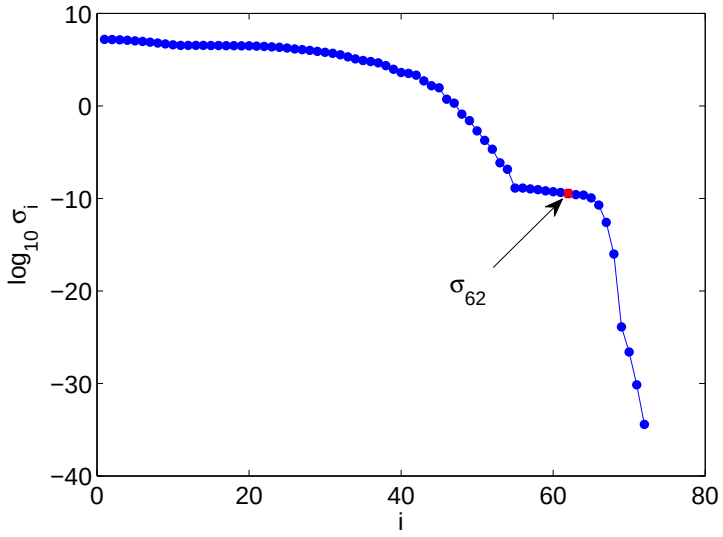


Figure 1: Singular values of the Sylvester matrix $S \in \mathbb{R}^{72\times 72}$ which has rank $r = 62$. To get the numerical rank that is equal to r, θ has to be chosen so that $\sigma_{62} > \theta \geq \sigma_{63}$. The threshold $\theta = \varepsilon_{\mathbf{mach}}\|S\| \approx 1.1 \times 10^{-9}$ returns $r = 57$ and the numerical-rank gap then gives $r = 68$ that are incorrect results.

Note that the numerical rank with $\theta = 0$ may be interpreted as the exact rank of a matrix.

The threshold θ in the definition should reflect the general level of noise (relative errors) involved in matrices. It is advised in [11] on p. 261 that the numerical rank of a m by n floating point matrix A, $m \geq n$, should be estimated with $\theta = \varepsilon_{\text{mach}}\|A\|$, where $\varepsilon_{\text{mach}}$ is the machine precision. We will work with noise polynomials with the

relative noise expressed by the signal-to-noise ratio ϵ, e.g. $\epsilon = 10^{-8}$. Then $\theta = \epsilon\|S\|$, where S is the Sylvester matrix of these polynomials, should be taken in the numerical rank computation.

In [1, 24, 25] a matrix is of numerical rank r if the ratio σ_r/σ_{r+1} is the largest gap among all ratios $\sigma_k/\sigma_{k+1}, k = 1, \ldots, n-1$. This ratio σ_r/σ_{r+1} is called the numerical-rank gap.

Chapter 1

Polynomials, matrices and scaling

In the first chapter we provide the basic discussion on used polynomials and Sylvester matrices that enable to gain a clear understanding of the thesis. This chapter also describes important preprocessing operations of polynomials that make the numerical rank of a matrix better defined computationally.

All sorts of appearing polynomials are summarised and two ways of perturbation that can be possibly applied to the polynomials are presented in Section 1.1. Section 1.2 covers Sylvester matrices. Finally, an unwilling property of badly conditioned Sylvester matrices needs to be dealt with. Sylvester matrices may be unbalanced whenever polynomials with varying coefficients in magnitude are used. In this case solutions of systems with these matrices may be inaccurate. Therefore, scaling of systems, i.e. preprocessing operations of polynomials are discussed in Section 1.3.

1.1 Exact and inexact polynomials

In the thesis, polynomials of one variable are only considered. Typically, we will work with a pair of two polynomials $f(x)$ and $g(x)$,

$$f(x) = \sum_{i=0}^{m} a_i x^{m-i} \quad \text{and} \quad g(x) = \sum_{j=0}^{n} b_j x^{n-j} \tag{1.1}$$

with $a_0 a_m \neq 0$ and $b_0 b_n \neq 0$.

The thesis in its later parts works with several other kinds of polynomials. In particular, there is a pair of inexact polynomials for which an approximate GCD is computed. These inexact polynomials $f(x)$ and $g(x)$ of degrees m and n, respectively, are of the form (1.1). Inexact polynomials are thought to have coefficients affected by a noise that, for example, may come from physical experiments. The noise level is expressed by a signal-to-noise ratio ϵ, mostly with $\epsilon = 10^{-8}$ in our examples.

The theoretically exact forms are denoted by $\hat{f}(x)$ and $\hat{g}(x)$, respectively. The inexact polynomials $f(x)$ and $g(x)$ are then derived from the exact forms.

We will use componentwise perturbation to simulate real processes that may produce noise, and so inexact data. In particular, if $\mathbf{c_f}$ is a vector of uniformly distributed

random numbers from $[-1, 1]$, c_i is its ith entry and $\hat{a}_i$ is the ith coefficient of the exact polynomial $\hat{f}(x)$, then

$$a_i = \hat{a}_i + \epsilon c_i \hat{a}_i \tag{1.2}$$

is the ith coefficient of $f(x)$.

On the other side, one can consider a normwise perturbation

$$\mathbf{f} = \hat{\mathbf{f}} + \epsilon \frac{\|\hat{\mathbf{f}}\|}{\|\mathbf{c_f}\|} \mathbf{c_f} \tag{1.3}$$

where $\hat{\mathbf{f}}$ and $\mathbf{f}$ are the vectors of coefficients of $\hat{f}(x)$ and $f(x)$, respectively.

Finally, there are two other sorts of polynomials: $\bar{f}(w)$ and $\bar{g}(w)$ derived from the inexact polynomials $f(x)$ and $g(x)$ by the substitution $x = \gamma w$ for some constant γ and $\tilde{f}(x)$ and $\tilde{g}(x)$ that appear in discussed AGCD methods.

Note that variables x, w following the names of polynomials are omitted where it does not lead to confusions. Hence, for example, f is written instead of $f(x)$, $\bar{f}$ instead of $\bar{f}(w)$.

Remark 1.1.1. It is important to emphasize that in the theoretical part of the thesis, we work with a general polynomial pair $\{f, g\}$ of some properties that are specified. These polynomials do not have to be necessarily inexact in sense described above.

1.2 Sylvester matrices

For the given two univariate polynomials f and g in (1.1) of degrees m and n, respectively, the Sylvester matrix $S(f, g) \in \mathbb{R}^{(m+n)\times(m+n)}$ is a structured matrix of the form

$$S(f,g) = \begin{bmatrix} a_0 & & & & b_0 & & & \\ a_1 & a_0 & & & b_1 & b_0 & & \\ \vdots & a_1 & \ddots & & \vdots & b_1 & \ddots & \\ a_{m-1} & \vdots & \ddots & a_0 & b_{n-1} & \vdots & \ddots & b_0 \\ a_m & a_{m-1} & \ddots & a_1 & b_n & b_{n-1} & \ddots & b_1 \\ & a_m & \ddots & \vdots & & b_n & \ddots & \vdots \\ & & \ddots & a_{m-1} & & & \ddots & b_{n-1} \\ & & & a_m & & & & b_n \end{bmatrix}, \tag{1.4}$$

where the coefficients of f occupy the first n columns and the coefficients of g occupy the last m columns.

There exist various other conventions for arranging the coefficients of f and g. Some authors define the Sylvester matrix of f and g as transposition of $S(f, g)$ in (1.4). In [17, 30] we can find $S(f, g)$ in (1.4) multiplied by a permutation matrix so that every column of the coefficients of f is followed by a column of the coefficients of g, and *vice versa*. Furthermore, the coefficients of f and g are written in reverse order from

the last coefficient to the leading coefficient, i.e. the coefficient a_0 in (1.4) is replaced by a_m, a_1 is replaced by a_{m-1}, *etc.* All these matrices are commonly called Sylvester matrices or resultant matrices. In the thesis, the form (1.4) is used.

The kth Sylvester subresultant matrix $S_k(f,g) \in \mathbb{R}^{(m+n-k+1)\times(m+n-2k+2)}$ is derived from $S(f,g)$ by deleting the last $k-1$ columns of the coefficients of f, the last $k-1$ columns of the coefficients of g, and the last $k-1$ rows of $S(f,g)$.

The kth Toeplitz (convolution, Cauchy) matrix $C_k(f) \in \mathbb{R}^{(m+k)\times k}$ is the matrix

$$C_k(f) = \begin{bmatrix} a_0 & & & \\ a_1 & a_0 & & \\ \vdots & a_1 & \ddots & \\ a_{m-1} & \vdots & \ddots & a_0 \\ a_m & a_{m-1} & \ddots & a_1 \\ & a_m & \ddots & \vdots \\ & & \ddots & a_{m-1} \\ & & & a_m \end{bmatrix}. \tag{1.5}$$

Hence, the Sylvester matrix (1.4) consists of the two convolution matrices $C_n(f)$ and $C_m(g)$, $S(f,g) = [C_n(f), C_m(g)]$. Similarly, $S_k(f,g) = [C_{n-k+1}(f), C_{m-k+1}(g)]$. In some cases, a short notation $S = S_1$ and S_k, for some $k \in \{1, 2, \ldots, \min\{m,n\}\}$, is used instead of $S(f,g) = S_1(f,g)$ and $S_k(f,g)$, respectively.

Remark 1.2.1. Sylvester and Toeplitz matrices are linear in sense that two polynomials f_1 and f_2 of the same degree and a polynomial g satisfy, $S(f_1 + f_2, g) = S(f_1, g) + S(f_2, g)$, $C_k(c_1 f_1 + c_2 f_2) = c_1 C_k(f_1) + c_2 C_k(f_2)$, $c_1, c_2 \in \mathbb{R}$, *etc.*

Remark 1.2.2. Algebraically, the convolution is the same operation as multiplying polynomials. Hence, the coefficients $\mathbf{p}$ of the product p of two polynomials f and g of degrees m and n can be computed as the convolution of their coefficients $\mathbf{f}$ and $\mathbf{g}$. It can be easily verified that the convolution $\mathbf{p}$ is expressed in matrix-vector notation as

$$\mathbf{p} = C_{n+1}(f)\mathbf{g} = C_{m+1}(g)\mathbf{f}, \tag{1.6}$$

where $\mathbf{f} \in \mathbb{R}^{m+1}$, $\mathbf{g} \in \mathbb{R}^{n+1}$ and $\mathbf{p} \in \mathbb{R}^{m+n+1}$.

1.3 Preprocessing operations

In real computations with matrices, outputs may be impaired by rounding errors. Measure of the asymptotically worst case of how much results can be modified in proportion to small changes in the inputs is expressed by the condition number. With a higher condition number a higher influence of rounding errors may occur.

We will see later that the computation of an AGCD requires a system

$$A\mathbf{x} = \mathbf{c} \tag{1.7}$$

to be solved, where $\mathbf{c}$ is a column of the Sylvester matrix S and A is formed from the remaining columns of S. However, the Sylvester matrix S, and so A, may have very large condition numbers. In particular, if the coefficients of a polynomial f are much larger than the coefficients of g and/or the ratio of the maximum coefficient of all coefficients of f and g to the minimum coefficient of all coefficients of f and g in magnitude is large, then the condition number of $S = S(f, g)$ may be large. That is why the calculation of an AGCD is an ill-condition problem.

Since a high condition number may lead to instabilities caused by rounding errors, a scaling is usually applied to make results more accurate. For instance, in [11] on p. 125 it is advised to solve a scaled system

$$D_1^{-1} A D_2 \mathbf{y} = D_1^{-1} \mathbf{c} \tag{1.8}$$

instead of (1.7) where $\mathbf{x} = D_2\mathbf{y}$. Matrices D_1 and D_2 are diagonal matrices that have to be obviously chosen so that the condition number of $D_1^{-1}AD_2$ is significantly smaller that the condition number of A.

In this section we show how to construct the matrices D_1 and D_2 through some operations applied to the coefficients of polynomials. However, we do not claim that this construction is the only correct way how to reduce condition numbers of Sylvester matrices. Moreover, our operations are rather heuristic and follow from the "behaviour" of the coefficients.

The first operation is normalisation of the coefficients of polynomials. In [24, 25, 27] it is recommended to use geometric mean instead of known norms included the 2-norm as the normalisation quotient. More notes on this choice can be found in Section 1.3.1. The second operation, discussed in Section 1.3.2, is a relative scaling of the coefficients. Relative scaling is also suggested in papers [25, 27]. Scaling as a preprocessing operation of polynomials makes the problem of an AGCD computation a non-linear problem.

1.3.1 Normalisation by the geometric mean

Condition number of a Sylvester matrix strongly depends on its entries, i.e. on the coefficients of polynomials. One possible way how to get a better conditioned Sylvester matrix is the normalisation of the coefficients of polynomials. The normalisation of the coefficients cannot be omitted especially when the coefficients of polynomials vary by several orders of magnitude. The scaling of the coefficients of f and g by the 2-norm can be found in [2, 7]. Polynomials can be then normalised by other norms, by the standard deviation of the coefficients of polynomials, *etc.*

It is advised in [24, 25, 27] to use geometric mean of coefficients, because, as it is said in [24], the normalisation by the geometric mean gives a "better average". Moreover, the geometric mean preserves a propagation of rounding errors better. In particular, consider a simple polynomial with the coefficients 1, 10^3 and 10^6, $\mathbf{f} = [1, 10^3, 10^6]$. Then every normalisation by

$$\|\mathbf{f}\|_1 \approx 10^6, \quad \|\mathbf{f}\|_2 \approx 10^6 \quad \text{and} \quad \|\mathbf{f}\|_\infty = 10^6$$

gives the coefficients which are numerically similar to the original coefficients. On the other side, the normalisation by the geometric mean 10^3 gives the numbers 10^{-3}, 1

and 10^3. Three norms above are almost independent of the small coefficients, but the normalisation by the geometric mean does not show this insensitivity.

The polynomials f and g defined in (1.1) are transformed by the normalisation by the geometric mean to the forms

$$\begin{aligned} f(x) &= \sum_{i=0}^{m} \bar{a}_i x^{m-i}, \qquad \bar{a}_i = \frac{a_i}{(\prod_{k=0}^{m} |a_k|)^{\frac{1}{m+1}}}, \\ g(x) &= \sum_{j=0}^{n} \bar{b}_j x^{n-j}, \qquad \bar{b}_j = \frac{b_j}{(\prod_{k=0}^{n} |b_k|)^{\frac{1}{n+1}}}, \end{aligned} \tag{1.9}$$

where the geometric means are computed only from the non-zero coefficients. Due to noise added to the coefficients in (1.2) and rounding errors, a zero coefficient may become a non-zero. In this case a threshold that specify whether a coefficient is zero or not has to be applied. That threshold should reflect the level of noise and therefore it is a disadvantage of the normalisation by the geometric mean, since in practice the level of noise is not usually known or it is known only approximately. See [26] for more details.

1.3.2 Relative scaling of polynomials

The second step of the preprocessing operations introduces two parameters α and γ. The relative scaling of the coefficients is developed by Winkler and his students [24, 25, 26, 27].

Since the Sylvester matrix $S(f, g)$ of the two polynomials f and g in (1.1) is the block Toeplitz matrix, it follows from (1.4) that

$$\operatorname{rank}(S(f, g)) = \operatorname{rank}(S(f, \alpha g))$$

for any non-zero constant $\alpha \in \mathbb{R}$, where *rank* in this case is the theoretical rank defined in (1). Although, the rank of $S(f, \alpha g)$ is not influenced by α, significant differences may occur whenever α is considered in the numerical rank computation as it is shown in Example 1.3.3.

Furthermore, the exact GCD of polynomials f and g is equal to, up to an arbitrary scalar multiplier, the GCD of f and αg. That is why it is possible to use $S(f, \alpha g)$ instead of $S(f, g)$ for the computation of an AGCD. Here, the parameter α is interpreted as the weight of g relative to the unit weight of f, where f and g are the normalised polynomials (1.9).

The normalised polynomials f and g given in (1.9) can be further scaled by the parameter substitution

$$x = \gamma w \tag{1.10}$$

where γ is a constant to be computed and w is the new independent variable. This substitution brings significantly better results in the numerical rank estimation in case when the coefficients of polynomials vary widely in magnitude, see [25, 26, 27]. The

substitution $x = \gamma w$ transforms the normalised polynomials f and g in (1.9) to the polynomials $\bar{f} = \bar{f}(w)$ and $\bar{g} = \bar{g}(w)$,

$$\begin{aligned} \bar{f}(w) &= \sum_{i=0}^{m} (\bar{a}_i \gamma^{m-i}) w^{m-i}, \qquad \bar{a}_i = \frac{a_i}{(\prod_{k=0}^{m} |a_k|)^{\frac{1}{m+1}}}, \\ \bar{g}(w) &= \sum_{j=0}^{n} (\bar{b}_j \gamma^{n-j}) w^{n-j}, \qquad \bar{b}_j = \frac{b_j}{(\prod_{k=0}^{n} |b_k|)^{\frac{1}{n+1}}}. \end{aligned} \tag{1.11}$$

The parameters α and γ need to be computed. However, since it is desired to minimise the ratio of the maximum coefficient to the minimum coefficient of all coefficients of the polynomials $\bar{f}(w)$ and $\alpha\bar{g}(w)$ in magnitude, unknown constants α and γ can be computed as a solution to the following minimisation problem:

$$\min_{\alpha,\gamma} \left\{ \frac{\max\left\{\max_{i=0,\ldots,m} |\bar{a}_i \gamma^{m-i}|, \max_{j=0,\ldots,n} |\alpha \bar{b}_j \gamma^{n-j}|\right\}}{\min\left\{\min_{i=0,\ldots,m} |\bar{a}_i \gamma^{m-i}|, \min_{j=0,\ldots,n} |\alpha \bar{b}_j \gamma^{n-j}|\right\}} \right\}. \tag{1.12}$$

It is shown in [25, 27] that this minimisation problem is a standard linear programming (LP) problem with the objective function

$$T - S = \begin{bmatrix} 1, & -1, & 0, & 0 \end{bmatrix} \begin{bmatrix} T \\ S \\ \phi \\ \mu \end{bmatrix} \tag{1.13}$$

and $2m + 2n + 4$ constrained conditions

$$\begin{array}{rcrcrcll} T & - & (m-i)\phi & & & \geq & \bar{\alpha}_i, & i = 0, \ldots, m \\ T & - & (n-j)\phi & - & \mu & \geq & \bar{\beta}_j, & j = 0, \ldots, n \\ -S & + & (m-i)\phi & & & \geq & -\bar{\alpha}_i, & i = 0, \ldots, m \\ -S & + & (n-j)\phi & + & \mu & \geq & -\bar{\beta}_j, & j = 0, \ldots, n, \end{array} \tag{1.14}$$

where $\mu = \log \alpha$, $\phi = \log \gamma$, $\bar{\alpha}_i = \log |\bar{a}_i|$ and $\bar{\beta}_j = \log |\bar{b}_j|$. Let us denote a solution of (1.12) as α_{opt} and γ_{opt}, respectively.

If there is a coefficient of f or g equal to zero, then the corresponding condition in (1.14) is removed. But there holds a similar conclusion as in the previous section, i.e. a coefficient is rarely equal to zero because of the presence of noise and rounding errors and so a threshold on the coefficients is applied.

Once a solution pair $\{\alpha_{opt}, \gamma_{opt}\}$ of the LP problem (1.13) is found, the polynomials (1.11) after the substitution become

$$\bar{f}(w) = \sum_{i=0}^{m} (\bar{a}_i \gamma_{opt}^{m-i}) w^{m-i} \quad \text{and} \quad \bar{g}(w) = \sum_{j=0}^{n} (\bar{b}_j \gamma_{opt}^{n-j}) w^{n-j}. \tag{1.15}$$

These polynomials are normalised by the geometric mean again because θ_{opt} is usually not equal to one.

Finally, the polynomials which we receive from the preprocessing operations and the coefficients of which are the entries of $S(\bar{f}, \alpha_{opt}\bar{g})$ are

$$\begin{aligned} \bar{f}(w) &= \sum_{i=0}^{m} (\bar{a}_i^* \gamma_{opt}^{m-i}) w^{m-i}, \qquad \bar{a}_i^* = \frac{\bar{a}_i}{\left(\prod_{k=0}^{m} |\bar{a}_k \gamma_{opt}^{m-k}|\right)^{\frac{1}{m+1}}}, \\ \bar{g}(w) &= \sum_{j=0}^{n} (\bar{b}_j^* \gamma_{opt}^{n-j}) w^{n-j}, \qquad \bar{b}_j^* = \frac{\bar{b}_j}{\left(\prod_{k=0}^{n} |\bar{b}_k \gamma_{opt}^{n-k}|\right)^{\frac{1}{n+1}}}, \end{aligned} \tag{1.16}$$

where $\bar{a}_i$, $i = 0, \ldots, m$ and $\bar{b}_j$, $j = 0, \ldots, n$ are defined in (1.11).

Note that the substitution (1.10) does not change the multiplicities of roots of involved polynomials as it is shown in [26]. Hence, as a conclusion we get that the Sylvester matrix $S(\bar{f}, \alpha_{opt}\bar{g})$ can be used in methods for AGCD computations instead of $S(f, g)$.

It is easy to verify that the operations discussed above can be written as a matrix multiplication of $S(f, g)$ by diagonal matrices. In particular, suppose that there are given the polynomials f and g in (1.9) of degrees m and n and, for simplicity, also assume that the substitution only is performed. If we now define the diagonal $(m+n)$ by $(m+n)$ matrices

$$\begin{aligned} D_1(\gamma) &= \operatorname{diag}(\gamma^m, \gamma^{m-1}, \ldots, \gamma, 1, \gamma^{-1}, \ldots, \gamma^{-(n-1)}), \\ D_2(\alpha, \gamma) &= \operatorname{diag}(1, \gamma, \ldots, \gamma^{n-1}, \alpha\gamma^{-(m-n)}, \alpha\gamma^{-(m-n)+1}, \ldots, \alpha\gamma^{-(m-n)+m-1}), \end{aligned} \tag{1.17}$$

then we can write

$$S(\bar{f}, \alpha\bar{g}) = D_1(\gamma) S(f, g) D_2(\alpha, \gamma), \tag{1.18}$$

where $\bar{f}$ and $\bar{g}$ are the polynomials from (1.11) with the coefficients not normalised by the geometric mean. Inclusion of the normalisation would render the structures of the diagonal matrices slightly complicated, since the normalisation quotients have to be included. Considering α_{opt} and θ_{opt} makes no difficulties and omitting the ith columns of $S(f, g)$ and $S(\bar{f}, \alpha\bar{g})$ will result in omitting the ith column and the ith row of $D_2(\alpha, \gamma)$. Hence, D_1 and D_2 can be used in (1.8).

Example 1.3.1. For $m = 3$ and $n = 2$, i.e. for

$$\begin{aligned} f(x) &= a_0 x^3 + a_1 x^2 + a_2 x + a_3, \\ g(x) &= b_0 x^2 + b_1 x + b_2 \end{aligned}$$

we have

$$S(\bar{f}, \alpha\bar{g}) = \begin{bmatrix} a_0\gamma^3 & & \alpha b_0\gamma^2 & & \\ a_1\gamma^2 & a_0\gamma^3 & \alpha b_1\gamma & \alpha b_0\gamma^2 & \\ a_2\gamma & a_1\gamma^2 & \alpha b_2 & \alpha b_1\gamma & \alpha b_0\gamma^2 \\ a_3 & a_2\gamma & & \alpha b_2 & \alpha b_1\gamma \\ & a_3 & & & \alpha b_2 \end{bmatrix}$$

$$= \begin{bmatrix} \gamma^3 & & & & \\ & \gamma^2 & & & \\ & & \gamma & & \\ & & & 1 & \\ & & & & \gamma^{-1} \end{bmatrix} \begin{bmatrix} a_0 & & b_0 & & \\ a_1 & a_0 & b_1 & b_0 & \\ a_2 & a_1 & b_2 & b_1 & b_0 \\ a_3 & a_2 & & b_2 & b_1 \\ & a_3 & & & b_2 \end{bmatrix} \begin{bmatrix} 1 & & & & \\ & \gamma & & & \\ & & \alpha\gamma^{-1} & & \\ & & & \alpha & \\ & & & & \alpha\gamma \end{bmatrix}$$

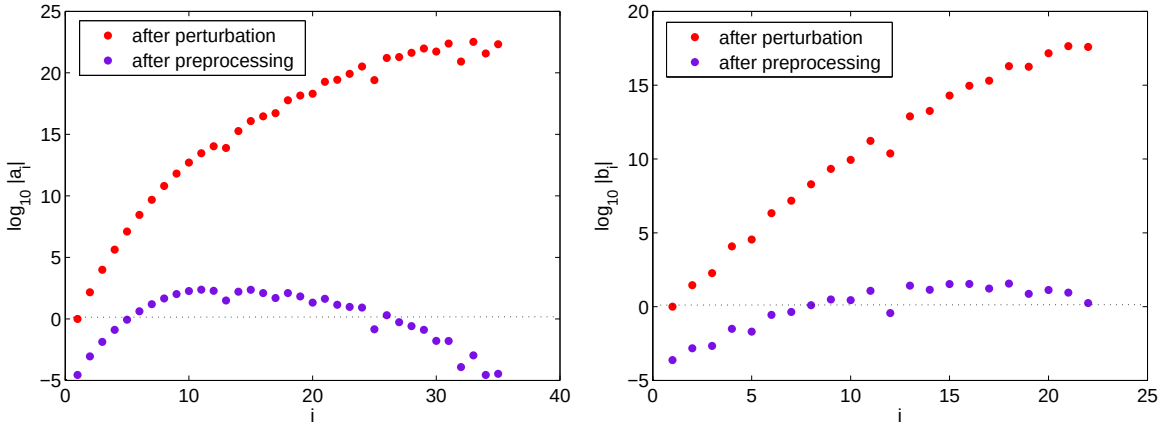
$= D_1(\gamma)S(f,g)D_2(\alpha,\gamma).$ ♣

Both mentioned preprocessing operations contribute to the reduction of condition number of Sylvester matrices.

Example 1.3.2. Consider the exact polynomials $\hat{f}$ and $\hat{g}$,

$$\begin{aligned}\hat{f}(x) &= (x+8.755)^9(x+3.921)^{10}(x-1.974)^9(x+7.496)^6,\\ \hat{g}(x) &= (x+8.755)^8(x+3.921)^5(x-7.688)^8,\end{aligned} \tag{1.19}$$

where $m = \deg \hat{f} = 34$ and $n = \deg \hat{g} = 21$. These polynomials are perturbed componentwisely with the signal-to-noise ratio $\epsilon = 10^{-8}$ and then preprocessed by the operations described in the previous sections. Coefficients of the involved inexact and preprocessed polynomials in magnitude are plotted in Figure 1.1. It is clearly seen



(a) Coefficients of $\hat{f}$ **(b) Coefficients of $\hat{g}$**

Figure 1.1: The coefficients of polynomials $\hat{f}$ and $\hat{g}$ in (1.19) in magnitude after perturbation by the signal-to-noise ratio $\epsilon = 10^{-8}$ (before preprocessing), •, and after preprocessing by the normalisation and the relative scaling, •.

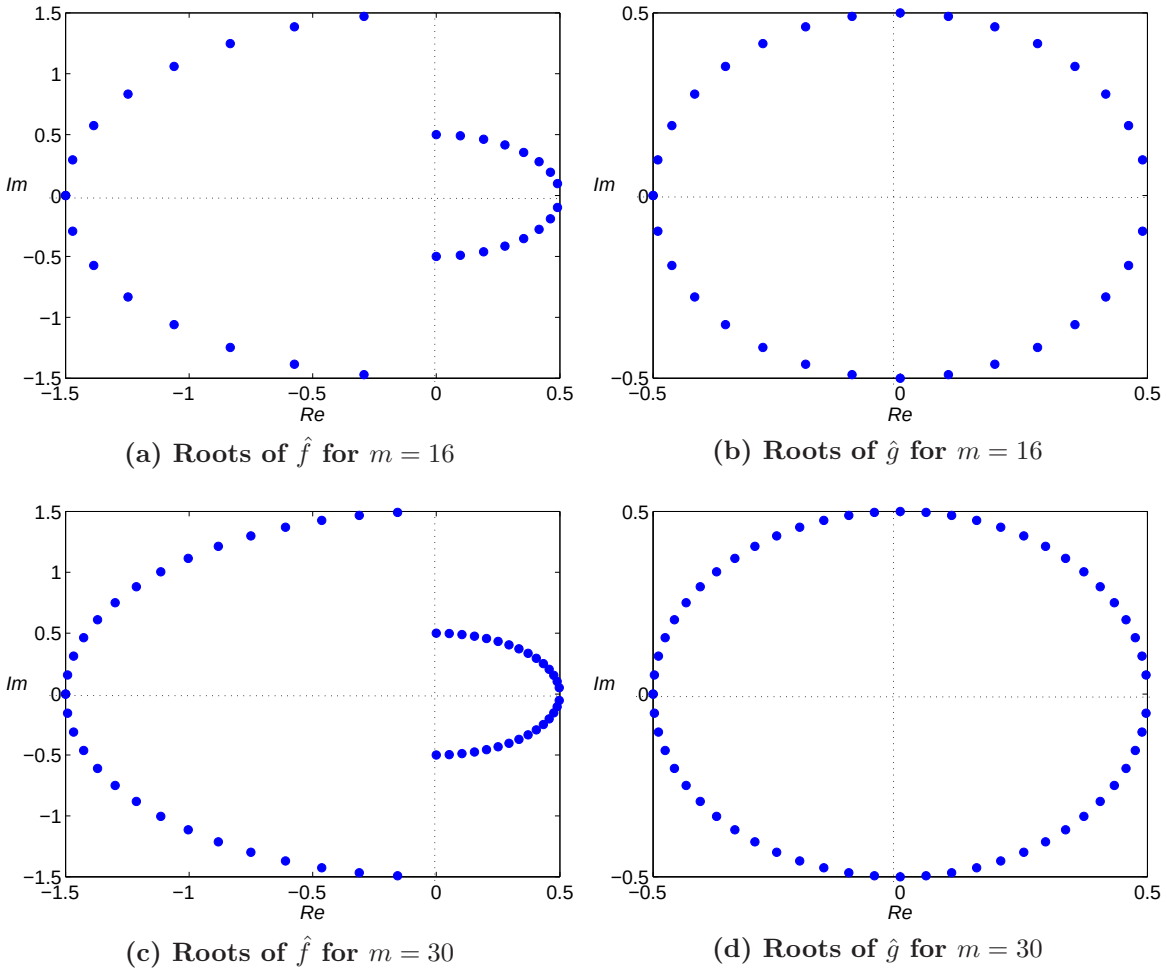
from figures that there is a reduction of several orders of magnitude in the coefficients. For example, the coefficients of f in magnitude are pressed from range $\{0, 10^{23}\}$ to the range $\{10^{-5}, 10^{3}\}$ in the logarithmic scale only by the normalisation by the geometric mean and the substitution $x = \gamma_{opt} w$. The condition number of the Sylvester matrix is reduced from the condition number $\kappa(S(f,g)) = 10^{21}$ of to $\kappa(S(\bar{f}\alpha_{opt}\bar{g})) = 10^{17}$. Constants $\alpha_{opt} = 4.38 \times 10^3$ and $\gamma_{opt} = 4.5069$ are computed from the LP problem (1.12). ♣

Example 1.3.3. For an even integer m consider the following polynomials $\hat{f}$ and $\hat{g}$,

$$\begin{aligned}\hat{f}(x) &= \prod_{i=1}^{\frac{m}{2}} \left[(x-r_1\alpha_i)^2 + r_1^2\beta_i^2\right] \prod_{i=\frac{m}{2}+1}^{m} \left[(x-r_2\alpha_i)^2 + r_2^2\beta_i^2\right],\\ \hat{g}(x) &= \prod_{i=1}^{m} \left[(x-r_1\alpha_i)^2 + r_1^2\beta_i^2\right],\end{aligned} \tag{1.20}$$

where $\alpha_i = \cos\left(\frac{\pi i}{m}\right), \beta_i = \sin\left(\frac{\pi i}{m}\right), i = 1, \ldots, n, r_1 = 0.5$ and $r_2 = 1.5$.

These polynomials are numerically interesting since their roots form a circle, respectively two half circles, and with higher m the roots are closer to each other, see Figure 1.2. Moreover, an additional noise may shift roots in arbitrary directions and therefore it may be more complicated to retrieve them.



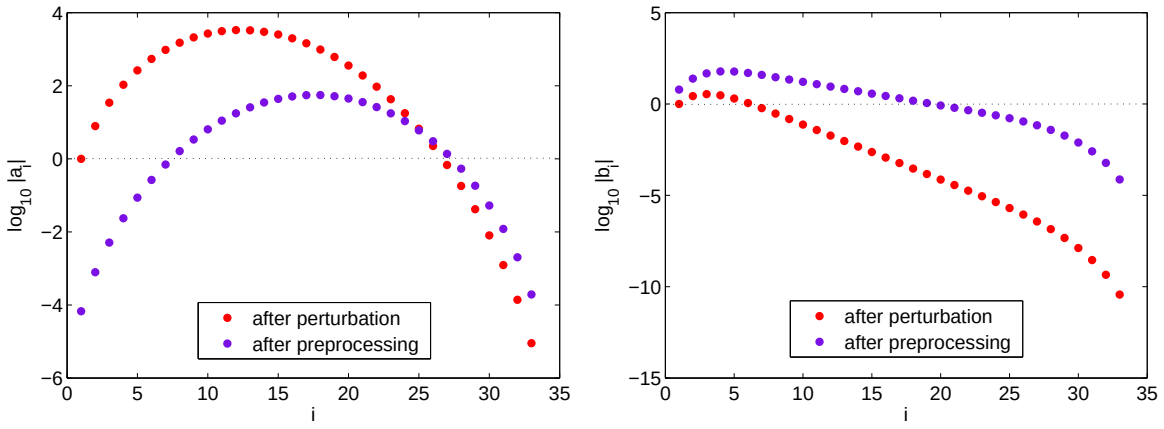
(a) Roots of $\hat{f}$ for $m = 16$ **(b) Roots of $\hat{g}$ for $m = 16$**

(c) Roots of $\hat{f}$ for $m = 30$ **(d) Roots of $\hat{g}$ for $m = 30$**

Figure 1.2: Roots of the exact polynomials defined in (1.20) for $m = 16$ and $m = 32$.

If we now perturb the polynomials in (1.20) by noise of the signal-to-noise ratio $\epsilon = 10^{-8}$ and process inexact polynomials by the operations including the normalisation by the geometric mean from Section 1.3.1 and the relative scaling from Section 1.3.2, then we get the polynomials $\bar{f}$ and $\bar{g}$ in (1.16) with the coefficients plotted in Figure 1.3. Computed values from the preprocessing operations are $\alpha_{opt} = 0.9168$ and $\gamma_{opt} = 0.6729$.

Pressing the coefficients around zero is clearly seen, moreover, scaling reduces the condition number $\kappa(S(f, g)) = 10^{24}$ to the condition number $\kappa(S(\bar{f}, \alpha_{opt}\bar{g})) = 10^{16}$. Table 1.1 then shows the condition numbers of Sylvester matrices of polynomials in (1.20) for various choices of m. Note for record that values α_{opt} and γ_{opt} computed



(a) Coefficients of $\hat{f}$ **(b) Coefficients of $\hat{g}$**

Figure 1.3: The coefficients of polynomials $\hat{f}$ and $\hat{g}$ in (1.20) for $m = 16$ in magnitude after perturbation by the signal-to-noise ratio $\epsilon = 10^{-8}$ (before preprocessing), •, and after preprocessing by the normalisation and the relative scaling, •.

in the LP problem (1.12) are computed simultaneously.[1] However, these constants can be also computed separately by solving the two similar LP problems. The first involves the coefficients of polynomials before substitution $x = \gamma w$ and so only α_{opt} is computed. Once α_{opt} is found, the second LP problem with computed α_{opt} is solved yielding γ_{opt}. Table 1.1 also compares condition numbers and computed values for both approaches. Condition numbers of the first approach, i.e. when α_{opt} and γ_{opt} are computed together, are denoted by κ_1. The second approach, i.e. when α_{opt} and γ_{opt} are computed separately, has the subscript 2 in κ_2.

We can see that Sylvester matrices of perturbed inexact polynomials are badly conditioned. Preprocessing operations bring better balanced and conditioned matrices, however, condition numbers after the scaling are still too high for high m. Hence, it may be worthless to use Sylvester matrices for the computation of an AGCD of polynomials of high degrees. Numerical experiments shown later are used only on polynomials of low and medium degrees (of degrees around 50, 60 and less). Note however, that we work with polynomials in power basis. Some improvements can be achieved by considering other basis, e.g. Chebyshev basis. ♣

Summary

A brief overview of the background needed for an AGCD computation has been given in this chapter. A small discussion on used polynomials, Sylvester matrices and the preprocessing operations of polynomials have been subsequently introduced.

[1] Condition numbers are computed by the Matlab function *cond.m*. Linear programming problem (1.12) is solved by *linprog.m*.

m	$\kappa(S(f,g))$	$\kappa_1(S(\bar{f},\alpha_{opt}\bar{g}))$	$\kappa_2(S(\bar{f},\alpha_{opt}\bar{g}))$
6	2.91×10^{14}	4.41×10^{10} $\alpha_{opt} = 1.3083, \gamma_{opt} = 0.5925$	1.23×10^{11} $\alpha_{opt} = 2.1525, \gamma_{opt} = 0.6175$
10	4.53×10^{19}	9.97×10^{12} $\alpha_{opt} = 1.1656, \gamma_{opt} = 0.6424$	1.06×10^{13} $\alpha_{opt} = 1.3630, \gamma_{opt} = 0.6474$
16	3.34×10^{24}	5.01×10^{16} $\alpha_{opt} = 0.9168, \gamma_{opt} = 0.6729$	5.33×10^{16} $\alpha_{opt} = 0.3219, \gamma_{opt} = 0.6512$
22	1.61×10^{28}	1.17×10^{20} $\alpha_{opt} = 0.6724, \gamma_{opt} = 0.6871$	1.45×10^{20} $\alpha_{opt} = 0.1122, \gamma_{opt} = 0.6597$
30	2.20×10^{34}	1.47×10^{23} $\alpha_{opt} = 0.4337, \gamma_{opt} = 0.6978$	1.33×10^{23} $\alpha_{opt} = 0.0132, \gamma_{opt} = 0.6583$
50	2.02×10^{50}	1.60×10^{33} $\alpha_{opt} = 0.2068, \gamma_{opt} = 0.7059$	1.20×10^{33} $\alpha_{opt} \approx 10^{-4}, \gamma_{opt} = 0.6621$
100	6.56×10^{82}	1.89×10^{54} $\alpha_{opt} = 0.0359, \gamma_{opt} = 0.6383$	2.21×10^{59} $\alpha_{opt} \approx 10^{4}, \gamma_{opt} = 0.6821$
200	4.91×10^{145}	8.23×10^{87} $\alpha_{opt} \approx 10^{-4}, \gamma_{opt} = 0.5916$	2.57×10^{142} $\alpha_{opt} \approx 10^{23}, \gamma_{opt} = 0.6888$

Table 1.1: The condition numbers of Sylvester matrices of the inexact polynomials, $\kappa(S(f,g))$, preprocessed polynomials with the values arising from LP problem (1.12), $\kappa_1(S(\bar{f},\alpha_{opt}\bar{g}))$, and preprocessed polynomials with values computed separately, $\kappa_2(S(\bar{f},\alpha_{opt}\bar{g}))$.

Sylvester matrices represent the basic structure for the AGCD computations and most of the work is performed on Sylvester matrices. However, we have seen that Sylvester matrices may be badly conditioned and unbalanced, for example, when polynomials of high degrees or with varying coefficients are considered. The scaling makes Sylvester matrices better balanced and it presses the condition numbers of involved matrices, so it should not be omitted. However, there are still limitations of involved polynomials (on their degrees, positions of their roots, multiplicities of the roots) in real computations with Sylvester matrices whether the scaling is used or not.

Chapter 2

Approximate Greatest Common Divisor

The computation of GCD of two polynomials is an ill-posed problem in sense that an arbitrarily tiny perturbation can reduce the degree of GCD. Even if the polynomials f and g have a non-trivial GCD, any perturbations δf and δg added to f and g, respectively, can cause that $f + \delta f$ and $g + \delta g$ become coprime, i.e. have no common factor. Similar difficulties may appear only by considering the rounding errors.

Implementations of traditionally known methods, e.g. the Euclidean algorithm, do not work properly in a floating point arithmetic. Therefore, new methods have been extensively studied in recent years and a term approximate greatest common divisor (AGCD) has been introduced, see [2, 13, 24, 30]. Different notation for an AGCD however can be found in the literature, for instance quasi-*GCD and ϵ-GCD. See the doctoral thesis of Boito [4] for a summary.*

In this chapter, firstly, an AGCD is defined in Section 2.1. Previous works on an AGCD are summarised in Section 2.2. In Section 2.3 we explore in detail the relationship between an AGCD and Sylvester resultant matrices. Importance of the correct rank estimation will immediately arise as a conclusion. Numerical rank and difficulties of its estimation are discussed in Section . New rank revealing algorithm that avoids the computation of the complete SVD of a matrix is presented in Section 2.4. The computation of an AGCD is left to the next chapters.

2.1 Definition of AGCD

In the thesis a greatest common divisor in approximate sense is defined with respect to the definition given in [17].

Definition 2.1.1. An approximate greatest common divisor (AGCD) of two given inexact polynomials f and g is a unique polynomial that satisfies the following three characteristics:

1.) *nearness:* an AGCD is the exact GCD of a set of polynomials $\left\{\{\tilde{f}_j, \tilde{g}_j\}\right\}_{j\in\mathcal{J}}$

"close" to the given pair $\{f,\ g\}$,

2.) *maximum degree:* an AGCD is of the highest possible degree among the exact GCDs of those polynomials satisfying *nearness*, i.e.

$$\deg \mathrm{AGCD}(f,g) = \max_{j\in\mathcal{J}} \deg \mathrm{GCD}(\tilde{f}_j, \tilde{g}_j),$$

where $\{\tilde{f}_j, \tilde{g}_j\} \in \left\{\{\tilde{f}_j, \tilde{g}_j\}\right\}_{j\in\mathcal{J}}$,

3.) *minimum distance:* from those polynomials of $\left\{\{\tilde{f}_j, \tilde{g}_j\}\right\}_{j\in\mathcal{J}}$ with the exact GCD of the highest degree, an AGCD minimises the distance between them and the given polynomials $\{f, g\}$.

Remark 2.1.1. Note that $\mathcal{J}$ represents an arbitrary set of indices. Nearness (distance) is measured by the 2-norm.

The condition of *nearness* from the definition is intuitively obvious since AGCD clearly is the exact GCD of another polynomial pair. However, we cannot be satisfied with arbitrary polynomials $\tilde{f}$ and $\tilde{g}$ with the AGCD as their exact GCD, but only with those polynomials for which $\|f-\tilde{f}\| < \mathrm{tol}_1$ and $\|g-\tilde{g}\| < \mathrm{tol}_2$ for some small tolerances that are the functions of a signal-to-noise ratio. Moreover, there cannot exist another polynomials satisfying the previous inequalities with the exact GCD of degree greater than the degree of the exact GCD of $\tilde{f}$ and $\tilde{g}$, i.e. the *maximum degree* condition holds.

Roughly speaking, one can create a *legitimate solution space of polynomial pairs* defined by the signal-to-noise ratio, similar to a space in [24]. The legitimate space contains only polynomial pairs with a non-constant GCD that are close to the given polynomials (in sense of mentioned inequalities). Then the AGCD is one of those exact GCDs with the highest degree. If there are two or more candidates, then the AGCD minimises the distance between the concerned polynomials.

The definition of the AGCD reflects a situation when empirical polynomials are considered, which is important since only polynomials specified within a tolerance are usually known. However, the given inexact polynomials are coprime with probability almost one. It is therefore necessary to perturb each polynomial slightly so that the perturbed polynomials have a non-constant GCD. Following the definition, it is necessary to compute the smallest perturbations such that the perturbed polynomials have a non-constant GCD with the highest possible degree.

Note that the AGCD definition also admits the situation that the degree of the AGCD is different from the degree of the exact GCD.

2.2 Preview of works on AGCD

The oldest non-trivial and well-known technique for computing GCD is the Euclidean algorithm, [15]. The algorithm is based on polynomial divisions that can be expensive and the algorithm itself usually gives very poor results when some noise is added to

the coefficients of one or both polynomials. Some stabilised versions of the Euclidean algorithm exist, for example, the version by Hribernig and Stetter [12]. Their modification computes the GCD within some given tolerance θ by changing the termination criterion in the algorithm. On the other side, the computed GCD does not have to be of maximum degree, i.e. only a common divisor of polynomials may be computed.

Since the degree of the GCD and the GCD itself can be computed from resultant matrices, some other methods based on the estimation of the rank of those matrices have been developed. Corless *et. al.* [6] use the SVD to identify the degree of the GCD. The QR decomposition of the Sylvester matrix is used in [7, 29]. Bini and Boito [2] use QR decomposition of the Bézout resultant matrix. A disadvantage of the methods is that they do not work with a possible additional noise imposed in polynomials, i.e. only the rounding errors are considered in the degree estimation.

There are some other strategies in the AGCD computation including optimisation techniques [14], e.g. algorithms developed by Noda and Sasaki [19], Zarowski [29] and Zeng [30]. However, only the rounding errors are taken into account.

Optimisation techniques are used by Karmarkar and Lakshman [20] to compute the smallest perturbations to be applied to the coefficients of a pair of polynomials in order that they have a non-constant GCD.

Another group of methods are structure preserving matrix methods [13, 24, 25, 5]. Unlike the previous methods, these methods assume that the given polynomials may be perturbed by some noise of unknown level. These methods are nothing more than a specific application of a known method for solving structured Total Least Square (TLS) problems. In particular, it is the Structured Total Least Norm (STLN) method developed in [21]. We have already outlined in Section 1.3 that a system $A\mathbf{x} = \mathbf{c}$ has to be solved, where $S = [\mathbf{c}, A]$ is the Sylvester matrix. The system is overdetermined, however, we will not look for a solution $\mathbf{x}$ in the least squares sense but as a TLS solution. More details are left to the next chapters. Note however, that the classical STLN method in [21] was customised for the AGCD purposes in [13]. Winkler and his students [24, 25] then embedded the STLN method to the Structured Non-Linear Total Least Norm (SNTLN) method by involving the preprocessing operations from Section 1.3.

A different approach for solving a structured TLS problem can be found in [8, 16, 23]. To the best of author's knowledge there is no method for the AGCD computation based on this approach. We develop here a new method and compare it with the method of STLN.

2.3 GCD and Sylvester matrices

Suppose that a polynomial d of degree k is the exact GCD of given polynomials $\hat{f}$ and $\hat{g}$ of degrees m and n. Then there exist two polynomials u and v such that

$$\hat{f} = ud,\ \hat{g} = vd, \quad \deg u = m - k,\ \deg v = n - k. \tag{2.1}$$

Polynomial multiplications can be written in matrix-vector notation in the following way:

$$C_{k+1}(u)\mathbf{d} = C_{m-k+1}(d)\mathbf{u} = \hat{\mathbf{f}} \quad \text{and} \quad C_{k+1}(v)\mathbf{d} = C_{n-k+1}(d)\mathbf{v} = \hat{\mathbf{g}}, \tag{2.2}$$

where $C_{k+1}(u) \in \mathbb{R}^{(m+1)\times(k+1)}$, $C_{m-k+1}(d) \in \mathbb{R}^{(m+1)\times(m-k+1)}$, $C_{k+1}(v) \in \mathbb{R}^{(n+1)\times(k+1)}$ and $C_{n-k+1}(d) \in \mathbb{R}^{(n+1)\times(n-k+1)}$ are the Toeplitz matrices defined in Section 1.2. Vectors of the coefficients of polynomials are denoted by small letters in bold,

$$\hat{\mathbf{f}} \in \mathbb{R}^{m+1}, \hat{\mathbf{g}} \in \mathbb{R}^{n+1}, \mathbf{d} \in \mathbb{R}^{k+1}, \mathbf{u} \in \mathbb{R}^{m-k+1} \quad \text{and} \quad \mathbf{v} \in \mathbf{R}^{n-k+1}.$$

By using very simple algebra, equations in (2.1) can be also written in the form

$$S_k(\hat{f}, \hat{g}) \begin{bmatrix} \mathbf{v} \\ -\mathbf{u} \end{bmatrix} = 0, \tag{2.3}$$

with $S_k(\hat{f}, \hat{g}) \in \mathbb{R}^{(m+n-k+1)\times(m+n-2k+2)}$. See Section 4.1 for more details.

The next theorem is crucial for further explanations. The first statement of the theorem is carefully proved, for example, in [22] on p. 186.

Theorem 2.3.1. (Relations between Sylvester matrices and GCD) *Suppose that $\hat{f}$ and $\hat{g}$ are polynomials of degrees m and n, $m \geq n$, and $d = GCD(\hat{f}, \hat{g})$. Then*

i) $rank(S(\hat{f}, \hat{g})) = m + n - k \iff \deg d = k$,

ii) $rank(S_k(\hat{f}, \hat{g})) = m + n - 2k + 1 \iff \deg d = k$,

iii) the coefficient vector $\mathbf{d}$ *is the solution to the linear system*

$$\begin{bmatrix} C_{k+1}(u) \\ C_{k+1}(v) \end{bmatrix} \mathbf{d} = \begin{bmatrix} \hat{\mathbf{f}} \\ \hat{\mathbf{g}} \end{bmatrix}. \tag{2.4}$$

Note that $S_k = S_k(\hat{f}, \hat{g})$ is rank-deficient by 1 or $\dim(\text{Null}(S_k)) = 1$.

Let $S_k = U_k \Sigma_k V_k^T$ be the SVD decomposition of S_k and denote the columns of V_k as $\left[\mathbf{v_1^k}, \ldots, \mathbf{v_{m+n-2k+2}^k}\right]$. Then it immediately follows from the theorem that the null space of S_k is spanned by the right singular vector $\mathbf{v_{m+n-2k+2}^k}$ associated with the zero singular value of S_k. Equation (2.3) then results in

$$\begin{bmatrix} \mathbf{v} \\ -\mathbf{u} \end{bmatrix} \in \langle \mathbf{v_{m+n-2k+2}^k} \rangle = \text{Null}(S_k). \tag{2.5}$$

Hence, the coefficients $\mathbf{u}$ and $\mathbf{v}$ of the cofactors u and v can be obtained from $\mathbf{v_{m+n-2k+2}^k}$. The coefficients $\mathbf{d}$ of the GCD d can be computed afterwards as a solution to (2.4).

We still have to keep in mind that all these computations are valid only if the polynomials are given exactly and computations are performed symbolically. Since in a real computation a zero singular value becomes non-zero and $\mathbf{v_{m+n-2k+2}^k}$ is computed only approximately, the computed vector $\mathbf{d}$, and so d is only an approximation of the GCD. It can be therefore considered to be the AGCD, however, a refinement, e.g. the

Gauss-Newton iteration in [30], should be always used to "repair loss of quality" caused by rounding errors.

Hence, the AGCD computation is now very straightforward. We have to estimate the numerical rank of the Sylvester matrix S and/or construct/find the rank deficient matrix S_k of the nullity 1, compute the corresponding right singular vector associated with the smallest singular value, extract the coefficients of cofactors and solve the system (2.4). The numerical rank computation of the Sylvester matrix of the polynomials needs a specified threshold to be applied on the singular values of the matrix. To avoid doing the SVD of the Sylvester matrix, the property *ii)* from the theorem can be used in a different way.

Let $\sigma_1, \sigma_2, \ldots, \sigma_{m+n-2k+2}$ be the singular values of S_k. If S_k is rank deficient by 1, then the singular values satisfy

$$\sigma_1(S_k) \geq \sigma_2(S_k) \geq \cdots \geq \sigma_{m+n-2k+1}(S_k) > \theta \geq \sigma_{m+n-2k+2}(S_k) \equiv \sigma_{\min}(S_k)$$

for the threshold θ from the numerical rank definition. Theorem 2.3.1 says that the degree of the AGCD can be readily obtained from the rank deficient matrix by 1. However, AGCD has the *maximum degree* property and since we assume that $n \leq m$, then $k = n$ is the index of the first Sylvester subresultant that has to be inspected. If $\sigma_{\min}(S_n) \leq \theta$, then S_n is rank deficient and n is the degree of the AGCD. Otherwise, the value of $k = n-1$ is considered, *etc.* Hence, we have to find the first rank deficient matrix in the sequence of the Sylvester matrices $S_n, S_{n-1}, \ldots, S_1 = S(\hat{f}, \hat{g})$ through the inspection of their smallest singular values. If there is no $k \in \{1, 2, \ldots, n\}$ such that $\sigma_{\min}(S_k) \leq \theta$ then $\hat{f}$ and $\hat{g}$ are said to be coprime or, possibly, θ is set to be too small.

In addition, the smallest singular values of the sequence of the Sylvester matrices $S_n, S_{n-1}, \ldots, S_1$ fulfil inequalities from the following lemma.

Lemma 2.3.2. *Let the rank of the Sylvester matrix S be r within the threshold θ. Then the smallest singular values of the Sylvester subresultant matrices $S_n, S_{n-1}, \ldots, S_r$ satisfy*

$$\sigma_{\min}(S_n) \geq \sigma_{\min}(S_{n-1}) \geq \cdots \geq \sigma_{\min}(S_{r+1}) > \theta \geq \sigma_{\min}(S_r).$$

Proof: The proof is very straightforward since

$$\sigma_{\min}(S_j) = \min_{\substack{\|\mathbf{x}\|=1 \\ \mathbf{x} \in \mathbb{R}^{m+n-2j+2}}} \|S_j \mathbf{x}\|.$$

Multiplying S_{j-1} by a suitable permutation matrix P, columns of S_{j-1} can be reordered so that

$$S_{j-1} = P \left[\begin{array}{ccc|c} & & & \mathbf{f} \\ & S_j & & \\ & & & \mathbf{g} \\ \hline 0 & \cdot & \cdot \;\; \cdot \;\; 0 & \end{array}\right].$$

Hence, we search the minimum $\sigma_{\min}(S_{j-1}) = \min_{\|\mathbf{y}\|=1} \|S_{j-1}\mathbf{y}\|$ over the larger space $\mathbb{R}^{m+n-2j+4}$ and $\sigma_{\min}(S_{j-1})$ can be therefore smaller or equal to $\sigma_{\min}(S_j)$ for all $j = n, n-1, \ldots, r+1$. Because S_r is the first rank deficient Sylvester submatrix, then $\sigma_{\min}(S_{r+1}) > \theta \geq \sigma_{\min}(S_r)$. ■

Next Section 2.4 describes a method for the computation of the smallest singular value of a matrix and the corresponding right singular vector. Moreover, it enables us to compute all singular values and so determine the numerical rank of a matrix.

2.4 Rank revealing algorithm

This section contains a basic theory that leads to the computation of singular values of a general matrix $A \in \mathbb{R}^{m\times n}$, $m \geq n$. This part is essential for the numerical rank estimation of Sylvester (subresultant) matrices, and so the degree of the AGCD.

Let $A = U\Sigma V^T$ be the SVD of A with singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$. If $\mathbf{u_n}$ and $\mathbf{v_n}$ are the left and the right singular vector associated with $\sigma_{\min} = \sigma_n$, $\|\mathbf{u_n}\| = \|\mathbf{v_n}\| = 1$ and $A\mathbf{v_n} = \sigma_{\min}\mathbf{u_n}$, then

$$\sigma_{\min} = \|A\mathbf{v_n}\|.$$

The smallest singular value $\sigma_{\min}$ can be then computed as

$$\sigma_{\min} = \min_{\|\mathbf{x}\|=1} \|A\mathbf{x}\|.$$

This problem of finding $\sigma_{\min}$ can be transformed to solving the weighted system

$$\begin{bmatrix} \tau\mathbf{x}^T \\ A \end{bmatrix} \mathbf{x} = \begin{bmatrix} \tau \\ \mathbf{0} \end{bmatrix}, \tag{2.6}$$

for $\tau > \sigma_n$. System (2.6) is solved by the Gauss-Newton iteration [3, 17, 31].

Before solving (2.6), a helpful proposition that confirms the relation between $\sigma_{\min}$ and a solution of (2.6) is stated. Proposition is formulated and proved in [17]. Note that from now on, singular vectors are always meant to be right singular vectors.

Proposition 2.4.1. *Let $\mathbf{u} \in \mathbb{R}^n$ be the least square solution of (2.6), i.e.*

$$\left\| \begin{bmatrix} \tau\mathbf{u}^T \\ A \end{bmatrix} \mathbf{u} - \begin{bmatrix} \tau \\ \mathbf{0} \end{bmatrix} \right\|^2 = \min_{\mathbf{x}\in\mathbb{R}^n} \left\| \begin{bmatrix} \tau\mathbf{x}^T \\ A \end{bmatrix} \mathbf{x} - \begin{bmatrix} \tau \\ \mathbf{0} \end{bmatrix} \right\|^2$$

for $\tau > \sigma_n$. Then $\mathbf{u}$ is an element of the subspace $\mathbf{W}$ spanned by the singular vector, or vectors, of A associated with the smallest singular value, respectively values.[1]

A method, that can be used for finding a minimum of

$$\mathcal{F}(\mathbf{x}) = \left\| \begin{bmatrix} \tau\mathbf{x}^T \\ A \end{bmatrix} \mathbf{x} - \begin{bmatrix} \tau \\ \mathbf{0} \end{bmatrix} \right\|^2 = \left(\begin{bmatrix} \tau\mathbf{x}^T \\ A \end{bmatrix} \mathbf{x} - \begin{bmatrix} \tau \\ \mathbf{0} \end{bmatrix} \right)^T \left(\begin{bmatrix} \tau\mathbf{x}^T \\ A \end{bmatrix} \mathbf{x} - \begin{bmatrix} \tau \\ \mathbf{0} \end{bmatrix} \right)$$

can be found in [3] chap. 9 and it is known as the *Gauss-Newton* method.

[1] It may happen that some last singular values are equal to each other, i.e. $\sigma_{n-j} = \sigma_{n-j+1} = \cdots = \sigma_n$ for some j. Then $\mathbf{W} = \langle \mathbf{v_{n-j}}, \ldots, \mathbf{v_n} \rangle$.

In general, for m non-linear functions $r_i(\mathbf{x})$, $i = 1, \ldots, m$, $r(\mathbf{x}) \equiv [r_1, \ldots, r_m]^T$, $\mathbf{x} = [x_1, \ldots, x_n]^T \in \mathbb{R}^n$, let us define a function $F(x)$,

$$F(\mathbf{x}) = \frac{1}{2} r(\mathbf{x})^T r(\mathbf{x}) = \frac{1}{2} \sum_{i=1}^{m} r_i(\mathbf{x})^2.$$

A minimum of $F(\mathbf{x})$ computed by the Gauss-Newton method can be reached iteratively as a limit of linear approximations $\{\mathbf{x}_i\}_i$, where

$$\mathbf{x}_{i+1} = \mathbf{x}_i - J(\mathbf{x}_i)^\dagger r(\mathbf{x}_i). \tag{2.7}$$

The matrix $J(\mathbf{x}) \in \mathbb{R}^{m \times n}$ is the Jacobian of $r(\mathbf{x})$, i.e.

$$J(\mathbf{x}) = \begin{bmatrix} \frac{\partial r_1}{\partial x_1} & \frac{\partial r_1}{\partial x_2} & \cdots & \frac{\partial r_1}{\partial x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial r_m}{\partial x_1} & \frac{\partial r_m}{\partial x_2} & \cdots & \frac{\partial r_m}{\partial x_n} \end{bmatrix}.$$

Hence, a minimum of the function of our interest $\mathcal{F}(\mathbf{x})$ can be computed by the iteration (2.7) with

$$r(\mathbf{x}) = \begin{bmatrix} \tau \mathbf{x}^T \\ A \end{bmatrix} \mathbf{x} - \begin{bmatrix} \tau \\ \mathbf{0} \end{bmatrix}$$

and

$$J(\mathbf{x}) = \begin{bmatrix} 2\tau \mathbf{x}^T \\ A \end{bmatrix}$$

as it can be easily verified.

Suppose that the smallest singular value σ_n of A is simple, $\sigma_n < \sigma_{n-1}$. Then by recalling Proposition 2.4.1, the Gauss-Newton iteration

$$\left.\begin{aligned} \mathbf{x}_{i+1} &= \mathbf{x}_i - \begin{bmatrix} 2\tau \mathbf{x}_i^T \\ A \end{bmatrix}^\dagger \begin{bmatrix} \tau \mathbf{x}_i^T \mathbf{x}_i - \tau \\ A\mathbf{x}_i \end{bmatrix} \\ \zeta_{i+1} &= \frac{\|A\mathbf{x}_{j+1}\|}{\|\mathbf{x}_{j+1}\|} \end{aligned}\right\} \tag{2.8}$$

gives the sequence $\{\mathbf{x}_i\}_i$ converging to the singular vector $\mathbf{v_n}$ associated with σ_n, and the sequence of numbers $\{\zeta_i\}_i$ approaching the smallest singular value σ_n. If σ_n is not simple, then $\{\mathbf{x}_i\}_i$ converges to a vector that is a linear combination of the singular vectors associated with the smallest singular values that are equal to each other, i.e. $\mathbf{x}_i \to_{i \to \infty} \mathbf{w} \in \mathbf{W}$.

It is shown in [3, 17] that the Gauss-Newton method is globally convergent and the convergent rate is at least linear.

Next theorem proved in [17] induces a way of computing all singular values of A.

Proposition 2.4.2. *Let $A = U\Sigma V^T$ be the SVD of m by n matrix A, $m \geq n$, with the singular values $\sigma_1 \geq \cdots \geq \sigma_n$. Suppose that A has numerical rank r, i.e. $\sigma_r > \theta \geq \sigma_{r+1}$ for the given threshold θ. Then for a unit vector $\mathbf{w} \in \mathbf{W}$, $\|\mathbf{w}\| = 1$, the matrix*

$$B = \begin{bmatrix} \rho \mathbf{w}^T \\ A \end{bmatrix}$$

with $\rho \geq \sigma_r$ has singular values $\{\sigma'_j\}_{j=1}^n$ satisfying

$$\sigma'_1 \geq \cdots \geq \sigma'_{k+1} \geq \sigma_r > \sigma_{r+1} \geq \sigma'_{k+2} \geq \cdots \geq \sigma'_n$$

and its null space $\mathbf{W}'$ spanned by the singular vectors of B associated with $\sigma'_{k+2}, \cdots, \sigma'_n$ is a subspace of $\mathbf{W}$.

Moreover, if $\rho = \|A\|_F = \sqrt{\sigma_1^2 + \cdots + \sigma_n^2}$ or $\rho = \|A\|_2 = \sigma_1$ and $\mathbf{w} = \mathbf{v_n}$, then B has the singular values $\sqrt{\rho^2 + \sigma_n^2} \geq \sigma_1 \geq \ldots \geq \sigma_{n-1}$, i.e. B has exactly the same singular values as A, except where the smallest singular value of A, σ_n, is replaced with $\sqrt{\rho^2 + \sigma_n^2}$ which becomes the largest singular value of B.

The Gauss-Newton iteration applied to B gives the smallest singular value of B and the associated singular vector. This singular value and singular vector are then the second smallest singular value of A and the corresponding singular vector, respectively. It is obvious that now all singular values of A can be computed by repeating the same processes. If the numerical rank of A is desired to be determined within a specified threshold θ, then only singular values less or equal to θ have to be computed.

Computationally it is more convenient to compute singular values by the algorithm described above than by using the SVD. It is shown in [17] that computation of singular values of A by using the Gauss-Newton iteration needs $\mathcal{O}(n^2)$ flops.

The whole rank revealing algorithm for a given threshold θ is summarised in the following Algorithm 2.4.1. Note that the Moore-Penrose inverse in (2.7) is not computed explicitly but the linear system

$$\begin{bmatrix} 2\tau\mathbf{x}_i^T \\ A \end{bmatrix} \Delta_i\mathbf{x} = \begin{bmatrix} \tau\mathbf{x}_i^T\mathbf{x}_i - \tau \\ A\mathbf{x}_i \end{bmatrix}$$

is solved instead. Then new iteration is then defined by $\mathbf{x}_{i+1} = \mathbf{x}_i + \Delta_i\mathbf{x}$. Note also that the algorithm computes not only singular values but a basis of the null space of A.

Algorithm 2.4.1 (RankRev)**.**

Input: Matrix $A \in \mathbb{R}^{m\times n}$, threshold $\theta > 0$.
Output: Numerical rank k of A with respect to θ, basis $\{\mathbf{w}_{k+1}, \ldots, \mathbf{w}_n\}$ of the null space of A.

(1) **begin**
(2) **initialization**
(3) compute the QR decomposition $A = QR$;
(4) set $\tau = \|A\|_F$;
(5) **for** $k = n, n-1, \ldots, 1$ **do**
(6) generate a random vector $\mathbf{x}_0, \|\mathbf{x}_0\| = 1$;
(7) **for** $j = 0, 1, \ldots,$ ***until convergence*** **do**
(8) $D = \begin{bmatrix} 2\tau\mathbf{x}_j^T \\ R \end{bmatrix}, \mathbf{b} = \begin{bmatrix} \tau\mathbf{x}_j^T\mathbf{x}_j - \tau \\ R\mathbf{x}_j \end{bmatrix}$;
(9) QR decomposition $D = Q_1R_1$;
(10) solve $R_1\Delta_j\mathbf{x} = Q_1^T\mathbf{b}$;

```
(11)            set x_{j+1} = x_j + Δ_j x;
(12)                ζ_{j+1} = ‖R_1 x_{j+1}‖/‖x_{j+1}‖;
(13)        end
(14)        if ζ_{j+1} > θ then
(15)          goto line (24)
(16)        else
(17)            σ_k = ζ_{j+1};
(18)            w_k = x_{j+1}/‖x_{j+1}‖;
(19)            QR decomposition [τ w_k^T; R] = Q_2 R_2;
(20)            set R = R_2;
(22)        end
(23)    end
(24) end
```

Producing new approximations ζ_j of a single singular value σ_k terminates when ζ_j converges to σ_k. However, we do not need to have singular values computed exactly. To find the numerical rank of a matrix A, it is sufficient to know whether computed approximations of singular values are smaller than the threshold θ. Hence, a ***until convergence*** condition that has to be specified on the line *(7)* can be replaced by the stopping criterion of ζ_j being smaller than θ for some j.

Note that if $A = QR$ is the QR decomposition of A, then we can replace A in (2.8) by R. We can do this because the 2-norm is orthogonally invariant and Q is unitary, $Q^TQ = I$.

Application of the algorithm is shown in the following section.

2.5 Examples of numerical rank estimation

The RankRev algorithm is demonstrated on several examples.

Example 2.5.1. In this example we consider the polynomials (1.20), i.e.

$$\begin{aligned}\hat{f}(x) &= \prod_{i=1}^{\frac{m}{2}} \left[(x - r_1\alpha_i)^2 + r_1^2\beta_i^2\right] \prod_{i=\frac{m}{2}+1}^{m} \left[(x - r_2\alpha_i)^2 + r_2^2\beta_i^2\right], \\ \hat{g}(x) &= \prod_{i=1}^{m} \left[(x - r_1\alpha_i)^2 + r_1^2\beta_i^2\right]\end{aligned} \tag{2.9}$$

for an even integer m with $\alpha_i = \cos\left(\frac{\pi i}{m}\right), \beta_i = \sin\left(\frac{\pi i}{m}\right), i = 1, \ldots, n, r_1 = 0.5$ and $r_2 = 1.5$. Both polynomials are of degree $2m$ with the exact GCD of degree m.

Consider $m = 16$. The RankRev algorithm 2.4.1 applied to the Sylvester matrix of the polynomials for $m = 16$ gives exactly 16 approximations of the 16 smallest singular values of $S(\hat{f}, \hat{g})$. This can be seen from Figure 2.1 where the computed approximations of the smallest singular values $\sigma_{64}, \ldots, \sigma_{49}$ smaller than $\theta = \varepsilon_{\text{mach}}\|S\| \approx 1.1 \times 10^{-11}$

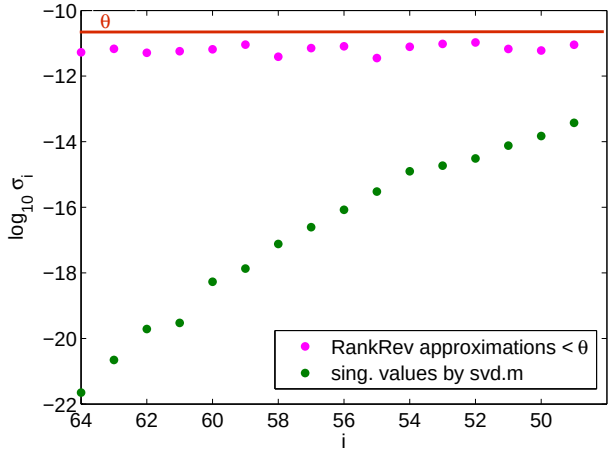


Figure 2.1: Approximations of the singular values of $S(\hat{f},\hat{g})$ that are less than θ computed by the RankRev algorithm 2.4.1, •, and the singular values computed by the svd.m, •.

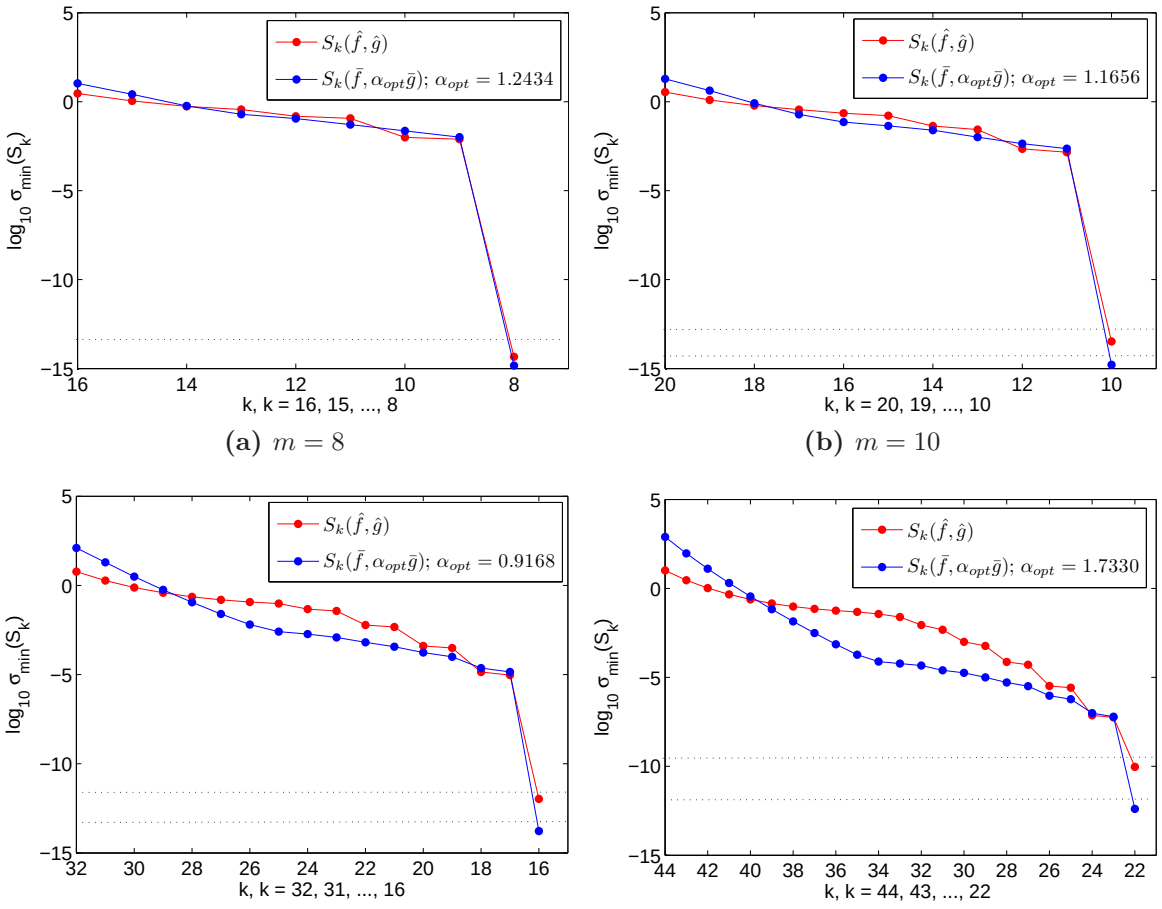
are plotted, $\sigma_{48} > \theta$. The singular values are compared with the values returned by the Matlab function *svd.m*. We can see big differences, however note that computing singular values in the RankRev algorithm runs until a first approximation smaller than the threshold θ appears.

As a conclusion we get that the RankRev algorithm returns the numerical rank of $S(\hat{f},\hat{g})$ equal 48 that is the exact rank of $S(\hat{f},\hat{g})$. By recalling Theorem 2.3.1 we then have that the degree of the AGCD is 16 that is the degree of the exact GCD. Computation of the coefficients is given in the next chapter.

The modified RankRev algorithm for the rank estimation computes the rank of a matrix through the inspection of the smallest singular values of the sequence of matrices $S_{2m}, S_{2m-1}, \ldots, S_1$. In Figure 2.2 the smallest singular values of matrices $S_{2m}, S_{2m-1}, \ldots, S_1$ are plotted for $m = 8, 10, 16$ and 22. The threshold θ is always chosen to be $\theta = \varepsilon_{\text{mach}}\|S(\hat{f},\hat{g})\|$. Figures also show the singular values of the Sylvester matrices of preprocessed polynomials by the operations from Section 1.3 yielding the polynomials $\bar{f}$ and $\bar{g}$ and the optimal values of α_{opt} and γ_{opt}.

Remind that in the modified RankRev algorithm the matrix S_{2m} is firstly formed and its smallest singular value $\sigma_{\min}(S_{2m})$ and the corresponding singular vector $\mathbf{v}_{\min}(S_{2m})$ then computed. If $\sigma_{\min}(S_n) <= \theta$ for the given threshold θ, then the rank deficient matrix is found. Otherwise, S_{2m-1} is considered. Note that only one full QR decomposition of S_{2m} is computed. The QR decomposition of the following S_{2m-1} is embedded from the computed decomposition by the updating technique described in [30].

This approach has the advantage that we immediately have the singular vector associated with the smallest singular value of the rank deficient matrix used for the AGCD computation. Applied RankRev algorithm to $S(\hat{f},\hat{g})$ computes firstly the nullity of $S(\hat{f},\hat{g})$ and the basis of its null space. However, we still have to form the corresponding Sylvester subresultant with the subscript that is equal to the nullity and then compute the smallest singular pair including right singular vector. A method for the AGCD computation can be used afterwards.



(a) $m = 8$ **(b)** $m = 10$

(c) $m = 16$ **(d)** $m = 22$

Figure 2.2: The smallest singular values of the sequence $S_{2m}, S_{2m-1}, \ldots, S_m$ of the polynomials $\hat{f}$ and $\hat{g}$ defined in (2.9) before, ●, and after the preprocessing operations applied to the coefficients, ●, $m = 8, 10, 16, 22$. The matrix S_m is the first rank deficient Sylvester matrix.

We can see from Figure 2.2 that the thresholds $\varepsilon_{\text{mach}}\|S(\hat{f},\hat{g})\|$ and $\varepsilon_{\text{mach}}\|S(\bar{f},\alpha_{opt}\bar{g})\|$, i.e. values identified by the dashed lines, are sufficient for the computation of the numerical ranks that correspond to the exact ranks of matrices. Moreover, it is seen that the preprocessing operations make numerical rank better defined since the rank deficient matrices can be identified for smaller thresholds in the case of preprocessed polynomials. Importance of the preprocessing operations is mainly shown in the next example. ♣

Example 2.5.2. Consider the exact polynomials $\hat{f}$ and $\hat{g}$ from Example 1.3.2,

$$\begin{aligned}\hat{f}(x) &= (x+8.755)^9(x+3.921)^{10}(x-1.974)^9(x+7.496)^6,\\ \hat{g}(x) &= (x+8.755)^8(x+3.921)^5(x-7.688)^8.\end{aligned} \tag{2.10}$$

The exact GCD of $\hat{f}$ and $\hat{g}$ is of degree 13, hence the rank of $S(\hat{f},\hat{g}) = 42$ by Theorem 2.3.1 and $S(\hat{f},\hat{g}) \in \mathbb{R}^{55\times 55}$. Applying the RankRev algorithm for the numerical

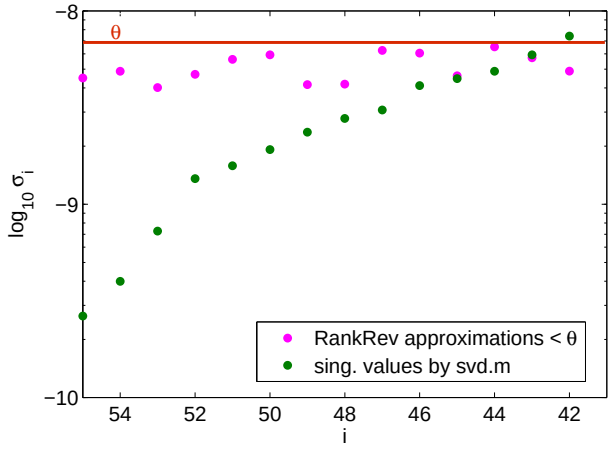


Figure 2.3: Approximations of the singular values of $S(\hat{f}, \hat{g})$ that are smaller than θ computed by the RankRev algorithm 2.4.1, •, and the singular values computed by the svd.m, •.

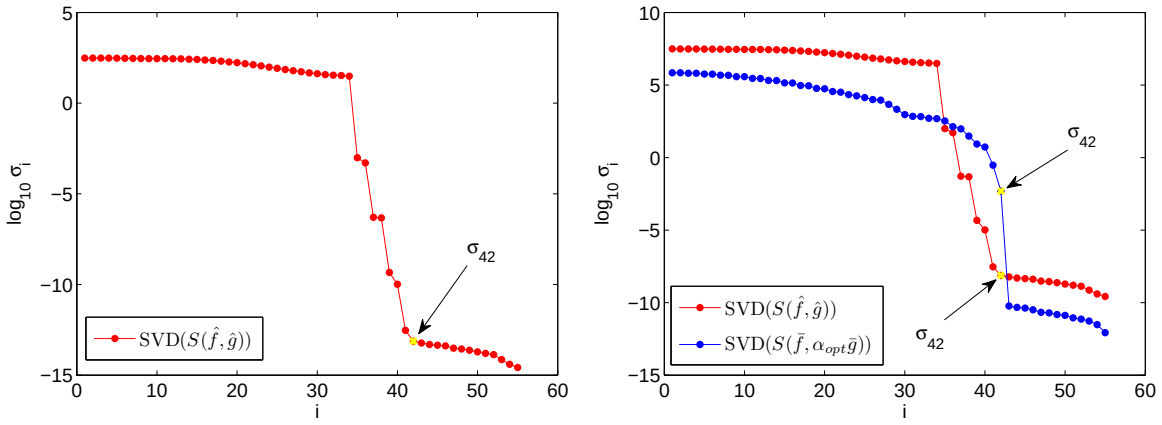
rank estimation within the threshold $\theta = \varepsilon_{\text{mach}}\|S(\hat{f}, \hat{g})\| \approx 6.85 \times 10^{-9}$ should give approximations of the 13 smallest singular values below θ. Figure 2.3 however shows that the RankRev algorithm returns 14 approximations of the singular values $\sigma_{55}, \ldots, \sigma_{41}$ less then θ. This implies that the numerical rank computed by the RankRev algorithm is 41, and so the degree of the AGCD is 14 — that is still valid with respect to the definition of the AGCD. Using the standard Matlab function *svd.m* gives the correct number 13 of singular values.

Explanation can be clear from the singular values of $S(\hat{f}, \hat{g})$ plotted in Figure 2.4a. In this figure we can see that the singular values $\sigma_{41}, \ldots, \sigma_{55}$ are very close to each other. Hence, the dependence of numerical rank on θ is significant in this case and slightly different θ may give the different numerical rank.

The RankRev algorithm is applied to the exact polynomials that are only normalised by the geometric mean. If we now consider the preprocessing operations from Section 1.3, then a remarkable improvement can be seen.[2] Applying normalisation and scaling returns polynomials defined in 1.16, and the values α_{opt} and γ_{opt}. Then if we look at the singular values of $S(\bar{f}, \alpha_{opt}\bar{g})$ in Figure 2.4b, we can see that the numerical-rank gap now clearly separates the correct number 13 of singular values from the others. The RankRev algorithm estimates the smallest 13 singular values counted to the matrix's nullity within $\theta = \varepsilon_{\text{mach}}\|S(\bar{f}, \alpha_{opt}\bar{g}\| \approx 1.55 \times 10^{-10}$, as can be seen from Figure 2.5.

We can also see from Figure 2.3 and Figure 2.5 that approximations given by the RankRev algorithm differs from the singular values computed by *svd.m*. This is caused by the stopping criterion in the algorithm where the computation of a single singular value terminates whenever an approximation smaller than θ appears. ♣

[2]The coefficients before and after preprocessing give similar figure as Figure 1.1. Preprocessing also brings reduction of condition number $\kappa(S(\hat{f}, \hat{g})) \approx 10^{21}$ to $\kappa(S(\bar{f}, \alpha_{opt}\bar{g})) \approx 10^{17}$, where $\bar{f}$ and $\bar{g}$ are the preprocessed polynomials defined in (1.16), $\alpha_{opt} = 4.41 \times 10^3$ is the value computed from the LP problem (1.12).



(a) SVD of $S(\hat{f},\hat{g})$ **(b) SVD of $S(\hat{f},\hat{g})$ and $S(\bar{f},\alpha_{opt}\bar{g})$**

Figure 2.4: (a) The singular values of $S(\hat{f},\hat{g})$ where $\hat{f}$ and $\hat{g}$ are the polynomials in (2.10) normalised by the geometric mean, •. (b) The singular values of $S(\hat{f},\hat{g})$, •, and $S(\bar{f},\alpha_{opt}\bar{g})$, •, where $\hat{f}$ and $\hat{g}$ are the polynomials in (2.10) normalised by the geometric mean and $\bar{f}$ and $\bar{g}$ are the preprocessed polynomials in (1.16). The singular value σ_{42} • denotes the last singular value that is counted to the numerical rank, i.e. σ_{43} should be the first singular value less than θ.

Example 2.5.3. From the previous example we know that the numerical rank estimation can fail by using the RankRev algorithm. The preprocessing operations may bring some improvements. However, it is important to emphasize that failure is usually caused by difficulties of choosing appropriate thresholds. Moreover, if a noise of unknown amplitude is involved in the coefficients of polynomials, then selecting thresholds to the algorithm is a non-trivial problem. Consider very simple polynomials

$$\begin{aligned}\hat{f}(x) &= (x+20.6)^2(x-4.7)^5(x-1.3)^4,\\ \hat{g}(x) &= (x+10.4)^3(x-4.7)^4(x-1.3)^3\end{aligned} \tag{2.11}$$

of degrees $m = 11$ and $n = 10$ with the GCD of degree 7. The Sylvester matrix $S(\hat{f},\hat{g})$ has rank equal to 14.

The numerical rank computed by the RankRev algorithm 2.4.1 is 14 within $\theta = \varepsilon_{\text{mach}}\|S(\hat{f},\hat{g})\| \approx 2.78\times 10^{-13}$. Problems however arise when the polynomials are perturbed componentwisely by adding of a noise of the signal-to-noise ratio $\epsilon = 10^{-8}$ yielding the polynomials f and g. Polynomials returned by the preprocessing operations applied to f and g are denoted by $\bar{f}$ and $\bar{g}$, respectively.

The singular values of the three matrices $S(\hat{f},\hat{g})$, $S(f,g)$ and $S(\bar{f},\alpha_{opt}\bar{g})$ are plotted in Figure 2.6. We can see that the seven smallest singular values of $S(\hat{f},\hat{g})$ are well separated and the RankRev algorithm successfully terminates. However, noise in the polynomials shifts the singular values so that the numerical rank computed by the RankRev algorithm is 19 within $\theta = \varepsilon_{\text{mach}}\|S(f,g)\| \approx 2.78\times 10^{-13}$. This can be seen from the same Figure 2.6 where only two singular values are below θ. Hence, the AGCD of inexact polynomials f and g is of degree 2. Situation does not get better when the

polynomials are preprocessed. Even more, the preprocessing operations return the polynomials $\bar{f}$ and $\bar{g}$, the Sylvester matrix of which has the singular values all greater than $\theta_1 = \varepsilon_{\text{mach}}\|S(\bar{f}, \alpha_{opt}\bar{g})\| \approx 1.94 \times 10^{-14} < \theta$ for $\alpha_{opt} = 1.5814$.

Of course, one can change the threshold θ so that the numerical rank of the Sylvester matrix of perturbed polynomials or preprocessed polynomials will be equal to 14. However, we would like to construct an AGCD solver that is θ independent.

Note that the signal-to-noise ratio ϵ is not known in practice. However, if we have an information about ϵ that $\theta = \epsilon\|S\|$ can be successfully used. In addition, the numerical-rank gap criterion gives the correct numerical rank 14 for the preprocessed polynomials, i.e. the same as the exact rank of $S(\hat{f}, \hat{g})$. While in the case of the inexact polynomials this criterion gives 13 that is incorrect.

We can make a conclusion that the RankRev algorithm works well for the exact polynomials considering only the rounding errors. For a complicated polynomials the preprocessing operations may be beneficial. Unsatisfactory results may be obtained whenever the noise is involved in the polynomials and its level is not known. ♣

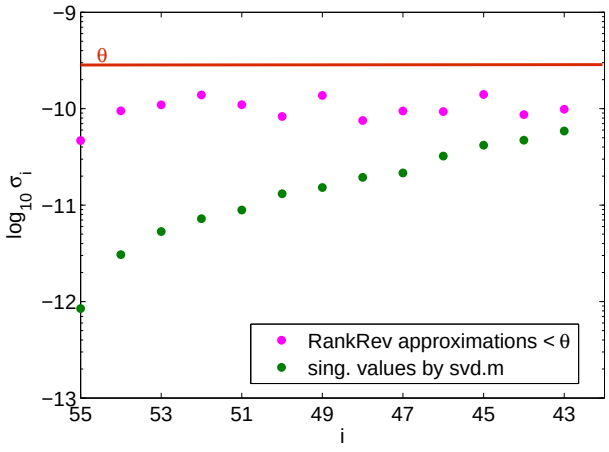


Figure 2.5: Approximations of the singular values of $S(\bar{f}, \alpha_{opt}\bar{g})$ that are less than θ computed by the RankRev algorithm 2.4.1, •, and the singular values computed by the svd.m, •. Polynomials $\bar{f}$ and $\bar{g}$ are the preprocessed forms of the polynomials in (2.10).

Summary

The main motivation of the presented chapter has been to introduce the term approximate polynomial GCD and the essential relations between the AGCD and the Sylvester matrices. In particular, it has been said that the degree of the AGCD of given polynomials is equal to the rank loss of the corresponding Sylvester matrix.

Importance of the rank estimation has resulted in the rank revealing Algorithm 2.4.1 that can be modified for the purpose of computing first rank deficient Sylvester subresultant matrix. The singular vector associated to the smallest singular value of the

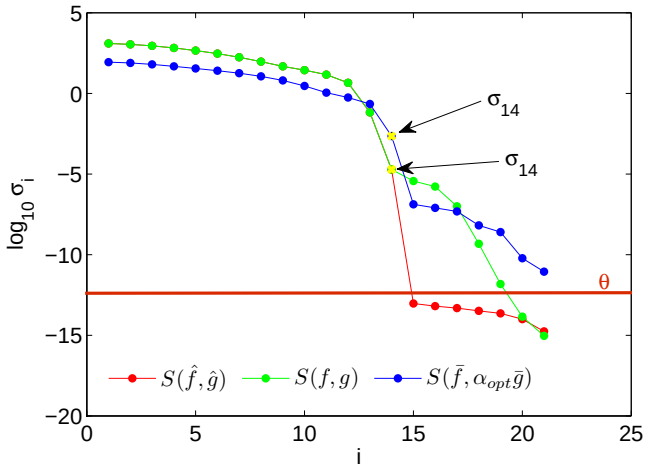


Figure 2.6: The singular values of $S(\hat{f}, \hat{g})$ where $\hat{f}$ and $\hat{g}$ are the polynomials in (2.11) normalised by the geometric mean, •, the singular values of $S(f, g)$ where f and g are perturbed polynomials by the signal-to-noise ratio $\epsilon = 10^{-8}$ and normalised by the geometric mean, •, and the singular values of $S(\bar{f}, \alpha_{opt}\bar{g})$ where $\bar{f}$ and $\bar{g}$ are the preprocessed polynomials, •. The threshold $\theta \approx 2.78 \times 10^{-13}$.

rank deficient matrix can be then used for the computation of the coefficients of the AGCD.

We have however shown that the computation of the numerical rank is non-trivial problem, since it depends strongly on the chosen threshold that decides whether a singular value still counts to the numerical rank or it is already small enough for the rank deficiency.

Moreover, including noise in the polynomials makes estimation very difficult.

Chapter 3

AGCD refinement

It is said in the general discussion on GCDs and Sylvester matrices in Section 2.3 that the coefficients of the AGCD can be computed from the singular vector that is associated with the smallest singular value of the first rank deficient matrix S_k. A modified version of the RankRev algorithm 2.4.1 from the previous Section 2.4 can be used to compute the singular pair $\{\sigma_{\min}(S_k), \mathbf{v}_{\min}(S_k)\}$.

The computed approximation of the AGCD obtained by this way can be further refined. In the next section a refinement of an approximation of the AGCD in the form of a least square solver is explained and then examined on some examples.

3.1 Gauss-Newton method

Recall from the previous sections that the AGCD computation consists of two steps: the numerical rank estimation, that can be determined from the sequence of the smallest singular values of the Sylvester subresultant matrices, and the computation of the coefficients of the AGCD. The modified RankRev algorithm is applied to the sequence $S_n, S_{n-1}, \ldots, S_1$ yielding the first rank deficient matrix S_k and its singular pair $\{\sigma_{\min}(S_k), \mathbf{v}_{\min}(S_k)\}$. The Gauss-Newton iteration (3.5) is then employed in the second step.

Suppose that there are given two polynomials $\hat{f}$ and $\hat{g}$ with a non-trivial GCD d of degree k. Then there exist polynomials u and v so that $\hat{f} = ud$ and $\hat{g} = vd$. We have already seen in Section 2.3 that these products can be written as

$$C_{k+1}(u)\mathbf{d} = \hat{\mathbf{f}} \quad \text{and} \quad C_{k+1}(v)\mathbf{d} = \hat{\mathbf{g}},$$

with the Toeplitz matrices $C_{k+1}(u)$ and $C_{k+1}(v)$.

For a scaling vector $\mathbf{r} \in \mathbb{R}^{k+1}$ satisfying $\mathbf{r}^T\mathbf{d} \neq 0$ define the system

$$F(\mathbf{z}) = \mathbf{b}, \tag{3.1}$$

where

$$F(\mathbf{z}) = \begin{bmatrix} \mathbf{r}^T\mathbf{d} - 1 \\ C_{k+1}(u)\mathbf{d} \\ C_{k+1}(v)\mathbf{d} \end{bmatrix}, \quad \mathbf{z} = \begin{bmatrix} \mathbf{d} \\ \mathbf{u} \\ \mathbf{v} \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 0 \\ \hat{\mathbf{f}} \\ \hat{\mathbf{g}} \end{bmatrix} \tag{3.2}$$

and

$$\mathbf{z} \in \mathbb{R}^{m+n-k+3}, \mathbf{b} \in \mathbb{R}^{m+n+3}, F(\mathbf{z}) \in \mathbb{R}^{m+n+3}.$$

The scaling vector $\mathbf{r}$ can be chosen by several ways, for example, as a random vector, the canonical vector $\mathbf{e}_1$ or as a solution to (2.4) for $\mathbf{u}$ and $\mathbf{v}$ extracted from the singular vector $\mathbf{v}_{\min}(S_k)$ associated with the smallest singular value $\sigma_{\min}(S_k)$ of the first rank deficient matrix S_k. Let us denote the solution to (2.4) by $\mathbf{d}_0$ and the polynomial with these coefficients by d_0. If we assume that d_0 is close to the GCD, then d_0 cannot be orthogonal to the GCD and so $\mathbf{d}_0^T\mathbf{d} \neq 0$. In the thesis we consider $\mathbf{r} = \mathbf{d}_0/\|\mathbf{d}_0\|^2$. A discussion about choosing $\mathbf{r}$ is given in more details in [30].

The system (3.1) is overdetermined except for the case $k = 0$ since it consists of $m+n+3$ equations of $m+n-k+3$ variables. That is why $\mathbf{z}$ is computed as a minimiser of $\|F(\mathbf{z}) - \mathbf{b}\|^2$ in the least square sense.

However, $\mathbf{z}$ as a minimum of $G(\mathbf{z}) = \|F(\mathbf{z}) - \mathbf{b}\|^2 \colon \mathbb{R}^{m+n-k+3} \to \mathbb{R}^1$ satisfies the gradient equation $\nabla G(\mathbf{z}) = \mathbf{0}^T$, i.e.

$$\nabla G(\mathbf{z}) = (F(\mathbf{z}) - \mathbf{b})^T J(\mathbf{z}) = \mathbf{0}^T,$$

where $J(\mathbf{z})$ is the Jacobian of $F(\mathbf{z})$. Hence, if $F(\mathbf{z})$ attains its minimum in $\mathbf{z}$, then

$$J(\mathbf{z})^T(F(\mathbf{z}) - \mathbf{b}) = \mathbf{0}. \tag{3.3}$$

The Jacobian $J(\mathbf{z})$ is $(m+n+3)$ by $(m+n-k+3)$ matrix of a special form shown in the following lemma.

Lemma 3.1.1. *The Jacobian of $F(\mathbf{z})$ is the matrix*

$$J(\mathbf{z}) = \begin{bmatrix} \mathbf{r}^T & & \\ C_{k+1}(u) & C_{m-k+1}(d) & \\ C_{k+1}(v) & & C_{n-k+1}(d) \end{bmatrix} \in \mathbb{R}^{(m+n+3)\times(m+n-k+3)}. \tag{3.4}$$

Proof: From the theory of Gateaux differential it is known, that

$$J(\mathbf{z})\mathbf{y} = \lim_{t\to 0} \frac{1}{t} \left(F(\mathbf{z} + t\mathbf{y}) - F(\mathbf{z})\right), \forall \mathbf{y} \in \mathbb{R}^{m+n-k+3}.$$

Let $\mathbf{y}$ be partitioned into three parts. In particular, set $\mathbf{y} = [\mathbf{y_0}, \mathbf{y_0}, \mathbf{y_1}]^T$ with $\mathbf{y_0} \in \mathbb{R}^{k+1}, \mathbf{y_1} \in \mathbb{R}^{m-k+1}$ and $\mathbf{y_2} \in \mathbb{R}^{n-k+1}$. Then, by using the definition of $F(\mathbf{z})$ we can write

$$J(\mathbf{z})\mathbf{y} = \lim_{t\to 0} \frac{1}{t} \left(\begin{bmatrix} \mathbf{r}^T(\mathbf{d} + t\mathbf{y_0}) - 1 \\ C_{k+1}(u + ty_1)(\mathbf{d} + t\mathbf{y_0}) \\ C_{k+1}(u + ty_2)(\mathbf{d} + t\mathbf{y_0}) \end{bmatrix} - \begin{bmatrix} \mathbf{r}^T\mathbf{d} - 1 \\ C_{k+1}(u)\mathbf{d} \\ C_{k+1}(u)\mathbf{d} \end{bmatrix} \right).$$

It follows from the linearity of the Toeplitz matrices and (2.2) that

$$J(\mathbf{z})\mathbf{y} = \lim_{t\to 0} \frac{1}{t} \left(\begin{bmatrix} t\mathbf{r}^T\mathbf{y_0} \\ tC_{k+1}(u)\mathbf{y_0} + tC_{k+1}(y_1)\mathbf{d} + t^2C_{k+1}(y_1)\mathbf{y_0} \\ tC_{k+1}(v)\mathbf{y_0} + tC_{k+1}(y_2)\mathbf{d} + t^2C_{k+1}(y_2)\mathbf{y_0} \end{bmatrix} \right) =$$

$$\begin{bmatrix} \mathbf{r}^T\mathbf{y_0} \\ C_{k+1}(u)\mathbf{y_0} + C_{k+1}(y_1)\mathbf{d} \\ C_{k+1}(v)\mathbf{y_0} + C_{k+1}(y_2)\mathbf{d} \end{bmatrix} = \begin{bmatrix} \mathbf{r}^T\mathbf{y_0} \\ C_{k+1}(u)\mathbf{y_0} + C_{m-k+1}(d)\mathbf{y_1} \\ C_{k+1}(v)\mathbf{y_0} + C_{n-k+1}(d)\mathbf{y_2} \end{bmatrix} =$$

$$\begin{bmatrix} \mathbf{r}^T & & \\ C_{k+1}(u) & C_{m-k+1}(d) & \\ C_{k+1}(v) & & C_{n-k+1}(d) \end{bmatrix} \begin{bmatrix} \mathbf{y_0} \\ \mathbf{y_1} \\ \mathbf{y_2} \end{bmatrix},$$

and the proof is completed. ■

Theorem 3.1.2. *Let $\hat{f}$ and $\hat{g}$ be polynomials with a common divisor d and let u and v be the cofactors, $\hat{f} = ud$ and $\hat{g} = vd$. Then for every vector $\mathbf{r}$ satisfying $\mathbf{r}^T\mathbf{d} \neq 0$, if d is the GCD of $\hat{f}$ and $\hat{g}$ (i.e. GCD$(u, v) = 1$), then the Jacobian of $F(\mathbf{z})$ in (3.3) is of full column rank.*

The theorem is proved in [30].

The Gauss-Newton method is used to find $\mathbf{z}$ as a solution of (3.1) in the least square sense. In particular, $\mathbf{z}$ is computed iteratively by

$$\mathbf{z}_{j+1} = \mathbf{z}_j + J(\mathbf{z}_j)^\dagger F(\mathbf{z_j}). \tag{3.5}$$

The method is convergent which is ensured by Theorem 3.1.2.

A first approximation of the coefficients is obtained by solving the system (2.4), i.e.

$$\begin{bmatrix} C_{k+1}(u) \\ C_{k+1}(v) \end{bmatrix} \mathbf{d} = \begin{bmatrix} \hat{\mathbf{f}} \\ \hat{\mathbf{g}} \end{bmatrix}.$$

Let us define

$$\mathbf{d}_0 = \mathbf{d}, \quad \mathbf{u}_0 = \mathbf{u}, \quad \mathbf{v}_0 = \mathbf{v},$$

then the vector $\mathbf{z}_0 = [\mathbf{d}_0,\ \mathbf{u}_0,\ \mathbf{v}_0]^T$ is taken as the initial iteration in (3.5), $\mathbf{u}_0$ and $\mathbf{v}_0$ are computed from the singular vector $\mathbf{v}_{\min}(S_k)$ of the first rank deficient matrix S_k.

The Moore-Penrose inverse in (3.5) is not computed explicitly. Instead of doing that, the system

$$J(\mathbf{z})\Delta_j\mathbf{z} = F(\mathbf{z}_j) - \mathbf{b}$$

is solved at each step of the iteration. Then $\mathbf{z}_{j+1} = \mathbf{z}_j - \Delta_j\mathbf{z}$.

Finally, the complete algorithm for the computation the AGCD can be presented. This algorithm computes for the given inexact polynomials their AGCD of the highest possible degree. From the previous discussions it follows that the degree of the AGCD may be different from the degree of the exact GCD of theoretically exact polynomials. The distance $\|F(\mathbf{z}_j) - \mathbf{b}\|/\|\mathbf{b}\|$ is measured as a termination criterion in the iteration process.

Algorithm 3.1.1 (GNAGCD)**.**

Input: Polynomials $\hat{f}, \hat{g}$ of degrees m and $n, m \geq n$, threshold $\theta > 0$, $tol > 0$.
Output: $AGCD$ of the maximal degree k.

```
(1)  begin
(2)    for k = n, n − 1, ..., 1 do
(3)      compute σ_min(S_k) and v_min(S_k) by the RankRev Algorithm 2.4.1;
(4)      if σ_min(S_k) < θ then
(5)        extract u_0 and v_0 from v_min(S_k);
(6)        compute d_0 as a least square solution of (2.4);
(7)        set r = d_0;
(8)        solve (3.1) by the iteration (3.5) for z;
(9)        if ‖F(z) − b‖ < tol then
(10)         extract d from z and set AGCD = d;
(11)         exit
(12)       end
(13)     end
(14)   end
(15)   set AGCD = const.
(16) end
```

The threshold θ applied to the singular values has to be chosen carefully. It was shown in the previous section that the rank estimation works well with $\theta = \varepsilon_{\text{mach}}\|S\|$ in case of the exact preprocessed polynomials. Other possible choices of θ are discussed in [30]. However, due to the sensitivity of singular values on noise, there is no strict restriction on thresholds.

3.2 Examples of Gauss-Newton method

An application of Algorithm 3.1.1 is demonstrated on the following examples.

Example 3.2.1. Suppose that the polynomials (2.11) from Example 2.5.3,

$$\begin{aligned} \hat{f}(x) &= (x+20.6)^2(x-4.7)^5(x-1.3)^4, \\ \hat{g}(x) &= (x+10.4)^3(x-4.7)^4(x-1.3)^3, \end{aligned} \tag{3.6}$$

are given. These polynomials have the GCD of degree 7. Denote the exact GCD by $\hat{d}$ and its coefficient vector by $\hat{\mathbf{d}}$.

The RankRev algorithm in Example 2.5.3 returns the numerical rank of $S(\hat{f}, \hat{g})$ that is equal to the rank of $S(\hat{f}, \hat{g})$ in case when the polynomials are not perturbed by any additional noise. Modification of the RankRev algorithm computes an approximation of the smallest singular value of the first rank deficient Sylvester subresultant matrix $S_7(\hat{f}, \hat{g})$ and an approximation of the associated singular vector $\mathbf{v}_{\min}(S_7)$. The cofactors of the AGCD given by (2.5) are then obtained from this approximation. Afterwards, (2.4) is solved yielding a first approximation d_0 of the AGCD d.

Algorithm 3.1.1, and so the refinement (3.5), is applied and returns the AGCD d of the polynomials $\hat{f}$ and $\hat{g}$. The coefficients $\hat{\mathbf{d}}$, $\mathbf{d}_0$ and $\mathbf{d}$ of the exact GCD, the first approximation of the AGCD and the AGCD, respectively, are shown in Table 3.1. Note that the AGCD is computed with the error $\|\hat{\mathbf{d}} - \mathbf{d}\| = 4.34 \times 10^{-9}$ that is much

	$\hat{d}$	d_0	d
x^7	1	1	1
x^6	−22.7000000000000	−22.6999993650910	−22.7000000000102
x^5	210.930000000000	210.929992657091	210.930000000124
x^4	−1029.71100000000	−1029.71096131310	−1029.71100000066
x^3	2820.88830000000	2820.88819009597	2820.88830000190
x^2	−4299.79641000000	−4299.79623911536	−4299.79641000295
x^1	3386.39479100000	3386.39465480595	3386.39479100235
x^0	−1072.06591570000	−1072.06587225264	−1072.06591570075

Table 3.1: The coefficients of the exact GCD $\hat{d}$ of the polynomials in (3.6) that are normalised by the geometric mean; the coefficients of the first approximation of the AGCD d_0 computed from (2.4); the coefficients of the AGCD d.

better than the distance of the first approximation from the exact GCD, $\|\hat{\mathbf{d}} - \mathbf{d}_0\| = 2.52 \times 10^{-4}$.

The computation of the first approximation is much more better when $\hat{f}$ and $\hat{g}$ are normalised and scaled by the operations from Section 1.3. In this case the distance of d_0 from $\hat{d}$ is $\|\hat{\mathbf{d}} - \mathbf{d}_0\| = 3.20 \times 10^{-8}$. The refinement (3.5) then gives the AGCD d with $\|\hat{\mathbf{d}} - \mathbf{d}\| = 7.85 \times 10^{-12}$. ♣

Example 3.2.2. Let us apply the componentwise perturbation with the signal-to-noise ratio $\epsilon = 10^{-8}$ to the polynomials $\hat{f}$ and $\hat{g}$ in (3.6).

Algorithm 3.1.1 applied to the perturbed polynomials f and g yields the AGCD of degree 2. This is unsatisfactory result, however, it perfectly follows the conclusions from Example 2.5.3 and Figure 2.6 where the RankRev algorithm returns $S_2(f, g)$ as the first rank deficient matrix (i.e. only the two singular values of $S(f, g)$ are below the threshold θ).

In the rank estimation the threshold $\theta = \varepsilon_{\text{mach}}\|S(f, g)\|$ is considered. This choice is preferred since we assume that the signal-to-noise ratio ϵ is not known. However, if we admit that ϵ is known, we can use the threshold $\theta = \epsilon\|S(f, g)\|$ in the numerical rank computation. The RankRev algorithm gives the correct numerical rank of $S(f, g)$ equal to 14 within the new threshold θ. Then, the accuracy of the AGCD d of the inexact polynomials f and g is $\|\hat{\mathbf{d}} - \mathbf{d}\| = 9.80 \times 10^{-5}$, where $\hat{d}$ is the exact GCD of $\hat{f}$ and $\hat{g}$. This is acceptable, even if $\|\hat{\mathbf{d}} - \mathbf{d}_0\|$ is 0.1737.

Inexact polynomials f and g have to be normalised and scaled, otherwise we get very poor AGCD with $\|\hat{\mathbf{d}} - \mathbf{d}_0\| = 9.80 \times 10^4$ and $\|\hat{\mathbf{d}} - \mathbf{d}\| = 1.8257$. ♣

Example 3.2.3. Consider the polynomials $\hat{f}$ and $\hat{g}$ in (1.20) of degrees $2m$ for $m = 6, 10, 16$ and 22, i.e.

$$\begin{aligned}\hat{f}(x) &= \prod_{i=1}^{\frac{m}{2}} \left[(x - r_1\alpha_i)^2 + r_1^2\beta_i^2\right] \prod_{i=\frac{m}{2}+1}^{m} \left[(x - r_2\alpha_i)^2 + r_2^2\beta_i^2\right], \\ \hat{g}(x) &= \prod_{i=1}^{m} \left[(x - r_1\alpha_i)^2 + r_1^2\beta_i^2\right],\end{aligned} \tag{3.7}$$

where $\alpha_i = \cos\left(\frac{\pi i}{16}\right), \beta_i = \sin\left(\frac{\pi i}{16}\right), i = 1, \ldots, n, r_1 = 0.5$ and $r_2 = 1.5$.. The exact GCD of the polynomials, denoted by $\hat{d}$, has degree m.

The RankRev algorithm 2.4.1 applied to the polynomials in Example 2.5.1 gives the numerical ranks of the Sylvester matrices that are equal to their ranks. The threshold $\theta = \varepsilon_{\text{mach}}\|S(\hat{f}, \hat{g})\|$ is always chosen. Also modification of the RankRev algorithm returns the expected rank deficient matrices S_m for each considered m, as can be seen from Figure 2.2. Along with revealing S_m, approximations of the smallest singular value of S_m and the corresponding singular vector are computed.

A first approximation d_0 of the AGCD is obtained by solving (2.4). Algorithm 3.1.1 then gives the AGCD d that is a refinement of d_0. The following tables compare distances of the computed AGCD d and its first approximation d_0, respectively, from the exact GCD $\hat{d}$. In particular, Table 3.2 compares the distances for the polynomials $\hat{f}$ and $\hat{g}$ in (3.7) that are not normalised and scaled. Table 3.3 then shows the distances in case when the polynomials in (3.7) are preprocessed (i.e. normalised by the geometric mean and scaled).

m	$\|\hat{\mathbf{d}} - \mathbf{d}_0\|$	$\|\hat{\mathbf{d}} - \mathbf{d}\|$	$\|F(\mathbf{z}) - \mathbf{b}\|$
6	4.56×10^{-15}	4.36×10^{-16}	9.83×10^{-16}
10	3.69×10^{-13}	5.68×10^{-14}	2.18×10^{-14}
16	2.09×10^{-9}	3.79×10^{-11}	1.06×10^{-12}
22	2.40×10^{-5}	3.42×10^{-8}	1.38×10^{-10}

Table 3.2: Distance of the exact GCD $\hat{d}$ from the first approximation d_0 of the AGCD, $\|\hat{\mathbf{d}} - \mathbf{d}_0\|$; the distance of $\hat{d}$ from the AGCD d, $\|\hat{\mathbf{d}} - \mathbf{d}\|$, for the polynomials in (3.7) for $m = 6, 10, 16$ and 22. Errors of the computation are measured by $\|F(\mathbf{z}) - \mathbf{b}\|$.

We can see that the refined AGCD is better that the first approximation computed from (2.4), mainly for the polynomials of higher degrees (up to 3 orders in magnitude). Unlike first Example 3.2.1, preprocessing operations do not bring any improvements in the computations.

Note that we do not add any noise to the coefficients of $\hat{f}$ and $\hat{g}$. If we add some noise of an unknown signal-to-noise ratio, the rank deficient matrix $S_k(f, g)$ returned by the modified RankRev algorithm has rank loss much smaller than required m, and so the computed AGCD has degree k that is smaller than m. If the signal-to-noise

m	$\|\hat{\mathbf{d}} - \mathbf{d}_0\|$	$\|\hat{\mathbf{d}} - \mathbf{d}\|$	$\|F(\mathbf{z}) - \mathbf{b}\|$
6	5.92×10^{-14}	6.27×10^{-16}	2.94×10^{-15}
10	1.21×10^{-12}	2.13×10^{-14}	1.03×10^{-13}
16	2.91×10^{-9}	9.40×10^{-11}	2.67×10^{-11}
22	1.22×10^{-5}	1.39×10^{-7}	1.06×10^{-8}

Table 3.3: Distance of the exact GCD $\hat{d}$ from the first approximation d_0 of the AGCD, $\|\hat{\mathbf{d}} - \mathbf{d}_0\|$; the distance of $\hat{d}$ from the AGCD d, $\|\hat{\mathbf{d}} - \mathbf{d}\|$, for the polynomials in (3.7) for $m = 6, 10, 16$ and 22. Polynomials are normalised by the geometric mean and scaled by the substitution from Section 1.3. Errors of the computation are measured by $\|F(\mathbf{z}) - \mathbf{b}\|$.

ratio ϵ is known, the threshold $\theta = \epsilon\|S(f, g)\|$ may be sufficient in the rank estimation for some m.

We can conclude that received results meet expectations in sense that for complicated polynomials, as those in (3.7), we can compute an acceptable AGCD. Even, our results are slightly better than results in [30] where the algorithm is tested on the same polynomials. ♣

Summary

In this chapter an AGCD solver based on the numerical rank estimation has been presented. This solver refines an approximation of the AGCD computed from (2.4) by the Gauss-Newton method (3.5). We have seen from examples that the algorithm brings improvements in accuracy of the AGCD (up to several orders in magnitude).

Preprocessing operations make results qualitatively better in Example 3.2.1 and Example 3.2.2. However, they do not play important role in the AGCD refinement in Example 3.2.3 (but their application do not make results worse than their omitting). Hence, we suggest to provide preprocessing operation in the AGCD computation in any case.

The correct numerical rank estimation is essential in the AGCD computation, and we know from Chapter 2 that it is not a trivial problem. We have seen that for exact polynomials the Gauss-Newton can be used as a tool for the AGCD computation. However, inexact polynomials perturbed by an unknown noise are subject of our interest and results returned by the Gauss-Newton for inexact polynomials do not satisfy us, since the returned AGCD of perturbed polynomials has degree (much) smaller than the degree of the exact GCD of exact polynomials. Therefore a different approach has to be studied.

Chapter 4

AGCD and Structured TLS

Following the paper [30], a method for the AGCD computation is presented in Chapter 3. This method has promising results when the exact polynomials are considered. However, it is said at the end of the chapter that the method does not work properly when the coefficients of polynomials are perturbed by some noise and the amplitude of noise is not known. The aim of further work is therefore to develop a robust AGCD solver that can eliminate difficulties caused by missing information about the level of involved noise.

We offer two methods that belong to the matrix structure preserving methods. So, their big advantage is that all data needed for the AGCD can be readily obtained from the particular Sylvester matrices.

Methods solve a structured Total Least Squares (STLS) problem that naturally appears in the AGCD computation. Relations between AGCD and Total Least Squares (TLS) are shown in the next Section 4.1. Discussion on TLS is provided in Section 4.2. Two methods for solving STLS are then considered in Sections 4.3 and 4.4. In particular, these methods are the STLN method originally made in [21] and a method that applies the Newton method to a suitable Lagrangian function in [16]. In [16] is the second method called STLS2.

Note that both methods STLN and SLTS2 work on Toeplitz matrices defined in (1.5). The structure of Sylvester matrices as the block Toeplitz matrices has to be taken into account. This block Toeplitz structure makes only slight changes in the algorithms.

4.1 TLS in AGCD computation

In this section the basic theory that leads to the TLS problem is explained. In particular, suppose that the theoretically exact forms of the inexact polynomials f and g are known. Denote them by $\hat{f}$ and $\hat{g}$,

$$\hat{f}(x) = \sum_{i=0}^{m} \hat{a}_i x^{m-i} \quad \text{and} \quad \hat{g}(x) = \sum_{j=0}^{n} \hat{b}_j x^{n-j}, \tag{4.1}$$

where $\hat{a}_0\hat{a}_m \neq 0$ and $\hat{b}_0\hat{b}_n \neq 0$, and suppose that these polynomials have a non-constant GCD of degree $\hat{d}$, i.e.

$$\deg \mathrm{GCD}(\hat{f}, \hat{g}) = \hat{d}.$$

Then for $k = 1, \ldots, \hat{d}$ there exist a common divisor d_k of degree k, and the quotient polynomials

$$u_k(x) = \sum_{i=0}^{m-k} u_{k,i} x^{m-k-i} \quad \text{and} \quad v_k(x) = \sum_{i=0}^{n-k} v_{k,i} x^{n-k-i} \tag{4.2}$$

with $\deg u_k = m - k < \deg \hat{f}$ and $\deg v_k = n - k < \deg \hat{g}$ such that

$$\hat{f} = u_k d_k \quad \text{and} \quad \hat{g} = v_k d_k.$$

It follows that

$$d_k = \frac{\hat{f}}{u_k} = \frac{\hat{g}}{v_k}, \tag{4.3}$$

$$\hat{f} v_k = \hat{g} u_k \tag{4.4}$$

and so

$$C_{n-k+1}(\hat{f})\mathbf{v_k} = C_{m-k+1}(\hat{g})\mathbf{u_k} \tag{4.5}$$

by (1.6). Equation (4.5) is equivalent to

$$\begin{bmatrix} C_{n-k+1}(\hat{f}), & C_{m-k+1}(\hat{g}) \end{bmatrix} \begin{bmatrix} \mathbf{v_k} \\ -\mathbf{u_k} \end{bmatrix} = \mathbf{0}, \tag{4.6}$$

respectively

$$S_k(\hat{f}, \hat{g}) \begin{bmatrix} \mathbf{v_k} \\ -\mathbf{u_k} \end{bmatrix} = \mathbf{0}, \tag{4.7}$$

where $S_k(\hat{f}, \hat{g}) = \Big[C_{n-k+1}(\hat{f}),\, C_{m-k+1}(\hat{g}) \Big] \in \mathbb{R}^{(m+n-k+1)\times(m+n-2k+2)}$ is the kth Sylvester subresultant matrix. Matrices $C_{n-k+1}(\hat{f})$ and $C_{m-k+1}(\hat{g})$ are the Toeplitz matrices.

The matrix $S_k = S_k(\hat{f}, \hat{g})$ does not have full column rank since the exact polynomials have a common divisor for every $k = 1, \ldots, \hat{d}$, and so the homogeneous system (4.7) has the exact solution for every $k = 1, \ldots, \hat{d}$. On the other side, if $k > \hat{d}$, then there is no common divisor of $\hat{f}$ and $\hat{g}$, and so S_k is of full column rank. The previous observations can be summarised as follows:

$$\begin{aligned} &\mathrm{rank}(S_k(\hat{f}, \hat{g})) < m + n - 2k + 2, && k = 1, \ldots, \hat{d}, \\ &\mathrm{rank}(S_k(\hat{f}, \hat{g})) = m + n - 2k + 2, && k = \hat{d} + 1, \ldots, n. \end{aligned}$$

If $\hat{f}$ and $\hat{g}$ have a common divisor for all values of $k = 1, \ldots, \hat{d}$, the homogeneous system (4.7) can be transformed to the linear system of algebraic equations

$$A_k \mathbf{x} = \mathbf{c_k}, \qquad k = 1, \ldots, \hat{d} \tag{4.8}$$

by setting $v_{k,0} = -1$, where $\mathbf{c_k} \in \mathbb{R}^{m+n-k+1}$ is the first column of S_k and $A_k \in \mathbb{R}^{(m+n-k+1)\times(m+n-2k+1)}$ is formed from the remaining $m + n - 2k + 1$ columns of S_k.

The vector $\mathbf{x} \in \mathbb{R}^{m+n-2k+1}$ contains the coefficients of v_k and u_k omitting the first coefficient $v_{k,0}$.

However, Theorem 2.3.1 implies that $S(f,g)$ is of full rank for any pair of inexact polynomials f and g. Therefore $S_k(f,g)$ is of full column rank, i.e. for every value $k = 1, \ldots, n$

$$\mathbf{c_k} \notin \text{Range}(A_k) \tag{4.9}$$

and so (4.8) does not possess any solution for every k.

On the other side, the definition of the AGCD needs the smallest perturbations of the coefficients of inexact polynomials to be computed. In particular, if f and g,

$$f(x) = \sum_{i=0}^{m} a_i x^{m-i} \quad \text{and} \quad g(x) = \sum_{j=0}^{n} b_j x^{n-j} \tag{4.10}$$

of degrees m and n, respectively, $m \geq n$, are two inexact polynomials, then it is necessary to compute *minimal* perturbations $\{\delta a_i\}_{i=0}^{m}$ and $\{\delta b_j\}_{j=0}^{n}$ of the coefficients $\{a_i\}_{i=0}^{m}$ and $\{b_j\}_{j=0}^{n}$ so that the polynomials defined by

$$\tilde{f}(x) = \sum_{i=0}^{m} (a_i + \delta a_i) x^{m-i} \quad \text{and} \quad \tilde{g}(x) = \sum_{j=1}^{n} (b_j + \delta b_j) x^{n-j} \tag{4.11}$$

have a non-constant GCD with the *largest* possible degree (so that the *nearness*, *min-distance* and *max-degree* conditions are satisfied).

If $S_k = S_k(f,g)$ is the kth Sylvester matrix of f and g, and δS_k is the kth Sylvester matrix of $\delta f = \sum_{i=0}^{m} \delta a_i x^{m-i}$ and $\delta g = \sum_{j=1}^{n} \delta b_j x^{n-j}$, then $S_k + \delta S_k$ is rank deficient. In other words $S_k + \delta S_k$ is a low rank approximation of S_k. Since we want the AGCD of maximum degree, the low rank approximation should be of the lowest possible rank.

Remark 4.1.1. Note that if δS_k has the same structure as S_k, then the required polynomials $\tilde{f}$ and $\tilde{g}$ can be directly retrieved from $S_k + \delta S_k$. Moreover, we can very easily control the distances $\|\mathbf{f} - \tilde{\mathbf{f}}\|$ and $\|\mathbf{g} - \tilde{\mathbf{g}}\|$. Hence, the Sylvester structure of S_k is crucial.

Remark 4.1.2. The AGCD of the inexact polynomials is then the exact GCD of $\tilde{f}$ and $\tilde{g}$, i.e. $\text{AGCD}(f,g) = \text{GCD}(\tilde{f}, \tilde{g})$.

However, if $S_k + \delta S_k$ is rank deficient, then there must exist a solution to the homogeneous system

$$(S_k(f,g) + \delta S_k(\delta f, \delta g)) \begin{bmatrix} \tilde{\mathbf{v}}_\mathbf{k} \\ -\tilde{\mathbf{u}}_\mathbf{k} \end{bmatrix} = \mathbf{0}. \tag{4.12}$$

Similarly, (4.12) can be transformed to the system

$$(A_k + E_k)\mathbf{x} = \mathbf{c_k} + \mathbf{h_k}, \tag{4.13}$$

where $\delta S_k = [\mathbf{h_k}, E_k]$, i.e. $\mathbf{h_k}$ is the first column of δS_k and E_k is formed from the remaining columns of δS_k.

Problem (4.13) is called the Total Least Squares (TLS) problem, since we want to find the minimal perturbations $\mathbf{z} = [\delta\mathbf{f}^T, \delta\mathbf{g}^T]^T$, and E_k and $\mathbf{h_k}$ are the functions of $\mathbf{z}$, [10, 11, 16, 23]. Note that solving TLS problem (4.13) can be thought as the computation of a low rank approximation $S_k + \delta S_k$ of the full column rank matrix S_k.

In the next section a brief discussion on solving TLS problems is stated.

4.2 Structured TLS

In many problems we need to solve the overdetermined system of equations

$$A\mathbf{x} \approx \mathbf{c} \tag{4.14}$$

where $A \in \mathbb{R}^{m\times n}$ is a data matrix and $\mathbf{c} \in \mathbb{R}^m$ is a vector of observations, $m > n$. See the book of Van Huffel, [23].

Classical Least Square (LS) approach assumes errors only in the vector $\mathbf{c}$ and A is free of error. However, in practice this is not often the case. For example, instrument errors, human errors and sampling errors usually affect the measurements in A. Hence, perturbations of A and $\mathbf{c}$ have to be imposed. This leads to the following TLS problem:

$$\begin{aligned} &\min_{E,\mathbf{h},\mathbf{x}} \|[\mathbf{h}, E]\|_F \\ &\text{subject to } (A+E)\mathbf{x} = \mathbf{c} + \mathbf{h}. \end{aligned} \tag{4.15}$$

In [16, 23], a solution $\mathbf{x}$ is called as the parameter vector. The parameter vector $\mathbf{x}$ is required to be a maximum likelihood (ML) estimate, [16]. This can be satisfied when $[\mathbf{h}, E]$ has the same structure as $[\mathbf{c}, A]$.

The second important argument for $[\mathbf{h}, E]$ having the same structure as $[\mathbf{c}, A]$ is pointed in Remark 4.1.1.

The TLS problem (4.15) can be reformulated to the Structured TLS (STLS) in the following way:

$$\begin{aligned} &\min_{E,\mathbf{h},\mathbf{x}} \|[\mathbf{h}, E]\|_F \\ &\text{subject to } (A+E)\mathbf{x} = \mathbf{c} + \mathbf{h} \\ &\text{and } [\mathbf{h}, E] \text{ is of the same structure as } [\mathbf{c}, A]. \end{aligned} \tag{4.16}$$

Hence, STLS is an extension of TLS where constraints on the matrix structure are involved.

Another advantage of considered (block) Toeplitz structure of $[\mathbf{h}, E]$ is that it contains only several different values. In particular, the Sylvester matrix $\delta S = [\mathbf{h}, E]$ contains the perturbations of the coefficients of polynomials f and g of degrees m and n, respectively. Hence, if we denote $\mathbf{z} = [\delta\mathbf{f}^T, \delta\mathbf{g}^T]^T$ as those perturbations, then $[\mathbf{h}, E]$ has at most $m + n + 2$ different entries. So, in (4.16) we are minimising $\|\mathbf{z}\|_2$ instead of $\|[\mathbf{h}, E]\|_F$.

Finally, the STLS that is solved is

$$\begin{aligned} &\min_{\mathbf{z},\mathbf{x}} \|\mathbf{z}\| \\ &\text{subject to } (A+E)\mathbf{x} = \mathbf{c}+\mathbf{h} \\ &\text{and } [\mathbf{h}, E] \text{ is of the same structure as } [\mathbf{c}, A], \end{aligned} \tag{4.17}$$

where $\mathbf{h} = \mathbf{h}(\mathbf{z})$ and $E = E(\mathbf{z})$.

Note that there are several methods for solving TLS problems, for example a method in [11] on p. 598 that uses the SVD of $D[A, \mathbf{c}]T$ where D and T are non-singular diagonal matrices. Disadvantage of the method is that it destroys the suitable structure of involved Sylvester matrices.

We propose two methods for solving STLS problem (4.17). The first method STLN is pretty straightforward, since it solves SLTS as a non-linearly constrained optimisation problem. The method is developed in [21] and uses a linearisation of the constraint function around a current solution point, see [21, 16]. In particular, let $r(\mathbf{z}, \mathbf{x}) = (A+E)\mathbf{x} - \mathbf{c} - \mathbf{h}$ be the residual related to the constraint equation in (4.17). If $\mathbf{\Delta z}$ and $\mathbf{\Delta x}$ are increments of $\mathbf{z}$ and $\mathbf{x}$, respectively, then linearisation is meant to be the expansion of $r(\mathbf{z}+\mathbf{\Delta z}, \mathbf{x}+\mathbf{\Delta x})$ around the point $[\mathbf{z}^T, \mathbf{x}^T]^T$. Section 4.3 describes the STLN method in more details.

The second method solves (4.17) in a different way. Problem (4.17) is reformulated into unconstrained optimisation problem in which the Newton method is applied to the Lagrangian function L,

$$L(\mathbf{z}, \mathbf{x}, \lambda) = 1/2\|\mathbf{z}\|^2 - \lambda^T \left(\mathbf{c} + \mathbf{h} - (A+E)\mathbf{x}\right),$$

λ is the vector of Lagrange multipliers. More on the Newton method and this approach can be found in Section 4.4. Note that this method is called STLS2 in [16], we will use an abbreviation LNSTLS for Lagrange-Newton STLS.

4.3 Structured total least norm

Previous section ends with some comments on the two possible methods that can be used for solving STLS problem (4.17). The first method is called the Structured Total Least Norm (STLN) method and can be found in [21].

Recall that in (4.17), the vector $\mathbf{z}$ represents perturbations of the coefficients of inexact polynomials, $S_k = [\mathbf{c}, A]$ is the kth Sylvester matrix for the fixed index k. Residual of the constraint function in (4.17) is defined by

$$r(\mathbf{z}, \mathbf{x}) = (A+E)\mathbf{x} - \mathbf{c} - \mathbf{h} \tag{4.18}$$

and the increments of $\mathbf{z}$ and $\mathbf{x}$ are denoted by $\Delta\mathbf{z}$ and $\Delta\mathbf{x}$, respectively, $\delta S_k = [\mathbf{h}, E]$.

Notice also, that in (4.18) we can write

$$E\mathbf{x} - \mathbf{h} = [\mathbf{h}, E] \begin{bmatrix} -1 \\ \mathbf{x} \end{bmatrix} = Y\mathbf{z}, \tag{4.19}$$

where $Y = Y(\mathbf{x})$ is a function of $\mathbf{x}$, and $E = E(\mathbf{z})$, $\mathbf{h} = \mathbf{h}(\mathbf{z})$. This swapping is the most important in the method, since it enables us to make substitutions between unknowns $\mathbf{x}$ and $\mathbf{z}$. A special procedure of constructing Y can be found in [21]. We do not show it here, since it is described in more details in Section 5.3 where the method of SNTLN, which is the extension of STLN customised for the AGCD computation, is given.

The Taylor expansion of $r(\mathbf{z} + \mathbf{\Delta z}, \mathbf{x} + \mathbf{\Delta x})$ can be written as

$$r(\mathbf{z} + \mathbf{\Delta z}, \mathbf{x} + \mathbf{\Delta x}) = r(\mathbf{z}, \mathbf{x}) + J \begin{bmatrix} \Delta \mathbf{z} \\ \Delta \mathbf{x} \end{bmatrix} + h.o.t.$$

where $h.o.t.$ stands for higher order terms of expansion and J is the Jacobian matrix of the residual $r(\mathbf{z}, \mathbf{x})$. In particular,

$$J = [\partial_z r(\mathbf{z}, \mathbf{x}), \partial_x r(\mathbf{z}, \mathbf{x})] = [Y, A + E]. \tag{4.20}$$

By neglecting $h.o.t.$ and setting $r(\mathbf{z} + \mathbf{\Delta z}, \mathbf{x} + \mathbf{\Delta x}) = \mathbf{0}$, we get the system

$$J \begin{bmatrix} \Delta \mathbf{z} \\ \Delta \mathbf{x} \end{bmatrix} = -r(\mathbf{z}, \mathbf{x}) \tag{4.21}$$

that needs to be solved to compute the increments $\mathbf{\Delta z}$ and $\mathbf{\Delta x}$.

Note that if A is Toeplitz (Sylvester) matrix, then Y is Toeplitz (Sylvester) as well. The matrix A is for inexact polynomials non-singular. Due to ML estimate property of $\mathbf{x}$ matrix $[Y, A + E]$ is also non-singular.

The objective function in (4.17), that is minimised, is $\|\mathbf{z}\|$, where the first $m + 1$ entries of $\mathbf{z}$ are perturbations of the coefficients of f and the remaining $n+1$ entries are perturbations of the coefficients of g. Following [21], let us define a diagonal weighting matrix $D \in \mathbb{R}^{(m+n+2)\times(m+n+2)}$ that represents the repetition of elements of $\mathbf{z}$ in E. In particular,

$$D = \begin{bmatrix} D_1 & 0 \\ 0 & D_2 \end{bmatrix}, \quad D_1 = (n - k + 1)I_{m+1} \quad \text{and} \quad D_2 = (m - k + 1)I_{n+1}. \tag{4.22}$$

Hence, computing $\mathbf{\Delta z}$ as a correction of $\mathbf{z}$ can be done through the minimising

$$\|D(\mathbf{z} + \mathbf{\Delta z})\| = \|[D, 0] \begin{bmatrix} \mathbf{\Delta z} \\ \mathbf{\Delta x} \end{bmatrix} - (-D\mathbf{z})\|.$$

Thus, it is necessary to solve

$$\begin{aligned} &\min_{\mathbf{\Delta z}, \mathbf{\Delta x}} \left\| [D, 0] \begin{bmatrix} \mathbf{\Delta z} \\ \mathbf{\Delta x} \end{bmatrix} - (-D\mathbf{z}) \right\| \\ &\text{subject to } J \begin{bmatrix} \mathbf{\Delta z} \\ \mathbf{\Delta x} \end{bmatrix} = -r(\mathbf{z}, \mathbf{x}). \end{aligned} \tag{4.23}$$

that is the equality constrained (LSE) problem, see for example [11], p. 585.

After solving (4.23) a new solution point is defined by

$$\mathbf{z} = \mathbf{z} + \mathbf{\Delta z},$$
$$\mathbf{x} = \mathbf{x} + \mathbf{\Delta x}.$$

Since we want to find the minimal perturbation $\mathbf{z}$, the initial value of $\mathbf{z}$ is $\mathbf{z} = \mathbf{0}$. Hence, $[\mathbf{h}, E]$ is the zero matrix. Initialisation of $\mathbf{x}$ is a standard LS solution of $A\mathbf{x} = \mathbf{c}$. The stopping criterion depends on the application. In [21] the method terminates whenever the correction do not change much, i.e. when $\|\mathbf{\Delta z}\| < \epsilon$ and $\|\mathbf{\Delta x}\| < \epsilon$ for some tolerance ϵ. We will however use condition $\|r(\mathbf{z}, \mathbf{x})\|/(\|\mathbf{c} + \mathbf{h}\|) < \epsilon$.

The described method is summarised in the following STLN algorithm.

Algorithm 4.3.1 (STLN)**.**

Input: Sylvester matrix $S = [\mathbf{c}, A]$, tolerance $tol > 0$,
Output: vector $\mathbf{z}$, parameter vector $\mathbf{x}$ and Sylvester matrix $\delta S(\mathbf{z}) = [\mathbf{h}, E]$ of the same structure as S satisfying $(A + E)\mathbf{x} = \mathbf{c} + \mathbf{h}$ with the minimal $\|\mathbf{z}\|$.

```
(1)  begin
(2)    initialization
(3)        compute x as a solution to Ax ≈ c;
(4)        set z = 0, [h, E] = 0;
(5)        form D in (4.22), r(z, x) in (4.18) and J in (4.20);
(6)        form Y from x;
(8)    while ||r(z,x)|| / ||c+h|| ≥ tol do
(9)          solve the LSE problem (4.23);
(10)         set z := z + Δz and x := x + Δx;
(11)         update Y, E, h, J, r(z, x), from z and x;
(12)   end
(13) end
```

Note that the STLN algorithm is used for the AGCD computation in [13, 24]. Winkler in [24] applies STLN to the matrix $S(f, \alpha g)$ for a scaled parameter α that needs to be specified. Furthermore, in [25] a variable substitution $x = \gamma w$ is used that together with α leads to the reduction of condition numbers of involved matrices as it was shown in Section 1.3. However, additional parameters α and γ make STLN a non-linear method. Some improvements of the method can be further achieved by a "column pivoting". That is all discussed in Chapter 5.

4.4 Lagrange-Newton STLS

The second method that solves STLS problem (4.17) can be found in [16, 23]. It is said in [21, 16] that non-linear optimisation in STLN may attain an arbitrary local minimum, especially when initial norm of residual $\|r(\mathbf{z}, \mathbf{x})\|$ is small.

Recall that we want to find a solution $[\mathbf{z}^T, \mathbf{x}^T]^T$ to the problem

$$\begin{aligned} &\min_{\mathbf{z},\mathbf{x}} f(\mathbf{z}, \mathbf{x}) \\ &\text{subject to } r(\mathbf{z}, \mathbf{x}) = \mathbf{0} \end{aligned} \tag{4.24}$$

where $f(\mathbf{z}, \mathbf{x}) = 1/2\|\mathbf{z}\|^2$ is the objective function and $r(\mathbf{z}, \mathbf{x}) = (A + E)\mathbf{x} - \mathbf{c} - \mathbf{h}$ is the residual of constrained conditions defined in (4.18).

Following book [8] on p. 138, an efficient way of solving this problem is to use linear approximations to the constraints $r(\mathbf{z}, \mathbf{x})$. This approach finds the stationary point $(\mathbf{z}^*, \mathbf{x}^*, \lambda^*)$ of the Lagrangian function $L(\mathbf{z}, \mathbf{x}, \lambda)$,

$$\begin{aligned} L(\mathbf{z}, \mathbf{x}, \lambda) &= f(\mathbf{z}, \mathbf{x}) + \lambda^T r(\mathbf{z}, \mathbf{x}) \\ &= 1/2\|\mathbf{z}\|^2 - \lambda^T (\mathbf{c} + \mathbf{h} - (A + E)\mathbf{x}) \\ &= 1/2\|\mathbf{z}\|^2 - \lambda^T (\mathbf{c} - A\mathbf{x} - Y\mathbf{z},) \end{aligned} \tag{4.25}$$

where λ are the Lagrange multipliers. Variable swapping on the last two rows is the same as in (4.19).

A stationary point of $L(\mathbf{z}, \mathbf{x}, \lambda)$ is a point $(\mathbf{z}^*, \mathbf{x}^*, \lambda^*)$ satisfying

$$\nabla L(\mathbf{z}^*, \mathbf{x}^*, \lambda^*) = \mathbf{0}, \tag{4.26}$$

i.e. the gradient of $L(\mathbf{z}, \mathbf{x}, \lambda)$ becomes zero at $(\mathbf{z}^*, \mathbf{x}^*, \lambda^*)$. It can be shown, see [8] p. 49, that if (4.26) can be solved, then a solution $(\mathbf{z}^*, \mathbf{x}^*)$ is a stationary point of $f(\mathbf{z}, \mathbf{x})$ satisfying $r(\mathbf{z}^*, \mathbf{x}^*) = \mathbf{0}$. Newton method can be then used to solve (4.26). Method for solving (4.17) is therefore called the Lagrange-Newton STLS (LNSTLS) method.

In particular, assume that $(\mathbf{z}, \mathbf{x}, \lambda)$ is the current approximation of the stationary point, and denote increments of the current point by $\mathbf{\Delta z}$, $\mathbf{\Delta x}$ and $\mathbf{\Delta}\lambda$, respectively. Expansion of $\nabla L(\mathbf{z} + \mathbf{\Delta z}, \mathbf{x} + \mathbf{\Delta x}, \lambda + \mathbf{\Delta}\lambda)$ around the current point gives

$$\nabla L(\mathbf{z} + \mathbf{\Delta z}, \mathbf{x} + \mathbf{\Delta x}, \lambda + \mathbf{\Delta}\lambda) = \nabla L(\mathbf{z}, \mathbf{x}, \lambda) + H \begin{bmatrix} \mathbf{\Delta z} \\ \mathbf{\Delta x} \\ \mathbf{\Delta}\lambda \end{bmatrix} + h.o.t. \tag{4.27}$$

where, again, *h.o.t.* represents higher order terms in expansion and H is the Hessian matrix of $L(\mathbf{z}, \mathbf{x}, \lambda)$. It can be easily verified that

$$\nabla L(\mathbf{z}, \mathbf{x}, \lambda) = \begin{bmatrix} \partial_z L \\ \partial_x L \\ \partial_\lambda L \end{bmatrix}_{(\mathbf{z},\mathbf{x},\lambda)} = \begin{bmatrix} I\mathbf{z} + Y^T\lambda \\ (A + E)^T\lambda \\ r(\mathbf{z}, \mathbf{x}) \end{bmatrix} \tag{4.28}$$

and

$$\begin{aligned} H = \begin{bmatrix} \partial_{zz} L & \partial_{zx} L & \partial_{z\lambda} L \\ \partial_{xz} L & \partial_{xx} L & \partial_{x\lambda} L \\ \partial_{\lambda z} L & \partial_{\lambda x} L & \partial_{\lambda\lambda} L \end{bmatrix}_{(\mathbf{z},\mathbf{x},\lambda)} &= \begin{bmatrix} I & \partial_x(Y^T\lambda) & Y^T \\ \partial_z(E^T\lambda) & 0 & (A + E)^T \\ Y & A + E & 0 \end{bmatrix} \\ &\approx \begin{bmatrix} I & 0 & Y^T \\ 0 & 0 & (A + E)^T \\ Y & A + E & 0 \end{bmatrix}. \end{aligned} \tag{4.29}$$

Matrices $\partial_x(Y^T\lambda)$ and $\partial_z(E^T\lambda)$ are the derivatives of $Y^T\lambda = Y^T(\mathbf{x})\lambda$ and $E^T\lambda = E^T(\mathbf{z})\lambda$ with respect to $\mathbf{x}$ and $\mathbf{z}$, respectively. Inclusion of these terms however makes the structure of S complicated. In [16] it is recommended to replace these terms by zero matrices of the corresponding sizes.

Neglecting *h.o.t.* in (4.27) and setting the right hand side to zero gives the system

$$H \begin{bmatrix} \boldsymbol{\Delta}\mathbf{z} \\ \boldsymbol{\Delta}\mathbf{x} \\ \boldsymbol{\Delta}\lambda \end{bmatrix} = -\nabla L(\mathbf{z}, \mathbf{x}, \lambda) \tag{4.30}$$

that needs to be solved to compute increments $\boldsymbol{\Delta}\mathbf{z}$, $\boldsymbol{\Delta}\mathbf{x}$ and $\boldsymbol{\Delta}\lambda$. New approximation of the stationary point is obtained by setting

$$\begin{aligned} \mathbf{z} &= \mathbf{z} + \boldsymbol{\Delta}\mathbf{z}, \\ \mathbf{x} &= \mathbf{x} + \boldsymbol{\Delta}\mathbf{x}, \\ \lambda &= \lambda + \boldsymbol{\Delta}\lambda. \end{aligned}$$

The initial vector of perturbations is set to be the zero vector, i.e. $\mathbf{z} = \mathbf{0}$. Also, $\lambda = \mathbf{0}$ and $\mathbf{x}$ is a LS solution to $A\mathbf{x} = \mathbf{c}$. Stopping criteria are similar to those in Section 4.3.

In [16] it is also developed a fast LDL^T factorisation of H to solve (4.30) efficiently. However, we will use a straightforward methods built in Matlab for solving (4.30) because we would like to compare this method with the STLN method given in the previous section where the standard Matlab functions only are used in our programs.

Just presented method for solving the STLS problem (4.17) by the Newton method applied to the Lagrangian function (4.25) is summed up in the following Algorithm 4.4.1.

Algorithm 4.4.1 (LNSTLS)**.**

Input: Sylvester (Toeplitz) matrix $S = [\mathbf{c}, A]$, tolerance $tol > 0$,
Output: vector $\mathbf{z}$, parameter vector $\mathbf{x}$ and Sylvester (Toeplitz) matrix $\delta S(\mathbf{z}) = [\mathbf{h}, E]$ of the same structure as S satisfying $(A + E)\mathbf{x} = \mathbf{c} + \mathbf{h}$ with the minimal $\|\mathbf{z}\|$.

```
(1)  begin
(2)    initialization
(3)        compute x as a solution to Ax ≈ c;
(4)        set z = 0, [h, E] = 0;
(5)        set λ = 0;
(6)        form Y from x and H, ∇L given in (4.28) and (4.29);
(8)    while ||r(z,x)|| / (||c||+||h||) ≥ tol do
(9)          solve (4.30) for Δz, Δx, Δλ;
(10)         set z := z + Δz, x := x + Δx and λ := λ + Δλ;
(11)         update Y, E, h, H, ∇L, from z, x and λ;
(12)   end
(13) end
```

Now the idea is to customise the algorithm for the purposes of the AGCD computation similarly, similarly to the STLN algorithm that is used for these purposes in [13, 24, 25]. Note again, that in [25] the STLN method is embedded to SNTLN method described in Section 5.3.

To the author's best knowledge there is no such method for the AGCD computation that is based on Algorithm 4.4.1. A new method is described in more details in Section 5.4 and compared with the SNTLN method.

Summary

This chapter has introduced the connection between the AGCD and a need to solve the structured TLS problem (4.17). Firstly, it has been shown how the AGCD computation problem can be transformed to the TLS problem. A general discussion on STLS and two different methods for solving it have bee then presented.

The computation of the AGCD through the solving STLS problem can be very convenient mainly when considered polynomials are perturbed by a noise, even of an unknown level. If we assume that the polynomials are perturbed by a noise, then these polynomials are coprime. The computation of the AGCD by looking for the first rank deficient Sylvester subresultant matrix may not lead to the success, since the numerical rank does not have to be defined or we are not able to compute it reliably. However, if we find a solution to the STLS problem, then, in fact, we have two modified forms of the inexact polynomials that are no longer coprime. The GCD of these modified polynomials is then the AGCD of inexact polynomials with respect to the definition of the AGCD.

Chapter 5

Non-linear extensions of TLS methods for AGCD

Methods for the AGCD computation discussed in this chapter are extensions of the methods solving STLS problems introduced in the previous chapter.

In particular, it is STLN method which is used firstly for the AGCD computation in [13] and then extended by Winkler to the Structured Non-linear Total Least Norm (SNTLN) method in [24, 25, 27]. The extension of STLN to SNTLN is in introducing balancing parameters α and γ from Section 1.3. Differences from STLN can be easily understood and implemented. It was said earlier, e.g. in Figures 2.4b and 2.6, and it can be found in [25] that SNTLN gives better results in the rank estimation of Sylvester matrices in sense that the numerical-rank gap is better defined for more complicated examples of polynomials. The method can be computationally made faster by considering a "column pivoting" in solved systems. The method of SNTLN is introduced in Section 5.3.

The second method used for the AGCD computation is based on the LNSTLS method, Algorithm 4.4.1. Similarly to SNTLN, the method of LNSTLS is embedded to non-linear version referred to as the Lagrange-Newton Structured Non-linear Total Least Squares (LNSNTLS) method. This method is introduced in Section 5.4.

All methods are tested on examples in Section 5.5.

5.1 Column pivoting

It is shown in Section 4.1 that the structured TLS problem

$$\begin{aligned} &\min_{\mathbf{z},\mathbf{x}} \|\mathbf{z}\| \\ &\text{subject to } (A_k + E_k)\mathbf{x} = \mathbf{c_k} + \mathbf{h_k} \\ &\text{and } [\mathbf{h_k}, E_k] \text{ is of the same structure as } [\mathbf{c_k}, A_k] \end{aligned} \tag{5.1}$$

needs to be solved to compute a rank deficient Sylvester matrix $S_k + \delta S_k$ with $S_k = [\mathbf{c_k}, A_k], \delta S_k = [\mathbf{h_k}, E_k]$ having the same structures. Recall that $\mathbf{c_k}, \mathbf{h_k} \in \mathbb{R}^{m+n-k+1}$

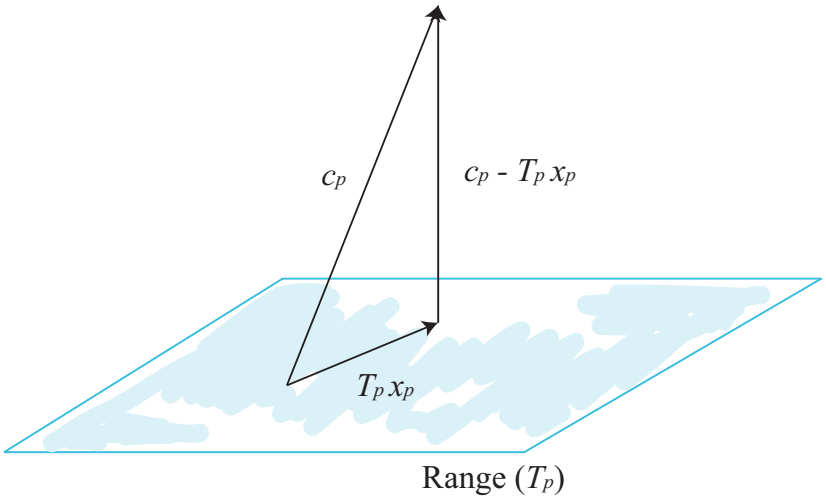


Figure 5.1: Linear independence of the columns of the Sylvester matrix of inexact polynomials.

are the first columns of S_k and δS_k, respectively, and $A_k, E_k \in \mathbb{R}^{(m+n-k+1)\times(m+n-2k+1)}$ are formed from the remaining $m+n-2k+1$ columns of S_k and δS_k, respectively. Sylvester subresultants $S_k = S_k(f,g)$, and $\delta S_k = \delta S_k(\mathbf{z}) = \delta S_k(\delta f, \delta g)$ are formed from the coefficients of the polynomials $f, g, \delta f$ and δg, where the coefficients of δf and δg form the minimised vector $\mathbf{z}$.

However, if the inexact polynomials f and g are considered, then for all values of $k = 1, \ldots, n$ the Sylvester matrix $S_k(f,g)$ is of full column rank, i.e. for every value $k = 1, \ldots, n$

$$\mathbf{c_k} \notin \text{Range}(A_k). \tag{5.2}$$

Moreover, let $\mathbf{c_k^i}$ be the ith column of $S_k(f,g)$, $i = 1, \ldots, m+n-2k+2$, i.e.

$$S_k(f,g) = [\mathbf{c_k^1}, \mathbf{c_k^2}, \ldots, \mathbf{c_k^{m+n-2k+2}}],$$

and set, for simplicity, for each $p = 1, \ldots, m+n-2k+2$

$$\mathbf{c_p} = \mathbf{c_k^p},$$
$$T_p = [\mathbf{c_1}, \ldots, \mathbf{c_{p-1}}, \mathbf{c_{p+1}}, \ldots, \mathbf{c_{m+n-2k+2}}],$$

i.e. T_p is formed from the remaining columns of S_k omitting $\mathbf{c_p}$. Then the full rank of $S(f,g)$ implies

$$\mathbf{c_p} \notin \text{Range}(T_p), \quad \forall p = 1, \ldots, m+n-2k+2, \tag{5.3}$$

and so $T_p \mathbf{x_p} = \mathbf{c_p}$ does not possess any solution for all $p = 1, \ldots, m+n-2k+2$ and $k = 1, \ldots, n$.

We can make a conclusion that for exact polynomials taking the first column of the Sylvester matrix S_k to be $\mathbf{c_k}$ in solving (4.8), i.e. $A_k\mathbf{x} = \mathbf{c_k}$, is reasonable, since the system (4.7) has a solution. However, in practice there is no good reason to prefer the first column when inexact polynomials are considered. Moreover, solving (5.1) with the first column of S_k as $\mathbf{c}_k$ may return absolutely inaccurate results while taking other columns can give acceptable results.

Since (5.3) holds, it can be therefore more convenient the following choice:

$$\mathbf{c_p^\star} = \arg \min_{1\le p\le m+n-2k+2} \|\mathbf{c_p} - T_p\mathbf{x_p}\|. \tag{5.4}$$

This choice is rather heuristic and follows from (5.3) and the notes above.

Once the column $\mathbf{c_p} = \mathbf{c_p^\star}$ is computed, then we can partition $S_k(f, g)$ and $\delta S_k(\delta f, \delta g)$ to the pth columns $\mathbf{c_p}$ and $\mathbf{h_p}$ and the matrices A_p and E_p formed from the remaining columns of S_k and δS_k, respectively, for the corresponding values of k and p.

The STLS problem (4.8) remains the same except where the first column $\mathbf{c_k}$ of S_k is replaced with $\mathbf{c}_p$ that solves (5.4), *etc.*, i.e.

$$\begin{aligned} &\min_{\mathbf{z},\mathbf{x}} \|\mathbf{z}\| \\ &\text{subject to } (T_p + E_p)\mathbf{x} = \mathbf{c_p} + \mathbf{h_p} \\ &\text{and } [\mathbf{h_p}, E_p] \text{ is of the same structure as } [\mathbf{c_p}, A_p] \end{aligned} \tag{5.5}$$

is solved STLS problem.

The methods of SNTLN and LNSNTLS are applied to the preprocessed polynomials by the operations from Section 1.3. That is why all possible polynomials appearing in the preprocessing procedure are briefly summarised in the following Section 5.2. Methods themself are then described in Sections 5.3 and 5.4.

We fix the parameter k smaller than the degree of the exact GCD of $\hat{f}$ and $\hat{g}$ to make our explanations as clear as possible and easier to follow for readers. The polynomials $\hat{f}$ and $\hat{g}$ are the theoretically exact forms of perturbed inexact polynomials f and g with a non-trivial GCD. In real computation the method SNTLN/LNSNTLS is applied to the first Sylvester subresultant $S_n = S_n(f, g)$ and it computes approximations of $\mathbf{x}$ and $\mathbf{z}$ until (5.5) has a solution. If there is a solution, then the *nearness* condition is checked. If $\|\mathbf{z}\| = \|\delta\mathbf{f}\| + \|\delta\mathbf{g}\|$ is less than a prescribed tolerance, then we can stop our computations with the degree of the AGCD n and the AGCD that is the exact GCD of $f+\delta f$ and $g+\delta g$. Coefficients of the AGCD can be computed by the Gauss-Newton method applied to the corresponding nth Sylvester matrix $S_n(f+\delta f, g+\delta g)$. However, if $\mathbf{z}$ is greater that the tolerance, then we continue solving (5.5) with S_{n-1}, *etc.*

5.2 Polynomials in algorithms

From Section 1.3 we know that it may be beneficial to run the preprocessing operations on polynomials before applying an AGCD solver. It was shown that the preprocessing operations make Sylvester matrices better balanced and their rank can be better defined (numerical-rank gap criterion).

Recall that the preprocessing operations consist of the normalisation by the geometric mean discussed in Section 1.3.1 and the scaling that includes two parameters α and γ, Section 1.3.2.

At the beginning we assume that the inexact polynomials

$$\begin{aligned} f(x) &= \sum_{i=0}^{m} a_i x^{m-i}, \\ g(x) &= \sum_{j=0}^{n} b_j x^{n-j} \end{aligned} \tag{5.6}$$

are given.

After the normalisation by the geometric mean they become

$$\begin{aligned} f(x) &= \sum_{i=0}^{m} \bar{a}_i x^{m-i}, \qquad \bar{a}_i = \frac{a_i}{(\prod_{k=0}^{m} |a_k|)^{\frac{1}{m+1}}}, \\ g(x) &= \sum_{j=0}^{n} \bar{b}_j x^{n-j}, \qquad \bar{b}_j = \frac{b_j}{(\prod_{k=0}^{n} |b_k|)^{\frac{1}{n+1}}}. \end{aligned} \tag{5.7}$$

Then the substitution (1.10), $x = \gamma w$, is carried out and the polynomials are transformed to the forms

$$\begin{aligned} \bar{f}(w) &= \sum_{i=0}^{m} (\bar{a}_i \gamma^{m-i}) w^{m-i}, \qquad \bar{a}_i = \frac{a_i}{(\prod_{k=0}^{m} |a_k|)^{\frac{1}{m+1}}}, \\ \bar{g}(w) &= \sum_{j=0}^{n} (\bar{b}_j \gamma^{n-j}) w^{n-j}, \qquad \bar{b}_j = \frac{b_j}{(\prod_{k=0}^{n} |b_k|)^{\frac{1}{n+1}}}. \end{aligned} \tag{5.8}$$

The liner programming problem (1.12) gives the parameters α_{opt} and γ_{opt} that is used in (5.8). After the normalisation of (5.8) we finally get

$$\begin{aligned} \bar{f}(w) &= \sum_{i=0}^{m} (\bar{a}_i^* \gamma_{opt}^{m-i}) w^{m-i}, \qquad \bar{a}_i^* = \frac{\bar{a}_i}{\left(\prod_{k=0}^{m} |\bar{a}_k \gamma_{opt}^{m-k}|\right)^{\frac{1}{m+1}}}, \\ \bar{g}(w) &= \sum_{j=0}^{n} (\bar{b}_j^* \gamma_{opt}^{n-j}) w^{n-j}, \qquad \bar{b}_j^* = \frac{\bar{b}_j}{\left(\prod_{k=0}^{n} |\bar{b}_k \gamma_{opt}^{n-k}|\right)^{\frac{1}{n+1}}}, \end{aligned} \tag{5.9}$$

with $\bar{a}_i$, $i = 0, \ldots, m$ and $\bar{b}_j$, $j = 0, \ldots, n$ defined in (5.7).

Studied methods are then applied to the Sylvester matrix $S_k(\bar{f}, \alpha_{opt}\bar{g})$. However, the parameters α_{opt} and γ_{opt} are not fixed through the computations. At each step of the iteration these constants are modified and α_{opt} and γ_{opt} are taken as their first approximations. Hence, the subscript *opt* of α and γ is omitted from notation.

5.3 Structured non-linear total least norm

The method of SNTLN solves (5.5) for the preprocessed data, i.e. computes a structured low rank approximation of $S(\bar{f}, \alpha_{opt}\bar{g})$ with the smallest possible rank $m+n-k$, where $\bar{f}$ and $\bar{g}$ are inexact preprocessed polynomials defined in (5.9), $k \in \{1, \ldots, \hat{d}\}$ and $\hat{d}$ is the degree of the GCD of the theoretically exact forms $\hat{f}$ and $\hat{g}$ of the inexact polynomials f and g. To explain the method of SNTLN properly, the papers [13, 25] are followed. However, there are some differences caused by choosing *optimal column* (5.4) discussed in Section 5.1.

The method of SNTLN becomes non-linear from the STLN method by considering the preprocessing operations, since the elements of the involved Sylvester matrix are differentiable non-linear functions of one or more parameters.

The method of SNTLN applied to the Sylvester matrix $S(\bar{f}, \alpha_{opt}\bar{g})$ computes another Sylvester matrix $\delta S(\delta\bar{f}, \alpha\delta\bar{g})$ with the same structure as $S(\bar{f}, \alpha_{opt}\bar{g})$ so that

$$S(\tilde{f}, \tilde{\alpha}\tilde{g}) := S(\bar{f} + \delta\bar{f}, \alpha_{opt}\bar{g} + \alpha\delta\bar{g}) = S(\bar{f}, \alpha_{opt}\bar{g}) + \delta S(\delta\bar{f}, \alpha\delta\bar{g}) \tag{5.10}$$

is the rank deficient matrix. Here, we set

$$\tilde{f}(w) = \bar{f}(w) + \delta\bar{f}(w), \quad \tilde{g}(w) = \bar{g}(w) + \delta\bar{g}(w) \quad \text{and} \quad \tilde{\alpha} = \alpha_{opt} + \alpha.$$

The method of SNTLN is an iterative method. So, it is convenient to write the polynomials in (5.9) in one unique form that appears at each step of the algorithm. Let therefore,

$$\bar{f}(w) = \sum_{i=0}^{m} (\bar{a}_i \gamma^{m-i}) w^{m-i} \qquad \text{and} \qquad \bar{g}(w) = \sum_{j=0}^{n} (\bar{b}_j \gamma^{n-j}) w^{n-j} \tag{5.11}$$

be these polynomial forms, where $\bar{a}_i = \bar{a}_i^*$ and $\bar{b}_j = \bar{b}_j^*$ for $i = 0, \ldots, m$ and $j = 0, \ldots, n$. The computed constants α_{opt} and γ_{opt} are used in the first iteration and then updated in the following iterations.

The subresultant matrix $S_k = S_k(\bar{f}, \alpha\bar{g}) \in \mathbb{R}^{(m+n-k+1)\times(m+n-2k+2)}$ is now

$$S_k = \begin{bmatrix} \bar{a}_0\gamma^m & & & \alpha\bar{b}_0\gamma^n & & \\ \bar{a}_1\gamma^{m-1} & \ddots & & \alpha\bar{b}_1\gamma^{n-1} & \ddots & \\ \vdots & \ddots & \bar{a}_0\gamma^m & \vdots & \ddots & \alpha\bar{b}_0\gamma^n \\ \bar{a}_{m-1}\gamma & \ddots & \bar{a}_1\gamma^{m-1} & \alpha\bar{b}_{n-1}\gamma & \ddots & \alpha\bar{b}_1\gamma^{n-1} \\ \bar{a}_m & \ddots & \vdots & \alpha\bar{b}_n & \ddots & \vdots \\ & \ddots & \bar{a}_{m-1}\gamma & & \ddots & \alpha\bar{b}_{n-1}\gamma \\ & & \bar{a}_m & & & \alpha\bar{b}_n \end{bmatrix}. \tag{5.12}$$

Similarly to Section 5.1, let $\mathbf{c_k^i}$ be the ith column of S_k, $i = 1, \ldots, m+n-2k+2$, i.e.

$$S_k = \left[\mathbf{c_k^1}, \mathbf{c_k^2}, \ldots, \mathbf{c_k^{m+n-2k+2}}\right],$$

and, again, set for $p = 1, \ldots, m + n - 2k + 2$

$$\mathbf{c_p} = \mathbf{c_p}(\alpha, \gamma) = \mathbf{c_k^p},$$

$$T_p = T_p(\alpha, \gamma) = [\mathbf{c_1}, \ldots, \mathbf{c_{p-1}}, \mathbf{c_{p+1}}, \ldots, \mathbf{c_{m+n-2k+2}}],$$

where $\mathbf{c_p} \in \mathbb{R}^{m+n-k+1}$ and $T_p \in \mathbb{R}^{(m+n-k+1)\times(m+n-2k+1)}$.

The entries of the Sylvester matrix $\delta S(\delta\bar{f}, \alpha\delta\bar{g})$ are perturbations of the coefficients of $\bar{f}$ and $\alpha\bar{g}$. Let us denote the perturbations of $\bar{f}$ as

$$\delta\bar{f}_i = z_i\gamma^{m-i}, \qquad i = 0, \ldots, m$$

and the perturbations of $\alpha\bar{g}$ as

$$\alpha\delta\bar{g}_i = \alpha z_{m+1+i}\gamma^{n-i}, \qquad i = 0, \ldots, n.$$

Hence, the perturbations of the coefficients of f and g create a vector

$$\mathbf{z} = [z_0, z_1, \ldots, z_{m+n+1}]^T \in \mathbb{R}^{m+n+2}.$$

From the perturbation we can now construct the kth subresultant matrix δS_k, which is obtained by deleting the corresponding number of rows and columns from $\delta S(\delta\bar{f}, \alpha\delta\bar{g})$. In particular,

$$\delta S_k = \begin{bmatrix} z_0\gamma^m & & & \alpha z_{m+1}\gamma^n & & \\ z_1\gamma^{m-1} & \ddots & & \alpha z_{m+2}\gamma^{n-1} & \ddots & \\ \vdots & \ddots & z_0\gamma^m & \vdots & \ddots & \alpha z_{m+1}\gamma^n \\ z_{m-1}\gamma & \ddots & z_1\gamma^{m-1} & \alpha z_{m+n}\gamma & \ddots & \alpha z_{m+2}\gamma^{n-1} \\ z_m & \ddots & \vdots & \alpha z_{m+n+1} & \ddots & \vdots \\ & \ddots & z_{m-1}\gamma & & \ddots & \alpha z_{m+n}\gamma \\ & & z_m & & & \alpha z_{m+n+1} \end{bmatrix}, \tag{5.13}$$

$\delta S_k = \delta S_k(\alpha, \gamma, \mathbf{z}) \in \mathbb{R}^{(m+n-k+1)\times(m+n-2k+2)}$. If the jth column of δS_k is denoted by $\mathbf{h_k^j}$, $j = 1, \ldots, m + n - 2k + 2$, then the matrix δS_k can be partitioned in the same fashion as S_k into the pth column $\mathbf{h_p} = \mathbf{h_p}(\alpha, \gamma, \mathbf{z}) = \mathbf{h_k^p} \in \mathbb{R}^{m+n-k+1}$ and the matrix $E_p \in \mathbb{R}^{(m+n-k+1)\times(m+n-2k+1)}$ of the remaining columns,

$$E_p = E_p(\alpha, \gamma, \mathbf{z}) = [\mathbf{h_1}, \ldots, \mathbf{h_{p-1}}, \mathbf{h_{p+1}}, \ldots, \mathbf{h_{m+n-2k+2}}].$$

The STLS problem (5.5) is equivalently stated as finding $\mathbf{z}, \mathbf{x}, \alpha$ and γ that solve

$$\begin{aligned} &\min_{\mathbf{z},\mathbf{x}} \|\mathbf{z}\| \\ &\text{subject to } (T_p(\alpha, \gamma) + E_p(\alpha, \gamma, \mathbf{z}))\mathbf{x} = \mathbf{c_p}(\alpha, \gamma) + \mathbf{h_p}(\alpha, \gamma, \mathbf{z}) \\ &\text{and } [\mathbf{h_p}, E_p] \text{ is of the same structure as } [\mathbf{c_p}, A_p] \end{aligned} \tag{5.14}$$

where $\mathbf{x} \in \mathbb{R}^{m+n-2k+1}$ is the parameter vector. Similarly to Section 4.3, the constrained conditions in (5.14), i.e.

$$r(\mathbf{z}, \mathbf{x}, \alpha, \gamma) = (T_p(\alpha, \gamma) + E_p(\alpha, \gamma, \mathbf{z}))\,\mathbf{x} - \mathbf{c_p}(\alpha, \gamma) - \mathbf{h_p}(\alpha, \gamma, \mathbf{z}) \tag{5.15}$$

are linearised around the current solution point $[\mathbf{z}^T, \mathbf{x}^T, \alpha, \gamma]^T$ yielding the increments $\Delta\alpha$, $\Delta\gamma$, $\mathbf{\Delta x}$ and $\mathbf{\Delta z}$ of the current values α, γ, $\mathbf{x}$ and $\mathbf{z}$, respectively. Let us define $\tilde{r}$,

$$\tilde{r} := r(\mathbf{z} + \mathbf{\Delta z}, \mathbf{x} + \mathbf{\Delta x}, \alpha + \mathbf{\Delta}\alpha, \gamma + \mathbf{\Delta}\gamma). \tag{5.16}$$

We have already seen in Section 4.3 that the expansion of $\tilde{r}$ can be written as

$$\tilde{r} = r(\mathbf{z}, \mathbf{x}, \alpha, \gamma) + J \begin{bmatrix} \mathbf{\Delta z} \\ \mathbf{\Delta x} \\ \mathbf{\Delta}\alpha \\ \mathbf{\Delta}\gamma \end{bmatrix} + h.o.t. \tag{5.17}$$

where J is the Jocobian matrix of the residual r. However,

$$\begin{aligned} \tilde{r} =& r(\mathbf{z}, \mathbf{x}, \alpha, \gamma) + \left(\left(\frac{\partial T_p}{\partial \gamma} + \frac{\partial E_p}{\partial \gamma} \right) \mathbf{x} - \left(\frac{\partial \mathbf{c_p}}{\partial \gamma} + \frac{\partial \mathbf{h_p}}{\partial \gamma} \right) \right) \mathbf{\Delta}\gamma \\ &+ (T_p + E_p)\mathbf{\Delta x} + \left(\left(\frac{\partial T_p}{\partial \alpha} + \frac{\partial E_p}{\partial \alpha} \right) \mathbf{x} - \left(\frac{\partial \mathbf{c_p}}{\partial \alpha} + \frac{\partial \mathbf{h_p}}{\partial \alpha} \right) \right) \mathbf{\Delta}\alpha + (Y_p - P_p)\, \mathbf{\Delta z}, \end{aligned} \tag{5.18}$$

where $\dfrac{\partial \mathbf{c_p}}{\partial \gamma}$ is the pth column of $\dfrac{\partial S_k}{\partial \gamma}$ and $\dfrac{\partial T_p}{\partial \gamma}$ is the matrix formed from the remaining columns of $\dfrac{\partial S_k}{\partial \gamma}$, where $\dfrac{\partial S_k}{\partial \gamma}$ is equal to

$$\begin{bmatrix} m\bar{a}_0\gamma^{m-1} & & & n\alpha\bar{b}_0\gamma^{n-1} & & \\ (m-1)\bar{a}_1\gamma^{m-2} & \ddots & & (n-1)\alpha\bar{b}_1\gamma^{n-2} & \ddots & \\ \vdots & \ddots & m\bar{a}_0\gamma^{m-1} & \vdots & \ddots & n\alpha\bar{b}_0\gamma^{n-1} \\ \bar{a}_{m-1} & \ddots & (m-1)\bar{a}_1\gamma^{m-2} & \alpha\bar{b}_{n-1} & \ddots & (n-1)\alpha\bar{b}_1\gamma^{n-2} \\ 0 & \ddots & \vdots & 0 & \ddots & \vdots \\ & \ddots & \bar{a}_{m-1} & & \ddots & \alpha\bar{b}_{n-1} \\ & & 0 & & & 0 \end{bmatrix}.$$

Similarly, $\dfrac{\partial \mathbf{h_p}}{\partial \gamma}$ is the pth column of $\dfrac{\partial \delta S_k}{\partial \gamma}$ and $\dfrac{\partial E_p}{\partial \gamma}$ is the matrix formed from the remaining columns of $\dfrac{\partial \delta S_k}{\partial \gamma}$, where $\dfrac{\partial \delta S_k}{\partial \gamma}$ is equal to

$$\begin{bmatrix} mz_0\gamma^{m-1} & & & n\alpha z_{m+1}\gamma^{n-1} & & \\ (m-1)z_1\gamma^{m-2} & \ddots & & (n-1)\alpha z_{m+2}\gamma^{n-2} & \ddots & \\ \vdots & \ddots & mz_0\gamma^{m-1} & \vdots & \ddots & n\alpha z_{m+1}\gamma^{n-1} \\ z_{m-1} & \ddots & (m-1)z_1\gamma^{m-2} & \alpha z_{m+n} & \ddots & (n-1)\alpha z_{m+2}\gamma^{n-2} \\ 0 & \ddots & \vdots & 0 & \ddots & \vdots \\ & \ddots & z_{m-1} & & \ddots & \alpha z_{m+n} \\ & & 0 & & & 0 \end{bmatrix}.$$

The partial derivations $\frac{\partial T_p}{\partial \alpha}$ and $\frac{\partial E_p}{\partial \alpha}$ are formed from the columns of $\frac{\partial S_k}{\partial \alpha}$ and $\frac{\partial B_k}{\partial \alpha}$, respectively, where the pth columns $\frac{\partial \mathbf{c_p}}{\partial \alpha}$ and $\frac{\partial \mathbf{h_p}}{\partial \alpha}$ are omitted, and where $\frac{\partial S_k}{\partial \alpha}$ is equal to

$$\begin{bmatrix} 0 & & & \bar{b}_0\gamma^n & & \\ 0 & \ddots & & \bar{b}_1\gamma^{n-1} & \ddots & \\ \vdots & \ddots & 0 & \vdots & \ddots & \bar{b}_0\gamma^n \\ 0 & \ddots & 0 & \bar{b}_{n-1}\gamma & \ddots & \bar{b}_1\gamma^{n-1} \\ 0 & \ddots & \vdots & \bar{b}_n & \ddots & \vdots \\ & \ddots & 0 & & \ddots & \bar{b}_{n-1}\gamma \\ & & 0 & & & \bar{b}_n \end{bmatrix}$$

and the matrix $\frac{\partial \delta S_k}{\partial \alpha}$ is equal to

$$\begin{bmatrix} 0 & & & z_{m+1}\gamma^n & & \\ 0 & \ddots & & z_{m+2}\gamma^{n-1} & \ddots & \\ \vdots & \ddots & 0 & \vdots & \ddots & z_{m+1}\gamma^n \\ 0 & \ddots & 0 & z_{m+n}\gamma & \ddots & z_{m+2}\gamma^{n-1} \\ 0 & \ddots & \vdots & z_{m+n+1} & \ddots & \vdots \\ & \ddots & 0 & & \ddots & z_{m+n}\gamma \\ & & 0 & & & z_{m+n+1} \end{bmatrix}.$$

A matrix $P_p = P_p(\alpha, \gamma) \in \mathbb{R}^{(m+n-k+1)\times(m+n+2)}$ is chosen so that

$$\mathbf{h_p} = P_p \mathbf{z}. \tag{5.19}$$

Since the column $\mathbf{h_p}$ is not dependent on α for every $p = 1, \ldots, n-k+1$, the matrix P_p is defined by[1]

$$P_p = \begin{array}{|c|c|} \hline 0 & 0 \\ \hline G_1 & 0 \\ \hline 0 & 0 \\ \hline \end{array} \begin{array}{l} j-1 \\ m+1 \\ n-j-k+1 \end{array}$$
$$\begin{array}{cc} m+1 & n+1 \end{array}$$

where $G_1 = \operatorname{diag}[\gamma^m, \gamma^{m-1}, \ldots, \gamma, 1] \in \mathbb{R}^{(m+1)\times(m+1)}$. On the other side, for $p =$

[1]The numbers like $n+1$ and $n-j-k+1$ below and on the right of the boxes in the definition of the matrix P_p denote the number of columns and rows, respectively.

$n-k+2,\ldots,m+n-2k+2$ the matrix P_p is

$$P_p = \begin{array}{|c|c|l} \hline 0 & 0 & j-n+k-2 \\ \hline 0 & G_2 & n+1 \\ \hline 0 & 0 & m+n-j-2k+2 \\ \hline \end{array} \quad \begin{array}{c} \\ \\ \\ \end{array}$$
$$\begin{array}{cc} m+1 & n+1 \end{array}$$

where $G_2 = \operatorname{diag}[\alpha\gamma^n, \alpha\gamma^{n-1}, \ldots, \alpha\gamma, \alpha] \in \mathbb{R}^{(n+1)\times(n+1)}$, because $\mathbf{h_p}$ is the function of α.

Finally, a matrix $Y_p = Y_p(\alpha, \gamma, \mathbf{x}) \in \mathbb{R}^{(m+n-k+1)\times(m+n+2)}$ is defined so that

$$Y_p\mathbf{z} = E_p\mathbf{x}. \tag{5.20}$$

The structure of Y_p is clear from the following example.

Example 5.3.1. Consider two polynomials of degrees $m = 3$ and $n = 4$,

$$\bar{f}(w) = \bar{a}_0\gamma^3w^3 + \bar{a}_1\gamma^2w^2 + \bar{a}_2\gamma w + \bar{a}_3,$$
$$\bar{g}(w) = \bar{b}_0\gamma^4w^4 + \bar{b}_1\gamma^3w^3 + \bar{b}_2\gamma^2w^2 + \bar{b}_3\gamma w + \bar{b}_4.$$

and set $k = 2$. Then

$$\mathbf{z} = \begin{bmatrix} z_0 & z_1 & z_2 & z_3 & z_4 & z_5 & z_6 & z_7 & z_8 \end{bmatrix}^T,$$

$$\mathbf{x} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix}^T$$

and for the column $p = 2$ we have

$$T_2 = \begin{bmatrix} \bar{a}_0\gamma^3 & 0 & \alpha\bar{b}_0\gamma^4 & 0 \\ \bar{a}_1\gamma^2 & 0 & \alpha\bar{b}_1\gamma^3 & \alpha\bar{b}_0\gamma^4 \\ \bar{a}_2\gamma & \bar{a}_0\gamma^3 & \alpha\bar{b}_2\gamma^2 & \alpha\bar{b}_1\gamma^3 \\ \bar{a}_3 & \bar{a}_1\gamma^2 & \alpha\bar{b}_3\gamma & \alpha\bar{b}_2\gamma^2 \\ 0 & \bar{a}_2\gamma & \alpha\bar{b}_4 & \alpha\bar{b}_3\gamma \\ 0 & \bar{a}_3 & 0 & \alpha\bar{b}_4 \end{bmatrix}, \quad \mathbf{c_2} = \begin{bmatrix} 0 \\ \bar{a}_0\gamma^3 \\ \bar{a}_1\gamma^2 \\ \bar{a}_2\gamma \\ \bar{a}_3 \\ 0 \end{bmatrix},$$

$$E_2 = \begin{bmatrix} z_0\gamma^3 & 0 & \alpha z_4\gamma^4 & 0 \\ z_1\gamma^2 & 0 & \alpha z_5\gamma^3 & \alpha z_4\gamma^4 \\ z_2\gamma & z_0\gamma^3 & \alpha z_6\gamma^2 & \alpha z_5\gamma^3 \\ z_3 & z_1\gamma^2 & \alpha z_7\gamma & \alpha z_6\gamma^2 \\ 0 & z_2\gamma & \alpha z_8 & \alpha z_7\gamma \\ 0 & z_3 & 0 & \alpha z_8 \end{bmatrix}, \quad \mathbf{h_2} = \begin{bmatrix} 0 \\ z_0\gamma^3 \\ z_1\gamma^2 \\ z_2\gamma \\ z_3 \\ 0 \end{bmatrix},$$

$$P_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \gamma^3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \gamma^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \gamma & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

and

$$Y_2 = \begin{bmatrix} \gamma^3 x_1 & 0 & 0 & 0 & \alpha\gamma^4 x_3 & 0 & 0 & 0 & 0 \\ 0 & \gamma^2 x_1 & 0 & 0 & \alpha\gamma^4 x_4 & \alpha\gamma^3 x_3 & 0 & 0 & 0 \\ \gamma^3 x_2 & 0 & \gamma x_1 & 0 & 0 & \alpha\gamma^3 x_4 & \alpha\gamma^2 x_3 & 0 & 0 \\ 0 & \gamma^2 x_2 & 0 & x_1 & 0 & 0 & \alpha\gamma^2 x_4 & \alpha\gamma x_3 & 0 \\ 0 & 0 & \gamma x_2 & 0 & 0 & 0 & 0 & \alpha\gamma x_4 & \alpha x_3 \\ 0 & 0 & 0 & x_2 & 0 & 0 & 0 & 0 & \alpha x_4 \end{bmatrix}.$$

And, for example, for $p = 4$ matrices P_4 and Y_4 are

$$P_4 = \begin{bmatrix} 0 & 0 & 0 & 0 & \alpha\gamma^4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \alpha\gamma^3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \alpha\gamma^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha\gamma & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

and

$$Y_4 = \begin{bmatrix} \gamma^3 x_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \gamma^3 x_2 & \gamma^2 x_1 & 0 & 0 & \alpha\gamma^4 x_4 & 0 & 0 & 0 & 0 \\ \gamma^3 x_3 & \gamma^2 x_2 & \gamma x_1 & 0 & 0 & \alpha\gamma^3 x_4 & 0 & 0 & 0 \\ 0 & \gamma^2 x_3 & \gamma x_2 & x_1 & 0 & 0 & \alpha\gamma^2 x_4 & 0 & 0 \\ 0 & 0 & \gamma x_3 & x_2 & 0 & 0 & 0 & \alpha\gamma x_4 & 0 \\ 0 & 0 & 0 & x_3 & 0 & 0 & 0 & 0 & \alpha x_4 \end{bmatrix}.$$

It is easy to verify that (5.19) and (5.20) are satisfied. ♣

Hence, the Jacobian function $J \in \mathbb{R}^{(m+n-k+1)\times(2m+2n-2k+5)}$ is the matrix of the form

$$J = \left[Y_p - P_p, T_p + E_p, \left(\frac{\partial T_p}{\partial \alpha} + \frac{\partial E_p}{\partial \alpha}\right)\mathbf{x} - \left(\frac{\partial \mathbf{c_p}}{\partial \alpha} + \frac{\partial \mathbf{h_p}}{\partial \alpha}\right), \ldots \right.$$
$$\left. \ldots \left(\frac{\partial T_p}{\partial \gamma} + \frac{\partial E_p}{\partial \gamma}\right)\mathbf{x} - \left(\frac{\partial \mathbf{c_p}}{\partial \gamma} + \frac{\partial \mathbf{h_p}}{\partial \gamma}\right) \right].$$

Neglecting *h.o.t.* in (5.17) and setting $\tilde{r} = 0$ along with

$$\mathbf{s} = r(\mathbf{z}, \mathbf{x}, \alpha, \gamma) \in \mathbb{R}^{m+n-k+1} \quad \text{and} \quad \mathbf{y} = \begin{bmatrix} \delta\mathbf{z} \\ \delta\mathbf{x} \\ \delta\alpha \\ \delta\gamma \end{bmatrix} \in \mathbb{R}^{2m+2n-2k+5}.$$

gives the system

$$J\mathbf{y} = -\mathbf{s}. \tag{5.21}$$

that needs to be solved.

The objective function in (5.14) that is minimised is $\|\mathbf{z}\|$, where the first $m+1$ entries of $\mathbf{z}$ are perturbations of the coefficients of f and the remaining $n+1$ entries

are perturbations of the coefficients of g. We can again define a diagonal weighting matrix $D \in \mathbb{R}^{(m+n+4)\times(2m+2n-2k+5)}$,

$$D = \begin{bmatrix} \tilde{D} & & & \\ & 0 & & \\ & & d & \\ & & & 1 \end{bmatrix}, \quad \tilde{D} = \begin{bmatrix} D_1 & \\ & D_2 \end{bmatrix} \tag{5.22}$$

where $D_1 = (n-k+1)I_{m+1}$ and $D_2 = (m-k+1)I_{n+1}$ represent the repetition of the elements of $\mathbf{z}$ in E_k and d counts the repetition of α in E_k that can be at most $(n+1)\times(m-k+1)$.

Hence, computing increments $\boldsymbol{\Delta}\mathbf{z}, \boldsymbol{\Delta}\mathbf{x}, \boldsymbol{\Delta}\alpha$ and $\boldsymbol{\Delta}\gamma$ can be done through the minimising

$$\left\| \begin{bmatrix} \tilde{D}(\mathbf{z}+\boldsymbol{\Delta}\mathbf{z}-\mathbf{z_0}) \\ d(\alpha+\boldsymbol{\Delta}\alpha-\alpha_{opt}) \\ \gamma+\boldsymbol{\Delta}\gamma-\gamma_{opt} \end{bmatrix} \right\| = \left\| D \begin{bmatrix} \boldsymbol{\Delta}\mathbf{z} \\ \boldsymbol{\Delta}\mathbf{x} \\ \Delta\alpha \\ \Delta\gamma \end{bmatrix} - \begin{bmatrix} -\tilde{D}\mathbf{z} \\ d(\alpha_{opt}-\alpha) \\ \gamma_{opt}-\gamma \end{bmatrix} \right\| =: \left\| D\mathbf{y}-\mathbf{t} \right\|$$

where

$$\mathbf{t} = \begin{bmatrix} -\tilde{D}\mathbf{z} \\ d(\alpha_{opt}-\alpha) \\ \gamma_{opt}-\gamma \end{bmatrix} \in \mathbb{R}^{m+n+4}.$$

Finally, it is necessary to solve the LSE problem

$$\begin{aligned} &\min_{\mathbf{y}} \left\| D\mathbf{y}-\mathbf{t} \right\| \\ &\text{subject to } J\mathbf{y} = -\mathbf{s} \end{aligned} \tag{5.23}$$

in order to compute the vector of increments $\mathbf{y}$.

The SNTLN algorithm for the computation of the AGCD is almost explained. It remains to said that the initial approximations are: $\mathbf{z}_0 = \mathbf{0}$ since the smallest perturbations are desired to be computed and the given polynomials are inexact; α_{opt} and γ_{opt} are computed from the preprocessing operations described in Section 1.3; $\mathbf{x}_0$ is a least square solution to a problem that arises from (5.15) by setting $r = \mathbf{0}$ and considering $\mathbf{z}_0$, α_{opt} and γ_{opt}, i.e.

$$\mathbf{x}_0 = \arg\min_{\mathbf{x}} \left\| T_p(\alpha_{opt}, \gamma_{opt})\mathbf{x} - \mathbf{c_p}(\alpha_{opt}, \gamma_{opt}) \right\|, \tag{5.24}$$

.

In each step of the iterative algorithm data are updated by setting

$$\mathbf{z} := \mathbf{z}+\Delta\mathbf{z}, \quad \mathbf{x} := \mathbf{x}+\Delta\mathbf{x}, \quad \alpha := \alpha+\boldsymbol{\Delta}\alpha \quad \text{and} \quad \gamma := \gamma+\boldsymbol{\Delta}\gamma.$$

The following Algorithm 5.3.1 solves STLS problem (5.14), i.e. computes a low rank approximation of the full column rank matrix $S_k(f,g)$ of the inexact polynomials f and g. It is slightly different from the algorithm in [25]. In particular, here we assume that the optimal column $\mathbf{c_p^\star}$ satisfying (5.4) for some $p = 1, \ldots, m+n-2k+2$ is taken into account.

Algorithm 5.3.1 (SNTLN)**.**

Input: Inexact polynomials f and g of degrees m and n, tolerance $tol > 0$, degree k of the exact GCD.
Output: Structured low rank approximation of $S(\bar{f}, \alpha_{opt}\bar{g})$ of the rank loss k.

```
(1)  begin
(2)    apply preprocessing operations from Section 1.3 to f and g;
(3)    compute α_opt and γ_opt;
(4)    find an index p for the optimal column c*_p;
(5)    do
(6)      initialization
(7)        determine T_p, c_p, ∂T_p/∂α, ∂T_p/∂γ, ∂c_p/∂γ, ∂c_p/∂α, Y_p and P_p;
(8)        compute x_0 from (5.24);
(9)        set z = z_0 = 0;
(10)       calculate matrices J and D, vector s, and set t = 0;
(11)     while ‖r(z,z,α,γ)‖ / ‖c_p+h_p‖ ≥ tol do
(12)       solve the LSE problem (5.23);
(13)       set z := z + Δz, x := x + Δx, α := α + Δα and γ := γ + Δγ;
(14)       update data from z, x, α and γ;
(15)     end
(16)   end
(17) end
```

The parameter k is set to be fixed. However, this limitation can be very easily replaced by the **for** loop that goes from $k = n, n-1, \ldots, 1$ and computes for each k a low rank approximation of $S(\bar{f}, \alpha_{opt}\bar{g})$. From the computed approximation we can very easily obtain the norm of new polynomials δf and δg. Whenever these polynomials satisfy *nearness* condition from the AGCD definition, then we can stop the algorithm with $f + \delta f$ and $g + \delta g$ having a non-trivial GCD that is said to be the AGCD of f and g. The coefficient of the AGCD are then computed by the iterative refinement from Chapter 3.

5.4 Non-linear Lagrange-Newton STLS

The Non-linear Lagrange-Newton STLS (LNSNTLS) method is a natural extension of the LNSTLS method explained in Section 4.4 and similar to the extension of SNTLN from STLN.

From the beginning, in fact, LNSNTLS is very similar to SNTLN. Indeed, the LNSNTLS method is applied to the Sylvester matrix $S_k(\bar{f}, \alpha_{opt}\bar{g})$ where $\bar{f}$ and $\bar{g}$ are the preprocessed polynomials (5.9). Both methods include column pivoting, i.e. a column $\mathbf{c}_p$ taken from S_k that satisfies (5.3) is employed in further computations. And

both methods solve the structured TLS problem

$$\begin{aligned} &\min_{\mathbf{z},\mathbf{x}} \|\mathbf{z}\| \\ &\text{subject to } (T_p(\alpha,\gamma) + E_p(\alpha,\gamma,\mathbf{z}))\mathbf{x} = \mathbf{c_p}(\alpha,\gamma) + \mathbf{h_p}(\alpha,\gamma,\mathbf{z}) \\ &\text{and } [\mathbf{h_p}, E_p] \text{ is of the same structure as } [\mathbf{c_p}, A_p] \end{aligned} \tag{5.25}$$

for getting $\mathbf{z} \in \mathbb{R}^{m+n+2}$ that stores the perturbations δf and δg of the inexact polynomials f and g. In (5.25), $\mathbf{x} \in \mathbb{R}^{m+n-2k+1}$ is the parameter vector, $\mathbf{c}_p = \mathbf{c_p}(\alpha,\gamma)$ and $\mathbf{h}_p = \mathbf{h_p}(\alpha,\gamma,\mathbf{z})$ are the pth columns of $S_k(\bar{f}, \alpha\bar{g})$ in (5.12) and $\delta S_k(\delta\bar{f}, \alpha\delta\bar{g})$ in (5.13), respectively, $T_p = T_p(\alpha,\gamma)$ and $E_p = E_p(\alpha,\gamma,\mathbf{z})$ are formed from the remaining columns of the matrices.

The STLS problem (5.25) is solved by employing the Newton method to a Lagrangian function

$$L(\mathbf{z},\mathbf{x},\alpha,\gamma,\lambda) = 1/2\|\mathbf{z}\|^2 + \lambda^T r(\mathbf{z},\mathbf{x},\alpha,\gamma) \tag{5.26}$$

where $\lambda \in \mathbb{R}^{m+n-k+1}$ represents the Lagrange multipliers and the residual $r(\mathbf{z},\mathbf{x},\alpha,\gamma)$ is defined by

$$r(\mathbf{z},\mathbf{x},\alpha,\gamma) = (T_p(\alpha,\gamma) + E_p(\alpha,\gamma,\mathbf{z}))\,\mathbf{x} - \mathbf{c_p}(\alpha,\gamma) - \mathbf{h_p}(\alpha,\gamma,\mathbf{z}). \tag{5.27}$$

The Lagrangian function can be then written as

$$\begin{aligned} L(\mathbf{z},\mathbf{x},\alpha,\gamma,\lambda) &= 1/2\|\mathbf{z}\|^2 - \lambda^T\left(\mathbf{c_p} + \mathbf{h_p} - T_p\mathbf{x} - E_p\mathbf{x}\right) \\ &= 1/2\|\mathbf{z}\|^2 - \lambda^T\left(\mathbf{c_p} - T_p\mathbf{x} - (Y_p - P_p)\,\mathbf{z}\right). \end{aligned} \tag{5.28}$$

where $Y_p = Y_p(\alpha,\gamma,\mathbf{x})$ and $P_p = P_p(\alpha,\gamma)$ are the matrices that realise substitutions

$$Y_p\mathbf{z} = E_p\mathbf{x} \quad \text{and} \quad P_p\mathbf{z} = \mathbf{h}_p$$

shown in the previous section.

To find a solution to the STLS problem (5.25) it remains to find a stationary point of the Lagrangian function, i.e. a point $(\mathbf{z}^*,\mathbf{x}^*,\alpha^*,\gamma^*,\lambda^*)$ satisfying

$$\nabla L(\mathbf{z}^*,\mathbf{x}^*,\alpha^*,\gamma^*,\lambda^*) = \mathbf{0}. \tag{5.29}$$

Similarly to the LNSTLS method, the Newton method gives the system

$$H\begin{bmatrix}\boldsymbol{\Delta}\mathbf{z}\\ \boldsymbol{\Delta}\mathbf{x}\\ \boldsymbol{\Delta}\alpha\\ \boldsymbol{\Delta}\gamma\\ \boldsymbol{\Delta}\lambda\end{bmatrix} = -\nabla L(\mathbf{z},\mathbf{x},\alpha,\gamma\lambda) \tag{5.30}$$

that needs to be solved. Here, $\nabla L(\mathbf{z},\mathbf{x},\alpha,\gamma,\lambda) \in \mathbb{R}^{3m+3n-3k+6}$ is of the form

$$\nabla L(\mathbf{z},\mathbf{x},\alpha,\gamma,\lambda) = \begin{bmatrix} I_{m+n+2}\mathbf{z} + (Y_p - P_p)^T\lambda \\ (T_p + E_p)^T\lambda \\ (1)\lambda \\ (2)\lambda \\ r(\mathbf{z},\mathbf{x},\alpha,\gamma) \end{bmatrix} \tag{5.31}$$

and $H \in \mathbb{R}^{(3m+3n-3k+6)\times(3m+3n-3k+6)}$ is the Hessian matrix of $L(\mathbf{z}, \mathbf{x}, \alpha, \gamma, \lambda)$,

$$H = \begin{bmatrix} I_{m+n+2} & \frac{\partial(Y_p^T\lambda)}{\partial_x} & \left(\frac{\partial Y_p^T}{\partial\alpha} - \frac{\partial P_p^T}{\partial\alpha}\right)\lambda & \dots \\ \frac{\partial(E_p^T\lambda)}{\partial_z} & 0 & \left(\frac{\partial T_p^T}{\partial\alpha} + \frac{\partial E_p^T}{\partial\alpha}\right)\lambda & \dots \\ \lambda^T\left(\frac{\partial Y_p}{\partial\alpha} - \frac{\partial P_p}{\partial\alpha}\right) & \lambda^T\left(\frac{\partial T_p}{\partial\alpha} + \frac{\partial E_p}{\partial\alpha}\right) & 0 & \dots \\ \lambda^T\left(\frac{\partial Y_p}{\partial\gamma} - \frac{\partial P_p}{\partial\gamma}\right) & \lambda^T\left(\frac{\partial T_p}{\partial\gamma} + \frac{\partial E_p}{\partial\gamma}\right) & \frac{\partial^2 L}{\partial\gamma\alpha} & \dots \\ Y_p - P_p & T_p + E_p & (1) & \dots \end{bmatrix}$$

$$\begin{bmatrix} \dots & \left(\frac{\partial Y_p^T}{\partial\gamma} - \frac{\partial P_p^T}{\partial\gamma}\right)\lambda & (Y_p - P_p)^T \\ \dots & \left(\frac{\partial T_p^T}{\partial\gamma} + \frac{\partial E_p^T}{\partial\gamma}\right)\lambda & (T_p + E_p)^T \\ \dots & \frac{\partial^2 L}{\partial\alpha\gamma} & (1)^T \\ \dots & \frac{\partial^2 L}{\partial\gamma\gamma} & (2)^T \\ \dots & (2) & 0 \end{bmatrix} \tag{5.32}$$

where

$$(1) = -\left(\frac{\partial \mathbf{c}_p}{\partial\alpha} + \frac{\partial \mathbf{h}_p}{\partial\alpha} - \frac{\partial T_p}{\partial\alpha}\mathbf{x} - \frac{\partial E_p}{\partial\alpha}\mathbf{x}\right),$$
$$(2) = -\left(\frac{\partial \mathbf{c}_p}{\partial\gamma} + \frac{\partial \mathbf{h}_p}{\partial\gamma} - \frac{\partial T_p}{\partial\gamma}\mathbf{x} - \frac{\partial E_p}{\partial\gamma}\mathbf{x}\right).$$

Some of the involved structures are explicitly shown in the previous section about SNTLN, namely T_p, $\mathbf{c}_p$, E_p, $\mathbf{h}_p$, Y_p, P_p, $\frac{\partial T_p}{\partial\alpha}$, $\frac{\partial \mathbf{c}_p}{\partial\alpha}$, $\frac{\partial E_p}{\partial\alpha}$, $\frac{\partial \mathbf{h}_p}{\partial\alpha}$ $\frac{\partial T_p}{\partial\gamma}$, $\frac{\partial \mathbf{c}_p}{\partial\gamma}$, $\frac{\partial E_p}{\partial\gamma}$ and $\frac{\partial \mathbf{h}_k}{\partial\gamma}$. So showing their structures is omitted here. Other structures as $\frac{\partial(Y_p^T\lambda)}{\partial_x}$, $\frac{\partial(E_p^T\lambda)}{\partial_z}$, $\frac{\partial^2 L}{\partial\gamma\alpha}$, $\frac{\partial^2 L}{\partial\alpha\gamma}$ and $\frac{\partial^2 L}{\partial\gamma\gamma}$ are complicated. Therefore, they are approximated by the zero matrices of the corresponding sizes.

It remains to show matrices $\frac{\partial Y_p}{\partial\alpha}$, $\frac{\partial P_p}{\partial\alpha}$, $\frac{\partial Y_p}{\partial\gamma}$, $\frac{\partial P_p}{\partial\gamma}$, however it is not difficult to deduce them from the structures of Y_p and P_p given earlier.

The Hessian matrix H may become very large for polynomial of high degrees and so some complications may arise. After solving (5.30), new approximation of a stationary point are obtained by setting

$$\mathbf{z} = \mathbf{z}+\boldsymbol{\Delta}\mathbf{z}, \quad \mathbf{x} = \mathbf{x} + \boldsymbol{\Delta}\mathbf{x},$$
$$\alpha = \alpha + \boldsymbol{\Delta}\alpha, \quad \gamma = \gamma + \boldsymbol{\Delta}\gamma, \quad \lambda = \lambda + \boldsymbol{\Delta}\lambda.$$

All the conclusion made in the SNTLN method about the computation of the AGCD are valid in here. Hence, finally, we can formulate an algorithm LNSNTLS:

Algorithm 5.4.1 (LNSNTLS)**.**

Input: Inexact polynomials f and g of degrees m and n, tolerance $tol > 0$, degree k of the exact GCD.
Output: Structured low rank approximation of $S(\bar{f}, \alpha_{opt}\bar{g})$ of the rank loss k.

begin
  apply preprocessing operations from Section 1.3 to f and g;
  compute α_{opt} and γ_{opt};
  find an index p for the optimal column $\mathbf{c}_\mathbf{p}^*$;
  do
    initialization
      determine $T_p, \mathbf{c}_p, Y_p, P_p, \frac{\partial T_p}{\partial \alpha}, \frac{\partial \mathbf{c}_p}{\partial \alpha}, \frac{\partial T_p}{\partial \gamma}, \frac{\partial \mathbf{c}_p}{\partial \gamma}, \frac{\partial Y_p}{\partial \alpha}, \frac{\partial P_p}{\partial \alpha}, \frac{\partial Y_p}{\partial \gamma}$ and $\frac{\partial P_p}{\partial \gamma}$;
      compute $\mathbf{x}_0$ from (5.24);
      set $\mathbf{z}_0 = \mathbf{0}$ and $\lambda_0 = \mathbf{0}$;
      form the Hessian matrix H and the gradient ∇L;
    while $\frac{\|r(\mathbf{z},\mathbf{z},\alpha,\gamma)\|}{\|\mathbf{c_p}+\mathbf{h_p}\|} \geq tol$ **do**
      solve the system (5.30);
      set $\mathbf{z} := \mathbf{z} + \mathbf{\Delta z}, \mathbf{x} := \mathbf{x} + \mathbf{\Delta x}, \alpha := \alpha + \mathbf{\Delta}\alpha$
        $\gamma := \gamma + \mathbf{\Delta}\gamma$ and $\lambda = \lambda + \mathbf{\Delta}\lambda$;
      update data from $\mathbf{z}, \mathbf{x}, \alpha, \gamma$ and λ;
    end
  end
end

The presented algorithms for the AGCD computation are tested on examples in the next section.

5.5 Numerical examples

In this section we show examples on which the methods of STLN 4.3.1, LNSTLS 4.4.1, SNTLN 5.3.1 and LNSNTLS 5.4.1 are tested.

The considered exact polynomials $\hat{f}$ and $\hat{g}$ are always perturbed componentwisely by a noise of the signal-to-ratio $\epsilon = 10^{-8}$ thereby yielding inexact polynomials f and g. The stopping criteria $\frac{\|r(\mathbf{z},\mathbf{x})\|}{\|\mathbf{c}+\mathbf{h}\|}$ and $\frac{\|r(\mathbf{z},\mathbf{z},\alpha,\gamma)\|}{\|\mathbf{c_p}+\mathbf{h_p}\|}$, respectively, are controlled with the tolerance $tol = 10^{-12}$. The preprocessing operations from Section 1.3 are always performed and the column $\mathbf{c}_p^*$ minimising (5.4) computed.

We focus on computing low rank approximations of $S(f, g)$ and $S(\bar{f}, \alpha_{opt}\bar{g})$ in our examples. However, all the methods return also the vector of perturbations $\mathbf{z}$ that stores perturbations of the inexact polynomials f and g, namely $\mathbf{z} = [\delta\mathbf{f}^T, \delta\mathbf{g}^T]^T$, such that $f + \delta f$ and $g + \delta g$ have a non-trivial GCD, $\text{AGCD}(f, g) = \text{GCD}(f + \delta f, g + \delta g)$. Hence,

if the methods terminate successfully, then $S(f+\delta f, g+\delta g)$ should be rank deficient. We will therefore compare the singular values of $S(\hat{f},\hat{g})$, $S(f,g)$ and $S(f+\delta f, g+\delta g)$ in the figures.

Note that $k = \deg \mathrm{GCD}(\hat{f},\hat{g})$ is fixed if it is not said otherwise. The methods can be used directly for the computation of the AGCD starting from computing a low rank approximation of $S_n(f,g), S_{n-1}(f,g),\ldots$, however in this case another bounds on $\|\mathbf{z}\|$ have to be specified. The coefficients of the AGCD can be computed from $S(f+\delta f, g+\delta g)$ by the Gauss-Newton method 3.1.1.

Example 5.5.1. Let us firstly consider the polynomials $\hat{f}$ and $\hat{g}$ in (2.11) from Examples 2.5.3 and 3.2.1, i.e.

$$\begin{aligned}\hat{f}(x) &= (x+20.6)^2(x-4.7)^5(x-1.3)^4,\\ \hat{g}(x) &= (x+10.4)^3(x-4.7)^4(x-1.3)^3\end{aligned} \tag{5.33}$$

We have seen that the exact GCD of degree 7 of these polynomials can be computed by using the Gauss-Newton method from Chapter 3. However, if the polynomials are perturbed componentwisely with a noise of the signal-to-noise ratio $\epsilon = 10^{-8}$, then the numerical rank loss of $S(f,g)$ computed by the modified RankRev algorithm 2.4.1 for the inexact polynomials f and g is 2 within the threshold $\theta = \varepsilon_{mach}\|S(f,g)\|$, see Figure 5.2a. Hence, the degree of the AGCD would be equal to 2.

Let us now apply the methods STLN, SNTLN, LNSTLS and LNSNTLS to the inexact polynomials f and g. Figure 5.2 then shows the singular values of $S(\hat{f},\hat{g})$, $S(f,g)$ and the singular values of $S(f+\delta f, g+\delta g)$ after each iteration in the STLN method until the stopping criterion is satisfied.

We can see that the only two singular values of $S(f,g)$ are smaller than θ for the inexact polynomials. Hence the direct application of the Gauss-Newton would give the AGCD of degree 2. Applying the STLN method that gives polynomials $f+\delta f$ and $g+\delta g$ which Sylvester matrix has perfectly defined numerical rank equal to 7. Hence, the Gauss-Newton gives the AGCD of degree 7.

Other methods return similar figures, hence they are omitted. Note however, that the non-linear methods SNTLN and LNSNTLS computes $\mathbf{z}$ with smaller norm and the computed AGCD by the Gauss-Newton method is more accurate.

	$\|\mathbf{z}\|$	$\|\hat{\mathbf{d}}-\mathbf{d}\|$
STLN	1.99×10^{-6}	1.20
LNSTLS	1.98×10^{-6}	1.83
SNTLN	1.24×10^{-9}	0.56
LNSNTLS	1.23×10^{-9}	0.07

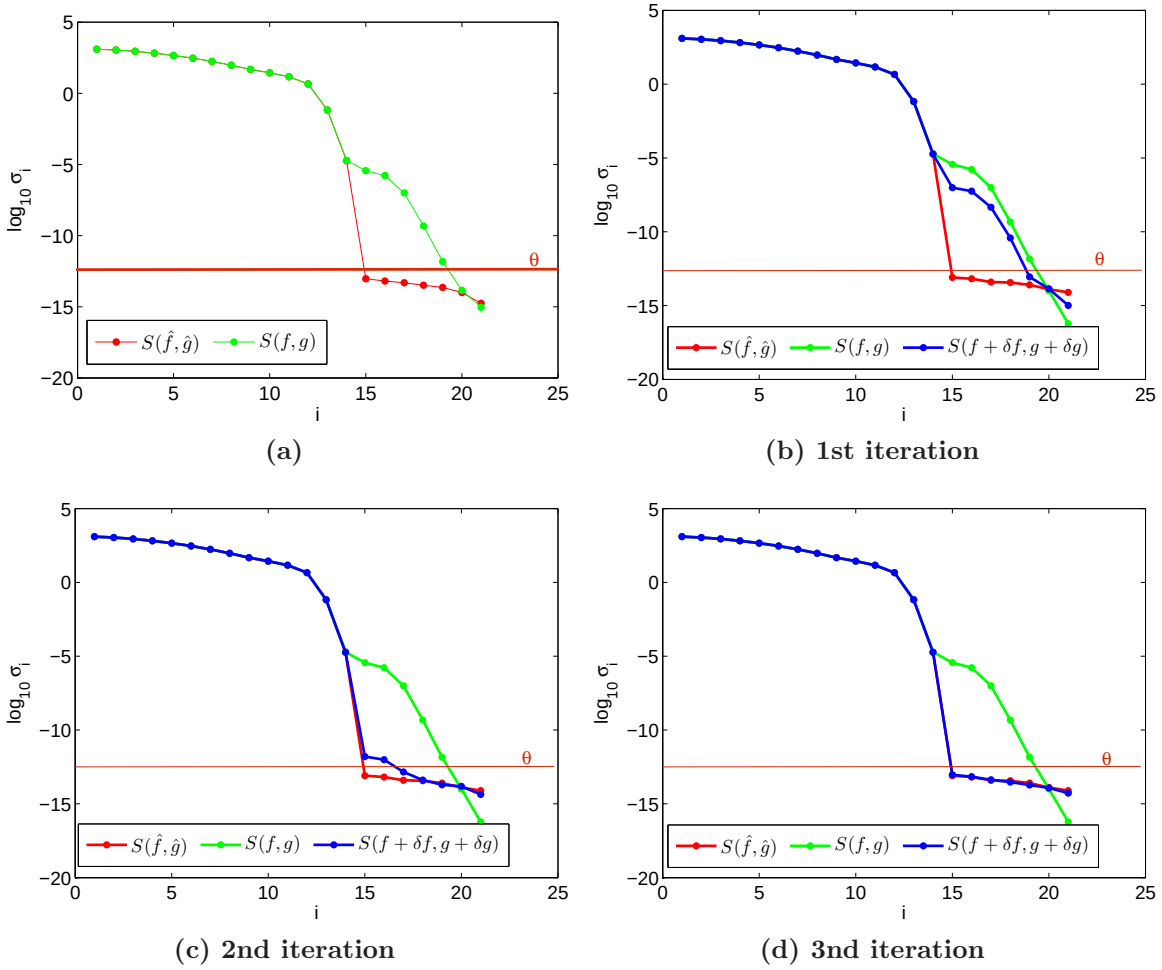
where $\hat{\mathbf{d}}$ is the coefficient vector of the GCD of $\hat{f}$ and $\hat{g}$, and $\mathbf{d}$ is the coefficient vector of the $d = \mathrm{AGCD}(f,g)$. Norms $\|\hat{\mathbf{d}}-\mathbf{d}\|$ are relatively high, but they do not contradict the theory or decrease the efficiency of the methods. We have to realise that $f+\delta f$ and $g+\delta g$ are different polynomials from $\hat{f}$ and $\hat{g}$. Although we get

$$\|\delta\mathbf{f}\| = 1.96\times10^{-6} \quad \text{and} \quad \|\delta\mathbf{g}\| = 3.76\times10^{-7}$$

in the STLN method what could satisfy us, the imposed componentwise perturbation causes quite big differences in the coefficients of the inexact polynomials,

$$\|\hat{\mathbf{f}} - \mathbf{f}\| = 0.27 \quad \text{and} \quad \|\hat{\mathbf{g}} - \mathbf{g}\| = 0.05.$$

Note that in this simple example, the column $\mathbf{c}_p^*$ minimising the residuals (5.4) is chosen to be the 4th column. The number of iterations is 3. Computing with the first column needs 4 iterations. Similar situation appears in other methods. ♣



(a) (b) 1st iteration

(c) 2nd iteration (d) 3nd iteration

Figure 5.2: The singular values of $S(\hat{f}, \hat{g})$ where $\hat{f}$ and $\hat{g}$ are the polynomials in (5.33) normalised by the geometric mean, •, the singular values of $S(f, g)$ where f and g are perturbed polynomials by the signal-to-noise ratio $\epsilon = 10^{-8}$ and normalised by the geometric mean, •, and the singular values of $S(f + \delta f, g + \delta g)$ where δf and δg are computed by the STLN method, •. (b) the singular values of $S(f + \delta f, g + \delta g)$ after the first iteration in STLN, (c) the singular values of $S(f + \delta f, g + \delta g)$ after the second iteration in STLN, (d) the singular values of $S(f + \delta f, g + \delta g)$ after the last third iteration in STLN.

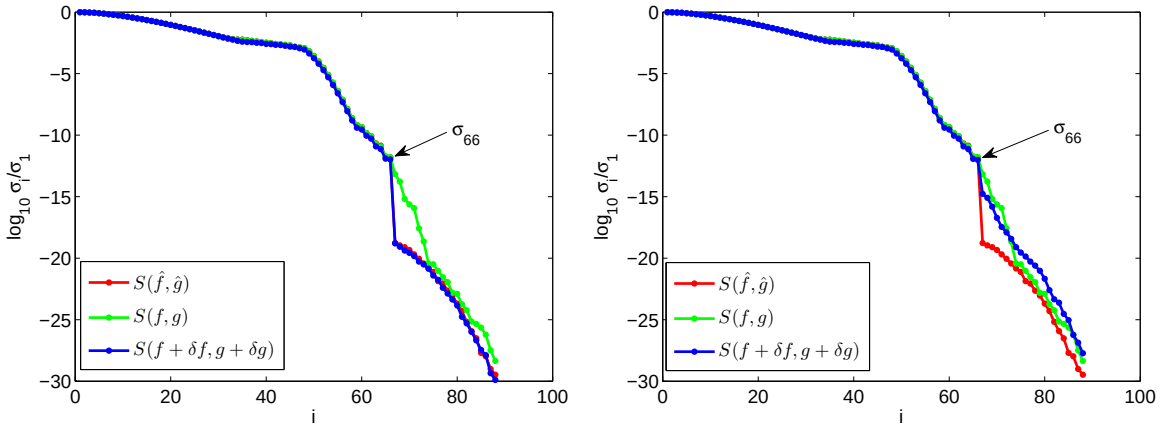
Example 5.5.2. Consider the polynomials $\hat{f}$ and $\hat{g}$ defined in (1.20) for $m = 22$, i.e.

$$\begin{aligned} \hat{f}(x) &= \prod_{i=1}^{11} \left[(x - r_1\alpha_i)^2 + r_1^2\beta_i^2\right] \prod_{i=12}^{22} \left[(x - r_2\alpha_i)^2 + r_2^2\beta_i^2\right], \\ \hat{g}(x) &= \prod_{i=1}^{22} \left[(x - r_1\alpha_i)^2 + r_1^2\beta_i^2\right]. \end{aligned} \tag{5.34}$$

These polynomials are of degrees 44 with the GCD of degree 22. Hence, the rank of $S(\hat{f}, \hat{g}) = 66$. Inexact polynomials f and g are obtained from (5.34) by the componentwise perturbation with the signal-to-noise ratio $\epsilon = 10^{-8}$.

The RankRev algorithm applied to $S(f, g)$ or $S(\bar{f}, \alpha_{opt}\bar{g})$ gives inaccurate numerical ranks 64 and 80 within the standard threshold $\theta = \varepsilon_{mach}\|S\|$. Moreover, employing the numerical-rank gap criterion also gives incorrect ranks 73 and 84. Polynomials $\bar{f}$, $\bar{g}$ and constant α_{opt} arise from the preprocessing operations applied to f and g.

Applied non-linear methods to $S(\bar{f}, \alpha_{opt}\bar{g})$ give perturbations δf and δg of the inexact polynomials f and g such that the numerical rank of $S(f + \delta f, g + \delta g)$ is equal to 66. The RankRev algorithm and the numerical-rank gap criterion can be used here successfully. Figure 5.3 shows the singular values of $S(\hat{f}, \hat{g})$, $S(f, g)$ and $S(f + \delta f, g + \delta g)$. We can see that LNSNTLS gives the correct numerical rank, but the maximal gap is in this case smaller than in the singular values of $S(f + \delta f, g + \delta g)$ computed by SNTLN.



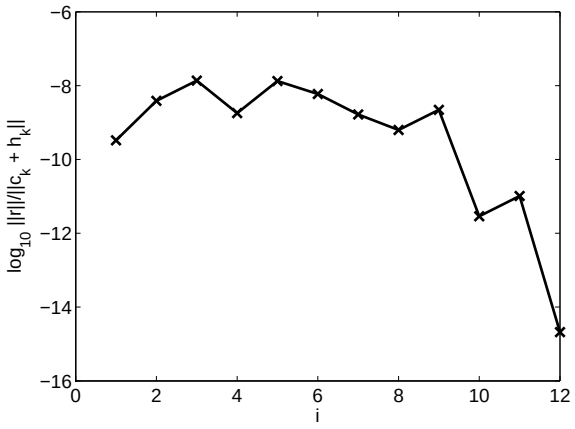
(a) SNTLN **(b) LNSNTLS**

Figure 5.3: The singular values of $S(\hat{f}, \hat{g})$ where $\hat{f}$ and $\hat{g}$ are the polynomials in (5.34) normalised by the geometric mean, •, the singular values of $S(f, g)$ where f and g are perturbed polynomials by the signal-to-noise ratio $\epsilon = 10^{-8}$ and normalised by the geometric mean, •, and the singular values of $S(f + \delta f, g + \delta g)$ where δf and δg are computed by (a) the SNTLN method, (b) the LNSNTLS method, •.

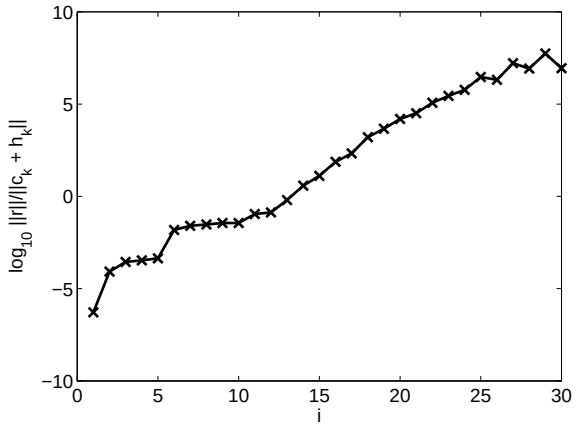
All the computations are carried out on the 9th column of $S(\bar{f}, \alpha_{opt}\bar{g})$. If the first column is considered, then LNSNTLS gives an inaccurate result. The SNTLN method

does not converge at all. It can be seen from Figure 5.4a and 5.4b. Figure 5.4c then shows the normalised residuals for the 9th column in the LNSNTLS method. We can see that the residual in LNSNTLS does not converge with the demanded accuracy 10^{-12}. A possible reason for this may be in solving (5.30) with the standard Matlab functions. Since the 300 by 300 Hessian matrix H has a specific structure, some other techniques may be more efficient (e.g. the modified LDL^T factorisation of H in [16]).

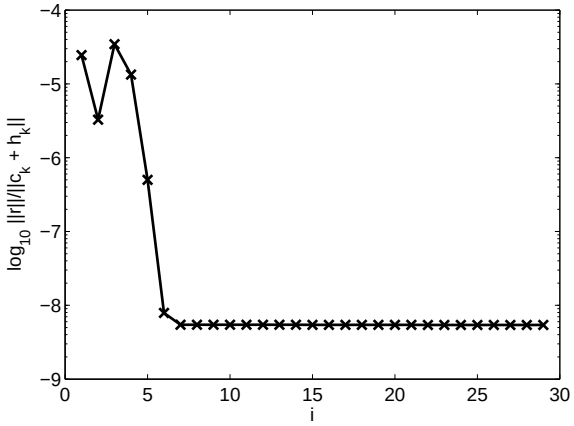
The RankRev algorithm can reveal to correct rank when it is applied to $S(f + \delta f, g + \delta g)$. ♣



(a) 9th column in SNTLN



(b) 1st column in SNTLN



(c) 9th column in LNSNTLS

Figure 5.4: The normalised residuals from the SNTLN method for (a) the 9th column of $S(\bar{f}, \alpha_{opt}\bar{g})$, (b) for the 1st column of $S(\bar{f}, \alpha_{opt}\bar{g})$; the normalised residuals from LNSNTLS for the 9th column of $S(\bar{f}, \alpha_{opt}\bar{g})$.

Many other examples have been tested. For some of them LNSTLS/LNSNTLS woks better, for other examples worse than STLN/SNTLN. So, we do not claim which method is better and should be therefore used.

Summary

This chapter has introduced two extensions of known methods for solving STLS problems customised for the AGCD computation.

While the first method STLN and its extension SNTLN are used for the AGCD computation in [13, 24, 25], the second presented method is new. This method employs the Newton method to the Lagrangian function (4.25) and it is referred to as the LNSTLS method. We have also presented the non-linear extension of LNSTLS method denoted by LNSNTLS in this chapter. Non-linearity to LNSTLS is imposed by the preprocessing operations from Section 1.3. The Newton method is however again applied to the Lagrangian function given in (5.26).

All the methods have been implemented and tested on examples.

Summary and Follow-up work

The present work focuses on computation of an approximate GCD of two univariate polynomials in presence of a noise.

One possible direction in computing can be led through the computation of the numerical rank of Sylvester matrices. Once two polynomials are given, we can form the corresponding Sylvester matrix and try to find its numerical rank by inspection of the singular values. It is however shown that this approach has to be used very carefully due to the sensitivity of singular values on a noise. The SVD or the RankRev algorithm from Chapter 2 can be used.

The second approach avoids direct computing singular values. Since it assumes that some noise is imposed to the polynomial coefficients, it formulates and solves the structured TLS problem (4.13). In other words it subsequently makes small changes in the coefficients of inexact polynomials until they have a non-constant GCD of the highest possible degree. This approach is discussed in Chapters 4 and 5. As a result two methods for the AGCD computation and their extensions are proposed. We think that the second of them has not been published.

In the thesis we work with polynomials in power basis that caused some limitations. Methods presented here can be successfully used for polynomials of small and medium degrees (≈ 50). Methods should be therefore investigated in other basis (Chebyshev and other).

Some other open tasks also exist: efficient implementation of the algorithm, efficient computation of an AGCD of multivariate polynomials in presence of a noise, efficient computation of an AGCD of many (univariate, multivariate) polynomials are just few of them.

Bibliography

[1] Barnett, S. *Polynomials and Linear Control Systems*. Marcel Dekker, New York, 1983. ISBN 0-8247-1898-4.

[2] Bini, D. A. and Boito, P. *Structured matrix based methods for polynomial ϵ-GCD: analysis and comparisons*. Proc. of the 2007 International Symposium on Symbolic and Algebraic Computation (Waterloo, ON), ACM Press, 2007, pp. 9–16.

[3] Björk, *Å*. *Numerical Method for Least Square Problems*. SIAM, Philadelphia, 1996. ISBN 0-89871-360-9.

[4] Boito, P. *Structured Matrix Based Methods for Approximate Polynomial GCD*. Ph.D. Thesis, 2007,
`http://www.mathcs.emory.edu/~boito/thesis.pdf`.

[5] Eliaš, J. *Problémy spojené s výpočtem největšího společného dělitele*. Bachelor thesis (in Slovak language), 2009.

[6] Corless, R. M., Gianni, P. M., Trager, B. M. and Watt, S. M. *The singular value decomposition for polynomial systems*. Proc. Int. Symp. Symbolic and Algebraic Computation, ACM Press, New York, 1995, pp. 195–207.

[7] Corless, R. M., Watt, S. M. and Zhi, L. *QR factoring to compute the GCD of univariate approximate polynomials*. IEEE Trans. Signal Processing 52, 2004, no. 12, pp. 3394–3402.

[8] Fletcher, R. *Practical Methods of Optimization, Vol. 2: Constrained Optimization*. John Wiley & Sons, Chichester, 1981. ISBN 0-471-27828-9.

[9] Golub, G. H., Klema, V. and Stewart, G.W. *Rank degeneracy and least squares problems*. Technical Report TR 456, University of Maryland, 1976.

[10] Golub, G. H. and Van Loan C. F. *An analysis of the total least squares problem*. SIAM J. Numer. Anal., 1980, no. 17, pp. 883–893.

[11] Golub, G. H. and Van Loan C. F. *Matrix Computations*. 3rd Ed. The John Hopkins University Press, Baltimore, 1996. ISBN 0-8018-5414-8.

[12] Hribernig, V. and Stetter, H. J. *Detection and validation of clusters of polynomial zeros.* Journal of Symbolic Computation, 2007, no. 24, pp. 667–681.

[13] Kaltofen, E., Yang, Z. and Zhi, L. *Structured low rank approximation of a Sylvester matrix.* Preprint, 2005.

[14] Karmarkar, N. and Lakshman, Y. N. *Approximate polynomial greatest common divisors and nearest singular polynomials.* ISSAC '96: Proceedings of the 1996 international symposium on Symbolic and algebraic computation, ACM, New York, 1996, pp. 35–39.

[15] Knuth, D. E. *The Art of Computer Programming.* 2nd Ed. Addison-Wesley, Reading, 1969.

[16] Lemmerling, P., Mastronardi, N. and Van Huffel, S. *Fast algorithm for solving the Hankel/Toeplitz Structured Total Least Squares Problem.* Numerical algorithms vol. 23, no. 4 pp. 371–392, 2000.

[17] Li, T. Y. and Zeng, Z. *A rank-revealing method with updating, downdating and applications.* SIAM J. Matrix Anal. Appl. 26, No. 4, 2005, pp. 918–946.

[18] Lurie, B. J. and Enright P. J. *Classical Feedback Control: With MATLAB.* Marcel Dekker, Inc., New York, Basel, 2000. ISBN 0-8247-0370-7.

[19] Noda, M. T. and Sasaki, T. *Approximate GCD and its application to ill-conditioned linear algebraic equations.* Journal of Computational and Applied Mathematics, 1991, no. 38, pp. 335–351.

[20] Pillai, S. U. and Liang, B. *Blind image deconvolution using a robust GCD approach.* IEEE Transactions on Image Processing, no. 8(2), pp. 295–301, 1999.

[21] Rosen, J. B., Park, H. and Glick, J. *Total least norm formulation and solution for structured problems.* SIAM Journal on Matrix Anal. Appl., no. 17(1), pp. 110–128, 1996.

[22] Stetter, H. J. *Numerical Polynomial Algebra.* SIAM: Society for Industrial and Applied Mathematics, 2004. ISBN 0-89871-557-1.

[23] Van Huffel, S. and Lemmerling, P. *Total Least Squares and Errors-in-variables Modeling: Analysis, Algorithms and Applications.* Kluwer Academic Publishers, Dordrecht, 2002. ISBN 1-4020-0476-1.

[24] Winkler, J. R. and Allan, J. D. *Structured total least norm and approximate GCDs of inexact polynomials.* Journal of Comp.and Appl. Math. 215, 2006, pp. 1–13.

[25] Winkler, J. R. and Hasan, M. *A non-linear structure preserving matrix method for the low rank approximation of the Sylvester resultant matrix.* Preprint, 2009.

[26] Winkler, J. R., Hasan, M. and Lao, X. Y. *Two methods for the calculation of the degree of an approximate greatest common divisor of two inexact polynomials.* Springer, 2012,
http://www.springerlink.com/content/a5853g1054114m0w/fulltext.pdf

[27] Winkler, J. R. and Lao, X. Y. *The calculation of the degree of an approximate greatest common divisor of two polynomials.* Preprint, 2009.

[28] Winkler, J. R. and Zítko, J. *Some questions associated with the calculation of the GCD of two univariate polynomials.* Winter School SNA'07, Ostrava, 2007, pp. 130–137.

[29] Zarowski, C. J., Ma, X. and Fairman, F. W. *QR-factorization method for computing the greatest common divisor of polynomials with inexact coefficients.* IEEE Trans. Signal Processing 48, 2000, no. 11, pp. 3042–3051.

[30] Zeng, Z. *The approximate GCD of inexact polynomials, Part I: univariate algorithm.* Preprint, 2004.

[31] Zítko, J. and Eliaš, J. *Application of the rank revealing algorithm for the calculation of the GCD.* Winter School SNA'12, Technická Univerzita v Liberci, Liberec, 2012, pp. 175–180.