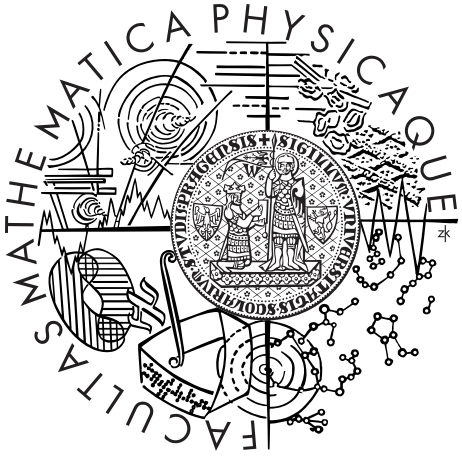
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE



Bc. Jiří Eckstein

Transformace Sylvestrovy matice a výpočet největšího společného dělitele dvou polynomů

Katedra numerické matematiky

Vedoucí diplomové práce: doc. RNDr. Jan Zítko, CSc.

Studijní program: Matematika

Studijní obor: Numerická a výpočtová matematika

Praha 2014

I would like to thank my supervisor, doc. Zítko, for his immense dedication in helping me and advising my work on this thesis, as well as Dr. Winkler for the most valuable consultations.

But foremost, I wish to thank my family for all their support, without which writing this thesis would not be possible.

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V Praze dne 30.07.2014 Bc. Jiří Eckstein

Název práce: Transformace Sylvestrovy matice a výpočet největšího společného dělitele dvou polynomů

Autor: Bc. Jiří Eckstein

Katedra: Katedra numerické matematiky

Vedoucí diplomové práce: doc. RNDr. Jan Zítko, CSc., Katedra numerické matematiky

Abstrakt: V této diplomové práci se zabýváme výpočtem největšího společného dělitele dvou polynomů. V první řadě studujeme vlastnosti Sylvestrových matic a jakým způsobem je lze využít pro daný záměr. Dále si všimneme, že výsledky lze přirozeně zobecnit i pro více polynomů. V předposlední části se zabýváme využitím Bézoutových matic ke stejnému účelu, abychom získali srovnání s maticemi Sylvestrovými. I zde výsledek rozšíříme pro víc než dva polynomy. Ke všem přístupům jsou prezentovány algoritmy. Na závěr algoritmy implementujeme v prostředí MATLAB a jednotlivé algoritmy porovnáme v numerických experimentech.

Klíčová slova: Největší společný dělitel, Sylvestrova matice, Bézoutova matice, určení hodnosti

Title: The transformation of the Sylvester matrix and the calculation of the GCD of two polynomials

Author: Bc. Jiří Eckstein

Department: Department of Numerical Mathematics

Supervisor: doc. RNDr. Jan Zítko, CSc., Department of Numerical Mathematics

Abstract: In this thesis we study the computation of the greatest common divisor of two polynomials. Firstly, properties of Sylvester matrices are considered as well as their role in computation. We then note, that this approach can be naturally generalized for several polynomials. In the penultimate section, Bézout matrices are studied as an analogy to the Sylvester ones, providing necessary comparison. Extension for more than polynomials is presented here as well. Algorithms corresponding to the individual approaches are presented as well. Finally, the algorithms are implemented in MATLAB and are compared in numerical experiments.

Keywords: Greatest common divisor, Sylvester matrix, Bézout matrix, rank determination

Contents

Introduction

This thesis aims to concern itself with the task of numerically computing the polynomial greatest common divisor (GCD). At first glance, this does not seem to be problematic, since for any input data, symbolical solution can be found with the use of the well known Euclid's algorithm. First trouble will start to appear as soon as we enter the floating point environment, since even the slightest change to the coefficients of the given polynomials can result in a dramatic change of their GCD. We say that finding GCD is an *ill-posed problem* (i.e. arbitrarily small perturbation of input can lead to a completely different answer).

Since this renders the original algorithm quite unstable, other ways of completing such a task are sought for. One of the approaches is to use structured matrices instead of polynomials, providing a more robust platform to carry out the computation. Of course, such matrices need to carry all the properties of the polynomials and are often related to the algebraic notion of a polynomial *resultant*. We speak of so-called *resultant matrices*.

In this text, firstly, we go over some necessary notions in Chapter 1, of course including the definition of the Greatest common divisor. Another important concept is that of numerical rank of a matrix. We also need a way of computing it, hence we present an algorithm used for rank determination. Correctly, or perhaps suitably, determining a rank of a given matrix is quite a difficult, but essential task in the process of finding the GCD as becomes clear later.

In Chapter 2, we study the Sylvester matrices, the first of the two kinds of resultant matrices we consider, and how they can be used for our goal. It is presented how the rank of Sylvester matrix is connected to the degree of the GCD of the corresponding polynomials. We then show how elimination can be used as a direct analogy of the Euclid's algorithm, but a more sophisticated algorithm is considered as well. We also find out, that the problem of finding the GCD of two polynomials can be quite naturally extended to include several of them, but what is more important, this new approach is applicable to the original problem for two polynomials as well. It turns out, that the notion of AGCD: *approximate greatest common divisor*, is needed in addition to the GCD.

Next, in Chapter 3, to provide some sort of comparison, a different kind of resultant matrix, the Bézout matrix, is considered. It gives us an example of a different kind of resultant matrix and enables us to have some context for the qualities of Sylvester resultants. We explore a great article [14] by V. Pták concerning Bézout matrices and then follow up with results of [2] to present a slight variation of their algorithm. As in the previous chapter, generalization for more than two polynomials is explored here as well.

Finally, in Chapter 4, we implement algorithms presented in previous chapters, compute GCD of some sample problems, and compare the presented methods and approaches.

1. Elementary Notions

We shall go through the notation used throughout this paper and define some of the elementary notions, with which we will work. Afterwards we concern ourself with a numerical rank of a matrix and a method of revealing it. A simple algorithm for this purpose is also presented.

1.1 Notation and Definitions

Let us first go through basic notation used in this paper.

Notation	Meaning
$n,\ k,\ a_i,\ \sigma,\ \tau$	Scalars are denoted by lower case letters.
$\boldsymbol{x}_0,\ \boldsymbol{g},\ \boldsymbol{\phi}$	Vectors are denoted by bold lower case letters. Unless explicitly stated, we assume column vectors.
$A,\ M,\ V,\ \Sigma$	Matrices are denoted by capital letters.
$\boldsymbol{W},\ \boldsymbol{P}$	Vector spaces are denoted by bold capital letters.
$f(x),\ g(y),\ f,\ g$	Polynomials are denoted by lower case letters. Variable can be omitted where it is unnecessary.
$\mathrm{Ker}(A)$	Kernel, or null space of a matrix, i.e. $\mathrm{Ker}(A) = \{\boldsymbol{x};\ A\boldsymbol{x} = \boldsymbol{0}\}$.
$\mathrm{span}\{\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_n\}$	Linear span of vectors $\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_n$, i.e. $\mathrm{span}\{\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_n\} =$ $= \{\boldsymbol{x};\ \exists\, a_1, \ldots a_n \in \mathbb{R} :\ \boldsymbol{x} = a_1\boldsymbol{\phi}_1 + \cdots + a_n\boldsymbol{\phi}_n\}$.
$\mathrm{rank}(A)$	Column rank of a matrix A, i.e. number of its linearly independent columns.
$\deg(f),\ \deg(f(x))$	The degree of the polynomial f (or $f(x)$), which is its highest power with a non-zero coefficient.
$\|\cdot\| = \|\cdot\|_2$	Euclidean norm of vector $\|\boldsymbol{x}\| = (\sum_{i=1}^{n} x_i^2)^{\frac{1}{2}}$ for $\boldsymbol{x} = [x_1, \ldots, x_n] \in \mathbb{R}^n$, or spectral norm $\|A\| = \sup_{\|\boldsymbol{x}\|=1} \|A\boldsymbol{x}\|$ for matrices $A \in \mathbb{R}^{m\times n}$.
$\boldsymbol{e}_i,\ I$	The ith canonical basis vector and identity matrix. For example, $I = [\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n]$ in $\mathbb{R}^n$.
$\mathrm{diag}(\alpha_1, \ldots, \alpha_n)$	An $n \times n$ diagonal matrix with entries $\alpha_1, \ldots, \alpha_n$.

We will obviously talk about the notion of the greatest common divisor of two polynomials.

Definition 1.1. Let us have two polynomials $f(x) = \sum_{i=0}^{m} a_i x^{m-i}$ and $g(x) = \sum_{i=0}^{n} b_i x^{n-i}$ of degrees $m,\ n$ respectively, $a_i,\ b_i \in \mathbb{R}$.

The polynomial $g(x)$ *divides* $f(x)$, if there exists a polynomial $p(x)$, such that $f(x) = p(x)g(x)$. We denote this by $g \mid f$, or $g \nmid f$ if no such $p(x)$ exists.

Polynomials $f(x)$ and $g(x)$ have a *common divisor* if there exists such a polynomial $c(x)$, that $c(x) \mid f(x)$ as well as $c(x) \mid g(x)$.

Two polynomials are *coprime* or *relatively prime* if for all their common divisors $c(x)$ it holds, that $\deg(c(x)) = 0$, i.e. it is a non-zero (real) constant.

Any polynomial $h(x)$ satisfying $f(x) = h(x)v(x),\ g(x) = h(x)w(x)$, with

$v(x)$, $w(x)$ coprime is called *the greatest common divisor* of $f(x)$ and $g(x)$, denoted by $h(x) = \mathrm{GCD}(f(x), g(x))$.

For computation and in algorithms, polynomials will be often represented as vectors, so for polynomial $f = f(x) = \sum_{i=0}^{m} a_i x^{m-i}$ we have a corresponding vector $\boldsymbol{f} = [a_0, a_1, \ldots, a_m]^T$.

The best known way of finding the GCD of two polynomials is the Euclid's algorithm. We work with a sequence of polynomials $f^{(i)}$, $i \in \mathbb{N} \cup \{0\}$, denote $\deg(f^{(i)}) = n_i \in \mathbb{N}$ and their coefficients as

$$f^{(i)}(x) = a_0^{(i)} x^{n_i} + a_1^{(i)} x^{n_i - 1} + \cdots + a_{n_i}^{(i)}.$$

With this notation, the Euclid's algorithm (see [20]) is as follows.

Algorithm 1.2. *Let us have two polynomials $f(x)$ and $g(x)$ such that $\deg(f) \geq \deg(g)$. We aim to compute $h(x) = \mathrm{GCD}(f, g)$.*

I. Set $f^{(0)} := f$ and $f^{(1)} := g$.

II. For $i = 0, 1, 2, \ldots$ do following:

(i) If $f^{(i+1)} = 0$, then continue with the step III.

(ii) Divide $f^{(i)}$ by $f^{(i+1)}$ so that $f^{(i)}(x) = q(x) f^{(i+1)}(x) + r(x)$, for some polynomials q and r, $\deg(r) < n_{i+1}$.

(iii) Set $f^{(i+2)}(x) := r(x)$.

(iv) Update $i := i + 1$ and repeat from the step (i).

III. We set $h(x) = f^{(i)}(x)$

End of algorithm.

As is presented later, degree of the GCD of two polynomials is tightly related to a rank of a specific matrix. Since even a slight perturbation of a matrix entries completely changes its mathematical rank, we must, in the context of numerical mathematics, concern ourselves with replacing such a notion.

To illustrate, simply consider matrices

$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 1 & 1+\varepsilon \\ 1 & 1 \end{bmatrix}.$$

Rank of the first one is obviously 1, but for any arbitrarily small real $\varepsilon > 0$, is the rank of the second one 2.

We find it beneficial to provide simple examples, illustrations and notes throughout this paper with a goal of improving understanding and overall clarity. These remarks will be graphically distinguished in this fashion, just as this one is, from the rest of the text.

Instead, we consider the distance of any given matrix A from the nearest matrices of deficient rank. For a certain given tolerance θ we consider the lowest rank of all the matrices that are sufficiently close to A to be its numerical rank. Specifically, in [8] the following is found.

Definition 1.3. Let us have an $m \times n$ real matrix A, $m \geq n$, and $\theta \in \mathbb{R}$ satisfying

$$k = \inf\{\text{rank}(B);\ B \in \mathbb{R}^{m\times n},\ \|A - B\| \leq \theta\}$$

Then k is the *numerical rank of A with respect to θ* and we write

$$\text{rank}_\theta(A) = k.$$

Note that if A has singular values satisfying

$$\sigma_1 \geq \cdots \geq \sigma_k > \theta \geq \sigma_{k+1} \cdots \geq \sigma_n.$$

then $\text{rank}_\theta A = k$ (see [8], Theorem 2.5.3 or [12]).

Let us now illustrate the whole situation on a simple example. Consider a diagonal matrix $A = \text{diag}(\lambda_1, \ldots, \lambda_n)$, where $\lambda_1 > \ldots > \lambda_n > 0$. Clearly, $\text{rank}(A) = \text{rank}_0(A) = n$. The nearest rank deficient (or in this case singular) matrix with respect to the spectral norm is $Z = \text{diag}(\lambda_1, \ldots, \lambda_{n-1}, 0)$ and its rank is $\text{rank}(Z) = n - 1$. We see, that $\|A - Z\| = \lambda_n$. Keeping that in mind, if we consider $\lambda_n > \theta > 0$, then $\text{rank}_\theta(A) = n$, but for $\lambda_{n-1} > \theta \geq \lambda_n$ that changes to $\text{rank}_\theta(A) = n - 1$ and A is rank deficient with respect to tolerance θ.

1.2 GCD and AGCD

In finite arithmetic, coefficients of given polynomials might not be known exactly due to rounding errors. In this setting, trying to find such a GCD would be an ill-posed problem, since even a miniscule perturbation of polynomial coefficients can change the degree of GCD drastically.

This means we shall usually consider only an *approximate greatest common divisor* – AGCD. With a set tolerance $\varepsilon > 0$ we allow our result to be the GCD of some perturbed (within ε with respect to some norm) polynomials instead of the original pair (see [17]). There are different definitions to be found in literature (e.g. in [19], [5]), though the general idea is shared.

1.3 Determining the Rank of a Matrix

In this section, we will shortly outline a Gauss-Newton iteration used for minimizing functionals and then present a theoretical background as well as algorithm using it to determine the numerical rank of a matrix.

1.3.1 Gauss-Newton method

Let us have a vector function $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^m$, $\boldsymbol{f} = [f_1, \ldots, f_m]^T$ and functional $F(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{f}^T(x)\,\boldsymbol{f}(x) = \frac{1}{2}[f_1, \ldots, f_m][f_1, \ldots, f_m]^T$. We shall consider ourselves with the problem of minimizing F (if only locally). For x to be a point of local extremum, it is necessary that $g(x) = \mathbf{0}$, where

$$g(x) = \left[\frac{\partial F}{\partial x_1}(x), \ldots, \frac{\partial F}{\partial x_n}(x)\right]^T.$$

To construct an iterative method, let us have $x^{(i)} \in \mathbb{R}^n$. We denote $\mathcal{G}$ the Hessian matrix of second partial derivatives of F and expand F about $x^{(i)}$

$$F(x) = F\left(x^{(i)}\right) + \left(x - x^{(i)}\right)^T g\left(x^{(i)}\right) + \tfrac{1}{2}\left(x - x^{(i)}\right)^T \mathcal{G}\left(x^{(i)}\right)\left(x - x^{(i)}\right) + \dots$$

and neglect all but the first three terms. We then have

$$g(x) = g\left(x^{(i)}\right) + \mathcal{G}\left(x^{(i)}\right)\left(x - x^{(i)}\right),$$

assuming $\mathcal{G}$ symmetric. Since $g(x) = 0$, we have

$$0 = g\left(x^{(i)}\right) + \mathcal{G}\left(x^{(i)}\right)\left(x - x^{(i)}\right).$$

We then set $x^{(i+1)} := x$, specifically

$$x^{(i+1)} = x^{(i)} - \mathcal{G}^{-1}\left(x^{(i)}\right) g\left(x^{(i)}\right).$$

Approximating $\mathcal{G}\left(x^{(i)}\right)$ by $J^T\left(x^{(i)}\right) J\left(x^{(i)}\right)$ and replacing $g\left(x^{(i)}\right)$ by $J^T\left(x^{(i)}\right) f\left(x^{(i)}\right)$, where J is the Jacobian matrix of f, we finally get Gauss-Newton's iteration

$$x^{(i+1)} = x^{(i)} - \underbrace{\left(J^T\left(x^{(i)}\right) J\left(x^{(i)}\right)\right)^{-1} J^T\left(x^{(i)}\right)}_{J^\dagger} f\left(x^{(i)}\right), \ i \in \mathbb{N} \cup \{0\},$$

where $J^\dagger$ is the Moore-Penrose pseudoinverse of J. This choice is a result of the following theorem found in [3].

Theorem 1.4. *Let us have $F, \boldsymbol{f}$. Hessian matrix $\mathcal{G}(x)$ of the functional $F(x)$ can be written in the form*

$$\mathcal{G}(x) = J^T(x) J(x) + \sum_{k=1}^{m} f_k(x) \mathcal{G}_k(x),$$

where $f_k(x)$ is the kth component of $\boldsymbol{f}(x) = [f_1(x), \dots, f_m(x)]$ and $\mathcal{G}_k(x)$ is its Hessian matrix.

For more details see [13], [3].

1.3.2 Rank Revealing Algorithm

In this section, we aim to present an algorithm based on the Gauss-Newton iteration, determining the numerical rank of a matrix. But first, the following theorem shows how can be minimization of a functional used for that purpose.

Let $A \in \mathbb{R}^{m \times n}, m \geq n$ be a matrix with singular values

$$\sigma_1 \geq \dots \geq \sigma_n$$

and corresponding right singular vectors $\boldsymbol{v}_1, \dots, \boldsymbol{v}_n$. There exists $1 \leq k \leq n$, such that $\sigma_{k+1} = \sigma_{k+2} = \dots = \sigma_n$. Let us define $\boldsymbol{W} := \operatorname{span}\{\boldsymbol{v}_{k+1}, \dots, \boldsymbol{v}_n\}$. The following theorem found in [12] shows a relationship between finding the smallest singular number σ_n of the matrix A and solving a certain least squares problem.

Theorem 1.5. *Let us have A with singular values $\sigma_1, \ldots, \sigma_n$ as above, $\tau \in \mathbb{R}$, such that $\tau > \sigma_1$ and $\boldsymbol{u} \in \mathbb{R}^n$ satisfying*

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

Then $\boldsymbol{u} \in \boldsymbol{W} = \operatorname{span}\{\boldsymbol{v}_{k+1}, \ldots, \boldsymbol{v}_n\}$.

Proof. Firstly, we have

$$\left\| \begin{bmatrix} \tau \boldsymbol{u}^T \\ A \end{bmatrix} \boldsymbol{u} - \begin{bmatrix} \tau \\ 0 \end{bmatrix} \right\|^2 = \left\| \begin{bmatrix} \tau \boldsymbol{u}^T \boldsymbol{u} - \tau \\ A\boldsymbol{u} \end{bmatrix} \right\|^2 = \tau^2 (\boldsymbol{u}^T \boldsymbol{u} - 1)^2 + \|A\boldsymbol{u}\|^2. \tag{1.1}$$

Le us now have a singular value decomposition of A, so that $A = U\Sigma V^T$, and set $\boldsymbol{t} := V^T \boldsymbol{u}$. Since V is an orthogonal matrix, we have $VV^T = I$ and we get $\boldsymbol{u} = V\boldsymbol{t}$. Using this as a substitution in (1.1), we obtain

$$\begin{aligned} \tau^2 (\boldsymbol{u}^T \boldsymbol{u} - 1)^2 + \|A\boldsymbol{u}\|^2 &= \tau^2 (\boldsymbol{t}^T V^T V \boldsymbol{t} - 1)^2 + \|AV\boldsymbol{t}\|^2 \\ &= \tau^2 (\boldsymbol{t}^T \boldsymbol{t} - 1)^2 + \|U\Sigma\boldsymbol{t}\|^2 \\ &= \tau^2 (\boldsymbol{t}^T \boldsymbol{t} - 1)^2 + \|\Sigma\boldsymbol{t}\|^2. \end{aligned}$$

Let now $\sigma_1, \ldots, \sigma_n$ be singular values of A, meaning that $\Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_n)$, and let $\boldsymbol{t} = [t_1, \ldots, t_n]^T$. This gives us

$$\tau^2 (\boldsymbol{t}^T \boldsymbol{t} - 1)^2 + \|\Sigma\boldsymbol{t}\|^2 = \tau^2 \left(\sum_{i=1}^n t_i^2 - 1 \right)^2 + \sum_{i=1}^n \sigma_i^2 t_i^2 =: f(\boldsymbol{t}).$$

Since we assumed that $\boldsymbol{u}$ minimizes the expression in (1.1), it follows from the necessary condition for local minimum, that $\frac{\partial f(\boldsymbol{x})}{\partial x_k}(\boldsymbol{t}) = 0$ for all $k \in \{1, \ldots, n\}$. Hence,

$$0 = \frac{\partial f(\boldsymbol{x})}{\partial x_k}(\boldsymbol{t}) = 2\tau^2 \left(\sum_{i=1}^n t_i^2 - 1 \right) (2t_k) + 2t_k \sigma_k^2.$$

Now, if it holds that $t_k \neq 0$ for some k, we have

$$\sigma_k^2 = 2\tau^2 \left(1 - \sum_{i=1}^n t_i^2 \right), \tag{1.2}$$

which is expression for σ_k independent of k, meaning there is $\sigma \in \mathbb{R}$, such that $\sigma_k = \sigma$ for any $k \in \{i;\ t_i \neq 0\} =: K$. But now

$$\begin{aligned} f(\boldsymbol{t}) &= \tau^2 \left(\sum_{i=1}^n t_i^2 - 1 \right)^2 + \sum_{i \in K} \sigma_i^2 t_i^2 \\ &= \tau^2 \left(\sum_{i=1}^n t_i^2 - 1 \right)^2 + \sum_{i \in K} \sigma^2 t_i^2 \\ &= \tau^2 \left(\sum_{i=1}^n t_i^2 - 1 \right)^2 + \sigma^2 \sum_{i \in K} t_i^2 - \sigma^2 + \sigma^2 \\ &= \tau^2 \left(\sum_{i=1}^n t_i^2 - 1 \right)^2 + \sigma^2 \left(\sum_{i \in K} t_i^2 - 1 \right) + \sigma^2. \end{aligned}$$

At this point, we can plug-in for $\left(1-\sum_{i=1}^{n} t_i^2\right)$ from (1.2), and obtain

$$\begin{aligned} f(\boldsymbol{t}) &= \tau^2\left(-\frac{\sigma^2}{2\tau^2}\right)^2 + \sigma^2\left(-\frac{\sigma^2}{2\tau^2}\right) + \sigma^2. \\ &= \sigma^2 - \frac{\sigma^4}{4\tau^2}. \end{aligned}$$

Therefore, if $\boldsymbol{t}$ is a point of local minimum, then either $f(\boldsymbol{t}) = \sigma^2 - \frac{\sigma^4}{4\tau^2}$, where $\sigma = \sigma_k$ for some $k \in K$, or K is an empty set, $\boldsymbol{t} = \boldsymbol{0}$ and $f(\boldsymbol{t}) = \tau^2$. Which one of these values is the smallest, i.e. the global minimum? Let us study the function $x^2 - \frac{x^4}{4\tau^2}$. We have

$$\frac{d}{dx}\left(x^2 - \frac{x^4}{4\tau^2}\right) = x\left(2 - \frac{x^2}{\tau^2}\right) \geq 0, \qquad 0 \leq x \leq \tau\sqrt{2},$$

meaning that the studied function is non-decreasing on $[0, \tau\sqrt{2}]$. Thanks to the assumption $\sigma_1 < \tau$, it is clear, that $\sigma_i \in [0, \tau\sqrt{2}]$ for $i \in 1, \ldots, n$. And since $\sigma_n < \tau$, the minimal value of $f(\boldsymbol{t})$ is $\sigma_n^2 - \frac{\sigma_n^4}{4\tau^2}$. This value can be achieved for $\boldsymbol{t} = \boldsymbol{e}_k\sqrt{1 - \frac{\sigma_k^2}{2\tau^2}}$ as can be readily checked, for any $k \in K$, since $\sigma_k = \sigma = \sigma_n$, $k \in K$. We now recall, that $\boldsymbol{u} = V\boldsymbol{t}$ and note, that $\boldsymbol{e}_k = V^T\boldsymbol{v}_k$, where $\boldsymbol{v}_k$ is the singular vector associated with σ_k, $k \in K$ (or one of the singular vectors associated with σ). Also, since $t_i = 0$, $i \notin K$, we have

$$\boldsymbol{u} = V\boldsymbol{t} = \sum_{i=1}^{n} t_i\boldsymbol{v}_i = \sum_{k\in K} t_k\boldsymbol{v}_k,$$

where all the vectors $\boldsymbol{v}_k$ are (once again) the singular vectors associated with the smallest singular value, namely σ. Hence, $\boldsymbol{u} \in \boldsymbol{W}$, which concludes the proof. ■

To solve the presented least squares problem, the following Gauss-Newton iteration can be used (see [12]):

$$\boldsymbol{x}_{j+1} = \boldsymbol{x}_j - \begin{bmatrix} 2\tau\boldsymbol{x}_j^T \\ A \end{bmatrix}^{\dagger} \begin{bmatrix} \tau\boldsymbol{x}_j^T\boldsymbol{x}_j - \tau \\ A\boldsymbol{x}_j \end{bmatrix}, \; j \in \mathbb{N} \cup \{0\} \tag{1.3}$$

where the symbol $M^{\dagger}$ denotes the Moore-Penrose pseudoinverse of the matrix M. Moreover let us set

$$\zeta_j = \frac{\|A\boldsymbol{x}_j\|_2}{\|\boldsymbol{x}_j\|_2}.$$

Then the sequence $\{\zeta_j, j = 1, 2, \ldots\}$ converges to the smallest singular number of the matrix A, i.e. σ_n.

The following lemma from [12] presents an alternative form of the aforementioned iteration.

Lemma 1.6. *Let $A \in \mathbb{R}^{m\times n}$ be a matrix of a full column rank, $\tau \in \mathbb{R}$ and let $\{\boldsymbol{x}_j\}_{j=0,1,\ldots}$ be a sequence generated by the iteration process* (1.3)*. We set*

$$c_j := \frac{2\tau^2(1 + \boldsymbol{x}_j^T\boldsymbol{x}_j)}{1 + 4\tau^2\boldsymbol{x}_j^T(A^TA)^{-1}\boldsymbol{x}_j}.$$

Then

$$\boldsymbol{x}_{j+1} = c_j(A^TA)^{-1}\boldsymbol{x}_j.$$

Proof. Firstly,

$$\begin{bmatrix} 2\tau \boldsymbol{x}_j^T \\ A \end{bmatrix}^\dagger \begin{bmatrix} \tau \boldsymbol{x}_j^T \boldsymbol{x}_j - \tau \\ A\boldsymbol{x}_j \end{bmatrix} = \left([2\tau \boldsymbol{x}_j, A^T] \begin{bmatrix} 2\tau \boldsymbol{x}_j^T \\ A \end{bmatrix} \right)^{-1} [2\tau \boldsymbol{x}_j, A^T] \begin{bmatrix} \tau \boldsymbol{x}_j^T \boldsymbol{x}_j - \tau \\ A\boldsymbol{x}_j \end{bmatrix}$$
$$= \left(4\tau^2 \boldsymbol{x}_j \boldsymbol{x}_j{}^T + A^T A\right)^{-1} \left((2\tau^2 \boldsymbol{x}_j \boldsymbol{x}_j{}^T + A^T A)\boldsymbol{x}_j - 2\tau^2 \boldsymbol{x}_j\right).$$

Next,

$$\boldsymbol{x}_j - \left(4\tau^2 \boldsymbol{x}_j \boldsymbol{x}_j{}^T + A^T A\right)^{-1} \left((2\tau^2 \boldsymbol{x}_j \boldsymbol{x}_j{}^T + A^T A)\boldsymbol{x}_j - 2\tau^2 \boldsymbol{x}_j\right)$$
$$= 2\tau^2 (1 + \boldsymbol{x}_j{}^T \boldsymbol{x}_j) \left(4\tau^2 \boldsymbol{x}_j \boldsymbol{x}_j{}^T + A^T A\right)^{-1} \boldsymbol{x}_j$$

and so

$$\begin{aligned} \left(4\tau^2 \boldsymbol{x}_j \boldsymbol{x}_j{}^T + A^T A\right) \boldsymbol{x}_{j+1} &= 2\tau^2 (1 + \boldsymbol{x}_j{}^T \boldsymbol{x}_j)\boldsymbol{x}_j \\ (A^T A)\, \boldsymbol{x}_{j+1} &= 2\tau^2 (1 + \boldsymbol{x}_j{}^T \boldsymbol{x}_j - 2\boldsymbol{x}_j{}^T \boldsymbol{x}_{j+1})\boldsymbol{x}_j \\ \boldsymbol{x}_{j+1} &= 2\tau^2 (1 + \boldsymbol{x}_j{}^T \boldsymbol{x}_j - 2\boldsymbol{x}_j{}^T \boldsymbol{x}_{j+1}) \left(A^T A\right)^{-1} \boldsymbol{x}_j \\ \boldsymbol{x}_{j+1}(1 + 4\tau^2 \boldsymbol{x}_j{}^T (A^T A)^{-1} \boldsymbol{x}_j) &= 2\tau^2 (1 + \boldsymbol{x}_j{}^T \boldsymbol{x}_j) \left(A^T A\right)^{-1} \boldsymbol{x}_j \end{aligned}$$

and the statement follows directly. ■

This shows that the iterative process (1.3) is in fact an matrix power iteration for the matrix $(A^T A)^{-1}$.

Definition 1.7. Let us have $\theta > 0$ and a matrix A. Moreover, let $\operatorname{rank}_\theta(A) = k$ and let $\{\boldsymbol{v}_i\}_{i=1}^n$ be its right singular vectors associated with its singular values $\{\sigma_i\}_{i=1}^n$, $\sigma_1 \geq \cdots \geq \sigma_n$. Linear space $\boldsymbol{W} := \operatorname{span}\{\boldsymbol{v}_{k+1}, \ldots, \boldsymbol{v}_n\}$ is the *approxi-null space* of A with respect to θ.

We would like to find all the smallest singular values of a given matrix. The following two theorems (also found in [12]) gives us directions on how to achieve that.

Theorem 1.8. *Let $A \in \mathbb{R}^{m \times n}$, $m \geq n$, be a matrix satisfying $rank_\theta(A) = k$ for some given $\theta > 0$, let*

$$\sigma_1 \geq \cdots \geq \sigma_k =: \hat{\sigma} > \theta \geq \check{\sigma} := \sigma_{k+1} \geq \ldots \sigma_n$$

and let $\boldsymbol{W}$ be its non-trivial approxi-null space. Then for any unit vector $\boldsymbol{w} \in \boldsymbol{W}$ and $\varrho \geq \hat{\sigma}$, the matrix

$$B = \begin{bmatrix} \varrho \boldsymbol{w}^T \\ A \end{bmatrix}$$

has singular values σ_i' satisfying

$$\sigma_1' \geq \ldots \geq \sigma_{k+1}' \geq \hat{\sigma} > \theta \geq \check{\sigma} \geq \sigma_{k+2}' \geq \ldots \geq \sigma_n'$$

and its approxi-null space $\boldsymbol{W}'$ is a subspace of $\boldsymbol{W}$.

Proof. Since $\boldsymbol{w} \in \boldsymbol{W}$, we have $\boldsymbol{w} = \rho_{k+1}\boldsymbol{v}_{k+1} + \ldots + \rho_n \boldsymbol{v}_n$, $\sum_{i=k+1}^{n} \rho_i^2 = 1$. The singular value decomposition $A = U\Sigma V^T$ then gives us:

$$\begin{bmatrix} 1 & \\ & U^T \end{bmatrix} BV = \begin{bmatrix} 0 & \cdots & 0 & \varrho\rho_{k+1} & \cdots & \varrho\rho_n \\ \sigma_1 & & & & & \\ & \ddots & & & & \\ & & \sigma_k & & & \\ & & & \sigma_{k+1} & & \\ & & & & \ddots & \\ & & & & & \sigma_n \end{bmatrix}.$$

Applying a certain permutation represented by a matrix P, we obtain

$$\begin{bmatrix} 1 & \\ & U^T \end{bmatrix} BV = P \begin{bmatrix} \sigma_1 & & & \\ & \ddots & & \\ & & \sigma_k & \\ & & & D \end{bmatrix},$$

where

$$D = \begin{bmatrix} \varrho\rho_{k+1} & \cdots & \varrho\rho_n \\ \sigma_{k+1} & & \\ & \ddots & \\ & & \sigma_n \end{bmatrix} = \hat{U} \begin{bmatrix} \hat{\sigma}_{k+1} & & \\ & \ddots & \\ & & \hat{\sigma}_n \end{bmatrix} \hat{V}^T,$$

using the singular value decomposition of D to acquire the second equality. Finally, we have

$$B = \begin{bmatrix} 1 & \\ & U \end{bmatrix} P \begin{bmatrix} I_{k\times k} & \\ & \hat{U} \end{bmatrix} \begin{bmatrix} \sigma_1 & & & & & \\ & \ddots & & & & \\ & & \sigma_k & & & \\ & & & \hat{\sigma}_{k+1} & & \\ & & & & \ddots & \\ & & & & & \hat{\sigma}_n \end{bmatrix} \begin{bmatrix} I_{k\times k} & \\ & \hat{V}^T \end{bmatrix} V^T, \quad (1.4)$$

which is a certain singular value decomposition of B, since permutation matrices are, as well as the other relevant matrices, orthogonal; the symbol $I_{k\times k}$ denotes an identity matrix of dimensions $k \times k$

To have a better insight into how the singular values behave, let us present a very simple example alongside this proof. We consider a diagonal 4×4 matrix

$$A = \begin{bmatrix} 4 & & & \\ & 3 & & \\ & & 0.02 & \\ & & & 0.01 \end{bmatrix}.$$

Its diagonal entries are also its singular values. We will use $\theta = 1$ and therefore $\operatorname{rank}_\theta(A) = 2$ and $\boldsymbol{W} = \operatorname{span}\{\boldsymbol{e}_3, \boldsymbol{e}_4\}$. Choosing e. g. $\boldsymbol{w} = \frac{\boldsymbol{e}_3+\boldsymbol{e}_4}{\sqrt{2}}$ and $\varrho = 3$ we construct the matrix B as above and study its singular values, using numerical tools such as MATLAB's `svd()`. The values we obtain (after rounding) are

$$[4,\ 3.00004,\ 3,\ 0.01581]^T$$

and we take note that the singular vakues above the threshold θ are unchanged. To investigate how does the choice of ϱ affect the singular values, let us repeat this for selection of values ; we get

$$[4.00003,\ 4,\ 3,\ 0.01581]^T, \text{ for } \varrho = 4,$$
$$[10.00001,\ 4,\ 3,\ 0.01581]^T, \text{ for } \varrho = 10 \text{ and}$$
$$[100,\ 4,\ 3,\ 0.0158]^T, \text{ for } \varrho = 100.$$

In conclusion, it seems that role of ϱ is rather straightforward.

Returning to the proof, since $\hat{\sigma}_{k+1}$ is the greatest singular value of D, we have

$$\hat{\sigma}_{k+1} = \max_{\|\boldsymbol{x}\|_2=1} \|D\boldsymbol{x}\|_2,$$

and setting $\boldsymbol{x} = [\rho_{k+1}, \ldots, \rho_n]^T$ we get

$$\hat{\sigma}_{k+1} \geq \left\| \begin{bmatrix} \varrho \\ \rho_{k+1}\sigma_{k+1} \\ \vdots \\ \rho_n\sigma_n \end{bmatrix} \right\|_2 \geq \varrho \geq \hat{\sigma}.$$

Similarly, since $\hat{\sigma}_n$ is the smallest singular value of D, we have

$$\hat{\sigma}_n = \min_{\|\boldsymbol{y}\|_2=1} \|D\boldsymbol{y}\|_2$$

and setting $\boldsymbol{y} = [0, \ldots, 0, y_{n-1}, y_n]^T$, where $\|\boldsymbol{y}\|_2 = 1$ and $y_{n-1}\rho_{n-1} + y_n\rho_n = 0$, we obtain

$$\hat{\sigma}_n \leq \|D\boldsymbol{y}\|_2 = \sqrt{(\sigma_{n-1}y_{n-1})^2 + (\sigma_n y_n)^2} \leq \sigma_{n-1}.$$

Let now $j \in \{k+1, \ldots, n-2\}$ be fixed. Let us denote the columns of the matrix $\hat{V}$ by $\hat{\boldsymbol{v}}_{k+1}, \ldots, \hat{\boldsymbol{v}}_n$. Let us then set $\boldsymbol{z} = [0, \ldots, 0, z_j, \ldots, z_n]^T$, where $z_i, i = j, \ldots, n$ satisfies $\|\boldsymbol{z}\|_2 = 1$, $\sum_{i=j}^n \rho_i z_i = 0$ and $\hat{\boldsymbol{v}}_l^T \boldsymbol{z} = 0$, for all $l = j+2, \ldots, n$. Note that we pose the same number of conditions as is the number of z_i to be set. Then we can write

$$\hat{\sigma}_{j+1} = \min\left\{\|D\boldsymbol{x}\|_2 : \|\boldsymbol{x}\|_2 = 1, \boldsymbol{x}^T\hat{\boldsymbol{v}}_l = 0, l = j+2, \ldots, n\right\}$$

and by plugging in $\boldsymbol{x} := \boldsymbol{z}$ we see, that

$$\hat{\sigma}_{j+1} \leq \|D\boldsymbol{z}\|_2 = \sqrt{(\sigma_j z_j)^2 + \ldots + (\sigma_n z_n)^2} \leq \sigma_j.$$

If we now set

$$\sigma'_l := \hat{\sigma}_l, l = k+2, \ldots, n$$

and

$$\{\sigma'_1, \ldots, \sigma'_{k+1}\} := \{\sigma_1, \ldots, \sigma_k, \hat{\sigma}_{k+1}\}$$

in a certain order, we get $\sigma'_i, i = 1, \ldots, n$, with the declared properties.

Lastly, for any given vector $\boldsymbol{w}' \in \boldsymbol{W}'$, $\|\boldsymbol{w}'\|_2 = 1$ we have

$$(\sigma'_{k+2})^2 \geq \|B\boldsymbol{w}'\|_2^2 = \|\varrho\boldsymbol{w}^T\boldsymbol{w}'\|_2^2 + \|A\boldsymbol{w}'\|_2^2.$$

And since $\sigma_{k+1} \geq \sigma'_{k+2}$, we obtain

$$\|A\boldsymbol{w}'\|_2^2 \leq \sigma_{k+1}^2 - \|\varrho\boldsymbol{w}^T\boldsymbol{w}'\|_2^2 \leq \sigma_{k+1}^2,$$

in other words $\boldsymbol{w}' \in \boldsymbol{W}$, $\forall \boldsymbol{w}' \in \boldsymbol{W}'$ and thus $\boldsymbol{W}' \subset \boldsymbol{W}$.

■

Some more information on how the singular values are altered by stacking a vector on a matrix is provided by the following result.

Theorem 1.9. *Let us have $A \in \mathbb{R}^{m\times n}$ as before and let σ be one of the singular values of A and let $\boldsymbol{v}$ be its associated right singular vector. Then for any $\rho > 0$ the matrix*

$$A_\rho := \begin{bmatrix} \rho \boldsymbol{v}^T \\ A \end{bmatrix}$$

has the same singular values as the matrix A, except σ is replaced by $\sqrt{\rho^2+\sigma^2}$.

Proof. Using singular value decomposition $A = U\Sigma V^T$ yields

$$\begin{bmatrix} 1 & 0 \\ 0 & U \end{bmatrix} \begin{bmatrix} 0 & \cdots & 0 & \rho & 0 & \cdots & 0 \\ \sigma_1 & & & 0 & & & \\ & \ddots & & \vdots & & & \\ & & & 0 & & & \\ & & & \sigma & & & \\ & & & & & \ddots & \\ & & & & & & \sigma_n \end{bmatrix} V^T = A_\rho$$

Then by applying Givens rotation (represented by an orthogonal matrix) from the left, we can substitute the column

$$[\rho, 0, \ldots, 0, \sigma, 0, \ldots, 0]^T$$

of that matrix by the column vector

$$[\sqrt{\rho^2+\sigma^2}, 0, \ldots, 0]^T,$$

without altering the other columns or singular vectors. And so we have a decomposition showing us the new singular values. ■

Using these theorems, we can now formulate the following rank revealing algorithm (adaptation of the one found in [12], also see [21]).

Algorithm 1.10. *Let $A \in \mathbb{R}^{m\times n}$ and let us have threshold $\theta > 0$. We aim to compute $k := \operatorname{rank}_\theta(A)$.*

I. We compute matrices Q, R as a QR decomposition of A and set scaling factor $\tau := \|A\|_\infty$.

II. For $i = n, \ldots, 1$ we repeat the following.

(i) We choose a random unit vector $\boldsymbol{x}_0$.

(ii) Using Gauss-Newton iteration (1.3) with $\boldsymbol{x}_0$ as a starting vector we approximate the singular vector $\boldsymbol{w}_i$ of the matrix R as well as the smallest singular number ζ with which it is associated. Number of iterations sufficient for a dependable approximation of the singular pair has to be chosen with respect to the particular problem being solved.

(iii) *If $\zeta > \theta$, we shall stop repeating the step II and set $k := i$; at this point, the computation is finished, i.e. k is the approximate rank of A with respect to θ. Otherwise, we update Q, R to be the QR decomposition of the upper Hessenberg matrix*

$$\begin{bmatrix} \tau \boldsymbol{w}_i^T \\ R \end{bmatrix},$$

update $i := i - 1$ and proceed to another repetition of the step II.

III. In case that $\zeta \leq \theta$ during all the repetitions of the step II, we set k to be 0 (and end the computation).

2. Sylvester Matrices

Now that we have all the basic tools we need, we shall introduce the Sylvester matrix for two univariate polynomials and the notion of its subresultant. We present some of its properties, most importantly the way its rank is connected to the degree of the GCD.

In the end of this chapter, an algorithm showing a reasonably effective way of obtaining the GCD using Sylvester subresultants.

2.1 Definition and Properties

First, let us recall the definition of Cauchy matrix.

Definition 2.1. *Cauchy* matrix of a polynomial $f(x) = \sum_{i=0}^{m} a_i x^{m-i}$ of order k is

$$C_k(f) = \underbrace{\begin{bmatrix} a_0 & & & \\ a_1 & a_0 & & \\ \vdots & a_1 & \ddots & \\ a_m & \vdots & \ddots & a_0 \\ & a_m & & a_1 \\ & & \ddots & \vdots \\ & & & a_m \end{bmatrix}}_{(k+1) \text{ columns}}.$$

Such a matrix is used to carry out polynomial multiplication with vector representation. To illustrate, let us consider polynomials $f(x) = \sum_{i=0}^{m} a_i x^{m-i}$ and $g(x) = \sum_{i=0}^{n} b_i x^{n-i}$, $\deg(f) = m$, $\deg(g) = n$. Then the vector representing the product of $f(x)g(x)$ can be obtained as $C_n(f)\boldsymbol{g}$ or $C_m(g)\boldsymbol{f}$, where $\boldsymbol{g} = [b_0, \ldots, b_n]^T \in \mathbb{R}^{n+1}$ and $\boldsymbol{f} = [a_0, \ldots, a_m]^T \in \mathbb{R}^{m+1}$.

Now that we have refreshed the basics, we can continue with the following definition. It shows that the Sylvester matrix is just a suitable pair of Cauchy matrices.

Definition 2.2. *Sylvester* matrix of polynomials $f(x) = \sum_{i=0}^{m} a_i x^{m-i}$ and $g(x) = \sum_{i=0}^{n} b_i x^{n-i}$, $\deg(f) = m \geq n = \deg(g)$ is ([1, 2, 5, 11])

$$S(f,g) = \underbrace{\begin{bmatrix} a_0 & & & & b_0 & & & \\ a_1 & a_0 & & & b_1 & b_0 & & \\ \vdots & a_1 & \ddots & & \vdots & b_1 & \ddots & \\ a_m & \vdots & \ddots & a_0 & b_n & \vdots & \ddots & b_0 \\ & a_m & & a_1 & & b_n & & b_1 \\ & & \ddots & \vdots & & & \ddots & \vdots \\ & & & a_m & & & & b_n \end{bmatrix}}_{n \text{ columns}} \underbrace{\qquad\qquad}_{m \text{ columns}} = [C_{n-1}(f), C_{m-1}(g)].$$

Moreover, for $k = 1, \ldots, n$ the kth Sylvester subresultant $S_k(f,g)$ is formed from $S(f,g)$ by truncating the last $k-1$ columns of the coefficients of $f(x)$, the last $k-1$ columns of the coefficients of $g(x)$ and the last $k-1$ rows. In other words

$$S_k(f,g) = \underbrace{\begin{bmatrix} a_0 & & & & b_0 & & & \\ a_1 & a_0 & & & b_1 & b_0 & & \\ \vdots & a_1 & \ddots & & \vdots & b_1 & \ddots & \\ a_m & \vdots & \ddots & a_0 & b_n & \vdots & \ddots & b_0 \\ & a_m & & a_1 & & b_n & & b_1 \\ & & \ddots & \vdots & & & \ddots & \vdots \\ & & & a_m & & & & b_n \end{bmatrix}}_{(n-k+1) \text{ columns}} \underbrace{\qquad\qquad}_{(m-k+1) \text{ columns}}.$$

Let us note, that $S(f,g) = S_1(f,g)$ is a square $(m+n) \times (m+n)$ matrix, while $S_k(f,g)$ is a $(m+n-k+1) \times (m+n-2k+2)$ (generally rectangular) matrix.

To show how the subresultants are constructed, let us have polynomials

$$f(x) = a_0 x^3 + a_1 x^2 + a_2 x + a_3$$
$$g(x) = b_0 x^2 + b_1 x^1 + b_2.$$

Then we have

$$S(f,g) = S_1(f,g) = \begin{bmatrix} a_0 & & b_0 & & \\ a_1 & a_0 & b_1 & b_0 & \\ a_2 & a_1 & b_2 & b_1 & b_0 \\ a_3 & a_2 & & b_2 & b_1 \\ & a_3 & & & b_2 \end{bmatrix} \rightarrow S_2(f,g) = \begin{bmatrix} a_0 & b_0 & \\ a_1 & b_1 & b_0 \\ a_2 & b_2 & b_1 \\ a_3 & & b_2 \end{bmatrix}$$

The subresultant $S_3(f,g)$ is not defined and we can see that if we tried to construct it, it would be lacking any coefficients of $f(x)$.

The following theorem shows the connection between Sylvester subresultants and polynomial GCD.

Theorem 2.3. *Let us have polynomials $f(x)$, $g(x)$, $\deg(f) = m \geq n = \deg(g)$. Then for $j \in \{0, \ldots, n\}$*

$$\deg(\mathrm{GCD}(f,g)) = j \Leftrightarrow \mathrm{rank}(S(f,g)) = m + n - j$$

and similarly for $k \in \{1, \ldots, n\}$

$$\deg(\mathrm{GCD}(f,g)) = k \Leftrightarrow \mathrm{rank}(S_k(f,g)) = m + n - 2k + 1.$$

This is also equivalent to $S_k(f,g)$ being rank deficient by 1.

Proof. This result is presented in [17]; for a complete proof see [1] or [18]. It follows from transformations of the subresultant matrix similar to the ones described in the next section. ■

As an example, let us have polynomials

$$\begin{aligned} f(x) &= (x-1)^2(x-2)^1 = x^3 - 4x^2 + 5x - 2, \\ g(x) &= (x-2)^2 \qquad\quad\ = x^2 - 4x + 4. \end{aligned}$$

It is easy to determine that $\mathrm{GCD}(f,g) = x - 2$ and therefore $\deg(\mathrm{GCD}(f,g)) = 1$. We form the Sylvester matrix

$$S(f,g) = S_1(f,g) = \begin{bmatrix} 1 & & 1 & & \\ -4 & 1 & -4 & 1 & \\ 5 & -4 & 4 & -4 & 1 \\ 2 & 5 & & 4 & -4 \\ & 2 & & & 4 \end{bmatrix}.$$

If we, for example, transform this matrix to its upper triangular form, we can see, that $\mathrm{rank}(S(f,g)) = 4 = \deg(f) + \deg(g) - 1$ and therefore, as was expected, $\deg(\mathrm{GCD}(f,g)) = 1$.

Returning to general theory, following the Theorem 2.3, we see that if $k = \deg(\mathrm{GCD}(f,g))$, then $\dim(\mathrm{Ker}(S_k(f,g))) = 1$. The null space of $S_k(f,g)$ is in a close relation to the GCD, as is shown in the following lemma found in [19] and also presented in [7].

Lemma 2.4. *Let us have polynomials $f(x)$, $g(x)$ as above and moreover let $k = \deg(\mathrm{GCD}(f,g))$. Set $\boldsymbol{x}$ to be a null vector of $S_k(f,g)$, i.e. $\boldsymbol{x} \in \mathbb{R}^{(m+n-2k+2)}$. Then*

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{w} \\ -\boldsymbol{v} \end{bmatrix},$$

where $\boldsymbol{v} \in \mathbb{R}^{m-k+1}$ and $\boldsymbol{w} \in \mathbb{R}^{n-k+1}$ are vectors containing coefficients of $v(x), w(x)$: $f(x) = v(x)\,\mathrm{GCD}(f,g)$, $g(x) = w(x)\,\mathrm{GCD}(f,g)$.

Proof. Let us have the vector $\boldsymbol{x} \in \mathbb{R}^{(m+n-2k+2)}$ in the above form. We note that $S_k(f,g) = [C_{n-k}(f), C_{m-k}(g)]$ and therefore $S_k(f,g)\boldsymbol{x}$ represents the polynomial $f(x)w(x) - g(x)v(x)$, where $\deg(v(x)) < deg(f(x))$ and $\deg(w(x)) < deg(g(x))$. Recalling that $\boldsymbol{x} \in \mathrm{Ker}(S_k(f,g))$ we get

$$f(x)w(x) - g(x)v(x) = 0$$

and so $f(x) = v(x)\,\mathrm{GCD}(f,g)$, $g(x) = w(x)\,\mathrm{GCD}(f,g)$. Since dimension of the kernel was just 1, any other vector would be a multiple of this particular $\boldsymbol{x}$. ■

In the setting of the example above, let us form a vector

$$\boldsymbol{x} = [\underbrace{1, -2}_{\boldsymbol{w}}, \underbrace{-1, 2, -1}_{-\boldsymbol{v}}]^T$$

and we easily check that $S_1(f, g)\boldsymbol{x} = 0$.

2.2 Matrix version of Euclid's Algorithm

The problem of finding the GCD of two polynomials can be mathematically solved via the classical Euclid's algorithm, which unfortunately can be unstable in the floating point environment.

We will demonstrate that a certain column-wise elimination applied on the Sylvester matrix is a direct analogy of the Euclid's algorithm. To keep it simple, we will only present it for a particular pair of polynomials, since the generalization is rather straightforward.

Let us then have real polynomials

$$f_1(x) = \sum_{i=0}^{4} a_i^1 x^{4-i}, \quad \deg(f_1) = 4,$$
$$f_2(x) = \sum_{i=0}^{2} a_i^2 x^{2-i}, \quad \deg(f_2) = 2$$

and prepare the appropriate Sylvester matrix

$$S(f_1, f_2) = \begin{bmatrix} a_0^1 & & a_0^2 & & & \\ a_1^1 & a_0^1 & a_1^2 & a_0^2 & & \\ a_2^1 & a_1^1 & a_2^2 & a_1^2 & a_0^2 & \\ a_3^1 & a_2^1 & & a_2^2 & a_1^2 & a_0^2 \\ a_4^1 & a_3^1 & & & a_2^2 & a_1^2 \\ & a_4^1 & & & & a_2^2 \end{bmatrix}$$

In Euclid's algorithm, we would now start to remove high power terms of f_1 using some multiples of f_2. In our case, we carry out a close analogy of this. Using column wise elimination, we will eliminate the coefficients a_0^1, a_1^1, a_2^1 in the first two columns just by subtracting suitable multiples of the columns $3, \ldots, 6$. This way we obtain a new polynomial of a lesser degree – let us denote it f_3 and its coefficients by a_i^3:

$$S(f_1, f_2) \to \begin{bmatrix} 0 & & a_0^2 & & & \\ 0 & 0 & a_1^2 & a_0^2 & & \\ 0 & 0 & a_2^2 & a_1^2 & a_0^2 & \\ a_0^3 & 0 & & a_2^2 & a_1^2 & a_0^2 \\ a_1^3 & a_0^3 & & & a_2^2 & a_1^2 \\ & a_1^3 & & & & a_2^2 \end{bmatrix}.$$

Again, similarly to Euclid's algorithm, we will now swap the roles of the two polynomials, form $S(f_2, f_3)$ and will eliminate terms of f_2 using f_3. This continues until one of the polynomials is trivial.

With the use of some permutation, we can carry out this whole process inside the original Sylvester matrix $S(f_1, f_2)$ in such a way that it is a transformation to a lower triangular form. Specifically, if we transfer the first two columns to the last two positions, we obtain

$$\begin{bmatrix} 0 & & a_0^2 & & & \\ 0 & 0 & a_1^2 & a_0^2 & & \\ 0 & 0 & a_2^2 & a_1^2 & a_0^2 & \\ a_0^3 & 0 & & a_2^2 & a_1^2 & a_0^2 \\ a_1^3 & a_0^3 & & & a_2^2 & a_1^2 \\ & a_1^3 & & & & a_2^2 \end{bmatrix} \rightarrow \begin{bmatrix} a_0^2 & & & 0 & \\ a_1^2 & a_0^2 & & 0 & 0 \\ a_2^2 & a_1^2 & a_0^2 & 0 & 0 \\ & a_2^2 & a_1^2 & a_0^2 & a_0^3 & 0 \\ & & a_2^2 & a_1^2 & a_1^3 & a_0^3 \\ & & & a_2^2 & & a_1^3 \end{bmatrix}$$

and we see that the bottom right 3×3 submatrix is precisely $S(f_2, f_3)$. As mentioned above, we now eliminate high power terms of f_2 using columns of f_3 and get

$$\begin{bmatrix} a_0^2 & & & & 0 & \\ a_1^2 & a_0^2 & & & 0 & 0 \\ a_2^2 & a_1^2 & a_0^2 & & 0 & 0 \\ & a_2^2 & a_1^2 & a_0^2 & a_0^3 & 0 \\ & & a_2^2 & a_1^2 & a_1^3 & a_0^3 \\ & & & a_2^2 & & a_1^3 \end{bmatrix} \rightarrow \begin{bmatrix} a_0^2 & & & & & \\ a_1^2 & a_0^2 & & & & \\ a_2^2 & a_1^2 & a_0^2 & & & \\ & a_2^2 & a_1^2 & 0 & a_0^3 & \\ & & a_2^2 & 0 & a_1^3 & a_0^3 \\ & & & a_0^4 & & a_1^3 \end{bmatrix}.$$

This of course yields yet another new polynomial, $f_4(x) := a_0^4$. One last permutation to transfer the forth column on the last position gives the desired lower triangular matrix:

$$\begin{bmatrix} a_0^2 & & & & & \\ a_1^2 & a_0^2 & & & & \\ a_2^2 & a_1^2 & a_0^2 & & & \\ & a_2^2 & a_1^2 & a_0^3 & & \\ & & a_2^2 & a_1^3 & a_0^3 & \\ & & & a_1^3 & a_0^4 & \end{bmatrix}.$$

This process ends when we obtain a zero polynomial and the GCD is then the last non-zero polynomial in the sequence $f_1, f_2, \ldots$ formed during calculation. In our case we implicitly assumed that the polynomials $f_1, \ldots, f_4$ are not identically zero, and therefore $f_4(x) := a_0^4$ would be the GCD. Of course, generally, it might happen that f_3 or f_4 will turn out to be trivial. This process can be carried out regardless, we just obtain some zero columns in the matrix. The last non-zero column then contains the coefficients of the GCD, as illustrated for $f_4(x) = 0$:

$$\begin{bmatrix} a_0^2 & & & & & \\ a_1^2 & a_0^2 & & & & \\ a_2^2 & a_1^2 & a_0^2 & & & \\ & a_2^2 & a_1^2 & a_0^3 & & \\ & & a_2^2 & a_1^3 & a_0^3 & \\ & & & & a_1^3 & 0 \end{bmatrix}.$$

The reason why this works is simply because we use the same operations to find the GCD as in the Euclid's algorithm, only using different polynomial representation.

2.3 Algorithm based on Sylvester Subrezultants

Let us now formulate an algorithm for determining the GCD of two univariate polynomials using Sylvester subresultant matrices in a more sophisticated way.

We will use the Theorem 2.3, specifically the fact that

$$\deg(\mathrm{GCD}(f,g)) = k \Leftrightarrow \mathrm{rank}(S_k(f,g)) = m+n-2k+1.$$

This enables us to construct a sequence of Sylvester subresultants

$$S_n(f,g),\ S_{n-1}(f,g),\ S_{n-2}(f,g)\ldots,$$

so the $S_k(f,g)$ is the first rank deficient one. Once it is identified, we obtain the degree of the GCD and continue to extract its coefficients according to Lemma 2.4. This algorithm is due to Li and Zeng (see [12]).

Algorithm 2.5. *Let there be $m,n \in \mathbb{N}:\ m > n$ and $f(x) = \sum_{i=0}^{m} a_i x^{m-i}$, $g(x) = \sum_{i=0}^{n} b_i x^{n-i}$ two polynomials of degree m and n respectively. We aim to compute the coefficients $s_0,\ldots,s_k$ of the polynomial $s = \mathrm{GCD}(f,g)$.*

I. Firstly, matrices Q and R are computed as the QR decomposition of the n-th Sylvester subresultant $S_n(f,g) = QR$. Note that this subresultant is a $(m+1)\times(m-n+2)$ matrix consisting of one column with the coefficients of $f(x)$ and $m-n+1$ columns with the coefficients of $g(x)$.

II. For $j = 1,\ldots,n$ the following is repeated:

(i) Approximate rank of R is determined. This can be accomplished with relative ease, since R is an upper triangular matrix. Gauss-Newton iteration (1.3) as well as Algorithm 1.10 can be used for this purpose.

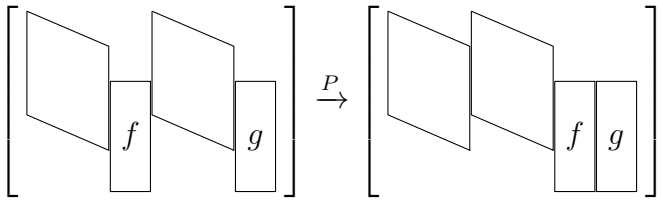
(ii) If R is rank-deficient, the right singular vector $\boldsymbol{z}$ associated to the smallest singular value of R is computed. This can again be done trough iteration 1.3. Coefficients of the polynomial $w(x)$ are then obtained from

$$\boldsymbol{z} = \begin{bmatrix} & \boldsymbol{w} \\ - & \boldsymbol{v} \end{bmatrix},$$

where $\boldsymbol{w} \in \mathbb{R}^j$. Since $f(x) = s(x)\cdot w(x)$, $s(x)$ is now obtained using some form of polynomial division and this algorithm is finished.

(iii) Otherwise, if R is of full column rank, we update Q and R to be the QR decomposition of $S_{n-j}(f,g)$. But since S_{n-j} is created from S_{n-j+1} by a simple addition of one row and two columns, we can use the already known decomposition of S_{n-j+1}.

Specifically, let us have a permutation P of the columns of S_{n-j} such that the two new columns are the last two columns of PS_{n-j}; schematically written:



It now holds that

$$\underbrace{\begin{bmatrix} Q & \\ & 1 \end{bmatrix}^T}_{=:Q_1^T} PS_{n-j} = \begin{bmatrix} R & & \\ & Q_1^T f & Q_1^T g \end{bmatrix}.$$

To obtain an upper triangular matrix, it is now only needed to deal with the bottom part of the last two columns of $Q_1^T PS_{n-j}$ The QR decomposition of this submatrix is computed and denoted by Q_2, R_2, so that

$$\begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \\ & & & Q_2^T \end{bmatrix} Q_1^T PS_{n-j} = \begin{bmatrix} R & & \\ & R_2 & \end{bmatrix}.$$

We repeat the step II with $j := j + 1$. In the case that $j = n$ already, the aforementioned update shall not be executed. Instead, we conclude, that $\deg(\mathrm{GCD}(f, g)) = 0$ *and* $s(x) = 1$.

End of algorithm.

2.4 Multiple Polynomials

We are now able to compute the GCD of two polynomials. Naturally, we explore possibilities of applications for our algorithms and find use for it in computation of GCD of multiple polynomials.

In this section, a multi-polynomial GCD finding algorithm, taking advantage of our previous procedures, is presented as a continuation of the two polynomial theory.

Let us now have $N \in \mathbb{N}$ and $N + 1$ polynomials f_0, f_1 ..., f_N, such that $\deg(f_0) \geq \deg(f_i)$, $i \in 1, \ldots, N$. We can make the following simple observation. If we construct a polynomial

$$h = \mathrm{GCD}\left(f_N, \mathrm{GCD}\left(f_{N-1}, \mathrm{GCD}\left(\ldots, \mathrm{GCD}\left(f_1, f_0\right)\right)\ldots\right)\right), \tag{2.1}$$

then it is surely the greatest common divisor of f_0, $f_1, \ldots, f_N$. Note that to compute such a polynomial, we would not necessarily need any additional tools, we can simply repeatedly use the algorithm for two polynomials as the construction of h suggests. But in the floating point environment, we never acquire a precise answer. Bearing this in mind, one can see how this approach could lead to big accumulated errors over the repeated computations. Hence, we will try to improve this strategy.

Having used the Sylvester matrix for two polynomials, it is only natural that we would try to use some analogous matrix for this problem. Recall, that

$S(f,g) = [C_{n-1}(f), C_{m-1}(g)]$, and let us have $N+1$ polynomials $f_0, \ldots, f_N$ as above. We consider the following system

$$F(\boldsymbol{x}) = \boldsymbol{b}, \tag{2.2}$$

where

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{h} \\ \boldsymbol{w_0} \\ \boldsymbol{w_1} \\ \cdot \\ \cdot \\ \cdot \\ \boldsymbol{w}_N \end{bmatrix}, \qquad \boldsymbol{b} = \begin{bmatrix} 1 \\ \boldsymbol{f_0} \\ \boldsymbol{f_1} \\ \cdot \\ \cdot \\ \cdot \\ \boldsymbol{f_N} \end{bmatrix}, \qquad F(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{r}^T\boldsymbol{h} \\ C_d(w_0)\boldsymbol{h} \\ C_d(w_1)\boldsymbol{h} \\ \cdot \\ \cdot \\ \cdot \\ C_d(w_N)\boldsymbol{h} \end{bmatrix}.$$

Here, $\boldsymbol{f}_i$ denotes the vector of coefficients of the polynomial f_i, $\boldsymbol{h}$ is the vector corresponding to the GCD, which we are trying to obtain and $\boldsymbol{w}_i$ corresponds to cofactors $w_i(x)$ so $f_i = w_i h,\ i = 0, \ldots, N$. The vector r is arbitrarily chosen for scaling of h. If we wanted, for example, the computed GCD to be monic, we would set $\boldsymbol{r} = \boldsymbol{e}_1 \in \mathbb{R}^d,\ d = \deg(h)$.

This approach has a major drawback in the fact that we need to know $d = \deg(h) = \deg(\mathrm{GCD}(f_0, \ldots, f_N))$ before we even start. But we already have a way of finding it. Let us remember, that we are trying to improve the accuracy of (2.1). Therefore, we can carry out the computation of (2.1) and then use the results as an input for solving (2.2). This also answers the question of how to get any initial approximation of h and w_i.

The system (2.2), is solved in the least squares sense as a problem of minimizing $\|F(\boldsymbol{x}) - b\|$. This is done via Gauss-Newton iteration. The following lemma describes how the Jacobian of $F(\boldsymbol{x})$ looks and some of its properties, that are presented in [19].

Lemma 2.6. *Let $\boldsymbol{x}$, $\boldsymbol{b}$ and $F(\boldsymbol{x})$ be defined as above, then the Jacobian $J(\boldsymbol{x})$ of $F(\boldsymbol{x})$ is*

$$J(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{r}^T & & & \\ C_d(w_0) & C_{\deg(w_0)}(h) & & \\ \vdots & & \ddots & \\ C_d(w_N) & & & C_{\deg(w_N)}(h) \end{bmatrix}.$$

If $\|F(\boldsymbol{x}) - \boldsymbol{b}\|$ has a local minimum at $\boldsymbol{x}$, then $J(\boldsymbol{x})^T (F(\boldsymbol{x}) - \boldsymbol{b}) = 0$. Moreover, for any $\boldsymbol{r}$, such that $\boldsymbol{r}^T\boldsymbol{u} \neq 0$ holds:

$$w_0, \ldots, w_N \text{ are all relatively coprime} \Rightarrow J(\boldsymbol{x}) \text{ has full rank.}$$

Proof. The structure of $J(\boldsymbol{x})$ can be obtained directly as a derivative of $F(\boldsymbol{x})$. Similarly, the second statement is a result of the necessary conditions for local minimum, that is gradient of $\|F(\boldsymbol{x}) - \boldsymbol{b}\|$, which is $J(\boldsymbol{x})^T (F(\boldsymbol{x}) - \boldsymbol{b}) = 0$, has to be 0. See [19], Lemma 1.

Finally, the proof of the last implication is found in [19], Proposition 1. ∎

The following is presented in [19] and provides us with the fact that the computation of AGCD is not an ill-posed problem.

Suppose that f_j are perturbed into $\tilde{f}_j$ and the corresponding AGCD factors are now $\tilde{h}$ and $\tilde{w_j}$, with the assumption, that $\deg(h) = \deg(\tilde{h})$. Then

$$F(\tilde{\boldsymbol{x}}) - F(\boldsymbol{x}) = J(\boldsymbol{x})(\tilde{\boldsymbol{x}} - \boldsymbol{x}) + O(\|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|)^2.$$

If we omit the last term, we have

$$\|F(\tilde{\boldsymbol{x}}) - F(\boldsymbol{x})\| = \|J(\boldsymbol{x})(\tilde{\boldsymbol{x}} - \boldsymbol{x})\| \geq \sigma_{min}\|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|,$$

denoting the smallest singular number of $J(\boldsymbol{x})$ by σ_{min}. Let $\|F(\tilde{\boldsymbol{x}}) - \tilde{\boldsymbol{b}}\| \leq \varepsilon$. Because $F(\boldsymbol{x}) = \boldsymbol{b}$, we can write that $\|F(\boldsymbol{x}) - \boldsymbol{b}\| \leq \varepsilon$ too. Hence

$$\begin{aligned}
\|\tilde{\boldsymbol{x}} - \boldsymbol{x}\| &\leq \frac{1}{\sigma_{min}} \|F(\tilde{\boldsymbol{x}}) - F(\boldsymbol{x})\|, \\
&\leq \frac{1}{\sigma_{min}} \Big(\Big\|F(\tilde{\boldsymbol{x}}) - \tilde{\boldsymbol{b}}\Big\| + \Big\|\tilde{\boldsymbol{b}} - \boldsymbol{b}\Big\| + \|\boldsymbol{b} - F(\boldsymbol{x})\| \Big), \\
&\leq \frac{1}{\sigma_{min}} \Big(2\varepsilon + \Big\|\tilde{\boldsymbol{b}} - \boldsymbol{b}\Big\| \Big).
\end{aligned}$$

Therefore we have an asymptotic bound of error of $\boldsymbol{x}$ by the error of $\boldsymbol{b}$. Now we just recall, that $\boldsymbol{x}$ contains the coefficients of GCD and the corresponding cofactors, i.e. the data to be computed, while $\boldsymbol{b}$ contains all the input data (coefficients of the given polynomials). In conclusion, computing the GCD in this sense is no longer an ill-posed problem.

Following is an algorithm based on this approach. It is a modification of one shown in [19].

Algorithm 2.7. *Let there be polynomials $f_0, \ldots, f_N$ of degrees n_0, $n_1, \ldots, n_N$ and given tolerance $\theta \in \mathbb{R}$. We seek the polynomial $h = \mathrm{GCD}(f_0, f_1, f_2, \ldots, f_N)$.*

I. We set $h := f_0$ and then the following is repeated for $i = 1, 2, \ldots, N$:

(i) We compute $h(x) := GCD(h, f_i)$ via some two-polynomial GCD *finder, e.g. Algorithm 2.5.*

(ii) Polynomial division is used to find the necessary cofactors $w_k := \frac{f_k}{h}$, $k = 0, \ldots, i$

(iii) We construct the vector of an initial approximation

$$\boldsymbol{x}^T := \left[h^T, w_0^T, \ldots, w_i^T\right]$$

and perform the Gauss-Newton iteration: repeat

$$x^+ := x - \begin{bmatrix} \boldsymbol{r}^T & & & \\ C_d(w_0) & C_{\deg(w_0)}(h) & & \\ \vdots & & \ddots & \\ C_d(w_N) & & & C_{\deg(w_N)}(h) \end{bmatrix}^{\dagger} \begin{bmatrix} \boldsymbol{r}^T\boldsymbol{h} \\ C_d(w_0)\boldsymbol{h} - f_0 \\ C_d(w_1)\boldsymbol{h} - f_1 \\ \vdots \\ C_d(w_N)\boldsymbol{h} - f_N \end{bmatrix},$$

$$x := x^+,$$

until $\|F(\boldsymbol{x}) - \boldsymbol{b}\| < \theta$.

(iv) Once $\|F(\boldsymbol{x}) - \boldsymbol{b}\| < \theta$*, we extract coefficients of the polynomial* $h(x)$ *from the vector* $\boldsymbol{x}$*.*

II. We now have $h(x) = GCD\ (f_0, f_1, f_2, \ldots, f_N)$*.*

End of algorithm.

3. Bézout Matrices

It is perhaps not surprising, that there are other types of matrices, that can be used in a similar way to the Sylvester matrix. Since we need to have some context, something to compare our results based on Sylvester subresultants against, we devote this chapter to the Bézout matrices and presenting an algorithm based on them.

After providing basic definition, we explore results of Pták (see [14]) based on properties of commutators. This enables us to construct Bézout matrices in a much simpler way than the definition. Properties of Bézout matrices are then presented so that we can in the end come up with an algorithm based on the one presented in [2].

The very last portion of this chapter summarizes some results about finding GCD of several polynomials via Bézout matrices and also concludes with an algorithm to serve as a parallel to the Sylvester based finder for several polynomials.

3.1 Definition and Properties

As before, the ith canonical basis vector of $\mathbb{R}^m$ is denoted by $\boldsymbol{e}_i$ and defined as the ith column of the $m \times m$ identity matrix I. Outer product matrices of these vectors are $E_{i,j} = \boldsymbol{e}_i \boldsymbol{e}_j{}^T$ and $E_i = \boldsymbol{e}_i \boldsymbol{e}_i{}^T$, for $i, j = 1, \ldots, m$.

Forward shift matrix is the matrix $S = [\mathbf{0}, \boldsymbol{e}_1, \ldots, \boldsymbol{e}_{m-1}]$, similarly its transpose $S^T = [\boldsymbol{e}_2, \ldots, \boldsymbol{e}_m, \mathbf{0}]$ is called the backward shift matrix. Flip matrix J is defined as $J = [\boldsymbol{e}_m, \ldots, \boldsymbol{e}_1]$ and clearly $J^2 = I$ and $S^T = JSJ$.

The commutator of two $m \times m$ matrices A and B is $[A, B] = AB - BA$.

For a real polynomial $f(x) = a_0 x^m + a_1 x^{m-1} \ldots + a_m$ of degree m, we define

$$f^*(x) := x^m f(\tfrac{1}{x}) = a_m x^m + a_{m-1} x^{m-1} \ldots + a_0.$$

Let us have two real polynomials of degrees m, n: $m \geq n$, namely

$$f(x) = \sum_{i=0}^{m} a_i x^{m-i} \text{ and } g(x) = \sum_{i=0}^{n} b_i x^{n-i}.$$

Definition 3.1. The *Bézout matrix* of f and g is a matrix $B = B(f, g) = (m_{ik})_{i,k=1}^{m}$, where the entries m_{ik} are obtained from the relation

$$\frac{f(x)g(y) - f(y)g(x)}{x - y} = \sum_{i,k=0}^{m-1} m_{ik} x^i y^k = [1, x, \ldots, x^{m-1}] B [1, y, \ldots, y^{m-1}]^T.$$

The polynomial $\frac{f(x)g(y)-f(y)g(x)}{x-y}$ itself is called the *Bézoutian* of f and g.

Let us now demonstrate this on an example.

Choosing

$$f(x) = a_0 x^2 + a_1 x + a_2 \quad \text{and} \quad g(y) = b_0 x^2 + b_1 x + b_2,$$

so $m = n = 2$, the expression $\frac{f(x)g(y)-f(y)g(x)}{x-y}$ equals to

$$\frac{(x-y)(b_2a_1 - a_2b_1) + (x^2 - y^2)(b_2a_0 - a_2b_0) + xy(x-y)(b_1a_0 - a_1b_0)}{x-y}$$

so the matrix is then

$$B(f,g) = (m_{ik})_{i,k=1}^{m} = \begin{bmatrix} a_1b_2 - a_2b_1, & a_0b_2 - a_2b_0 \\ a_0b_2 - a_2b_0, & a_0b_1 - a_1b_0 \end{bmatrix}.$$

Definition 3.2. Let us have an $m \times m$ matrix $M = (m_{ik})_{i,k=1}^{m}$. Than the bivariate polynomial

$$\sum_{i,k=1}^{m} m_{ik} x^k y^i = [1, y, \ldots, y^{m-1}] M [1, x, \ldots, x^{m-1}]^T$$

is the *generating function* of M. Conversely, if there exist a matrix so that a given bivariate polynomial is its generating function, we call it the *generating matrix* of such polynomial.

Note that the Bézout matrix is the generating matrix of the Bézoutian of two polynomials. It is by definition a transpose of the generating matrix of Bézoutian, but since it is symmetric, both definitions merge in this case.

3.1.1 Commutators

This section explores the excellent article [14] by Vlastimil Pták. Naturally, all of the results presented here are adapted from that text.

Lemma 3.3. *Let $r,\ s \in \mathbb{N} \cup \{0\}$ satisfy $0 \le r \le s < m$. Then the following equalities holds:*

$$S^r (S^T)^s = \begin{bmatrix} 1 \leftarrow_{(1+s-r,1)} & & \\ & 1 \leftarrow_{(m-r,m-s)} & \\ & 0 & \\ & & {}_{(m,m+r-s)} \rightarrow 0 \end{bmatrix}, \quad (S^T)^s S^r = \begin{bmatrix} 0 \leftarrow_{(1+s-r,1)} & & \\ & 0 & \\ & & 1 \leftarrow_{(1+s,1+r)} \\ & & {}_{(m,m+r-s)} \rightarrow 1 \end{bmatrix}.$$

On the other hand, if $s < r$, then

$$S^r (S^T)^s = \begin{bmatrix} & 1 \leftarrow_{(1,1+r-s)} & \\ {}_{(m-r,m-s)} \rightarrow 1 & & \\ & 0 & \\ & & {}_{(m-r+s,m)} \rightarrow 0 \end{bmatrix}, \quad (S^T)^s S^r = \begin{bmatrix} 0 \leftarrow_{(1,1-s+r)} & & \\ & 0 & \\ {}_{(1+s,1+r)} \rightarrow 1 & & \\ & {}_{(m+s-r,m)} \rightarrow 1 & \end{bmatrix}.$$

Proof. Firstly, let us recall that

$$S^r = \begin{bmatrix} & 1 \leftarrow_{(1,1+r)} & \\ {}_{(k,k+r)} \rightarrow 1 & & \\ & & {}_{(m-r,m)} \rightarrow 1 \end{bmatrix}, \ k = 1, \ldots, m-r, \tag{3.1}$$

$$(S^T)^s = \begin{bmatrix} 1 \leftarrow_{(1+s,1)} & & \\ & 1 \leftarrow_{(k+s,k)} & \\ & & {}_{(m,m-s)} \rightarrow 1 \end{bmatrix}, \ k = 1, \ldots, m-s, \tag{3.2}$$

According to identities (3.1), (3.2) we see, that non-zero elements of the product $S^r(S^T)^s$ (equal to 1) are on the following positions.

On kth row of the matrix S^r is a unit term only at the $(r+k)$th position, for $k = 1, \ldots, m-r$. For a resulting term at position (k, j) of $S^r(S^T)^s$ not to be zero, $s + j = r + k$ must hold.

Assuming that $s \geq r$, i.e. $s - r \geq 0$, we have $k = s - r + j$ and going trough $j = 1, \ldots, m - s$ we get $k = (s - r + 1), \ldots, \underbrace{(s - r + m - s)}_{(m-r)}$ respectively.

Conversely, assuming $r > s$, we have $j = r - s + k$, where $r - s > 0$ and setting $k = 1, \ldots, m - r$ we in turn obtain $j = (r - s + 1), \ldots, (m - s)$.

In conclusion, the only non-zero terms of $S^r(S^T)^s$ are at positions

$$\begin{aligned} &(s - r + j, j),\ j = 1, \ldots, m - s \quad \text{for } s \geq r, \\ &(k, r - s + k),\ k = 1, \ldots, m - r \quad \text{for } s < r. \end{aligned}$$

The calculation of $(S^T)^s S^r$ is done analogously. ■

Theorem 3.4. *Let us have two polynomials f, g of degrees m, n respectively, $m \geq n$. Then*

$$[f(S^T), g^*(S)] = [g(S^T), f^*(S)].$$

Proof. The operator $[\cdot, \cdot]$ is bilinear, so it is sufficient to only show that

$$[S^r, (S^T)^s] = [S^{m-s}, (S^T)^{m-r}],$$

where r,$s \in \mathbb{N} \cup \{0\}$ and $0 \leq r, s \leq m$.

According to the definition of commutator, it is

$$\begin{aligned} [S^r, (S^T)^s] &= S^r(S^T)^s - (S^T)^s S^r, \\ [S^{m-s}, (S^T)^{m-r}] &= S^{m-s}(S^T)^{m-r} - (S^T)^{m-r} S^{m-s}. \end{aligned}$$

Let us assume $s \geq r$. Using Lemma 3.3 we obtain either

$$[S^r, (S^T)^s] = \begin{bmatrix} 1 \overset{\text{(1+s-r,1)}}{\leftarrow} 1 & & & \\ & 0 \overset{\text{(1+s,1+r)}}{\leftarrow} 0 \overset{\text{(m-r,m-s)}}{\leftarrow} & & \\ & & -1 & \\ & & & \overset{\text{(m,m+r-s)}}{\rightarrow} -1 \end{bmatrix}$$

for $1 + s < m - r$, or

$$[S^r, (S^T)^s] = \begin{bmatrix} 1 \overset{\text{(1+s-r,1)}}{\leftarrow} 1 & & & \\ & 0 \overset{\text{(m-r+1,m-s+1)}}{\leftarrow} & & \\ & & 0 \overset{\text{(s,r)}}{\leftarrow} & \\ & & & -1 \overset{\text{(m,m+r-s)}}{\rightarrow} -1 \end{bmatrix}$$

for $1 + s \geq m - r$.

For the second part, we note that now $m-s \leq m-r$ and have either

$$[S^{m-s}, (S^T)^{m-r}] = \begin{bmatrix} 1 \overset{\text{(1+s-r,1)}}{\leftarrow} & & & \\ & 1 & & \\ & & 0 \overset{\text{(1+m-r,1+m-s)}}{\leftarrow} & \\ & & & 0 \overset{\text{(m-m+s,m-m+r)}}{\leftarrow} \\ & & & -1 \\ & & \overset{\text{(m,m+r-s)}}{\rightarrow} & -1 \end{bmatrix}$$

for $1+s \geq m-r$, or

$$[S^{m-s}, (S^T)^{m-r}] = \begin{bmatrix} 1 \overset{\text{(1+s-r,1)}}{\leftarrow} & & & \\ & 1 & & \\ & & 0 \overset{\text{(m-m+s+1,m-m+r+1)}}{\leftarrow} & \\ & & & 0 \overset{\text{(m-r,m-s)}}{\leftarrow} \\ & & & -1 \\ & & \overset{\text{(m,m+r-s)}}{\rightarrow} & -1 \end{bmatrix}$$

for $1+s < m-r$.

The case of $r < s$ can be treated in the same way. We then see that the identity holds. ■

Theorem 3.5. *Let f, g be real polynomials, $\deg f = m$, $\deg g \leq m$. Then the generating matrix of the Bézoutian of f and g can be written in the following form:*

$$\Big(f(S^T)g^*(S) - g(S^T)f^*(S)\Big)J \tag{3.3}$$

Proof. Since the mapping $B(f,\cdot)$ is linear, it is sufficient to prove that

$$B(f,x^k) = \Big(f(S^T)S^{m-k} - (S^T)^k f^*(S)\Big)J,\ k = 0,\ldots,m.$$

We prove that the generating function of $B(f,x^k)$ is

$$\begin{aligned} \frac{f(x)y^k - x^k f(y)}{x-y} &= \frac{f(x)-f(y)}{x-y}y^k - f(y)\frac{y^k-x^k}{x-y} \\ &= \Pi\left(\frac{f(x)-f(y)}{x-y}y^k\right) - \Pi\left(f(y)\frac{y^k-x^k}{x-y}\right), \end{aligned}$$

where Π is the truncation operator discarding polynomial terms with powers of y exceeding $m-1$. This operator ensures that higher degree terms, which in the end cancel out, do not appear and therefore the respective generating matrices dimensions are kept at $m \times m$.

For $f(x) = a_0x^m + a_1x^{m-1}\ldots + a_m$, we compute

$$\begin{aligned} \frac{f(x)-f(y)}{x-y} &= \frac{a_0x^m + \ldots + a_m - a_0y^m - \ldots - a_m}{x-y} \\ &= \frac{a_0(x^m-y^m) + \ldots + (x-y)a_{m-1} + 0}{x-y}, \\ &= a_0\sum_{i+j=m-1} x^iy^j + a_1\sum_{i+j=m-2} x^iy^j + \ldots + a_{m-1} \end{aligned}$$

and so the generating matrix H of $\frac{f(x)-f(y)}{x-y}$ is

$$H = \begin{bmatrix} a_{m-1} & \cdots & a_0 \\ \vdots & & \vdots \\ a_0 & \cdots & 0 \end{bmatrix}$$
$$= \begin{bmatrix} a_{m-1} & 0 & \cdots \\ 0 & 0 & \\ \vdots & & \ddots \end{bmatrix} + \begin{bmatrix} 0 & a_{m-2} & 0 \\ a_{m-2} & \iddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix} + \ \ldots + \begin{bmatrix} 0 & \cdots & a_0 \\ \vdots & \iddots & \vdots \\ a_0 & \cdots & 0 \end{bmatrix}$$

$$= \left(a_{m-1}S^{m-1} + a_{m-2}S^{m-2} + \ldots + a_0 S^0\right) J = f^*(S)J.$$

Generating matrix of $\Pi\frac{f(x)-f(y)}{x-y}y^k$ is then $(S^T)^k f^*(S)J$ since the multiplication by y shifts the generating matrix in the same way as the multiplication by S^T. Finally,

$$\frac{y^k - x^k}{y - x} = \sum_{i+j=k-1} x^i y^j$$

and so by substituting $[0, \ldots, 0, 1, 0, \ldots, 0]$ for $[a_0, \ldots, a_{m-k}, \ldots, a_{m-1}]$ in the previous analysis, we see that its generating matrix can be written as $S^{m-k}J$. Next, since the mapping between a bivariate polynomial and its generating matrix is bilinear, the multiplication by $f(y)$ translates to multiplying the generating matrix by $f(S^T)$. Therefore we have the generating matrix of the polynomial $\Pi\left(f(y)\frac{y^k-x^k}{y-x}\right)$ in the form $f(S^T)S^{m-k}J$, which concludes the proof. ■

3.1.2 Properties of Bézout Matrices

To present an algorithm in the next section, we first study useful properties of Bézout matrices. Most of them are based on relations between Bézout and other matrices, like Hankel or Frobenius. In the end, it turns out, that Bézout and Sylvester matrices are very similar in their relation to the GCD. All of the results in this section are adapted from [2].

Theorem 3.5 allows for an alternative construction of $B(f, g)$. Let us have polynomials $f(x) = \sum_{i=0}^{m} a_i x^{m-i}$ and $g(x) = \sum_{i=0}^{n} b_i x^{n-i}$ of degrees m and n respectively, $m \geq n$ as before. It follows from 3.3, that

$$B(f,g) = \\ = \begin{bmatrix} a_{m-1} & \cdots & a_0 \\ \vdots & \iddots & \\ a_0 & & 0 \end{bmatrix} \cdot \begin{bmatrix} b_n & \cdots & b_{n-m+1} \\ & \ddots & \vdots \\ 0 & & b_n \end{bmatrix} - \begin{bmatrix} b_{n-1} & \cdots & b_{n-m} \\ \vdots & \iddots & \\ b_{n-m} & & 0 \end{bmatrix} \cdot \begin{bmatrix} a_m & \cdots & a_1 \\ & \ddots & \vdots \\ 0 & & a_m \end{bmatrix}. \tag{3.4}$$

Where we set $b_i = 0$ whenever it is not a properly defined polynomial coefficient, i.e. when $i > n$ or $i < 0$.

Using the same convention for a_i as well, the relation

$$\begin{bmatrix} a_0 & & & & 0 \\ a_1 & a_0 & & & \\ a_2 & a_1 & a_0 & & \\ \vdots & & \ddots & \ddots & \\ a_{2m-2} & a_{2m-3} & \cdots & a_1 & a_0 \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ h_2 \\ \vdots \\ h_{2m-2} \end{bmatrix} = \begin{bmatrix} b_{n-m+1} \\ b_{n-m+2} \\ b_{n-m+3} \\ \vdots \\ b_{n+m-1} \end{bmatrix}$$

defines entries h_i of so called *Hankel matrix* $H(f,g) = [h_{i,j}]_{i,j=1}^{m}$, where $h_{i,j} = h_{i+j-2}$, i.e.

$$H(f,g) = \begin{bmatrix} h_0 & h_1 & h_2 & \cdots \\ h_1 & h_2 & & \\ h_2 & & \ddots & \\ \vdots & & & h_{2m-2} \end{bmatrix}.$$

Equivalently, we can write

$$\begin{bmatrix} h_0 & & & & 0 \\ h_1 & h_0 & & & \\ h_2 & h_1 & h_0 & & \\ \vdots & & \ddots & \ddots & \\ h_{2m-2} & h_{2m-3} & \cdots & h_1 & h_0 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{2m-2} \end{bmatrix} = \begin{bmatrix} b_{n-m+1} \\ b_{n-m+2} \\ b_{n-m+3} \\ \vdots \\ b_{n+m-1} \end{bmatrix}, \tag{3.5}$$

since that gives us the same system of equations, or we can (for $f \neq 0$) rewrite in the form of a power series (setting $h_i = 0$ where previously undefined)

$$\frac{g(x)}{f(x)} = \sum_{i=0}^{+\infty} h_i x^{-(i+1)}. \tag{3.6}$$

Theorem 3.6. *Let $m > n$ and f, g be coprime polynomials of degree m, n respectively. Then the matrix $H(f,g)$ is nonsingular.*

Conversely, for any given nonsingular $m \times m$ Hankel matrix H, there exist coprime polynomials f, g of degrees $\deg f = m > n = \deg g$, such that f is monic and $H = H(f,g)$.

Moreover, the following relations between f, g and $H(f,g)$ hold:

$$H(f,g) \begin{bmatrix} a_m \\ \vdots \\ a_1 \end{bmatrix} = -a_0 \begin{bmatrix} h_m \\ \vdots \\ h_{2m-1} \end{bmatrix}, \tag{3.7}$$

$$\begin{bmatrix} b_{n-m+1} \\ \vdots \\ b_n \end{bmatrix} = \begin{bmatrix} h_0 & & 0 \\ \vdots & \ddots & \\ h_{m-1} & \cdots & h_0 \end{bmatrix} \begin{bmatrix} a_0 \\ \vdots \\ a_{m-1} \end{bmatrix}. \tag{3.8}$$

Proof. The system (3.8) is just the first m rows of (3.5). Using the next m rows of that very system and remembering, that $a_i = 0$, for $i \geq (m+1)$ and $b_i = 0$, for $i \geq (n+1)$, we have the subsystem

$$\begin{bmatrix} h_m & \cdots & h_1 & h_0 \\ \vdots & \ddots & \vdots & \vdots \\ h_{2m-1} & \cdots & h_m & h_{m-1} \end{bmatrix} \begin{bmatrix} a_0 \\ \vdots \\ a_{m-1} \\ a_m \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix},$$

which can be simply rearranged into (3.7).

Next, assuming H nonsingular means that the system (3.7) has a unique solution $[a_m, \ldots, a_1]$ corresponding to some polynomial $f(x)$ and (3.8) then defines coefficients of the polynomial g, so that $H = H(f, g)$. If these f and g had a common divisor, we could simplify left hand side of (3.6), yielding a different solution of (3.7), contradicting non-singularity of H.

Finally, for the given polynomials $f(x)$ and $g(x)$, having $H(f, g) = H$ singular means there exists more than one solution of (3.7). Let p, q be such a solution, where $\deg(f) \geq \deg(p) > \deg(q)$. From (3.6) follows that

$$\frac{g}{f} = \frac{q}{p}$$

and therefore

$$f = \frac{gp}{q}.$$

Assuming p and q to be coprime, it is necessary for q to divide g and hence there exists a polynomial $r(x)$, $deg(r) > 1$ such that

$$\frac{g}{q} = r, \quad \text{or} \quad g = qr.$$

We have

$$f = \frac{gp}{q} = pr$$

and r is a common divisor of f and g. Conversely, if p and q are not coprime, there exist $\bar{p}$, $\bar{q}$ coprime, such that

$$\frac{g}{f} = \frac{q}{p} = \frac{\bar{q}}{\bar{p}}$$

and we continue as above. In conclusion, we have proven the first statement of our theorem by contraposition. ■

Let F_f be the *Frobenius companion matrix* of a polynomial f, specifically

$$F_f = \begin{bmatrix} 0 & 1 & 0 & \cdots \\ \vdots & \ddots & \ddots & \\ 0 & \cdots & 0 & 1 \\ -a_m & -a_{m-1} & \cdots & -a_1 \end{bmatrix}.$$

The following identities can be found in [10]:

$$B(f, g) = B(f, 1)H(f, g)B(f, 1), \tag{3.9}$$
$$B(f, g) = B(f, 1)g(F_f). \tag{3.10}$$

From (3.4), we have

$$B(f, 1) = \begin{bmatrix} a_{m-1} & \cdots & a_0 \\ \vdots & \iddots & \\ a_0 & & 0 \end{bmatrix}.$$

It is therefore regular and we can rewrite (3.10) as $g(F_f) = B(f,1)^{-1}B(f,g)$ and then, by plugging in from (3.9)

$$g(F_f) = B(f,1)^{-1}B(f,1)H(f,g)B(f,1)$$

and therefore

$$H(f,g) = g(F_f)B(f,1)^{-1}. \tag{3.11}$$

Lemma 3.7. *Having $f(x)$, $g(x)$ as above, with $s(x) = \mathrm{GCD}(f(x), g(x))$, it holds that*

$$g(F_f^T)\boldsymbol{z} = 0 \Leftrightarrow g(x)z(x) = q(x)f(x),$$

where $\boldsymbol{z} = [z_{m-1}, \ldots, z_0]^T$, $z(x) = \sum_{i=0}^{m-1} z_i x^{m-1-i}$ and $q(x)$ is some polynomial. Moreover the matrix $g(F_f^T)$ being singular is equivalent to $s(x)$ being a non-constant polynomial. Finally, if $f(x) = w(x)s(x)$, then

$$\mathrm{Ker}\left(g(F_f^T)\right) = \left\{ [z_{m-1}, \ldots, z_0]^T : \exists\, p(x),\ \sum_{i=0}^{m-1} z_i x^{m-1-i} = p(x)w(x) \right\}$$

and $\dim\left(\mathrm{Ker}\, g(F_f^T)\right) = \deg(s(x))$.

Proof. Firstly, it is verified by direct computation, that $(F_f^T)^i \boldsymbol{e}_1 = \boldsymbol{e}_{i+1}$, for $i = 0, \ldots, (m-1)$ and so $\boldsymbol{z} = z(F_f^T)\boldsymbol{e}_1$ and therefore

$$g(F_f^T)\boldsymbol{z} = g(F_f^T)z(F_f^T)\boldsymbol{e}_1 = (gz)(F_f^T)\boldsymbol{e}_1 = r(F_f^T)\boldsymbol{e}_1,$$

where $r(x) = g(x)z(x) - q(x)f(x)$ for any polynomial $q(x)$, since $f(F_f^T) = 0$, and $\deg(r(x)) < m$. Otherwise, if $\deg(r(x)) \geq m$, we can divide it by $f(x)$ and write $r(x) = h(x)f(x) + k(x)$, where $\deg(k(x)) < m$, and since $f(F_f^T) = 0$ we would set $r(x) := k(x)$.

It follows, that $g(F_f^T)$ is singular if and only if there exist a non-trivial $z(x)$ (or $\boldsymbol{z}$) such that

$$0 = g(F_f^T)\boldsymbol{z} = r(F_f^T)\boldsymbol{e}_1 \ \Leftrightarrow\ r(x) = 0 \ \Leftrightarrow\ 0 = g(x)z(x) - f(x)q(x).$$

Having now $f(x) = w(x)s(x)$ and $g(x) = v(x)s(x)$, $v(x), w(x)$ co-prime, and following the relation $g(x)z(x) = f(x)q(x)$, we obtain $v(x)z(x) = w(x)q(x)$. Now, since $v(x) \nmid w(x)$ we have $z(x) = p(x)w(x)$ for some polynomial $p(x)$, i.e.

$$\boldsymbol{z} \in \mathrm{Ker}\left(g(F_f^T)\right) \Rightarrow z(x) = p(x)w(x).$$

Finally, since $\deg(p) \leq m - 1 - \deg(w)$ we see that $\dim\left(\mathrm{Ker}\left(g(F_f^T)\right)\right)$ is

$$m - 1 - \deg(w) + 1 = \deg(s).$$

■

Let $(\cdot)_k$ denote the leading principal $k \times k$ submatrix, i.e. generally for $A = (a_{ij})_{i,j=1}^m$ we have $A_k = (a_{ij})_{i,j=1}^k$.

Theorem 3.8. *Let polynomials f, g be monic and have the greatest common divisor $s = \mathrm{GCD}(f, g)$, $\deg(s) = m - k$. Then:*

(i) $\mathrm{rank}\,(H(f,g)) = k$, $\det(H_k) \neq 0$ *but* $\det(H_i) = 0$ *for* $i > k$.

(ii) If $H_{k+1}\boldsymbol{w} = 0$, *where* $\boldsymbol{w} = [w_k, \ldots, w_0]^T$, $w_0 = 1$, *then we have* $f = sw$ *where* $w(x) = \sum_{i=0}^{k} w_i x^{k-i}$.

Proof. Let us assume $f = sw$, $g = st$, with t, w coprime polynomials. Following (3.11), we see that $H(f,g)\boldsymbol{z} = 0 \Leftrightarrow g(F_f^T)\boldsymbol{z} = 0$. Writing $\boldsymbol{z} = [z_m, \ldots, z_1]$ and using Lemma 3.7 we see that this is if and only if $z(x) = \sum_{i=1}^{m} z_i x^{m-i} = p(x)w(x)$ for some polynomial $p(x)$. Note that $dim\{z : \ z = pw, \ \deg p \leq n-k-1\} = n-k$ and so $\mathrm{rank}\, g(F_f^T) = n - (n-k) = k$. Setting now $p(x) := 1$ as a particular case, i.e. $z(x) = w(x)$, we are finished. ■

We can now finally provide some basis for computation of GCD via Bézout matrices. Returning back to (3.9), we have

$$JB(f,g)J = JB(f,1)H(f,g)B(f,1)J$$

and so

$$\left[JB(f,g)J\boldsymbol{y} = 0, \ y_{k+1} = 1, \ y_i = 0, \ i > k+1\right] \Leftrightarrow \left[\boldsymbol{w} = B(f,1)J\boldsymbol{y}.\right]$$

Similarly, $\det(JB(f,g)J)_k \neq 0$, while $\det(JB(f,g)J)_i = 0, \ i > k$, with $\boldsymbol{w}$ and $(\cdot)_i$ as above.

This allows us to compute polynomial GCD via the following algorithm.

3.2 Algorithm based on Bézout Matrices

Let us now formulate an algorithm for determining the GCD of two univariate polynomials using Bézout matrices.

Algorithm 3.9. *Let $m > n$ be natural numbers and $f(x) = \sum_{i=0}^{m} a_i x^{m-i}$, $g(x) = \sum_{i=0}^{n} b_i x^{n-i}$ two polynomials. We aim to compute the coefficients $s_0, \ldots, s_k$ of the polynomial $s = gcd(f, g)$.*

I. We first form the matrix $B := B(f,g)$ and compute $k := rank(B)$. This can be done by using Algorithm 1.10 or any other rank revealing algorithm.

II. We set J to be the $m \times m$ antidiagonal matrix

$$J := \begin{bmatrix} 0 & & 1 \\ & \iddots & \\ 1 & & 0 \end{bmatrix},$$

$m \times m$ and $k \times k$ matrices C and C_k as

$$C := JBJ, \ C_k := (C)_k$$

where $(\cdot)_k$ denotes the leading principal $k \times k$ submatrix. We then solve the system $C\boldsymbol{y} = \boldsymbol{b}$ for $\boldsymbol{y} = [y_1, \ldots, y_k]^T$ with $\boldsymbol{b} = [-c_{1,k+1}, \ldots, -c_{k,k+1}]^T$ using notation $C = (c_{i,j})_{i,j=1}^{m}$.

III. We directly compute $[w_k, \ldots, w_0]^T = \boldsymbol{w} = (B(f,1)J)_{k+1} \cdot [y_1, \ldots, y_k, 1]^T$. *As a final step, we only need to find* $s(x)$ *from the relation* $f = s \cdot w$, *where* $w(x) = \sum_{i=0}^{k} w_i x^{k-i}$. *Again, this can be done in more than one way.*

Example 3.10. Let us consider polynomials

$$f(x) = (x-2)(x-1)^2(x+1) = x^4 - 3x^3 + x^2 + 3x - 2 \text{ and}$$
$$g(x) = (x-2)^2(x+2) = x^3 - 2x^2 - 4x + 8.$$

Using the formula (3.4) we calculate

$$B = B(f,g) = \begin{bmatrix} 16 & 4 & -22 & 8 \\ 4 & -20 & 17 & -4 \\ -22 & 17 & 1 & -2 \\ 8 & -4 & -2 & 1 \end{bmatrix}$$

Using Algorithm 1.10, we determine $k = \operatorname{rank}(B) = 3$ and proceed to compute

$$C = JBJ \quad \text{and} \quad C_k = \begin{bmatrix} 1 & -2 & -4 \\ -2 & 1 & 17 \\ -4 & 17 & -20 \end{bmatrix}$$

In the next step we set $\boldsymbol{b} = [-8, 22, -4]^T$, solve $C\boldsymbol{y} = \boldsymbol{b}$ and obtain $y = [8, 4, 2]^T$. Following the algorithm, we compute $[w_3, w_2, w_1, w_0]^T = (B(f,1))_{k+1} [8, 4, 2, 1]^T = [1, -1, -1, 1]$. We have $w(x) = x^3 - x^2 - x + 1$ and all that is left is to compute

$$s(x) = \gcd(f, g) = x - 2.$$

3.3 Multiple Polynomials

In this section, we present a algorithm for computing the GCD of several polynomials based on results published in [4, 9].

Definition 3.11. Let us have $N+1$ polynomials f_0, f_1, …,f_N satisfying $\deg(f_0) = m > \deg(f_i)$ for all $i = 1, \ldots, N$. The *multi-polynomial Bézout matrix* is

$$B_{f_0}(f_1 \ldots, f_N) = \begin{bmatrix} B(f_0, f_1) \\ B(f_0, f_2) \\ \vdots \\ B(f_0, f_N) \end{bmatrix}$$

The following two theorems are found and proven in [9] and show the needed characterization of GCD.

Theorem 3.12. *Let* f_0, f_1, …, f_N *be polynomials,* $\deg(f_0) = m > \deg(f_i) =: n_i$ *for all* $i \in \{1, \ldots, N\}$, *and*

$$f_0(x) = a_0 x^m + a_1 x^{m-1} + \cdots + a_m.$$

Additionally, let

$$h(x) = h_0 x^d + h_1 x^{d-1} + \cdots + h_{d-1} x + h_d$$

be their greatest common divisor of degree d. The columns of the matrix

$$F = \begin{bmatrix} a_0^{n_1} f_1\left(\frac{1}{a_0}F_{f_0}^T\right) \\ a_0^{n_2} f_2\left(\frac{1}{a_0}F_{f_0}^T\right) \\ \vdots \\ a_0^{n_N} f_N\left(\frac{1}{a_0}F_{f_0}^T\right) \end{bmatrix}$$

are denoted by $\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_m$, so that $F = [\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_m]$. If there are $z_{i,j} \in \mathbb{R}$, such that

$$\boldsymbol{\phi}_{m-d+i} = \sum_{j=1}^{m-d} z_{i,j}\boldsymbol{\phi}_j, \quad \forall i \in \{1, \ldots, d\}, \tag{3.12}$$

then

$$a_0 \begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_d \end{bmatrix} = h_0 \begin{bmatrix} a_0 & & & \\ a_1 & a_0 & & \\ \vdots & \vdots & \ddots & \\ a_d & a_{d-1} & \cdots & a_0 \end{bmatrix} \begin{bmatrix} 1 \\ z_{1,m-d} \\ \vdots \\ z_{d,m-d} \end{bmatrix}.$$

Theorem 3.13. *Let us have polynomials $f_0, \ldots, f_N$ and matrix $F = [\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_m]$ as in Theorem 3.12 and let*

$$d = \deg\big(\operatorname{GCD}(f_0, f_1, \ldots, f_N)\big).$$

Then the vectors $\{\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_{m-d}\}$ are linearly independent, while for $i \in \{m-d+1, \ldots, m\}$, the vectors $\boldsymbol{\phi}_i$ can be expressed as a linear combination of vectors from $\{\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_{m-d}\}$.

And finally, the following theorem is an expansion of Theorem 3.4 in [4]

Theorem 3.14. *Let f_0, f_1, …, f_N be polynomials,*

$$\begin{aligned} f_0(x) &= a_0x^m + a_1x^{m-1} + \ldots + a_m, \\ f_1(x) &= b_{1,0}x^{n_1} + b_{1,1}x^{n_1-1} + \ldots + b_{1,n_1}, \\ f_2(x) &= b_{2,0}x^{n_2} + b_{2,1}x^{n_2-1} + \ldots + b_{2,n_1}, \\ &\vdots \\ f_N(x) &= b_{N,0}x^{n_N} + b_{N,1}x^{n_N-1} + \ldots + b_{N,n_N}, \end{aligned}$$

with $a_0 \neq 0$, $\deg(f_0) = m$ and $n_j = \deg(f_j) \leq m-1$ for all $j \in \{1, 2, \ldots, N\}$. Then

$$d := \deg(\operatorname{GCD}(f_0, f_1, \ldots, f_N)) = m - \operatorname{rank}(B_{f_0}(f_1, \ldots, f_N)).$$

If $\boldsymbol{t}_1, \boldsymbol{t}_2, \ldots, \boldsymbol{t}_m$ are the columns of $B_{f_0}(f_1, \ldots, f_N)$ then the last $m-d$ columns $\boldsymbol{t}_{d+1}$, $\boldsymbol{t}_{d+2}$, …$\boldsymbol{t}_m$, are linearly independent. If we denote $T_1 = [\boldsymbol{t}_d, \boldsymbol{t}_{d-1}, \ldots, \boldsymbol{t}_1]$ and $T_2 = [\boldsymbol{t}_m, \boldsymbol{t}_{m-1}, \ldots, \boldsymbol{t}_{d+1}]$, then there exists a matrix $W \in \mathbb{R}^{(m-d)\times d}$,

$$W = \begin{bmatrix} w_d^m & w_{d-1}^m & \cdots & w_1^m \\ w_d^{m-1} & w_{d-1}^{m-1} & \cdots & w_1^{m-1} \\ \vdots & \vdots & & \vdots \\ w_d^{d+1} & w_{d-1}^{d+1} & \cdots & w_1^{d+1} \end{bmatrix}$$

such that

$$T_2 W = T_1,$$

and if we put

$$\begin{bmatrix} h_0 \\ h_1 \\ h_2 \\ \vdots \\ h_d \end{bmatrix} = h_0 \begin{bmatrix} 1 \\ w_d^{(d+1)} \\ w_{d-1}^{(d+1)} \\ \vdots \\ w_1^{(d+1)} \end{bmatrix}, \tag{3.13}$$

then

$$\mathrm{GCD}(f_0, f_1, \ldots, f_N) = h_0 x^d + h_1 x^{d-1} + \cdots + h_{d-1} x + h_d.$$

Proof. The Barnett factorization (see [4], Theorem 3.1.) gives

$$B(f_0, f_i) = f_i(F^T_{f_0/a_0}) B(f_0, 1),\ i = 1, \ldots, N,$$

and so

$$\begin{bmatrix} B(f_0, f_1) \\ \vdots \\ B(f_0, f_N) \end{bmatrix} = \begin{bmatrix} f_1(F^T_{f_0/a_0}) \\ \vdots \\ f_N(F^T_{f_0/a_0}) \end{bmatrix} B(f_0, 1). \tag{3.14}$$

Now, by applying Theorem 3.12 on the polynomials $\frac{1}{a_0} f_0, f_1, \ldots, f_N$, we see that the coefficients of the desired GCD are determined by

$$\begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_d \end{bmatrix} = \frac{h_0}{a_0} \begin{bmatrix} a_0 & & & \\ a_1 & a_0 & & \\ \vdots & \vdots & \ddots & \\ a_d & a_{d-1} & \cdots & a_0 \end{bmatrix} \begin{bmatrix} 1 \\ z_{1,m-d} \\ \vdots \\ z_{d,m-d} \end{bmatrix}. \tag{3.15}$$

We can also rewrite (3.14), using F from Theorem 3.12, as

$$B_{f_0}(f_1, \ldots, f_N) = F \cdot B(f_0, 1).$$

and since

$$\begin{aligned} B_{f_0}(f_1, \ldots, f_N) &= [\boldsymbol{t}_1, \ldots, \boldsymbol{t}_m], \\ F &= [\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_m], \\ B(f_0, 1) &= \begin{bmatrix} a_{m-1} & \cdots & a_0 \\ \vdots & \iddots & \\ a_0 & & 0 \end{bmatrix}, \end{aligned}$$

we obtain:

$$\begin{aligned} \boldsymbol{t}_1 &= \boldsymbol{\phi}_1 a_{m-1} + \cdots + \boldsymbol{\phi}_{m-1} a_1 + \boldsymbol{\phi}_m a_0, \\ \boldsymbol{t}_2 &= \boldsymbol{\phi}_1 a_{m-2} + \cdots + \boldsymbol{\phi}_{m-1} a_0, \\ &\vdots \\ \boldsymbol{t}_m &= \boldsymbol{\phi_1} a_0, \end{aligned}$$

or generally, for $i = 1, \ldots, m$

$$\boldsymbol{t}_i = \sum_{k=1}^{m-i+1} \boldsymbol{\phi}_k a_{m-i-k+1}.$$

Considering Theorem 3.13 and the fact that

$$\begin{aligned}
\boldsymbol{t}_m &\in \operatorname{span}\{\boldsymbol{\phi}_1\}, \\
\boldsymbol{t}_{m-1} &\in \operatorname{span}\{\boldsymbol{\phi}_1,\ \boldsymbol{\phi}_2\}, \\
&\vdots \\
\boldsymbol{t}_1 &\in \operatorname{span}\{\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_m\} \overset{\text{Theorem 3.13}}{=} \operatorname{span}\{\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_{m-d}\},
\end{aligned}$$

the vectors $\{t_{d+1}, \ldots, t_m\}$ have to be linearly independent and all of the vectors $t_1, \ldots, t_d$ are linear combinations of the vectors from $\{t_{d+1}, \ldots, t_m\}$. Hence, there exist numbers $w_i^j \in \mathbb{R},\ i = 1, \ldots, d,\ j = d+1, \ldots, m$, such that

$$\boldsymbol{t}_i = \sum_{j=d+1}^{m} w_i^j \boldsymbol{t}_j.$$

Plugging now in for $\boldsymbol{t}_i$ gets us

$$\sum_{k=1}^{m-i+1} \boldsymbol{\phi}_k a_{m-i-k+1} = \sum_{j=d+1}^{m} w_i^j \boldsymbol{t}_j$$

and the same for $\boldsymbol{t}_j$

$$\begin{aligned}
\sum_{k=1}^{m-i+1} \boldsymbol{\phi}_k a_{m-i-k+1} &= \sum_{j=d+1}^{m} w_i^j \left(\sum_{k=1}^{m-j+1} \boldsymbol{\phi}_k a_{m-j-k+1} \right), \\
\sum_{k=1}^{m-d} \boldsymbol{\phi}_k a_{m-i-k+1} + \sum_{k=m-d+1}^{m-i+1} \boldsymbol{\phi}_k a_{m-i-k+1} &= \sum_{j=d+1}^{m} w_i^j \left(\sum_{k=1}^{m-j+1} \boldsymbol{\phi}_k a_{m-j-k+1} \right).
\end{aligned}$$

From the relation (3.12), we have for any $k \in \{m-d+1, \ldots, m\}$

$$\boldsymbol{\phi}_k = \sum_{j=1}^{m-d} z_{k+d-m,j} \boldsymbol{\phi}_j$$

and using that, we obtain

$$\sum_{k=1}^{m-d} \boldsymbol{\phi}_k a_{m-i-k+1} + \sum_{k=m-d+1}^{m-i+1} a_{m-i-k+1} \left(\sum_{j=1}^{m-d} z_{k+d-m,j} \boldsymbol{\phi}_j \right) = \sum_{j=d+1}^{m} w_i^j \left(\sum_{k=1}^{m-j+1} \boldsymbol{\phi}_k a_{m-j-k+1} \right).$$

We shall now study the coefficients of $\boldsymbol{\phi}_{m-d}$. Let us start with the right hand side of this identity. Since we are interested in $\boldsymbol{\phi}_{m-d}$, we have $k = m-d$, which is only possible for $j = d+1$. Therefore the coefficient is $w_i^{d+1} a_{m-j-k+1} = w_i^{d+1} a_{m-d-1-m+d+1} = w_i^{d+1} a_0$.

Next, the left hand side. In the first sum, the situation is quite clear, we only get a_{d-i+1}. In the second (double) sum, the only terms interesting to us are the

ones with $j = m - d$, while the outer summation remains unchanged. Altogether, we have

$$a_{d-i+1} + a_{d-i} z_{1,m-d} + \cdots + a_0 z_{d-i+1}$$

as a coefficient on the left hand side. By comparing both sides of the identity, we finally obtain

$$\begin{aligned} w_i^{d+1} a_0 &= a_{d-i+1} + a_{d-i} z_{1,m-d} + \cdots + a_0 z_{d-i+1,m-d}, \\ w_i^{d+1} &= \frac{1}{a_0} \left(a_{d-i+1} + a_{d-i} z_{1,m-d} + \cdots + a_0 z_{d-i+1,m-d} \right). \end{aligned}$$

This also means that $w_i^{d+1},\ i \in \{1, \ldots, d\}$ satisfy the following:

$$h_0 \begin{bmatrix} 1 \\ w_d^{(d+1)} \\ w_{d-1}^{(d+1)} \\ \vdots \\ w_1^{(d+1)} \end{bmatrix} = \frac{h_0}{a_0} \begin{bmatrix} a_0 & & & \\ a_1 & a_0 & & \\ \vdots & \vdots & \ddots & \\ a_d & a_{d-1} & \cdots & a_0 \end{bmatrix} \begin{bmatrix} 1 \\ z_{1,m-d} \\ \vdots \\ z_{d,m-d} \end{bmatrix}.$$

By simply comparing this to (3.15), we see that (3.13) in fact describes the coefficient of the GCD, which concludes the proof. ■

We present an algorithm based on these results. Note, that we actually do not need to fully solve the system $T_2 W = T_1$ in order to obtain $w_i^{d+1},\ i = 1, \ldots, d$. Since it is just one row of W that we seek, only QR decomposition and one step of backward substitution is required, as it is done in the following algorithm.

Algorithm 3.15. *Let us have polynomials f_0, f_1,..., f_N of degrees m, n_1,..., n_N respectively, with additional assumption $m > \max\{n_1, \ldots, n_N\}$. We are computing the polynomial $h = \mathrm{AGCD}(f_0, f_1, \ldots, f_N)$.*

I. Determine $d = m - \mathrm{rank}(B_{f_0}(f_1, \ldots, f_N))$ using any kind of rank revealing algorithm, e.g. Algorithm 1.10.

II. Let $\boldsymbol{t}_1, \ldots, \boldsymbol{t}_m$ be column vectors of $B_{f_0}(f_1, \ldots, f_N) = [\boldsymbol{t}_1, \ldots, \boldsymbol{t}_m]$.

III. Construct $T_2 = [\boldsymbol{t}_m, \boldsymbol{t}_{m-1}, \ldots, \boldsymbol{t}_{d+1}]$ and $T_1 = [\boldsymbol{t}_d, \boldsymbol{t}_{d-1}, \ldots, \boldsymbol{t}_1]$.

IV. Calculate QR decomposition of T_2, i.e. $T_2 = QR$, where $Q \in \mathbb{R}^{m \times m}$ is orthogonal and $R \in \mathbb{R}^{m \times (m-d)}$ is an upper triangular matrix.

V. We set $r := (R)_{m-d,m-d}^{-1}$ and compute $w_i^{d+1} = r \left(Q^T T_1 \right)_{m-d,i}$, for $i = d, \ldots, 1$.

VI. Setting $h_i := w_{d-i+1}^{d+1},\ i = 1, \ldots, d$ and $h_0 := 1$, we finally have $h(x) = h_0 x^d + h_1 x^{d-1} + \ldots + h_{d-1} x + h_d = \mathrm{GCD}(f_0, f_1, \ldots, f_N)$.

End of algorithm.

4. Numerical Experiments

There are several algorithms presented in this thesis, although only Algorithms 2.5, 2.7, 3.9 and 3.15 are of any interest to us. In this chapter, these four algorithms are tested on sample problems of varying size and difficulty.

Firstly, the practical aspects of the computation are shortly discussed. This includes implementation as well as possible preprocessing.

Section considering finding the GCD of two polynomials follows. In that section, all four of the algorithms can be compared, since the multiple polynomial algorithms do not exactly reduce to their two polynomial counterparts.

The next section then naturally concerns itself with problems comprising several polynomials, where only Algorithms 2.7 and 3.15 are applicable.

4.1 Computation

All the algorithms were implemented as functions in MATLAB. Attached to this thesis is a CD containing all the code used.

In our sample problems we shall quite inaccurately use the term AGCD to refer to whatever result is computed by any of the algorithm and the term GCD as the precise greatest common divisor. The way the problems are set up, we always know the GCD beforehand, so that we can later measure error in our computation as

$$e = |AGCD - GCD|,$$

where the symbol $|\cdot|$ denotes a polynomial norm defined for a polynomial $f(x) = \sum_{i=0}^{m} a_i x^{m-i}$ as

$$|f| = \frac{\sqrt{\sum_{i=0}^{m} a_i^2}}{m}.$$

Since scaling of the input polynomials does not affect the resulting GCD, all the polynomials can be normalized to prevent big differences in coefficient values. In our case, we use geometric mean of coefficients, so the polynomial $f(x) = \sum_{i=0}^{m} a_i x^{m-i}$ would be rescaled as follows:

$$f(x) \quad \rightarrow \quad \left(\prod_{i=0}^{m} a_i \right)^{\frac{1}{m}} f(x).$$

4.2 Two polynomials

Example 4.1. Let us consider the following polynomials:

$$f(x) = (x - 0.5)^5 (x + 0.4)^6 (x - 2)^8 (x + 2)^3,$$
$$g(x) = (x - 0.5)^3 (x + 0.4)^3 (x - 2)^3 (x + 3)^3,$$

then clearly, $\mathrm{GCD}(f,g) = (x-0.5)^3(x+0.4)^3(x-2)^3$. We employ all four of the presented algorithms and compare errors in the results as shown in Table 4.1. As expected, Algorithm 2.7 provides much more accurate result, than Algorithm 2.5, since it has an additional refining process. We also note that both of the algorithms based on Bézout matrices performed with a rather poor accuracy by comparison.

This may be caused by the fact, that Bézout matrices are defined using subtractions. This can possibly lead to the effect of so called cancelation and overall decrease in accuracy. The Sylvester matrices have a drawback in that they are essentially twice as big, but that does not seem to matter, at least in this case.

	Algorithm 2.5	Algorithm 3.9	Algorithm 2.7	Algorithm 3.15
e	3.0×10^{-11}	5.3×10^{-5}	1.3×10^{-16}	8.4×10^{-8}

Table 4.1: The error e of the AGCD computed by different algorithms in Example 4.1.

The theory showed, that determining the GCD is based on determining a numerical rank of some specific matrix. Figure 4.1 illustrates this very well; singular values of $S(f,g)$ and $B(f,g)$, which have significant role in our algorithms, are plotted We see 9 singular values are under the set tolerance of 1×10^{-10} in both cases, corresponding to $\deg(GCD(f,g)) = 9$.

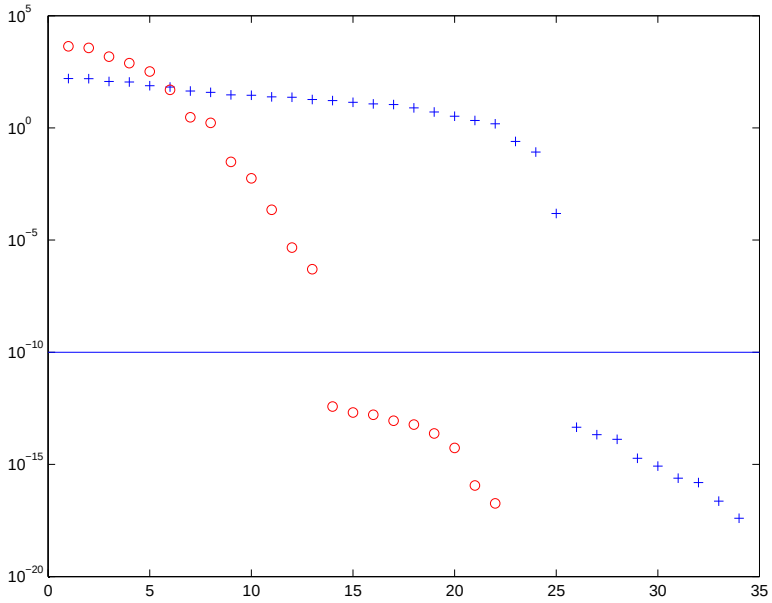


Figure 4.1: Singular values of $B(f,g)$ as $\circ$ and $S(f,g)$ as $+$. The horizontal line specifies the set tolerance 1×10^{-10}.

Example 4.2. We shall now consider polynomials

$$f(x) = (x+2)^4(x+1.5)^3(x+1)^3(x-5)^7,$$
$$g(x) = (x+3)^3(x+1.5)^4(x-5)^5.$$

Their GCD $= (x+1.5)^3(x-5)^5$ and their roots are slightly clustered, i.e. there are two subsets of closer roots, although this effect is not really prominent here.

Again, their GCD is computed by all four algorithms and errors are compared in Table 4.2. The trends set in Example 4.1 seem to continue, only this time the

	Algorithm 2.5	Algorithm 3.9	Algorithm 2.7	Algorithm 3.15
e	7.8×10^{-4}	1.9×10^{3}	1.6×10^{-7}	7.2×10^{0}

Table 4.2: The error e of the AGCD computed by different algorithms in Example 4.2.

Bézout based algorithms provide results so inaccurate, that they are practically useless. Figure 4.2 again presents the singular values of $S(f, g)$ and $B(f, g)$.

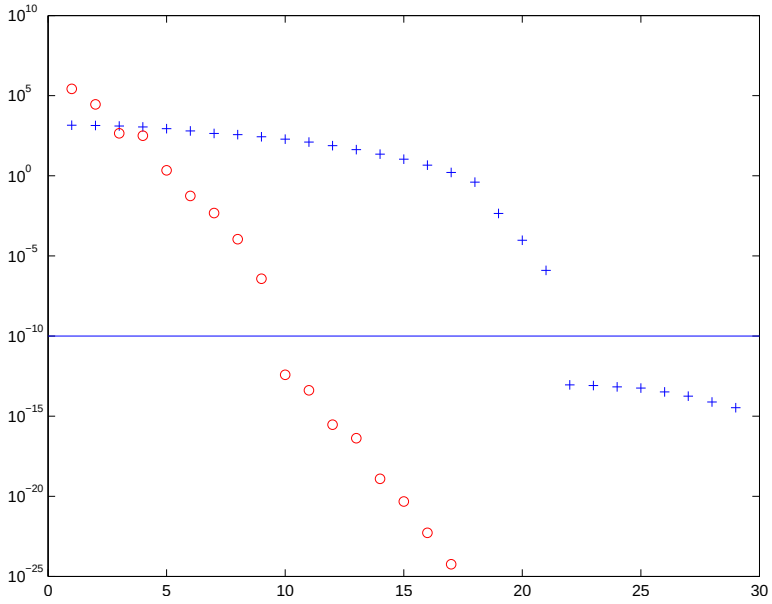


Figure 4.2: Singular values of $B(f, g)$ as $\circ$ and $S(f, g)$ as $+$ in Example 4.2. The horizontal line specifies the set tolerance 1×10^{-10}.

Example 4.3. Let us have $m \in \mathbb{N}$ even, so $m = 2k$ for some $k \in \mathbb{N}$. The following polynomials with real coefficients but complex roots are presented in [19]. Set $\alpha_i = \cos\left(\frac{\pi i}{m}\right)$, $\beta_i = \sin\left(\frac{\pi i}{m}\right)$, for $i = 1, \ldots, m$. Moreover, let $r_1 = 0.5$ and $r_2 = 1.5$ and define

$$f(x) = \prod_{i=1}^{k} \left((x - r_1\alpha_i)^2 + r_1^2\beta_i^2\right) \prod_{i=k+1}^{m} \left((x - r_2\alpha_i)^2 + r_2^2\beta_i^2\right),$$

$$g(x) = \prod_{i=1}^{m} \left((x - r_1\alpha_i)^2 + r_1^2\beta_i^2\right).$$

The roots of these polynomials are distributed over two concentric circles in complex plane. The values $m = 12$ and $m = 16$ are tested. Note that the more roots (or higher degree) we have, the closer the roots are. As usual, all four algorithms are used to compute the greatest common divisor for both $m = 12$ and $m = 16$. The results are summarized in Table 4.3. While for $m = 12$, the problem does not seem to cause any troubles, for $m = 16$ both Bézout based algorithms fail. This is most likely caused by the closeness of the roots.

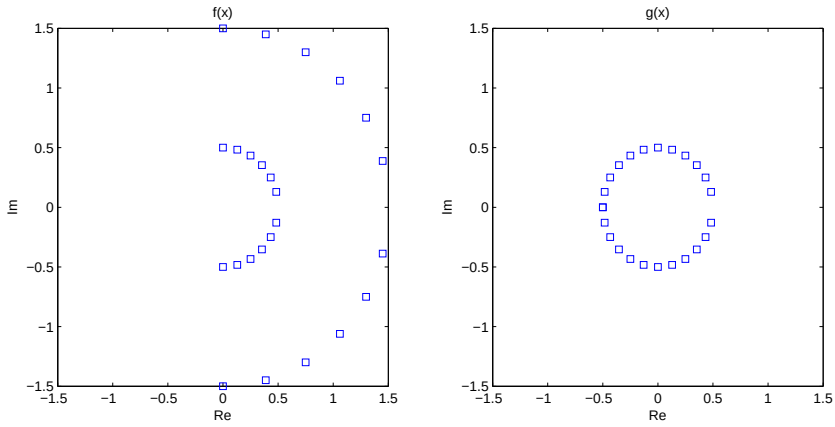


Figure 4.3: Roots of the polynomial $f(x)$ (left) and $g(x)$ (right) from Example 4.3 for $m = 12$.

	Algorithm 2.5	Algorithm 3.9	Algorithm 2.7	Algorithm 3.15
e for $m = 12$	1.7×10^{-12}	5.5×10^{-13}	9.7×10^{-14}	2.6×10^{-12}
e for $m = 16$	1.9×10^{-10}	$*$	1.0×10^{-8}	$*$

Table 4.3: The error e of the AGCD computed by different algorithms in Example 4.3 for $m = 12$ and $m = 16$. The symbol $*$ signifies, that the degree of the GCD was determined incorrectly and therefore the algorithm does not provide any useful solution.

4.3 Multiple polynomials

Example 4.4. In the case of multiple polynomials, a new issue arises. In Algorithm 2.7 , we can select the order in which the first phase processes the given polynomials. This example is to demonstrate, that this affects the results greatly. We consider these polynomials:

$$\begin{aligned}
f_0(x) &= (x-0.9)^5(x-0.8)^5(x-0.7)^5(x+0.7)^5(x+0.5)^5(x-10)^5,\\
f_1(x) &= (x-0.9)^4(x-0.8)^4(x+0.5)^4(x-2)^4(x-6)^4,\\
f_2(x) &= (x-3)^3(x-0.8)^3(x+0.5)^3(x+2)^3,\\
f_3(x) &= (x-1)^3(x-0.8)^3(x+0.5)^3(x+2)^3.
\end{aligned}$$

Since the ordering is not really possible in Algorithm 3.15, we shall only concern ourselves with computation via Algorithm 2.7 . If we carry out the computation in this default order f_0, f_1, f_2, f_3 the algorithm fails. But computing in the reverse order, i.e. f_3, f_2, f_1, f_0, the calculation succeeds with an error of only 1.3×10^{-16}. Figure 4.4 shows the singular values of matrices

$$\begin{aligned}
S(f_0, f_1)\ldots+,\\
S(\mathrm{AGCD}(f_0, f_1), f_2)\ldots\circ,\\
S(\mathrm{AGCD}(\mathrm{AGCD}(f_0, f_1), f_2), f_3)\ldots\triangledown,
\end{aligned}$$

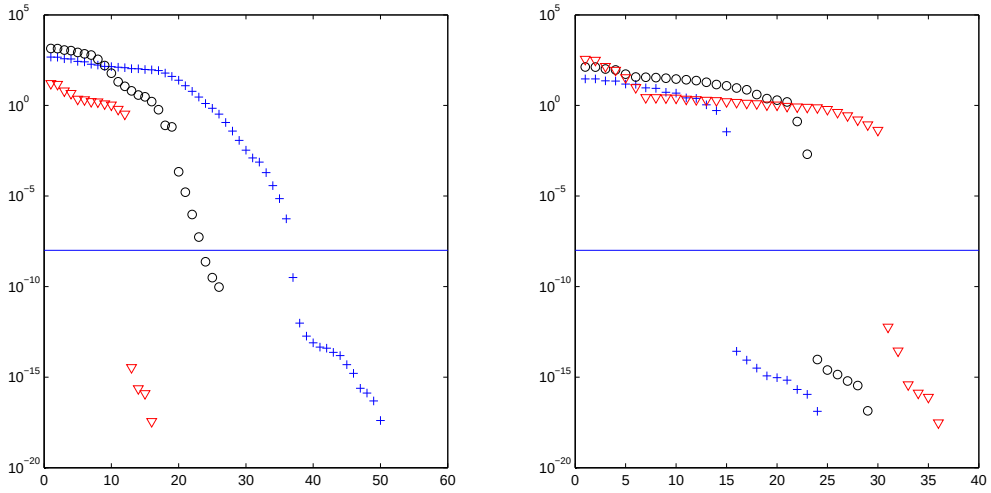


Figure 4.4: Singular values of Sylvester matrices appearing in the first phase of Algorithm 2.7 ; in order $+$, $\circ$, $\triangledown$; polynomials ordered f_0, f_1, f_2, f_3 (left) and f_3, f_2, f_1, f_0 (right).

on the left hand side and in reversed order, i.e.

$$\begin{aligned} S(f_3, f_2)\ldots+,\\ S(\mathrm{AGCD}(f_3, f_2), f_1)\ldots\circ,\\ S(\mathrm{AGCD}(\mathrm{AGCD}(f_3, f_2), f_1), f_0)\ldots\triangledown, \end{aligned}$$

on the right hand side. We can see how much more distinct the approxi-null values are in the case of the reversed ordering. The main point is, that we start working with the polynomials of the smaller degrees first. That way, the size of our problem is have reduced and that makes for easier computation.

Example 4.5. To compare the two presented algorithms, let us now have the following polynomials:

$$\begin{aligned} f_0 &= (x-0.9)^5(x-0.8)^5(x-0.7)^5(x+0.3)^5(x+0.5)^5(x+0.7)^5,\\ f_1 &= (x-2)^5(x-0.9)^5(x-0.8)^5(x+0.5)^5(x+2)^5,\\ f_2 &= (x-3)^5(x-0.8)^5(x+0.5)^5(x+2)^5 \text{ and}\\ f_3 &= (x-0.8)^4(x+0.5)^4. \end{aligned}$$

It is easily seen, that $\mathrm{GCD}(f_0, \ldots, f_3) = f_3$.

Accuracy of these computations is shown in Table 4.4. The errors made in determining the coefficients are about two orders of magnitude smaller in case of Algorithm 2.7 than in the case of Algorithm 3.15, which continues the trend set by the previous examples.

In conclusion, it seems, that the Sylvester matrix based algorithms provide better accuracy.

Coefficients	Error in coefficients	
GCD	Algorithm 2.7	Algorithm 3.15
1.0000	0.0000e+00	0.0000e+00
-1.2000	-9.1038e-15	1.8050e-12
-1.0600	-6.8834e-15	-7.6916e-13
1.3320	2.6645e-15	-2.6426e-12
0.5361	1.2212e-15	3.3151e-13
-0.5328	-2.6645e-15	1.3358e-12
-0.1696	-2.0539e-15	1.1419e-13
0.0768	-7.2164e-16	-2.3732e-13
0.0256	-3.8164e-16	-5.7697e-14

Table 4.4: Comparison of computational error in AGCD coefficients produced by Algorithm 2.7 and Algorithm 3.15.

Conclusion

In this thesis, we have presented theoretic background as well as algorithms needed to compute the greatest common divisor of two polynomials via transformations of the Sylvester matrices. We have then gone further and studied a generalization of such an approach for multiple polynomials. Moreover, we tried to use this generalization in the original two polynomial problem and found, that it yields rather accurate results.

To compare the accuracy of our Sylvester based algorithms, we concerned ourselves with their Bézout based analogues. Indeed, it turned out, that the Bézout matrices share some properties with the Sylvester ones. It also turned out, that the Bézout matrices can be used to find the GCD of multiple polynomials as well. Even though the accuracy or robustness of this approach does not seem to be all that good, it fulfilled it's role as a comparison and alternative resultant matrix.

Bibliography

[1] Barnett, Stephen. *Polynomials and linear control systems.* New York: M. Dekker, 1983. ISBN 08-247-1898-4.

[2] Bini, Dario and Pan, Victor Y. *Polynomial and Matrix Computation, vol. 1 Fundamental Algorithms.* Boston: Birkhäuser, 1994. ISBN 3-7643-3786-9

[3] Björck, Åke. *Numerical Methods for Least Square Problems.* 1st ed. Philadelphia: SIAM, 1996. ISBN 08-987-1360-9.

[4] Diaz-Toca, Gema M. and Gonzáles-Vega, Laureano. Barnett's Theorems About the Greatest Common Divisor of Several Univariate Polynomials Through Bezout-like Matrices. *Journal of Symbolic Computation.* 2002, vol. 34, issue 1, pp. 59-81. DOI: 10.1006/jsco.2002.0542.

[5] Eliaš, Ján. *Approximate Polynomial Greatest Common Divisor.* Prague, 2012. Master thesis. Charles University. Faculty of Mathematics and Physics.

[6] Eliaš, Ján. *Problémy spojené s výpočtem největšího společného dělitele.* Prague, 2009. Bachelor thesis. Charles University. Faculty of Mathematics and Physics.

[7] Eliaš, Ján and Zítko, Jan. Approximate polynomial GCD. *Programs and algorithms of numerical Mathematics 16: Proceedings of seminar.* Praha: Institute of Mathematics, Academy of Sciences of the Czech Republic, 2013, pp. 63-68.

[8] Golub, Gene H. and Van Loan, Charles F. *Matrix Computations.* 3rd Ed. Baltimore: John Hopkins University Press, 1996. ISBN 08-018-5414-8.

[9] Gonzáles-Vega, Laureano. An elementary proof of Barnett's theorem about the greatest common divisor of several univarlate polynomials. *Linear Algebra and its Applications.* 1996, vol. 247, pp. 185-202. DOI: 10.1016/0024-3795(95)00099-2.

[10] Helmke, Uwe R. and Fuhrmann, Paul A. Bezoutians. *Linear Algebra and its Applications.* 1989, 122-124, pp. 1039-1097. DOI: 10.1016/0024-3795(89)90684-8.

[11] Kaltofen, Erich, Yang, Zhengfeng and Zhi, Lihong. Structured Low Rank Approximation of a Sylvester Matrix. *Symbolic-Numeric Computation.* Basel: Birkhäuser Basel, 2007, pp. 69-83. DOI: 10.1007/978-3-7643-7984-1_5.

[12] Li, Tien-Yien and Zeng, Zhonggang. A Rank-Revealing Method with Updating, Downdating, and Applications. *SIAM Journal on Matrix Analysis and Applications.* 2005, vol. 26, issue 4, pp. 9178-946. DOI: 10.1137/S0895479803435282.

[13] Lukšan, Ladislav. *Numerické optimalizační metody.* Prague : ICS AS CR, 2005. Available at: http://hdl.handle.net/11104/0134392.

[14] Pták, Vlastimil. Explicit Expressions for Bezoutians. *Linear Algebra and its Applications.* Firenze: Tipografia Raccini, 1984, vol. 59, pp. 43-54. DOI: 10.1016/0024-3795(84)90157-5.

[15] Winkler, Joab R. *Polynomial roots and approximate common divisor.* Technical report. Oxford: The University of Oxford, 2007. Lecture notes for summer school at the computer laboratory, The University of Oxford.

[16] Winkler, Joab R. and Allan John D. Structured total least norm and approximate GCDs of inexact polynomials. *Journal of Computational and Applied Mathematics.* 2008, vol. 215, issue 1, pp. 1-13. DOI: 10.1016/j.cam.2007.03.018.

[17] Winkler, Joab R., Hasan, Madina and Lao, Xin. Two methods for the calculation of the degree of an approximate greatest common divisor of two inexact polynomials. *Calcolo.* 2012, vol. 49, issue 4, pp. 241-267. DOI: 10.1007/s10092-012-0053-5.

[18] Winkler, Joab R. and Zítko, Jan. Some questions associated with the calculation of the GCD of two univariate polynomials. *Seminar on numerical analysis SNA'07*, pp. 130-137. Ostrava: Institute of Geonics AS CR. 2007.

[19] Zeng, Zhonggang. The Approximate GCD of Inexact Polynomials, Part I: a Univariate Algorithm. [cit. 10.07.2014]. Available at: http://orion.neiu.edu/ zzeng/uvgcd.pdf

[20] Zítko, Jan. Nový pohled na numerický výpočet největšího společného dělitele dvou polynomů. *Pokroky matematiky, fyziky a astronomie.* 2010, vol. 55, issue 3, pp. 231-242.

[21] Zítko, Jan and Eliaš, Ján. Application of the rank revealing algorithm for the calculation of the GCD. *Seminar on numerical analysis SNA '12*, pp. 175-180. Liberec : Technical university of Liberec, 2012.