

# Università di Pisa

Dipartimento di Matematica
Corso di Laurea in Matematica


Tesi di Laurea Magistrale

# Computing the common roots of two bivariate polynomials via the Chebyshev-Bézout resultant


Candidato:
Miryam Gnazzo

Relatori:
Prof. Paola Boito
Dott. Leonardo Robol

Controrelatore:
Prof. Luca Gemignani



Anno Accademico 2019/2020

# Contents

# Introduction

The multivariate polynomial rootfinding problem consists in detecting the common zeros of two or more multivariate polynomials, that is, computing numerically all the solutions of the system

$$\begin{cases} p_1(x_1, x_2, \ldots, x_d) = 0 \\ p_2(x_1, x_2, \ldots, x_d) = 0 \\ \qquad\qquad \vdots \\ p_d(x_1, x_2, \ldots, x_d) = 0 \end{cases}, \qquad (x_1, x_2, \ldots, x_d) \in \mathbb{R}^d, \tag{1}$$

where $d \geq 2$ and $p_1, p_2, \ldots, p_d$ are polynomials with real coefficients. Throughout this work, we assume that the solution set is zero-dimensional, that is, the solutions are isolated. This hypothesis implies that the number of solutions is finite. The problem of finding the common zeros of two or more multivariate functions $f_1, f_2, \ldots, f_d$ can be treated in a similar way, approximating the functions with their polynomial interpolants.

Rootfinding problems arise in several applications. For instance, the design of chemical processes can require the solution of systems of nonlinear equations [52]. Moreover, rootfinding problems often arise in computer graphics and computer aided design (CAD). Parametric and algebraic curves and surfaces are fundamental tools of solid modeling and the problem of computing their intersections can be solved as a rootfinding problem [5, 34]. Multivariate rootfinding can be employed in order to solve optimization problems. For instance, the problem of minimizing an objective function can be converted in solving a system of multivariate functions. Finding the global minimum of an oscillatory function was one of the SIAM 100-Digit Challenge problems [49], and Chebfun2, a MATLAB package for dealing with smooth functions, was employed in order to solve it. The package Chebfun uses an approach based on the interpolation of smooth functions on the Chebyshev points and expresses the polynomial interpolants in the Chebyshev basis [50]. In particular, its command `roots` solves the bidimensional rootfinding and one possible application can be finding the global minimum of bivariate functions.

Many techniques have been proposed in order to solve this problem. These include a class of algorithms known as resultant-based methods. The main idea of this approach consists in converting the problem of rootfinding in two or more eigenvalue problems; this is done by associating a multidimensional resultant matrix with the rootfinding problem (1). Several different types of resultant matrices have been proposed in the literature, including Sylvester [13], Cayley [38] and Macaulay matrices [30]. A resultant-based method is employed in the command `roots` of Chebfun2, in order to compute the common roots of

two functions [36].

Another class of numerical rootfinding methods is based on homotopy continuation [43]. These methods construct a homotopy between the original polynomial system in the form (1) and another one, which is chosen arbitrarily and is easy to solve. The inverse homotopy is then applied to the computed solutions to recover the ones of the original problem. The challenging point of this method consists in accurately tracking the original points to the final points along a continuous path. Recent studies on this approach propose a robust strategy for path tracking [46]. This class of methods is mostly used for rootfinding problems involving small degree polynomials. One drawback of homotopy continuation consists in the difficulties in the study of the conditioning of the problem. For instance, the deformation of the original problem leads to intermediate steps whose conditioning is difficult to control in advance.

In closing, we also mention approaches based on Gröbner bases, generally used for small degree polynomial systems, mostly in a symbolic context [7]. The use of Gröbner bases could lead to instability issues that are not relevant in a symbolic representation, but become significant in the numerical setting.

In this thesis, we focus on bivariate rootfinding, that is, computing the common roots of two bivariate polynomials $p(x, y)$ and $q(x, y)$. We are interested in finding all the solutions of the system (1) where the pair $(x, y)$ is real and belongs to $[-1, 1]^2$. By change of variable, this allows to consider roots in any box $[a, b] \times [c, d]$.

While the monomial basis is a very common choice for a basis of polynomials, we prefer to express the polynomial in the Chebyshev basis, since it is better suited for numerical applications. In order to solve the rootfinding problem, we rely on the resultant-based method in combination with a hidden variable technique, as proposed in [36]. Among the different resultant matrices, we choose the Bézout resultant matrix over the more widely used Sylvester matrix. This choice is motivated by the Vandermonde form of the eigenvectors of the Bézout matrix, which simplifies the study of the conditioning. Moreover, the Bézout matrix is easier to generalize to the multivariate case, involving three or more polynomials. This generalization is known as the Cayley matrix [38].

Using the hidden variable technique, the Bézout matrix can be expressed as a matrix polynomial in the variable $y$, denoted as $B(y)$, as explained in Section 2.1. The efficient construction of this resultant matrix is a non-trivial step, for which we analyze two possible approaches. The first technique exploits the more general connection between matrix polynomials and a class of linearizations, known in the literature as double ansatz space $\mathbb{DL}$ [37]. The second approach relies on a three-dimensional interpolation on the grid of Chebyshev-Lobatto points.

The core of the resultant-based method consists in converting the rootfinding problem into an eigenvalue problem associated with the matrix polynomial $B(y)$. In order to solve this eigenvalue problem, we employ the colleague pencil, which is a companion-like matrix pencil designed for matrix polynomials expressed in the Chebyshev basis [1]. We develop an estimate of the forward error associated with each computed eigenvalue of the matrix polynomial $B(y)$. To this aim, we prove some results based on the work of Tisseur [47] that yield disk of inclusion for each approximate eigenvalue; this is achieved by combining bounds for the backward error and the condition number of the eigenvalue problem.

This thesis is organized as follows. In Chapter 1, we provide an introduction about the Chebyshev basis. In particular, we describe its main properties and algorithms for the evaluation of Chebyshev polynomials, such as the Clenshaw algorithm. We also introduce the problem of interpolation, pointing out the advantages of using the Chebyshev points from a numerical stability viewpoint. In Chapter 2, we present the resultant-based method in combination with the hidden variable technique. We introduce two resultant matrices for the bidimensional case, the Sylvester and the Bézout matrices, and we point out the main differences between them. In addition, we analyze the Cayley matrix, which is a generalization of the Bézout matrix to the multidimensional case. The definition of this matrix requires the use of a tensor-based formulation and of tensor unfoldings. In Chapter 3 we focus on the bivariate case of the rootfinding problem. In particular, we propose a resultant method, using the Bézout matrix. Then, we analyze two possible approaches for the construction of this resultant matrix: the first one is based on the theory of double ansatz spaces $\mathbb{DL}$, presented in [37], while the second one employs a three-dimensional interpolation approach. In Chapter 4, we propose an upper bound for the forward error of each approximate eigenvalue of the resultant matrix, employing estimates for the backward error and the condition number of the eigenvalues. Moreover, we analyze the condition number of the rootfinding problem and possible loss of accuracy of the resultant-based method. In Chapter 5, we propose several numerical experiments performed in MATLAB, in order to test our implementation of the resultant method, in combination with three-dimensional interpolation.

# Chapter 1

# Chebyshev polynomials

Let $n$ be a positive integer and denote by $\mathcal{P}_n$ the vector space of univariate polynomials of degree at most $n$. The monomial basis is a very common choice for a basis of $\mathcal{P}_n$; however other bases have better stability properties on the domain $[-1, 1]$ and are better suited for numerical applications. Throughout this thesis, we prefer to use the Chebyshev basis in order to express univariate and multivariate polynomials. For this reason, we provide several useful results about this set of polynomials [4, 50].

**Definition 1.0.1.** Let $k \geq 0$ be an integer. The $k$-th *Chebyshev polynomial of the first kind* is defined as:

$$T_k(x) = \cos(k(\cos^{-1} x)), \quad \text{for } x \in [-1, 1]\,. \tag{1.1}$$

From the previous definition, it is not immediately clear that $T_k$ is a polynomial. Consider a natural number $k$: we can equivalently write the definition (1.1)(of the $k$-th Chebyshev polynomial of the first kind equivalently) as follows:

$$T_k(x) = T_k(\cos(\theta)) = \cos(k\theta)\,, \quad \text{for } \theta \in [0, \pi]\,. \tag{1.2}$$

Then, we prove by induction on $k$ that $T_k(x)$ is a polynomial in the variable $x$, for every $k \in \mathbb{N}$. In fact, for $k = 0, 1, 2$ we have that

$$\begin{aligned}
T_0(x) &= T_0(\cos(\theta)) = \cos(0 \cdot \theta) = 1 \\
T_1(x) &= T_1(\cos(\theta)) = \cos(1 \cdot \theta) = x \\
T_2(x) &= T_2(\cos(\theta)) = \cos(2 \cdot \theta) = 2\cos^2(\theta) - 1 = 2x^2 - 1.
\end{aligned}$$

For the induction step consider $k \geq 2$. Let us prove that $T_{k+1}(x)$ is a polynomial. Using the trigonometric formulas for product and sum, it holds that

$$\cos((k+1)\,\theta) + \cos((k-1)\,\theta) = 2\cos(\theta)\cos(k\theta). \tag{1.3}$$

Hence, $T_{k+1}$ can be written using this relation and we have

$$\begin{aligned}
T_{k+1}(x) = \cos(k + 1(\theta)) &= 2\cos(\theta)\cos(k\theta) - \cos((k-1)\,\theta) \\
&= 2xT_k(x) - T_{k-1}(x),
\end{aligned}$$

where $T_k$ and $T_{k-1}$ are polynomials, by inductive hypothesis.

Moreover, it can be proved that the set $\{T_0(x), T_1(x), \ldots\}$ of the Chebyshev polynomials is a family of orthogonal polyomials. Let us recall the definition of a family of *orthogonal polynomials*. We denote by $\mathcal{P}$ the vector space of univariate polynomials with real coefficients.

**Definition 1.0.2.** Let $\langle \cdot, \cdot \rangle$ be a scalar product on $\mathcal{P}$. A family of polynomials $\{p_i(x) \in \mathcal{P}_i,\ i = 0, 1, \ldots\}$ such that $\deg(p_i) = i$ for $i = 0, 1, \ldots$ and $\langle p_i, p_j \rangle = 0$ if $i \neq j$ is called a set of *orthogonal polynomials* with respect to the scalar product $\langle \cdot, \cdot \rangle$.

Let $a < b$ be two constants in $\bar{\mathbb{R}} := \mathbb{R} \cup \{+\infty, -\infty\}$ and let $\omega(x) : [a, b] \mapsto \bar{\mathbb{R}}$ be a real valued function such that $\omega(x) > 0$ for every $x \in (a, b)$ and that $\int_a^b f(x)\omega(x)\,dx$ is finite for every $f(x) \in \mathcal{P}$. It can be proved that the function that associates a pair of polynomials $(f(x), g(x))$, where $f(x), g(x) \in \mathcal{P}$, with

$$\langle f, g \rangle := \int_a^b f(x)g(x)\omega(x)\,dx \tag{1.4}$$

is a scalar product. The function $\omega(x)$ is usually called *weight function*. Any set of orthogonal polynomials $\{p_0(x), p_1(x), \ldots\}$ satisfies a relation among three consecutive polynomials, usually known as *three-term recurrence*.

**Theorem 1.0.3.** *Let $\{p_i(x)\}$, $i = 0, 1, \ldots$ be a set of orthogonal polynomials on the interval $[a, b]$ with respect to a scalar product* (1.4)*. There exist $A_i, B_i, C_i \in \mathbb{R}$ such that:*

$$p_{i+1}(x) = (xA_{i+1} + B_{i+1})\,p_i(x) - C_i p_{i-1}(x), \quad i \geq 1, \tag{1.5}$$

*where the coefficients $A_{i+1}, C_i$ are non zero for $i \geq 1$. More precisely, indicate by $a_i$ and $b_i$ the coefficients of degree $i$ and $i - 1$ of the polynomial $p_i(x)$, then for every $i = 0, 1, \ldots$, it holds:*

$$A_i = \frac{a_{i+1}}{a_i}, \quad B_i = \frac{a_{i+1}}{a_i}\left(\frac{b_{i+1}}{a_{i+1}} - \frac{b_i}{a_i}\right), \quad C_i = \frac{a_{i+1}a_{i-1}}{a_i^2}\frac{h_i}{h_{i-1}}, \tag{1.6}$$

*where $h_i := \langle p_i, p_i \rangle$.*

The three-term recurrence for a set of orthogonal polynomials can be used to prove another useful property of this family of polynomials. Consider the matrix pencil $S_i(x)$ defined as:

$$S_i(x) = \begin{bmatrix} a_1x + b_1 & -a_0 & & & \\ -C_1 & A_2x + B_2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -C_{i-1} & A_ix + B_i \end{bmatrix}, \tag{1.7}$$

where $A_k, B_k$ and $C_{k-1}$ for $k = 2, \ldots, i$ are the real constants in Theorem 1.0.3 and $p_0(x) := a_0$ and $p_1(x) := a_1x + b_1$ are orthogonal polynomials of degree 0 and 1, respectively. Applying Laplace's formula for the determinant of the matrix $S_i(x)$ along the last row, we have

$$\det S_i(x) = (A_ix + B_i)\det S_{i-1}(x) - C_{i-1}\det S_{i-2}(x). \tag{1.8}$$

Moreover, using that $\det S_1(x) = a_1 x + b_1 = p_1(x)$ and $\det S_2(x) = A_2 x p_1(x) + B_2 p_1(x) - C_1 p_0(x) = p_2(x)$, it can be proved by induction that:

$$\det S_i(x) = p_i(x), \quad \text{for every } i \geq 2. \tag{1.9}$$

The Chebyshev polynomials of the first kind $\{T_0(x),\ T_1(x), \ldots\}$ are orthogonal with respect to the weight function [45]:

$$\omega(x) := \left(1 - x^2\right)^{-\frac{1}{2}}, \qquad x \in [-1, 1]\,. \tag{1.10}$$

More precisely, we have that:

$$\begin{aligned} &\langle T_n(x), T_m(x)\rangle = 0, \quad \text{if } n \neq m, \\ &\langle T_n(x), T_n(x)\rangle = \frac{\pi}{2}, \quad \text{for } n > 0 \\ &\langle T_0(x), T_0(x)\rangle = \pi, \end{aligned}$$

where the scalar product is defined as $\langle f, g\rangle := \int_a^b f(x)g(x)\left(1 - x^2\right)^{-\frac{1}{2}}\, dx$. Therefore, the Chebyshev polynomials inherit the properties stated in Theorem 1.0.3 for a general set of orthogonal polynomials. In particular, the three-term recurrence is satisfied with $A_i = 2$, $B_i = 0$ and $C_i = 1$ for $i \geq 1$ and it holds:

$$T_{i+1}(x) = 2xT_i(x) - T_{i-1}(x), \quad x \in [-1, 1]\,, \tag{1.11}$$

starting from $T_0(x) \equiv 1$ and $T_1(x) = x$.

We now introduce another family of orthogonal polynomials known as Chebyshev polynomials of the second kind.

**Definition 1.0.4.** Let $k \geq 0$ be an integer. The $k$-th *Chebyshev polynomial of the second kind* is defined as:

$$U_k(x) = \frac{\sin\left((k+1)\cos^{-1} x\right)}{\sin\left(\cos^{-1} x\right)}, \quad x \in [-1, 1]\,. \tag{1.12}$$

Similarly, the Chebyshev polynomials of the second kind $\{U_0(x), U_1(x), \ldots\}$ are a set of orthogonal polynomials with respect to the weight function $\omega(x) = \left(1 - x^2\right)^{\frac{1}{2}}$ [45]. Therefore, it can be proved that the three-term recurrence introduced in Theorem 1.0.3 holds with $A_i = 2$, $B_i = 0$ and $C_i = 1$ for $i \geq 1$ and we have

$$U_{i+1}(x) = 2xU_i(x) - U_{i-1}(x), \quad x \in [-1, 1]\,, \tag{1.13}$$

where $U_0(x) = 1$ and $U_1(x) = 2x$. Moreover, the Chebyshev polynomials of the first and the second kind are related with each other by the following property:

**Proposition 1.0.5.** *Let $k = 1, 2, \ldots$, then it holds:*

$$T_k'(x) = k\, U_{k-1}(x), \qquad x \in [-1, 1]\,. \tag{1.14}$$

## 1.1 Clenshaw algorithm

We frequently need to evaluate polynomials at one or more points. There are several methods that we can use in order to perform this step: one of them is the Clenshaw algorithm. Let us explain in detail the steps of this approach. Consider a polynomial $q(x) \in \mathcal{P}_n$ expressed in the Chebyshev basis:

$$q(x) = \sum_{k=0}^{n} q_k T_k(x). \tag{1.15}$$

Our aim consists in evaluating the polynomial $q(x)$ at a point $\widetilde{x} \in [-1, 1]$. Define the vectors $\mathbf{q} = (q_0, q_1, \ldots, q_n)^T$ and $\mathbf{t} = (T_0(\widetilde{x}), T_1(\widetilde{x}), \ldots, T_n(\widetilde{x}))^T$. Then the evaluation of (1.15) at $\widetilde{x}$ can be written as

$$q(\widetilde{x}) = \mathbf{q}^T \mathbf{t} = (q_0, q_1, \ldots, q_n) \begin{pmatrix} T_0(\widetilde{x}) \\ T_1(\widetilde{x}) \\ \vdots \\ T_n(\widetilde{x}) \end{pmatrix}. \tag{1.16}$$

The three-term recurrence (1.11) allows us to write the following matrix relation:

$$\begin{pmatrix} 1 & & & & & \\ -2\widetilde{x} & 1 & & & & \\ 1 & -2\widetilde{x} & 1 & & & \\ & 1 & -2\widetilde{x} & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2\widetilde{x} & 1 \end{pmatrix} \begin{pmatrix} T_0(\widetilde{x}) \\ T_1(\widetilde{x}) \\ T_2(\widetilde{x}) \\ T_3(\widetilde{x}) \\ \vdots \\ T_n(\widetilde{x}) \end{pmatrix} = \begin{pmatrix} 1 \\ -\widetilde{x} \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}. \tag{1.17}$$

Denote as $\mathbf{A}$ the $(n+1) \times (n+1)$ matrix in (1.17) and let $\mathbf{d} = (1, -\widetilde{x}, \ldots, 0)^T$. Consider the vector $\mathbf{b} = (b_0, b_1, \ldots, b_n)^T$ that solves the linear system:

$$\mathbf{b}^T \mathbf{A} = \mathbf{q}^T. \tag{1.18}$$

Combining the equations (1.16), (1.17) and (1.18), we have:

$$q(\widetilde{x}) = \mathbf{q}^T \mathbf{t} = \mathbf{b}^T \mathbf{A} \mathbf{t} = \mathbf{b}^T \mathbf{d} = b_0 - b_1 \widetilde{x}. \tag{1.19}$$

Therefore, the evaluation of $q(x)$ at $\widetilde{x}$ can be obtained by computing the row vector $\mathbf{b}^T$. From the linear system (1.18), we obtain the recurrence relation:

$$b_r - 2\widetilde{x} b_{r+1} + b_{r+2} = p_r, \quad r = 0, 1, \ldots, n, \tag{1.20}$$

setting $b_{n+1} = 0$ and $b_{n+2} = 0$. The previous relation is solved in the backward direction, starting with $r = n$. Finally, exploiting the recurrence relation for the coefficients $b_r$, we have:

$$q(\widetilde{x}) = b_0 - b_1 \widetilde{x} = b_1 \widetilde{x} - b_2 + p_0. \tag{1.21}$$

The process is summarized in **Algorithm** 1 and is known as *Clenshaw algorithm*.

This algorithm allows us to evaluate polynomials expressed in the Chebyshev basis in an efficient way. For instance, the evaluation of a polynomial of degree $n$ has a computational cost of $2n$ additions and $n$ products [4]. Moreover, it can be proved that the Clenshaw algorithm has good stability properties [17, 42].

**Algorithm 1** Clenshaw algorithm

**Input**: $\widetilde{x}; q_0, q_1, \ldots, q_n$
**Output**: $q(\widetilde{x}) := \sum_{k=0}^{n} q_k T_k(\widetilde{x})$
1: **set** $b_{n+1} = 0$, $b_n = q_n$
2: **for** $r = n-1 : -1 : 1$ **do**
3: $\quad b_r = 2\widetilde{x} b_{r+1} - b_{r+2} + q_r$
4: **end for**
5: **compute** $q(\widetilde{x}) = \widetilde{x} b_1 - b_2 + q_0$

**Definition 1.1.1.** Consider $\widetilde{x} \in [-1, 1]$ and the evaluation $w = \sum_{k=0}^{n} q_k T_k(\widetilde{x})$. We say that an evaluation algorithm is *componentwise backward stable* if it produces a computed value $\widetilde{w}$ such that there exists a vector $\delta q = (\delta q_0, \ldots, \delta q_n)^T$ such that

$$\widetilde{w} = \sum_{k=0}^{n} (q_k + \delta q_k) T_k(\widetilde{x}), \tag{1.22}$$

where there exists $L$ such that $L = L(n)$ and for $k = 0, \ldots, n$

$$|\delta q_k| \leq \epsilon_M L |q_k| . \tag{1.23}$$

Here $\epsilon_M$ indicates the machine precision. In addition, we say that the algorithm is *normwise backward stable* if

$$\max_{k=0,\ldots,n} |\delta q_k| \leq \epsilon_M L \max_{k=0,\ldots,n} |q_k| . \tag{1.24}$$

Here $L = L(n)$ is a growing function of $n$. The Clenshaw algorithm for the computation of $w = \sum_{k=0}^{n} q_k T_k(\widetilde{x})$ is normwise backward stable with the constant $L$ of order $n^2$. Moreover, under the additional condition $|q_0| \geq |q_1| \geq \ldots \geq |q_n| \geq 0$, the Clenshaw algorithm is componentwise backward stable with the same constant $L$ [42].

The Clenshaw algorithm was originally designed for polynomials expressed in the basis of the Chebyshev polynomials of the first kind [10], but it can also be extended to every class of functions that can be defined by any three-term recurrence relation. Generalizations of this method can be developed in order to treat degree-graded bases, as proposed in [38]. Note that, when applied to polynomials expressed through the monomial basis, the Clenshaw algorithm reduces to the well-known Horner method.

## 1.2 Chebyshev interpolation

In this section, we describe the problem of interpolation. In fact, we frequently need to approximate a univariate function using a polynomial and for this reason, we recall the main theoretical results about polynomial interpolation. Then we focus on Chebyshev interpolation, analyzing its main advantages.

**Definition 1.2.1.** Given an interval $[a, b] \subset \mathbb{R}$ and an integer $n \geq 1$, we define *nodes of interpolation* a set of points such that

$$\left\{ x_i^{(n)} \in [a, b] : i = 0, \ldots, n,\ x_i^{(n)} \neq x_j^{(n)},\ \text{if } i \neq j \right\}. \tag{1.25}$$

Let $n$ be a natural number and $x_0, x_1, \ldots, x_n$ a set of $n+1$ interpolation nodes. Suppose that the values of the function $f$ at the nodes $x_i$, $i = 0, \ldots, n$, that is $f_i := f(x_i)$, are known. The problem of interpolation consists in computing the coefficients of a univariate polynomial $p_n(x) \in \mathcal{P}_n$ such that $p_n(x_i) = f_i$. This problem has a unique solution; see e.g. Chapter 5 of [4].

**Lemma 1.2.2.** *Consider $n \in \mathbb{N}$ and let $\left\{x_i^{(n)}\right\}$ be a set of nodes. Let $f : [a,b] \mapsto \mathbb{R}$ be a function of class $C^0$ in the interval $[a,b]$ and define $f_i := f\left(x_i^{(n)}\right)$ for $i = 0, 1, \ldots, n$. Then there exists a unique polynomial $p_n(x) \in \mathcal{P}_n$ such that:*

$$p_n\left(x_i^{(n)}\right) = f_i, \quad i = 0, 1, \ldots, n. \tag{1.26}$$

*We define $p_n(x)$ as the* polynomial interpolant *of degree $n$ for $f(x)$.*

The problem of interpolation can be formulated through a linear operator [29]. We denote as $C^0[a,b]$ the set of continuous function on the interval $[a,b]$. Given a set of $n+1$ interpolation nodes $x_0, x_1, \ldots, x_n$, the operator $\mathcal{A}_n$ that associates with $f(x) \in C^0[a,b]$ its polynomial interpolant $p_n(x) \in \mathcal{P}_n$ is defined as:

$$\begin{aligned} \mathcal{A}_n[x_0, \ldots, x_n] : C^0[a,b] &\mapsto \mathcal{P}_n \\ f(x) &\mapsto p_n(x) = \sum_{i=0}^{n} f_i L_{i,n}(x), \end{aligned}$$

where $L_{i,n}(x)$ for $i = 0, 1, \ldots, n$ are the Lagrange polynomials defined as:

$$L_{i,n}(x) = \prod_{j=0,\ldots,n, j\neq i} \frac{x - x_j}{x_i - x_j}. \tag{1.27}$$

In order to study the conditioning of the interpolation problem, we can study the conditioning of the linear operator $\mathcal{A}_n$. In particular, it can be proved that the norm of the operator $\mathcal{A}_n$, that is, $\|\mathcal{A}_n\|_\infty := \max_{g \in C^0[a,b], \|g\|_\infty = 1} \|\mathcal{A}_n(g)\|_\infty$, is equal to the *Lebesgue constant* defined as [41]:

$$\Lambda_n := \max_{x \in [a,b]} \sum_{i=0}^{n} |L_{i,n}(x)|. \tag{1.28}$$

Moreover, the Lebesgue constant provides a measure of the precision of the approximation through interpolation polynomials. We recall that given a function $f \in C^0[a,b]$, there exists a polynomial $p_n^* \in \mathcal{P}_n$ such that

$$\|f - p_n^*\|_\infty \leq \|f - q_n\|_\infty, \tag{1.29}$$

for any $q_n \in \mathcal{P}_n$ [40]. The polynomial $p_n^*(x)$ is called the polynomial of *best uniform approximation* to $f(x)$ in $[a,b]$.

**Proposition 1.2.3.** *Given $f \in C^0([a,b])$ and let $p_n(x) \in \mathcal{P}_n$ be the interpolant polynomial with respect to the nodes $x_0, x_1, \ldots, x_n$. Consider $p_n^*(x)$ the polynomial of best approximation to $f(x)$ in $[a,b]$. Then, it holds*

$$\|f - p_n\|_\infty \leq (1 + \Lambda_n) \|f - p_n^*\|_\infty. \tag{1.30}$$

*Proof.* Consider the interpolant polynomial $p_n(x)$ and the best polynomial approximation $p_n^*(x)$. Since $f - p_n = f - p_n^* + p_n^* - p_n$, it holds

$$\begin{aligned}\|f - p_n\|_\infty \leq \|f - p_n^*\|_\infty + \|p_n^* - p_n\|_\infty &= \|f - p_n^*\|_\infty + \|\mathcal{A}_n\left(p_n^* - f\right)\|_\infty \\ &\leq (1 + \Lambda_n)\,\|f - p_n^*\|_\infty.\end{aligned}$$

□

Note that the value of the Lebesgue constant depends only on the choice of the interpolation nodes. Therefore, we choose a set of interpolation nodes that keeps the Lebesgue constant small with the growth of the degree $n$ of the interpolation. Since the interval $[-1, 1]$ can be scaled to $[a, b]$ using the affine transformation

$$\begin{aligned} s : [-1, 1] &\mapsto [a, b] \\ x &\mapsto s(x) := \frac{b-a}{2}x + \frac{b+a}{2}, \end{aligned}$$

from now on, we consider univariate continuous functions $f(x)$ defined on the domain $[-1, 1]$. Several cases of interpolation nodes on the interval $[-1, 1]$ have been studied [41, 50]. One possible choice is the set of $n + 1$ *equispaced points*:

$$x_k^{(n)} = -1 + \frac{2k}{n}, \quad k = 0, 1, \ldots, n. \tag{1.31}$$

In this case, the value of the Lebesgue constant associated with $n+1$ equidistant points on the interval $[-1, 1]$ grows exponentially with the degree $n$ of the interpolation and it holds:

$$\Lambda_n \approx \frac{2^n}{en \log n}. \tag{1.32}$$

A more convenient choice is the set of *Chebyshev points* on the interval $[-1, 1]$.

**Definition 1.2.4.** Given $n \in \mathbb{N} \setminus \{0\}$, we define the $n$ *Chebyshev points* $x_k^{(n)}$ for $k = 0, 1, \ldots, n$ as:

$$x_k^{(n)} := \cos \frac{k\pi}{n}. \tag{1.33}$$

Some authors refer to this set as *Chebyshev-Lobatto points*. It can be easily proved that the Chebyshev points are the zeros of the $(n + 1)$-th Chebyshev polynomial of the second kind $U_{n+1}(x)$. Moreover, this set of points is also a useful choice for the numerical computation of integrals of functions. For instance, the Chebyshev points are used as nodes in the Clenshaw-Curtis quadrature formulas [22]. Choosing the Chebyshev points as interpolation nodes leads to a different asymptotic behaviour of the Lebesgue constant. In fact, it can be proved that [50]:

$$\Lambda_n \approx \frac{2}{\pi} \log n. \tag{1.34}$$

Throughout this thesis we prefer to use the Chebyshev points for any interpolation problems and we refer to this problem as *Chebyshev interpolation*. Furthermore, we choose to express Chebyshev interpolants using the Chebyshev basis.

If we deal with sufficiently regular functions defined in the interval $[-1, 1]$, the use of the Chebyshev interpolation has several advantages [50].

**Theorem 1.2.5.** *Let $\nu \geq 0$ be an integer. Consider $f : [-1,1] \mapsto \mathbb{R}$ an absolutely continuous function, whose derivatives $f^{(1)}, \ldots, f^{(\nu-1)}$ are absolutely continuous and suppose that the derivative $f^{(\nu)}$ is of bounded variation $V$, that is, $\|f^{(\nu+1)}\|_1 = V < \infty$. Then for any $n \in \nu$ the Chebyshev interpolants $p_n$ of the function $f$ satisfy*

$$\|f - p_n\|_\infty \leq \frac{4V}{\pi\nu\,(n-\nu)^\nu}. \tag{1.35}$$

Following the previous theorem, we deduce that the Chebyshev interpolants $p_n$ for $n \geq \nu$ have an accuracy of $\mathcal{O}(n^{-\nu})$. Moreover, if we consider functions infinitely differentiable on $[-1,1]$, we reach better results for the convergence of Chebyshev interpolants. Recall that a function $f : [-1,1] \mapsto \mathbb{R}$ is called smooth if it is of class $C^\infty$ on the interval $[-1,1]$. We know that a smooth function $f(x)$ defined on $[-1,1]$ is also analytic on $[-1,1]$. A function $f$ is *analytic* on a set $D$ of the complex plane if for any $s \in D$ the function $f$ has a Taylor series about $s$ that converges to $f$ in a neighborhood of $s$. There exist results of convergence for analytic function on regions of the complex plane known as Bernstein ellipses.

**Definition 1.2.6.** Given a real number $\rho > 1$, we define the *Bernstein ellipse* $E_\rho$ with *elliptical radius* $\rho$ the open region of the complex plane

$$E_\rho := \left\{ z = \frac{re^{i\theta} + r^{-1}e^{-i\theta}}{2},\ 1 \leq r \leq \rho,\ \theta \in [0, 2\pi) \right\}. \tag{1.36}$$

For functions that are analytic and bounded inside $E_\rho$, the Chebyshev interpolant $p_n$ converges at a rate $\mathcal{O}(\rho^{-n})$ [50].

**Theorem 1.2.7.** *Consider $\rho > 1$ and let a function $f$ be analytic inside the Bernstein ellipse $E_\rho$, where it satifies $|f(x)| \leq M$ for some $M$. Then for each $n \geq 0$ the Chebyshev interpolant $p_n$ of the function $f$ satisfies*

$$\|f - p_n\|_\infty \leq \frac{4M\rho^{-n}}{\rho - 1}. \tag{1.37}$$

An analytic function on $[-1,1]$ can be analytically continued to a neighborhood $D$ of its domain in the complex plane. Let us choose a number $\rho > 1$ such that $E_\rho \subset D$; Theorem 1.2.7 implies that the Chebyshev interpolants $p_n$ have an accuracy of rate $\mathcal{O}(\rho^{-n})$.

An example of the advantage of choosing Chebyshev interpolation nodes is described in [50] and consists in the approximation of the Runge function, that is $R(x) := \frac{1}{1+25x^2}$ for $x \in [-1,1]$. In particular, consider the polynomial interpolant $p_n$ constructed using $n+1$ equispaced points. The interpolation error $\|R(x) - p_n(x)\|_\infty$ increases as $n \to \infty$ and the succession $\{p_n\}_n$ diverges from $R(x)$ as $n$ increases, near the edges of the interval $[-1,1]$. The Chebyshev points, instead, overcome this difficulty. In fact, consider the polynomial interpolant $p_n$ constructed using $n+1$ Chebyshev points, the convergence is geometrically, as mentioned in Theorem 1.2.5. Further details about the Runge phenomenon can be found in [18].

Given a function $f \in C[-1,1]$ and a natural number $n$, the Chebyshev interpolation of $f$ can be written in matrix form. We construct the generalized Vandermonde matrix $V$ associated with the $n+1$ Chebyshev points $x_k$, $k = 0, 1, \ldots, n$

$$V = \begin{bmatrix} 1 & T_1(x_0) & T_2(x_0) & \cdots & T_n(x_0) \\ 1 & T_1(x_1) & T_2(x_1) & \cdots & T_n(x_1) \\ \vdots & \vdots & & & \vdots \\ 1 & T_1(x_{n-1}) & T_2(x_{n-1}) & \cdots & T_n(x_{n-1}) \\ 1 & T_1(x_n) & T_2(x_n) & \cdots & T_n(x_n) \end{bmatrix} \in \mathbb{R}^{(n+1)\times(n+1)}. \quad (1.38)$$

Let $f_k = f(x_k)$, for $k = 0, 1, \ldots, n$, be the values of the function $f(x)$, sampled at the Chebyshev points. In order to compute the coefficients of the polynomial interpolant $p_n(x) = \sum_{k=0}^{n} a_k T_k(x)$, we solve the following linear system

$$V\mathbf{a} = \mathbf{f}, \quad \text{where } \mathbf{a} := \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} \text{ and } \mathbf{f} := \begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_n \end{bmatrix}. \quad (1.39)$$

Similarly, we can evaluate a Chebyshev polynomial at the Chebyshev points through a matrix-vector multiplication. In fact, given the vector of coefficients $\mathbf{a}$, we perform the product between the generalized Vandermonde matrix $V$ and the column vector $a$ and we obtain the vector $\mathbf{f}$ of the values of the Chebyshev polynomial $p(x)$ at the $n+1$ Chebyshev points. Due to the particular choice of interpolation nodes, the solution and evaluation steps can be performed in an efficient way using the Fast Fourier Transform.

### 1.2.1 Chebyshev interpolation by FFT

In this section, we describe the equivalence between the Chebyshev interpolation at $n+1$ Chebyshev points and the trigonometric polynomial interpolation at $2n$ equispaced points in $[-\pi, \pi]$. The correspondence between these two interpolants can be studied using the Laurent function and it allows us to introduce the use of the Fast Fourier Transform (FFT) [3, 48].

Starting with the Chebyshev case, we have a polynomial $q(x) \in \mathcal{P}_n$ written in the Chebyshev basis:

$$q(x) = \sum_{k=0}^{n} a_k T_k(x), \qquad x \in [-1,1]. \quad (1.40)$$

The real variable $x \in [-1,1]$ corresponds to the complex variable $z \in \mathbb{S}^1$, where $\mathbb{S}^1 := \{z \in \mathbb{C} : |z| = 1\}$, through the relation

$$x = \operatorname{Re} z = \frac{1}{2}\left(z + z^{-1}\right). \quad (1.41)$$

Then, given $q(x)$, we define a function $Q(z)$ on the unit circle $\mathbb{S}^1$, by imposing $Q(z) = q(x)$. For each value of $x \in [-1,1]$, there exist two corresponding values

of $z \in \mathbb{S}^1$ and therefore the function satisfies the symmetry $Q(z) = Q(z^{-1})$. As a consequence, $Q(z)$ can be written as a *Laurent polynomial*

$$Q(z) = Q(z^{-1}) = \frac{1}{2}\sum_{k=0}^{n} a_k \left(z^k + z^{-k}\right), \qquad z \in \mathbb{S}^1, \tag{1.42}$$

which is a sum of powers, involving both the positive and the negative powers of $z$. Moreover, we can reformulate the Laurent polynomial $Q(z)$ introducing a change of variable. Let $\theta \in [-\pi, \pi]$ be such that $z = e^{i\theta}$ and then $x = \cos\theta$. We consider the *Fourier polynomial* $\mathcal{Q}$ defined as $\mathcal{Q}(\theta) = Q(e^{i\theta}) = q(\cos(\theta))$ on the domain $[-\pi, \pi]$. Using the correspondence between $z \in \mathbb{S}^1$ and $\theta \in [-\pi, \pi]$, we have that the function $\mathcal{Q}$ satisfies the symmetry $\mathcal{Q}(\theta) = \mathcal{Q}(-\theta)$. Therefore, the Fourier function can be expressed through the trigonometric polynomial:

$$\mathcal{Q}(\theta) = \mathcal{Q}(-\theta) = \frac{1}{2}\sum_{k=0}^{n} a_k \left(e^{ik\theta} + e^{-ik\theta}\right), \qquad \theta \in [-\pi, \pi]. \tag{1.43}$$

For these three equivalent polynomials, we consider three corresponding grids of points, which are related with each other. Starting from the Chebyshev formulation, we consider the grid of the $n+1$ Chebyshev points:

$$x_j = \cos\left(\frac{j\pi}{n}\right), \qquad j = 0, 1, \ldots, n. \tag{1.44}$$

Using the relation $x = \operatorname{Re} z$, we consider the $2n$-th roots of unity:

$$z_j = e^{\frac{i\pi}{n}j}, \qquad j = -n+1, -n+2, \ldots, n, \tag{1.45}$$

and through the correspondence between $\theta$ and $z$, we have the $2n$ equispaced points:

$$\theta_j = \frac{j\pi}{n}, \qquad j = -n+1, -n+2, \ldots, n. \tag{1.46}$$

We note that the Chebyshev interpolation is equivalent to the well-known problem of trigonometric interpolation at equispaced points, which can be solved using the discrete Fourier transform [11]. Let us recall the definition of the one dimensional discrete Fourier transform.

**Definition 1.2.8.** Consider $M$ a positive integer and a sequence of $M$ complex numbers $y_0, y_1, \ldots, y_{M-1}$. The *discrete Fourier transform* (DFT) of $(y_0, \ldots, y_{M-1})$ is the sequence $Y_0, Y_1, \ldots, Y_{M-1}$ defined as:

$$Y_k := \sum_{n=0}^{M-1} y_n \omega_M^{-nk}, \qquad k = 0, 1, \ldots, M-1, \tag{1.47}$$

where $\omega_M := e^{\frac{i2\pi}{M}}$ is a principal $M$-th root of unity.

We can rewrite (1.47) using the matrix formulation

$$\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_M & \omega_M^2 & \cdots & \omega_M^{M-1} \\ 1 & \omega_M^2 & \omega_M^4 & \cdots & \omega_M^{2M-2} \\ \vdots & \vdots & & & \vdots \\ 1 & \omega_M^{M-1} & \cdots & \cdots & \omega_M^{(M-1)^2} \end{bmatrix} \begin{bmatrix} \widetilde{Y}_0 \\ \widetilde{Y}_1 \\ \widetilde{Y}_2 \\ \vdots \\ \widetilde{Y}_{M-1} \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{M-1} \end{bmatrix}. \tag{1.48}$$

Denote by $\Omega_M$ the $M \times M$ matrix in (1.48). This matrix is known as *Fourier matrix* and it can be expressed as $\Omega_M = \left(\omega_M^{ij \bmod M}\right)$. The matrix $\Omega_M$ is a unitary matrix, if we re-scale its entries.

**Proposition 1.2.9.** *The matrix $\Omega_M$ defined in* (1.48) *satisfies the property*

$$\Omega_M^H \Omega_M = M I_M, \tag{1.49}$$

*where $I_M$ is the identity matrix of size $M$ and $\Omega_M^H$ denotes the conjugate transpose of $\Omega_M$.*

*Proof.* To prove that the columns of $\Omega_M$ are orthogonal, we perform the product between the $i$-th row of $\Omega_M$ and the $j$-th column of $\Omega_M$:

$$\sum_{k=0}^{M-1} \bar{\omega}_M^{ik} \omega_M^{jk} = \sum_{k=0}^{M-1} \omega_M^{(j-i)k}. \tag{1.50}$$

In order to evaluate this sum, we distinguish two cases. If $i = j$, the sum (1.50) is a sum of $M$ terms equal to 1. If $i \neq j$, recall the well-known factorization:

$$\left(1 - x^M\right) = (1 - x)\left(1 + x + x^2 + \ldots + x^{M-1}\right) \tag{1.51}$$

and set $x = \omega_M^{j-i}$. Since $j - i \neq 0$, we have that $1 - \omega_M \neq 0$, but $1 - \omega_M^M = 0$, and therefore, using (1.51),

$$1 + \omega_M^{(j-i)} + \omega_M^{2(j-i)} + \ldots + \omega_M^{(M-1)(j-i)} = \sum_{k=0}^{M-1} \omega_M^{(j-i)k} = 0. \tag{1.52}$$

This process leads to:

$$\sum_{k=0}^{M-1} \bar{\omega}_M^{ik} \omega_M^{jk} = \begin{cases} M & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}, \tag{1.53}$$

which implies that $\Omega_M^H \Omega_M = M I_M$. □

Unlike other Vandermonde matrices, the Fourier matrix has good stability properties [27]. In fact, from Proposition 1.2.9, we have that the condition number associated with the linear system (1.48) is equal to 1. Moreover, it is clear that $\Omega_M^{-1} = \frac{1}{M}\Omega_M^H$. Then, we have that the solution $\left(\widetilde{Y}_0, \widetilde{Y}_1, \ldots, \widetilde{Y}_{M-1}\right)^T$ of the linear system (1.48) and the sequence $Y_0, Y_1, \ldots, Y_{M-1}$ obtained via the discrete Fourier transform are related by the relation

$$\widetilde{Y}_k = \frac{1}{M} Y_k, \qquad \text{for } k = 0, 1, \ldots, M-1. \tag{1.54}$$

Therefore, the discrete Fourier transform solves an interpolation problem on the $M$-th roots of unity. The implementation of the discrete Fourier transform can be performed using the Fast Fourier Transform algorithm and requires $\mathcal{O}(M \log M)$ operations [27].

In our case, we are dealing with a DFT on $2n$ points, associated with the trigonometric formulation (1.43) of the interpolation problem. Consider the vector $[q(x_0), q(x_1), \ldots, q(x_n)]^T$ containing all the values of the polynomial $q(x)$, sampled at the grid of the Chebyshev points. Using the symmetry of the trigonometric polynomial, we extend the vector to:

$$\mathbf{q} := [q(x_0), q(x_1), \ldots, q(x_n), q(x_{n-1}), \ldots, q(x_1)]^T . \tag{1.55}$$

Then, we apply the Discrete Fourier Transform using the vector $\mathbf{q}$ of size $2n$ as input data and for this step, we employ the command `fft` in MATLAB. We select the first $n+1$ elements of the output vector and divide the first and the last coefficients by $2n$ and the others by $n$. In this way, we have computed the coefficients $a_0, a_1, \ldots, a_n$ of the polynomial that interpolates $q(x)$ at $x_j$ for $j = 0, \ldots, n$.

In a similar way, we can treat the evaluation problem at the $n+1$ Chebyshev points. For this purpose, we recall the definition of inverse discrete Fourier transform.

**Definition 1.2.10.** Consider $M$ a positive integer and a sequence of $M$ complex numbers $Y_0, Y_1, \ldots, Y_{M-1}$. The *inverse discrete Fourier transform* (IDFT) of $(Y_0, \ldots, Y_{M-1})$ is the sequence $y_0, y_1, \ldots, y_{M-1}$ defined as:

$$y_n := \frac{1}{M} \sum_{k=0}^{M-1} Y_k \omega_M^{nk}, \qquad n = 0, 1, \ldots, M-1, \tag{1.56}$$

where $\omega_M := e^{\frac{i2\pi}{M}}$ is a principal $M$-th root of unity.

In the same way, we find a correspondence between the equations (1.56) and the matrix vector product in the formulation (1.48). The problem of the evaluation of $q(x) = \sum_{k=0}^{n} a_k T_k(x)$ at the Chebyshev points can be solved employing the IDFT. We proceed as in the previous case, applying the `ifft` MATLAB command on the double-sized vector of the coefficients.

# Chapter 2

# Resultant matrices

The core of this thesis consists in the study and the development of a method for multivariate rootfinding. We focus in particular on the bivariate case, in view of a possible generalization to an arbitrary number of variables. In the literature, the problem of multivariate rootfinding may refer to at least two different tasks, which can both be seen as generalizations of the usual univariate rootfinding problem. In the two dimensional case, for instance, rootfinding may mean:

- finding the zero level curves of a bivariate function, or
- finding the common roots of two bivariate polynomials.

In this thesis, the term *multivariate rootfinding* refers to the second case. We denote as *grade* of a polynomial an integer at least as large as its degree. Note that the grade of a polynomial is not a straightforward deduction from the expression of the polynomial. Let $d \geq 1$ be an integer, we denote by $\mathbb{C}_n[x_1, x_2, \ldots, x_d]$ the set of the multivariate polynomials of grade $n$ (that is, of degree at most $n$) in the variables $x_1, x_2, \ldots, x_d$ and with coefficients in $\mathbb{C}$. Let $p_1, p_2, \ldots, p_d \in \mathbb{C}_n[x_1, x_2, \ldots, x_d]$: the rootfinding problem associated with these polynomials consists in approximating numerically the solutions of:

$$\begin{pmatrix} p_1(x_1, \ldots, x_d) \\ p_2(x_1, \ldots, x_d) \\ \vdots \\ p_d(x_1, \ldots, x_d) \end{pmatrix} = 0, \quad (x_1, \ldots, x_d) \in \mathbb{C}^d. \tag{2.1}$$

In particular, we focus on finding the common roots that belong to the set $[-1, 1]^d$. We consider rootfinding problems that have zero dimensional solution sets, that is, such that the common roots are isolated, and therefore, the polynomials do not share a common factor [13, 24]. We also assume that the common zeros are simple. We recall the definition of simple root of (2.1).

**Definition 2.0.1.** Let $x^* = (x_1^*, \ldots, x_d^*) \in \mathbb{C}^d$ be a solution of (2.1). Then, $x^*$ is called a *simple root* if the Jacobian matrix $J(x^*)$

$$J(x^*) = \begin{bmatrix} \frac{\partial p_1}{\partial x_1}(x^*) & \cdots & \frac{\partial p_1}{\partial x_d}(x^*) \\ \vdots & \ddots & \vdots \\ \frac{\partial p_d}{\partial x_1}(x^*) & \cdots & \frac{\partial p_d}{\partial x_d}(x^*) \end{bmatrix} \in \mathbb{C}^{d \times d}, \tag{2.2}$$

is invertible.

Several techniques and algorithms have been developed in order to solve the problem (2.1). We focus our attention on the class of resultant-based methods. For this reason, we introduce the notion of resultant matrices, with particular consideration to the Bézout resultant matrix. Let us start with the general notion of resultant associated with $d$ multivariate polynomials.

**Definition 2.0.2.** Let $d \geq 2$ and $n \geq 0$ be two natural numbers. A function

$$\mathcal{R} : \left(\mathbb{C}_n \left[x_1, \ldots, x_{d-1}\right]\right)^d \mapsto \mathbb{C} \tag{2.3}$$

is called a *multidimensional resultant* if, for any set of polynomials $q_1, \ldots, q_d \in \mathbb{C}_n \left[x_1, \ldots, x_{d-1}\right]$, $\mathcal{R}\left(q_1, \ldots, q_d\right)$ is a polynomial in the coefficients of $q_1, \ldots, q_d$ and $\mathcal{R}\left(q_1, \ldots, q_d\right) = 0$ if and only if there exists an $x^* \in \mathbb{P}^{d-1}\left(\mathbb{C}\right)$ such that $q_k(x^*) = 0$ for $k = 1, \ldots, d$, where $\mathbb{P}^{d-1}\left(\mathbb{C}\right)$ indicates the projective space of dimension $d-1$ over $\mathbb{C}$.

Recall that the projective space $\mathbb{P}^{d-1}\left(\mathbb{C}\right)$ is defined as $\left(\mathbb{C}^d \setminus \{0\}\right) / \sim$, where the relation $\sim$ identifies the points $x \sim \lambda x$ for every $\lambda \in \mathbb{C} \setminus \{0\}$. We mention that the theory of multivariate resultants is generally developed in the context of homogeneous polynomial systems, where the solutions belong to $\mathbb{P}^{d-1}\left(\mathbb{C}\right)$. An example of resultant for homogeneous polynomials is provided in Section 2.2.1; we refer the reader to [8, 20] for further details. In this work, however, we are interested in computing finite solutions of (2.1); therefore, we apply the notion of resultant to non-homogeneous polynomials and check that the solutions $x^*$ belong to $\mathbb{C}^{d-1}$.

We assume to have chosen a multidimensional resultant $\mathcal{R}$ and we compute the solutions of the problem (2.1) employing the hidden variable technique, described as follows. This method consists in selecting one variable and writing the $d$ polynomials as functions of the remaining variables. For instance, we select the variable $x_d$ and we rewrite the polynomials $p_k(x_1, \ldots, x_d)$ for $k = 1, \ldots, d$ as

$$p_k(x_1, \ldots, x_{d-1}, x_d) = p_k\left[x_d\right](x_1, \ldots, x_{d-1}). \tag{2.4}$$

In this way, we transform the previous system (2.1) into a system of $d$ polynomials in $d-1$ variables and we search for the values of $x_d^* \in \mathbb{C}$ such that the polynomials $p_1\left[x_d^*\right], \ldots, p_d\left[x_d^*\right]$ have a common root in $\left[-1, 1\right]^{d-1}$. To this aim, we consider a multidimensional resultant $\mathcal{R}$ and we have that for any $x_d^* \in \mathbb{C}$

$$\begin{aligned} \mathcal{R}\left(p_1\left[x_d^*\right], \ldots, p_d\left[x_d^*\right]\right) = 0 \iff & \exists \left(x_1^*, \ldots, x_{d-1}^*\right) \in \mathbb{C}^{d-1} \\ & \text{such that } p_1\left(x^*\right) = \cdots = p_d\left(x^*\right) = 0, \end{aligned}$$

where $x^* = (x_1^*, \ldots, x_d^*) \in \mathbb{C}^d$. Hence, we compute all the roots of $\mathcal{R}\left(p_1\left[x_d\right], \ldots, p_d\left[x_d\right]\right)$ and select the ones inside $\left[-1, 1\right]$. In this way, we compute the $d$-th component of all the solutions of (2.1). Instead of computing directly the roots of the function $\mathcal{R}$, we propose to consider a resultant matrix associated with the multivariate resultant $\mathcal{R}$. The idea of converting the rootfinding problem (2.1) with $d = 1$ into an eigenvalue problem was first proposed by Good in [25] and

then elaborated in several studies (see e.g. [6]). The eigenvalue problem associated with the rootfinding problem (2.1) can be solved using the QZ algorithm, which is a stable method for the computation of the eigenvalues [14]. However, the recast of the rootfinding problem (2.1) into an eigenvalue problem can worsen the conditioning of the problem, as analyzed in [36,38], therefore particular attention must be given to the potential loss of accuracy of the computed solutions.

**Definition 2.0.3.** Let $d \geq 2$ and $n \geq 0,\ N \geq 1$, and $\mathcal{R}$ be a multidimensional resultant. A function

$$R : \left(\mathbb{C}_n \left[x_1, \ldots, x_{d-1}\right]\right)^d \mapsto \mathbb{C}^{N \times N} \tag{2.5}$$

is a *multidimensional resultant matrix* associated with $\mathcal{R}$ if for any set of $d$ polynomials $q_1, \ldots, q_d \in \mathbb{C}_n \left[x_1, \ldots, x_d\right]$ we have that:

$$\det\left(R\left(q_1, \ldots, q_d\right)\right) = \mathcal{R}\left(q_1, \ldots, q_d\right). \tag{2.6}$$

A multivariate resultant matrix $R(q_1, \ldots, q_d)$ has entries that are combinations of the coefficients of the polynomials $q_1, \ldots, q_d$. In our case, the aim consists in computing the roots of $\mathcal{R}\left(p_1\left[x_d\right], \ldots, p_d\left[x_d\right]\right)$, where the polynomials $p_k\left[x_d\right]\left(x_1, \ldots, x_{d-1}\right)$ are defined as in (2.4). Note that the coefficients of the polynomials $p_k\left[x_d\right]\left(x_1, \ldots, x_{d-1}\right)$, for $k = 1, \ldots, d$, are polynomials in the variable $x_d$. Therefore, a multivariate resultant matrix $R$ associated with these polynomials has entries that are polynomials in the variable $x_d$ of finite degree. In our case, we conclude that the resultant matrix $R\left(p_1\left[x_1\right], \ldots p_d\left[x_d\right]\right)$ is a matrix polynomial.

## 2.1 Resultant matrix as matrix polynomial

Using the notion of resultant matrix, we transform the problem of finding the roots of the multivariate function $\mathcal{R}$ into an eigenvalue problem associated with the matrix $R$. In fact, we observe that the entries of the matrix $R\left[x_d\right] := R\left(p_1\left[x_d\right], \ldots, p_d\left[x_d\right]\right)$ are polynomials in the variable $x_d$. Therefore, $R$ is a matrix polynomial in the variable $x_d$ and computing the solutions of the equation $\det\left(R\left(p_1\left[x_d\right], \ldots, p_d\left[x_d\right]\right)\right) = 0$ corresponds to finding the eigenvalues of the matrix polynomial $R\left[x_d\right]$. Since matrix polynomials are essential tools in the hidden variable technique, let us recall the definitions and their main properties [24].

**Definition 2.1.1.** Consider two positive integers $k, h \geq 1$ and a natural number $n \geq 0$. A $k \times h$ *matrix polynomial* $P(\lambda)$ of degree $n$ in the variable $\lambda$ is a $k \times h$ matrix whose entries are polynomials in $\lambda$ of degree $\leq n$ and at least one entry is of degree exactly $n$.

In order to express a matrix polynomial, as in the scalar case, we can choose a polynomial basis. In particular, a matrix polynomial of degree $n$ can be expressed using a degree-graded basis $\{\phi_0, \phi_1, \ldots, \phi_n\}$. Let us recall the definition of degree-graded basis for $\mathbb{C}_n\left[\lambda\right]$.

**Definition 2.1.2.** A polynomial basis $\{\phi_0, \phi_1, \ldots, \phi_n\}$ for $\mathbb{C}_n\left[\lambda\right]$ is called *degree-graded* if, for each $i = 0, 1, \ldots, n,\ \phi_i(\lambda)$ is a univariate polynomial of degree exactly $i$.

A matrix polynomial $P(\lambda)$ of degree $n$ and size $k \times h$ can be written as

$$P(\lambda) = \sum_{i=0}^{n} A_i \phi_i(\lambda), \quad \text{where } A_i \in \mathbb{C}^{k \times h}. \tag{2.7}$$

In order to express the resultant matrix as a matrix polynomial, we consider the degree-graded basis formed by the Chebyshev polynomials of the first kind $\{T_0, T_1, \ldots, T_n\}$. Furthermore, resultant matrices are square matrix polynomials and hence we give particular consideration to matrix polynomials (2.7) where the coefficients $A_i$, for $i = 0, 1, \ldots, n$ are square matrices.

We consider rootfinding problems (2.1) where the solution set is zero dimensional. This hypothesis leads to resultant matrices that are regular matrix polynomials.

**Definition 2.1.3.** A matrix polynomial $P(\lambda)$ is *regular* if it is square and $\det(P(\lambda))$ is not identically zero. Otherwise, $P(\lambda)$ is *singular*.

For a regular matrix polynomial, we can define eigenvalues and eigenvectors.

**Definition 2.1.4.** Consider a regular matrix polynomial $P(\lambda)$ of size $k \times k$ and degree $n$. An *eigenvalue* $\lambda \in \mathbb{C}$ is a scalar such that $\det(P(\lambda)) = 0$. We say that a nonzero vector $v \in \mathbb{C}^k$ is a *right eigenvector* corresponding to $\lambda$ if $P(\lambda)v = 0$. In the same way, we say that a nonzero vector $w \in \mathbb{C}^k$ is a *left eigenvector* corresponding to $\lambda$ if $w^T P(\lambda) = 0$.

From these definitions, it is clear that the resultant matrix $R(p_1[x_d], \ldots, p_d[x_d])$ introduced above can be expressed as a matrix polynomial associated with the variable $x_d$. In order to compute all the roots of the multivariate resultant $\mathcal{R}(p_1[x_d], \ldots, p_d[x_d])$, we can solve the eigenvalue problem associated with the matrix polynomial $R[x_d]$. A possible method to perform this resolution step consists in costructing a matrix pencil with the same eigenvalues as the matrix polynomial. For this reason, we introduce the notion of linearization.

**Definition 2.1.5.** Consider a matrix polynomial $P(\lambda)$ of size $k \times k$ and degree $n$. A pencil $L(\lambda) = \lambda X + Y$ with $X, Y \in \mathbb{C}^{nk \times nk}$ is a *linearization* of $P(\lambda)$ if there exist two square unimodular matrix polynomials $E(\lambda)$ and $F(\lambda)$, that is, $\det(E(\lambda)) = \pm 1$, $\det(F(\lambda)) = \pm 1$, such that

$$E(\lambda)L(\lambda)F(\lambda) = \left[\begin{array}{c|c} P(\lambda) & 0 \\ \hline 0 & I_{(n-1)k} \end{array}\right] \tag{2.8}$$

The use of linearizations converts the eigenvalue problem associated with the matrix polynomial $R[x_d]$ into a eigenvalue problem associated with a matrix of larger size. Several classes of linearizations can be employed in order to solve the eigenvalue problem associated with the resultant matrix $R[x_d]$. Here we use the companion-like linearization introduced in [1]. Its construction is described in Chapter 3.

## 2.2 Resultant matrix in two dimensions

We start by considering the resultant matrix associated with two bivariate polynomials $p_1(x_1, x_2)$ and $p_2(x_1, x_2)$. In particular, we choose to focus on the

Sylvester and the Bézout resultant matrices. We also stress the main differences between these two matrices. Furthermore in the literature, we find several different types of resultant matrices, for example Macaulay and Cayley matrices, as described in [13, 30, 38].

### 2.2.1 The Sylvester matrix

A classical choice as resultant matrix in the bidimensional case is the Sylvester matrix [13]. We can define the Sylvester resultant matrix for two univariate polynomials expressed in any degree-graded basis [44], even though it is mostly applied to polynomials written in the monomial basis.

**Definition 2.2.1.** Consider $q_1, q_2 \in \mathbb{C}_n[x]$ two univariate polynomials expressed in a degree-graded basis $\{\phi_0, \dots \phi_n\}$ and of degrees exactly $\tau_1$ and $\tau_2$, respectively. We define the *Sylvester matrix* $R_{Sylv} \in \mathbb{C}^{(\tau_1+\tau_2)\times(\tau_1+\tau_2)}$ associated with the polynomials $q_1$ and $q_2$ row by row as

$$R_{Sylv}(i,:) = Y^{i,1}, \quad 0 \le i \le \tau_2 - 1, \tag{2.9}$$

where $Y^{i,1}$ is the row vector of coefficients such that

$$q_1(x)\,\phi_i(x) = \sum_{k=0}^{\tau_1+\tau_2-1} Y_k^{i,1}\phi_k(x)$$

and

$$R_{Sylv}(i+\tau_2,:) = Y^{i,2}, \quad 0 \le i \le \tau_1 - 1, \tag{2.10}$$

where $Y^{i,2}$ is the row vector of coefficients such that

$$q_2(x)\,\phi_i(x) = \sum_{k=0}^{\tau_1+\tau_2-1} Y_k^{i,2}\phi_k(x).$$

In the monomial basis, that is, for $\phi_k(x) = x^k$, the Sylvester matrix associated with $q_1(x) = \sum_{k=0}^{\tau_1} a_k x^k$ and $q_2(x) = \sum_{k=0}^{\tau_2} b_k x^k$ can be easily determined by the coefficients of $q_1$ and $q_2$ as follows

$$R_{Sylv} = \begin{pmatrix} a_0 & a_1 & \cdots & a_{\tau_1} & & \\ & \ddots & \ddots & \ddots & \ddots & \\ & & a_0 & a_1 & \cdots & a_{\tau_1} \\ b_0 & b_1 & \cdots & b_{\tau_2} & & \\ & \ddots & \ddots & \ddots & \ddots & \\ & & b_0 & b_1 & \cdots & b_{\tau_2} \end{pmatrix} \begin{matrix} \left.\vphantom{\begin{matrix}a\\a\\a\end{matrix}}\right\} \tau_2 \text{ rows} \\ \left.\vphantom{\begin{matrix}a\\a\\a\end{matrix}}\right\} \tau_1 \text{ rows} \end{matrix} \tag{2.11}$$

At this point, for polynomials expressed in the monomial basis we have a possible notion of resultant $\mathcal{R}_{Sylv}$, that is the determinant of the Sylvester matrix (2.11). We give a pratical example of the use of homogeneous polynomials. For simplicity, let us explain in detail the bivariate case involving two polynomials expressed in the monomial basis. For instance, consider two univariate polynomials of grade $d_1$ and $d_2$ with coefficients in $\mathbb{C}$

$$P(z) = a_0 + a_1 z + a_2 z^2 + \ldots + a_{d_1} z^{d_1}$$
$$Q(z) = b_0 + b_1 z + b_2 z^2 + \ldots + b_{d_2} z^{d_2}.$$

If $a_{d_1} \neq 0$ and $b_{d_2} \neq 0$, the resultant $\mathcal{R}(P, Q)$ vanishes if and only if $P$ and $Q$ have a common root in $\mathbb{C}$ [8]. The use of homogeneous polynomials associated with $P$ and $Q$ is essential if $a_{d_1} = 0$ and $b_{d_2} = 0$. In this case, the last column of $\mathcal{R}_{Sylv}(P, Q)$ defined as (2.11) is zero, but the polynomials $P$ and $Q$ do not need to share a common root in $\mathbb{C}$. In order to have a clear equivalence between the vanishing of the resultant $\mathcal{R}_{Sylv}$ and the existence of a common root, we introduce the homogenizations $P^h$ and $Q^h$ of the polynomials $P$ and $Q$:

$$P^h(z, w) = a_0 w^{d_1} + a_1 z w^{d_1 - 1} + \ldots + a_{d_1} z^{d_1}$$
$$Q^h(z, w) = b_0 w^{d_2} + b_1 z w^{d_2 - 1} + \ldots + b_{d_2} z^{d_2}.$$

The polynomials $P$ and $Q$ can be recovered from the corresponding homogeneous polynomials via evaluation at $w = 1$. Moreover, $P^h(1, 0) = Q^h(1, 0) = 0$ when $a_{d_1} = b_{d_2} = 0$. This property suggests to study the common roots of $P^h$ and $Q^h$ on the projective space $\mathbb{P}^1(\mathbb{C})$. Note that it holds $P^h(\lambda z, \lambda w) = \lambda^{d_1} P^h(z, w)$ for each $\lambda \in \mathbb{C}$ and similarly for $Q^h$. Therefore the problem of finding the common roots in $\mathbb{P}^1(\mathbb{C})$ is well-defined.

The Sylvester matrix written in the monomial basis is a common choice for the resultant matrix between two univariate polynomials. However, it is difficult to generalize this resultant matrix to the multidimensional case. To this end, we propose different types of resultant matrices, such as the Bézout matrix.

### 2.2.2 The Bézout matrix

A valid alternative to the Sylvester resultant matrix is the Bézout matrix. In this thesis, we prefer to use the Bézout resultant matrix and we describe the reasons throughout this Section. One of the advantages consists in the fact the the Bézout matrix is easier to generalize to the rootfinding problem (2.1) where $d > 2$. This generalization is usually known as the Cayley matrix and we analyze its properties in Section 2.3. As in the definition of the Sylvester matrix, the Bézout resultant matrix can be expressed using any degree-graded basis of polynomials.

**Definition 2.2.2.** Given two univariate polynomials $q_1, q_2 \in \mathbb{C}_n[x_1]$, we define the *Bézoutian function* associated with $q_1(x_1)$ and $q_2(x_1)$ as the bivariate function

$$\mathcal{B}(q_1, q_2) := \frac{q_1(s)q_2(t) - q_1(t)q_2(s)}{s - t}. \tag{2.12}$$

Writing the previous expression using the degree-graded basis $\{\phi_0, \ldots, \phi_n\}$, the *Bézout matrix* $B(q_1, q_2) := (b_{ij})_{0 \leq i,j \leq n-1}$ associated with $q_1$ and $q_2$ has coefficients defined by:

$$\frac{q_1(s)q_2(t) - q_1(t)q_2(s)}{s - t} = \sum_{i,j=0}^{n-1} b_{ij} \phi_i(s) \phi_j(t). \tag{2.13}$$

The bivariate function $\mathcal{B}(q_1, q_2)$ defined in (2.12) is a polynomial in the variables $s$ and $t$ [26]. This result can be proved with an easy observation. We fix the second variable $t = t_0$. Then, the numerator of $\mathcal{B}(q_1, q_2)$ becomes $q_1(s)q_2(t_0) - q_1(t_0)q_2(s)$ and vanishes at the point $s = t_0$. We divide the numerator by $s - t_0$ and then we obtain that $\mathcal{B}(q_1, q_2)(s, t_0)$ is a polynomial in the variable $s$. We repeat this process fixing the first variable $s = s_0$. The numerator $q_1(s_0)q_2(t) - q_1(t)q_2(s_0)$ vanishes at the point $t = s_0$. Then we divide the numerator by $s_0 - t$ and obtain a polynomial in the variable $t$.

From now on, we focus our attention on the Chebyshev Bézout matrix, that is, the resultant matrix expressed in the basis formed by Chebyshev polynomials of the first kind $\{T_0, \ldots, T_n\}$. In this thesis, the construction of the Bézout matrix is instrumental for the resolution of the multivariate rootfinding problem (2.1), in the case where $d$ is equal to 2. Therefore, we provide several important theoretical results that can be useful in order to solve the bivariate case of the rootfinding problem.

Firstly, we consider two bivariate polynomials $q_1(x_1, x_2), q_2(x_1, x_2) \in \mathbb{C}_n[x_1, x_2]$, and, after the application of the hidden variable technique, we construct the matrix $B(q_1, q_2)$. Following the remarks in Section 2.1 about the class of resultant matrices, this matrix can be written as a matrix polynomial in the variable $x_2$, that is:

$$B(x_2) := B(q_1, q_2)(x_2) = \sum_{i=0}^{n-1} A_i T_i(x_2), \quad A_i \in \mathbb{C}^{n \times n}. \tag{2.14}$$

One of the advantages in choosing the Bézout matrix consists in the special form of its eigenvectors, which are in a Vandermonde form in the Chebyshev basis [38]. Recall the following definition.

**Definition 2.2.3.** A vector $v = [v_0, \ldots, v_{N-1}]^T \in \mathbb{R}^N$ is in *Vandermonde form* with respect to the Chebyshev basis if there exists a value $x \in \mathbb{R}$ such that $v_i = T_i(x)$ for $0 \leq i \leq N - 1$.

Let us exhibit a lemma about the relation between the Bézoutian function $\mathcal{B}$ and the Bézout resultant matrix $B$.

**Lemma 2.2.4.** *Let $t^* \in \mathbb{C}$ and consider the Bézoutian function $\mathcal{B}$ and the Bézout resultant matrix $B$ associated with $q_1, q_2 \in \mathbb{C}_n[x_1]$. Consider $v \in \mathbb{C}^n$ such that $v_i := T_i(t^*)$ for $0 \leq i \leq n - 1$, then it holds:*

$$B(q_1, q_2)\, v = y, \tag{2.15}$$

*where $y$ is the vector of size $n$ satisfying*

$$\mathcal{B}(s, t^*) = \sum_{i=0}^{n-1} y_i T_i(s). \tag{2.16}$$

*Proof.* The relation (2.15) can be rewritten row by row and obtaining, for each $i = 0, \ldots, n - 1$:

$$\sum_{j=0}^{n-1} b_{ij} T_j(t^*) = y_i. \tag{2.17}$$

Then evaluating $\mathcal{B}$ at $t^*$, we get

$$\mathcal{B}(s,t^*) = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} b_{ij}T_i(s)T_j(t^*). \tag{2.18}$$

In this way, we obtain the equation (2.16). □

This relation between performing a matrix-vector product with $B$ and evaluating $\mathcal{B}$ is employed in the following lemma.

**Lemma 2.2.5.** *Let $x^* = (x_1^*, x_2^*) \in \mathbb{C}^2$ be a simple common root of the polynomials $q_1$ and $q_2$ and let $v$ be a vector of size $n$, defined by:*

$$v_i := T_{i-1}(x_1^*), \qquad \text{for } 1 \le i \le n. \tag{2.19}$$

*Then the vector $v$ is a right eigenvector of the matrix polynomial $B(q_1[x_2^*], q_2[x_2^*])$ corresponding to the eigenvalue $x_2^*$.*

*Proof.* Consider the Bézoutian function $\mathcal{B}$ associated with the polynomials $q_1[x_2^*]$ and $q_2[x_2^*]$. Using Definition (2.12) and the fact that $(x_1^*, x_2^*)$ is a common zero of the polynomials $q_1$ and $q_2$, it holds that $\mathcal{B}(s, x_1^*) = 0$. At this point, thanks to Lemma 2.2.4, we have:

$$0 = \mathcal{B}(s, x_1^*) = \sum_{i=0}^{n-1} y_i T_i(s). \tag{2.20}$$

Since $\{T_0, T_1, \ldots, T_{n-1}\}$ is a polynomial basis, we conclude that $y = 0$, and therefore $B(x_2^*)v = 0$, that is, $v$ is a right eigenvector of the resultant matrix $B$ with respect to the eigenvalue $x_2^*$. Moreover, since $T_0(x_1^*) \neq 0$, the vector $v$ is nonzero. It can be proved similarly that the left eigenvector has the same Vandemonde form. □

The form of the eigenvectors of the Bézout matrix $B(q_1, q_2)$ is particularly useful during the study of the conditioning of the eigenvalue problem associated with the original rootfinding problem. This is one of the main reasons that motivate the choice of the Bézout matrix over the Sylvester one. Moreover, we point out that the several important properties of the Sylvester resultant matrix also hold true for the Bézout matrix. To this end, we state a theorem introduced by Kravitsky in [32].

**Theorem 2.2.6.** *Let $q_1$ and $q_2$ be two univariate polynomials with real coefficients. Then the resultant defined via the Sylvester matrix and the resultant defined as the determinant of the Bézout matrix are equal in absolute value.*

This theorem ensures that the results obtained using the notion of resultant via the Sylvester matrix can be transposed to the Bézout resultant. In particular, we focus our attention on the properties used in order to compute the solutions of the equation $\det(B) = 0$ and we introduce a result about the common roots of two bivariate polynomials (see e.g. [13]).

**Theorem 2.2.7.** *Given $f, g \in \mathbb{C}[x_1]$ of positive degree, then $f$ and $g$ have a common factor in $\mathbb{C}[x_1]$ if and only if the Sylvester resultant associated with $f$ and $g$ is identically zero.*

**Theorem 2.2.8.** *Suppose that $f, g \in \mathbb{C}[x_1, x_2]$ have positive degree in $x_1$, then the polynomials share a common factor in $\mathbb{C}[x_1, x_2]$ of positive degree in $x_1$ if and only if they have a common factor in $\mathbb{C}(x_2)[x_1]$, which is the ring of polynomials in the variable $x_1$ with coefficients that are polynomials in $\mathbb{C}[x_2]$.*

Note the the previous theorems can also be applied if we are dealing with a Bézout resultant, because of the relation provided by Theorem 2.2.6. Theorems 2.2.7 and 2.2.8 imply that a common factor in $\mathbb{C}[x_1, x_2]$ leads to a resultant that is identically zero. Throughout our discussion, we assume that the solution set of the bivariate rootfinding problem is zero dimensional, that is, the common zeros are isolated. This is enough to ensure that the two bivariate polynomials $q_1(x_1, x_2)$ and $q_2(x_1, x_2)$ do not share a common factor. This additional hypothesis implies that the number of solutions of the rootfinding problem in the form (2.1) is finite (see Chapter 3 in [31]).

Summarizing the foregoing results, we choose to include the Bézout resultant matrix in the development of our resultant-based method. This choice is motivated by the fact that the eigenvectors of the Bézout matrix have a special form, which makes it easier to study the conditioning of the eigenvalue problem. Moreover, the Bézout matrix inherits several useful properties of the Sylvester resultant matrix.

## 2.3 The Cayley matrix

In the previous section, we proposed two possible resultant matrices to employ in order to solve the bidimensional rootfinding problem. Starting from the definition of the Bézout matrix, we generalize this notion to the $d$ dimensional case, with the introduction of a the Cayley resultant matrix. To this end, we provide a definition of the Cayley function.

**Definition 2.3.1.** Given $q_1, \ldots, q_d \in \mathbb{C}_n[x_1, \ldots, x_{d-1}]$, we define the *Cayley resultant function* associated with these $d$ polynomials as the multivariate function in $2d-2$ variables such that:

$$f_{Cayley} := \frac{\det \begin{pmatrix} q_1(s_1, s_2, \ldots, s_{d-1}) & \cdots & q_d(s_1, s_2, \ldots, s_{d-1}) \\ q_1(t_1, s_2, \ldots, s_{d-1}) & \cdots & q_d(t_1, s_2, \ldots, s_{d-1}) \\ \vdots & \ddots & \vdots \\ q_1(t_1, t_2, \ldots, t_{d-1}) & \cdots & q_d(t_1, t_2, \ldots, t_{d-1}) \end{pmatrix}}{\prod_{i=1}^{d-1}(s_i - t_i)}. \quad (2.21)$$

Following the idea presented in [9], we prove that the Cayley function $f_{Cayley}$ is a multivariate polynomial. The numerator in (2.21) is a polynomial in $s_1, \ldots, s_{d-1}, t_1, \ldots, t_{d-1}$. Moreover, the numerator vanishes when $s_i = t_i$ for $i = 1, \ldots, d-1$, since the matrix in the numerator has equal rows. Then, the numerator is divisible for the polynomial $\prod_{i=1}^{d-1}(s_i - t_i)$. By an application of Laplace's formula for the determinant of the matrix at the numerator in the definition (2.21), it can be proved that the function $f_{Cayley}$ is a polynomial of degree $\tau_k \leq kn-1$ in the variables $s_k$ and $t_k$ for every $1 \leq k \leq d-1$. Therefore, we can write the polynomial $f_{Cayley}$ using a degree-graded basis $\{\phi_0, \phi_1, \ldots\}$,

that is

$$f_{Cayley} = \sum_{i_1=0}^{\tau_1} \cdots \sum_{i_{d-1}=0}^{\tau_{d-1}} \sum_{j_1=0}^{\tau_{d-1}} \sum_{j_{d-1}=0}^{\tau_1} A_{i_1,\ldots,i_{d-1},j_1,\ldots,j_{d-1}} \prod_{k=1}^{d-1} \phi_{i_k}(s_k) \prod_{k=1}^{d-1} \phi_{j_k}(t_k), \tag{2.22}$$

where $A$ is a $(2d-2)$-dimensional tensor of the coefficients of size $(\tau_1+1) \times \cdots \times (\tau_{d-1}+1) \times (\tau_{d-1}+1) \times \cdots \times (\tau_1+1)$.

In the two dimensional case, the Cayley function is equal to the Bézoutian function (2.12). In fact, given two polynomials $q_1, q_2 \in \mathbb{C}_n[x_1]$, we compute the Cayley function following (2.21)

$$f_{Cayley} = \frac{1}{s_1 - t_1} \det \begin{pmatrix} q_1(s_1) & q_2(s_1) \\ q_1(t_1) & q_2(t_1) \end{pmatrix} = \frac{q_1(s_1)q_2(t_1) - q_2(s_1)q_1(t_1)}{s_1 - t_1}, \tag{2.23}$$

and we obtain the definition of Bézoutian function, which is a polynomial of degree at most $n-1$ both in $s_1$ and $t_1$.

### 2.3.1 Unfolding of a tensor

The final step for the definition of the Cayley resultant matrix requires several notions about the unfolding of a tensor. For this reason, we recall some general definitions and useful properties. We refer the reader to [39] for further details.

Consider a $d$-dimensional tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ and $\mathbf{i} = (i_1, \ldots, i_d)$ a vector of indices. We denote by $\mathcal{A}(\mathbf{i})$ the entry in position $(i_1, \ldots, i_d)$ of the tensor $\mathcal{A}$.

**Definition 2.3.2.** Let $\mathcal{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ be a $d$-dimensional tensor and let $N = n_1 \cdots n_d$. The *vectorization* of $\mathcal{A}$ is a column vector in $\mathbb{R}^N$ inductively defined by

$$v := \mathrm{vec}(\mathcal{A}) = \mathcal{A} \tag{2.24}$$

if $d = 1$, and as follows otherwise:

$$v := \mathrm{vec}(\mathcal{A}) = \begin{bmatrix} \mathrm{vec}\left(\mathcal{A}^{(1)}\right) \\ \vdots \\ \mathrm{vec}\left(\mathcal{A}^{(n_d)}\right) \end{bmatrix}, \tag{2.25}$$

where $\mathcal{A}^{(k)}$ is the $(d-1)$-dimensional tensor defined as:

$$\mathcal{A}^{(k)}(i_1, \ldots, i_{d-1}) = \mathcal{A}(i_1, \ldots, i_{d-1}, k), \quad \text{for } 1 \le k \le n_d \tag{2.26}$$

and where $1 \le i_j \le n_j$ for $j = 1, \ldots, d-1$.

Note that each entry of the tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ corresponds to an element in its vectorization $v$. Given the vector $\mathbf{n} = (n_1, \ldots, n_d)$, the following relation holds

$$v_{\mathrm{ivec}(\mathbf{i},\mathbf{n})} = \mathcal{A}(\mathbf{i}), \quad \text{for all } \mathbf{1} \le \mathbf{i} \le \mathbf{n}, \tag{2.27}$$

where we indicate by $\mathrm{ivec}(\cdot, \mathbf{n})$ the index:

$$\mathrm{ivec}(\mathbf{i}, \mathbf{n}) = i_1 + (i_2 - 1)n_1 + (i_3 - 1)n_1 n_2 + \cdots + (i_d - 1)n_1 \cdots n_{d-1}. \tag{2.28}$$

We denote as $\mathbf{1} \leq \mathbf{i} \leq \mathbf{n}$ the set of inequalities $1 \leq i_k \leq n_k$ for $k = 1, \ldots, d$. In order to provide the general definition of unfolding of a tensor, we need the notion of **p**-transpose of a tensor.

**Definition 2.3.3.** Consider a $d$-dimensional tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ and a permutation $\mathbf{p} = (p_1, \ldots, p_d)$ of the vector $(1, 2, \ldots, d)$. We define the **p**-*transpose* of $\mathcal{A}$ as the tensor $\mathcal{A}^{\langle \mathbf{p} \rangle} \in \mathbb{R}^{n_{p_1} \times \cdots \times n_{p_d}}$ such that

$$\mathcal{A}^{\langle \mathbf{p} \rangle} (i_{p_1}, \ldots, i_{p_d}) = \mathcal{A} (i_1, \ldots, i_d) \quad \mathbf{1} \leq \mathbf{i} \leq \mathbf{n}. \tag{2.29}$$

In order to perform the unfolding of a tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, we need to choose a positive integer $e$ such that $1 \leq e \leq d$ and a permutation $\mathbf{p}$ of $(1, \ldots, d)$.

**Definition 2.3.4.** Given an integer $e$ and a permutation $\mathbf{p}$, consider

$$\begin{aligned} \mathbf{r} &= \mathbf{p}(1, \ldots, e) \\ \mathbf{c} &= \mathbf{p}(e+1, \ldots, d). \end{aligned} \tag{2.30}$$

Then the $\mathbf{r} \times \mathbf{c}$ *unfolding* of $\mathcal{A}$ is the matrix $\mathcal{A}_{\mathbf{r} \times \mathbf{c}}$ such that:

$$\mathcal{A}_{\mathbf{r} \times \mathbf{c}}(\alpha, \beta) = \mathcal{A}^{\langle \mathbf{p} \rangle}(i_1, \ldots, i_e, j_1, \ldots, j_{d-e}), \tag{2.31}$$

where

$$\alpha = \text{ivec}(\mathbf{i}, \mathbf{n}(\mathbf{r})) \quad \mathbf{1} \leq \mathbf{i} \leq \mathbf{n}(\mathbf{r}) \tag{2.32}$$

$$\beta = \text{ivec}(\mathbf{j}, \mathbf{n}(\mathbf{c})) \quad \mathbf{1} \leq \mathbf{j} \leq \mathbf{n}(\mathbf{c}). \tag{2.33}$$

Therefore, the unfolding operation transforms a $d$-dimensional tensor $\mathcal{A}$ in a matrix with $n_{p_1} \cdots n_{p_e}$ rows and $n_{p_{e+1}} \cdots n_{p_d}$ columns.

At this point, we can define the Cayley resultant matrix via an unfolding step, employing the $(2d-2)$-dimesional tensor of coefficients $A$ introduced in (2.22).

**Definition 2.3.5.** The *Cayley resultant matrix* $R_{Cayley}$ associated with the polynomials $q_1, \ldots, q_d \in \mathbb{C}_n [x_1, \ldots, x_{d-1}]$ with respect to the degree-graded basis $\{\phi_0, \phi_1, \ldots\}$ is the matrix of size $\left(\prod_{k=1}^{d-1} (\tau_k + 1)\right) \times \left(\prod_{k=1}^{d-1} (\tau_k + 1)\right)$ obtained through the $\{1, \ldots, d-1\} \times \{d, \ldots, 2d-2\}$ unfolding of the tensor $A$ in (2.22).

The Cayley matrix is particularly useful in the multidimensional rootfinding problem. One of the main reasons is that it can be employed for an arbitrary number of polynomials and variables, unlike the Sylvester matrix. Another advantage in choosing the Cayley matrix consists in the special form of the eigenvectors of the resultant matrix. In fact, the Cayley resultant matrix $R_{Cayley}$ can be seen as a matrix polynomial in the variable $x_d$, when we employ a hidden variable technique in order to solve the rootfinding problem (2.1). In particular, it can be proved that its eigenvectors have a generalized Vandermonde form, similarly to the case of the eigenvectors of the Bézout matrix in Lemma 2.2.5.

# Chapter 3

# Bivariate rootfinding

In this chapter, we focus on the bivariate rootfinding problem. As mentioned before, we choose a resultant-based method and in particular we prefer to use the Bézout matrix. A non-trivial step in the hidden variable resultant-based method is the construction of the Bézout matrix associated with two polynomials. One possible approach to perform this task is presented in [36], where the authors exploit the relation between this class of matrices and a particular set of linearizations of matrix polynomials. Here, however, we decide to implement the resultant-based method using a different technique, based on interpolation. The choice of relying on interpolation is particularly interesting because this approach is easier to generalize to multivariate rootfinding problems involving three or more polynomials. In the following sections, we describe both the idea introduced in [36] and the approach through interpolation.

We start by recalling the setting of the bivariate rootfinding problem. Consider two bivariate scalar polynomials $p(x,y), q(x,y) \in \mathbb{R}[x,y]$. Our aim consists in numerically approximating all the finite solutions of:

$$\begin{cases} p(x,y) = 0 \\ q(x,y) = 0 \end{cases}, \qquad \text{where } (x,y) \in [-1,1]^2 . \tag{3.1}$$

A more general rootfinding problem consists in finding the common zeros of two bivariate functions. In this case, the rootfinding problem can be treated in an analogous way, replacing the functions with their polynomial interpolants. Let us explain this step. Consider two smooth functions $f(x,y)$ and $g(x,y)$ defined on $[-1,1]^2$. We substitute the functions with their polynomials interpolants $p(x,y)$ and $q(x,y)$, which approximate $f(x,y)$ and $g(x,y)$ in the following way:

$$\|f-p\|_\infty = \mathcal{O}(u)\|f\|_\infty, \quad \|g-q\|_\infty = \mathcal{O}(u)\|g\|_\infty, \tag{3.2}$$

where we indicate by $\|f\|_\infty = \max_{x,y\in[-1,1]^2} |f(x,y)|$ and by $u$ the machine unit roundoff. For this reason, we treat rootfinding problems involving only polynomial functions. Consider $p(x,y)$ a polynomial of degree $n_p$ in the variable $x$ and of degree $m_p$ in the variable $y$ and similarly a polynomial $q(x,y)$ of degree $n_q$ in the variable $x$ and of degree $m_q$ in the variable $y$. The Chebyshev polynomials described in Chapter 1 can be used in order to express bivariate polynomials, using the tensor product basis $\{T_i(x)\cdot T_j(y)\}_{ij}$. Following this

idea, we express the bivariate polynomials $p(x, y)$ and $q(x, y)$:

$$p(x,y) = \sum_{i=0}^{n_p}\sum_{j=0}^{m_p} P_{ij}T_i(x)T_j(y), \qquad q(x,y) = \sum_{i=0}^{n_q}\sum_{j=0}^{m_q} Q_{ij}T_i(x)T_j(y), \tag{3.3}$$

where $P \in \mathbb{R}^{(n_p+1)\times(m_p+1)}$, $Q \in \mathbb{R}^{(n_q+1)\times(m_q+1)}$ and $T_i$ is the $i$-th Chebyshev polynomial of the first kind. Throughout this chapter, we assume that the solution set of (3.1) is zero dimensional and hence that the polynomials $p(x, y)$ and $q(x, y)$ do not share a common factor.

## 3.1 Resultant-based method

Our resolution algorithm consists in solving the bivariate rootfinding problem (3.1) using a hidden variable technique in combination with the resultant-based method. We start by applying a step of the hidden variable technique. We choose one of the two variables, for instance $y$, and rewrite the polynomials $p(x, y)$ and $q(x, y)$ as functions of the other variable $x$, that is:

$$p_y(x) := \sum_{i=0}^{n_p} \alpha_i(y)T_i(x), \quad q_y(x) := \sum_{i=0}^{n_q} \beta_i(y)T_i(x), \quad x \in [-1, 1] \tag{3.4}$$

where $\alpha_i$ for $i = 0, \ldots, n_p$ and $\beta_i$ for $i = 0, \ldots, n_q$ are polynomials in the variable $y$ with real coefficients:

$$\alpha_i(y) := \sum_{j=0}^{m_p} P_{ij}T_j(y), \quad \beta_i(y) := \sum_{j=0}^{m_q} Q_{ij}T_j(y), \quad y \in [-1, 1]\,. \tag{3.5}$$

Equivalently, we may choose the variable $x$ and rewrite the polyomials $p(x, y)$ and $q(x, y)$ as functions of the variable $y$. This choice can change the size of the resultant matrices that we construct, and consequently, the computational cost could change, as described in [36]. It is advisable to choose the variable that makes the resulting eigenvalue problem cheaper, minimizing the size of the resultant matrix. For simplicity, we assume it to be $y$ throughout the chapter.

At this point, we construct the Chebyshev-Bézout matrix associated with the polynomials $p_y(x)$ and $q_y(x)$, as previously described in (2.13), that is, the matrix $B(p_y, q_y) := (b_{ij}(y))_{0\le i,j\le N-1}$ such that

$$\frac{p_y(s)q_y(t) - p_y(t)q_y(s)}{s - t} = \sum_{i,j=0}^{N-1} b_{ij}(y)T_i(s)T_j(t), \tag{3.6}$$

where $N := \max(n_p, n_q)$. We observe that the elements $b_{ij}(y),\ i, j = 0, \ldots, N-1$ are polynomials in the variable $y$ with real coefficients. For this reason, we can rewrite the Bézout matrix as a matrix polynomial in the variable $y$, using the Chebyshev basis:

$$B(y) := B(p_y, q_y) = \sum_{i=0}^{M} A_iT_i(y), \tag{3.7}$$

where $M = m_p + m_q$ and $A_i \in \mathbb{R}^{N\times N}$. We define the *resultant* between the polynomials $p_y(x)$ and $q_y(x)$ as the determinant of $B(y)$. Using Theorems 2.2.7 and 2.2.8, we have that the previously made hypotheses about the solution set ensure that the resultant is not identically zero. Therefore the matrix polynomial $B(y)$ is regular [24]. Moreover, using the additional hypotheses on the solution set, we also have that the number of the common zeros between the bivariate polynomials $p(x, y)$ and $q(x, y)$ is finite. Note that this result presented by Kirwan [31] was developed for the Sylvester resultant, but Theorem 2.2.6 ensures that it is also valid for the Bézout resultant.

At this point, since $B(y)$ is a regular matrix polynomial, we have that the solutions of

$$\det\left(B(y)\right) = \det\left(B(p_y, q_y)\right) = 0 \tag{3.8}$$

are the finite eigenvalues of $B(y)$. Hence the rootfinding problem becomes a polynomial eigenvalue problem. We propose to solve this problem using a linearization approach. In particular, we need to construct a companion-like pencil for matrix polynomials expressed using the Chebsyhev basis. We use the linearization proposed in [1] and first introduced by Good in [25].

**Proposition 3.1.1.** *Consider a matrix polynomial $P(\lambda)$ of degree $d$, expressed in the Chebyshev basis, that is:*

$$P(\lambda) := P_0T_0(\lambda) + P_1T_1(\lambda) + \ldots + P_dT_d(\lambda),$$

*where $P_i \in \mathbb{C}^{m\times m}$ for each $i = 0, 1, \ldots d$. Then the eigenvalues $\lambda \in \mathbb{C}$ of the matrix polynomial $P(\lambda)$ are the generalized eigenvalues of the pencil $\mathcal{L}_0 - \lambda\mathcal{L}_1$, that is, the solutions of:*

$$\det\left(\mathcal{L}_0 - \lambda\mathcal{L}_1\right) = 0, \tag{3.9}$$

*where*

$$\mathcal{L}_0 := \begin{bmatrix} 0 & I & & & \\ I & 0 & I & & \\ & \ddots & \ddots & \ddots & \\ & & I & 0 & I \\ -P_0 & \cdots & -P_{d-3} & P_d - P_{d-2} & -P_{d-1} \end{bmatrix}, \quad \mathcal{L}_1 := \begin{bmatrix} I & & & & \\ & 2I & & & \\ & & 2I & & \\ & & & \ddots & \\ & & & & 2P_d \end{bmatrix}. \tag{3.10}$$

*Proof.* A value $\lambda \in \mathbb{C}$ is an eigenvalue of the matrix polynomial $P(\lambda)$ if:

$$\left(P_0T_0(\lambda) + \ldots + P_dT_d(\lambda)\right) v = 0, \tag{3.11}$$

where $v \in \mathbb{C}^m$ is the eigenvector associated with $\lambda$. Considering the vectors $v_k := T_k(\lambda)v$ for $k = 0, 1, \ldots, d$, the problem (3.11) can be rewritten as:

$$P_0v_0 + \ldots + P_dv_d = 0. \tag{3.12}$$

Recalling the three-term recurrence (1.11) for the Chebyshev polynomials of the first kind, we have the following relations for the vectors $v_k$:

$$\begin{cases} v_1 = \lambda v_0, \\ v_k = 2\lambda v_{k-1} - v_{k-2}, \quad \text{for } k = 2, \ldots, d. \end{cases} \tag{3.13}$$

Employing this relation for $k = d$, we can delete the vector $v_d$ from the eigenvalue problem (3.12) and obtain:

$$P_0 v_0 + \ldots + P_{d-3} v_{d-3} + (P_{d-2} - P_d)\, v_{d-2} + P_{d-1} v_{d-1} + 2\lambda P_d v_{d-1} = 0. \quad (3.14)$$

The final step consists in reformulating this problem as the generalized linear eigenvalue problem

$$\mathcal{L}_0 w = \lambda \mathcal{L}_1 w, \quad (3.15)$$

where we choose $\mathcal{L}_0$ and $\mathcal{L}_1$ as in (3.10) and we define the vector $w$ as

$$w := \begin{bmatrix} v_0 \\ v_1 \\ \vdots \\ v_{d-1} \end{bmatrix} \in \mathbb{C}^{md}. \quad (3.16)$$

At this point, we show that the problem (3.11) is equivalent to the generalized problem (3.15) associated with the pencil $(\mathcal{L}_0, \mathcal{L}_1)$. From the previous construction, it is straightforward to prove that if $\lambda$ is an eigenvalue of $P(\lambda)$ with eigenvector $v$, then the pair $(\lambda, w)$ solves the generalized eigenvalue problem (3.15).

Furthermore, consider a generalized eigenvector $u \in \mathbb{C}^{md}$ for the problem (3.15) associated to the eigenvalue $\lambda \in \mathbb{C}$

$$u = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{d-1} \end{bmatrix} \in \mathbb{C}^{md}, \quad \text{where } u_i \in \mathbb{C}^m. \quad (3.17)$$

Hence starting from the equation

$$\begin{bmatrix} \lambda I & -I & & & \\ -I & 2\lambda I & -I & & \\ & \ddots & \ddots & \ddots & \\ & & -I & 2\lambda I & -I \\ P_0 & \cdots & P_{d-3} & -(P_d - P_{d-2}) & 2\lambda P_d + P_{d-1} \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{d-2} \\ u_{d-1} \end{bmatrix} = 0, \quad (3.18)$$

we have the following relations:

$$\begin{aligned} u_1 &= \lambda u_0, \\ -u_0 + 2\lambda u_1 - u_2 &= 0, \\ &\vdots \\ -u_{d-4} + 2\lambda u_{d-3} - u_{d-2} &= 0. \end{aligned} \quad (3.19)$$

From this three-term recurrence, we deduce that $u_k = T_k(\lambda) u_0$ for $k = 0, \ldots, d-1$. Using the last block-row of (3.18) and employing that $u_k = T_k(\lambda) u_0$ for $k = 0, \ldots, d-1$, we have that:

$$\begin{aligned} 0 &= P_0 u_0 + \ldots + P_{d-3} u_{d-3} - (P_d - P_{d-2})\, u_{d-2} + (2\lambda P_d + P_{d-1})\, u_{d-1} \\ &= (P_0 + \ldots + P_{d-3} T_{d-3}(\lambda) - (P_d - P_{d-2})\, T_{d-2}(\lambda) + (2\lambda P_d + P_{d-1})\, T_{d-1}(\lambda))\, u_0 \\ &= (P_0 + \ldots + P_{d-1} T_{d-1}(\lambda))\, u_0 + P_d\, (2\lambda T_{d-1}(\lambda) - T_{d-2}(\lambda))\, u_0 \\ &= (P_0 + P_1 T_1(\lambda) + \ldots + P_{d-1} T_{d-1}(\lambda) + P_d T_d(\lambda))\, u_0. \end{aligned}$$

This is enough to conclude that $\lambda$ is an eigenvalue for the matrix polynomial $P(\lambda)$ and $u_0$ is the corresponding eigenvector. Moreover, the vector $u_0 \in \mathbb{C}^m$ is nonzero: if $u_0 = 0$, then we have that the vector $u = 0$. □

We refer to the pencil $\lambda\mathcal{L}_1 - \mathcal{L}_0$ as a *colleague matrix pencil*. Moreover the pencil $\mathcal{L}_0 - \lambda\mathcal{L}_1$ is a linearization for the polynomial eigenvalue problem associated with $P(\lambda)$ [1]. Similar linearizations exist for different degree-graded bases and the pencil associated with the generalized eigenvalue problem has a block structure similar to the one presented for the Chebyshev basis.

Returning to the rootfinding problem, we employ the colleague pencil in order to compute all the finite eigenvalues of the Bézout resultant matrix $B(y)$ and among them we select the values $y_*$ that are real and belong to the interval $[-1, 1]$. We use the MATLAB command `eig` for pencils to solve the generalized eigenvalue problem $\mathcal{L}_0 w = \lambda\mathcal{L}_1 w$. The command `eig` employs the $QZ$ algorithm in order to solve generalized eigenvalue problems [35]. This step usually takes the majority of the computational cost of the resultant-based method. In fact, the Bézout matrix $B(y)$ is a square matrix polynomial of size $N \times N$ and degree $M$ and hence the colleague pencil $\lambda\mathcal{L}_1 - \mathcal{L}_0$ is a square pencil of size $MN \times MN$. This leads to a generalized eigenvalue problem of size $MN$ and therefore the computational cost of the `eig` command, which is known to be cubic with respect to the size of the pencil, is approximately $\mathcal{O}((MN)^3)$. This computational cost can be reduced incorporating efficient algorithms for the computation of the generalized eigenvalues. We describe the main steps of this approach. In order to compute the generalized eigenvalues of the pencil $\mathcal{L}_0 - \lambda\mathcal{L}_1$ in the form (3.10), we can apply the QR algorithm to the matrix $A := (\mathcal{L}_1)^{-1}\mathcal{L}_0$. The matrix $A$ can be rewritten in the form

$$A = F + UV^H, \tag{3.20}$$

where $F \in \mathbb{C}^{dm \times dm}$ such that $F = F^H$ and $U, V$ are block matrices of size $dm \times m$ such that $UV^H$ has rank equal to $m$. In order to apply an algorithm for the fast computation of QR, we perform an Hessenberg reduction, as proposed in [21], by means of particular Givens rotations. In our case, the structure of the Bézout matrix polynomial $B(y)$ leads to a block Hessenberg matrix. After a step of Hessenberg reduction, we transform the matrix $A$ into a matrix in upper Hessenberg form. At this point, we can apply a generalization of the method proposed in [16], which reduces both the computational cost and the memory storage, with respect to the standard QR method. The use of the QR algorithm could introduce potential stability problems, hence strategies to prevent this issues need to be applied. However, the study of these efficient methods for the computation of the generalized eigenvalues are beyond the scope of this thesis.

The next step of our algorithm consists in finding all the possible values of the variable $x$ for the common zeros of $p(x, y)$ and $q(x, y)$. Then, for each of the previously selected values $y_*$, we solve two independent univariate rootfinding problems, that are:

$$p_{y_*}(x) = p(x, y_*) = 0, \qquad q_{y_*}(x) = q(x, y_*) = 0. \tag{3.21}$$

To find all the solutions of these two univariate scalar rootfinding problems, we rely on the construction of a companion-like pencil and then we use the `eig` command in MATLAB. We employ the colleague pencil proposed in (3.10) for

univariate polynomials expressed in the Chebyshev basis and then we have that the roots of a univariate polynomial are equal to the eigenvalues of its colleague pencil. In this way, for each $y_*$, we obtain two sets of values for the variable $x$ corresponding to $y_*$ and we keep only the values $x_*$ that are real and belong to the interval $[-1, 1]$. If a value $x_*$ appears in both these sets, we have found a common zero $(x_*, y_*)$ of the polynomials $p(x, y)$ and $q(x, y)$.

## 3.2 Construction of the Bézout matrix

A non-trivial part of the resultant-based method consists in the construction of the Bézout matrix and consequently in the computation of the coefficient matrices $A_i$ for $i = 0, 1, \ldots, M$ presented in (3.7). Several techniques can be applied in order to perform these steps of our algorithm. We focus our attention on two possible ideas: the first, proposed in [36], exploits the connection between the Bézout matrices and a particular class of linearizations for matrix polynomials, usually denoted by $\mathbb{DL}$ [37], and the second one is an approach based on interpolation. We choose to incorporate interpolation in the algorithm. This second option is easier to generalize to rootfinding problems in three or more dimensions. Furthermore, efficient techniques are available, as the use of the discrete Fourier transform, in order to perfom the interpolation step.

### 3.2.1 Vector spaces of linearization for matrix polynomials

We introduce several results on the class of linearizations known as ansatz spaces and afterwards we describe how these theoretical achievements can be applied to the Bézout matrix. Consider a regular matrix polynomial of degree $m$ expressed in a degree-graded basis $\{\phi_0, \phi_1, \ldots\}$, that is:

$$P(\lambda) := \sum_{i=0}^{m} P_i \phi_i(\lambda), \quad \text{where } P_i \in \mathbb{R}^{k\times k}. \tag{3.22}$$

Throughout this section, we indicate by $\Lambda(\lambda)$ the column vector $[\phi_{m-1}(\lambda), \phi_{m-2}(\lambda), \ldots, \phi_0(\lambda)]^T$ and by $^{\mathcal{B}}$ the blockwise transpose of a matrix, that is, given an integer $k \geq 1$ and a matrix $X = (X_{ij})_{1\leq i,j\leq m}$, where $X_{ij} \in \mathbb{R}^{k\times k}$, we write $X^{\mathcal{B}} := (X_{ji})_{1\leq i,j\leq m}$.

**Definition 3.2.1.** Consider a matrix polynomial $P(\lambda)$. We define two vector spaces $\mathbb{L}_1(P)$ and $\mathbb{L}_2(P)$ associated with $P(\lambda)$ [33]:

$$\mathbb{L}_1(P) = \Big\{ L(\lambda) = \lambda X + Y \in \mathbb{R}[\lambda]^{km\times km}, \\ L(\lambda)\cdot(\Lambda(\lambda)\otimes I_k) = v \otimes P(\lambda), v \in \mathbb{R}^m \Big\}. \tag{3.23}$$

The vector $v \in \mathbb{R}^m$ is called the *right ansatz* vector.

$$\mathbb{L}_2(P) = \Big\{ L(\lambda) = \lambda X + Y \in \mathbb{R}[\lambda]^{km\times km}, \\ \big(\Lambda(\lambda)^T \otimes I_k\big)\cdot L(\lambda) = w^T \otimes P(\lambda), w \in \mathbb{R}^m \Big\}. \tag{3.24}$$

The vector $w \in \mathbb{R}^m$ is called the *left ansatz* vector.

These vector spaces can be rewritten using a characterization of the action of the pencil $L(\lambda)$ on the matrices $(\Lambda(\lambda) \otimes I_k)$ and $\left(\Lambda(\lambda)^T \otimes I_k\right)$, which allows us to work on block matrices. For this reason, we introduce the *column shifted sum* and the *row shifted sum*.

**Definition 3.2.2.** Consider $X$ and $Y$ two block matrices

$$X = \begin{bmatrix} X_{11} & \cdots & X_{1m} \\ \vdots & & \vdots \\ X_{m1} & \cdots & X_{mm} \end{bmatrix}, \quad Y = \begin{bmatrix} Y_{11} & \cdots & Y_{1m} \\ \vdots & & \vdots \\ Y_{m1} & \cdots & Y_{mm} \end{bmatrix}, \tag{3.25}$$

where $X_{ij}, Y_{ij} \in \mathbb{R}^{k\times k}$ for $i, j = 1, \ldots, m$. Define the *column shifted sum* of $X$ and $Y$ as:

$$X \boxplus Y := XM + [\mathbf{0} \quad Y], \tag{3.26}$$

and the *row shifted sum* of $X$ and $Y$ as:

$$X \underset{\vdots}{\boxplus} Y := M^{\mathcal{B}} X + \begin{bmatrix} \mathbf{0}^T \\ Y \end{bmatrix}, \tag{3.27}$$

where $\mathbf{0} \in \mathbb{R}^{km\times k}$ and $M \in \mathbb{R}^{km\times k(m+1)}$ is a block matrix such that $M_{pq} = m_{p,q} I_k$ for $1 \le p \le m$ and $1 \le q \le m+1$ and $m_{p,q}$ is defined as

$$x\phi_{i-1}(x) = \sum_{j=0}^{m} m_{m+1-i,m+1-j}\phi_j(x), \quad 1 \le i \le m. \tag{3.28}$$

The matrix $M$ can be written as

$$M = \begin{bmatrix} M_{11} & M_{12} & \cdots & M_{1m} & M_{1,m+1} \\ \mathbf{0} & M_{22} & \ddots & \ddots & M_{2,m+1} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{0} & M_{mm} & M_{m,m+1} \end{bmatrix}. \tag{3.29}$$

When dealing with matrix polynomials (3.22) written in the Chebyshev basis, the matrix $M$ (3.29) has an easier form, that is

$$M = \begin{bmatrix} \frac{1}{2}I_k & 0 & \frac{1}{2}I_k & & \\ & \ddots & \ddots & \ddots & \\ & & \frac{1}{2}I_k & 0 & \frac{1}{2}I_k \\ & & & I_k & 0 \end{bmatrix} \in \mathbb{R}^{km\times k(m+1)}. \tag{3.30}$$

A block matrix can be used in order to express a bivariate matrix polynomial. Consider a block matrix $X \in \mathbb{R}^{km\times kn}$ with $X_{ij}$, for $i = 1, \ldots, m$ and $j = 1, \ldots, n$ that are blocks of size $k \times k$ and let $\{\phi_0, \phi_1, \ldots\}$ be a degree-graded basis. We can define the map

$$\varphi : X = \begin{bmatrix} X_{11} & \cdots & X_{1n} \\ \vdots & \ddots & \vdots \\ X_{m1} & \cdots & X_{mn} \end{bmatrix} \mapsto F(x, y) = \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} X_{m-i,n-j}\phi_i(y)\phi_j(x). \tag{3.31}$$

The map $\varphi$ provides a correspondence between a block matrix and a bivariate matrix polynomial. In particular, we can employ block matrices in order to obtain the coefficients of a matrix polynomial in two variables. This relation can be used to recast the definitions of the vector spaces $\mathbb{L}_1(P)$ and $\mathbb{L}_2(P)$ in terms of operations among two bivariate matrix polynomials. The connection between the action of the pencils $L(\lambda)$ and the operations on block matrices is highlighted in the following proposition [33].

**Proposition 3.2.3.** *Consider a $k \times k$ matrix polynomial $P(\lambda) = \sum_{i=0}^{m} P_i \phi_i(\lambda)$ of degree $m$ and a pencil $L(\lambda) = \lambda X + Y$ of size $mk \times mk$. Then for $v \in \mathbb{R}^m$ it holds:*

$$(\lambda X + Y) \cdot (\Lambda \otimes I_k) = v \otimes P(\lambda) \Longleftrightarrow X \boxplus Y = v \otimes [P_m \; P_{m-1} \; \cdots \; P_0], \quad (3.32)$$

*then $\mathbb{L}_1(P)$ can be seen as*

$$\mathbb{L}_1(P) = \{\lambda X + Y : \; X \boxplus Y = v \otimes [P_m \; P_{m-1} \; \cdots \; P_0], \; v \in \mathbb{R}^m\}. \quad (3.33)$$

*In the same way, let $w$ be a vector in $\mathbb{R}^m$, we have:*

$$\left(\Lambda^T \otimes I_k\right) \cdot (\lambda X + Y) = w^T \otimes P(\lambda) \iff X \underset{\downarrow}{\boxplus} Y = w^T \otimes \begin{bmatrix} P_m \\ \vdots \\ P_0 \end{bmatrix}, \quad (3.34)$$

*then $\mathbb{L}_2(P)$ can be written as:*

$$\mathbb{L}_2(P) = \left\{\lambda X + Y : \; X \underset{\downarrow}{\boxplus} Y = w^T \otimes [P_m \; \cdots \; P_0]^{\mathcal{B}}, \; w \in \mathbb{R}^m\right\}. \quad (3.35)$$

The matrix $M$ (3.29) can be used in order to represent operations on bivariate polynomials. For instance, consider a block matrix $X$ that represents the bivariate matrix polynomial $F(x, y)$ through the map $\varphi$ (3.31), then it can be proved that the matrix product $XM$ corresponds to the multiplication $F(x, y)x$. Similarly, given the block matrices $X$ and $Y$ that represent the coefficients of the matrix polynomials $F(x, y)$ and $G(x, y)$, it can be proved that the coefficients of $H(x, y) := F(x, y)x + G(x, y)$ are the blocks of the matrix $Z := X \boxplus Y$. Similarly, the matrix $M^{\mathcal{B}} X$ corresponds to the coefficients of the matrix polynomial $yF(x, y)$ and, given the block matrices $X$ and $Y$ corresponding to the coefficients of $F(x, y)$ and $G(x, y)$, we have that the block matrix $Z := X \underset{\downarrow}{\boxplus} Y$ collects the coefficients of $H(x, y) := yF(x, y) + G(x, y)$. These operations on the block matrices, combined with Proposition 3.2.3, lead to a characterization of the vector spaces $\mathbb{L}_1(P)$ and $\mathbb{L}_2(P)$ through operations on bivariate polynomials.

**Proposition 3.2.4.** *Consider the matrix polynomial $P(\lambda)$ in* (3.22)*. The space $\mathbb{L}_1(P)$ can be written as*

$$\mathbb{L}_1(P) = \{L(\lambda) = \lambda X + Y : F(x, y)x + G(x, y) = v(y)P(x), \; v \in \Pi_{m-1}(\mathbb{R})\}, \quad (3.36)$$

*where we indicate by $\Pi_{m-1}(\mathbb{R})$ the space of univariate polynomials in $\mathbb{R}[y]$ of degree at most $m - 1$, and $F(x, y)$ and $G(x, y)$ are the matrix polynomials corresponding to the block matrices $X$ and $Y$, through the map $\varphi$* (3.31)*. Moreover,*

*writing the polynomial* $v(y) = \sum_{i=0}^{m-1} v_i \phi_i(y)$*, the right ansatz vector is given by* $v := [v_{m-1}, \cdots, v_0]^T$*. In the same way, the space* $\mathbb{L}_2(P)$ *can be written as*

$$\mathbb{L}_2(P) = \{L(\lambda) = \lambda X + Y : yF(x,y) + G(x,y) = P(y)w(x),\ w \in \Pi_{m-1}(\mathbb{R})\}. \tag{3.37}$$

*In this case, writing the polynomial* $w(y) = \sum_{i=0}^{m-1} w_i \phi_i(y)$*, the left ansatz vector is given by* $w := [w_{m-1}, \cdots, w_0]^T$*.*

This relation is useful in order to construct a further vector space of possible linearizations.

**Definition 3.2.5.** Given a $k \times k$ matrix polynomial $P(\lambda)$ of degree $m$, defined as in (3.22), we define the *double ansatz space* associated with $P$ as:

$$\mathbb{DL}(P) := \mathbb{L}_1(P) \cap \mathbb{L}_2(P). \tag{3.38}$$

In particular, using the characterization in Proposition (3.2.4), it holds that for any pencil $L(\lambda)$ in $\mathbb{DL}(P)$ the right and the left ansatz vectors are equal. Furthermore, it can also be proved that the pencils in $\mathbb{DL}(P)$ are block symmetric [28].

At this point, we provide some remarks that relate the Bézoutian function with the linearizations in $\mathbb{DL}(P)$. In [37], the authors present a result for $k \times k$ matrix polynomials $P(\lambda)$ in the form (3.22) and they find a correlation between linearizations of the matrix polynomial and the Lérer-Tismenetsky Bézoutian function, which is a generalization of the Bézoutian function (2.12) for matrix polynomials. In our case, we can restrict the result to scalar polynomials. In fact, in the algorithm for bivariate rootfinding through the hidden variable technique, we need to compute the Bézout matrix associated with two univariate scalar polynomials.

**Proposition 3.2.6.** *Consider a univariate scalar polynomial* $p(\lambda)$ *of degree* $m$ *expressed in a degree-graded basis* $\{\phi_0, \phi_1, \ldots\}$ *and let* $L(\lambda) \in \mathbb{DL}(p)$ *a pencil with ansatz* $v \in \Pi_{m-1}$*. Then we have that:*

$$L(\lambda) = \widetilde{B}\left((x - \lambda)v, p\right), \tag{3.39}$$

*where* $\widetilde{B} := B^{\mathcal{B}}$ *and* $B$ *is the Bézout matrix* (2.13) *expressed in the basis* $\{\phi_0, \phi_1, \ldots\}$*.*

*Proof.* Consider the scalar polynomial $p(\lambda)$ and let $L(\lambda) = \lambda X + Y$ be the matrix pencil in $\mathbb{DL}(p)$ associated with $v(\lambda)$. Then, $F(x,y)$ and $G(x,y)$, that are the bivariate polynomials associated with the block matrices $X$ and $Y$ through the map $\varphi$, satisfy the formulas:

$$\begin{aligned} yF(x,y) - F(x,y)x &= p(y)v(x) - v(y)p(x) \\ yG(x,y) - G(x,y)x &= yv(y)p(x) - p(y)v(x)x. \end{aligned} \tag{3.40}$$

In this way, we obtain that the two bivariate polynomials can be written as

$$F(x,y) = \frac{p(y)v(x) - v(y)p(x)}{y - x}, \quad G(x,y) = \frac{yv(y)p(x) - p(y)v(x)x}{y - x}, \tag{3.41}$$

hence, using the definition (2.12) of the Bézout function, we have that $F(x,y) = \mathcal{B}(p,v)$ and $G(x,y) = \mathcal{B}(xv,p)$. Moreover, recalling that $\mathcal{B}$ is skew-symmetric, we obtain

$$\begin{aligned} L(\lambda) = \lambda X + Y = \lambda \widetilde{B}(p,v) + \widetilde{B}(xv,p) = \\ = -\lambda \widetilde{B}(v,p) + \widetilde{B}(xv,p) = \widetilde{B}\left((x-\lambda)v,p\right). \end{aligned} \tag{3.42}$$

□

The previous proposition provides a clear connection between the matrix pencil in the double ansatz vector space $\mathbb{DL}(p)$ and the Bézout matrix associated with two scalar polynomials. In particular, using the equation (3.42), we have that, in order to compute the Bézout matrix between the univariate scalar polynomials $p(\lambda)$ and $v(\lambda)$, we can construct the pencil $L(\lambda) = \lambda X + Y$ associated with the ansatz polynomial $v$ and then select the matrix $X$. This observation is particularly useful, because there exists an easy way to compute the matrix pencil $L(\lambda) = \lambda X + Y$. In fact, to this end we propose a method that exploits the correspondence between the operations of bivariate matrix polynomials and the row and the column shift operators defined on block matrices, as described before.

Throughout this section, we consider polynomials expressed through an orthogonal basis $\{\phi_0, \phi_1, \ldots\}$. Recall that we need an approach for the construction of the Bézout matrix expressed in the Chebyshev basis, which is an orthogonal basis, as mentioned in Chapter 1. For this reason, it is sufficient to present a method for problems involving polynomials expressed in an orthogonal basis.

Consider the scalar univariate polynomials $p(\lambda) = \sum_{i=0}^{m} p_i \phi_i(\lambda)$ and $v(\lambda) = \sum_{i=0}^{m-1} v_i \phi_i(\lambda)$. Using the equations (3.40) and the correspondence with the operators defined on block matrices, it can be proved that the coefficient matrices $X, Y$ of the pencil $L(\lambda)$ satisfy the following matrix equations:

$$\begin{aligned} \begin{bmatrix} 0 \\ Y \end{bmatrix} M - M^T \begin{bmatrix} 0 & Y \end{bmatrix} = TM - M^T S, \\ XM = T - \begin{bmatrix} 0 & Y \end{bmatrix}, \end{aligned} \tag{3.43}$$

where the matrix $M$ represents the shift operation (3.29) and the matrices $S$ and $T$ are defined as:

$$S = \begin{bmatrix} v_{m-1} \\ \vdots \\ v_0 \end{bmatrix} \otimes \begin{bmatrix} p_m, \ldots, p_0 \end{bmatrix}, \quad \text{and} \quad T = \begin{bmatrix} v_{m-1}, \ldots, v_0 \end{bmatrix} \otimes \begin{bmatrix} p_m \\ \vdots \\ p_0 \end{bmatrix}. \tag{3.44}$$

Note that the matrices $S$ and $T$ represent the coefficients of the bivariate functions $p(y)v(x)$ and $v(y)p(x)$, respectively. The main idea consists in solving the first equation in (3.43) and computing the matrix $Y$. Then, once $Y$ is obtained, we compute $X$ solving the second matrix equation. The Sylvester equations in (3.43) involve rectangular matrices, as the matrix $M$. The first equation is singular Sylvester equation, for which the conditions for esistence of a unique generic solution are non-trivial. Hence, we force the zero block in the first columns of the solution, that is $\begin{bmatrix} 0 & Y \end{bmatrix}$. In order to prove that the solution exists and is unique, we propose a direct construction of the matrix $Y$. The

**Algorithm 2** Construction of $\lambda X + Y$

**Input**: vector of coefficients $p := [p_m, \dots, p_0]$ and $v := [v_{m-1}, \dots, v_0]$
**Output**: pair $(X, Y)$

**construct** matrices $S$, $T$ and $M$
**set** $R = TM - M^T S$
**set** $Y_1 = -\frac{1}{m_{1,1}} R_1$
**set** $Y_2 = \frac{1}{m_{2,2}} (Y_1 M - m_{1,2} Y_1 - R_2)$
**for** $i = 3 : m$ **do**
    **set** $Y_i = \frac{1}{m_{i,i}} (Y_{i-1} M - m_{i-2,i} Y_{i-2} - m_{i-1,i} Y_{i-1} - R_i)$
**end for**
**solve** $XM = T - [0 \quad Y]$

computation of the matrices $X$ and $Y$ proceeds row by row, starting from the first one. Let us explain in detail the method.

We indicate by $Y_i$ and $R_i$ for $i = 1, \dots, m$ the rows of the matrices $Y$ and $R$, respectively. Since we deal with scalar polynomials expressed in an orthogonal basis, the matrix $M$ defined in (3.29) has only three non-zero diagonals

$$M = \begin{bmatrix} m_{1,1} & m_{1,2} & m_{1,3} & \cdots & 0 \\ 0 & m_{2,2} & \ddots & \ddots & \\ \vdots & \ddots & \ddots & \ddots & \ddots \\ 0 & \cdots & 0 & m_{m,m} & m_{m,m+1} \end{bmatrix} \in \mathbb{R}^{m \times (m+1)}. \tag{3.45}$$

In this way, the algorithm constucts the matrix pair $(X, Y)$, that forms the matrix pencil $L(\lambda)$.

Following Proposition (3.2.6), we have that the matrix $X$ corresponds to the entries of the Bézout matrix $B(v, p)$ associated with the polynomials $v$ and $p$. Moreover, it can be proved that the computational cost of this algorithm is $\mathcal{O}(m^2)$, where $m$ is the degree of the polynomial $p(\lambda)$ [37].

### 3.2.2 Interpolation in three dimensions

In this section, we describe a different approach that can be employed in order to construct the Bézoutian matrix associated with two polynomials. Let us recall the main steps for the construction of the Bézout resultant matrix. Consider two bivariate polynomials $p(x, y)$ and $q(x, y)$ of degrees $n_p$ and $n_q$ in the variable $x$ and of degrees $m_p$ and $m_q$ in the variable $y$, respectively. The application of the hidden variable technique leads to two polynomials in the variable $x$ with coefficients in $\mathbb{R}[y]$:

$$p_y(x) = \sum_{j=0}^{n_p} \alpha_j(y) T_j(y), \quad q_y(x) = \sum_{j=0}^{n_q} \beta_j(y) T_j(x), \tag{3.46}$$

where $x \in [-1, 1]$ and $\alpha_j, \beta_j \in \mathbb{R}[y]$. The Chebyshev Bézout matrix associated with $p_y$ and $q_y$ is defined as the matrix $B(p_y, q_y) = (b_{ij})_{0 \le i,j \le N-1}$, where $N = \max(n_p, n_q)$ and the entries satisfy

$$\frac{p_y(s)q_y(t) - p_y(t)q_y(s)}{s-t} = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} b_{ij}T_i(s)T_j(t). \tag{3.47}$$

As previously observed, the entries $b_{ij}$ of the Bézout matrix are univariate scalar polynomials in the variable $y$, so we rewrite the relation (3.47) as

$$f(s,t,y) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\sum_{k=0}^{M} D_{ijk}T_i(s)T_j(t)T_k(y) \quad M = m_p + m_q, \tag{3.48}$$

where we define $f(s,t,y) = \frac{p_y(s)q_y(t)-p_y(t)q_y(s)}{s-t}$ and $D \in \mathbb{R}^{N\times N\times(M+1)}$ is the three dimensional tensor of the coefficients of the polynomial $f(s,t,y)$ in three variables. Our aim is the construction of the tensor $D$ of the coefficients. A possible approach relies on the use of interpolation. Let us explain our method. The first step consists in the choice of a three dimensional grid of points

$$(s_i, t_j, y_k), \quad \text{for } i,j = 0,\dots,N-1,\ k = 0,\dots,M. \tag{3.49}$$

We choose the grid formed by the Chebyshev points, as defined in Chapter 1. In fact, we are dealing with polynomials expressed in the Chebyshev basis, hence the choice of the Chebyshev points (1.33) is particularly useful, since we can apply techniques such as the Fast Fourier Transform, in order to speed up the calculation. Then we choose $s_0,\dots,s_{N-1}$ and $t_0,\dots,t_{N-1}$ as the $N$ Chebyshev points, and $y_0,\dots,y_M$ as the $M$ Chebyshev points.

Now we need to evaluate the function $f(s,t,y)$ at each point of the three dimensional grid $(s_i,t_j,y_k)_{i,j,k}$ and we construct a three dimensional tensor $C \in \mathbb{R}^{N\times N\times(M+1)}$ such that

$$C_{ijk} := f(s_j,t_j,y_k), \quad \text{for } i,j = 0,\dots,N-1, \text{ and } k = 0,\dots,M. \tag{3.50}$$

Note that the denominator of the multivariate function $f(s,t,y)$ is equal to zero if $s$ is equal to $t$. Then, we decide to treat separately the case in which the points $s_i$ and $t_j$ are equal. Consider a point $(s_i,t_j,y_k)$ where $i \neq j$ and therefore the points $s_i$ and $t_j$ are different. In this case, we evaluate the polynomial $f(s,t,y)$ at the points of the grid $(s_i,t_j,y_k)$ for $k = 0,\dots,M$. A possible way to proceed consists in evaluating the functions $p_y$, $q_y$ and $\frac{1}{s-t}$ separately and then combining them. Instead, the evaluation of $f(s,t,y)$ at points $(s_i,t_i,y_k)$ needs to be treated with a different approach. One possible way consists in computing the limit of $f(s,t,y)$ for $s \to t$, that is

$$\begin{aligned}\lim_{s\to t}\frac{p_y(s)q_y(t) - p_y(t)q_y(s)}{s-t} &= \lim_{s\to t}\frac{p_y(s)q_y(t) - p_y(s)q_y(s) + p_y(s)q_y(s) - p_y(t)q_y(s)}{s-t}\\ &= \lim_{s\to t}\frac{p_y(s)\left(q_y(t)-q_y(s)\right)}{s-t} + \frac{q_y(s)\left(p_y(s)-p_y(t)\right)}{s-t}\\ &= q_y(t)p_y'(t) - p_y(t)q_y'(t).\end{aligned}$$

Then when we consider the points $(s_i,t_i,y_k)$ for $i = 0,\dots,N-1$ and $k = 0,\dots,M$, we evaluate the multivariate function $q_y(t)p_y'(t) - p_y(t)q_y'(t)$. To this

end, recall that Proposition 1.0.5 provides an easy expression of the derivative of the Chebyshev polynomials of the first kind, employing the set of the Chebyshev polynomials of the second kind. Once we have evaluated the function at every point of the three dimensional grid, we obtain the tensor $C$ of size $N \times N \times (M+1)$.

At this point, we construct the tensor $D \in \mathbb{R}^{N \times N \times (M+1)}$ of the coefficients. In order to perform the interpolation step, we propose to permute the entries of the tensor $C$ and to treat every variable separately. In the following process, we frequently need to use interpolation in one dimension. In particular, we deal with transformations of the type

$$z \mapsto V^{-1} z, \quad z \in \mathbb{R}^d, \tag{3.51}$$

where the matrix $V$ is the generalized Vandermonde matrix in the Chebyshev basis (1.38) of size $d \times d$. This step can be solved in an efficient way using the discrete Fourier transform and requires $\mathcal{O}(d \log d)$ operations [27]. In a similar way, we can perform the transformation

$$Z \mapsto V^{-1} Z, \quad Z \in \mathbb{R}^{d_1 \times d_2}, \tag{3.52}$$

where $V$ is the generalized Vandermonde matrix in the Chebyshev basis of size $d_1 \times d_1$. In fact we can apply the discrete Fourier transform to each column of the matrix $Z$, in order to obtain the final matrix $V^{-1}Z$. Hence, this operation requires $\mathcal{O}(d_1 d_2 \log(d_1))$ operations.

Let us explain in detail how we can construct the tensor $D$ of coefficients, using the tensor $C$ of evaluations at the Chebyshev points. Recall that the tensor $D$ satisfies the relation

$$f(s,t,y) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \sum_{k=0}^{M} D_{ijk} T_i(s) T_j(t) T_k(y). \tag{3.53}$$

Fix the variables $t$ and $y$, for instance set $t := t_0$ and $y := y_0$. We obtain a univariate function in $s$, which can be written as

$$f(s,t_0,y_0) = \sum_{i=0}^{N-1} E_i^{(0,0)} T_i(s), \tag{3.54}$$

where the coefficients $E_i^{(0,0)}$, for $i = 0, \ldots, N-1$ satisfy the relation

$$E_i^{(0,0)} = \sum_{j=0}^{N-1} \sum_{k=0}^{M} D_{ijk} T_j(t_0) T_k(y_0). \tag{3.55}$$

At this point, we apply a step of interpolation on the univariate function in (3.54). In fact, the evaluations of $f(s,t_0,y_0)$ at the Chebyshev points $s_0, \ldots, s_{N-1}$ are equal to the entries $C_{000}, \ldots, C_{(N-1)00}$ of the previously computed tensor $C$. Then, in order to compute the coefficients $E_0^{(0,0)}, \ldots, E_{N-1}^{(0,0)}$ of the polynomial $f(s,t_0,y_0)$, we solve the linear system introduced in Section 1.2, that is

$$\begin{bmatrix} 1 & T_1(s_0) & \cdots & T_{N-1}(s_0) \\ 1 & T_1(s_1) & \cdots & T_{N-1}(s_1) \\ \vdots & \vdots & \cdots & \vdots \\ 1 & T_1(s_{N-1}) & \cdots & T_{N-1}(s_{N-1}) \end{bmatrix} \begin{bmatrix} E_0^{(0,0)} \\ E_1^{(0,0)} \\ \vdots \\ E_{N-1}^{(0,0)} \end{bmatrix} = \begin{bmatrix} C_{000} \\ C_{100} \\ \vdots \\ C_{(N-1)00} \end{bmatrix}. \quad (3.56)$$

We repeat this process for every choice of the indices $h = 0, \dots, N-1$ and $l = 0, \dots, M$, that is for every choice of the points $t_h$ and $y_l$ in the sets $t_0, \dots, t_{N-1}$ and $y_0, \dots, y_M$, respectively. In general, consider the Chebyshev points $t_h$ and $y_l$. Then, we obtain the univariate interpolation problem

$$f(s, t_h, y_l) = \sum_{i=0}^{N-1} E_i^{(h,l)} T_i(s), \quad (3.57)$$

where the coefficients satisfy the relation for $i = 0, \dots, N-1$

$$E_i^{(h,l)} = \sum_{j=0}^{N-1} \sum_{k=0}^{M} D_{ijk} T_j(t_h) T_k(y_l). \quad (3.58)$$

As mentioned before, the interpolation problem can be solved through the linear system

$$\begin{bmatrix} 1 & T_1(s_0) & \cdots & T_{N-1}(s_0) \\ 1 & T_1(s_1) & \cdots & T_{N-1}(s_1) \\ \vdots & \vdots & \cdots & \vdots \\ 1 & T_1(s_{N-1}) & \cdots & T_{N-1}(s_{N-1}) \end{bmatrix} \begin{bmatrix} E_0^{(h,l)} \\ E_1^{(h,l)} \\ \vdots \\ E_{N-1}^{(h,l)} \end{bmatrix} = \begin{bmatrix} C_{0hl} \\ C_{1hl} \\ \vdots \\ C_{(N-1)hl} \end{bmatrix}. \quad (3.59)$$

Note that this entire process can be performed in an equivalent way solving the matrix equation $V_s E_s = C_s$, where we indicate by $V_s$ the generalized Vandermonde matrix (1.38) associated with the Chebyshev points $s_0, \dots, s_{N-1}$ and the matrix $C_s$ is defined as

$$C_s := \begin{bmatrix} C_{000} & C_{010} & \cdots & C_{0(N-1)(M)} \\ C_{100} & C_{110} & \cdots & C_{1(N-1)(M)} \\ \vdots & & & \vdots \\ C_{(N-1)00} & C_{(N-1)10} & \cdots & C_{(N-1)(N-1)(M)} \end{bmatrix} \in \mathbb{R}^{N \times (N(M+1))}. \quad (3.60)$$

The solution $E_s$ is a matrix of size $N \times (N \cdot (M+1))$ with coefficients

$$E_s := \begin{bmatrix} E_0^{(0,0)} & E_0^{(1,0)} & \cdots & E_0^{((N-1),0)} & E_0^{(0,1)} & \cdots & E_0^{((N-1),M)} \\ E_1^{(0,0)} & E_1^{(1,0)} & \cdots & E_1^{((N-1),0)} & E_1^{(0,1)} & \cdots & E_1^{((N-1),M)} \\ \vdots & & & \vdots & \vdots & & \vdots \\ E_{N-1}^{(0,0)} & E_{N-1}^{(1,0)} & \cdots & E_{N-1}^{((N-1),0)} & E_{N-1}^{(0,1)} & \cdots & E_{N-1}^{((N-1),M)} \end{bmatrix} \quad (3.61)$$

that satisfy the relation (3.58) for every $i = 0, \dots, N-1$, $h = 0, \dots, N-1$ and $l = 0, \dots, M$. Note that the matrix equation $V_s E_s = C_s$ can be solved in an

efficient way through the use of the discrete Fourier transform to each column of the matrix $C_s$. Then, we rearrange the matrix $E_s$ in a tensor of dimensions $N \times N \times (M+1)$.

The next step of our method consists in repeating a similar process on the tensor $E$ and using the relation (3.58), in order to recover the coefficients of the tensor $D$. Consider the bivariate polynomials $g_0(t,y), \ldots, g_{N-1}(t,y)$ defined as

$$g_i(t,y) := \sum_{j=0}^{N-1} \sum_{k=0}^{M} D_{ijk} T_j(t) T_k(y), \quad \text{for } i = 0, \ldots, N-1. \tag{3.62}$$

We fix the variable $y$, choosing an index $l$ among $0, \ldots, M$ and consider the corresponding Chebyshev point $y_l$. For example, starting with $g_0$, we choose the first Chebyshev point $y_0$ and we obtain the univariate polynomial

$$g_0(t, y_0) = \sum_{j=0}^{N-1} F_j^{(0,0)} T_j(t), \tag{3.63}$$

where the coefficients $F_j^{(0,0)}$ satisfy

$$F_j^{(0,0)} := \sum_{k=0}^{M} D_{0jk} T_k(y_0), \quad \text{for } j = 0, \ldots, N-1. \tag{3.64}$$

Then, we apply an interpolation step for the univariate polynomial $g_0(t, y_0)$ and for this reason we consider the evaluations of $g_0(t, y_0)$ at the Chebyshev points $t_0, \ldots, t_{N-1}$, which are equal to $E_0^{(0,0)}, E_0^{(1,0)}, \ldots, E_0^{((N-1),0)}$. Note that these values are stored in the entries $E_{0i0}$ of the three dimensional tensor $E$. We proceed in the same way for each choice of the indices $i = 0, \ldots, N-1$ and $l = 0, \ldots, M$ and obtain the univariate function $g_i(t, y_l)$

$$g_i(t, y_l) = \sum_{j=0}^{N-1} F_j^{(i,l)} T_j(t), \tag{3.65}$$

where the coefficients $F_j^{(i,l)}$ satisfy

$$F_j^{(i,l)} := \sum_{k=0}^{M} D_{ijk} T_k(y_l), \quad \text{for } j = 0, \ldots, N-1. \tag{3.66}$$

This process can be rewritten using a matrix formulation and then we obtain the matrix equation of the type $V_t F_t = E_t$, where $V_t$ is the generalized Vandermonde matrix (1.38) associated with the Chebyshev points $t_0, \ldots, t_M$ and $E_t$ is a matrix of size $N \times (N \cdot (M+1))$ with entries

$$E_t = \begin{bmatrix} E_0^{(0,0)} & E_1^{(0,0)} & \cdots & E_{N-1}^{(0,0)} & E_0^{(0,1)} & \cdots & E_{N-1}^{(0,M)} \\ E_0^{(1,0)} & E_1^{(1,0)} & \cdots & E_{N-1}^{(1,0)} & E_0^{(1,1)} & \cdots & E_{N-1}^{(1,M)} \\ \vdots & & & \vdots & \vdots & & \vdots \\ E_0^{((N-1),0)} & E_1^{((N-1),0)} & \cdots & E_{N-1}^{((N-1),0)} & E_0^{((N-1),1)} & \cdots & E_{N-1}^{((N-1),M)} \end{bmatrix}. \tag{3.67}$$

Note that an easy way to convert the tensor $E$ in the matrix $E_t$ consists in using a combination of the commands `permute` and `reshape` of MATLAB. The resolution matrix $F_t$ is in the form

$$F_t := \begin{bmatrix} F_0^{(0,0)} & F_0^{(1,0)} & \cdots & F_0^{((N-1),0)} & F_0^{(0,1)} & \cdots & F_0^{((N-1),M)} \\ F_1^{(0,0)} & F_1^{(1,0)} & \cdots & F_1^{((N-1),0)} & F_1^{(0,1)} & \cdots & F_1^{((N-1),M)} \\ \vdots & \vdots & & \vdots & \vdots & & \vdots \\ F_{N-1}^{(0,0)} & F_{N-1}^{(1,0)} & \cdots & F_{N-1}^{((N-1),0)} & F_{N-1}^{(0,1)} & \cdots & F_{N-1}^{((N-1),M)} \end{bmatrix}, \quad (3.68)$$

where the entries $F_j^{(i,l)}$ satisfy the relations in (3.66). We rearrange the entries of the matrix $F_t$ in order to obtain the tensor $F \in \mathbb{R}^{N\times N\times(M+1)}$, such that the entry in position $(i,j,k)$ is equal to $F_j^{(i,k)}$, for $i,j = 0,\dots,N-1$ and $k = 0,\dots,M$.

In the end, we treat the variable $y$. Starting from the relation (3.66), we consider the $N^2$ univariate polynomials with coefficients $G_k^{(i,j)} := D_{ijk}$

$$r_{i,j}(y) := \sum_{k=0}^{M} G_k^{(i,j)} T_k(y), \quad \text{for } i,j = 0,\dots,N-1. \quad (3.69)$$

We perform the Chebyshev interpolation of each polynomial, choosing two indices among $0,\dots,N-1$. For instance, consider $i = 0$ and $j = 0$, then we perform the interpolation step on the polynomial $r_{0,0}$, that is, we solve the system

$$\begin{bmatrix} 1 & T_1(y_0) & \cdots & T_M(y_0) \\ 1 & T_1(y_1) & \cdots & T_M(y_1) \\ \vdots & \vdots & & \vdots \\ 1 & T_1(y_M) & \cdots & T_M(y_M) \end{bmatrix} \begin{bmatrix} G_0^{(0,0)} \\ G_1^{(0,0)} \\ \vdots \\ G_M^{(0,0)} \end{bmatrix} = \begin{bmatrix} F_0^{(0,0)} \\ F_0^{(0,1)} \\ \vdots \\ F_0^{(0,N-1)} \end{bmatrix}. \quad (3.70)$$

As explained before, this process can be performed solving the matrix equation $V_y G_y = F_y$, where $V_y$ is the generalized Vandermonde matrix (1.38) associated with the Chebyshev points $y_0,\dots,y_M$ and the matrix $F_y$ has size $(M+1)\times N^2$ and is in the form

$$F_y = \begin{bmatrix} F_0^{(0,0)} & F_1^{(0,0)} & \cdots & F_{N-1}^{(0,0)} & F_0^{(1,0)} & \cdots & F_{N-1}^{(N-1,0)} \\ F_0^{(0,1)} & F_1^{(0,1)} & \cdots & F_{N-1}^{(0,1)} & F_0^{(1,1)} & \cdots & F_{N-1}^{(N-1,1)} \\ \vdots & \vdots & & \vdots & \vdots & & \vdots \\ F_0^{(0,M)} & F_1^{(0,M)} & \cdots & F_{N-1}^{(0,M)} & F_0^{(1,M)} & \cdots & F_{N-1}^{((N-1),M)} \end{bmatrix} \quad (3.71)$$

and it can be constructed from the three dimensional tensor $F$ using the commands `reshape` and `permute` in MATLAB. The matrix $G_y$ is in the form

$$G_y = \begin{bmatrix} G_0^{(0,0)} & G_0^{(0,1)} & \cdots & G_0^{((N-1),(N-1))} \\ G_1^{(0,0)} & G_1^{(0,1)} & \cdots & G_1^{((N-1),(N-1))} \\ \vdots & \vdots & & \vdots \\ G_M^{(0,0)} & G_M^{(0,1)} & \cdots & G_M^{((N-1),(N-1))} \end{bmatrix} \in \mathbb{R}^{(M+1)\times N^2}. \quad (3.72)$$

Our last step consists in rearraing the entries of $G_y$ in a three-dimensional tensor $D \in \mathbb{R}^{N\times N\times(M+1)}$ such that $D_{ijk} = G_k^{(i,j)}$. This process constructs the tensor of coefficients $D$ that satisfies the equation (3.48). Our algorithm proceeds as follows.

**Algorithm 3** Construction of the tensor $D$ via interpolation

**Input**: tensor of the evaluations $C \in \mathbb{R}^{N\times N\times(M+1)}$
**Output**: tensor of the coefficients $D \in \mathbb{R}^{N\times N\times(M+1)}$

1: $C_s =$ `reshape(`$C$`,N,N*(M+1))`
2: **solve** $V_s E_s = C_s$
3: $E =$ `reshape(`$E_s$`,N,N,(M+1))`
4: $E =$ `permute(`$E$`,[2 1 3])`
5: $E_t =$ `reshape(`$E$`,N,N*(M+1))`
6: **solve** $V_t F_t = E_t$
7: $F =$ `reshape(`$F_t$`,N,N,(M+1))`
8: $F =$ `permute(`$F$`,[2 1 3])`
9: $F =$ `permute(`$F$`,[3 2 1])`
10: $F_y =$ `reshape(`$F$`,(M+1),N*N)`
11: **solve** $V_y G_y = F_y$
12: $G =$ `reshape(`$G_y$`,N,N,(M+1))`
13: $G =$ `permute(`$G$`,[3 2 1])`
14: **set** $D = G$

In order to estimate the total cost of this method, we consider separately the three main steps involving the majority of the computational cost. Let us start from Step 2 of Algorithm 3, that is, solving the matrix equation $V_s E_s = C_s$. Recall that the $C_s$ is a matrix of size $N \times (N \cdot (M+1))$. As mentioned at the beginning of Section 3.2.2, this solution step can be perfomed using the discrete Fourier transform on each column of the matrix $E_s$ and therefore, it requires $\mathcal{O}(N^2(M+1)\log(N))$ operations. In the same way, we can estimate the computational cost for the steps 6 and 11, which involve the matrices $E_t \in \mathbb{R}^{N\times(N(M+1))}$ and $F_y \in \mathbb{R}^{(M+1)\times N^2}$. Therefore, performing the steps 6 and 11 requires $\mathcal{O}(N^2(M+1)\log(N))$ and $\mathcal{O}(N^2(M+1)\log(M+1))$ operations, respectively. At this point, we conclude that the total computational cost of Algorithm 3 is $\mathcal{O}\left(N^2\,(M+1)\log\left(N^2\,(M+1)\right)\right)$.

# Chapter 4

# Backward error and conditioning

In this chapter, we present an error analysis associated with the resultant-based method described in Chapter 3. Since we deal with finite arithmetic precision, we need an analysis of the error generated by operations in floating point arithmetic. One possible way to perform this study consists in the development of a backward error analysis. To this end, we quickly recall the generalized eigenvalue problem proposed in Section 3.1. Consider two bivariate polynomials $p(x, y)$ and $q(x, y)$, defined for $(x, y) \in [-1, 1]^2$, of degrees $n_p$ and $n_q$ in the variable $x$ and of degrees $m_p$ and $m_q$ in the variable $y$, respectively. Let $B(y)$ be the matrix polynomial defined by (3.7) corresponding to the Bézout resultant matrix

$$B(y) = \sum_{i=0}^{M} A_i T_i(y), \quad y \in [-1, 1], \tag{4.1}$$

where $M = m_p + m_q$ and $A_0, \ldots, A_M$ are square matrices of size $N \times N$, where $N = \max(n_p, n_q)$. We are interested in computing all the eigenvalues of $B(y)$ that are real and belong to the interval $[-1, 1]$. For each eigenvalue $y$ of the matrix polynomial $B(y)$, we compute an approximate eigenvalue $\widetilde{y}$ in floating point arithmetic. Since the computed valued $\widetilde{y}$ does not coincide with the exact eigenvalue $y$, we need a criterion in order to decide if the computed value can be accepted or if it must be rejected.

Let us recall the notions of *forward* and *backward error* in the general case [27]. Consider a function $f : \mathbb{R}^n \mapsto \mathbb{R}^n$ and a value $t$ such that $t = f(s)$ for a certain $s \in \mathbb{R}^n$. In finite precision, we compute an approximation $\widetilde{t}$ of the value $t$. We indicate as backward error the smallest, in absolute value, quantity $\Delta s$ such that $f(s + \Delta s) = \widetilde{t}$ and we refer to the error on the computed value $\widetilde{t}$ as forward error.

Our aim consists in finding a radius of a disk of inclusion, for each approximate eigenvalue. We consider an approximate eigenvalue $\widetilde{y}$ and compute a radius $r_{\widetilde{y}}$ such that for the corresponding eigenvalue $y$ it holds

$$y \in D(\widetilde{y}, r_{\widetilde{y}}), \ \forall y \in \Lambda(B(y)), \tag{4.2}$$

where we denote as $\Lambda(B(y))$ the set of the eigenvalues of the matrix polynomial and where

$$D(\widetilde{y}, r_{\widetilde{y}}) := \left\{ z \in \mathbb{C} : \, |z - \widetilde{y}| \leq r_{\widetilde{y}} \right\}. \tag{4.3}$$

A first order approximation of such $r_{\widetilde{y}}$ may be obtained by estimating the forward error as [27]

$$r_{\widetilde{y}} \cong \kappa(\widetilde{y}) \cdot \eta(\widetilde{y}), \tag{4.4}$$

where $\kappa$ is the condition number of the eigenvalue $\widetilde{y}$ and $\eta$ is the backward error associated with $\widetilde{y}$. If this inclusion disk $D(\widetilde{y}, r_{\widetilde{y}})$ intersects the interval $[-1, 1]$, then we conclude that there exists a solution of the bivariate rootfinding problem in the disk.

We provide the notions of backward error and condition number of an approximate eigenvalue of the matrix polynomial $B(y)$. We start by introducing the definition of *normwise backward error*, following the ones proposed in [19,47]. Given the matrix polynomial $B(y)$, denote as

$$\Delta B(y) := T_0(y)\Delta A_0 + T_1(y)\Delta A_1 + \ldots + T_M(y)\Delta A_M \tag{4.5}$$

a generic perturbation to $B(y)$, where the coefficients $\Delta A_i$ are square matrices of size $N \times N$, for each $i = 0, \ldots, M$.

**Definition 4.0.1.** Let $B(y)$ be a matrix polynomial and $(\widetilde{y}, \widetilde{v})$ be an approximate eigenpair of the eigenvalue problem $B(y)v = 0$. The *normwise backward error* $\eta$ associated with the approximate eigenpair $(\widetilde{y}, \widetilde{v})$ is defined as

$$\begin{aligned} \eta\left(\widetilde{y}, \widetilde{v}\right) := \min \{ \, \epsilon : \, & \left(B(\widetilde{y}) + \Delta B(\widetilde{y})\right) \widetilde{v} = 0, \\ & \|\Delta A_i\|_F \leq \epsilon \| \left[A_M \; A_{M-1} \; \ldots \; A_0\right] \|_F, \; i = 0, \ldots, M \}, \end{aligned} \tag{4.6}$$

where the Frobenius norm of a matrix $A$ is defined as $\|A\|_F := \sqrt{\mathrm{tr}\left(AA^H\right)}$.

In order to measure the perturbations $\Delta A_i$ of the coefficients $A_i$ for any $i = 0, \ldots, M$, we choose the tolerance $\| \left[A_M \; A_{M-1} \; \ldots \; A_0\right] \|_F$. This choice is motivated by the method that we employ in order to compute all the eigenvalues of the matrix polynomial $B(y)$. As presented in Section 3.1, we rely on the construction of the linearization (3.10) and we compute the generalized eigenvalue of the pencil $\mathcal{L}_0 - y\mathcal{L}_1$. This step can be performed using the QZ algorithm, which is a backward stable method. Therefore, the coefficients of the matrix polynomial $B(y)$ are slightly perturbed with respect to the Frobenius norm of the vector of the coefficients [51].

Since the backward error associated with the eigenpair $(\widetilde{y}, \widetilde{v})$ is difficult to calculate from the definition (4.6), we provide an easier expression.

**Proposition 4.0.2.** *The normwise backward error $\eta$ associated with the approximate eigenpair $(\widetilde{y}, \widetilde{v})$ is equal to*

$$\eta\left(\widetilde{y}, \widetilde{v}\right) = \frac{1}{\alpha \| \left[A_M \; A_{M-1} \; \ldots \; A_0\right] \|_F} \frac{\|B(\widetilde{y})\widetilde{v}\|_2}{\|\widetilde{v}\|_2}, \tag{4.7}$$

*where $\alpha := \sum_{i=0}^{M} |T_i(y)|$.*

*Proof.* Denote by $\theta$ the right hand side of (4.7). We start by proving that $\theta$ is a lower bound for the backward error $\eta\left(\widetilde{y},\widetilde{v}\right)$. To this end, we consider a value $\epsilon$ that satisfies the inequalities $\|\Delta A_i\|_F \leq \epsilon\| \left[A_M \; A_{M-1} \; \ldots \; A_0\right] \|_F$, for each $i = 0, \ldots, M$ and such that

$$\left(B(\widetilde{y}) + \Delta B(\widetilde{y})\right) \widetilde{v} = 0. \tag{4.8}$$

Then, using these $\Delta A_i$ for the matrix polynomial $\Delta B(y)$, we start from

$$B(\widetilde{y})\widetilde{v} = -\Delta B(\widetilde{y})\widetilde{v}. \tag{4.9}$$

and computing the 2-norm for each side of (4.9), we obtain

$$\|B(\widetilde{y})\widetilde{v}\|_2 = \|\Delta B(\widetilde{y})\widetilde{v}\|_2 \leq \|\Delta B(\widetilde{y})\|_2 \|\widetilde{v}\|_2. \tag{4.10}$$

At this point, we prove an upper bound for the term $\|\Delta B(\widetilde{y})\|_2$:

$$\begin{aligned}
\|\Delta B(\widetilde{y})\|_2 = \|\sum_{i=0}^{M} \Delta A_i T_i(\widetilde{y})\|_2 &\leq \sum_{i=0}^{M} \|\Delta A_i\|_2 \left|T_i(\widetilde{y})\right| \\
&\leq \sum_{i=0}^{M} \|\Delta A_i\|_F \left|T_i(\widetilde{y})\right| \\
&\leq \epsilon \sum_{i=0}^{M} \| \left[A_M \; A_{M-1} \; \ldots \; A_0\right] \|_F \left|T_i(\widetilde{y})\right| \\
&\leq \epsilon\alpha \| \left[A_M \; A_{M-1} \; \ldots \; A_0\right] \|_F.
\end{aligned}$$

Replacing this estimate in (4.10), we have

$$\|\Delta B(\widetilde{y})\widetilde{v}\|_2 \leq \epsilon\alpha \| \left[A_M \; A_{M-1} \; \ldots \; A_0\right] \|_F \|\widetilde{v}\|_2, \tag{4.11}$$

and hence we obtain the lower bound

$$\epsilon \geq \frac{1}{\alpha \| \left[A_M \; A_{M-1} \; \ldots \; A_0\right] \|_F} \frac{\|B(\widetilde{y})\widetilde{v}\|_2}{\|\widetilde{v}\|_2} = \theta. \tag{4.12}$$

Since by definition

$$\begin{aligned}
\eta\left(\widetilde{y},\widetilde{v}\right) := \min \{ \, \epsilon : \; & \left(B(\widetilde{y}) + \Delta B(\widetilde{y})\right) \widetilde{v} = 0, \\
& \|\Delta A_i\|_F \leq \epsilon \| \left[A_M \; A_{M-1} \; \ldots \; A_0\right] \|_F, \; i = 0, \ldots, M \} \, ,
\end{aligned}$$

we take the minimum on both sides of the inequality to to yield

$$\eta(\widetilde{y},\widetilde{v}) \geq \frac{1}{\alpha \| \left[A_M \; A_{M-1} \; \ldots \; A_0\right] \|_F} \frac{\|B(\widetilde{y})\widetilde{v}\|_2}{\|\widetilde{v}\|_2}. \tag{4.13}$$

In order to prove the equality (4.7), we show that the lower bound $\theta$ is attained for the perturbations

$$\Delta A_i = -\frac{1}{\alpha} \frac{\operatorname{sign}\left(T_i(\widetilde{y})\right) r w^H}{\|\widetilde{v}\|_2}, \qquad \text{for } i = 0, \ldots, M, \tag{4.14}$$

where we denote as $r := B(\widetilde{y})\widetilde{v}$ and the vector $w := \frac{\widetilde{v}^H}{\|\widetilde{v}\|_2}$ that satisfies

$$w^H\widetilde{v} = \|\widetilde{v}\|_2, \qquad \|w^H\|_2 = 1. \tag{4.15}$$

Consider the matrix polynomial $\Delta B(y)$ defined by the coefficients in (4.14). At this point, we verify that the matrix polynomial $\Delta B(y)$ satifies the equality:

$$B(\widetilde{y})\widetilde{v} + \Delta B(\widetilde{y})\widetilde{v} = 0. \tag{4.16}$$

Replacing the matrix coefficients of $\Delta B(y)$ for $i = 0, \ldots, M$, we have

$$\begin{aligned}
&B(\widetilde{y})\widetilde{v} + \Delta B(\widetilde{y})\widetilde{v} = \\
&= B(\widetilde{y})\widetilde{v} - \sum_{i=0}^{M} T_i(\widetilde{y})\frac{1}{\alpha}\frac{\operatorname{sign}\left(T_i(\widetilde{y})\right) r w^H}{\|\widetilde{v}\|_2}\widetilde{v} = \\
&= B(\widetilde{y})\widetilde{v} - \sum_{i=0}^{M} \frac{1}{\alpha}\left|T_i(\widetilde{y})\right| B(\widetilde{y})\widetilde{v}\frac{\|\widetilde{v}\|_2}{\|\widetilde{v}\|_2} = \\
&= B(\widetilde{y})\widetilde{v} - \frac{\sum_{i=0}^{M}\left|T_i(\widetilde{y})\right|}{\alpha} B(\widetilde{y})\widetilde{v} = \\
&= B(\widetilde{y})\widetilde{v} - B(\widetilde{y})\widetilde{v} = 0.
\end{aligned}$$

The last step consists in checking that the matrix polynomial $\Delta B(y)$ satisfies the inequalities in (4.6):

$$\begin{aligned}
\|\Delta A_i\|_F &= \frac{1}{\alpha}\frac{\|B(\widetilde{y})\widetilde{v}\|_2}{\|\widetilde{v}\|_2}\|w^H\|_2 = \\
&= \underbrace{\frac{1}{\alpha\|\left[A_M\ A_{M-1}\ \ldots\ A_0\right]\|_F}\frac{\|B(\widetilde{y})\widetilde{v}\|_2}{\|\widetilde{v}\|_2}}_{=\theta}\|\left[A_M\ A_{M-1}\ \ldots\ A_0\right]\|_F = \\
&= \theta\|\left[A_M\ A_{M-1}\ \ldots\ A_0\right]\|_F.
\end{aligned}$$

Therefore, we conclude that the lower bound $\theta$ is actually reached. □

Definition 4.0.1 is particularly useful when we deal with an approximate eigenpair $(\widetilde{y}, \widetilde{v})$. However, in several situations, the eigenvectors are not computed. For this reason, we propose a different notion of normwise backward error associated with an approximate eigenvalue.

**Definition 4.0.3.** Consider a matrix polynomial $B(y)$ and an approximate eigenvalue $\widetilde{y}$. The *backward error* $\eta$ associated with $\widetilde{y}$ is defined as

$$\eta(\widetilde{y}) = \min_{\widetilde{v}\neq 0} \eta(\widetilde{y}, \widetilde{v}). \tag{4.17}$$

As in the previous definition of backward error, we can rewrite $\eta(\widetilde{y})$ using an expression easier to compute.

**Proposition 4.0.4.** *The backward error $\eta$ associated with an approximate eigenvalue $\widetilde{y}$ is given by*

$$\eta(\widetilde{y}) = \frac{1}{\alpha \| \left[ A_M \; A_{M-1} \; \dots \; A_0 \right] \|_F} \sigma_{\min} \left( B(\widetilde{y}) \right), \tag{4.18}$$

*where* $\alpha := \sum_{i=0}^{M} |T_i(\widetilde{y})|$.

*Proof.* Using Proposition 4.0.2, it holds

$$\begin{aligned} \eta(\widetilde{y}) = & \min_{\widetilde{v} \neq 0} \eta(\widetilde{y}, \widetilde{v}) = \\ & = \min_{\widetilde{v} \neq 0} \frac{1}{\alpha \| \left[ A_M \; A_{M-1} \; \dots \; A_0 \right] \|_F} \frac{\| B(\widetilde{y}) \widetilde{v} \|_2}{\| \widetilde{v} \|_2} = \\ & = \frac{1}{\alpha \| \left[ A_M \; A_{M-1} \; \dots \; A_0 \right] \|_F} \sigma_{\min} \left( B(\widetilde{y}) \right). \end{aligned}$$

□

In order to provide an upper bound for the forward error of the eigenvalue problem $B(y)v = 0$, we need an estimate of the condition number of the eigenvalue $y$. To this end, we give a definition of condition number analogous to the one introduced in [47].

**Definition 4.0.5.** Let $y$ be a nonzero simple eigenvalue of the matrix polynomial $B(y)$ and consider $v$ its corresponding eigenvector. The *normwise condition number* of $y$ is defined by

$$\begin{aligned} \kappa(y, B) = \lim_{\epsilon \to 0} \sup \Bigg\{ \frac{|\Delta y|}{\epsilon |y|} \; : \; & \left( B(y + \Delta y) + \Delta B(y + \Delta y) \right) (v + \Delta v) = 0, \\ & \| \Delta A_i \|_F \leq \epsilon \| \left[ A_M \; A_{M-1} \; \dots \; A_0 \right] \|_F, \; i = 0, \dots, M \Bigg\}. \end{aligned} \tag{4.19}$$

Note that the matrix polynomial $B(y)$ is symmetric, thanks to the symmetry of the Bézout resultant matrix. Hence, for each eigenvalue $y$ the corresponding right eigenvector and left eigenvector coincide.

**Proposition 4.0.6.** *Let $y$ be a simple nonzero eigenvalue of the matrix polynomial $B(y)$ and consider $v$ the corresponding right eigenvector. The normwise condition number $\kappa(y, B)$ is given by*

$$\kappa(y, B) = \frac{\alpha \| \left[ A_M \; A_{M-1} \; \dots \; A_0 \right] \|_F \| v \|_2^2}{|y| \, |v^H B'(y) v|}, \tag{4.20}$$

*where* $\alpha = \sum_{i=0}^{M} |T_i(y)|$ *and* $B'(y) = \sum_{i=0}^{M} A_i (T_i(y))' = \sum_{i=1}^{M} A_i i U_{i-1}(y)$.

*Proof.* We start by proving that the right hand side of (4.20) is an upper bound for the condition number $\kappa$ associated with the eigenvalue $y$. Consider a perturbed matrix polynomial $B + \Delta B$ and the corresponding perturbed eigenpair $(y + \Delta y, v + \Delta v)$

$$\left( B(y + \Delta y) + \Delta B(y + \Delta y) \right) (v + \Delta v) = 0. \tag{4.21}$$

The next step consists in expanding the equation (4.21) and keeping only the first order terms:

$$\begin{aligned} &B(y)(v+\Delta v) + \Delta y B'(y) + \mathcal{O}(\epsilon^2) + \Delta B(y)(v+\Delta v) + \Delta B(\Delta y)(v+\Delta v) = \\ &= B(y)\Delta v + \Delta y B'(y)v + \Delta B(y)v + \mathcal{O}(\epsilon^2) = 0, \end{aligned}$$

where we use that $B(y)v = 0$. Using the equality

$$B(y)\Delta v + \Delta y B'(y)v + \Delta B(y)v = \mathcal{O}(\epsilon^2), \tag{4.22}$$

and multiplying by $v^H$ on the left, we have that

$$v^H \Delta y B'(y)v + v^H \Delta B(y)v = \mathcal{O}(\epsilon^2), \tag{4.23}$$

where we used that $v^H B(y) = 0$. Since the eigenvalue $y$ is simple, we have that the term $v^H \Delta B(y)v \neq 0$ (see Theorem 3.2 in [2]). Then, it holds

$$\Delta y = -\frac{v^H B(y)v}{v^H B'(y)v} + \mathcal{O}(\epsilon^2). \tag{4.24}$$

Recall that $\Delta B(y) = \sum_{i=0}^{M} \Delta A_i T_i(y)$. We now give an upper bound for the numerator of (4.24)

$$\begin{aligned} \left|v^H \Delta B(y)v\right| &\leq \|v\|_2 \|\Delta B\|_2 \|v\|_2 \leq \left(\sum_{i=0}^{M} \|\Delta A_i\|_2 \left|T_i(y)\right|\right) \|v\|_2^2 \\ &\leq \epsilon \| \left[A_M \; A_{M-1} \; \ldots \; A_0\right] \|_F \left(\sum_{i=0}^{M} \left|T_i(y)\right|\right) \|v\|_2^2 \\ &\leq \epsilon \| \left[A_M \; A_{M-1} \; \ldots \; A_0\right] \|_F \alpha \|v\|_2^2. \end{aligned}$$

Using this bound in the relation (4.23), we obtain

$$\begin{aligned} \frac{|\Delta y|}{\epsilon\,|y|} &\leq \frac{\left|v^H \Delta B(y)v\right|}{\epsilon\,|y|\,|v^H B'(y)v|} + \mathcal{O}(\epsilon^2) \\ &\leq \frac{\|v\|_2^2 \; \alpha \; \| \left[A_M \; A_{M-1} \; \ldots \; A_0\right] \|_F}{|y|\,|v^H B'(y)v|} + \mathcal{O}(\epsilon). \end{aligned}$$

In this way, we have that the term $\frac{\|v\|_2^2 \; \alpha \; \|[A_M \; A_{M-1} \; \ldots \; A_0]\|_F}{|y||v^H B'(y)v|}$ is an upper bound for the condition number $\kappa(y, B)$. The last step consists in proving that this bound is attained. Consider a matrix $Q$ such that

$$Q = \frac{vv^H}{\|v\|_2^2}, \tag{4.25}$$

then $Q$ satisfies the properties

$$\|Q\|_2 = 1, \quad \text{and} \quad v^H Q v = \|v\|_2^2. \tag{4.26}$$

We choose the matrices $\Delta A_i$ for $i = 0, \dots, M$ as

$$\Delta A_i = -\text{sign}\left(T_i(y)\right) \epsilon \| \left[A_M \; A_{M-1} \; \dots \; A_0\right] \|_F Q. \tag{4.27}$$

Since $\|\Delta A_i\|_F = \epsilon \| \left[A_M \; A_{M-1} \; \dots \; A_0\right] \|_F$ for $i = 0, \dots, M$, all the inequalities in (4.19) are satisfied and, for this choice of the coefficients, we have

$$\left|v^H \Delta B(y) v\right| = \epsilon \alpha \| \left[A_M \; A_{M-1} \; \dots \; A_0\right] \|_F \|v\|_2^2. \tag{4.28}$$

Using the relation (4.28) and taking the limit for $\epsilon \to 0$, we obtain

$$\frac{|\Delta y|}{\epsilon \, |y|} = \frac{\left|v^H \Delta B(y) v\right|}{\epsilon \, |y| \, |v^H \Delta B'(y) v|} + \mathcal{O}(\epsilon) =$$

$$= \frac{\alpha \, \| \left[A_M \; A_{M-1} \; \dots \; A_0\right] \|_F \|v\|_2^2}{|y| \, |v^H \Delta B'(y) v|} + \mathcal{O}(\epsilon).$$

□

Combining Proposition 4.0.4 and Proposition 4.0.6 and using the relation (4.4), we obtain an upper bound for the forward error associated with an approximate eigenvalue $\widetilde{y}$ of the matrix polynomial $B(y)$, given by the product between the backward error and the condition number. At this point, we can proceed in two different ways:

- choosing the normwise backward error $\eta(\widetilde{y}, \widetilde{v})$, associated with an approximate eigenpair $(\widetilde{y}, \widetilde{v})$;
- choosing the normwise backward error $\eta(\widetilde{y})$, associated with an approximate eigenvalue $\widetilde{y}$, without computing the corresponding eigenvector.

We decide to rely on the second approach. In fact, recall that the application of the QZ algorithm on the pencil (3.10) encloses the majority of the computational cost of the resultant-based method. Furthermore, when we employ the command `eig` in MATLAB for a pencil, we prefer to compute only the eigenvalues. The computation of the associated eigenvectors leads to an additional cost, due to the computation of the matrices needed for each transformation step [35]. In order to avoid the increase of the computational cost, we prefer to follow the second approach, which does not need the computation of all the eigenvectors.

Therefore, the forward error $r_{\widetilde{y}}$ associated with an approximate eigenvalue $\widetilde{y}$ can be bound by the quantity

$$\begin{aligned} r_{\widetilde{y}} &\leq \eta(\widetilde{y}) \cdot \kappa(y, B) \\ &\leq \frac{1}{\alpha \| \left[A_M \; A_{M-1} \; \dots \; A_0\right] \|_F} \sigma_{\min}\left(B(\widetilde{y})\right) \cdot \frac{\alpha \| \left[A_M \; A_{M-1} \; \dots \; A_0\right] \|_F \|v\|_2^2}{|y| \, |v^H B'(y) v|} \\ &\leq \sigma_{\min}\left(B(\widetilde{y})\right) \frac{\|v\|_2^2}{|y| \, |v^H B'(y) v|}. \end{aligned} \tag{4.29}$$

Note that the vector $v$, which is an approximation of the eigenvector corresponding to $\widetilde{y}$, occurs in the bound for the forward error introduced in (4.29). In order to compute the eigenvector $v$, we evaluate the matrix polynomial $B(y)$ at $\widetilde{y}$, obtaining a symmetrical matrix $B(\widetilde{y})$ and then we perform the SVD decomposition, that is

$$B(\widetilde{y}) = U\Sigma V^H, \tag{4.30}$$

where $U, V \in \mathbb{C}^{N\times N}$ are unitary matrices and $\Sigma \in \mathbb{R}^{N\times N}$ has diagonal entries that are positive and in descending order, that is $\sigma_1 \geq \sigma_2 \ldots \geq \sigma_N \geq 0$. Note that $B(\widetilde{y})$ is symmetric, therefore the SVD decomposition coincides to its eigendecomposition, taking the absolute values of the singular values $\sigma_i$ for $i = 0, \ldots, N$. We need the smallest singular value, which we denote as $\sigma_N(B(\widetilde{y}))$, and then we select the corresponding singular vector, that is, the last column of the unitary matrix $V$, and we use it as an approximation of the eigenvector $v$ in (4.29).

Since we are interested in real eigenvalues, we can restrict the analysis of the forward error to the real parts of the approximate eigenvalues. In Algorithm 4, we summarize the method for computing the radius $r_{\widetilde{y}}$ of the disk of inclusion associated with the approximate eigenvalue $\widetilde{y}$.

**Algorithm 4** Disk of inclusion

**construct** pencil $\mathcal{L}_0 - y\mathcal{L}_1$
**compute** eigenvalue $\widetilde{y}$
**compute** $B(\widetilde{y})$
**compute** $[U, S, V] = \text{svd}(B(\widetilde{y}))$
**select** $\sigma_{\min}(B(\widetilde{y}))$ and corresponding $\widetilde{v}$
**compute** $\left|\widetilde{v}^H B'(\widetilde{y})\widetilde{v}\right|$
**compute** bound $r_{\widetilde{y}}$
**if** $D(\widetilde{y}, r_{\widetilde{y}}) \cap [-1, 1] \neq \emptyset$ **then**
    **accept** $\widetilde{y}$
**else**
    **reject** $\widetilde{y}$
**end if**

As mentioned before, the computation of the inclusion radius $r_{\widetilde{y}}$ for each eigenvalue $\widetilde{y}$ could lead to an increase of the computational cost. However, if we employ the upper bound in the form (4.29), we avoid the computation of all the eigenvectors of the pencil (3.10). Let us give an idea of the complexity of this step. Consider two bivariate polynomials $p(x, y)$ and $q(x, y)$ of the same degree $n$, both in the variable $x$ and in the variable $y$. The matrix polynomial $B(y)$ representing the Bézout matrix has size $n \times n$ and degree $2n$. Therefore the colleague pencil $\mathcal{L}_0 - y\mathcal{L}_1$ has size $2n^2 \times 2n^2$ and the computational cost employed in order to obtain all its generalized eigenvalues is $\mathcal{O}(n^6)$, since we rely on the QZ algorithm.

At this point, for each generalized eigenvalue of the colleague pencil, we perform a SVD of the matrix polynomial $B(y)$. Since one step needs $\mathcal{O}(n^3)$ operations and we repeat the same process for all the $n^2$ eigenvalues, the whole procedure has a computational cost $\mathcal{O}(n^5)$. Note that adding this procedure

to the resultant-based method does not increase the complexity and the QZ algorithm represents the majority of the computational cost.

Further studies can be done in order to improve the estimate for the backward error and the condition number associated with the eigenvalue problem. For instance, we can take into account that we deal with the Bézout resultant and therefore the matrix polynomial of perturbation should have the same structure. Another possibility consists in considering the special form of the eigenvectors of the colleague pencil, as mentioned in the proof of Proposition 3.1.1.

## 4.1 Condition analysis

A different way to study the conditioning of the problem uses the fact that the eigenvalue problem derives from bivariate rootfinding. This approach, developed in [36, 38], introduces an absolute condition number associated with the eigenvalues of the matrix polynomial and analyzes the connection with the conditioning of the original rootfinding problem. Let us outline the main steps of this analysis. Consider two bivariate polynomials $p(x, y)$ and $q(x, y)$ and let $(x_*, y_*)$ be a simple common root. Thoughout this section, we assume that $\|p\|_\infty = \|q\|_\infty = 1$. Then, consider $(\widetilde{x}, \widetilde{y}) = (x_* + \Delta x, y_* + \Delta y)$ the common root of the perturbed polynomials $\widetilde{p} = p + \Delta p$ and $\widetilde{q} = q + \Delta q$. We associate a notion of *absolute condition number* $\kappa_r$ to a simple root $(x_*, y_*)$ [27]

**Definition 4.1.1.** The *absolute condition number* associated with the common root $(x_*, y_*)$ is defined as

$$\kappa_r := \lim_{\epsilon \to 0^+} \sup \left\{ \frac{1}{\epsilon} \min \left\| \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \right\|_2 \; : \; \widetilde{p}(\widetilde{x}, \widetilde{y}) = \widetilde{q}(\widetilde{x}, \widetilde{y}) = 0 \right\}, \tag{4.31}$$

where the supremum is taken over the set $\left\{ (\widetilde{p}, \widetilde{q}) \; : \; \left\| \begin{bmatrix} \|\Delta p\|_\infty \\ \|\Delta q\|_\infty \end{bmatrix} \right\|_2 \leq \epsilon \right\}$.

The absolute condition number $\kappa_r$ associated with the root $(x_*, y_*)$ can be expressed in a more convienient form. In fact, expanding to the first order, we have

$$0 = \begin{bmatrix} \widetilde{p}(\widetilde{x}, \widetilde{y}) \\ \widetilde{q}(\widetilde{x}, \widetilde{y}) \end{bmatrix} = \begin{bmatrix} \partial_x p(x_*, y_*) & \partial_y p(x_*, y_*) \\ \partial_x q(x_*, y_*) & \partial_y q(x_*, y_*) \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} + \begin{bmatrix} \Delta p(\widetilde{x}, \widetilde{y}) \\ \Delta q(\widetilde{x}, \widetilde{y}) \end{bmatrix}. \tag{4.32}$$

Then, the condition number $\kappa_r$ of the rootfinding problem is equal to $\|J^{-1}\|_2$, where $J$ is the Jacobian matrix (2.2). The introduction of this condition number allows us to relate the original rootfinding problem and the eigenvalue problem associated with the matrix polynomial. To this end, we introduce a notion of condition number for eigenvalue of the matrix polynomial $B(y)$, which is slightly different from (4.19), and we follow the analysis proposed in [36]:

$$\kappa_e(y_*, B) := \lim_{\epsilon \to 0^+} \sup \left\{ \frac{1}{\epsilon} \min |\Delta y| : \left(B(y + \Delta y) + \Delta B(y + \Delta y)\right)(v + \Delta v) = 0 \right\} \tag{4.33}$$

where the supremum is taken over the set of matrix polynomials such that $\max_{y\in[-1,1]} \|\Delta B(y)\|_2 \leq \epsilon$.

Employing the structure of the Bézout matrix and the generalized Vandermonde form of its eigenvectors, the absolute condition number $\kappa_e$ associated with the eigenvalue $y_*$ assumes the form

$$\kappa_e(y_*, B) = \frac{\|v\|_2^2}{|\det J|}, \tag{4.34}$$

where $J$ is the Jacobian matrix (2.2) and $v = [T_0(x_*), \ldots, T_{N-1}(x_*)]^T$ (see Theorem 1 in [36] for a complete proof). At this point, we compare the condition number $\kappa_r(x_*, y_*)$ associated with the rootfinding problem and the condition number $\kappa_e(y_*, B)$ for the eigenvalue problem [38].

**Theorem 4.1.2.** *The condition number $\kappa_e(y_*, B)$ associated with the eigenvalue problem and the condition number $\kappa_r(x_*, y_*)$ associated with the rootfinding problem satisfy the relations*

$$\frac{1}{2}\frac{\kappa_r^2}{\|J\|_2\|J^{-1}\|_2} \leq \kappa_e(y_*, B) \leq 2N\frac{\kappa_r^2}{\|J\|_2\|J^{-1}\|_2}. \tag{4.35}$$

*Moreover, there exist two polynomials $\widehat{p}(x, y)$ and $\widehat{q}(x, y)$ with a simple root $(x_*, y_*)$ such that*

$$\kappa_e(y_*, B) \geq \|J^{-1}\|_2^2 = \kappa_r(x_*, y_*)^2, \tag{4.36}$$

*where $\|J^{-1}\|_2 > 1$.*

This result points out that there exist situations in which the resultant-based method worsens the conditioning. In these cases, the resultant-based method could lose accuracy and produce inaccurate solutions or miss some of them. However, the potentially ill-conditioned rootfinding problems can be detected in advance computing the norm of the Jacobian matrix.

# Chapter 5

# Numerical results

In this chapter, we present several numerical experiments in order to test the behaviour of the resultant-based method discussed so far. We implement this method in MATLAB R2017a and run the numerical experiments using an Intel(R) Core(TM) i5-4200 CPU running at 2.30GHz and 4GB of RAM. We propose an implementation of the resultant method for bivariate polynomials, in combination with the hidden variable technique. In order to construct the Bézout resultant matrix, we rely on the approach based on interpolation, as proposed in Section 3.2.2. Moreover, we employ the multidimensional discrete Fourier transform, in order to perform the three-dimensional interpolation step. We incorporate the bound on the forward error, presented in Chapter 4. To this end, for each possible value of the variable $y$, we compute the corresponding radius of the inclusion disk. Then, we look for the intersection between this disk and the interval $[-1, 1]$, filtering out the values of $y$ that are guaranteed to belong to $\mathbb{C} \setminus [-1, 1]$.

We represent bivariate polynomials by storing their coefficients in matrix form. Consider a polynomial in two variables $x$ and $y$, defined on the domain $[-1, 1] \times [-1, 1]$:

$$p(x, y) = \sum_{i=0}^{n_p} \sum_{j=0}^{m_p} p_{ij} T_i(x) T_j(y). \tag{5.1}$$

In order to represent the polynomial $p(x, y)$, we use the matrix `p`

$$\mathtt{p} = \begin{bmatrix} p_{00} & p_{01} & \cdots & p_{0(m_p-1)} & p_{0m_p} \\ p_{10} & p_{11} & \cdots & p_{1(m_p-1)} & p_{1m_p} \\ \vdots & \vdots & & & \vdots \\ p_{n_p0} & p_{n_p1} & \cdots & p_{m_p(m_p-1)} & p_{n_pm_p} \end{bmatrix}, \tag{5.2}$$

containing all the coefficients of $p(x, y)$. In the numerical experiments, we consider two matrices `p` and `q`, representing two different polynomials, and we search for their common roots in the domain $[-1, 1]^2$. We run the experiments using our implementation of the resultant-based method involving the Bézout resultant and we compare the results with the ones obtained using the command `roots` in Chebfun2.

## 5.1 Random bivariate polynomials

The first set of experiments are carried out using random bivariate polynomials. To this end, we consider a set of matrices, generated using the command `rand` in MATLAB, which produces uniformly distributed numbers in $[0, 1]$, and we employ them in order to represent the bivariate polynomials. The test matrices are square matrices of increasing size $n$. For simplicity, we run the methods on matrices of the same size $n$, both for the polynomial $p(x, y)$ and the polynomial $q(x, y)$. We run the resultant-based method, using the construction of the Bézout resultant matrix proposed in Section 3.2.2, and we compare the results with the function implemented `roots` in Chebfun2. For each choice of the parameter $n$, we run the methods on 100 different matrices of the same size, and we report in Table 5.1 the average timings.

| $n$ | ConstBézout(s) | ForwardErr(s) | TotInt(s) | TotRoots(s) |
|---|---|---|---|---|
| 5 | $1.6435 \cdot 10^{-3}$ | $8.2362 \cdot 10^{-3}$ | $3.9354 \cdot 10^{-2}$ | $3.5764 \cdot 10^{-1}$ |
| 7 | $1.8274 \cdot 10^{-3}$ | $1.6816 \cdot 10^{-2}$ | $6.5617 \cdot 10^{-2}$ | $6.6900 \cdot 10^{-1}$ |
| 10 | $2.2394 \cdot 10^{-3}$ | $4.3251 \cdot 10^{-2}$ | $1.6773 \cdot 10^{-1}$ | 1.3617 |
| 15 | $4.0355 \cdot 10^{-3}$ | $1.8354 \cdot 10^{-1}$ | $9.4083 \cdot 10^{-1}$ | 3.4990 |
| 17 | $4.7535 \cdot 10^{-3}$ | $3.4414 \cdot 10^{-1}$ | 1.9124 | 12.6786 |
| 20 | $7.1480 \cdot 10^{-3}$ | $5.9093 \cdot 10^{-1}$ | 5.9977 | 20.2876 |
| 25 | $1.6803 \cdot 10^{-2}$ | 1.4047 | 27.1178 | 35.4819 |
| 30 | $1.7870 \cdot 10^{-2}$ | 2.9365 | 96.7149 | 57.9397 |

Table 5.1: In the first column, we report the size $n$ of the matrices `p` and `q`. In the second column, we report the time (seconds) employed by the interpolation step. In the third column, we report the time (seconds) employed by the computation of the disks of inclusion. In the fourth and the fifth column, we report the total time (seconds) employed by our implementation and the command `roots` in Chebfun2.

In Table 5.1, we compare the time employed by the two methods. In the second column of the table, we report the time, expressed in seconds, spent on the interpolation step in order to compute the coefficients of the matrix polynomial representing the Bézout resultant matrix. In the third column, we report the time employed in order to compute the disks of inclusion, as proposed in Algorithm 4. The fourth and the fifth column, instead, collect the total time spent by the resultant-based method combined with the interpolation and by the command `roots` in Chebfun2, respectively. Note that the time employed in order to complete the three-dimensional interpolation step is negligible, compared to the total time required by the algorithm. For both the methods, the $QZ$ algorithm accounts for the majority of the computational cost. In fact, if the matrices representing the polynomials have size $n \times n$, the coefficients of the colleague pencil (3.10) have size $n^2 \times n^2$, leading to a cost of the $QZ$ algorithm of $\mathcal{O}(n^6)$. From the results of the numerical experiments, we note that the complexity of the technique proposed in Algorithm 4 does not exceed the complexity of the $QZ$ algorithm.

For the test matrices of size $n = 30$, we note that the time employed by the command `roots` in Chebfun2 is lower than the time employed by our method. This is because `roots` implements several additional techniques, developed in

order to mitigate the computational cost of the $QZ$ step. One of them is the subdivision of the domain $[-1,1]^2$, proposed in [36]. We remark that such a subdivision scheme might be applied to our approach as well, thus preventing an excessive growth of the degrees of the polynomials.

In the second column of Table 5.2, we report the number of successes over 100 experiments. Our implementation does not find any spurious solutions. We note that even when the method does not succeed in computing all the solutions, it still computes the large majority of the roots. In particular, in the situation in which the method does not compute all the solutions, our implementation reaches 478 of the 479 common roots, in the case $n = 30$. A possible way to deal with this issue could be rely on suitable estimates for the forward error associated with each approximate common root $(\widetilde{x}, \widetilde{y})$.

We now compute the residual of the polynomial evaluations at the approximate intersections $(\widetilde{x}, \widetilde{y})$, to assess the accuracy of the proposed scheme.

**Definition 5.1.1.** Consider a pair $(\widetilde{x}, \widetilde{y}) \in [-1,1] \times [-1,1]$ and let $p(x,y) = \sum_{i,j=0}^{n-1} p_{ij} T_i(x) T_j(y)$ be a bivariate polynomial defined in $[-1,1]^2$. The *relative residual* of $p(x,y)$ in $(\widetilde{x}, \widetilde{y})$ is defined as

$$r_p(\widetilde{x}, \widetilde{y}) = \frac{p(\widetilde{x}, \widetilde{y})}{\sum_{i,j=0}^{n-1} |p_{ij}| \, |T_i(\widetilde{x})| \, |T_j(\widetilde{y})|}. \tag{5.3}$$

| size | Successes | MaxResInt | MaxResRoots |
|---|---|---|---|
| 5 | 100/100 | $8.2857 \cdot 10^{-14}$ | $2.6542 \cdot 10^{-15}$ |
| 7 | 100/100 | $2.3889 \cdot 10^{-13}$ | $7.4004 \cdot 10^{-15}$ |
| 10 | 100/100 | $3.1597 \cdot 10^{-12}$ | $2.9072 \cdot 10^{-14}$ |
| 15 | 100/100 | $7.5637 \cdot 10^{-12}$ | $1.2363 \cdot 10^{-13}$ |
| 17 | 100/100 | $3.3017 \cdot 10^{-11}$ | $1.6733 \cdot 10^{-13}$ |
| 20 | 100/100 | $7.6861 \cdot 10^{-11}$ | $2.0163 \cdot 10^{-13}$ |
| 25 | 100/100 | $1.2955 \cdot 10^{-10}$ | $2.8029 \cdot 10^{-13}$ |
| 30 | 99/100 | $3.6488 \cdot 10^{-10}$ | $3.4927 \cdot 10^{-13}$ |

Table 5.2: In the first column, we report the size $n$ of the matrices `p` and `q`. In the second column, we report the number of experiments in which our implementation succeeds in computing all the roots. In the third and in the fourth column, we report the maximum value of the relative residual, computed by our implementation and by the command `roots`.

In order to evaluate the accuracy of the approximate zeros, for each pair $(\widetilde{x}, \widetilde{y})$ we compute the 2-norm of the vector

$$\mathbf{r}_{(\widetilde{x},\widetilde{y})} := [\, r_p(\widetilde{x}, \widetilde{y}) \quad r_q(\widetilde{x}, \widetilde{y}) \,]. \tag{5.4}$$

In the columns MaxResInt and MaxResRoots of Table 5.2, we report the maximum value $\mathbf{r}_{(\widetilde{x},\widetilde{y})}$, computed over the set of approximate pairs $(\widetilde{x}, \widetilde{y})$ obtained via our implementation of the method and via `roots` in Chebfun2, respectively. Note that our implementation computes the common roots of $p(x,y)$ and $q(x,y)$ with a potential loss of accuracy. This issue could be mitigated, for instance, by applying a step of Newton's method, or relying on different techniques, such as local Bézout refinement [36]. Similar strategies are employed as

post-processing by Chebfun2. However, the study of this approaches is beyond the scope of this thesis.

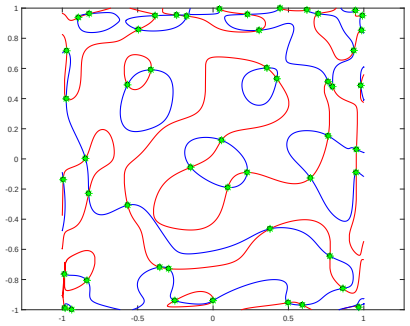

Figure 5.1: Common roots of two bivariate polynomials of degrees 10

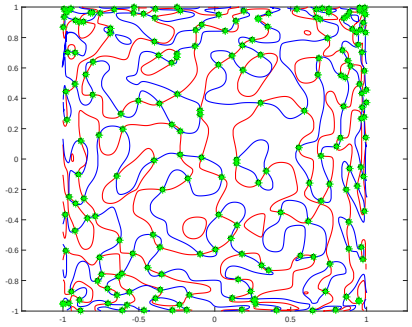

Figure 5.2: Common roots of two bivariate polynomials of degrees 20

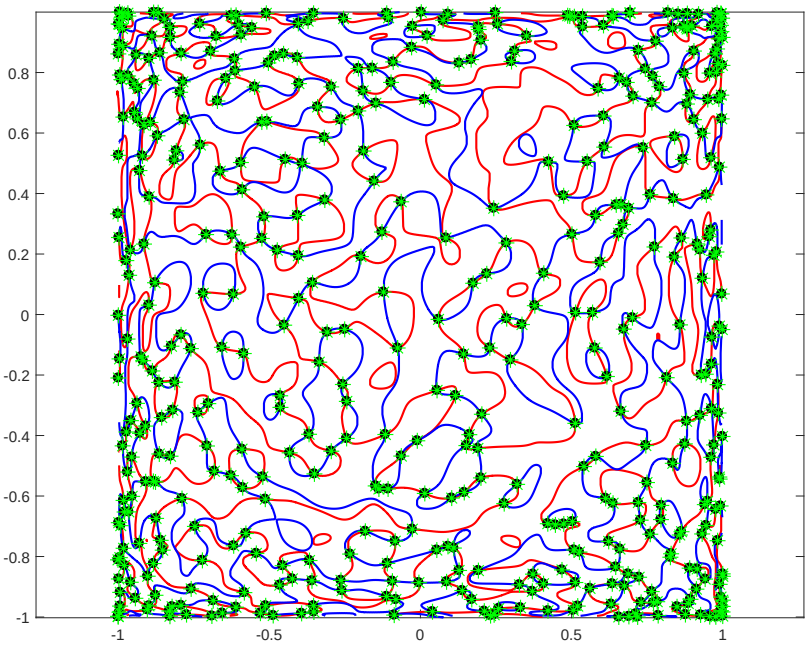


Figure 5.3: Common roots of two bivariate polynomials of degrees 30

In Figures 5.1, 5.2 and 5.3 we plot several examples of our numerical experiments. Consider two bivariate polynomials of the same degrees both in the variable $x$ and in the variable $y$, whose matrices of coefficients `p` and `q` are generated using the command `rand` in MATLAB. We plot the results obtained for three different choices of the degrees of the polynomials, that is, $n = 10$, $n = 20$ and $n = 30$. We plot the level sets $\left\{(x, y) \in [-1, 1]^2 : \ p(x, y) = 0\right\}$ (in blue) and $\left\{(x, y) \in [-1, 1]^2 : \ q(x, y) = 0\right\}$ (in red). We indicate the solution set found using our implementation of the method (in green) and the zeros computed by

`roots` in Chebfun2 (in black). The two solution sets are indistinguishable in the picture.

In Figure 5.3, in particular, we note that the method computes all the solutions of the problem in the cluster in the upper-right corner of the domain $[-1,1]^2$.

## 5.2 Common zeros of two bivariate functions

We can treat the rootfinding problem associated with two smooth bivariate functions, using the resultant-based method. In this case, we replace the functions with their polynomial approximations, expressed in Chebyshev basis. Recall that for an analytic function $f(x)$, we can easily find a polynomial $p(x)$ expressed in Chebyshev basis, such that $|f(x) - p(x)| < \epsilon$, for every $\epsilon > 0$ [50]. In Figure 5.4, we plot two bivariate smooth functions, generated using the command `randnfun2` in Chebfun2. In order to represent these functions using the matrices of coefficients, we employ the command `chebcoeffs2` of Chebfun2. This process leads to the construction of two matrices, containing the coefficients of the expansion of the smooth functions in Chebyshev basis. We decide to keep the first 30 coefficients of the expansion both in the $x$ variable and in the $y$ variable.

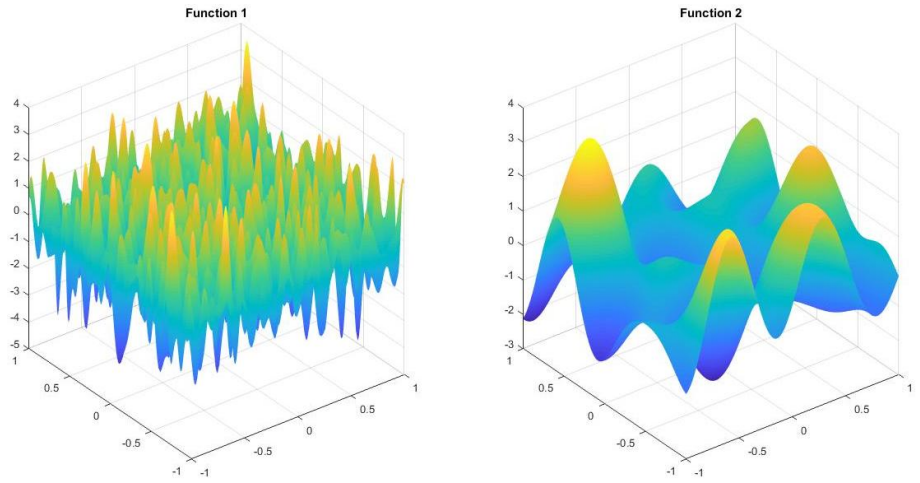


Figure 5.4: Smooth functions generated by `randnfun2`

In Figure 5.5, we plot in the zero level curves of the first smooth function (blue) and of the second smooth function (red). We indicate the common roots of the two functions computed by our implementation of the method (green) and by the command `roots` of Chebfun2 (black).

In this example, the resultant-based method locates all the 73 common roots in $[-1,1]^2$. The maximum distance between the solutions computed by our implementation and the zeros found by `roots` is equal to $1.2995 \cdot 10^{-11}$. In Table 5.3 we report the maximum and the minimum value reached by the residual associated with the approximate pair $(\widetilde{x}, \widetilde{y})$.

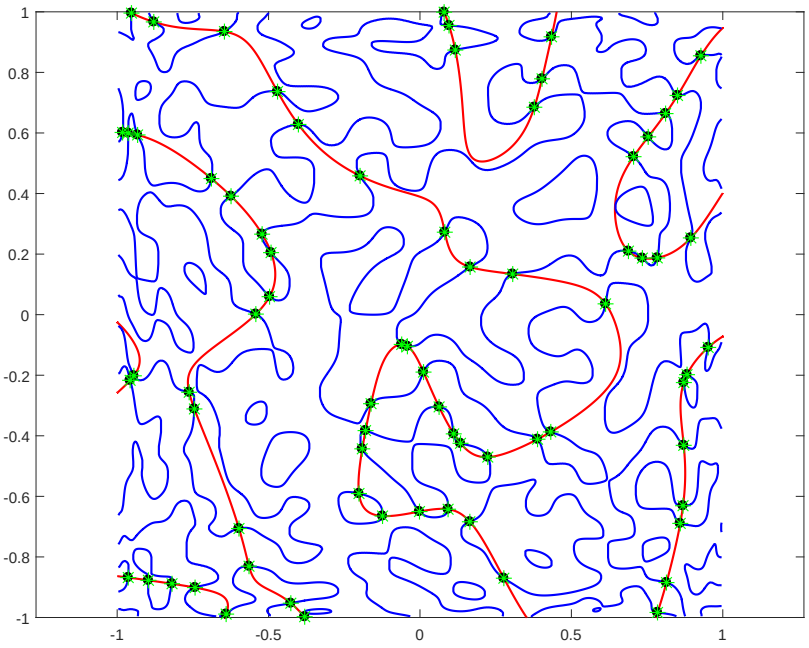


Figure 5.5: Common roots of two smooth functions

| maxInt | minInt | maxRoots | minRoots |
|---|---|---|---|
| $1.9936 \cdot 10^{-11}$ | $6.5902 \cdot 10^{-16}$ | $1.2200 \cdot 10^{-14}$ | $5.8944 \cdot 10^{-17}$ |

Table 5.3: In the first and in the second column, we report the maximum and the minimum relative residual computed for the pairs $(\widetilde{x}, \widetilde{y})$ computed by our implementation. In the third and in the fourth column, we report the maximum and the minimum value of the residual for zeros computed by `roots`.

Finally, we consider a different example, that is, solving:

$$\begin{cases} \cos(2(x^2+y^2)) = 0 \\ \cos(5(x+y)) = 0 \end{cases}, \qquad (x,y) \in [-1,1]^2 \tag{5.5}$$

Thanks to the special form of this rootfinding problem, we can compute analytically the common zeros of $f(x,y) = \cos(2(x^2+y^2))$ and $g(x,y) = \cos(5(x+y))$ and we can compare them with the roots located using our implementation of the method. In order to represent the bivariate functions, we employ the command `chebcoeffs2` in Chebfun2. The matrices of the coefficients are of size $29 \times 29$ and $26 \times 26$, for the functions $f(x,y)$ and $g(x,y)$, respectively. In Table 5.4, we report the maximum and the minimum distance between the roots located by the method and the ones computed analytically and the maximum and the minimum of the residuals. In Figure 5.6, we plot the zero level curves of the functions $f(x,y)$ and $g(x,y)$ (in blue and red respectively) and the approximate roots (green).

| numInt | maxDist | minDist | maxRes | minRes |
|---|---|---|---|---|
| 8 | $4.3916 \cdot 10^{-12}$ | $5.4174 \cdot 10^{-15}$ | $1.5002 \cdot 10^{-10}$ | $1.3307 \cdot 10^{-14}$ |

Table 5.4: In the first column, we report the number of computed roots. In the second and third column, the maximum and the minimum value of the the distance between the approximate roots and the solutions computed analytically. In the fourth and in the fifth column, the maximum and the minimum value of the relative residual.

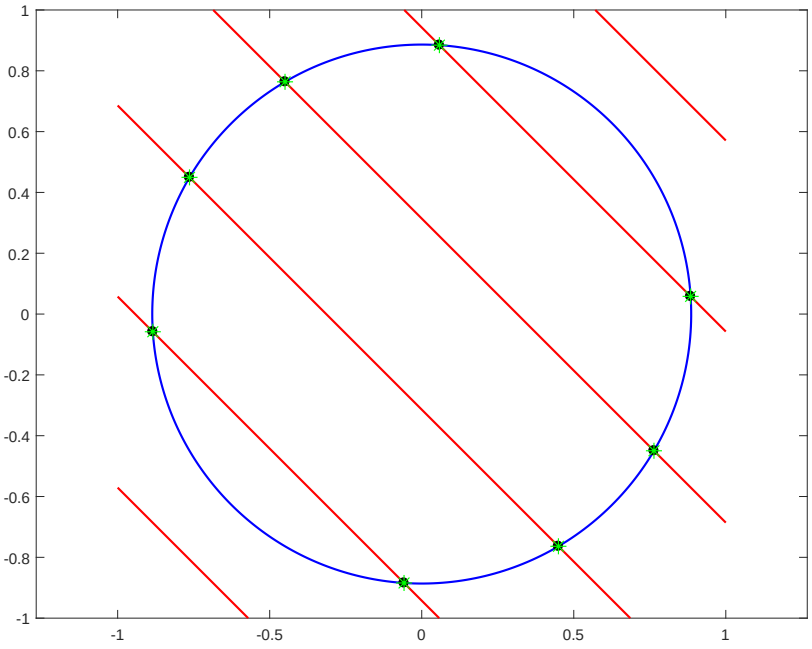


Figure 5.6: Common roots of $f(x, y)$ and $g(x, y)$

# Conclusions

In this thesis, we analyze the Bézout resultant-based method for computing the common roots of two bivariate polynomials expressed in Chebyshev basis. Our main reference is the work of Nakatsukasa et al. [36]. In order to treat the step of the construction of the Bézout matrix, we study two possible approaches. The first one is based on the theory of the vector spaces of linearizations $\mathbb{DL}$ for matrix polynomials, presented in [37], and the second one involves a three-dimensional interpolation. We choose to perform the interpolation step using the multidimensional discrete Fourier transform. This construction has the added benefit of being easier to generalize to the multidimensional case.

We develop estimates for the backward error and the condition number of the approximate eigenvalues of the matrix polynomial representing the Bézout resultant matrix, using the work by Tisseur [47] as a starting point. Then, we propose an upper bound for the forward error of each eigenvalue of the matrix polynomial associated with the Bézout resultant matrix, defining disks of inclusion for the approximate eigenvalues.

This thesis can be considered as a starting point in order to develop an analogous resultant-based method for the three-dimensional rootfinding problem. In particular, we can replace the Bézout matrix with the Cayley resultant matrix, defined in Section 2.3. In this case, the results based on the double ansatz space $\mathbb{DL}$, described in Section 3.2.1, are difficult to generalize. For this reason, we develop an approach based on the three-dimensional interpolation, as described in Section 3.2.2. In fact, this method can be generalized and applied to the Cayley matrix using a four-dimensional tensor-based formulation.

Further work could also focus on the development of estimates of the backward error and condition number for each individual eigenvalue of the matrix polynomial. For instance, we can perform an analysis that takes into account in consideration the structure of the resultant matrix and the Vandermonde form of the eigenvectors.

In closing, we mention several techniques that can be studied in order to improve the accuracy of the computed solutions. For instance, approaches as the subdivision of the domain or local Bézout refinement are employed in the command `roots` in Chebfun2, as described in [36]. We propose to generalize these techniques to the three-dimensional case and to incorporate them in our algorithm.

# Bibliography


[1] A. Amiraslani, R. M. Corless, and P. Lancaster. Linearization of matrix polynomials expressed in polynomial bases. *IMA J. Numer. Anal.*, 29(1):141–157, 2009.

[2] Alan L. Andrew, K.-W. Eric Chu, and Peter Lancaster. Derivatives of eigenvalues and eigenvectors of matrix functions. *SIAM J. Matrix Anal. Appl.*, 14(4):903–926, 1993.

[3] Zachary Battles and Lloyd N. Trefethen. An extension of MATLAB to continuous functions and operators. *SIAM J. Sci. Comput.*, 25(5):1743–1770, 2004.

[4] R. Bevilacqua, D. Bini, M. Capovani, and O. Menchi. *Metodi numerici*. Collana di matematica. Testi e manuali. Zanichelli, 1992.

[5] Dario A. Bini and Ana Marco. Computing curve intersection by means of simultaneous iterations. *Numer. Algorithms*, 43(2):151–175 (2007), 2006.

[6] John P. Boyd. Computing zeros on a real interval through Chebyshev expansion and polynomial rootfinding. *SIAM J. Numer. Anal.*, 40(5):1666–1682, 2002.

[7] Bruno Buchberger. Introduction to Gröbner bases. In *Gröbner bases and applications (Linz, 1998)*, volume 251 of *London Math. Soc. Lecture Note Ser.*, pages 3–31. Cambridge Univ. Press, Cambridge, 1998.

[8] Eduardo Cattani and Alicia Dickenstein. Introduction to residues and resultants. In *Solving polynomial equations*, volume 14 of *Algorithms Comput. Math.*, pages 1–61. Springer, Berlin, 2005.

[9] Eng-Wee Chionh, Ming Zhang, and Ronald N. Goldman. Fast computation of the Bezout and Dixon resultant matrices. *Journal of Symbolic Computation*, 33(1):13 – 29, 2002.

[10] C. W. Clenshaw. A note on the summation of Chebyshev series. *Math. Tables Aids Comput.*, 9:118–120, 1955.

[11] James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Math. Comp.*, 19:297–301, 1965.

[12] David A. Cox, John Little, and Donal O'Shea. *Using algebraic geometry*, volume 185 of *Graduate Texts in Mathematics*. Springer, New York, second edition, 2005.

[13] David A. Cox, John Little, and Donal O'Shea. *Ideals, varieties, and algorithms*. Undergraduate Texts in Mathematics. Springer, Cham, fourth edition, 2015. An introduction to computational algebraic geometry and commutative algebra.

[14] A. Edelman and H. Murakami. Polynomial roots from companion matrix eigenvalues. *Mathematics of Computation*, 64:763–776, 1995.

[15] Cedric Effenberger and Daniel Kressner. Chebyshev interpolation for nonlinear eigenvalue problems. *BIT*, 52(4):933–951, 2012.

[16] Yuli Eidelman, Luca Gemignani, and Israel Gohberg. Efficient eigenvalue computation for quasiseparable hermitian matrices under low rank perturbations. *Numerical Algorithms*, 47:253–273, 03 2008.

[17] David Elliott. Error analysis of an algorithm for summing certain finite series. *J. Austral. Math. Soc.*, 8:213–221, 1968.

[18] James F. Epperson. On the Runge example. *Amer. Math. Monthly*, 94(4):329–341, 1987.

[19] Valérie Frayssé and Vincent Toumazou. A note on the normwise perturbation theory for the regular generalized eigenproblem. *Numer. Linear Algebra Appl.*, 5(1):1–10, 1998.

[20] I. M. Gelfand, M. M. Kapranov, and A. V. Zelevinsky. *Discriminants, resultants and multidimensional determinants*. Modern Birkhäuser Classics. Birkhäuser Boston, Inc., Boston, MA, 2008. Reprint of the 1994 edition.

[21] Luca Gemignani and Leonardo Robol. Fast Hessenberg reduction of some rank structured matrices. *SIAM Journal on Matrix Analysis and Applications*, 38, 12 2016.

[22] W. Morvin Gentleman. Implementing Clenshaw-Curtis quadrature. II. Computing the cosine transformation. *Comm. ACM*, 15:343–346, 1972.

[23] Amparo Gil, Javier Segura, and Nico M. Temme. *Numerical Methods for Special Functions*. Society for Industrial and Applied Mathematics, USA, 1st edition, 2007.

[24] I. Gohberg, P. Lancaster, and L. Rodman. *Matrix polynomials*, volume 58 of *Classics in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2009.

[25] I. J. Good. The colleague matrix, a Chebyshev analogue of the companion matrix. *Quart. J. Math. Oxford Ser. (2)*, 12:61–68, 1961.

[26] Georg Heinig and Karla Rost. Introduction to Bezoutians. In *Numerical methods for structured matrices and applications*, volume 199 of *Oper. Theory Adv. Appl.*, pages 25–118. Birkhäuser Verlag, Basel, 2010.

[27] Nicholas J. Higham. *Accuracy and stability of numerical algorithms*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, second edition, 2002.

[28] Nicholas J. Higham, D. Steven Mackey, Niloufer Mackey, and Françoise Tisseur. Symmetric linearizations for matrix polynomials. *SIAM J. Matrix Anal. Appl.*, 29(1):143–159, 2006/07.

[29] Bayram Ali Ibrahimoglu. Lebesgue functions and Lebesgue constants in polynomial interpolation. *J. Inequal. Appl.*, pages Paper No. 93, 15, 2016.

[30] Guđbjörn F. Jónsson and Stephen A. Vavasis. Accurate solution of polynomial equations using Macaulay resultant matrices. *Math. Comp.*, 74(249):221–262, 2005.

[31] Frances Kirwan. *Complex algebraic curves*, volume 23 of *London Mathematical Society Student Texts*. Cambridge University Press, Cambridge, 1992.

[32] Naftali Kravitsky. On the discriminant function of two commuting nonselfadjoint operators. *Integral Equations Operator Theory*, 3(1):97–124, 1980.

[33] D. Steven Mackey, Niloufer Mackey, Christian Mehl, and Volker Mehrmann. Vector spaces of linearizations for matrix polynomials. *SIAM J. Matrix Anal. Appl.*, 28(4):971–1004, 2006.

[34] Dinesh Manocha and James Demmel. Algorithms for intersecting parametric and algebraic curves i: Simple intersections. *ACM Trans. Graph.*, 13(1):73–100, January 1994.

[35] C. B. Moler and G. W. Stewart. An algorithm for generalized matrix eigenvalue problems. *SIAM J. Numer. Anal.*, 10:241–256, 1973.

[36] Yuji Nakatsukasa, Vanni Noferini, and Alex Townsend. Computing the common zeros of two bivariate functions via Bézout resultants. *Numer. Math.*, 129(1):181–209, 2015.

[37] Yuji Nakatsukasa, Vanni Noferini, and Alex Townsend. Vector spaces of linearizations for matrix polynomials: a bivariate polynomial approach. *SIAM J. Matrix Anal. Appl.*, 38(1):1–29, 2017.

[38] Vanni Noferini and Alex Townsend. Numerical instability of resultant methods for multidimensional rootfinding. *SIAM J. Numer. Anal.*, 54(2):719–743, 2016.

[39] Stefan Ragnarsson and Charles F. Van Loan. Block tensor unfoldings. *SIAM J. Matrix Anal. Appl.*, 33(1):149–169, 2012.

[40] Theodore J. Rivlin. *An introduction to the approximation of functions*. Blaisdell Publishing Co. Ginn and Co., Waltham, Mass.-Toronto, Ont.-London, 1969.

[41] Simon J. Smith. Lebesgue constants in polynomial interpolation. *Ann. Math. Inform.*, 33:109–123, 2006.

[42] Alicja Smoktunowicz. Backward stability of Clenshaw's algorithm. *BIT*, 42(3):600–610, 2002.

[43] Andrew J. Sommese and Charles W. Wampler, II. *The numerical solution of systems of polynomials*. World Scientific Publishing Co. Pte. Ltd., Hackensack, NJ, 2005. Arising in engineering and science.

[44] Laurent Sorber, Marc Van Barel, and Lieven De Lathauwer. Numerical solution of bivariate and polyanalytic polynomial systems. *SIAM J. Numer. Anal.*, 52(4):1551–1572, 2014.

[45] Gábor Szegő. *Orthogonal polynomials*. American Mathematical Society, Providence, R.I., fourth edition, 1975. American Mathematical Society, Colloquium Publications, Vol. XXIII.

[46] Simon Telen, M. Barel, and J. Verschelde. A robust numerical path tracking algorithm for polynomial homotopy continuation. *ArXiv*, abs/1909.04984, 2019.

[47] Françoise Tisseur. Backward error and condition of polynomial eigenvalue problems. *Linear Algebra and its Applications*, 309(1):339 – 361, 2000.

[48] Lloyd N. Trefethen. *Spectral methods in MATLAB*, volume 10 of *Software, Environments, and Tools*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2000.

[49] Lloyd N. Trefethen. A Hundred-dollar, hundred-digit challenge. *SIAM-News*, 35(1), January 2002.

[50] Lloyd N. Trefethen. *Approximation theory and approximation practice*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2013.

[51] Paul Van Dooren and Patrick Dewilde. The eigenstructure of an arbitrary polynomial matrix: computational aspects. *Linear Algebra Appl.*, 50:545–579, 1983.

[52] T.L. Wayburn and J.D. Seader. Homotopy continuation methods for computer-aided process design. *Computers & Chemical Engineering*, 11(1):7 – 25, 1987.