SCUOLA
NORMALE
SUPERIORE

Classe di Scienze Matematiche, Fisiche e Naturali

PhD Thesis

Exploiting rank structures for the numerical treatment of matrix polynomials

Leonardo Robol

Advisor
Prof. Dario A. Bini
Dipartimento di Matematica,
Università di Pisa

2012 – 2015

CONTENTS

INTRODUCTION

The main purpose of this thesis is to analyze the relation between matrix polynomials and quasiseparable matrices. We show that these two apparently different topics have a very strict connection that becomes evident in the construction of linearizations for the solutions of polynomial eigenvalue problems. We say that $P(x)$ is a matrix polynomial if it is a polynomial with matrix coefficients or a matrix with polynomial entries. The two definitions are easily seen to be equivalent, and the set of $m_1 \times m_2$ matrix polynomials is denoted by $\mathbb{F}^{m_1 \times m_2}[x]$ where $\mathbb{F}$ is a field. Often, in this thesis, we consider the case where $m := m_1 = m_2$. In this case we say that the matrix polynomials are square. In the following we will often assume that $\mathbb{F} = \mathbb{C}$, but some of the results also hold in a more general setting.

A matrix is said quasiseparable if each of its off-diagonal submatrices has rank bounded by a small constant. The quasiseparability rank is defined as the maximum of the ranks of lower (resp. upper) off-diagonal submatrices. We often use QS rank as an abbreviation for these ranks when they coincide. A typical example of these kinds of structures are banded matrices (which originally inspired the definition of quasiseparable structures) and also diagonal plus low rank matrices. Quasiseparable matrices enjoy some nice properties, such as invariance of the QS rank under inversion and its subadditivity for addition and multiplication. These structures have been studied by many authors in the past, such as Boito, Eidelman and Gemignani [23], Chandrasekaran and Gu [28], Eidelman, Gohberg and Haimovici [49, 50], Eidelman, Gemignani and Gohberg [48], Mastronardi, Van Barel, Vandebril [89, 90] and many others.

The work by Gohberg, Lancaster and Rodman [61] is a standard reference in the field of matrix polynomials, together with the book of Gantmacher [58] where the name λ-matrix is also used in place of matrix polynomials. More recently an interesting overview of the properties of linearizations and, more generally, the different equivalences relations between matrix polynomials has been given by De Terán, Dopico and Mackey in [35].

Many scientific problems rely on the solution of matrix polynomials (see [8] for a well organized collection of such problems), either by requiring the computation of a solvent, i.e., a matrix X such that $\sum_{i=0}^{n} P_i X^i = 0$, or by requiring the computation of the polynomial eigenvalues and eigenvectors, i.e., by finding the solutions of $\det P(x) = 0$[1] and the relative vectors in the kernel. The latter problems can be seen as a generalization of standard eigenvalue problems, i.e., problems of the form $(A - xI)v = 0$, and scalar polynomials, i.e., problems of the form $p(x) = \sum_{i=0}^{n} p_i x^i = 0$, $x \in \mathbb{C}$. In fact, polynomial eigenvalue problems (usually called PEPs) reduce to the former when the degree is $n = 1$ and the polynomial is monic, and to the latter when the size of the matrices is $m = 1$.

The main idea of this thesis is to exploit quasiseparable structures that are found in some linearizations for scalar polynomials in order to obtain an effective approximation algorithm for the rootfinding problem, and then generalize most of the ideas used in this setting to the context of matrix polynomials and polynomial eigenvalue problems.

We start our work in Chapter 1 by analyzing the scalar rootfinding problem. We consider a scalar polynomial $p(x) = \sum_{i=0}^{n} p_i x^i \in \mathbb{C}[x]$ and we want to find its roots ξ_i, with $i = 1, \dots, n$,

[1] The general definition of eigenvalues of a matrix polynomial does not rely on the determinant in order to be able to handle infinite eigenvalues and non-square matrix polynomials. It is given with all the details in Section 2.1

such that $p(x) = p_n \prod_{i=1}^{n}(x - \xi_i)$. We briefly review some of the most widespread methods available in the literature for the approximations of the roots. Section 1.1 is devoted to the analysis of classical functional iterations usually obtained as generalizations of the Newton's method, such as the Durand–Kerner iteration of [45] and the Ehrlich–Aberth iteration of [1]. In Section 1.2 we analyze the approaches based on linearizations, such as the well-known Frobenius linearization (see [35] or [61]) and the linearizations for other polynomial bases. These constructions are typically used to solve polynomials by applying an eigenvalues solver (such as the celebrated QR method of [55] and [73]). This is the method on which the `roots` command of MATLAB relies.

Then, in Section 1.3 we introduce a class of linearizations that can be used as a building block for an approximation algorithm. We also show the relation that these linearizations have with secular equations, i.e., equations of the form $S(x) = \sum_{i=1}^{n} \frac{a_i}{x - b_i} - 1 = 0$. This kind of equations is interesting on its own since they arise in the computation of eigenvalues of rank 1 corrected matrices [62] and the application of Divide and Conquer techniques as proposed in [32] and [64]. The family of linearizations that we introduce can be used to solve secular equations directly or to solve polynomials by constructing intermediate secular equations with good numerical properties. We turn our attention to the latter problem, and provide a practical and easy way to compute these intermediate secular equations based on tropical algebra ([59] of Gaubert and Sharify is a good reference, and handles also the more involved case of matrix polynomials).

In Section 1.5 we show how it is possible to use functional iterations such as Ehrlich–Aberth's method in order to find the eigenvalues of matrices like the one of Section 1.3. We show how to exploit the connection with secular equations in order to obtain a fast algorithm and also to derive guaranteed bounds for the location of the roots. We combine all these elements together in order to provide a stable algorithm that can approximate the roots of a polynomial with a guaranteed arbitrary number of digits (at least in the case where the input coefficients are known exactly). In Section 1.7 some numerical experiments are reported that validate our approach by comparing the results with the older version of the `MPSolve` package [15] and the rootfinding package `eigensolve` [53].

The algorithm and the theoretical analysis of secular equations in the context of polynomial rootfinding have been published as an original contribution in [20].

In Section 1.6 we show that our framework is very easy to extend to particular classes of polynomials and we show how we have been able to compute the roots of the Mandelbrot polynomial of degree $2^{22} - 1$. The roots of this polynomial are the periodic point of the Mandelbrot map of order 22. Its coefficient in the monomial basis are very large integers and the rootfinding problem is badly conditioned. For this reason the computation of the roots starting from the coefficients is a very difficult problem. We show that it is possible to build a customized approach for the solution of these polynomials based on their recursive definition, and we show how it is possible to use the software package implementing the algorithm of Section 1.5, called `MPSolve`, in order to apply this strategy. Numerical experiments that validate the approach are reported at the end of Section 1.6. This algorithm is, as of now, the fastest method available for the approximation of roots of Mandelbrot polynomials.

Other approaches found in the literature are the one of Corless and Lawrence [31] and of Schleicher et al. [84]. The former approach allowed to approximate the roots of the polynomial of degree $2^{20} - 1$ on a large cluster in approximately the equivalent of 31 years of sequential computational time. In the latter instead, the authors propose to accelerate the computations by using an heuristic that is quite effective in practice but typically misses a non negligible portion of the roots. We have been able to approximate the roots of the polynomial of degree $2^{22} - 1$ in about one month of computational time on a single machine. The approach is a novel contribution developed in the context of the PhD thesis.

In Chapter 2 we deal with the solution of PEPs. This kind of problems is usually solved through the use of linearizations, typically using the Frobenius companion linearization [61]. In Section 2.1 we review the concept of linearizations and ℓ-ifications for matrix polynomials. Recently much interest has been devoted in finding new linearizations that preserve particular structures of the original matrix polynomial. This is the reason that has inspired the work in [74], where new vector spaces of linearizations for matrix polynomials are introduced with the aim of finding new structured linearizations for structured polynomials. Other contributions are [2] where linearizations in non-monomial basis are analyzed, and [33, 34], where another family of linearizations called Fiedler linearizations has been studied by De Terán, Dopico and Mackey. These linearizations have been originally introduced by Fiedler in [52] for scalar polynomials. They generalize the Frobenius companion matrices and are very easy to construct. Some interesting structured matrices are available in this class, such as a block pentadiagonal linearization. Exploiting these structures, though, remains an open problem.

In Section 2.2 we construct a family of linearizations and ℓ-ifications inspired by the one introduced in Section 1.3 for the scalar case. We show that when rough estimates for the moduli of the eigenvalues are known the linearizations obtained in this framework have very good conditioning properties. In Section 2.3 we discuss the use of tropical roots to determine the parameter defining the secular linearizations, that is, the estimates for the moduli of the eigenvalues, in a similar way of what is done in Section 1.4 for scalar polynomials. Section 2.4 reports numerical experiments that show the effectiveness of the approach in many interesting cases. In Section 2.5 we introduce a new extension of the class of secular linearizations that is built using a different strategy. We show that the two linearizations coincide in the simplest cases and that the former can be extended to higher degree (building an ℓ-ification with $\ell > 1$ instead of a linearization) while the latter has the same optimal properties regarding the conditioning of the scalar linearization of Section 1.3, at least when good estimates for the eigenvectors are available. The class of secular linearizations of Section 2.2 has been analyzed in [21], and is one of the original contributions to the literature developed in the context of the PhD. The latter class, instead, is part of a work that is currently in preparation and will be submitted soon [81].

In Chapter 3 the definitions of quasiseparable matrices, along with a brief review of the main results that we use, are introduced. We show that all the linearizations and ℓ-ifications that we have presented share this structure, which justify our interest in this topic. The secular linearizations and ℓ-ifications are special in this sense because they have the quasiseparability structure but they do not have the sparsity that is common to other linearizations such as the Frobenius or the ones for matrix polynomials expressed in orthogonal basis (even if in Section 2.2 a sparse version of the secular ℓ-ification is presented).

For this reason, we are interested in exploiting this structure in order to make many common operations faster on these classes of matrices. In order to obtain better results we restrict to a subclass of the quasiseparable matrices that is the one interesting for us: the class of diagonal plus low rank matrices. This class, in fact, contains all the matrix coefficients of the secular linearizations and ℓ-ifications. We show how to efficiently construct their Hessenberg form and the Hessenberg triangular form for pencils. We take particular care of having a low complexity bound with respect to the rank k. In the literature there are several methods to compute the Hessenberg reduction of quasiseparable matrices but they usually have complexity $O(n^2 k^\alpha)$ with $\alpha > 1$ where n is the size of the matrices and k is the quasiseparability rank (see for example [56] and [48]). While this is not important if k is negligible (the most studied case is when $k = 1$) in the case of linearizations $k < n$ but, in general, we do not have $k \ll n$. We design and analyze an algorithm that has $O(n^2 k)$ complexity and so, up to constants, is always competitive with the classical $O(n^3)$ algorithm for dense unstructured matrices. We show two different approaches for the reduction. The first is given in Section 3.3 and has been published in [22]. It is then extended to the Hessenberg triangular reduction in Section 3.4. The latter is based on different ideas and is presented in Section 3.5.

In Section 3.6 we discuss another application of quasiseparable structures to the solution of matrix polynomials. This problem arises in the solution of particular Markov chains called QBDs (Quasi-Birth and Death) where, in order to find the stationary vector characterizing the limit probability distribution, one needs to find a solvent of a quadratic matrix polynomial. One of the most efficient methods available for the computation of such a solvent is the cyclic reduction, which is carefully analyzed in relation to this application in [18]. We introduce a new kind of approximated quasiseparable structures that allows to efficiently represent matrices with a particular property of decaying singular values in off-diagonal submatrices. We show that this allows an acceleration of the cyclic reduction method used to solve quadratic matrix equations. We report some numerical experiments that validate this approach both from the point of view of speed and from the one of accuracy. A paper describing this approach in detail is currently in preparation and will be soon submitted [16].

At the end we draw some conclusions from the work presented in this thesis and we give some ideas for future development of these topics.

NOTATION AND SYMBOLS

Here we recap a list of notations and symbols used throughout the thesis. Every new symbol or notation that is introduced is explained in detail in the relevant chapter. When available, the reference to the page with the definition is added in the rightmost column of the following table.

Symbol	Meaning	Reference
$\mathbb{C}$	The field of complex numbers.	
$\mathbb{R}$	The field of real numbers.	
$\mathbb{F}$	A generic field.	
$\mathbb{N}$	The positive integers, starting from 0.	
$\mathbb{N}^+$	The strictly positive integers, without the 0.	
$\mathbb{Z}$	The integer ring.	
$I \sqcup J$	The disjoint union of the sets I and J.	
$\mathbb{F}^{n\times m}$	The set of $n \times m$ matrices on the field $\mathbb{F}$. Usually in this thesis we will have $\mathbb{F} \in \{\mathbb{C}, \mathbb{R}\}$, but some more general results are also presented.	

$\mathbb{F}^{n\times m}[x]$	The set of matrix polynomials with coefficients in the matrix ring $\mathbb{F}^{n\times m}$, or, that is the same, the set of matrices with coefficients in the ring $\mathbb{F}[x]$.	Page 33
A^t	The transpose of the matrix A	
A^*	The complex conjugated transpose of the matrix A	
$\operatorname{diag}(d_1,\dots,d_n)$	The diagonal matrix with $d_1,\dots,d_n$ on the diagonal. When $d_1,\dots,d_n$ are matrices themselves this notation is used to mean the block diagonal matrix with diagonal blocks equal to $d_1,\dots,d_n$.	
$\mathrm{fl}(F(x))$	The result of the floating point evaluation of the function $F(x)$.	
$\deg P(x)$	The degree of the polynomial $P(x)$.	
$P^{\#}(x)$	The reversed version of $P(x)$, that is, the matrix polynomial $x^{\deg P(x)}P(x^{-1})$. This can be also seen as the polynomial with the coefficients in the reversed order.	Page 35
$A(x) \sim B(x)$	The matrix polynomial $A(x)$ is unimodularly equivalent to the matrix polynomial $B(x)$.	Page 35
$A(x) \cong B(x)$	The matrix polynomial $A(x)$ is strictly equivalent to the matrix polynomial $B(x)$.	Page 35
$A(x) \smile B(x)$	The matrix polynomial $A(x)$ is extended unimodularly equivalent to the matrix polynomial $B(x)$.	Page 35
$A(x) \asymp B(x)$	The matrix polynomial $A(x)$ is spectrally equivalent to the matrix polynomial $B(x)$.	Page 36
$A \oplus B$	The block diagonal matrix with diagonal blocks equal to A and B	
$A \otimes B$	The Kronecker product of the matrices A and B. This is the block matrix whose blocks are equal to $a_{ij}B$ where $A = (a_{ij})$.	
$A(x) \mod b(x)$	When $A(x)$ is a matrix polynomial and $b(x)$ a scalar polynomial this means the element-wise projection of $A(x)$ in the ring $\mathbb{F}[x]/(b(x))$, that can be seen as the remainder of the division by $b(x)$.	
$\mathrm{tp}(x)$	A tropical polynomial. The sum, product and exponentiation in the tropical semiring are denoted as $a \oplus b$, $a \otimes b$ and $a^{\otimes b}$, respectively.	Page 16
κ_i	The condition number of the i-th eigenvalue of a matrix polynomial.	Page 55
QSH_k^n	The set of $n \times n$ Hermitian quasiseparable matrices of QS rank (k,k).	Page 82

QSH_k	The same as $\mathrm{QSH}_k n$ with the n omitted when it is clear from the context.	Page 82
$\mathcal{G}$	This notation is used for the sequences of Givens rotations.	Page 82
$\mathcal{G}v$, $\mathcal{G}^*v$	The action of sequences of Givens rotations on a vector.	Page 82
$\mathcal{G}[I]$	A subsequence of a sequence of Givens rotations, with only the one acting on the indices contained in I.	Page 82
$i : j$	The (ordered) subset of $\mathbb{Z}$ containing the integers between i and j, i.e., $i : j := \mathbb{Z} \cap [i, j]$.	Page 83
$\mathcal{G}[: j]$	A shorthand for $\mathcal{G}[2 : j]$.	Page 82
$\mathcal{G}[i :]$	A shorthand for $\mathcal{G}[i : n-1]$.	Page 82
	A pictorial representation of a Givens rotation G_i. The index i is determined by the vertical alignment of the symbol.	Page 83.
$\mathcal{S}_G^n$	The set of Givens sequences built using $n \times n$ Givens rotations	Page 82.
$\mathcal{M}(\mathcal{G})$	The matrix operator on the set of Givens sequences.	Page 88

.

1 SOLVING SCALAR POLYNOMIALS

Solving scalar polynomials is a very well-studied problem in numerical analysis. In this chapter we present a review of the most well-known methods available to tackle this problem, together with a new algorithm that has been developed to improve an available numerical package for polynomial rootfinding called `MPSolve`.

It is possible to distinguish two different classes of methods for the approximation of polynomials roots:

functional iterations This class of methods contains iterations based on the repeated application of a fixed point map that has the roots of the polynomials as fixed points. As we will see, this kind of iterations often has very good local convergence properties but it is difficult, in general, to provide proofs for the global convergence, and in some cases it is even possible to show that not all the starting points lead to convergence. Anyway, it is often possible to choose starting points wisely in order to avoid these unfortunate cases.

eigenvalue methods This class of methods relies on linear algebra in order to obtain the roots of the polynomials. The basic idea is to find a matrix such that its eigenvalues are the roots of the polynomials, and then to use a standard eigenvalue method to obtain them. The resulting matrices (the so-called companion matrices) are endowed with particular structures, that are not always easy to use in the practical computations.

1.1 STANDARD SIMULTANEOUS ITERATION METHODS

In this section we present some classical iteration algorithms for the solution of scalar polynomials. We are mainly interested in functional iterations, i.e., algorithms that start from a certain number of approximations of the roots $x_1^{(0)}, \dots, x_n^{(0)}$, and perform an iteration of this kind:

$$x_i^{(k+1)} = F_i(x_1^{(k)}, \dots, x_n^{(k)}).$$

Notice that F_i can be a different function depending on the index of the approximation that is being updated, and that the update to a single approximation might depend from all the other approximations.

As we will see, the main tool used to develop such simultaneous methods is the Newton's iteration, that we recall here for completeness and to fix the notation.

Definition 1.1.1. Let $p(x) = \sum_{i=0}^{n} p_i x^i$ be a scalar polynomial of degree n. The complex number

$$N_p(x) = \frac{p(x)}{p'(x)}$$

is called the *Newton's correction* of $p(x)$ at the point x.

The Newton's method can be defined as the iteration of the Newton's correction to a single approximation:

$$x^{(k+1)} = x^{(k)} - N_p(x^{(k)})$$

where $x^{(0)}$ is chosen to be a guess for a root ξ of the polynomial $p(x)$. The Newton's method has local quadratic convergence for simple roots and linear convergence to multiple roots. Its main limitation is that it is not easy to use it to approximate all the roots for a polynomial at once.

In fact, it is difficult to control to which root the method is convergent, or if it is convergent at all. It should be mentioned that it is possible to use the method to approximate all the roots by using the strategy proposed in [70]. In this work the authors propose to choose more than n starting approximations placed on a circle of radius sufficiently large. They show that in this case it is guaranteed that at least one approximation is contained in the attraction basin of each root, so that the Newton's method applied to all these approximations will converge to all the roots (possibly more than once). More precisely, the authors show that $O(n \log^2 n)$ starting points are enough in order to approximate all the roots. However, the cost of this approach may be not optimal, since in general the computation of $N_p(x)$ costs $O(n)$ flops. This leads to a total cost of $O(t \cdot n^2 \log^2 n)$ flops where t is the mean number of iterations needed to obtain the approximations within an error bound ϵ. Numerical experiments show that t may be as large as $O(n)$, thus leading to a total cost of $O(n^3 \log^2 n)$ flops [15].

Several authors tried in the past to modify Newton's method to achieve simultaneous approximation of all the roots of a polynomial. We recall here two basic methods, namely the Durand–Kerner and the Ehrlich–Aberth algorithm.

Both can achieve simultaneous approximation with a cost of $O(n^2)$ flops per step. We show that, experimentally, it is possible to choose the starting points in a smart way so that practically a constant number of iterations per roots are required, thus obtaining an algorithm with quadratic cost for the approximations of the roots.

1.1.1 *Durand–Kerner algorithm*

The Durand–Kerner iteration, also known as the Weierstrass iteration, has been first introduced by Weierstrass in [92] and then by Durand in [45] and Kerner in [72].

Given a set of approximations $x_1^{(k)}, \dots, x_n^{(k)}$ of the roots of a polynomial $p(x)$, the Durand–Kerner iteration is defined by

$$x_i^{(k+1)} = x_i^{(k)} - \frac{p(x_i^{(k)})}{\prod_{\substack{j=1 \\ j \neq i}}^{n} (x_i^{(k)} - x_j^{(k)})}.$$

The Durand–Kerner iteration has a local quadratic convergence to simple roots, and linear convergence on multiple roots. However, it is important to note that if ξ is a multiple root of $p(x)$ of order s then s approximations $x_{i_1}^{(k)}, \dots, x_{i_s}^{(k)}$ will converge to ξ. In this case it can be shown that even if each of the $x_{i_j^{(k)}}$ converges only linearly to ξ, their mean $\frac{1}{s}(x_{i_1}^{(k)} + \dots + x_{i_s}^{(k)})$ still has the quadratic convergence property [54].

1.1.2 *Ehrlich–Aberth iteration*

Here we describe another important iteration for the approximation of the roots of a polynomial, originally introduced by Oliver Aberth in [1] and by Ehrlich in [46]. As usual, suppose that we have a set of approximations $x_1^{(k)}, \dots, x_n^{(k)}$ of the roots of the monic polynomial $p(x)$. We consider, for each $i = 1, \dots, n$, the rational function defined by

$$f_i(x) = \frac{p(x)}{\prod_{\substack{j=1\\ j\neq i}}^{n} (x - x_j^{(k)})}, \qquad i = 1, \dots, n.$$

Recall that since we can write $p(x) = p_n \prod_{i=1}^{n} (x - \xi_i)$ if the approximations $x_j^{(k)}$ are good approximations of ξ_j then $f_i(x)$ is a good approximation of $p_n(x - \xi_i)$. Being aware of this we can define the following iteration:

$$x_i^{(k+1)} = x_i^{(k)} - N_{f_i}(x_i^{(k)}). \tag{1.1}$$

If $x_j^{(k)} = \xi_j$ for every $j \neq i$ then $f_i(x)$ is a linear function and so we can expect that $x_i^{(k+1)} = \xi_i$, since the Newton's method applied to linear functions converges in one step. When this is not the case we can show that the method converges cubically to simple roots. This is proved, for example, in [1].

It should be noted that the efficiency of these methods is strictly connected with the quality of the starting approximations. Both for Ehrlich–Aberth and Durand–Kerner iterations, in fact, the total cost for the factorization of a polynomial is $O(t \cdot n^2)$ where t is the mean number of iterations needed for every root. In the original paper by Aberth [1] the author suggests to place the initial approximations on a circle of radius large enough to contain all the roots. This method works relatively well, and experimentally it is possible to show that this leads to $t = O(n)$ (see for example [15]). We show in Section 1.4 how to make a smarter choice that often provides starting points accurate enough in order to make $t = O(1)$. This leads to a method with quadratic complexity.

Expanding the derivatives of formula 1.1 leads to the following more explicit (but somewhat involved) expression

$$x_i^{(k+1)} = x_i^{(k)} - \frac{\frac{p(x_i^{(k)})}{p'(x_i^{(k)})}}{1 - \frac{p(x_i^{(k)})}{p'(x_i^{(k)})} \cdot \sum_{j=1}^{n} \frac{1}{x_i^{(k)} - x_j^{(k)}}} \tag{1.2}$$

It is important to stress the following fact, that will used several times in the following.

Remark 1.1.2. In order to apply the Ehrlich–Aberth iteration to a polynomial $p(x)$ we do not necessarily need to explicitly know its coefficients. It is sufficient to being able to evaluate (possibly in a numerically stable way) its Newton's correction $\frac{p(x)}{p'(x)}$.

1.2 LINEARIZATIONS

A standard approach for the solution of scalar polynomial equations is the use of linearizations. In this section we survey the basic notions about linearizations and we provide the theory needed to cover the scalar case. Let $p(x) = \sum_{i=0}^{n} p_i x^i$ be a scalar polynomial of degree n and assume that $\xi_1, \dots, \xi_n$ are its roots.

1.2.1 *Companion matrices*

We introduce the concept of companion matrix for a polynomial $p(x)$.

Definition 1.2.1. We say that A is a *companion matrix* for a polynomial $p(x)$ if its entries are obtained directly from the coefficients of the polynomial through the use of elementary operations and its eigenvalues and their multiplicities matches the ones of the roots of the polynomial $p(x)$.

Observe that the previous definition is not really precise, in a mathematical sense, since we avoided to be very specific on what we consider "elementary" operations. This is intended, since the concept of companion matrix changes in different contexts but is linked to this principle: the companion matrix can be obtained starting from the coefficients of $p(x)$ with little effort, typically in a way that each entry of the matrix A can be computed in a constant time, independent of the degree n.

The main property of such a matrix A is that the spectrum of A coincides with the roots of $p(x)$. More precisely, it can be directly verified that $\det(xI - A) = p(x)$ if $p(x)$ is monic. This property can be exploited to approximate the roots of $p(x)$ by means of an eigenvalue solver, such as the QR method. This is precisely the strategy used by popular mathematical software when the user requests the roots of a polynomial specified via its coefficients. A well-known example is the command `roots` found in MATLAB$^{\mathrm{TM}}$.

Many companion matrices have been introduced over the years in the literature. The most famous and widespread are the Frobenius forms. We present here one of the possible choices for the Frobenius form. Other possibilities are reported in Section 2.1 where the case of matrix polynomials is discussed. We note that in the scientific literature these companion matrices, having being the de-facto standard for a long time, are sometimes called simply companion matrices.

Definition 1.2.2. Given a scalar polynomial $p(x)$ of degree n the *Frobenius form* of $p(x)$ is the $n \times n$ matrix F defined as

$$F = \begin{bmatrix} -p_n^{-1}p_{n-1} & \cdots & \cdots & -p_n^{-1}p_0 \\ 1 & & & 0 \\ & \ddots & & \vdots \\ & & 1 & 0 \end{bmatrix}.$$

Observe that the matrix F is highly structured. More precisely, we can observe that only $O(n)$ elements are different from 0 and that the matrix is already in upper Hessenberg form. Moreover, F can be decomposed as $F = Z - p_n^{-1} e_1 q^t$ where

$$Z = \begin{bmatrix} 0 & \cdots & 0 & 1 \\ 1 & & & \\ & \ddots & & \\ & & 1 & 0 \end{bmatrix}, \qquad q = \begin{bmatrix} p_{n-1} \\ \vdots \\ p_1 \\ p_0 + p_n \end{bmatrix}, \qquad e_1 = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

so that F is a rank 1 correction of a unitary matrix. The sparsity of F is not preserved in the iterations of the QR method, but the unitary plus rank 1 structure is maintained. In [5] the authors have exploited this structure in order to devise a fast QR iteration that only uses $O(n)$

storage and $O(n^2)$ time. Others algorithms exploiting this structure have been introduced by many authors in recent years. Several contributions have been given by Aurentz, Bini, Boito, Chandrasekaran, Del Vaux, Eidelman, Gemignani, Gohberg, Gu, Mach, Van Barel, Vandebril, Watkins and Xia. A selection of these works can be found in [13, 14, 28, 56, 87, 23].

1.2.2 *Linearizations in different basis*

In principle, it is possible to construct many different matrices that have the roots of a polynomial as eigenvalues. This suggests that there is some room to make better choices in order to improve some of the properties of the companion matrix.

In the previous subsection we have introduced companion matrices built starting from the coefficients of the polynomial in the monomial basis. While this is a very typical choice to represent a polynomial, it is not always the case and, more importantly, it is not very well-suited from a numerical point of view. For this reason, we introduce some other companion forms, that can be used to solve polynomials represented in different basis.

In order to achieve this result we present a slightly more abstract version of these companion matrices. Consider the ring of univariate polynomials on a field $\mathbb{F}$ that we denote by $\mathbb{F}[x]$, and a polynomial $p(x) \in \mathbb{F}[x]$. In the following we assume for simplicity that $p(x)$ is monic, but this is not a strict requirement. Consider $\overline{\mathbb{F}} \supseteq \mathbb{F}$ one algebraic closure of $\mathbb{F}$ so that there exist $\xi_1, \dots, \xi_n \in \overline{\mathbb{F}}$ such that $p(x) = \prod_{i=1}^{n}(x - \xi_i)$. We have that the quotient ring $S_p := \mathbb{F}[x]/(p(x))$ is a n-dimensional vector space on the field $\mathbb{F}$. Consider the linear application L defined by the multiplication by x in S_p. We have that for every $\xi_i \in \mathbb{F}$ the polynomial $p_i(x) := \prod_{j \neq i}(x - \xi_j) \in \mathbb{F}[x]$ and $L(\overline{p_i(x)}) = \xi_i \overline{p_i(x)}$ where $\overline{q(x)}$ denotes the projection of $q(x)$ in the quotient ring. In particular, ξ_i is an eigenvalue for the linear operator L. More precisely, it can be shown that the roots of $p(x)$ are the only eigenvalues of L since

$$L(\overline{q(x)}) = \xi \overline{q(x)} \iff (x - \xi) q(x) \equiv 0 \mod p(x)$$

and so $(x - \xi)$ divides $p(x)$ thanks to the condition $\overline{q(x)} \neq 0$. The above remarks suggest the following definition:

Definition 1.2.3. We say that a linear operator $L(x)$ on a vector space of dimension n is a *companion linear operator* for a polynomial of degree n if and only if its eigenvalues coincides with the roots of $p(x)$ in an algebraic closure.

Observe that at this point we can choose an arbitrary basis of the polynomials of degree less than n and represent a companion operator $L(x)$ as a matrix. Since the concept of eigenvalue carries over from the more abstract formulation, every choice of basis automatically provides a companion matrix for the polynomial $p(x)$.

Moreover, since a basis for the quotient ring S_p can be seen as a basis for the polynomial of degree less than n, we also have a companion matrix for every choice of a basis for the polynomials of degree at most $n - 1$, thus removing the dependency on the specific polynomial $p(x)$.

As a first example, consider the basis $1, x, \dots, x^{n-1}$. We have that, in the quotient ring S_p, $L(x^i) = x \cdot x^i = x^{i+1}$ for every $i < n - 1$. For $i = n - 1$ we have

$$L(x^{n-1}) = x^{n-1} \cdot x \equiv - \sum_{i=0}^{n-1} p_i x^i \mod p(x).$$

With this information we can build the matrix for the linear operator L in this basis, and we obtain the companion matrix of Definition 1.2.2.

$$F = \begin{bmatrix} 0 & & & -p_0 \\ 1 & & & \vdots \\ & \ddots & & \vdots \\ & & 1 & -p_{n-1} \end{bmatrix}.$$

This makes clear how this more abstract version of the companion matrices is in fact a generalization of the definitions given in the previous section.

Here we present another concrete choice of basis and the resulting companion form. An additional choice for the basis in the rootfinding setting is given in Section 1.3.

1.2.3 *Chebyshev linearizations*

As a practical example of the above framework, we consider the set of Chebyshev polynomials of the first kind defined by the recurrence relation

$$T_i(x) = \begin{cases} 1 & \text{if } i = 0 \\ x & \text{if } i = 1 \\ 2xT_{i-1}(x) - T_{i-2}(x) & \text{otherwise} \end{cases}.$$

Assume that we have the coefficients of a polynomial $p(x)$ represented in the Chebyshev basis:

$$p(x) = T_n(x) + \sum_{i=0}^{n-1} p_i T_i(x).$$

Here we assume that the polynomial is monic for simplicity but, as in the previous case, this can be obtained without loss of generality by simply dividing the polynomial for the leading coefficient. Notice that a monic polynomial in the Chebyshev basis does not correspond to a monic polynomial in the monomial basis, since T_n has 2^{n-1} as leading coefficient.

We study the usual operator L representing the multiplication for the polynomial $x = T_1(x)$ in the quotient ring $S_p = \mathbb{F}[x]/(p(x))$. A direct inspection shows that for any $0 < i < n-1$ we have $xT_i(x) = \frac{1}{2}T_{i+1}(x) + \frac{1}{2}T_{i-1}(x)$. For $i = 0$ we have $xT_0(x) = T_1(x)$ and when $i = n-1$ the following relation holds:

$$xT_{n-1}(x) \equiv \frac{1}{2}T_{n-2}(x) + \frac{1}{2}T_n(x) \equiv \frac{1}{2}T_{n-2}(x) - \frac{1}{2}\sum_{i=0}^{n-1} p_i T_i(x) \mod p(x).$$

This leads to the following representation for the operator L in the basis defined by $T_0(x), \dots, T_{n-1}(x)$:

$$C = \begin{bmatrix} 0 & \frac{1}{2} & & & -\frac{1}{2}p_1 \\ 1 & \ddots & \ddots & & \vdots \\ & \frac{1}{2} & \ddots & \frac{1}{2} & -\frac{1}{2}p_{n-3} \\ & & \ddots & 0 & \frac{1}{2} - \frac{1}{2}p_{n-2} \\ & & & \frac{1}{2} & -\frac{1}{2}p_{n-1} \end{bmatrix}.$$

Note that also in this case the companion matrix for the Chebyshev basis is highly structured. If an appropriate scaling is introduced, C can be decomposed as the sum of a real symmetric tridiagonal matrix plus a rank 1 correction.

These two examples show how this framework can be used to obtain companion matrices for any polynomial basis. The same procedure can be applied almost verbatim to obtain companion matrices for any orthogonal basis, for example. The presence of a recurrence relation with a limited number of terms guarantees that the resulting matrix will be sparse, while asking that the basis is degree-graded gives the upper Hessenberg structure.

1.3 SECULAR BASIS AND REGENERATIONS

In this section we introduce a new kind of linearization by looking at somewhat different algebraic objects with respect to polynomials: secular equations. Consider, for complex coefficients $a_i \neq 0$ and pairwise distinct b_i, the following *secular equation*:

$$S(x) = \sum_{i=1}^{n} \frac{a_i}{x - b_i} - 1 = 0. \tag{1.3}$$

where $S(x)$ is called a *secular function*. In the following we will call b_i the *nodes* of the secular equation and a_i its *weights*.

1.3.1 *Polynomials and secular equations*

Since the nodes b_i cannot be solutions of (1.3) we can solve it by multiplying by $\Pi(x) = \prod_{i=1}^{n}(x - b_i)$. Then the solutions of $S(x) = 0$ are the roots of the monic polynomial $p(x)$ defined by

$$p(x) = -\Pi(x)S(x) = \Pi(x) - \sum_{i=1}^{n} a_i \prod_{j \neq i}(x - b_j). \tag{1.4}$$

This equation can also be read the other way round: every polynomial of the form of the right-hand side of (1.4) can be solved by solving the associated secular equation with nodes b_i and weights a_i.

Lemma 1.3.1. *Let $b_1, \dots, b_n$ be pairwise distinct points in the complex plane. Then for every monic polynomial $p(x) \in \mathbb{C}[x]$ of degree n such that $p(b_i) \neq 0$ for $i = 1, \dots, n$ there exist nonzero weights $a_1, \dots, a_n$ such that*

$$p(x) = \Pi(x) - \sum_{i=1}^{n} a_i \prod_{j \neq i}(x - b_j).$$

Proof. Notice that $\prod_{j \neq i}(x - b_j)$ are scaled versions of the Lagrange polynomials of degree $n - 1$. Since $p(x)$ is monic $\Pi(x) - p(x)$ is of degree $n - 1$ and admits a representation

$$\Pi(x) - p(x) = \sum_{i=1}^{n} a_i \prod_{j \neq i}(x - b_j).$$

Moreover, since the a_i are simply scaled version of the Lagrange coefficients of $p(x) - \Pi(x)$ and $\Pi(b_i) = 0$ we have

$$a_i = \frac{-p(b_i)}{\prod_{j \neq i}(b_i - b_j)}, \qquad i = 1, \dots, n. \tag{1.5}$$

The hypothesis on $p(b_i) \neq 0$ guarantees that $a_i \neq 0$. □

Definition 1.3.2. If $S(x) = \sum_{i=1}^{n} \frac{a_i}{x-b_i} - 1 = 0$ is a secular equation whose weights have been computed using Lemma 1.3.1, that is the equality (1.5) holds, we say that $S(x)$ is a secular equation *equivalent* to $p(x)$. In particular, in this case the solution of $S(x) = 0$ are exactly the roots of $p(x)$.

Lemma 1.3.1 allows to shift from the problem of finding the roots of a polynomial to the one of solving a secular equation. It is natural to ask if there is a practical way to exploit this fact.

For us, the main interest in secular equations is the following Theorem, originally presented in [51].

Theorem 1.3.3. *Let* $S(x) = \sum_{i=1}^{n} \frac{a_i}{x-b_i} - 1$ *be a secular function as defined in* (1.3)*. Then the matrix* $A = D - ea^t$ *where*

$$D = \begin{bmatrix} b_1 & & \\ & \ddots & \\ & & b_n \end{bmatrix}, \qquad e = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}, \qquad a = \begin{bmatrix} a_1 & \cdots & a_n \end{bmatrix}$$

has the solutions of $S(x) = 0$ *as eigenvalues.*

Proof. Note that for any $x \neq b_i$

$$\det(xI - D + ea^t) = \det(xI - D)\det(I + (xI - D)^{-1}ea^t).$$

Since $\det(I + uv^t) = 1 - v^t u$ we have that

$$\det(xI - D + ea^t) = \prod_{i=1}^{n}(x - b_i)\left(1 - \sum_{i=1}^{n} \frac{a_i}{x - b_i}\right) = -\Pi(x)S(x).$$

Since $\Pi(x)$ vanishes only at $x = b_i$ which are not solutions of $S(x) = 0$ by hypothesis, we conclude that the eigenvalues of $xI - D + ea^t$ are exactly the solutions of our equation. □

Theorem 1.3.3, combined with the remarks about the connection between secular equations and polynomials, leads to the following corollary.

Corollary 1.3.4. *Let* $p(x)$ *be a scalar polynomial and* $b_1, \dots, b_n$ *be complex numbers such that* $p(b_i) \neq 0$ *for any* $i = 1, \dots, n$*. Then the matrix polynomial*

$$A(x) = xI - D + ea^t, \qquad a_i = \frac{-p(b_i)}{\prod_{j \neq i}(b_i - b_j)}$$

is a linearization for $p(x)$*.*

1.3.2 *Numerical properties of secular equations*

We have shown that it is easy to move from polynomials to secular functions. We want to show that the latter class of functions enjoys very nice numerical properties. Moreover, we introduce two concepts that will be useful for the analysis of the convergence: the *root neighborhood* and the *secular neighborhood*.

Consider Algorithm 1 for the evaluation of $S(x) + 1$. The algorithm evaluates the sums recursively by splitting the set of summands in two (almost) equal parts. We have then

Algorithm 1 Evaluation of a secular function at a point in the complex plane.

```
1: function SecularEvaluation(a, b, x)
2:     $n = \textsc{Length}(a)$
3:     if $n == 1$ then
4:         return $\frac{a_1}{x-b_1}$
5:     else
6:         $\ell \leftarrow \lfloor \frac{n}{2} \rfloor$
7:         $s_- \leftarrow \textsc{SecularEvaluation}(a[1:\ell], b[1:\ell], t)$
8:         $s_+ \leftarrow \textsc{SecularEvaluation}(a[\ell+1:n], b[\ell+1:n], t)$
9:         return $s_- + s_+$
10:    end if
11: end function
```

Lemma 1.3.5. *Consider the use of Algorithm 1 for the evaluation of the secular function $S(x)$ and assume that the coefficients a_i and b_i can be represented as floating point numbers. We can use it to evaluate the sum $S_+(x)$ and then obtain $S(x) = S_+(x) - 1$ where*

$$S_+(x) = \sum_{i=1}^{n} \frac{a_i}{x - b_i}.$$

We have that, if u is the unit roundoff at the current machine precision,

$$\mathrm{fl}(S_+(x)) = \left(\sum_{i=1}^{n} \frac{a_i(1+\theta_i)}{x - b_i}(1+\delta) \right), \qquad |\delta| \leqslant u, \quad |\theta_i| \leqslant \frac{u}{1-u}(\lceil \log_2 n \rceil + 7\sqrt{2}).$$

Proof. By using the bounds on floating points arithmetic reported in [69] it is easy to show that

$$\mathrm{fl}\left(\frac{a_i}{x-b_i} \right) = \left(\frac{a_i}{x-b_i} \right)(1+\epsilon_i), \qquad |\epsilon_i| \leqslant \frac{u}{1-u}(1+7\sqrt{2})u.$$

The result is then obtained by moving the error term ϵ_i on the a_i and taking into account the error propagation due to the parallel sum of Algorithm 1. □

Define κ_n and $\sigma(x)$ as follows

$$\kappa_n := \frac{1}{1-u}\lceil \log_2 n \rceil + 7\sqrt{2}, \qquad \sigma(x) = \sum_{i=1}^{n} \frac{|a_i|}{|x-b_i|}.$$

Then, by Lemma 1.3.5 we obtain the following.

Corollary 1.3.6. *The floating point evaluation of a secular function at a machine precision with unit roundoff u is such that*

$$|\mathrm{fl}(S(x))| \leqslant (1+u)|S(x)| + \kappa_n \sigma(x) u (1+u), \qquad |S(x)| \leqslant \frac{1}{1-u} |\mathrm{fl}\, S(x)| + u\kappa_n \sigma(x).$$

Proof. The statement is a direct consequence of Lemma 1.3.5 by writing

$$\mathrm{fl}(S(x)) = S(x) + \sum_{i=1}^{n} \frac{a_i \theta_i}{x - b_i} + \delta S(x).$$

Recalling that $|\theta_i| \leqslant \kappa_n u$ and that $|\delta| \leqslant u$ leads to the thesis by rearranging the terms and using the triangular inequality. □

The consequences of this corollary deserve to be stressed: we are able to rigorously bound the error on the evaluation of a secular function using its floating point evaluation, and at the same time we are sure that whenever the actual evaluation of the secular function is "small" the same will also hold for its floating point evaluation.

We are now interested in studying the set of "nearby" solutions of a secular equation. The following definitions will make this concept less vague.

Definition 1.3.7 (Secular neighborhood)**.** The ϵ-secular neighborhood of a secular function $S(x)$ is the set of secular functions $SN_\epsilon(S(x))$ defined by

$$SN_\epsilon(S(x)) := \left\{ \tilde{S}(x) = \sum_{i=1}^{n} \frac{\tilde{a}_i}{x - b_i} - 1 \mid |\tilde{a}_i - a_i| \leqslant \epsilon |a_i| \right\}$$

Notice that the definition is given by perturbing only the weights of the secular equations, and not the nodes. This is linked to the fact that Lemma 1.3.5 guarantees that the floating point evaluation of a secular function is the evaluation of a nearby secular function with only the weights perturbed.

It is now natural to give the following definition.

Definition 1.3.8 (Root neighborhood)**.** The ϵ-root neighborhood of a secular function is the set of solutions of $\tilde{S}(x) = 0$ for $\tilde{S}(x) \in SN_\epsilon(S(x))$. More formally we have

$$RN_\epsilon(S(x)) = \left\{ \bar{x} \mid \exists\ \tilde{S}(x) \in SN_\epsilon(S(x)),\ \tilde{S}(\bar{x}) = 0 \right\}.$$

In Figure 1.1 the root neighborhoods of a simple secular equation are plotted with different colors for different values of ϵ (darker colors represents larger ϵ).

Remark 1.3.9. Recalling that the solution of a secular equation $S(x) = 0$ are the eigenvalues of a diagonal plus rank 1 matrix we can conclude that the root neighborhood can also be seen as a structured pseudospectra of these matrices. In fact, a perturbation on the vector of the a_i coefficients corresponds to a perturbation that transforms the generalized companion matrix $D - ea^t$ into $D - e\tilde{a}^t$.

The root neighborhood is linked with the concept of conditioning of the rootfinding problem. The condition number of the secular equation $S(x) = 0$ is a measure of how much the roots change when we perturb the coefficients of the equation.

Since we have seen that in our analysis the error can be offloaded entirely on the coefficients a_i, we define the condition number of a secular equation $S(x) = 0$ in the following way.

Definition 1.3.10 (Condition number)**.** Let ξ_i be one of the (simple) solutions of a secular equation $S(x) = 0$ as in Equation (1.3). Consider the function $\Gamma_S(a)$ defined as

$$\Gamma_{S,i}(\tilde{a}) := \min \left\{ |\xi_i - \xi|, \quad \sum_{j=1}^{n} \frac{\tilde{a}_j}{\xi - b_j} = 1 \right\}.$$

We define the condition number of $S(x) = 0$ relative to ξ_i as $|\Gamma'_{S,i}(a)|$.

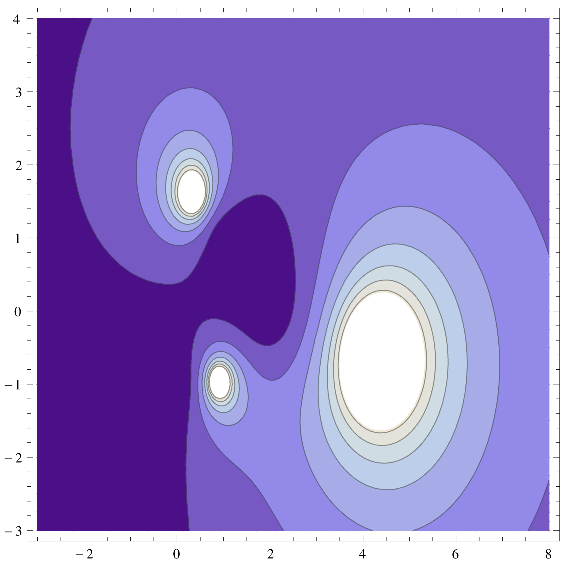


Figure 1.1: Root neighborhoods of the secular equation $\frac{5}{2(x-2)} - \frac{2}{x-1-i} + \frac{2}{x+i} - 1 = 0$ for various values of ϵ.

Remark 1.3.11. In order to guarantee that the above definition makes sense we need to be sure that Γ_S is derivable. In fact, this can be guaranteed at least in a neighborhood of ξ_i. This can be obtained, for example, remembering that the roots of the secular equations are the eigenvalues of the matrix $D + ea^t$. A small enough perturbation will move the eigenvalues so that the nearest eigenvalue to ξ_i is uniquely determined and so that $\Gamma_{S,i}$ is analytic [71].

The concepts of secular and root neighborhoods are strictly linked together. To design an algorithm that is able to approximate the roots of a secular equation we are likely to want to prove that the final approximations are contained in $RN_\epsilon(S(x))$ for a small enough value of ϵ. Apparently, though, it is much easier to verify if a $\tilde{S}(x) \in SN_\epsilon(S(x))$ than to check if a set of approximations are contained in the root neighborhood relative to a certain ϵ.

The following theorem gives an explicit criterion for the verification of the latter condition.

Proposition 1.3.12. *Let $S(x)$ be a secular equation as in (1.3). Then the set $RN_\epsilon(S(x))$ can be described as*

$$RN_\epsilon(S(x)) = \{x \mid |S(x)| \leqslant \epsilon\sigma(x)\}, \qquad \sigma(x) = \sum_{i=1}^{n} \frac{|a_i|}{|x - b_i|}.$$

Proof. Let us call $Y = \{x \mid |S(x)| \leqslant \epsilon\sigma(x)\}$. We want to prove that $Y = RN_\epsilon(S(x))$. Note that if $\xi \in RN_\epsilon(S(x))$ then there exists a slightly perturbed secular equation $\tilde{S}(x)$ such that $\tilde{S}(\xi) = 0$. In particular the weights $\tilde{a}_i$ of $\tilde{S}(x)$ are equal to $a_i(1 + \epsilon_i)$, $|\epsilon_i| \leqslant \epsilon$. This leads to

$$0 = \tilde{S}(\xi) = \sum_{i=1}^{n} \frac{\tilde{a}_i}{\xi - b_i} - 1 = \sum_{i=1}^{n} \frac{a_i}{\xi - b_i} - 1 + \sum_{i=1}^{n} \frac{\epsilon_i a_i}{\xi - b_i} = S(\xi) + \sum_{i=1}^{n} \frac{\epsilon_i a_i}{\xi - b_i}.$$

Rearranging the terms we have

$$|S(\xi)| = \left| \sum_{i=1}^{n} \frac{\epsilon_i a_i}{\xi - b_i} \right| \leqslant \epsilon\sigma(\xi).$$

On the other hand, if $|S(\xi)| \leqslant \epsilon\sigma(\xi)$ then we can set $\eta = S(\xi)$ and choose

$$\tilde{a}_i = a_i \left(1 - \frac{\xi - b_i}{a_i} \frac{|a_i|}{|\xi - b_i|} \frac{\eta}{|\sigma(\xi)|} \right).$$

A direct computation then shows that $|\tilde{a}_i - a_i| \leqslant |a_i|\epsilon$ and that $\tilde{S}(\xi) = 0$. □

The above result gives us a practical check to verify if a complex point x lies in the root neighborhood for $S(x)$ relative to a certain ϵ. It is natural to ask whether it is safe to check this in floating point. Can we trust the result obtained from this computation?

In view of Proposition 1.3.12 let us define the floating point version of the root neighborhood as

$$\widetilde{\mathrm{RN}}_\epsilon(S(x)) := \{x \in \mathbb{C} \mid \mathrm{fl}\,|S(x)| \leqslant \epsilon\sigma(x)\}.$$

The following lemma helps us to characterize the connection between the floating point and the standard root neighborhoods.

Lemma 1.3.13. *Let $S(x)$ be a secular function. Then, for any $\epsilon \geqslant (1+u)u\kappa_n$, the following inclusions hold:*

$$\mathrm{RN}_{\frac{\epsilon}{1+u} - u\kappa_n}(S(x)) \subseteq \widetilde{\mathrm{RN}}_\epsilon(S(x)) \subseteq \mathrm{RN}_{\frac{1}{1-u}\epsilon + u\kappa_n}(S(x)),$$

$$\widetilde{\mathrm{RN}}_{(1-u)(\epsilon - u\kappa_n)}(S(x)) \subseteq \mathrm{RN}_\epsilon(S(x)) \subseteq \widetilde{\mathrm{RN}}_{(1+u)(\epsilon + u\kappa_n)}(S(x)).$$

Proof. Let us start from the inclusion $\mathrm{RN}_\epsilon(S(x)) \subseteq \widetilde{\mathrm{RN}}_{(1+u)(\epsilon+u\kappa_n)}(S(x))$. If $\xi \in \mathrm{RN}_\epsilon(S(x))$ then $|S(x)| \leqslant \epsilon\sigma(x)$. Applying Corollary 1.3.6 we have that

$$|\,\mathrm{fl}\,S(\xi)| \leqslant (1+u)|S(x)| + \kappa_n\sigma(x)u(1+u) \leqslant (1+u)\,(\epsilon + u\kappa_n)\,\sigma(x)$$

that gives us the thesis. Corollary 1.3.6 can be applied also to obtain the other inclusion, that is $\widetilde{\mathrm{RN}}_\epsilon(S(x)) \subseteq \mathrm{RN}_{\frac{1}{1-u}\epsilon + u\kappa_n}(S(x))$ by noting again that if $\xi \in \widetilde{\mathrm{RN}}_\epsilon(S(x))$ then $|\,\mathrm{fl}\,S(x)| \leqslant \epsilon\sigma(x)$ and then

$$|S(x)| \leqslant \frac{1}{1-u}\epsilon\sigma(x) + u\kappa_n\sigma(x) \leqslant \left(\frac{1}{1-u}\epsilon + u\kappa_n \right) \sigma(x).$$

□

1.3.3 *The relevance of the choice of nodes*

As we have seen in the previous subsection, when a polynomial $p(x)$ is given, the nodes $b_1, \ldots, b_n$ can be chosen freely provided that they are pairwise distinct. We can investigate if smart choices of the nodes lead to secular equations with good numerical properties.

In fact, it is worth remembering that this kind of secular equations is very much related to Lagrange interpolation. If we want to find the roots of a polynomial, that is, the solutions of a secular equation, it might be reasonable to choose b_i as good approximations to the zeros. The aim of this subsection is exactly to analyze the behavior of the conditioning for the rootfinding problem when the $b_i \to \xi_i$, where $p(\xi_i) = 0$.

We have already given the definition of the condition number in Definition 1.3.10, and we recall that it is the modulus of the derivative of the change of the solutions relative to the perturbations of the weights a_i. So, in order to evaluate it, we study the change of a root ξ when we perturb a weight a_j, with $1 \leqslant j \leqslant n$ fixed. The general case can be seen as a composition of n such perturbations.

Let $\hat{a}_i = a_i$ for $i \neq j$ and $\hat{a}_j = a_j(1+\epsilon_j)$. Suppose then that ξ is a root of $S(x)$ and $\hat{\xi}$ is a root of $\hat{S}(x)$ the secular equation with $\hat{a}_j$ as weights and b_i as nodes. We have

$$\begin{cases} S(\xi) := \sum_{i=1}^{n} \frac{a_i}{\xi - b_i} - 1 = 0 \\ \hat{S}(\hat{\xi}) := \sum_{i=1}^{n} \frac{\hat{a}_i}{\hat{\xi} - b_i} - 1 = 0 \end{cases}$$

Let $\delta_\xi := \hat{\xi} - \xi$. Subtracting on both sides yields

$$S(x) - \hat{S}(\xi) = \sum_{i=1}^{n} \frac{a_i \delta_\xi}{(\xi - b_i)(\hat{\xi} - b_i)} - \frac{a_j \epsilon_j}{\hat{\xi} - b_j} = 0.$$

Recall that we want to estimate the variation of the roots as a function of the variation of the coefficients, so in this case we can write

$$\frac{\delta_\xi}{\epsilon_j} = \frac{a_j}{(\hat{\xi} - b_j) \cdot \left(\sum_{i=1}^{n} \frac{a_i}{(\xi - b_i)(\hat{\xi} - b_i)} \right)}.$$

The first order expansion of the above expression with respect to δ_ξ gives

$$\left| \frac{\delta_\xi}{\epsilon_j} \right| = \frac{|a_j|}{|\xi - b_j| \cdot |S'(\xi)|} + O(\delta_\xi)$$

Moreover, if we suppose that all the coefficients a_i have been perturbed with a relative perturbation $|\epsilon_i| \leqslant |\epsilon|$ we can compose the above bound for n times and obtain the first order estimate

$$\left| \frac{\delta_\xi}{\epsilon} \right| \dot{\leqslant} \frac{|\sigma(\xi)|}{|S'(\xi)|}, \qquad \sigma(x) = \sum_{i=1}^{n} \frac{|a_i|}{|x - b_i|}. \tag{1.6}$$

Notice that the expression $\limsup_{\epsilon \to 0} \left| \frac{\delta_\xi}{\epsilon} \right|$ can be taken as another definition of conditioning, which is defined also in the case of multiple roots. In the following we will consider this formulation in order to be able to bound the conditioning also in the case of multiple roots.

Now it is natural to ask whether accurate choices of nodes lead to small conditioning number of the rootfinding problem. This would be interesting in order to devise an effective numerical algorithm for the computation of the roots of $p(x)$. We can state the following theorem.

Theorem 1.3.14. *Let $b_i^{(k)}$ be a sequence of nodes and $S^{(k)}(x)$ the secular equations equivalent to a polynomial $p(x)$ computed on the nodes $b_i^{(k)}$, $i = 1, \dots, n$. Suppose that $\lim_{k \to \infty} b_i^{(k)} = \xi_i$ where ξ_i are the roots of $p(x)$. Then we have*

(i) *If ξ_i is a simple root then the conditioning of Equation (1.6) relative to ξ_i for the secular equation $S^{(k)}(x)$ goes to 0 as $k \to \infty$, that is,*

$$\lim_{k\to\infty} \frac{\sigma^{(k)}(\xi)}{(S^{(k)}(\xi))'} = 0, \qquad \sigma^{(k)}(x) := \sum_{i=1}^{n} \frac{|a_i^{(k)}|}{|x - b_i^{(k)}|}.$$

(ii) *If $\xi := \xi_i = \ldots = \xi_{i+m-1}$ is a multiple root of order m and $b_i^{(k)}, \ldots, b_{i+m-1}^{(k)}$ converge to ξ in a way that the ratio $\frac{(\xi - b_i^{(k)})}{b_j^{(k)} - b_i^{(k)}}$ remains bounded for every j as $k \to \infty$ then the conditioning of $S^{(k)}(x)$ relative to ξ goes to 0.*

Proof. Let ℓ be a fixed index. We start to analyze the easiest case where the root ξ_ℓ is simple. Recall that, by Equation (1.5) we have

$$a_i^{(k)} = \frac{-p(b_i^{(k)})}{\prod_{j\neq i}(b_j^{(k)} - b_i^{(k)})}.$$

In our hypothesis, since the $b_i^{(k)}$ go to the roots ξ_i, the $a_i^{(k)}$ tends to $\frac{p(x)}{p'(x)}$ and so to 0. In particular, the only relevant term of the sum defining $\sigma^{(k)}(x)$ is the ℓ-th one since the others go to 0. We can write

$$a_\ell^{(k)} = \frac{\epsilon_\ell p'(\xi_\ell) + O(\epsilon_\ell^2)}{\prod_{j\neq \ell}(\xi_\ell - \xi_j + \epsilon_\ell - \epsilon_j)}, \qquad \epsilon_j = \xi_j - b_j^{(k)},$$

so that

$$\frac{|a_\ell^{(k)}|}{|\xi_\ell - b_\ell^{(k)}|} = \frac{|a_\ell^{(k)}|}{|\epsilon_\ell|} = \frac{p'(\epsilon_\ell) + O(\epsilon_\ell)}{\prod_{j\neq \ell}(\xi_\ell - \xi_j + \epsilon_\ell - \epsilon_j)}.$$

When all the $\epsilon_j \to 0$, the denominator of the right-hand side goes to $p'(\xi_\ell)$ so that we have $\frac{|a_\ell^{(k)}|}{|\xi_\ell - b_\ell^{(k)}|} \to 1$. It remains to understand the behavior of $(S^{(k)})'(\xi_\ell)$. Notice that, once again, all the terms of $(S^{(k)})'(\xi_\ell)$ but the ℓ-th go to zero, for the same reason of above (the a_i go to zero). It remains to study

$$\frac{|a_\ell^{(k)}|}{|\xi_\ell - b_\ell^{(k)}|^2} = |\epsilon_\ell^{-1}| \frac{|a_\ell^{(k)}|}{|\xi_\ell - b_\ell^{(k)}|} \approx \frac{1}{|\epsilon_\ell|},$$

since we have already studied the second term in the previous case. Given that the expression above is unbounded when $\epsilon_\ell \to 0$ we automatically have the thesis and the conditioning of the ℓ-th root goes to 0 as $k \to \infty$.

The same analysis can be carried out for multiple roots, but the computations are more involved. We have

$$a_i^{(k)} = \frac{\epsilon_i^m p^{(m)}(\xi_i) + O(\epsilon_i^{m+1})}{\prod_{j\neq i}(\xi_i - \xi_j + \epsilon_i - \epsilon_j)}, \qquad \epsilon_j = \xi_j - b_j.$$

and we can write, recalling that for $i = \ell, \dots, \ell + m - 1$ all the ξ_i are equal to ξ,

$$\begin{aligned}\prod_{j\neq i}(\xi_i - \xi_j + \epsilon_i - \epsilon_j) &= \prod_{j=\ell}^{\ell+m-1}(\epsilon_i - \epsilon_j) \cdot \prod_{\substack{j<\ell \text{ or } j\geqslant \ell+m\\ j\neq i}}(\xi - \xi_j + \epsilon_i - \epsilon_j)\\ &= \prod_{j=\ell}^{\ell+m-1}(\epsilon_i - \epsilon_j)\cdot(p^{(m)}(\xi) + O(\epsilon)).\end{aligned}$$

Applying a similar argument as before we obtain

$$\frac{a_i}{\xi - b_i} = \frac{\epsilon_i^{m-1}}{\prod_{j=\ell, j\neq i}^{l+m-1}(b_i^{(k)} - b_j^{(k)})} + O(\epsilon). \tag{1.7}$$

Notice that if $|b_i^{(k)} - b_j^{(k)}| \sim |\xi - b_i^{(k)}| = |\epsilon_i|$ then the above quantity goes (in modulus) to 1. Since we know that the ratio of the twos is uniformly bounded, we have that also the limit of Equation (1.7) is bounded. We can now conclude by noting that $(S^{(k)})'(\xi)$ is unbounded by following the same steps of the simple root case. This completes the proof. □

Remark 1.3.15. In the case of multiple roots, the requirement on the quantities $b_i^{(k)}$, stated in part (ii) of Theorem 1.3.14 is rather strong. Nevertheless, this is not a problem since the iterative method that we use (namely Ehrlich–Aberth) lead to approximations that have this property, and so Theorem 1.3.14 can also be applied in practice.

Now we are aware of how good choices for the nodes b_i look like: we need to choose them as good approximations to the roots of $p(x)$. Unfortunately, this happens to be exactly what we would like to compute, so this information does not seem to be useful. We show that this is not the case and that we can exploit this information to create an efficient rootfinding algorithm for polynomials. We propose the following high level scheme:

Sketch of the approximation algorithm:

(i) We first get some rough approximation to the roots. Several strategies are available for this purpose, but we rely on tropical roots that are presented in Section 1.4.

(ii) We compute a secular equation equivalent to the polynomial $p(x)$ using these approximations as nodes.

(iii) We use some functional iteration scheme to approximate the roots. We choose Ehrlich–Aberth implicitly applied on the polynomial $p(x)$ for this purpose. More details will follow in Section 1.5.

(iv) We use the computed approximations as new nodes, and we go back to 2. We continue to do this until we reach the desired accuracy.

The various steps of the algorithm deserve to be analyzed carefully. We do this in the following sections. We first introduce the concept of tropical roots in order to provide the results used to implement part (i) of the above scheme and then give a complete description of the algorithm for the approximation of polynomial roots.

Before going on, though, we shall make an important remark.

Remark 1.3.16. The steps highlighted in the sketch of the algorithm show that only the evaluation of the polynomial is necessary in order to implement the strategy. In fact, Equation (1.5) provides a formula to compute the weights of the secular equation that only uses evaluation of $p(x)$ at the points b_i. This makes it very easy to generalize the above procedure to general classes of polynomials as soon as a stable evaluation procedure is available.

1.4 TROPICAL ROOTS

In this section we review some of the results on *tropical roots*, a tool that can be used to give estimates on the moduli of roots of scalar polynomials and eigenvalues of matrices as well. The use of strategies based on tropical algebra for this estimation has been investigated by Bini, Gaubert, Noferini, Sharify, Tisseur, et al. in the papers [59, 19, 82] and also in the PhD thesis [85]. The results that we present are also strictly connected with the Newton polygon based analysis of Bini carried out in [12].

The main idea behind these strategies is the following: suppose that we are given a "classical" polynomial $p(x)$ that we want to solve. We construct a so-called tropical polynomial $tp(x)$ that is much easier to solve and we use the information about its roots to locate the roots of $p(x)$.

1.4.1 *Localization of the roots*

We start by giving the formal definitions of the objects that we need to study.

Definition 1.4.1. The *tropical max-plus* semiring T (tropical semiring from now on) is the set $\mathbb{R} \cup \{-\infty\}$ with the operations $\oplus$, $\otimes$ defined as

$$a \oplus b := \max(a, b), \qquad a \otimes b := a + b.$$

where the $+$ is the usual operation on the field $\mathbb{R}$ and we set $-\infty + a = \infty$.

Remark 1.4.2. The 0 and 1 elements of the ring are $-\infty$ and 0, respectively. In fact, it is very easy to check that

- $a \oplus -\infty = \max(a, -\infty) = a$.
- $a \otimes 0 = a + 0 = a$.

We can define the set of polynomials with coefficients in the (semi)ring T as the set T[x] of polynomials with coefficients in T, as usual. We have $tp(x) \in T[x]$ if

$$tp(x) = a_n \otimes x^{\otimes n} \oplus \ldots \oplus a_1 \otimes x \oplus a_0, \quad x^{\otimes i} := x \otimes \ldots \otimes x \text{ multiplied } i \text{ times.}$$

Note that expanding the definition of the operations on the ring T yields

$$tp(x) = \max_{i=0,\ldots,n} (a_i + ix)$$

so $tp(x)$ can be considered as a piecewise linear function given as the pointwise maximum of n linear functions. Taking this into account we can give the following definition

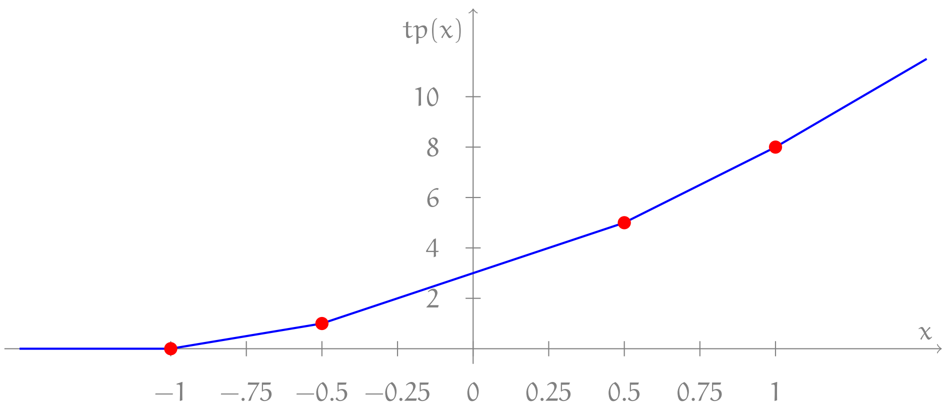


Figure 1.2: Plot of the function associated with the tropical polynomial defined by $tp(x) = x^{\otimes 7} \oplus 2x^{\otimes 6} \oplus 3x^{\otimes 4} \oplus x^{\otimes 3} \oplus 2x^{\otimes 2}$. The points mark the tropical roots of the polynomial.

Definition 1.4.3. The *tropical roots* of the tropical polynomial $tp(x)$ defined as above are the points where the maximum of the linear functions is attained at least twice. The *multiplicity* of each roots is equal to the number of linear functions attaining the maximum value minus one.

More intuitively, when looking at the plot of the function associated with the tropical polynomial the roots are the non differentiable points, i.e., the "spikes" of the graph as shown in Figure 1.2.

Remark 1.4.4. It might be questioned why these points are called tropical roots, since it is not true that $tp(r_i) = 0$ where $r_1, \dots, r_\ell$ are the tropical roots. Nevertheless, recall that since r_i are the points where the slope of the function changes, it holds that

$$tp(x) = (x \oplus r_1)^{m_1} \otimes \dots\dots\dots \otimes (x \oplus r_\ell)^{m_\ell}$$

where r_i are the tropical roots and m_i their multiplicities. This can be seen as a kind of fundamental theorem of tropical algebra that justifies the choice of the name tropical roots for these objects.

Remark 1.4.5. It is worth noting that the above definitions and comments could be given in an alternative framework by defining the *max-times tropical semiring* as $\mathbb{R}_+$, the set of positive real numbers and defining the operations as

$$a \oplus b := \max(a, b), \qquad a \otimes b := a \cdot b.$$

The max-plus and max-times semirings are isomomorphic through the exponential map. In fact, it is easy to see that the function $x \mapsto e^x$ maps $\mathbb{R} \cup \{-\infty\}$ into $\mathbb{R}_+$ and that the operations defined above are preserved, since e^x is an increasing function (thus preserving the maximum) and $e^{a+b} = e^a e^b$.

1.4.2 *Computing the tropical roots*

We have given the formal definition of the tropical roots, but as of now we do not have a method for computing them. Fortunately this task is much easier than computing the roots of a scalar polynomial. In fact, we can compute the tropical roots of a degree n tropical polynomial in $O(n)$ flops. To achieve this result, we need to introduce the Newton polygon.

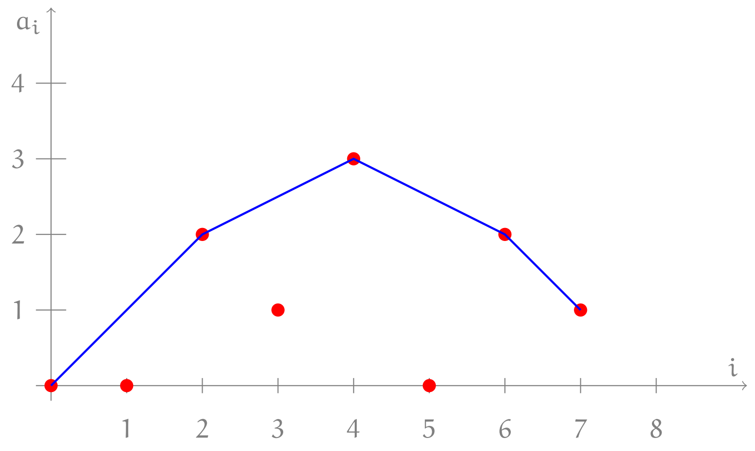


Figure 1.3: Newton polygon of the tropical polynomial $tp(x) = x^{\otimes 7} \oplus 2x^{\otimes 6} \oplus 3x^{\otimes 4} \oplus x^{\otimes 3} \oplus 2x^{\otimes 2}$. The boundary of the upper convex hull is marked with the blue line. The opposite of the slopes of the linear pieces are the tropical roots and the distance on the x axis between the relative endpoints is their multiplicity.

Definition 1.4.6 (Newton polygon)**.** The *Newton polygon* associated with a tropical polynomial $tp(x)$ is the upper convex hull of the points (i, a_i) where a_i are the coefficients of $tp(x)$ of degree i.

The opposites of the slopes of the linear pieces of the Newton polygon are the roots of the tropical polynomial, while their width correspond to their multiplicity. An example of the Newton polygon is depicted in Figure 1.3. This is the Newton polygon relative to the tropical polynomial of Figure 1.2. In this example the polynomial $tp(x) = x^{\otimes 7} \oplus 2x^{\otimes 6} \oplus 3x^{\otimes 4} \oplus x^{\otimes 3} \oplus 2x^{\otimes 2}$ is considered. It can be seen that only some points are vertices of the upper convex hull, namely $(0, 0)$, $(2, 2)$, $(4, 3)$, $(6, 2)$ and $(7, 1)$. Measuring the slopes of the linear pieces and taking the opposites yields the tropical roots $-1, -\frac{1}{2}, \frac{1}{2}, 1$. Their multiplicities are, respectively, $2, 2, 2$ and 1.

The computation of these roots can be carried out in $O(n)$ flops by using an algorithm for the computation of the convex hull of a planar set like the Graham scan of [63]. This algorithm generally runs in $O(n \log n)$ flops but since in this case we have the points already sorted by the x coordinate we can obtain the result in only $O(n)$ steps.

The main point of interest, for us, is to derive information on the roots of scalar polynomials from the roots of the associated tropical polynomials. More precisely, we first present a criterion for root localization based on an application of the Rouché theorem and then we show how the use of tropical roots can make the application of the criterion feasible in practice.

We consider, until the end of the chapter, a polynomial $p(x) = \sum_{i=0}^{n} p_i x^i$ with roots $\xi_1, \dots, \xi_n$ counted with multiplicities (so it might happen that $\xi_i = \xi_j$) and

$$tp(x) := \bigoplus_{i=0}^{n} \log(|p_i|) x^{\otimes i}$$

the associated tropical polynomial obtained with the convention that $\log(0) = -\infty$.

Here is the main result that we make use of, that can be found in [68, Theorem 4.10b].

Theorem 1.4.7 (Rouché). *Let $f(z)$ and $g(z)$ be two holomorphic functions defined on an open connected set $A \subseteq \mathbb{C}$. Let Γ be a simple closed path contained in A. If $|f(z)| > |g(z)|$ on Γ then f and $f + g$ have the same number of zeros inside Γ.*

The above theorem can be directly applied to obtain bounds on the location of the roots of a scalar polynomial. In fact, we can note that, for every $k = 0, \dots, n$, the polynomial

$$s_k(x) = |p_0| + \dots + |p_{k-1}|x^k - |p_k|x^k + |p_{k+1}|x^{k+1} + \dots + x^n|p_n|$$

has either two or one change of signs in its (real) coefficients. The latter happens only when $k \in \{0, n\}$. This means that for $k = 0, n$ the polynomial can have only one positive root, while in the other cases it can have at most 2 positive real roots. These facts can be used to obtain the following result, proved in [12].

Lemma 1.4.8. *Let $s_k(x)$ be the polynomials defined above. We have the following*

(i) For $k = 0$ or $k = n$ the polynomials $s_0(x)$ and $s_n(x)$ have exactly one positive real root. We call these two roots r_0^u and r_n^l, respectively. Moreover, we set $r_0^l = 0$ and $r_n^u = \infty$.

(ii) For $0 < k < n$ the polynomial $s_k(x)$ has either two or no positive real roots. If it has two roots we call them $r_k^l < r_k^u$.

(iii) For every k such that the polynomial $s_k(x)$ has at least one positive real root the annulus $\{r_k^l < |z| < r_k^u\}$ contains no roots of $p(x)$. Moreover, the ball $B(0, r_k^l)$ contains exactly k roots of $p(x)$.

This directly leads to the following consequence:

Corollary 1.4.9. *Let $r_{k_i}^l$ and $r_{k_i}^u$ be the roots of the polynomials $s_{k_i}(x)$ for the values of k_i where they have at least one positive real root. We set $r_0^l = 0$ and $r_n^u = \infty$. Then the annuli*

$$\mathcal{A}_{k_i} := \{r_{k_i}^u \leqslant |z| \leqslant r_{k_{i+1}}^l\} \subseteq \mathbb{C}$$

contain exactly $k_{i+1} - k_i$ roots of $p(x)$.

The following theorem, whose proof can be found in [12, 85], shows the connection between tropical roots and Corollary 1.4.9. Notice that the proof in [12] does not rely on the formalism of tropical roots, since they had not been yet introduced at the time.

Theorem 1.4.10. *The tropical roots $r_1, \dots, r_\ell$ of the tropical polynomial $\mathrm{tp}(x)$ associated with $p(x)$ are such that the circles centered in 0 and of radius e^{r_i} are contained in the annuli $\mathcal{A}_{k_i}$ for some k_i. Moreover, the sum of the multiplicities of the tropical roots contained in each annulus is equal to the number of roots of $p(x)$ in it.*

The main consequence of Theorem 1.4.10 is that, instead of computing explicitly the values $r_{k_i}^l$ and $r_{k_i}^u$, that would require the solution of a polynomial, we can obtain some radii that are guaranteed to be contained inside the interesting annuli $\mathcal{A}_{k_i}$. We can then use the multiplicities of the tropical roots to determine how many roots are contained in those annuli. An example of this kind of approximations is reported in Figure 1.4.

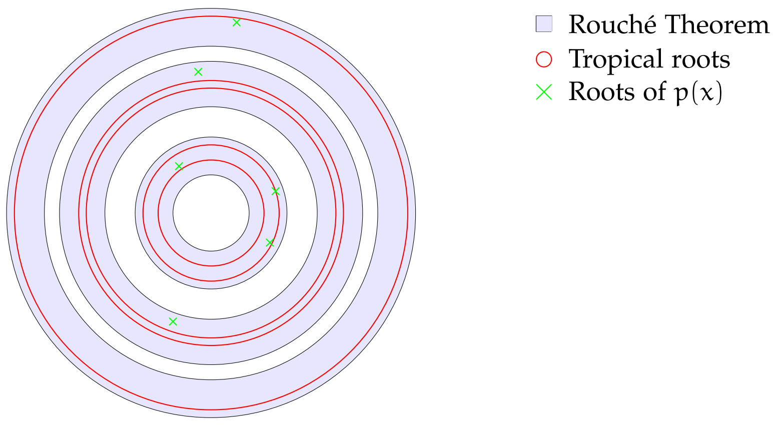


Figure 1.4: An example of the kind of bounds obtained by the application of Corollary 1.4.9 and by the computation of the tropical roots of the associated tropical polynomial.

Remark 1.4.11. Notice that we have considered as radii the values e^{r_i} where r_i are the tropical roots of $tp(x)$, that has been defined as max-plus tropical polynomial by taking the logarithm of the moduli of the original polynomial $p(x)$. This shows that we could have instead considered the max-times tropical polynomial defined using the moduli of the coefficients. Its tropical roots represent the estimates for the moduli of the roots that we are looking for. The two approaches are equivalent, but we have to shift to max-plus polynomials in any case in order to use the Newton polygon.

The tropical roots provide the cheap and effective strategy that we need for choosing the initial approximations in the outlined strategy for the algorithm sketched on page 15.

1.5 A MULTIPRECISION ALGORITHM

The aim of this section is to combine all the pieces introduced in the previous ones in order to build a fast and effective algorithm for the computation of the roots of scalar polynomials. It is our interest to build an algorithm with the following features:

- The algorithm should be a "black-box", so that we can adapt it to any kind of polynomial (and not just the ones defined through their coefficients in the monomial basis) without much effort, ideally by just providing the necessary tools to evaluate it and its derivative at a point.
- It should approximate the roots of the polynomial to any desired precision and guarantee the digits that it computes.
- The algorithm should exploit the information obtained on the approximated components to speed up the convergence on the other ones. We want to achieve this with a kind of implicit deflation, in order to avoid introducing numerical errors often present in explicit deflation strategies (see [15] for a motivation around this choice).

The algorithm developed in this section is called `secsolve` (from secular equation `solver`). It is currently part of the `MPSolve` suite, that was developed by Bini and Fiorentino in the

90s [15], and has been recently updated with the implementation of the new algorithm. The original version of `MPSolve` has been one of the fastest rootfinders available for a long time. Recently Fortune proposed the `eigensolve` algorithm [53] that is faster than `MPSolve` on some sets of polynomials. We show here that our new approach is able to be faster than both the original `MPSolve` and `eigensolve` on most test cases. This claim is validated by the numerical experiments of Section 1.7.

Here we elaborate the initial proposal of the algorithm of page 15. We follow these steps which are also reported in the pseudocode of Algorithm 2.

(i) Choose a set of starting points $x_1, \dots, x_n$ that are likely to be good approximations for the roots. Several strategies can be used and we rely on the tropical roots presented in Section 1.4.

(ii) Compute the weights of a secular equation $S(x) = 0$ equivalent to the polynomial equation $p(x) = 0$. We choose $b_i = x_i$ as nodes.

(iii) Apply the Ehrlich–Aberth iteration on $p(x)$ implicitly represented as $-S(x)\Pi(x)$. This allows to exploit the good conditioning properties of $S(x)$ when the b_i are good approximations of the roots. We continue to iterate until the approximations enter the root neighborhood for the secular equation relative to a small multiple of the current unit roundoff (chosen in a way that allows to guarantee that computations that we carry out in floating point).

(iv) Check if the approximations obtained are accurate enough using the bounds given by Gerschgorin's discs and Newton inclusions. If this is the case, we exit, otherwise we go back to (ii).

Algorithm 2 `secsolve` algorithm for the approximation of the roots of a polynomial. Here $|\cdot|$, $\log(\cdot)$ and $<$ are applied component-wise to vectors. A vector is said to be true if all the components are true.

```
1:  function SECSOLVE(p, threshold)
2:      $tp \leftarrow \textsc{TropicalPolynomial}(\log(|p|))$
3:      $[r, m] \leftarrow \textsc{TropicalRoots}(tp)$
4:      $x \leftarrow \textsc{ComputeApproximations}(r, m)$
5:      while not approximated do
6:          $b \leftarrow x$
7:          $S \leftarrow \textsc{ComputeWeights}(p, b)$
8:          $x \leftarrow \textsc{ImplicitEhrlichAberth}(S, x)$
9:          $r \leftarrow \textsc{ComputeInclusionRadii}(p, x)$
10:         $approximated \leftarrow r < |x| \cdot threshold$
11:     end while
12: end function
```

The roles of the functions in Algorithm 2 are rather clear by their name. Nevertheless, we do not have discussed yet how to implicitly apply the Ehrlich–Aberth iteration and how to compute a set of inclusion radii for the current approximations. These two topics are described in the next sections.

1.5.1 *Applying the implicit Ehrlich–Aberth iteration*

In this section we show how it is possible to apply the Ehrlich–Aberth iteration to a polynomial knowing only a secular function equivalent to it. Recall that, in view of Definition 1.3.2, a secular function $S(x)$ is said to be *equivalent* to a polynomial $p(x)$ if and only if

$$-S(x)\Pi(x) = p(x), \qquad S(x) = \sum_{i=1}^{n} \frac{a_i}{x - b_i} - 1, \qquad \Pi(x) = \prod_{i=1}^{n}(x - b_i).$$

Recall that, in view of Remark 1.1.2, in order to evaluate the Ehrlich–Aberth correction at a certain point $x \in \mathbb{C}$ it is sufficient to evaluate the Newton correction $\frac{p(x)}{p'(x)}$.

We have

$$\begin{cases} p(x) = -S(x)\Pi(x) \\ p'(x) = -S'(x)\Pi(x) - S(x)\Pi'(x) \end{cases}.$$

Since $\Pi'(x) = \Pi(x)\sum_{i=1}^{n} \frac{1}{x-b_i}$ we can conclude that

$$\frac{p(x)}{p'(x)} = \frac{S(x)}{S'(x) + \sum_{i=1}^{n} \frac{1}{x-b_i}} = \frac{\sum_{i=1}^{n} \frac{a_i}{x-b_i} - 1}{\sum_{i=1}^{n} \frac{1}{x-b_i} - \sum_{i=1}^{n} \frac{a_i}{(x-b_i)^2}}.$$

The above can be evaluated with $4n+2$ additions and subtractions, $n+1$ inversions and $2n+1$ multiplications.

Remark 1.5.1. We stress that the evaluation of $\frac{p(x)}{p'(x)}$ does not require to know anything about $p(x)$, since it relies on the coefficients of $S(x)$. In particular, since $S(x)$ is constructed starting only from evaluations of $p(x)$ (see Formula 1.5) being able to explicitly evaluate $p'(x)$ is not relevant at all in our setting. This could be an advantage in some cases where we have a stable procedure for the evaluation but not for the computation of the Newton correction.

1.5.2 *Partial regeneration of secular equations*

In some cases it might happen that we have already approximated some of the roots of the polynomial to the desired precision, while some others are missing. Assume, for simplicity, that the first ℓ components have been approximated while the others are not. In those cases we would like to have that $b_1, \dots, b_\ell$ for the new secular function are equal to the old ones.

Recalling, by Equation (1.5), that

$$a_i = \frac{p(b_i)}{\prod_{j\neq i}(b_i - b_j)}, \qquad i = 1, \dots, n$$

we can write an "improved" formula for a_i when $i \leqslant \ell$. We call $\hat{a}_i$ the weights of the updated secular function and a_i the old ones. The same notation is used for the nodes b_i. Recalling that $b_i = \hat{b}_i$ when $i \leqslant \ell$ we have

$$\hat{a}_i = \frac{p(\hat{b}_i)}{\prod_{j\neq i}(\hat{b}_i - \hat{b}_j)} = \frac{p(b_i)}{\prod_{\substack{j\leqslant \ell \\ j\neq i}}(b_i - b_j)\cdot\prod_{\substack{j>\ell \\ j\neq i}}(b_i - \hat{b}_j)} = a_i \prod_{\substack{j>\ell \\ j\neq i}} \frac{b_i - b_j}{b_i - \hat{b}_j}, \qquad i \leqslant \ell.$$

This implies that the new weights $\hat{a}_i$ can be computed from the old ones by a simple update formula that costs $O(n-\ell)$ flops, that is, it is equal to the number of changed approximations. Since in general the computation of each weight costs $O(n)$ flops plus the evaluation of the polynomial (that is usually $O(n)$), we have that the regeneration cost for the secular equation goes down from $O(n^2)$ to $O(n \cdot (n-\ell))$, because we have to compute $(n-\ell)$ weights at a cost $O(n)$ and ℓ at a cost $O(n-\ell)$.

1.5.3 *Stopping criterion and inclusion bounds*

In order to construct a working algorithm for the approximation of polynomials roots we still need to introduce some other tools.

First of all, we need to devise an efficient stopping criterion that allows to obtain as accurate as possible approximations for each secular equation. This way we can rely on regeneration only when no further improvement is possible, without wasting computational time.

Recalling the secular and root neighborhoods defined in Definition 1.3.7 and 1.3.8 we have that if the approximations $x_1, \dots, x_n$ are included in $RN_\epsilon(S(x))$ for $\epsilon \sim u$ then each of the approximation is a root of a slightly perturbed secular equation. In particular, from our numerical viewpoint, we are not able to distinguish these approximations from the correct solutions of the secular equation. When this situation is reached, no further improvement can be obtained iterating in floating point with a unit roundoff equal to u. Exploiting Lemma 1.3.13 we use this condition as a stopping condition. More precisely, we implement the following stopping criterion: if an approximation x is included in the floating point root neighborhood $\tilde{RN}_{u(1+\kappa_n)}(S(x))$ then we stop the iteration on that component, since we have $x \in \mathbb{RN}_{u(1+2\kappa_n)}(S(x))$ (ignoring second order terms) and so it is a root of a slightly perturbed secular equation. If this is not the case, $x \notin \tilde{RN}_u(S(x))$. Proposition 1.3.12 guarantees that, in this case, the evaluation of $S(x)$ still contains some valuable information (the relative error is smaller than 1) and so we can further improve the approximation.

When all the approximations enter the root neighborhood the floating point iteration cannot improve the approximations and so we need some new information in order to proceed. For this reason we employ multiprecision floating point arithmetic in order to compute a new equivalent secular equation according to Definition 1.3.2 and Lemma 1.3.1 by setting $b_i := x_i$. Since the current approximations have improved from the old ones, we have that the conditioning of the new secular equation will be smaller. Thus, the components of the root neighborhoods relative to each root (that by definition can be estimated as $u \cdot \kappa_i(S(x))$ where κ_i is the conditioning of the i-th root) will be shrinked versions of the original ones. We continue to iterate this process until the approximations obtained are satisfying.

A natural question is what is the desirable degree of approximation that we want to reach. When can we stop? The answer can be given by the following theorem, which is a rephrased version of [86, Theorem 2.4].

Theorem 1.5.2. *Let α_i be a root of the polynomial $p(x) = \prod_{i=1}^n (x - \alpha_i)$ and x a point in the complex plane. If*

$$|\alpha_i - x| \leqslant \frac{1}{3(n-1)} |\alpha_j - x|, \qquad j \neq i,$$

then the Newton method starting from x is quadratically convergent to α_i.

In view of this result we want to find a practical way to detect when we encounter such a situation for some index i. We show two possible methods to obtain this result.

The first result uses Gerschgorin's theorem and the construction of the secular linearization of Corollary 1.3.4. Before stating it, we report Gerschgorin's theorem for completeness.

Theorem 1.5.3 (Gerschgorin)**.** *Let A be an $n \times n$ complex matrix. We denote its elements as a_{ij}, and we define the discs B_i as*

$$B_i := \left\{ z \in \mathbb{C} \mid |z - a_{ii}| \leqslant \sum_{j \neq i} |a_{ij}| \right\}, \qquad i = 1, \dots, n.$$

Then the union of the discs B_i contains the eigenvalues of A and every connected component of this union made of k discs contain exactly k eigenvalues counted with multiplicity.

Theorem 1.5.4. *Let $p(x)$ be a scalar monic polynomial, and x_i for $i = 1, \dots, n$ a set of n pairwise distinct approximations. Then the roots of the polynomial are included in the union of the discs B_i defined by*

$$B_i = \left\{ z \in \mathbb{C} \mid |z - x_i| \leqslant n \left| \frac{p(x_i)}{\prod_{j \neq i}(x_i - x_j)} \right| \right\}.$$

Moreover, each connected component of the union contain a number of roots equal to the number of discs included in the connected component (counted with multiplicity).

Proof. The result is obtained by applying Gerschgorin's theorem to the linearization for $p(x)$ obtained from Corollary 1.3.4 with $b_i := x_i$. □

We report another result that can be used to bound the roots of the polynomial based on a set of approximations. The following can be found on [68].

Theorem 1.5.5. *Let w_i be the i-th coefficient of the Taylor expansion of the polynomial $p(x)$ in $\bar{x}$, i.e. the i-th coefficient of the shifted polynomial $p(x - \bar{x})$. If we define*

$$\beta(\bar{x}) = \min_{1 \leqslant m \leqslant n} \left[\binom{n}{m} \left| \frac{w_0}{w_m} \right| \right]^{1/m}$$

then the ball of center $\bar{x}$ and radius $\beta(\bar{x})$ contains at least one root of $p(x)$.

The above theorem induces a family of inclusion results by just changing the value of m. For example, $m = 1$ yields the following

Corollary 1.5.6. *Let $p(x)$ be a polynomial of degree n. Then the set $\left\{ z \mid |z - \bar{x}| \leqslant n \cdot \left| \frac{p(\bar{x})}{p'(\bar{x})} \right| \right\}$ always contains a root of the polynomial.*

Proof. Apply Theorem 1.5.5 with $m = 1$. □

We are particularly interested in the case when the approximations $x_1, \dots, x_n$ are such that the discs induced by Corollary 1.5.6 or by Theorem 1.5.4 allow to prove the hypothesis of Theorem 1.5.2. In particular, this happens when the discs induced by any of the two theorems with radius enlarged by a factor of $3n$ have empty intersection with the others. We call this condition the *Newton isolation* condition. Whenever this happens on an approximation x_i we are sure that the Newton method is quadratically convergent starting from that approximation, and so we can rely on it if other digits of the approximated root are needed.

1.6 SOLVING MANDELBROT POLYNOMIALS

In this section we show an example of how it is possible to adapt the algorithm developed in the previous section in order to approximate the roots of a polynomial belonging to a particular class.

In particular, we consider the class of Mandelbrot polynomials, defined by recurrence on $d \in \mathbb{N}$ by the following relations:

$$\begin{cases} p_0(z) = 1 \\ p_{d+1}(z) = zp_d^2 + 1 \end{cases} . \tag{1.8}$$

The Mandelbrot polynomial $p_d(z)$ has degree $2^d - 1$ and its roots are periodic points of the Mandelbrot set of order d. A representation of the roots for $d = 10$ (so that the associated polynomial has degree 1023) are reported in Figure 1.5.

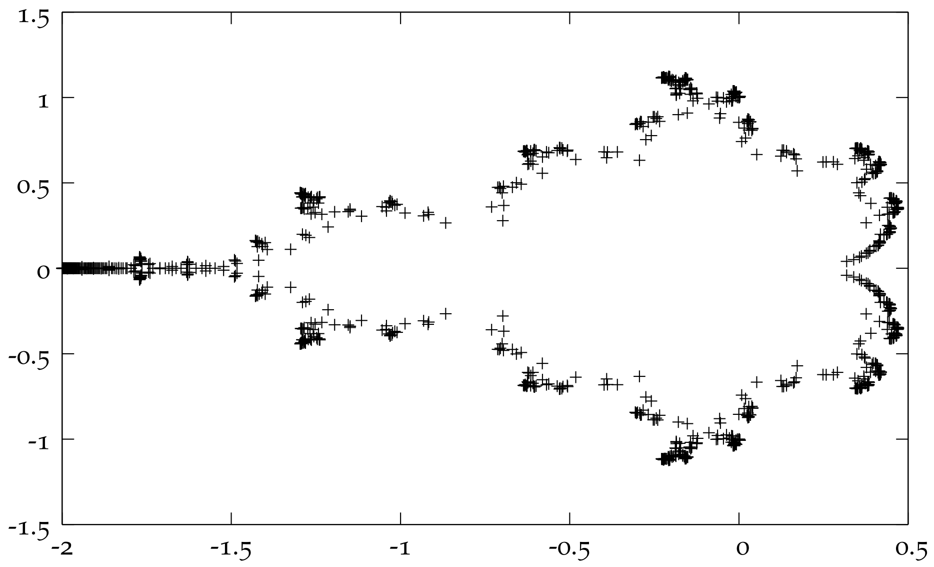


Figure 1.5: Roots of the Mandelbrot polynomial of order $d = 10$. The polynomial has degree $2^d - 1 = 1023$.

The problem of solving these polynomials has been attacked in the past by others. Mandelbrot polynomials have been used as a difficult test case by the first version of `MPSolve` (see [15]) and also in the software `eigensolve`, proposed by Steven Fortune in [53]. Recently the problem of approximating the roots of very high degree Mandelbrot polynomials has been tackled by Corless and Lawrence in [31]. Their approach is based on using sparse linearizations for these polynomials and then solving the eigenvalue problem on a large cluster. They have been able to find the roots of $p_{20}(x)$ in about the equivalent of 31 years of sequential computational time.

Another recent work by Schleicher and Stoll analyzes possible choices for starting points so that the Newton's method converges to all the roots of Mandelbrot polynomials. A preprint of their work is available in [84] at the time of writing.

The method presented here is faster than any other implementation available (at least to our knowledge), as reported in the next Section 1.6.2, where we have compared our algorithm with the software that is available for testing.

1.6.1 *Evaluating the Mandelbrot polynomials*

A possible approach to compute the roots of the polynomial is to explicitly compute the coefficients of the polynomial in the monomial basis and then use the general solver to obtain approximations to the roots. While this is possible, and in fact these kinds of polynomials are included in the test suite for the `MPSolve` package, the integer coefficients of the Mandelbrot polynomials become very large as the degree increases, and so it is not very practical if one wants to reach very high degrees.

For this reason we consider the idea of an implicit solver that only reads the integer d and solves the polynomial by just knowing the recurrence relation of its definition.

Remark 1.6.1. It can be noticed that the recurrence relation is enough to evaluate the polynomial at a point in the complex plane. In fact, the use of the recurrence relation allows to evaluate the polynomial with only $d \approx \log n$ operations that is much cheaper than the Horner method. Moreover, the evaluation through the recurrence relation turns out to be much more stable since the (very large) coefficients would need to be truncated in order to be represented in floating point before executing the Horner scheme. Given the bad conditioning of the Mandelbrot polynomials this would lead to very large errors in the evaluation. This is actually the reason that makes them a very good test-cases for general polynomial solvers.

We propose to use the procedure reported in Algorithm 3 in order to evaluate the polynomial $p_d(x)$ at a certain point in the complex plane. Notice that the function is also used to retrieve a bound on the evaluation error. To do this we use the following result.

Lemma 1.6.2. *Let $p_d(x)$ be the Mandelbrot polynomial of order d that has degree $2^d - 1$. If $p_d(x)$ is evaluated at a point $x \in \mathbb{C}$ by means of the recurrent relations of Equation (1.8) then*

$$\mathrm{fl}(p_d(x)) = \left(x\, \mathrm{fl}(p_{d-1}(x))^2 \cdot (1+\epsilon_1) + 1 \right)(1+\epsilon_2)$$

with $|\epsilon_1| \leqslant 1 + \frac{4\sqrt{2}}{1-2u}$ and $|\epsilon_2| \leqslant u$ up to second order factors in u. In particular, the floating point error $e_d := \mathrm{fl}(p_d(x)) - p_d(x)$ can be recursively bounded using the formulas

$$\begin{cases} |e_0| = 0 \\ |e_{d+1}| \leqslant \frac{(|\epsilon_1||x||e_{d-1}|+1)(1+|\epsilon_2|)+|\epsilon_2|\,\mathrm{fl}(p_d(x))}{1-|\epsilon_2|} \end{cases}$$

Proof. According to [69] we know that for every complex number z and w

$$\mathrm{fl}(z+w) = (z+w)(1+\epsilon_+), \qquad |\epsilon_+| \leqslant u$$
$$\mathrm{fl}(z \cdot w) = zw(1+\epsilon_*), \qquad |\epsilon_*| \leqslant \frac{2\sqrt{2}}{1-2u}$$

where u is the current unit roundoff. Applying these bounds to the floating point multiplication and addition in the recursive relation yields the first statement of the thesis.

The recursive formula to bound the error can be obtained by noting that

$$\begin{aligned} e_d(x) &= |\,\mathrm{fl}(p_d(x)) - p_d(x)| \\ &= |(x\,\mathrm{fl}(p_{d-1}(x))^2 \cdot (1+\epsilon_1) + 1)(1+\epsilon_2) - p_d(x)| \\ &= |(\epsilon_1 x e_{d-1}(x) + 1)(1+\epsilon_2) + \epsilon_2 p_d(x)| \\ &= |(\epsilon_1 x e_{d-1}(x) + 1)(1+\epsilon_2) + \epsilon_2\,\mathrm{fl}(p_d(x)) + \epsilon_2 e_d(x)|. \end{aligned}$$

Taking the last term to the left-hand side and dividing by $(1-\epsilon_2)$ gives the thesis. □

Algorithm 3 Function that evaluates the Mandelbrot polynomial $p_d(z)$ at a certain point in the complex plane. The function takes as input the positive integer d and the point $x \in \mathbb{C}$ and returns a tuple containing the result of the evaluation and a bound for the floating point error.

```
function EvaluateMandelbrot(d, x)
    $u \leftarrow$ UnitRoundoff()
    $v \leftarrow \frac{2*\sqrt{2}}{1-2*u}$
    if d == 0 then
        return $(1, 0)$
    else
        $(p, e) \leftarrow$ EvaluateMandelbrot$(d-1, x)$
        $p \leftarrow x * p^2 + 1$
        $e \leftarrow ((u * |x| * e + 1) * (1 + v) + v * p)/(1 - v)$
        return $(p, e)$
    end if
end function
```

The evaluation of the polynomial, as seen in Section 1.5, is sufficient in order to implement the approximation algorithm based on `MPSolve`. Nevertheless, in order to make the algorithm even faster, we can also implement the evaluation of the Newton's correction $p_d(x)/p'_d(x)$. This can be used when Newton isolation is guaranteed in order to refine the approximations using Newton's iterations.

We notice that a recurrence relation can be used also to evaluate the derivative of $p_d(x)$, in fact we have

$$\begin{cases} p'_0(x) = 0 \\ p'_{d+1}(x) = p_d(x)(2p'_d(x) + p_d(x)) \end{cases} \tag{1.9}$$

This relation admits an error analysis very similar to the one given for the evaluation of $p_d(x)$, and allows to evaluate the Newton correction in $O(d)$ floating point operations. Algorithm 3 can be easily modified to compute the Newton correction together with the evaluation of $p_d(x)$ at a point.

These algorithms have been implemented in the `MPSolve` package which provides an executable file called `mandelbrot-solver` that allows to efficiently approximate the roots of Mandelbrot polynomials.

The above approach has been further refined using the following remark.

Remark 1.6.3. The roots of the Mandelbrot polynomials are placed on the Mandelbrot fractal. For this reasons, the roots of the polynomial $p_d(x)$ can be used as good starting approximations for the roots of the polynomial $p_{d+1}(x)$.

Degree	Standard `MPSolve`	`mandelbrot-solver`
511	0.87	0.18
1023	5.27	0.57
2047	38.82	1.56
4095	319.57	4.45
8191	2936.32	13.85
16383	*	53.96
32767	*	193.35
65535	*	824.39

Table 1.1: Timings for the approximation of the roots of the Mandelbrot polynomials for degree between 511 and 65535. The timings of the standard `MPSolve` package and of the custom `mandelbrot-solver` have been reported. The timings are expressed in seconds.

Based on the above remark, we chose to implement `mandelbrot-solver` in a recursive way so that to solve $p_d(x)$ it first solves $p_{d-1}(x)$ and then uses the obtained approximations as starting points by just taking two copies of each approximation and perturbing them by multiplying by $(1+\epsilon)$ and $(1-\epsilon)$, respectively, with $|\epsilon|$ chosen as a small multiple of the unit roundoff.

1.6.2 *Numerical experiments*

Here we report the results of some numerical experiments that provide evidence of the effectiveness of the algorithm. We have tested the standard version of `MPSolve` using the coefficients of the Mandelbrot polynomial computed in the monomial basis.

Here the timings reported are for the Newton isolation of the roots, that is, the algorithms stops as soon as it founds isolated inclusion discs for the approximations that guarantee quadratic convergence of the Newton method from the start. Given the bad conditioning of the roots and the fact that they are clustered on the border of the Mandelbrot set, this happens when the approximation have a large amount of digits already correct, typically very near to the number of digits representable on a modern computer.

The tests have been run with multithreading enabled, so all the cores of the system have been used in the experiments. The computer that ran the tests is a dual Xeon server with the two CPUs running at 3.33 GHz. The results are reported in Table 1.1. It is easy to see from the table that not only `mandelbrot-solver` is much faster than the standard approach of solving the polynomial expressed through its coefficients, but also has a lower asymptotic complexity. In fact, we have verified experimentally that `MPSolve` has a quadratic complexity in the degree when considering "comparable" polynomials (see [15]). Since the coefficients of Mandelbrot polynomials are growing very large at a fast pace it is soon impossible to represent them in floating point. Moreover, the conditioning of the polynomials also increases very fast. In this case our recursive approach, that provides very good approximations using the roots of the lower level polynomial, helps to accelerate the algorithm. In particular our implementation of `mandelbrot-solver` has a cost that, experimentally, is less than quadratic in the degree.

The effectiveness of the strategy allowed to reach very high degrees. The algorithm has been run on a dual Xeon server for about a month and has been able to solve the polynomial

$p_{22}(x)$ that has degree $2^{22} - 1 = 4.194.303$. Some of the images obtained by plotting the roots given by the software are available at the website hosting the code for MPSolve: http://numpi.dm.unipi.it/mpsolve/. The roots of the polynomial $p_{21}(x)$ are reported in Figure 1.6.

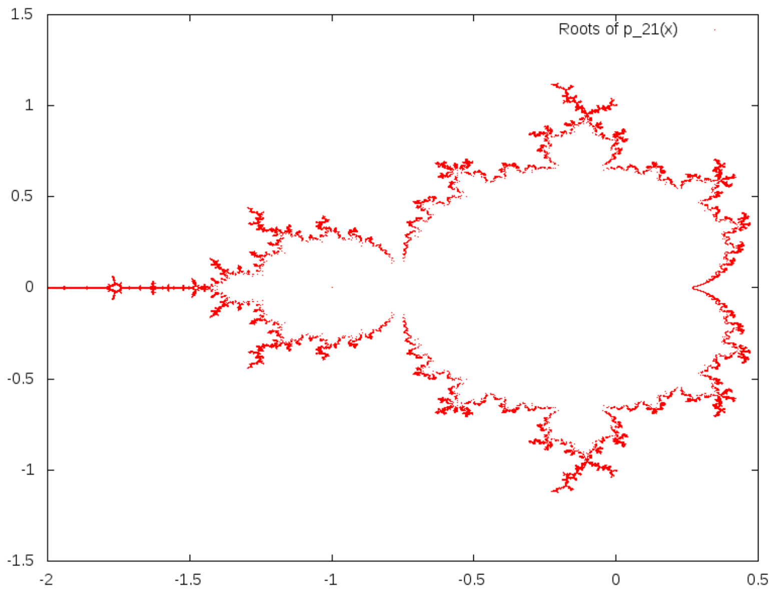


Figure 1.6: Roots of the Mandelbrot polynomial of degree $2^{21} - 1$.

1.7 NUMERICAL EXPERIMENTS

In this section we report the results of some numerical experiments run to validate the `secsolve` algorithm for the case of general polynomials where the features of each class of polynomial that is tested here are reported. The timings that we show here are slightly better than the one of [20] due to recent improvement in the code and the optimization of some procedures. Since the algorithm gives guaranteed results we do not measure the accuracy of the computed roots (that needs to have at least the required quantity of correct digits) but instead we measure the timings of different software. We compare the following implementations and algorithms:

- Our algorithm (called `secsolve`), based on the diagonal plus rank one companion form and using several representations of the polynomial in different secular basis.
- The old version of `MPSolve`, implemented by Bini and Fiorentino in [15].
- The `eigensolve` algorithm by Steven Fortune [53]. This algorithm was originally compared with `MPSolve` in the paper by Fortune and so it is directly comparable since they share the same set of test polynomials.

In [15] `MPSolve` 2.2 is benchmarked against other common software packages such as Pari, Mathematica and Maple. We do not have included them in the comparison since, as already shown in [15], they tend to be several orders of magnitude slower than `MPSolve` and eigensolve and more importantly the timings are rather unpredictable based on the degree and conditioning only. In particular for Mathematica and Maple, being closed source software, it is not possible to know which procedures are used.

Here we provide a brief comment on the different classes of polynomials that we have tested. An in-depth analysis of them can be found in [83].

NROOTS The polynomials in this family are $x^n - 1$. They are very well-conditioned and their coefficients are sparse (since only two coefficients are non-zero). This can be exploited when computing the Newton correction by using a sparse version of the Horner algorithm. MPSolve 2.2 can take fully advantage of this fact, while secsolve, being forced to work with a secular equation with with n terms different from zero, can only exploit this in the regeneration stage. This motivates the fact that MPSolve 2.2 is usually faster on these test cases.

ORTHOGONAL POLYNOMIALS We have used orthogonal polynomials from various families as test cases (here we report the cases of Hermite and Chebyshev). Here no particular structure is exploited by the methods presented. We see that in all the tested cases the secsolve algorithm is faster.

CHROMATIC POLYNOMIALS The polynomials chrma* are chromatic polynomials. They are found in applications linked to graph colorings. Their roots are usually very ill-conditioned, and in these cases the secsolve algorithm is very effective.

MANDELBROT The Mandelbrot polynomials that we have tested are the ones for which we have designed a custom method in Section 1.6. However, in this case we have used them as a difficult test case and the input has been given through their coefficients in the monomials basis.

MULTIPLE ROOTS We have tested several families of polynomials having multiple or numerically multiple roots. In particular the spiral* polynomials have roots on a spiral, which cluster around zero. The Kirinniss polynomials (named kir*) have 4 multiple roots in $1, -1, i$ and $-i$. Moreover, a simple root is also present nearby (at a distance of about $1e-3$). These polynomials have been designed to test the effectiveness of the cluster detection strategies in difficult cases where a multiple root and a numerically multiple root coexist. The Mignotte polynomials, instead, have integer coefficients and the feature of having two roots whose distance is very close to the Mahler bound (that bounds the minimum distance between two simple roots).

TRUNCATED EXPONENTIAL The polynomials named exp* are truncated Taylor series of the exponential function.

PARTITION POLYNOMIALS Partition polynomials are a difficult test case that motivated the initial development of secsolve. Their k-th coefficient is the number of different ways in which k can be obtained as a sum of positive integer numbers. The roots of these polynomials converge to a curve in the plane as the degree goes to infinity.

	MPSolve 2.2	eigensolve	secsolve	secsolve †
nroots800	**0.11**	8.37	0.18	0.08
nroots1600	**0.34**	57.44	0.59	0.15
chebyshev80	0.09	**0.08**	**0.06**	0.06
chebyshev160	0.97	0.69	**0.23**	0.12
chebyshev320	9.36	7.00	**1.54**	0.33
hermite80	**0.05**	0.06	**0.05**	0.05
hermite160	0.55	0.54	**0.35**	0.07
hermite320	5.33	5.51	**1.02**	0.23
chrma342	19.56	4.43	**2.48**	0.51
chrmad340	29.71	4.46	**4.36**	0.63
exp100	0.29	0.14	**0.07**	0.07
exp200	1.05	1.19	**0.30**	0.13
exp400	10.28	10.67	**2.23**	0.44
mand127	0.30	0.21	**0.14**	0.11
mand255	2.82	2.24	**0.65**	0.20
mand511	31.18	20.08	**4.65**	0.84
mand1023	456.91	229.60	**37.38**	5.05
mand2047	11158.72	3860.10	**320.36**	39.19
spiral20 (50 digits)	0.70	**0.07**	0.25	0.23
spiral20 (1000 digits)	0.76	**0.13**	0.37	0.24
mig1_200 (50 digits)	**0.04**	0.85	0.18	0.19
mig1_200 (1000 digits)	**0.15**	3.18	1.02	0.19
kir1_10 (100 digits)	0.23	**0.06**	0.15	0.20
kir1_10 (1000 digits)	7.23	**1.50**	1.84	1.30
kir1_10 (4000 digits)	55.30	77.38	**13.99**	8.34
partition400	0.67	4.81	**0.41**	0.14
partition800	5.49	36.90	**2.04**	0.42
partition1600	38.45	390.62	**10.62**	1.81
partition3200	213.34	7038.98	**52.86**	7.83
partition6400	1303.83	45953.33	**284.52**	38.55
partition12800	7845.62	-	**1576.16**	212.48

Table 1.2: Timings of the test runs of MPSolve 2.2, eigensolve and secsolve. The columns marked with † refer to the tests with multithreading enabled.

2 POLYNOMIAL EIGENVALUE PROBLEMS

2.1 LINEARIZATIONS AND ℓ-IFICATIONS

The aim of this chapter is to introduce the concepts of linearizations and ℓ-ifications and to present some tools that allow to construct such objects starting from a matrix polynomial $P(x)$. The ideas that we present are meant to generalize the secular basis introduced in the scalar case in Section 1.3.

Before proceeding with the introduction of these new tools it is important to briefly recap some basic definitions and notions about matrix polynomials and their linearizations and ℓ-ifications. Matrix polynomials and their linearizations are a very classical topic both in matrix theory and in numerical analysis and have been very well analyzed in classical books like [61] and [58]. ℓ-ifications, on the other hand, are much more recent and have been formally introduced and analyzed by De Terán, Dopico and Mackey in [35]. Both these concepts aim to find simpler matrix polynomials that share the same spectral properties of a given matrix polynomial $P(x) = \sum_{i=0}^{n} P_i x^i$. In this section the symbols P_i will be used to denote the coefficients of the matrix polynomial that we want to study, while the linearization (or ℓ-ification) will be denoted with $L(x)$ or $A(x)$, depending on the context.

Polynomial eigenvalue problems (in short PEPs), in their simplest version, can be formulated as finding the eigenvalues λ and the eigenvectors $v \neq 0$ such that $P(\lambda)v = 0$. This is a generalization of scalar polynomials and of standard and generalized eigenvalue problems. In fact, whenever the size of the matrix coefficients is 1 the problem reduces to a scalar rootfinding problem (the only possible eigenvector being 1, since it has to be non-zero and it is defined up to a constant factor). In the opposite case, when the matrix coefficients are bigger but the degree is 1, we have the generalized eigenvalue problem $(xA - B)v = 0$ or, when the polynomial is monic, the standard eigenvalue problem $(xI - A)v = 0$.

Remark 2.1.1. Our purpose is to design tools for the solution of the polynomial eigenvalue problem that are as efficient as possible. Notice that the faster algorithms available for polynomial rootfinding require $O(n^2)$ flops and the more efficient algorithm for standard (and generalized) eigenvalue problems require $O(m^3)$ flops. For this reason, recalling that a PEP reduces to these two problems in particular cases, we expect the complexity of these tools to be at least $O(n^2 m^3)$. Current mainstream algorithms for these problems rely on QR and QZ iterations on companion matrices, and usually have a complexity of $O(n^3 m^3)$. We investigate if it is possible to lower this complexity and so we look for operations with a cost of (at most) $O(n^2 m^3)$ flops.

2.1.1 *Basic facts about matrix polynomials*

Definition 2.1.2. We say that $P(x)$ is a *matrix polynomial* if it is a polynomial whose coefficient are matrices or, equivalently, a matrix whose coefficients are polynomials. If $P(x)$ has $m_1 \times m_2$ matrices coefficients with elements in the field $\mathbb{F}$ we write that $P(x) \in \mathbb{F}^{m_1 \times m_2}[x]$. Very often we will deal with square matrix polynomials so that $m_1 = m_2 =: m$.

This duality of the definition of matrix polynomials can often be useful to analyze them under different viewpoints. Often it happens that one perspective is lacking while the other one offers a much more accurate understanding of the underlying structure of these algebraic objects.

As an immediate example, we can give two definitions of eigenvalues of matrix polynomials. The first one is obtained by thinking of matrix polynomials as polynomials with matrix coefficients, the latter by thinking of them as matrices with polynomial coefficients.

Before giving the definitions, we need to introduce the concept of regularity.

Definition 2.1.3. Let $P(x)$ be a matrix polynomial. We say that $P(x)$ is *regular* if it is square and $\det P(x)$ is not identically zero. A matrix polynomial that is not regular (i.e., that is either rectangular or such that $\det P(x) \equiv 0$) is called *singular*.

We can then state the following definition.

Definition 2.1.4. A scalar $\lambda \in \mathbb{F}$ is said to be an *eigenvalue* of a regular matrix polynomial $P(x) \in \mathbb{F}^{n\times n}[x]$ if the matrix $P(\lambda)$ is singular, i.e., if there exists a vector $v \neq 0$ such that $P(\lambda)v = 0$. In this case such a vector v is called a *right eigenvector* of $P(x)$ relative to the eigenvalue λ. A *left eigenvector* is a vector u such that $u^t P(\lambda) = 0$.

We can observe that this definition has both an advantage and a disadvantage. The advantage is that it directly yields a definition of what an eigenvector is, and the disadvantage is that it does not make sense for rectangular and singular matrix polynomials.

Looking at the problem from a different perspective we can give an alternative definition, for which we need to recall some basic algebraic tools.

Definition 2.1.5. A matrix polynomial $E(x)$ is said to be *unimodular* if it is regular and $\det E(x)$ is a nonzero constant.

Notice that the above definition implies that the matrix polynomial $E(x)$ is invertible in the ring of matrix polynomials, i.e., there exists another matrix polynomial $F(x)$ such that $E(x)F(x) = I$. With the definition of unimodular matrices it is possible to state the following classical theorem.

Theorem 2.1.6 (Smith form). *Let $P(x) \in \mathbb{C}^{m\times n}[x]$ be a matrix polynomial, not necessarily regular. Then there exist two unimodular matrices $E(x)$ and $F(x)$ such that*

$$S(x) = E(x)P(x)F(x) = \mathrm{diag}(d_1(x), \dots, d_r(x), 0, \dots, 0)$$

is a diagonal matrix polynomial such that $d_i(x) \mid d_{i+1}(x)$ for every $i = 1, \dots, r-1$. The number r is the rank *of the matrix polynomial $P(x)$ and the Smith form is unique up to multiplication of the polynomials by invertible factors.*

Remark 2.1.7. The Smith form provides an alternative definition of regularity. In fact, it is clear that a polynomial is regular if it is square and of maximum rank.

Moreover, the Smith form of a matrix polynomial allows to state the following definition. Here and hereafter we assume that the field is algebraically closed. If this is not the case most definitions are still valid even if the polynomials could have other non-trivial factors different from $x - \lambda$.

Definition 2.1.8. Let $P(x)$ be a matrix polynomial, and $S(x)$ its Smith form. Then the roots of the diagonal elements of $S(x)$ are called the *eigenvalues* of $P(x)$. Moreover, whenever λ is an eigenvalue of $P(x)$, the diagonal elements of $S(x)$ can be factored as $d_i(x) = (x-\lambda)^{\alpha_i} r_i(x)$ with $r_i(\lambda) \neq 0$. The increasing sequence $(\alpha_1, \dots, \alpha_r)$ is called the *partial multiplicity sequence* of P at λ. The scalar polynomials $d_i(x)$ are called the *invariant factors* of $P(x)$, while the factors $(x-\lambda)$ are called *elementary divisors*.

2.1.2 *Linearizations*

The main topic of this section is the concept of linearizations and ℓ-ifications. They are tools aimed at finding equivalent representation of a matrix polynomial that are easier to solve.

A natural question that arises in this context is what we really mean by equivalence. Intuitively, we would like to preserve the spectral structure of the matrix polynomial but, as we will find out, the concept can be rather vague and some care need to be taken when giving the definitions. In fact, while it is clear that we want to preserve the eigenvalues, several other questions might arise, some of them being quite subtle:

- Shall we maintain only the eigenvalues, or also the eigenvectors? If the latter are required, in which sense they should be preserved? More precisely, which of their features do we want to maintain?
- Do we need to care about infinity eigenvalues (which will be defined precisely in this section), and do we also have to preserve multiplicities and/or Jordan chains?

A very in-depth analysis of what is meant by *equivalence* of matrix polynomials is carried out in [35]. We mostly refer to this paper regarding notation and definition of equivalences. Here we report, in a very brief way, the basic concepts that will be useful in the following, so the reader can make sure to master all the tools needed for a complete understanding of this section.

Definition 2.1.9 (Strict equivalence)**.** A matrix polynomial $P(x)$ is said to be *strictly equivalent* to another matrix polynomial $Q(x)$ if there exist two invertible constant matrices E and F such that $EP(x)F = Q(x)$. In this case we write that $P(x) \cong Q(x)$.

Definition 2.1.10 (Unimodular equivalence)**.** A matrix polynomial $P(x)$ is said to be *unimodularly equivalent* to $Q(x)$ if there exist two unimodular matrix polynomials $E(x)$ and $F(x)$ such that $E(x)P(x)F(x) = Q(x)$. In this case we write that $P(x) \sim Q(x)$.

Definition 2.1.11 (Extended unimodular equivalence)**.** A matrix polynomial $P(x)$ is said to be *extended unimodularly equivalent* to another matrix polynomial $Q(x)$ if there exist two unimodular matrices $E(x)$ and $F(x)$ and two positive integers r and s such that

$$E(x)\operatorname{diag}(I_r, P(x))F(x) = \operatorname{diag}(I_s, Q(x)).$$

In this case we write $P(x) \smile Q(x)$.

For the next definition we need to define the reversal of a matrix polynomial $P(x)$.

Definition 2.1.12. The *reversal* of the polynomial $P(x)$, denoted by $P^{\#}(x)$ is the polynomial with the coefficients in reversed order, i.e.,

$$P^{\#}(x) = x^{\deg P(x)} P(x^{-1}).$$

The definition of the reversal of a matrix polynomial allows to introduce infinite eigenvalues.

Definition 2.1.13. If $P(x)$ is a matrix polynomial we say that ∞ is one of its eigenvalues if 0 is an eigenvalue of $P^{\#}(x)$. Moreover, its eigenvectors relative to the infinite eigenvalue are defined as the eigenvectors relative to 0 of $P^{\#}(x)$.

Definition 2.1.14 (Spectral equivalence)**.** Two matrix polynomials $P(x)$ and $Q(x)$ are said to be *spectrally equivalent* if $P(x) \smile Q(x)$ and $P^{\#}(x) \smile Q^{\#}(x)$. In this case we write $P(x) \asymp Q(x)$.

The definitions of these equivalence relations have some immediate consequences on the properties that are preserved. First of all, it is worth noting that the following implications are true:

- If $P(x) \cong Q(x)$, then $P(x) \sim Q(x)$.
- If $P(x) \cong Q(x)$, then $P(x) \asymp Q(x)$.
- If $P(x) \asymp Q(x)$ then $P(x) \smile Q(x)$.
- If $P(x) \sim Q(x)$ then $P(x) \smile Q(x)$.

In particular, the relations $\sim$, $\smile$ are weaker than $\cong$ and both $\asymp$ and $\sim$, respectively, so any property that we can prove in the weaker case also holds in the stronger one. These implications can also be represented in the graph of Figure 2.1, where there exists a path connecting A to B if and only if $A \implies B$:

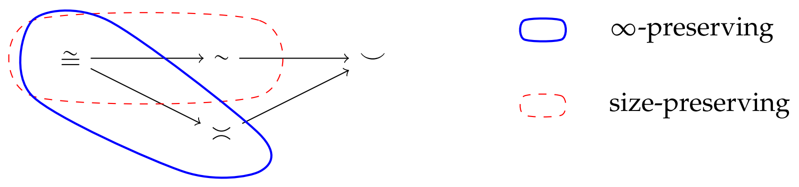


Figure 2.1: Implications between the different equivalence relations. A path connects A to B if and only if $A \implies B$. The relations are included in appropriate sets if they preserve the size of the matrix polynomial or the infinite eigenstructure.

Moreover, we have highlighted the equivalence relations that preserve the size of the matrix polynomial and the ones that don't change the infinite eigenstructure.

As we will see, size-preserving equivalence relations are too strict for our purposes, since our main aim is to preserve the eigenstructure but to lower the degree, and this is impossible without increasing the dimensions. In fact, this can be easily seen in the generic case: if the leading coefficient of a matrix polynomial $P(x)$ of size n and degree d is invertible then $\det P(x)$ has degree nd and so the matrix polynomial has nd different eigenvalues (counted with multiplicity). There is no hope of finding a matrix polynomial of the same size but with lower degree and the same number of eigenvalues.

We can turn our attention to the relations $\smile$ and $\asymp$. We have the following

Theorem 2.1.15. *Let* $P(x) \smile Q(x)$ *and, in particular, suppose that*

$$\mathrm{diag}(I_r, P(x)) = E(x)\,\mathrm{diag}(I_s, Q(x))F(x).$$

Then the following statements hold:

(i) If λ *is a finite eigenvalue of* $P(x)$ *then* λ *is also an eigenvalue of* $Q(x)$ *with the same partial multiplicities. Moreover,* $P(x)$ *and* $Q(x)$ *share the same elementary divisors with the only exception of* 1 *that might appear a different number of times.*

(ii) If v *is a right eigenvector of* $P(x)$ *relative to an eigenvalue* λ *then* $F(\lambda)[0, \dots, 0, v]^t$ *is of the form* $[0, \dots, 0, \tilde{v}]^t$ *where* $\tilde{v}$ *is a right eigenvector for* $Q(x)$*.*

(iii) The same holds for left eigenvectors, appropriately replacing $F(\lambda)$ *with* $E(\lambda)$*.*

Proof. Notice that, whenever $T(x)A(x)U(x) = D(x)$ is a Smith form for $A(x)$ then the diagonal matrix

$$\begin{bmatrix} I_r & \\ & T(x) \end{bmatrix} \begin{bmatrix} I_r & \\ & A(x) \end{bmatrix} \begin{bmatrix} I_r & \\ & U(x) \end{bmatrix} = \begin{bmatrix} I_r & \\ & D(x) \end{bmatrix}$$

is a Smith form for $\mathrm{diag}(I_r, A(x))$. Recalling that the form is unique we can conclude that this is the only possibility. Note that it is immediate to prove that $\mathrm{diag}(I_r, A(x))$ and $A(x)$ share the same elementary divisors except a finite number of additional ones on the top. Moreover, recalling that the Smith form is invariant for multiplication by unimodular matrices we have $\mathrm{diag}(I_r, P(x)))$ and $\mathrm{diag}(I_s, Q(x))$ must have the same Smith form. This implies, in particular, that $P(x)$ and $Q(x)$ share the same elementary divisors (except for 1 that might have different multiplicity) and so the same eigenvalues. This also implies that the partial multiplicities must be preserved.

The second statement can be directly verified by noting that $\mathrm{diag}(I_r, P(\lambda))v = 0$ implies that v has the first r components equal to 0 and so the remaining part must be an eigenvector for $P(x)$ relative to λ. Then we conclude by $\mathrm{diag}(I_s, Q(\lambda))E(\lambda)v = 0$ since $F(\lambda)$ is invertible. The same holds for the left eigenvectors. □

2.1.3 *Some classical examples of linearizations*

With these definitions available we can define the concepts of linearizations and ℓ-ifications.

Definition 2.1.16. A matrix polynomial $L(x)$ is said to be an *ℓ-ification* of a matrix polynomial $P(x)$ if $P(x) \smile L(x)$ and $\deg L(x) \leqslant \ell$. Whenever $\ell = 1$ we say that $L(x)$ is a *linearization* for $P(x)$. Moreover, if $P(x) \asymp L(x)$ we say that the ℓ-ification (or the linearization) is *strong*.

A very classical example of linearization is the so called Frobenius linearization. It is part of a bigger set of linearizations that are usually called *companion linearizations*.

We give the following (slightly informal) definition

Definition 2.1.17. We say that a linearization $L(x)$ of $P(x)$ is a *companion linearization* if its coefficients are obtained directly from the coefficient of $P(x)$ without any further computation. If $L(x)$ is monic of the form $xI - A$ we say that A is a *companion matrix* for $P(x)$.

The above definition is not completely precise, since we have not specified how the coefficients can or cannot be manipulated when creating the linearization. However, it is not very important in our setting and we just need to keep in mind that a companion linearization is something that we can construct easily by just looking at the coefficients of the matrix polynomial and by placing them in some order in a (usually) bigger matrix.

This concept is made more precise in [35], where templates are introduced.

Remark 2.1.18. The reader might recall that in Section 1.2 we have given the same definition for scalar polynomials. Indeed, in that context, we have only requested that the matrix A is obtained easily from the coefficients of the polynomial and that the eigenvalues of the two matches. This is enough in the scalar case since there is no need to match also the eigenvectors, given that in the scalar case there is no freedom for choosing them.

We now briefly report some classical choices for companion linearizations of a matrix polynomial $P(x)$. We have the following.

Definition 2.1.19 (First Frobenius form)**.** We say that the matrix polynomial $L(x) = xF_1^P - F_0^P$ is a *Frobenius linearization of the first kind* for $P(x)$ if

$$L(x) = x \begin{bmatrix} P_n & & & \\ & I & & \\ & & \ddots & \\ & & & I \end{bmatrix} - \begin{bmatrix} -P_{n-1} & -P_{n-1} & \cdots & -P_0 \\ I & & & \\ & \ddots & & \\ & & I & \end{bmatrix}.$$

Definition 2.1.20 (Second Frobenius form)**.** We say that the matrix polynomial $L(x) = xF_1^P - F_0^P$ is a *Frobenius linearization of the second kind* for $P(x)$ if

$$L(x) = x \begin{bmatrix} P_n & & & \\ & I & & \\ & & \ddots & \\ & & & I \end{bmatrix} - \begin{bmatrix} -P_{n-1} & I & & \\ -P_{n-2} & & \ddots & \\ \vdots & & & I \\ -P_0 & & & \end{bmatrix}.$$

We recall this classical result, found both in [61] and more recently in [35], with the latter using a notation more similar to ours.

Theorem 2.1.21. *Let $P(x)$ be a matrix polynomial. Then both the first and the second companion form of $P(x)$ are strong linearizations.*

Proof. We do not give the proof in this work, and instead we refer to [35] for a detailed analysis. The basic idea is that we can perform right and left multiplication by unimodular matrices until we take the linearization $L(x)$ in the form $\mathrm{diag}(I, P(x))$. This proves that $L \smile P(x)$. Then it suffices to perform a similar reduction on $L^\#(x)$ in order to obtain $\mathrm{diag}(I, P^\#(x))$ and the proof is complete. □

We now recall some basic results, reported for example in [35], that will be useful in the following. We give the statements referring to the first Frobenius form, but the analogous statements hold also for the second Frobenius form.

Lemma 2.1.22. *Let* $L(x)$ *be the first Frobenius form of a matrix polynomial* $P(x)$. *Then the right eigenvectors of* $L(x)$ *relative to an eigenvalue* λ *are of the form*

$$\hat{v} = \begin{bmatrix} \lambda^{n-1} v \\ \vdots \\ \lambda v \\ v \end{bmatrix}, \qquad P(\lambda)v = 0.$$

Proof. The statement proof can be proved by direct verification, by checking that

$$L(\lambda)\hat{v} = \begin{bmatrix} P(\lambda)v \\ 0 \\ \vdots \\ 0 \end{bmatrix} = 0.$$

By partitioning the vector $\hat{v}$ as $\hat{v}^t = [v_1^t \ \dots \ v_n^t]$ and imposing that $L(\lambda)\hat{v} = 0$ we can see that this is the only possibility. □

A similar result can be proved for left eigenvectors. We have the following.

Lemma 2.1.23. *Let* $L(x)$ *be the first Frobenius form of a matrix polynomial* $P(x)$. *Then the left eigenvectors of* $L(x)$ *relative to an eigenvalue* λ *are of the form*

$$\hat{v} = \begin{bmatrix} v \\ (\lambda I + P_{n-1}^t)v \\ (\lambda^2 I + \lambda P_{n-1}^t + P_{n-2}^t)v \\ \vdots \\ (\lambda^{n-1} I + \dots + P_1^t)v \end{bmatrix}, \qquad v^t P(\lambda) = 0.$$

Proof. Note that by partitioning $\hat{v}^t = \begin{bmatrix} v_1^t & \dots & v_n^t \end{bmatrix}$ and imposing $\hat{v}^t L(\lambda) = 0$ yields the equations

$$\begin{cases} -v_1^t P_{n-j} + v_{j+1}^t = \lambda v_j^t & \text{if } j < n \\ -v_1^t P_0 = \lambda v_n^t & \text{otherwise} \end{cases} \tag{2.1}$$

If we set $v := v_1$ we can recursively compute v_j for $j > 1$ by the above equations and we get that

$$\hat{v} = \begin{bmatrix} v \\ (\lambda I + P_{n-1}^t)v \\ (\lambda^2 I + \lambda P_{n-1}^t + P_{n-2}^t)v \\ \vdots \\ (\lambda^{n-1} I + \dots + P_1^t)v \end{bmatrix}$$

The last line of Equation (2.1) says that $\lambda v_n^t + v_1^t P_0 = 0$ and in our case this becomes $P(\lambda)^t v = 0$, concluding the proof. □

Some work have recently been carried out in order to generalize this family of linearizations. We can distinguish the following areas of study:

MONOMIAL COMPANION FORMS The study of natural companion forms for polynomial represented in the monomial basis, but different from the Frobenius form has received much attention recently. An example of this are the work by De Terán et al. in [34] for Fiedler linearizations. These linearizations are a generalization of the Frobenius forms. In [52] Fiedler notices that a Frobenius companion form can be factored in the product of matrices containing only one of the coefficients of the polynomial each. Since the eigenvalues of a matrix product do not change when reordering the elements, this can be used to generate a family of linearizations. Some have possibly interesting properties (such as banded structure, for example).

NON-MONOMIAL BASIS The Frobenius linearization is, in a certain sense, the natural linearization for polynomial expressed in the monomial basis. It is a natural question to ask whether this can be generalized to more general basis, i.e., if we can naturally define some families of linearizations for matrix polynomials of the form $P(x) = \sum_{i=0}^{n} P_i \phi_i(x)$ with $\{\phi_0, \dots, \phi_n\}$ generating a vector spaces of polynomials. This topic has been extensively studied by Corless, Amiraslani and Lancaster in [2] and [30]. In [2] they analyze bases of polynomials with recurrent relations (such as the monomial basis and orthogonal polynomial basis) and define linearizations with a strategy similar to the one reported here in Section 1.2.

STRUCTURED LINEARIZATIONS This category contains, in particular, the study of vector spaces containing linearization for polynomials. This is the approach that the authors use in [74] and is particularly effective in finding linearizations with particular structures. The main idea that motivated this work is to find linearizations for matrix polynomials that share the spectral symmetries available in the original matrix polynomial formulation. This has the advantage of allowing the use of structured methods (such as [76]) when dealing with this kind of problems. Some examples are matrix polynomials with symmetric or skew-symmetric coefficients or palindromic polynomials. All these structures induce different symmetries in the spectrum, and is desirable to preserve them when solving the matrix polynomial using a linearization.

2.2 THE CLASS OF SECULAR ℓ-IFICATIONS

In this section we introduce a new class of ℓ-ifications and linearizations inspired by the scalar linearizations presented in Section 1.3. Recall that, in that case, given a monic scalar polynomial $p(x)$ of degree n, we had to choose n nodes $b_1, \dots, b_n$ and compute n weights $a_1, \dots, a_n$ so that

$$-S(x)\Pi(x) = p(x), \quad S(x) = \sum_{i=1}^{n} \frac{a_i}{x - b_i} - 1, \quad \Pi(x) = \prod_{i=1}^{n} (x - b_i). \tag{2.2}$$

We want to generalize the above to matrix polynomials $P(x) = \sum_{i=0}^{n} P_i x^i$ with $P_i \in \mathbb{C}^{m \times m}$. Several problem will arise:

- For a scalar polynomial requiring the monicity condition is not very restrictive. In fact, if we are interested in the roots, we can always rescale the polynomial so that $p_n = 1$. The same holds for matrix polynomials, but only when $\det P_n \neq 0$. In the other cases it is not possible to transform the matrix polynomial into a monic one. Moreover, also in the cases where P_n is invertible but it is not well conditioned it might not be a good idea to invert it.

- Matrix product is not commutative so some care is needed in stating the above conditions for matrix polynomials. Due to this we will need to make some strong assumptions on the matrix polynomials that will take the role of $x - b_i$ in the scalar case.

2.2.1 *A scalar ℓ-ification*

In order to provide our generalization we start by generalizing the linearization for scalar polynomials introduced in Section 1.3. Note that the relations of Equation (2.2) can be rephrased by setting $b_i(x) = x - b_i$, $w_i = -a_i$ and

$$p(x) = b(x) + \sum_{i=1}^{n} w_i c_i(x), \quad c_i(x) = \prod_{\substack{j=1 \\ j \neq i}}^{n} b_i(x), \quad b(x) = \prod_{i=1}^{n} b_i(x). \tag{2.3}$$

In this framework we can rephrase Corollary 1.3.4 by saying that the matrix polynomial

$$A(x) = \begin{bmatrix} b_1(x) & & \\ & \ddots & \\ & & b_n(x) \end{bmatrix} + \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \begin{bmatrix} w_1 & \dots & w_n \end{bmatrix} =: D(x) + ew^t$$

is a linearization for $p(x)$. One might ask if for different choices of $b_i(x) \neq (x - b_i)$, provided that relation (2.3) still holds, the matrix polynomial $A(x)$ is still a linearization for $p(x)$.

We have the following

Theorem 2.2.1. *Let $b_1(x), \dots, b_q(x)$ be pairwise coprime polynomials and $w_1(x), \dots, w_q(x)$ be such that the polynomial $p(x)$ is equal to*

$$p(x) = b(x) + \sum_{i=1}^{q} w_i(x) c_i(x), \quad c_i(x) = \prod_{\substack{j=1 \\ j \neq i}}^{q} b_i(x), \quad b(x) = \prod_{i=1}^{q} b_i(x).$$

Then the matrix polynomial $A(x)$ defined by

$$A(x) := D(x) + ew^t(x) = \begin{bmatrix} b_1(x) & & \\ & \ddots & \\ & & b_q(x) \end{bmatrix} + \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \begin{bmatrix} w_1(x) & \dots & w_q(x) \end{bmatrix}$$

is a an ℓ-ification for $p(x)$ with degree $\ell = \max_{i=1,\dots,q} \max(\deg b_i(x), \deg w_i(x))$.

Proof. We do not give a complete proof of this result, since this is a special case of Theorem 2.2.5. Nevertheless, we show that it is immediate to prove that $\det A(x) = p(x)$. In fact, by applying the formula for the determinant of a rank 1 update to a matrix yields

$$\det A(x) = \det D(x)(1 + w^t(x) D(x)^{-1} e) = \prod_{i=1}^{q} b_i(x) \cdot (1 + \sum_{i=1}^{q} \frac{w_i(x)}{b_i(x)}) = p(x)$$

for every x that is not a root of $b(x)$. Since the equality holds for an infinite number of values of x and the determinant is a polynomial, we have the thesis. $\square$

2.2.2 *An ℓ-ification for matrix polynomials*

Our aim here is to generalize the construction of Theorem 2.2.1 to matrix polynomials. In the following we will consider, as usual, a regular matrix polynomial $P(x) = \sum_{i=0}^{n} P_i x^i$, not necessarily monic. In particular, we allow $\det P_n = 0$, so that infinity eigenvalues might appear.

For this construction we require some basic results and definitions that are presented here.

Definition 2.2.2. Let $P(x)$, $Q(x)$ two matrix polynomials. We say that $P(x)$ and $Q(x)$ are *left coprime* if there exist two matrix polynomials $\alpha(x)$ and $\beta(x)$ such that

$$P(x)\alpha(x) + Q(x)\beta(x) = I.$$

The above definition coincides with the more usual one of asking that for any matrix polynomial $D(x)$ of degree at least one $D^{-1}(x)P(x)$ and $D^{-1}(x)Q(x)$ cannot be both polynomials, that is, $P(x)$ and $Q(x)$ do not share any left factor.

Lemma 2.2.3. *Let $B_1(x), B_2(x) \in \mathbb{C}^{m\times m}[x]$ be regular and such that $B_1(x)B_2(x) = B_2(x)B_1(x)$. Assume that $B_1(x)$ and $B_2(x)$ are left coprime, that is, there exist $\alpha(x), \beta(x) \in \mathbb{C}^{m\times m}[x]$ such that $B_1(x)\alpha(x) + B_2(x)\beta(x) = I_m$. Then the 2×2 block-matrix polynomial $F(x) = \begin{bmatrix} \alpha(x) & B_2(x) \\ -\beta(x) & B_1(x) \end{bmatrix}$ is unimodular.*

Proof. From the decomposition

$$\begin{bmatrix} I_m & 0 \\ B_1(x) & -B_2(x) \end{bmatrix} \begin{bmatrix} \alpha(x) & B_2(x) \\ -\beta(x) & B_1(x) \end{bmatrix} = \begin{bmatrix} \alpha(x) & B_2(x) \\ I_m & 0 \end{bmatrix}$$

we have $-\det B_2(x) \det F(x) = -\det B_2(x)$. Since $B_2(x)$ is regular then $\det F(x) = 1$. □

Lemma 2.2.4. *Let $P(x)$, $Q(x)$ and $T(x)$ be matrix polynomials such that $P(x)$ and $Q(x)$ are both left coprime with $T(x)$ and $P(x)T(x) = T(x)P(x)$. Then $P(x)Q(x)$ and $T(x)$ are also left coprime.*

Proof. We know that there exist $\alpha_P(x)$, $\beta_P(x)$, $\alpha_Q(x)$, $\beta_Q(x)$, matrix polynomials such that

$$P(x)\alpha_P(x) + T(x)\beta_P(x) = I, \quad Q(x)\alpha_Q(x) + T(x)\beta_Q(x) = I.$$

We shall prove that there exist appropriate $\alpha(x)$, $\beta(x)$ matrix polynomials such that $P(x)Q(x)\alpha(x) + T(x)\beta(x) = I$. We have

$$\begin{aligned} &P(x)Q(x)(\alpha_Q(x)\alpha_P(x)) + T(x)(P(x)\beta_Q(x)\alpha_P(x) + \beta_P(x)) = \\ &P(x)(Q(x)\alpha_Q(x) + T(x)\beta_Q(x))\alpha_P(x) + T(x)\beta_P(x) = \\ &P(x)\alpha_P(x) + T(x)\beta_P(x) = I, \end{aligned}$$

where the first equality holds since $T(x)P(x) = P(x)T(x)$. So we can conclude that also $P(x)Q(x)$ and $T(x)$ are left coprime, and a possible choice for $\alpha(x)$ and $\beta(x)$ is:

$$\alpha(x) = \alpha_Q(x)\alpha_P(x), \qquad \beta(x) = P(x)\beta_Q(x)\alpha_P(x) + \beta_P(x).$$

□

With the above tools and results we can state the generalization of theorem 2.2.1:

Theorem 2.2.5. *Let* $B_1(x), \dots, B_q(x)$ *be regular, pairwise left-coprime matrix polynomials such that* $B_i(x)B_j(x) = B_j(x)B_i(x)$ *and* $W_1(x), \dots, W_q(x)$ *so that the matrix polynomial* $P(x)$ *can be decomposed as*

$$P(x) = B(x) + \sum_{i=1}^{q} W_i(x)C_i(x), \quad C_i(x) = \prod_{\substack{j=1 \\ j \neq i}}^{q} B_i(x), \quad B(x) = \prod_{i=1}^{q} B_i(x).$$

Then the matrix polynomial $A(x)$ *defined by*

$$A(x) := D(x) + (e \otimes I_m)W(x) = \begin{bmatrix} B_1(x) & & \\ & \ddots & \\ & & B_q(x) \end{bmatrix} + \begin{bmatrix} I_m \\ \vdots \\ I_m \end{bmatrix} \begin{bmatrix} W_1(x) & \dots & W_q(x) \end{bmatrix}$$

is a an ℓ*-ification for* $P(x)$ *with degree* $\ell = \max_{i=1,\dots,q} \max(\deg B_i(x), \deg W_i(x))$.

Proof. Recall that, in view of Definition 2.1.16, we need to prove that $A(x) \smile P(x)$, i.e., that there exists two unimodular matrix polynomials $E(x)$ and $F(x)$ such that $E(x)A(x)F(x) = \mathrm{diag}(I, P(x))$. We do this by providing an explicit construction for $E(x)$ and $F(x)$.

Let E_0 be the following matrix:

$$E_0 = \begin{bmatrix} I_m & -I_m & & & \\ & I_m & -I_m & & \\ & & \ddots & \ddots & \\ & & & I_m & -I_m \\ & & & & I_m \end{bmatrix}.$$

A direct inspection shows that

$$E_0A(x) = \begin{bmatrix} B_1(x) & -B_2(x) & & & \\ & B_2(x) & -B_3(x) & & \\ & & \ddots & \ddots & \\ & & & B_{q-1}(x) & -B_q(x) \\ W_1(x) & W_2(x) & \cdots & W_{q-1}(x) & B_q(x) + W_q(x) \end{bmatrix}.$$

Using the fact that the polynomials $B_i(x)$ are left co-prime, we transform the latter matrix into block diagonal form. We start by annihilating $B_1(x)$. Since $B_1(x), B_2(x)$ are left co-prime, there exist matrix polynomials $\alpha(x)$, $\beta(x)$ such that $B_1(x)\alpha(x) + B_2(x)\beta(x) = I_m$. For the sake of brevity, from now on we write α, β, W_i and B_i in place of $\alpha(x)$, $\beta(x)$, $W_i(x)$ and $B_i(x)$, respectively, dropping the variable x. Observe that the matrix

$$F_1(x) = \begin{bmatrix} \alpha & B_2 \\ -\beta & B_1 \end{bmatrix} \oplus I_{m(q-2)}.$$

is unimodular in view of Lemma 2.2.3 and

$$E_0A(x)F_1(x) = \begin{bmatrix} I_m & & & & \\ -B_2\beta & B_1B_2 & -B_3 & & \\ & & \ddots & \ddots & \\ & & & B_{q-1} & -B_q \\ W_1\alpha - W_2\beta & W_1B_2 + W_2B_1 & \cdots & W_{m-1} & B_q + W_q \end{bmatrix}.$$

Using row operations we set to zero all the elements in the first column of this matrix (by just adding multiples of the first row to the others). That is, there exists a suitable unimodular matrix $E_1(x)$ such that

$$E_1(x)E_0A(x)F_1(x) = \begin{bmatrix} I_m & & & & \\ & B_1B_2 & -B_3 & & \\ & & \ddots & \ddots & \\ & & & B_{q-1} & -B_q \\ & W_2B_1 + W_1B_2 & \cdots & W_{q-1} & B_q + W_q \end{bmatrix}.$$

In view of Lemma 2.2.4, B_1B_2 is left coprime with B_3. Thus, we can recursively apply the same process until we arrive at the final reduction step:

$$E_{q-1}(x)\dots E_1(x)E_0A(x)F_1(x)\dots F_{q-1}(x) = I_{m(q-1)} \oplus \left(\prod_{i=1}^{q} B_i(x) + \sum_{i=1}^{q} W_i(x)C_i(x)\right)$$

where the last diagonal block is exactly $P(x)$ in view of the hypothesis of the Theorem. □

Theorem 2.2.5 provides a framework to build linearizations for matrix polynomials. However, several questions might arise from its formulation:

- Is the constructed ℓ-ification strong in the sense of Definition 2.1.16?
- How do we find the $B_i(x)$ and the $W_i(x)$ for the decomposition given in the hypothesis of Theorem 2.2.5?

The next two subsections will be devoted to answer these questions.

2.2.3 *Building a strong ℓ-ification*

In this section we prove that under suitable (relatively mild) conditions, the ℓ-ification built with Theorem 2.2.5 is strong.

We see that as soon as the basic requirements for being a strong ℓ-ifications, that is the invariants provided in [35], are satisfied, then the ℓ-ification is strong.

More precisely, we have the following.

Theorem 2.2.6. *Assume that the hypothesis of Theorem 2.2.5 are satisfied and that the following conditions hold:*

(i) All the matrix polynomials $B_i(x)$ have the same degree d and $n = dq$.

(ii) For every $i = 1, \dots, q$ we have $\deg W_i(x) < \deg B_i(x)$.

(iii) The matrix polynomials $B_i{}^{\#}(x)$ are pairwise left coprime.

Then the ℓ-ification $A(x)$ of the matrix polynomial $P(x)$ is a strong ℓ-ification.

Proof. Recall that, in order to prove the strength of our ℓ-ification, we need to ensure that $A^{\#}(x)$ is extended unimodularly equivalent to $P^{\#}(x)$, that is, $A^{\#}(x) \smile P^{\#}(x)$. Note that, since $\deg A(x) = d$ we have $A^{\#}(x) = x^d A(x^{-1})$ and so

$$A^{\#}(x) = \begin{bmatrix} B_1^{\#}(x) & & \\ & \ddots & \\ & & B_q^{\#}(x) \end{bmatrix} + (e \otimes I_m) \begin{bmatrix} x^d W_1(x^{-1}) & \dots & x^d W_q(x^{-1}) \end{bmatrix}.$$

Since we have assumed that the matrix polynomials $B_i^{\#}(x)$ are pairwise left coprime the above matrix polynomial satisfies the hypothesis of Theorem 2.2.5 and so we have

$$A^{\#}(x) \smile \prod_{i=1}^{q} B_i^{\#}(x) + \sum_{i=1}^{q} x^d W_i(x^{-1}) \prod_{j \neq i} B_j^{\#}(x).$$

Replacing $B_i^{\#}(x)$ with $x^d B(x^{-1})$ yields the following

$$A^{\#}(x) \smile x^{dq} \prod_{i=1}^{q} \left(B_i(x^{-1}) + \sum_{i=1}^{q} W_i(x^{-1}) C_i(x^{-1}) \right) = x^{dq} P(x^{-1}) = P^{\#}(x)$$

that gives us the thesis. □

2.2.4 *Some practical choices of the nodes*

The generality of Theorem 2.2.5 makes it difficult to figure out how these nodes $B_i(x)$ should be chosen. In fact, we are given some freedom for this. In this section we present a set of practical choices that are (almost) always applicable and that try to mimic the secular linearization for scalar polynomials presented in Section 1.3.

In that context we chose the (scalar polynomial) $b_i(x) = x - b_i$. where b_i are not roots of the polynomial. Here we propose a similar pattern. Let $B_i(x) = b_i(x) I_m$ where $b_i(x)$ is a scalar polynomial. Then we have the following

Theorem 2.2.7. *Let* $\{b_i(x) \mid i = 1, \dots, q\}$ *be a set of monic pairwise coprime scalar polynomials and* $B_i(x) := b_i(x) I_m$. *Then, for every monic matrix polynomial* $P(x)$ *of degree* $n := \sum_{i=1}^{q} \deg b_i(x)$ *there exist* $W_i(x)$ *for* $i = 1, \dots, q$ *so that* $\deg W_i(x) < \deg B_i(x)$ *and*

$$P(x) = B(x) + \sum_{i=1}^{q} W_i(x) C_i(x)$$

where $B(x)$ *and* $C_i(x)$ *are defined as in Theorem 2.2.5.*

Proof. We can see the Theorem as an interpolation problem. In fact, since both the $B_i(x)$ and $P(x)$ are monic, we have $P(x) - B(x)$ is a polynomial of degree (at most) $n - 1$. Then, recalling that the $C_i(x)$ are multiple of the identity we have that, entry-wise,

$$(P(x) - B(x))_{s,k} = \sum_{i=1}^{q} W_i(x)_{s,k} c_i(x).$$

The above shows that $W_i(x)_{s,k}$, the polynomial in position (s,k) of $W_i(x)$ depends only on the other polynomials in position (s,k) of $W_j(x)$ with $j \neq i$. This implies that the problem is completely decoupled and we can simply prove that there exist scalar polynomials $w_i(x)$ such that

$$q(x) = \sum_{i=1}^{q} w_i(x) c_i(x)$$

for every $q(x)$ of degree at most $n-1$. Recalling that the $b_i(x)$ are relatively prime the above implies that

$$q(x) \mod b_i(x) \equiv \left(\sum_{j=1}^{q} w_j(x) c_j(x) \right) \mod b_i(x) \equiv w_i(x) c_i(x) \mod b_i(x).$$

Since $c_i(x)$ is invertible modulo $b_i(x)$ we can write

$$q(x) c_i(x)^{-1} \mod b_i(x) = w_i(x) \mod b_i(x)$$

so that we can choose $w_i(x)$ as the remainder of the division by $b_i(x)$ of $q(x)$ times the inverse of $c_i(x)$ modulo $b_i(x)$. This yields a polynomial $w_i(x)$ of degree at most $\deg b_i(x) - 1$ so that $\sum_{i=1}^{q} w_i(x) c_i(x)$ has degree at most $n-1$. Moreover, we have that for every root ξ_j of $b_i(x)$ $q(\xi_j) = w_i(\xi_j) c_i(\xi_j)$ so $q(x)$ and $\sum_{i=1}^{q} w_i(x) c_i(x)$ coincide in at least n points, thus they are equal. Choosing $q(x) := (P(x) - B(x))_{s,k}$ and setting $W_i(x)_{s,k}$ as the computed $w_i(x)$ gives us the thesis. □

Remark 2.2.8. Theorem 2.2.7 provides a practical way to compute the ℓ-ification by means of computations on scalar polynomials. These can be carried out in an efficient way as shown in the next subsection 2.2.7. In the case where the $b_i(x)$ are of degree 1 of the form $b_i(x) = x - b_i$ the projection modulo $b_i(x)$ is simply an evaluation at the point b_i and so we have the following formula for the computation of W_i (that are now constants, since $\deg W_i(x) < \deg b_i(x) = 1$):

$$W_i = P(b_i) c_i(b_i)^{-1} = \frac{P(b_i)}{\prod_{j \neq i} (b_i - b_j)}$$

that can be efficiently evaluated using the Horner rule if the polynomial $P(x)$ is given in the monomial basis.

In the next theorem we discuss another choice of nodes $B_i(x)$ that can be seen as a generalization of the one of Theorem 2.2.7.

Theorem 2.2.9. *Let $B_i(x)$ be a set of monic diagonal matrix polynomials for $i = 1, \dots, q$ such that for every $j = 1, \dots, n$ the scalar polynomials $\{B_i(x)_{j,j} \mid i = 1, \dots, q\}$ are pairwise prime. Then, for every monic matrix polynomial $P(x)$ of degree $n := \sum_{i=1}^{q} \deg B_i(x)$ there exist $W_i(x)$ for $i = 1, \dots, q$ so that $\deg W_i(x) < \deg B_i(x)$ and*

$$P(x) = B(x) + \sum_{i=1}^{q} W_i(x) C_i(x)$$

where $B(x)$ and $C_i(x)$ are defined as in Theorem 2.2.5.

Proof. It is immediate to verify that the condition on the coprimality of the diagonal polynomials yields the left (and right) coprimality of the $B_i(x)$, so that Theorem 2.2.5 can be applied.

Moreover, we can mimic the proof of Theorem 2.2.7 and write

$$(P(x) - B(x)) = \sum_{i=1}^{q} W_i(x) C_i(x).$$

Since the $C_i(x)$ are diagonal we have that for every (s, k)

$$(P(x) - B(x)) = \sum_{i=1}^{q} W_i(x)_{s,k} \prod_{j \neq i} B_i(x)_{k,k}$$

This is the same kind of equation found in the proof of Theorem 2.2.7 and its solvability is guaranteed by the primality conditions that we have imposed and the fact that $P(x) - B(x)$ has degree at most $n - 1$. □

Remark 2.2.10. Note that, in general, the ℓ-ifications provided by Theorem 2.2.5 will be dense matrix polynomials. Nevertheless, it is possible to apply another unimodular (actually constant) transformation in order to make them sparse. Sparsity is often a desirable feature since it might make it easy to exploit the structure in evaluations and computations. Note that, if we choose

$$E = \begin{bmatrix} I_m & & & \\ -I_m & I_m & & \\ & \ddots & \ddots & \\ & & -I_m & I_m \end{bmatrix}$$

we have that $E \cdot A(x)$ is the sum of a bidiagonal matrix polynomial and a matrix polynomial with only the first block row different from 0:

$$E \cdot A(x) = \begin{bmatrix} B_1(x) + W_1(x) & W_2(x) & \dots & W_{q-1}(x) & W_q(x) \\ -B_1(x) & B_2(x) & & & \\ & -B_2(x) & \ddots & & \\ & & \ddots & B_{q-1}(x) & \\ & & & -B_{q-1}(x) & B_q(x) \end{bmatrix}.$$

Since $\det E = 1$ and E is a constant matrix we have $A(x) \asymp E \cdot A(x)$.

It is worth stressing that it is not necessary to transform the matrix polynomial to a sparse form in order to exploit its structure. The diagonal plus low rank structure of $A(x)$ is a particular kind of quasiseparable structure that is the topic of Chapter 3. In that chapter we show how to perform the typical operations that are needed to solve eigenvalue problems (such Hessenberg reduction and Hessenberg triangular reduction) efficiently on these kinds of matrices.

2.2.5 *Handling non-monic polynomials*

In the previous subsection we have shown practical choices of $B_i(x)$ that allows to build ℓ-ifications for monic matrix polynomials. It is clear that, since the $B_i(x)$ that we have proposed are monic so is also the product $B(x)$ and given our degree constraints it is impossible to find

$W_i(x)$ so that $P(x) = B(x) + \sum_{i=1}^{q} W_i(x)C_i(x)$ if $P(x)$ is not monic. Here we provide an easy generalization of the previous approach that works for general (even with singular leading coefficient) matrix polynomials.

Let $P(x) = \sum_{i=0}^{n} P_i x^i$ be a regular matrix polynomial, not necessarily monic, $b_i(x)$ for $i = 1, \dots, q$ a set of scalar polynomials and s a parameter. Assume that the following conditions hold:

- The polynomials $b_i(x)$ are pairwise coprime.
- For every λ eigenvalue of $P(x)$ and for every root ξ of $b_i(x)$, $i < q$ we have that $\lambda b_q(\xi) + s \neq 0$.
- The degree of $b_i(x)$ is d_i and $\sum_{i=1}^{q} d_i = n = \deg P(x)$.

Then we can choose $B_i(x)$ to be

$$B_i(x) = \begin{cases} b_i(x) I_m & \text{if } i < q \\ b_q(x) P_n + s I_m & \text{otherwise} \end{cases}. \tag{2.4}$$

We give here a technical Lemma that will be used to obtain the existence of the $W_i(x)$:

Lemma 2.2.11. *Let $b_i(x)$, $i = 1, \dots, q$ be co-prime scalar polynomials of degree $d_1, \dots, d_q$, respectively, such that $\sum_{i=1}^{q} d_i = n$. If $P_1(x)$, $P_2(x)$ are matrix polynomials of degree at most $n-1$ then $P_1(x) = P_2(x)$ if and only if $P_1(x) - P_2(x) \equiv 0 \mod b_i(x)$, for $i = 1, \dots, q$.*

Proof. The implication $P_1(x) - P_2(x) = 0 \Rightarrow P_1(x) - P_2(x) \equiv 0 \mod b_i(x)$ is trivial. Conversely, if $P_1(x) - P_2(x) \equiv 0 \mod b_i(x)$ for every b_i then the entries of $P_1(x) - P_2(x)$ are multiples of $\prod_{i=1}^{q} b_i(x)$ for the co-primality of the polynomials $b_i(x)$. But this implies that $P_1(x) - P_2(x) = 0$ since the degree of $P_1(x) - P_2(x)$ is at most $n-1$ while $\prod_{i=1}^{q} b_i(x)$ has degree n. □

We can now state the following.

Theorem 2.2.12. *Let $B_i(x)$ be matrix polynomials chosen as in Equation (2.4) for a matrix polynomial $P(x)$. Then there exist $W_i(x)$ matrix polynomials, for $i = 1, \dots, q$, such that*

$$P(x) = B(x) + \sum_{i=1}^{q} W_i(x) C_i(x)$$

where $B(x)$ and $C_i(x)$ are defined according to the notation of Theorem 2.2.5. Moreover, the formula for the computation of the $W_i(x)$ can be given explicitly and is the one here reported in Equation (2.5).

Proof. We have, by construction, that $\deg(P(x) - B(x)) < n$. If we restrict to choose $W_i(x)$ with $\deg W_i(x) < d_i$ then the equality

$$P(x) - B(x) = \sum_{i=1}^{q} W_i(x) C_i(x)$$

holds if and only if it holds modulo $b_i(x)$ for $i = 1, \dots, q$ in view of Lemma 2.2.11. Projecting the equation leads to the following formulas for the computation of $W_i(x)$:

$$W_i(x) = \begin{cases} \frac{P(x)}{\prod_{j=1, j\neq i}^{q-1} b_j(x)} (b_q(x) P_n + s I_m)^{-1} \mod b_i(x) & \text{if } i < q \\ \frac{1}{\prod_{j=1}^{q-1} b_j(x)} P(x) - s I_m - s \sum_{j=1}^{q-1} \frac{W_j(x)}{b_j(x)} \mod b_q(x) & \text{otherwise} \end{cases} \tag{2.5}$$

Notice that the hypothesis on $\lambda b_q(\xi) + s \neq 0$ guarantees the invertibility of $b_q(x)P_n + sI_m$ modulo $b_i(x)$, so that the above formulas are well-defined. □

An important feature of Theorem 2.2.12 is that, under mild hypothesis, the ℓ-ification obtained following the procedure is strong.

Theorem 2.2.13. *Let $A(x)$ be the secular ℓ-ification for a polynomial $P(x)$ built according the hypothesis of Theorem 2.2.12. Assume that the degrees d_i are all equal to d and $dq = n$ and that the shift s is chosen so that $\lambda b_q^{\#}(\frac{1}{\xi}) + s\xi^{-q} \neq 0$ for every ξ root of $b_i(x)$, $i < q$. Then $A(x)$ is a strong ℓ-ification for $P(x)$.*

Proof. According to Theorem 2.2.6 we only need to check that the reversal of the matrix polynomials $B_i(x)$ are pairwise coprime. This is trivial for the reversals of $B_i(x)$ and $B_j(x)$ for $i \neq j$ and $i, j < q$.

It remains to show that this holds for $B_i^{\#}(x)$ and $B_q^{\#}(x)$, $i < q$. We have $B_i^{\#}(x) = b_i^{\#}(x)I_m$ and $B_q^{\#}(x) = sx^qI + b_q^{\#}(x)P_n$. If V is a Jordan basis for P_n we can write

$$V^{-1}B_q^{\#}(x)V = \begin{bmatrix} b_q^{\#}(x)J_{\lambda_1} + sx^qI & & & \\ & \ddots & & \\ & & b_q^{\#}(x)J_{\lambda_k} + sx^qI & \\ & & & sx^qI \end{bmatrix}$$

In particular, for every root ξ of $b_i(x)$ with $i < q$ the matrix $V^{-1}B_q^{\#}(\frac{1}{\xi})V$ is invertible, and so $B_i^{\#}(x)$ and $B_q^{\#}(x)$ are left coprime. This concludes the proof. □

2.2.6 *Characterization of right and left eigenvectors*

Usually, when solving polynomial eigenvalues problems, one is not only interested in the eigenvalues of $P(x)$, but also in the eigenvectors. As we have already shown in Theorem 2.1.15 the eigenvalues of $P(x)$ and of one of its linearizations (or ℓ-ifications) are the same. The eigenvectors, instead, are connected with a relation involving $E(\lambda)$ and $F(\lambda)$, the unimodular matrices defining the equivalence. Usually the explicit expression of $E(x)$ and $F(x)$ is not available or it is not easy to handle, so an explicit characterization of the eigenvectors of the ℓ-ification in terms of the ones of $P(x)$ (and the other way round) is desirable.

Providing this characterization is precisely the aim of this subsection. We suppose that $A(x)$ is an ℓ-ification provided by Theorem 2.2.5, a so-called *secular ℓ-ification*.

Theorem 2.2.14. *Let $P(x)$ be a matrix polynomial, $A(x)$ its secular ℓ-ification defined in Theorem 2.2.5, $\lambda \in \mathbb{C}$ such that $\det P(\lambda) = 0$, and assume that $\det B_i(\lambda) \neq 0$ for all $i = 1, \dots, q$. If $v_A = (v_1^t, \dots, v_q^t)^t \in \mathbb{C}^{mq}$ is such that $A(\lambda)v_A = 0$, $v_A \neq 0$ then $P(\lambda)v = 0$ where $v = -\prod_{i=1}^q B_i(\lambda)^{-1} \sum_{j=1}^q W_j v_j \neq 0$. Conversely, if $v \in \mathbb{C}^m$ is a nonzero vector such that $P(\lambda)v = 0$, then the vector v_A defined by $v_i = \prod_{j \neq i} B_j(\lambda)v$, $i = 1, \dots, q$ is nonzero and such that $A(\lambda)v_A = 0$.*

Proof. Let $v_A \neq 0$ be such that $A(\lambda)v_A = 0$, so that

$$B_i(\lambda)v_i + \sum_{j=1}^q W_j(\lambda)v_j = 0, \quad i = 1, \dots, q. \tag{2.6}$$

Let $v = -(\prod_{i=1}^q B_i(\lambda)^{-1}) \sum_{j=1}^q W_j(\lambda)v_j$. Combining the latter equation and (2.6) yields

$$v_i = -B_i(\lambda)^{-1}\left(\sum_{j=1}^q W_j(\lambda)v_j\right) = \prod_{j=1, j\neq i}^q B_j(\lambda)v. \tag{2.7}$$

Observe that if $v = 0$ then, by definition of v, one has $\sum_{j=1}^q W_j(\lambda)v_j = 0$ so that, in view of (2.6), we find that $B_i(\lambda)v_i = 0$. Since $\det B_i(\lambda) \neq 0$ this would imply that $v_i = 0$ for any i so that $v_A = 0$ which contradicts the assumptions. Now we prove that $P(\lambda)v = 0$. In view of (2.7) we have

$$P(\lambda)v = \prod_{j=1}^q B_j(\lambda)v + \sum_{i=1}^q W_i(\lambda) \prod_{j=1, j\neq i}^q B_j(\lambda)v = \prod_{j=1}^q B_j(\lambda)v + \sum_{i=1}^q W_i(\lambda)v_i.$$

Moreover, by definition of v we get

$$P(\lambda)v = -\prod_{j=1}^q B_j(\lambda)(\prod_{i=1}^q B_i(\lambda)^{-1}) \sum_{i=1}^q W_i(\lambda)v_i + \sum_{i=1}^q W_i(\lambda)v_i = 0.$$

Similarly, we can prove the opposite implication. □

A similar result can be proven for left eigenvectors. The following theorem relates left eigenvectors of $A(x)$ and left eigenvectors of $P(x)$.

Theorem 2.2.15. *Let $P(x)$ be a matrix polynomial, $A(x)$ its secular ℓ-ification defined in Theorem 2.2.5, $\lambda \in \mathbb{C}$ such that $\det P(\lambda) = 0$, and assume that $\det B_i(\lambda) \neq 0$. If $u_A^t = (u_1^t, \dots, u_q^t) \in \mathbb{C}^{mq}$ is such that $u_A^t A(\lambda) = 0$, $u_A \neq 0$, then $u^t P(\lambda) = 0$ where $u = \sum_{i=1}^q u_i \neq 0$. Conversely, if $u^t P(\lambda) = 0$ for a nonzero vector $u \in \mathbb{C}^m$ then $u_A^t A(\lambda) = 0$, where u_A is a nonzero vector defined by $u_i^t = -u^t W_i(\lambda) B_i(\lambda)^{-1}$ for $i = 1, \dots, q$.*

Proof. If $u_A^t A(\lambda) = 0$ then from the expression of $A(x)$ given in Theorem 2.2.5 we have

$$u_i^t B_i(\lambda) + \left(\sum_{j=1}^q u_j^t\right) W_i(\lambda) = 0, \quad i = 1, \dots, q. \tag{2.8}$$

Assume that $u = \sum_{j=1}^q u_j = 0$. Then from the above expression we obtain, for any i, $u_i^t B_i(\lambda) = 0$ that is $u_i = 0$ for any i since $\det B_i(\lambda) \neq 0$. This is in contradiction with $u_A \neq 0$. From (2.8) we obtain $u_i^t = -u^t W_i(\lambda) B_i(\lambda)^{-1}$. Moreover, multiplying (2.8) to the right by $\prod_{j=1, j\neq i}^q B_j$ yields

$$0 = u_i^t \prod_{j=1}^q B_j(\lambda) + u^t W_i(\lambda) \prod_{j=1, j\neq i}^q B_j(\lambda).$$

Taking the sum of the above expression for $i = 1, \dots, q$ yields

$$0 = \left(\sum_{i=1}^q u_i^t\right) \prod_{j=1}^q B_j(\lambda) + u^t \sum_{i=1}^q W_i(\lambda) \prod_{j=1, j\neq i}^q B_j(\lambda) = u^t P(\lambda).$$

Conversely, assuming that $u^t P(\lambda) = 0$, from the representation

$$P(x) = \prod_{j=1}^{n} B_j(x) + \sum_{i=1}^{q} W_i(x) \prod_{j=1, j\neq i}^{q} B_i(x),$$

defining $u_i^t = -u^t W_i(\lambda) B_i(\lambda)^{-1}$ we obtain

$$\sum_{i=1}^{q} u_i^t = -u^t \sum_{i=1}^{q} W_i(\lambda) B_i(\lambda)^{-1} = -u^t (P(\lambda) \prod_{j=1}^{q} B_j(\lambda)^{-1} - I) = u^t$$

and therefore from (2.8) we deduce that $u_A^t A(\lambda) = 0$. □

The above results do not cover the case where $\det B_i(\lambda) = 0$ for some i.

2.2.7 *Computations in polynomial rings*

As explained in Theorem 2.2.7 and Theorem 2.2.9 in order to compute a secular ℓ-ification it is necessary to perform operations on polynomials. Whilst for choices such as $B_i(x) = (x - b_i) I_m$ the operations turn out to be equivalent to evaluations and so they can be carried out efficiently by means of the Horner rule, other cases with higher degree $B_i(x)$ need to be handled differently.

In the particular case that we have analyzed we have chosen the $B_i(x)$ to be diagonal. This leads to a complete decoupling of the equation for the determination of the $W_i(x)$, and so the problem can be reduced to a scalar one. In this case, one just needs to solve the problem of computing the inverse of a scalar polynomial modulo another one, that is, given two polynomials $s(x)$ and $u(x)$ we need to find $t(x)$ such that $t(x)s(x) = 1 \mod u(x)$. In the following we suppose that $\deg s(x), \deg t(x) < \deg u(x)$. This is not restrictive since these two polynomials are only defined uniquely in the quotient ring $\mathbb{F}[x]/(u(x))$ and so they can always be replaced with the remainder of the division by $u(x)$. We propose the following interpolating strategy for the computation of the coefficients of $t(x)$:

- We evaluate $s(x)$ at n points in the complex plane, where $n = \deg u(x)$. These points are chosen to be the roots of $u(x)$.
- We compute the inverses of the evaluations of $s(x)$. These must be equal to the evaluations of $t(x)$ since there must exist a polynomial $v(x)$ such that $s(x)t(x) + u(x)v(x) = 1$ and so evaluating in a root of $u(x)$ gives us $s(\xi)t(\xi) = 1$.
- We recover the coefficients of $t(x)$ by interpolation on these nodes. Note that the interpolation points are enough for the task since $\deg t(x) < n$.

Remark 2.2.16. If the scalar polynomials have high degree some issues might arise due to bad conditioning of the interpolation problem. In these cases it might be a good idea to choose the nodes $B_i(x)$ (when they are a scalar polynomial times the identity matrix) so that their eigenvalues are near to the roots of the unity (properly scaled). This will lead to an interpolation problem near to a discrete Fourier transform, thus removing any problem of numerical conditioning. Note that, if $b_i(x) = x^d - 1$ then the algorithm for FFT can be used for the computation of the $W_i(x)$, making the cost of the computation of $O(m^2 d \log d)$ flops. Moreover, this approach can be easily generalized to the case of $b_i(x) = x^d - \alpha$ for $\alpha \neq 1$.

The general case, where the polynomials $B_i(x)$ are not diagonal, could be more involved, since the problem cannot be reduced to a scalar one.

In general, we have to solve the equation

$$P(x) - B(x) = \sum_{i=1}^{q} W_i(x) C_i(x).$$

where the coefficients of the $W_i(x)$ are the unknowns.

2.2.8 *Correlation with the Frobenius linearization*

In this section we show that, in a certain sense, our secular ℓ-ification is a generalization of the classical Frobenius linearization. We show that, for an appropriate choice of the nodes $B_i(x)$, we obtain a linearization that is unitarily similar to the standard Frobenius forms. In particular, the conditioning of the eigenvalue problem is the same. One might wonder if "smarter" choice of nodes could lead to better conditioned eigenvalue problems. This is in fact the case, and is the topic of the next Section 2.3.

Let $P(x) = \sum_{i=0}^{n} P_i x^i$ be a monic $m \times m$ matrix polynomial of degree n and ζ_n a primitive n-th root of the unity. Consider the choice $B_i(x) = (x - \zeta_n^i) I_m$ that leads to the choice of W_i according to Remark 2.2.8

$$W_i = P(\zeta_n^i) \cdot \prod_{j \neq i} \frac{1}{\zeta_n^i - \zeta_n^j}.$$

We can observe that if we define $q(x) = x^n - 1$ then $q(x) = \prod_{j=1}^{n}(x - \zeta_n^j)$ so that $\prod_{j \neq i}(\zeta_n^i - \zeta_n^j) = q'(\zeta_n^i) = n\zeta_n^{-i}$.

Consider now Ω_n the unitary Fourier matrix defined by $(\Omega_n)_{i,j} = \frac{\zeta_n^{ij}}{\sqrt{n}}$. We have $\Omega^* \Omega = I$ and

$$(\Omega^* \otimes I_m) \left(\begin{bmatrix} B_1(x) & & \\ & \ddots & \\ & & B_n(x) \end{bmatrix} + \begin{bmatrix} I_m \\ \vdots \\ I_m \end{bmatrix} \begin{bmatrix} W_1 & \dots & W_n \end{bmatrix} \right) (\Omega_n \otimes I_m) = F(x)$$

where

$$F(x) = xI_m - C \otimes I_m + \begin{bmatrix} P_0 + I_m \\ P_1 \\ \vdots \\ P_{n-1} \end{bmatrix} (e_n^t \otimes I_m) = xI_m - \begin{bmatrix} 0 & & & -P_0 \\ I_m & & & \vdots \\ & \ddots & & \vdots \\ & & I_m & -P_{n-1} \end{bmatrix}$$

is (one of) the classical Frobenius linearizations. Since Ω_n is unitary it does not alter the conditioning of the eigenvalue problem, and so the ℓ-ification, which in this particular case is a linearization, is neither worse nor better than the usual choice (regarding the conditioning).

2.3 THE USE OF TROPICAL ROOTS FOR PEPS

The aim of this section is to generalize the results that we have given for scalar polynomials in Section 1.4 to the case of matrix polynomials.

We want to answer the following question: given a matrix polynomial $P(x) = \sum_{i=0}^{n} P_i x^i$ is it possible to derive simple and cheap bounds on its eigenvalues using tropical algebra? We review the main results of [19, 77, 82], where this problem is analyzed.

We have the following result reported in Theorem 2.1 of [19]:

Theorem 2.3.1 (Pellet). $P(x) = \sum_{i=0}^{n} P_i x^i \in \mathbb{C}^{m\times m}[x]$ *be a regular matrix polynomial. Then, for every* k *such that* $\det P_k \neq 0$ *we have*

- *If* $0 < k < n$ *and* P_k *is invertible then the scalar polynomial equation*
$$q_k(x) = x^k - \sum_{\substack{i=1\\ i\neq k}}^{n} \|P_k^{-1} P_i\| x^i = 0$$
has either zero or two positive real solutions $r_k^l \leqslant r_k^u$.
- *In the latter case, the annulus* $\{z \in \mathbb{C} \mid r_k^l < |z| < r_k^u\}$ *contains no eigenvalues of* $P(x)$. *Moreover, the disc of center* 0 *and radius* r_k^l *contains* mk *eigenvalues of* $P(x)$.
- *If* $k = 0$ *then the equation has only one positive real solution* r_0^u *and the matrix polynomial has no eigenvalues of modulus smaller than* r_0^u.
- *If* $k = n$ *then the equation has only one positive real solution* r_n^l *and the matrix polynomial has no eigenvalues bigger than* r_n^l *in modulus.*

Note that the above theorem gives bounds for the locations of the eigenvalues that are very similar to the bounds obtained for the root of the polynomials given by Corollary 1.4.9. In fact, we can state a similar result also in this case.

Corollary 2.3.2. *Let* $r_{k_i}^l$ *and* $r_{k_i}^u$ *be the roots of the polynomial equation of Theorem 2.3.1 for the values of* k_i *where they have at least one positive real root. We set* $r_0^l = 0$ *and* $r_n^u = \infty$. *Then the annuli*
$$\mathcal{A}_{k_i} := \{r_{k_i}^u \leqslant |z| \leqslant r_{k_{i+1}}^l\} \subseteq \mathbb{C}$$
contain exactly $m(k_{i+1} - k_i)$ *eigenvalues of* $P(x)$.

Proof. It is possible to show that $r_{k_i}^u \leqslant r_{k_{i+1}}^l$ (see [19] for the complete proof). Then looking at the complementaries of the annuli defined in Theorem 2.3.1 gives us the thesis. □

As in the scalar case we can use the Newton polygon and the tropical roots to compute some radii inside these annuli. We have the following result, proven in [19].

Theorem 2.3.3. *Let* $P(x)$ *be a matrix polynomial of degree* n *with coefficients of order* $m \times m$. *If* k_i *is an index such that the Pellet polynomial of Theorem 2.3.1 has two positive real roots* $r_{k_i}^l \leqslant r_{k_i}^u$ *then* $u_{k_i} \leqslant r_{k_i}^l \leqslant r_{k_i}^u \leqslant v_{k_i}$ *where*
$$u_{k_i} := \max_{j<k_i} \|P_{k_i}^{-1} P_j\|^{\frac{1}{k_i - j}}, \qquad v_{k_i} := \min_{j<k_i} \|P_{k_i}^{-1} P_j\|^{\frac{1}{k_i - j}}.$$
Moreover, the same holds for the only available lower (resp. upper) bound whenever P_0 *(resp.* P_n*) is invertible. In particular we have that the following relation holds in every case in which they are computable:*
$$r_0^u \leqslant v_0 := \min_{j>0} \|P_0^{-1} P_j\|^{-\frac{1}{j}}, \qquad r_n^l \geqslant u_0 := \max_{j<n} \|P_n^{-1} P_j\|^{\frac{1}{n-j}}.$$

Remark 2.3.4. In this formulation the connection with tropical roots might not be apparent at first sight. However, if we take the logarithm of the values u_{k_i} and v_{k_i} we note that the maximums are taken on the slopes of the Newton polygon and so these values are tropical roots of the tropical polynomial

$$\mathrm{tp}_{k_i}(x) := \bigoplus_{i=0}^{n} \log \|P_{k_i}^{-1} P_i\|.$$

A noteworthy difference with the scalar case is that here we are considering n different tropical polynomials and we only take a pair of roots from each of them. This is related to the fact that, in the generic matrix case, $\|A\|^{-1} \neq \|A\|^{-1}$ and $\|AB\| \neq \|A\|\|B\|$.

The above remark shows that considering the usual tropical polynomial

$$\mathrm{tp}(x) := \bigoplus_{i=0}^{n} \log \|P_i\| x^{\otimes i} \tag{2.9}$$

might not be possible in the matrix polynomial case. However, when the coefficients of the polynomials are well-conditioned this formulation can be used again. Theorem 2.7 of [82] justifies this claim by providing bounds to the annuli that contain the roots in this case. The theorem is rather technical and making the bounds explicit requires some work, so we refer to the original paper for the details.

The theory of Pellet bounds to the eigenvalues of matrix polynomials have received also other recent contributions such as the work by Melman [77, 78]. In particular, Theorem 3.3 of [77] provides an alternative formulation to the one that we have given in Theorem 2.3.1 by defining a replacement for $q_k(x)$ by

$$\tilde{q}_k(x) = \|P_k^{-1}\|^{-1} x^k - \sum_{j \neq k} \|P_j\| x^j.$$

The roots of these polynomials provide alternative $\tilde{r}_k^l \leqslant \tilde{r}_k^u$. Moreover, in the proof of the Pellet bound we are interested in identifying the regions with no eigenvalues, that are identified by complex numbers with modulus contained in the real interval where $q_k(x) \geqslant 0$. Since $\|P_k^{-1} P_j\| \leqslant \|P_k^{-1}\| \|P_j\|$ it is immediate that $\|P_k^{-1}\| \tilde{q}_k(x) \leqslant q_k(x)$, so that $\{x \in \mathbb{R} \mid \tilde{q}_k(x) \geqslant 0\} \subseteq \{x \in \mathbb{R} \mid q_k(x) \geqslant 0\}$. In particular, the bounds obtained from Theorem 2.3.1 are stricter than the one of [77]. However, the $\tilde{q}_k(x)$ are cheaper to compute (since they require $O(n)$ norms instead of $O(n^2)$) and so they can be a good compromise.

In [82] the authors analyze the three strategies that we have presented (Theorem 2.3.1, Theorem 3.3 of [77] and their own strategy based on the polynomial $\mathrm{tp}(x)$ of Equation (2.9)) and prove that the bounds obtained from the tropical polynomial of Equation (2.9) are looser than the one of Melman in [77].

Concluding, we have provided three different bounds for the location of the eigenvalues. The bounds are of decreasing quality but also of decreasing cost for their computation, which justify their use in practice.

As we show in Section 2.4 the tropical roots of [82] are already "good enough" for most cases. However, in that context we show also a case (the `orr_sommerfeld` problem) in which using Pellet's Theorem is more effective, and where the tropical roots only provide a partial picture of the eigenvalue distribution.

Remark 2.3.5. In the scalar polynomial case we excluded the possibility of explicitly computing the roots of the polynomial $s_k(x)$ of Lemma 1.4.8 that are the analogous of the $q_k(x)$ in the matrix case. In fact, it would not have been wise to solve n polynomials just to get some good approximations for the roots of another one (even if we only need the positive real roots of the first n polynomials, so we might use some ad-hoc technique such as bisection plus refinement).

In the matrix polynomial case, instead, this technique might be worth using since the cost of the solution of a polynomial eigenvalue problem of degree n in $\mathbb{C}^{m\times m}[x]$ is expected to be at least $O(n^2m^3)$ (see Remark 2.1.1). In view of this fact, the cost of the solution of the n scalar polynomials might be negligible, especially when the degree n is not big compared to the size of the matrices.

2.4 NUMERICAL EXPERIMENTS

Our aim in this section is to provide numerical evidence that what we have proved for the scalar case holds also in the matrix case. In particular, we have seen that the secular linearization constructed for scalar polynomial $p(x)$ is such that if the nodes b_i are chosen as good approximations of the polynomial roots then the conditioning of the eigenvalue problem is small.

In the matrix case we can make similar choices with the secular ℓ-ification. Here we will analyze the simpler case where $\ell = 1$ and the matrix polynomial is monic. We can immediately spot a difference with the scalar case: now we have mn eigenvalues (counted with multiplicities and with the infinity ones) and only n nodes to choose, so it is generally impossible to fit the nodes to the eigenvalues. However, we can obtain a similar result trying to fit the nodes to the moduli of groups of eigenvalues. To this end, the tropical roots are a very good way to choose the nodes, since they are suitable to detect cluster of eigenvalues of similar moduli [19, 59].

We report some examples taken from [21]. We first show some random polynomials with coefficients with unbalanced norms (some very large, and some very small). These are good examples in our setting since tropical roots work very well in detecting the moduli of different blocks of eigenvalues that often appear in these cases.

Then, we show some real world example (or at least more realistic) taken from the NLEVP collection [8, 9, 10]. We analyze the Orr-Sommerfeld problem and the planar waveguide one, since they are degree 4 PEPs. This allows us to obtain more information from tropical roots (4 tropical roots) with respect to lower degree polynomials.

All the experiments have been run in MATLAB R2011b (7.13.0.564) on a Linux PC.

2.4.1 *Measuring the conditioning of the eigenvalue problem*

In order to judge the quality of our results we need to evaluate the conditioning of the eigenvalues of the linearizations that we have built. We have used the MATLAB function `condeig` for this. This function measures the conditioning of each eigenvalue by the well-known formula

$$\kappa_i = \frac{\|v_i\|\,\|w_i\|}{\langle v_i, w_i\rangle}. \tag{2.10}$$

where v_i and w_i are the right and left eigenvectors relative to the eigenvalues λ_i, respectively [91].

2.4.2 *A first example: polynomials with unbalanced norms*

We have generated polynomials with unbalanced coefficient norms using the MATLAB code of Algorithm 4. In the example we have reported a matrix polynomial of degree 5 and size 64×64.

Algorithm 4 MATLAB code that generates a random matrix polynomial with unbalanced coefficient norms.

```
n = 5; m = 64;
P = {};
 for i = 1 : n
  P{i} = exp(12 * randn) * randn(m);
end
P{n+1} = eye(m);
```

We have then applied three different strategies for the computation of a linearization:

(i) We have computed a classical Frobenius linearization placing the block coefficients of the polynomial in the usual way.

(ii) We have computed a secular linearization using Theorem 2.2.5 by choosing the values b_i as the mean of blocks of 64 eigenvalues of $P(x)$. More precisely, we have ordered the eigenvalues by modulus and then we have taken the mean of each block vector of size 64. We have then obtained 5 increasing estimates for the moduli of the eigenvalues of $P(x)$.

(iii) We have used the tropical roots as the nodes b_i.

In the last two cases we have sometimes changed the sign of the nodes, so to have some b_i in the positive half plane and some in the negative one. We have done this in order to spread them as much as possible and trying to be near to the actual eigenvalues.

In Figure 2.2 are reported the conditioning of the eigenvalues of 4 such random problems. The plots are reported by relating the conditioning (on the y axis) and the moduli of the eigenvalues (on the x axis). Recall that a classical Frobenius linearization is equivalent (regarding the conditioning) to a secular one with nodes chosen as the roots of the unity. For this reason, it might be natural to expect it to be very well-conditioned on eigenvalues of moduli around 1, while the secular linearizations might be better on the others. In fact, it can be spotted in Figure 2.2 on the rightmost pictures that the eigenvalues of moduli 1 are well-conditioned in all the examples.

It is also interesting to note that the secular linearizations manage to be well-conditioned on all the clusters of eigenvalues. Moreover, the tropical roots estimates of the moduli are just as good as the block mean computed a-posteriori. This seems to confirm that they can be used to obtain good starting approximations for the nodes b_i.

2.4.3 *Some examples from the NLEVP collection*

Here we show some other examples taken from the NLEVP collection that have been analyzed in [21]. These examples might be more challenging for our framework since, in general, we do not expect the eigenvalues to come in blocks with similar modulus. However, it can be seen

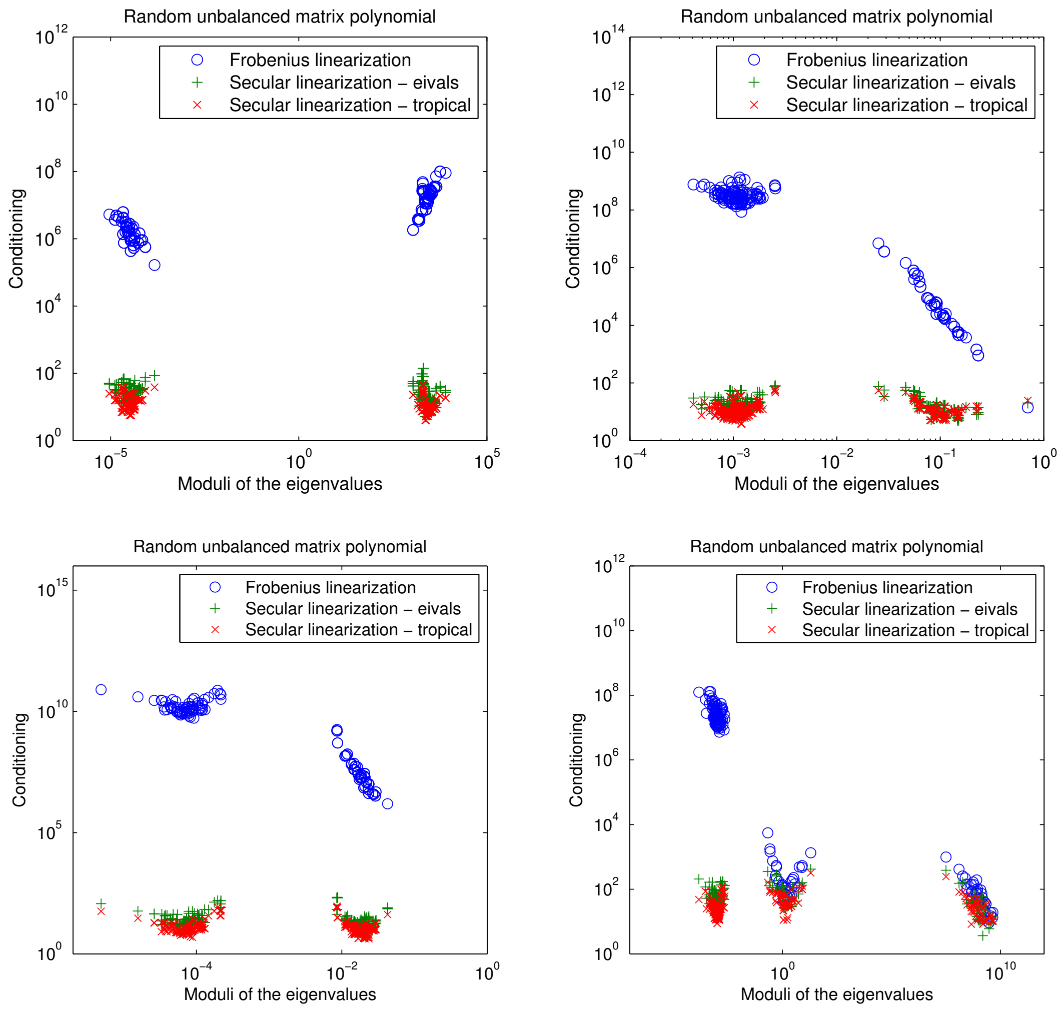


Figure 2.2: Conditioning of the eigenvalues of different linearizations for some matrix polynomials with random coefficients having unbalanced norms.

that many problems have eigenvalues on different scales that cause difficulties, and this kind of implicit scaling provided by the secular linearization coupled with the tropical roots can be helpful.

As a first example we consider the problem `orr_sommerfeld`. Using the tropical roots we can find some values inside the unique annulus that is identified by Theorem 1.4.10. In this example the values obtained only give a partial picture of the eigenvalue distribution. The Pellet theorem gives about `1.65e-4` and `5.34` as lower and upper bound to the moduli of the eigenvalues, but the tropical roots are rather small and near to the lower bound. More precisely, the tropical roots are `1.4e-3` and `1.7e-4` with multiplicities 3 and 1, respectively.

This leads to a secular linearization that is well-conditioned for the smaller eigenvalues but with a higher conditioning on the eigenvalues of bigger modulus as can be seen in Figure 2.3 on the left (the eigenvalues are sorted in non-increasing order with respect to their modulus).

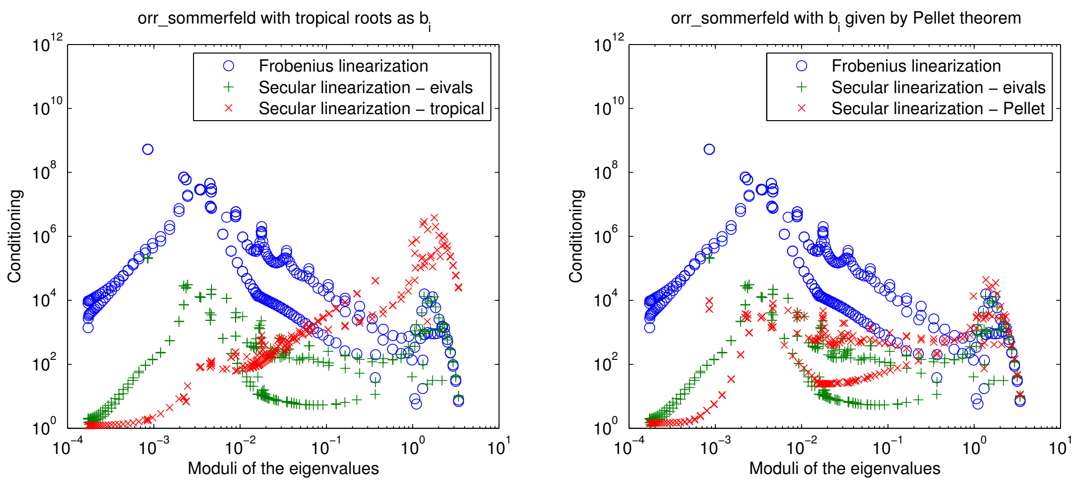


Figure 2.3: On the left we report the conditioning of the Frobenius and of the secular linearization with the choices of b_i as mean of subsets of eigenvalues with close moduli and as the estimates given by the tropical roots. On the right the tropical roots are coupled with estimates given by the Pellet theorem.

It can be seen, though, that coupling the tropical roots with the results from Theorem 1.4.10 and altering the b_i by adding a value slightly smaller than the upper bound (in this example we have chosen 5 but the result is not very sensitive to this choice) leads to a much better result that is reported in Figure 2.3 on the right. In the right figure we have used the vector of the nodes b_i defined by `b = [ 1.7e-4, 1.4e-3, -1.4e-3, 5 ]`. In this case coupling the Pellet theorem is helpful and is in fact a relatively cheap procedure, since it only require the solution of a scalar polynomial of degree 4 and the computation of some norms. This seems to justify that there exists a link between the quality of the approximations obtained through the tropical roots and the conditioning properties of the secular linearization.

We analyzed also the `planar_waveguide` example problem from the NLEVP collection. The results are shown in Figure 2.4. This problem is a PEP of degree 4 with two tropical roots approximately equal to 127.9 and 1.24. Again, it can be seen that for the eigenvalues of smaller modulus (that are near the tropical root 1.24) the Frobenius linearization and the secular one behave in the same way, whilst for the bigger ones the secular linearization has some advantage in the conditioning. This may be justified by the fact that the Frobenius linearization is similar to a secular linearization on the roots of the unity.

Since in this case the information given by the tropical roots is more accurate, the secular linearization built without altering them already has good conditioning properties. Moreover, the results obtained with the tropical roots and with the block mean of the eigenvalues are approximately the same.

These two examples might suggest that the tropical roots can not be trusted as a "universal solution", but it might be worth to couple them with other analysis, such as the bounds obtained by Theorem 1.4.10. In practice, some hand-tuning of the parameters might be needed to obtain good linearizations, but automatic tuning seems to work well enough to give an advantage over the classical Frobenius linearization.

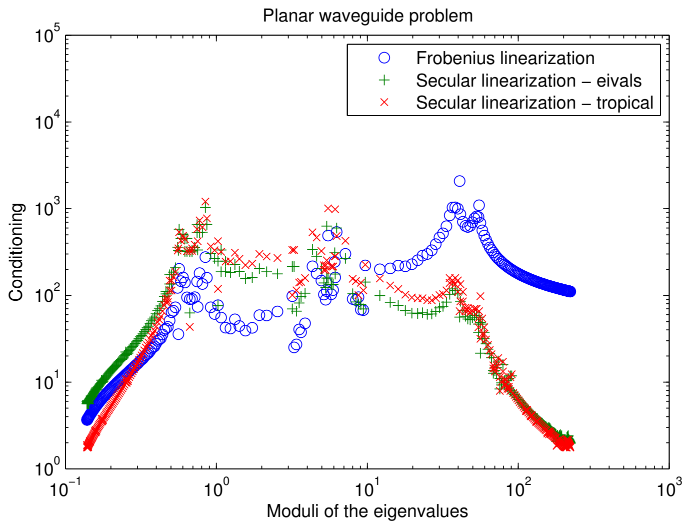


Figure 2.4: Conditioning of the eigenvalues for three different linearizations on the `planar_waveguide` problem.

As a final example, it is natural to wonder if computing the secular linearization in floating point actually gives any advantage over solving the classical Frobenius form. In fact, even if the new linearization is better conditioned, the transformation from the polynomial to the secular linearization might be bad conditioned in general, since the global conditioning of the problem is an intrinsic feature that we cannot avoid. However, an advantage here is that we are limiting the effect of the bad conditioning to the computation of $A(x)$ that only requires the evaluation of $P(x)$ at a point. We know that the Horner scheme to be a stable algorithm, and it is also very simple to analyze so we can derive precise bounds on the error that we make.

In particular, the Horner scheme is relatively cheap, since it requires only $O(nm^2)$ flops and we need to carry out $O(n)$ evaluations. For this reason the cost of computing $A(x)$ is $O(n^2m^2)$ flops and we can possibly use multiprecision for this step to overcome these difficulties (assuming that the coefficients are know precisely).

We report a last example, where we have tried to find the eigenvalues of a matrix polynomial defined by integer coefficients using only standard floating point arithmetic and by using `polyeig`, the Frobenius and our secular linearization (using the tropical roots as b_i) coupled with the QZ method. We have chosen the polynomial $P(x) = P_{11}x^{11} + P_9x^9 + P_2x^2 + P_0$ where

$$P_{11} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ & 1 & 1 & 1 \\ & & 1 & 1 \\ & & & 1 \end{bmatrix}, \quad P_9 = 10^8 \begin{bmatrix} 3 & 1 & & \\ 1 & 3 & 1 & \\ & 1 & 3 & 1 \\ & & 1 & 3 \end{bmatrix}, \quad P_2 = 10^8 P_{11}^t, \quad P_0 = \begin{bmatrix} 1 & & & \\ & 2 & & \\ & & 3 & \\ & & & 4 \end{bmatrix}$$

Computing the tropical roots yields good estimates of the blocks of eigenvalues of the matrix polynomial. We obtain the tropical roots $1.2664 \cdot 10^4$, 0.9347 and $1.1786 \cdot 10^{-4}$ with multiplicities 2, 7 and 2, respectively. To have a trusted result to compare with we have computed

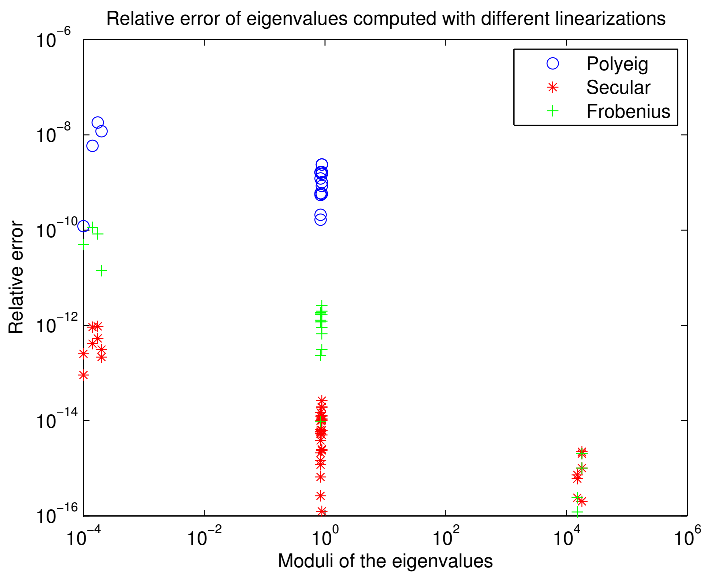


Figure 2.5: The accuracy of the computed eigenvalues using polyeig, the Frobenius linearization and the secular linearization with the b_i obtained through the computation of the tropical roots.

the eigenvalues with higher precision and we have compared them with the results obtained by our two approaches. We stress that in this case the secular linearization has been computed with standard floating point arithmetic. As shown in Figure 2.5 we have achieved much better accuracy in the secular case with respect to `polyeig`. The secular linearization has achieved a relative error of the order of the machine precision on all the eigenvalues except the smaller block (with modulus about 10^{-4}). In that case the relative error is about 10^{-12} but the absolute error is, again, of the order of the machine precision. Moreover, `polyeig` fails to detect the eigenvalues with larger modulus, marking them as infinite eigenvalues.

2.5 ANOTHER KIND OF SECULAR LINEARIZATIONS

In this section we introduce a new kind of linearization that can be seen as a generalization of the secular ℓ-ifications introduced in Section 2.2 when $\ell = 1$. The construction that we report here does not provide a framework to generate ℓ-ifications but solves another issue that is undesired with the secular linearizations previously introduced: while in most cases we have been able to experimentally prove that the numerical conditioning can be low, we can not guarantee that this will always be the case (given good choices for the nodes). This is essentially related to the fact that we can choose a little number of parameters (only n, the degree of the polynomial) with respect to the number of eigenvalues that we have to compute.

Here we provide an alternative framework for building linearizations that uses estimates for eigenvalues and eigenvectors in order to build linearizations as well-conditioned as desired. This is done in the same spirit of the results obtained for scalar polynomials in Section 1.3.

We start this analysis by extending the observations of Section 2.2.8, where we have shown the connection of secular linearizations with the Frobenius linearization.

For simplicity, we start by analyzing the case of a monic matrix polynomial $P(x) = \sum_{i=0}^{n} P_i x^i$ and we extend the results to non-monic matrix polynomials later.

2.5.1 *Linearizations for monic matrix polynomials*

In this section we consider two matrix polynomials at the same time. We already introduced the polynomial $P(x)$, that is, the matrix polynomial that we want to linearize and of which we would like to find the eigenstructure. Throughout this section we assume that $P(x)$ is monic. At the same time we consider another monic polynomial $Q(x)$ that has semisimple eigenvalues, and we suppose to have built it artificially. In fact, as we will see, this polynomial can be seen as a technical tool for the construction but it will not be needed to know it explicitly.

We introduce the following notation: we denote by $xF_1^P - F_0^P$ the Frobenius linearization for the matrix polynomial $P(x)$ so that we have

$$xF_1^P - F_0^P = x \begin{bmatrix} P_n & & & \\ & I & & \\ & & \ddots & \\ & & & I \end{bmatrix} - \begin{bmatrix} -P_{n-1} & -P_{n-2} & \dots & -P_0 \\ I & & & \\ & \ddots & & \\ & & I & \end{bmatrix}.$$

Clearly the same construction can be done for the matrix polynomial $Q(x)$, so we also have a linearization $xF_1^Q - F_0^Q$ that we know to be diagonalizable (thanks to the hypothesis on the semisimpleness of the eigenvalues).

In particular we know that if we call V the matrix whose columns are the eigenvectors of the (monic) polynomial $xF_1^Q - F_0^Q$ then

$$V^{-1}(xF_1^Q - F_0^Q)V = xI - D, \qquad D = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_{mn} \end{bmatrix}.$$

We observe that the numerical conditioning of the eigenvalue problem related to the matrix polynomial $xI - D$ is 1, since a basis for the eigenvectors are the columns of the identity matrix. In this sense the change of basis induced by V is a good choice for this problem.

We now assume that the eigenvalues λ and the right eigenvectors $v_i := Ve_i$ are good approximations to the eigenvalues and eigenvectors of the matrix F_0^P. Can we still say that the change of basis V is a good choice for the solution of the matrix polynomial? As we will see, the answer is yes. In particular, if $Q(x)$ is a nearby polynomial of $P(x)$ then, by continuity, we have

$$V^{-1}(xF_1^P - F_0^P)V = xI - D + E$$

where E is small. Moreover, $xI - D + E$ is a linearization of $P(x)$. The purpose of this section is to show that these claims hold and also to provide a method to compute such a linearization without the need of explicitly building and inverting V.

We have the following lemma that summarizes the previous remarks.

Lemma 2.5.1. *Let* $P(x)$, $Q(x)$ *be two monic matrix polynomials, and assume that* $Q(x)$ *has only finite, simple and pairwise different eigenvalues. Let then* V_Q *and* D_Q *be the matrices with the eigenvectors and the eigenvalues of* $Q(x)$*, respectively, and* $xI - F_0^P$ *be a linearization for* $P(x)$. *Then the pencil*

$$A(x) = xI - \begin{bmatrix} V_Q D_Q^{n-1} \\ \vdots \\ V_Q D_Q \\ V_Q \end{bmatrix}^{-1} F_0^P \begin{bmatrix} V_Q D_Q^{n-1} \\ \vdots \\ V_Q D_Q \\ V_Q \end{bmatrix}$$

is a linearization for $P(x)$ *that can be decomposed as the sum of a diagonal matrix polynomial plus a rank* m *constant matrix. We call this linearization the* secular linearization *for* $P(x)$ *associated with* $Q(x)$.

Proof. Let $\hat{V}_Q$ be the $nm \times nm$ invertible matrix defined by

$$\hat{V}_Q = \begin{bmatrix} V_Q D_Q^{n-1} \\ \vdots \\ V_Q D_Q \\ V_Q \end{bmatrix}.$$

The invertibility of $\hat{V}_Q$ is guaranteed by the fact that we have required the eigenvalues of $Q(x)$ to be semisimple [61]. Recall that, if we call F_0^Q the companion matrix for $Q(x)$ and F_0^P the companion matrix for $P(x)$, we have

$$F_0^P - F_0^Q = \begin{bmatrix} Q_{n-1} - P_{n-1} & \dots & Q_0 - P_0 \\ 0 & \dots & 0 \\ \vdots & & \vdots \\ 0 & \dots & 0 \end{bmatrix} = (e_1 \otimes I_m) \begin{bmatrix} Q_{n-1} - P_{n-1} & \dots & Q_0 - P_0 \end{bmatrix}.$$

In particular we can note that $F_0^P - F_0^Q$ has rank (at most) m. Moreover, we know that $\hat{V}_Q^{-1} F_0^Q \hat{V}_Q = D_Q = \mathrm{diag}(\lambda_1, \dots, \lambda_{nm})$ and so we have

$$\begin{aligned} \hat{V}_Q^{-1}(xI - F_0^P)\hat{V}_Q &= xI - \hat{V}_Q^{-1} F_0^Q \hat{V}_Q - \hat{V}_Q^{-1}(F_0^P - F_0^Q)\hat{V}_Q && (2.11) \\ &= xI - D_Q - \hat{V}_Q^{-1}(e_1 \otimes I_m) \begin{bmatrix} Q_{n-1} - P_{n-1} & \dots & Q_0 - P_0 \end{bmatrix} \hat{V}_Q && (2.12) \end{aligned}$$

that clearly has the required diagonal plus low rank (and constant) structure. □

Before getting to our characterization of the structure of the low rank term we need some technical results. We prove the following also in the general case of a non-monic matrix polynomial.

Lemma 2.5.2. *Let $Q(x)$ be a matrix polynomial with only finite and semisimple eigenvalues. Suppose that $\lambda_1, \dots, \lambda_{nm}$ are its eigenvalues and v_i, u_i^t the corresponding right and left eigenvectors, respectively. Then if $D_Q = \mathrm{diag}(\lambda_1, \dots, \lambda_{nm})$ and $\hat{V}_Q$, $\hat{W}_Q$ are the matrices defined as*

$$\hat{V}_Q = \begin{bmatrix} V_Q D_Q^{n-1} \\ \vdots \\ V_Q D_Q \\ V_Q \end{bmatrix}, \qquad V_Q = \left[\begin{array}{c|c|c} v_1 & \dots & v_{nm} \end{array}\right], \qquad \hat{W}_Q = F_1^Q \hat{V}_Q$$

the inverse $\hat{W}_Q^{-1}$ has the form

$$\hat{W}_Q^{-1} = \begin{bmatrix} U_Q^t & D_Q U_Q^t Q_n + U_Q^t Q_{n-1} & \cdots & D_Q^{n-1} U_Q^t Q_n + \dots + U_Q^t Q_1 \end{bmatrix}$$

where U_Q is a matrix containing the left eigenvectors of $Q(x)$, i.e.,

$$U_Q^t = \left[\begin{array}{c} u_1^t \\ \hline \vdots \\ \hline u_{nm}^t \end{array}\right], \quad u_i^t Q(\lambda_i) = 0,\ \forall i = 1, \dots, nm.$$

Proof. Consider a diagonalizable pencil $xA - B$ so that there exist W and V such that

$$W^{-1}(xA - B)V = xI - D.$$

The right and left eigenvectors of the pencil on the right-hand side are the vectors of the canonical basis e_i, for $i = 1, \dots, mn$. We automatically have that the right eigenvectors of $xA - B$ are $v_i := Ve_i$ and the left ones are $w_i := e_i^t W^{-1}$. When A is invertible we have that $W = AV$. Since $Q(x)$ has no infinite eigenvalues we have that Q_n is invertible and so is $\mathrm{diag}(Q_n, I, \dots, I)$ that is the leading coefficient of the Frobenius pencil $xF_1^Q - F_0^Q$. In view of Lemma 2.1.23 and Lemma 2.1.22 we conclude that the eigenvector matrices are of the form

$$\hat{V}_Q = \begin{bmatrix} V_Q D_Q^{n-1} \\ \vdots \\ V_Q D_Q \\ V_Q \end{bmatrix}, \qquad V_Q = \left[\begin{array}{c|c|c} v_1 & \dots & v_{nm} \end{array}\right]$$

and

$$\hat{W}_Q^{-1} = \begin{bmatrix} U_Q^t & D_Q U_Q^t Q_n + U_Q^t Q_{n-1} & \cdots & D_Q^{n-1} U_Q^t Q_n + \dots + U_Q^t Q_1 \end{bmatrix}$$

with $\hat{W}_Q = \mathrm{diag}(Q_n, I, \dots, I)\hat{V}_Q$ as requested. □

We now want to characterize the structure of the constant coefficient of this linearization $A(x)$. We have the following

Theorem 2.5.3. *Let $A(x)$ be the secular linearization of Lemma 2.5.1 for a monic polynomial $P(x)$. Then $A(x)$ has the form*

$$A(x) = xI + UW^t$$

where U is a $nm \times m$ matrix whose rows are the left eigenvectors of $Q(x)$ and W is a $nm \times m$ matrix such that the columns of W^t are of the form $P(\lambda_i)v_i$ where λ_i is an eigenvalue of $Q(x)$ and v_i is a right eigenvector of $Q(x)$. In particular, if we set $R := UW^t$ we have

$$R = (r_{ij}), \qquad r_{ij} = u_i^t P(\lambda_j) v_j.$$

where u_i and v_j are left and right eigenvectors corresponding to λ_i and λ_j, respectively, normalized so that $u_i^t Q'(\lambda_i) v_i = 1$.

Proof. Notice that we can write

$$\begin{aligned}
A(x) = \hat{W}_Q^{-1}(xF_1^P - F_0^P)\hat{V}_Q &= \hat{W}_Q^{-1}(xF_1^Q - F_0^Q +)\hat{V}_Q + \hat{W}_Q^{-1}(F_0^Q - F_0^P)\hat{V}_Q \\
&= xI - D_Q + \hat{W}_Q^{-1}\begin{bmatrix} P_{n-1} - Q_{n-1} & \dots & P_0 - Q_0 \\ & & \\ & & \end{bmatrix}\begin{bmatrix} VD^{n-1} \\ \vdots \\ V \end{bmatrix} \\
&= xI - D_Q + \hat{W}_Q^{-1}(e_1 \otimes I_m)\left(\sum_{i=0}^{n-1} P_i V D^i - \sum_{i=0}^{n-1} Q_i V D^i\right) \\
&= xI - D_Q + \hat{W}_Q^{-1}(e_1 \otimes I_m)\sum_{i=0}^{n} P_i V D^i.
\end{aligned}$$

where the last step is obtained by adding VD^n to both sums (since $Q(x)$ and $P(x)$ share the same leading coefficient equal to I) and recalling that, by definition, $\sum_{i=0}^n Q_i V D^i = 0$. Since $\hat{W}_Q^{-1}$ is multiplied by $e_1 \otimes I_m$ we only need its first block column that in view of Lemma 2.1.23 is equal to the left eigenvectors of $Q(x)$ organized in rows.

We then note that $\hat{W}_Q^{-1}\hat{V}_Q = I$ since they diagonalize the pencil $\hat{W}_Q^{-1}(xF_1^Q - F_0^Q)\hat{V}_Q = xI - D_Q$. This implies that, for each i,

$$e_i^t \hat{W}_Q^{-1}\hat{V}_Q e_i = 1 \iff \begin{bmatrix} u_i \\ (\lambda_i I + Q_{n-1}^t)u_i \\ (\lambda_i^2 I + \lambda_i Q_{n-1}^t + Q_{n-2}^t)u_i \\ \vdots \\ (\lambda_i^{n-1} I + \ldots + Q_1^t)u_i \end{bmatrix}^t \begin{bmatrix} \lambda_i^{n-1} v_i \\ \vdots \\ \lambda v_i \\ v_i \end{bmatrix} = 1 \iff u_i^t Q'(\lambda_i) v_i = 1.$$

This completes the proof. □

2.5.2 *Properties of the linearization*

We have some direct consequences of the construction above. The first natural consequence is that we can prove a matrix version of Theorem 1.3.14 that states that the conditioning of the secular linearization for scalar polynomials goes to zero as the nodes go to the roots. We have the following

Theorem 2.5.4. *Let $A_k(x)$ be the secular linearization for a monic matrix polynomial $P(x)$ with only semisimple and finite eigenvalues relative to a matrix polynomial $Q_k(x)$ for $k \in \mathbb{N}$. Then if $Q_k(x) \to P(x)$ as $k \to \infty$ the conditioning of the eigenvalue problem associated with $A_k(x)$ goes to 1.*

Proof. The conditioning of the eigenvalue problem of the secular linearization $xI - D_Q$ associated with $Q(x)$ where D_Q contains the eigenvalues of $Q(x)$ is clearly 1, since both the eigenvector matrices are the identity. Since all the operations involved in constructing $A_k(x)$ are continuous functions of $Q_k(x)$ and its spectral data and $Q_k(x) \to P(x)$ we have

$$\lim_{k\to\infty} \kappa_i A_k(x) = \kappa_i(xI - D_Q) = 1$$

for every i, since the spectral data (both the eigenvectors and the eigenvalues) are at least locally continuous [71]. This concludes the proof. $\square$

The above theorem implicitly states another interesting result. Since $A_k(x) = xI - D_Q + R_k$ and $R_k \to 0$ as $k \to \infty$ we have that if we know good approximations for eigenvalues and eigenvectors of $P(x)$ we can probably compute a linearization $A_k(x)$ such that R_k is small in norm. This makes it very advantageous to apply Gerschgorin's theorem to obtain bounds on the eigenvalues of $P(x)$. The version of the Gerschgorin's theorem used here is the one of Theorem 1.5.3. See [27] for a comprehensive analysis of this tool.

Remark 2.5.5. Since the eigenvalues of A and A^t are the same Theorem 1.5.3 can be equivalently stated summing on rows or columns of the matrix A. In the following we will freely swap between the two options.

With this tool we can give the following result that relies on the eigenvalues of the linearization $A_k(x)$. Theorem 2.5.4 guarantees that the following result gives good bounds for $k \to \infty$, since the radii of the inclusion discs tend to 0.

Theorem 2.5.6. *Let $P(x)$ be a monic matrix polynomial and $Q(x)$ a monic nearby polynomial with the usual eigenvalue and eigenvector matrices of the companion form $xF_1^Q - F_0^Q$ equal to $\hat{W}_Q$, D_Q and $\hat{V}_Q$. We then have that if $D_Q = \mathrm{diag}(\lambda_1, \dots, \lambda_{nm})$ the eigenvalues of $P(x)$ are contained in the union of the discs*

$$S := \bigcup_{i=1}^{mn} B_i, \qquad B_i := \left\{ z \in \mathbb{C} \mid |z - \lambda_i| \leqslant \|P(\lambda_i)v_i\| \cdot \sum_{j=1}^{n} \|u_j^t\| \right\}$$

where u_j and V_i are left and right eigenvectors of $Q(x)$ normalized according to Theorem 2.5.3. Moreover, every connected component of the union made up of k discs contain exactly k eigenvalues (counted with multiplicity).

Proof. The proof is a straightforward application of Gerschgorin's Theorem 1.5.3 to the linearization $A_k(x)$ built according to Theorem 2.5.3. We note that such a linearization can be written in the form

$$A_k(x) = xI - D_Q + R_k, \qquad R_k = U \begin{bmatrix} P(\lambda_1)v_1 & \dots & P(\lambda_{mn})v_{mn} \end{bmatrix}$$

where U is the matrix with the left eigenvectors u_i^t of $Q(x)$ as rows and v_i are the right eigenvectors. The statements follows by applying Gerschgorin's Theorem and noting that diagonal elements are of the form $u_i^t P(\lambda_i)v_i - \lambda_i$ and offdiagonal elements of column i are of the form $u_j^t P(\lambda_i)v_i$. This yields the inclusion discs

$$B_i := \left\{ z \in \mathbb{C} \mid |z + u_i^t P(\lambda_i)v_i - \lambda_i| \leqslant \sum_{j \neq i} |u_j^t P(\lambda_i)v_i| \right\}$$

We can bound the radius and recenter the discs, finding a set of bigger inclusion discs, by writing

$$B_i \subseteq \left\{ z \in \mathbb{C} \mid |z - \lambda_i| \leqslant \|P(\lambda_i)v_i\| \cdot \sum_{j=1}^{n} \|u_j^t\| \right\}$$

which gives the thesis. □

We show here that this framework actually provides a generalization of the construction of Theorem 2.2.5. In fact, we have the following.

Theorem 2.5.7. *Let $A(x)$ be the secular linearization for a monic matrix polynomial $P(x)$ relative to the matrix polynomial $Q(x) = \prod_{j=1}^{n}(x - b_j)I_m$. Then*

$$A(x) = xI - \begin{bmatrix} b_1 I_m & & \\ & \ddots & \\ & & b_n I_m \end{bmatrix} + \begin{bmatrix} I_m \\ \vdots \\ I_m \end{bmatrix} \begin{bmatrix} W_1 & \dots & W_n \end{bmatrix}, \quad W_j = \frac{P(b_j)}{\prod_{i \neq j}(b_i - b_j)}$$

that is, the secular linearization for $P(x)$ relative to the nodes $b_1, \dots, b_n$.

Proof. It is evident that the eigenvalues of $Q(x)$ are semisimple and their left eigenvectors are e_i^t so they are exactly the rows of the matrix $(e \otimes I_m)$, that is, the left factor of the outer product defining the low rank term of $A(x)$. Each column of the right term, in view of Theorem 2.5.3 is equal to $P(\lambda_i)v_i$ but since if we look at one of the blocks W_j the vectors v_i are just the columns of the identity scaled so that $u_i^t Q'(\lambda_i)v_i = 1$ and $\lambda_i = b_j$ we have $W_j = \alpha_j P(b_j)$ where α_j is $\frac{1}{e_j^t Q'(b_j) e_j}$. This implies that $\alpha_j = \frac{1}{\prod_{i \neq j}(b_i - b_j)}$, which gives the thesis. □

2.5.3 *Handling the non-monic case*

In this section we show how to extend the previous results when the polynomial $P(x)$ is non-monic. In particular, some care is needed when the leading coefficient P_n is not invertible and so infinity eigenvalues appear.

To better represent this scenario we modify the univariate polynomial $P(x) = \sum_{i=0}^{n} P_i x^i$ and we consider the bivariate homogeneous polynomial

$$P^h(x, y) = \sum_{i=0}^{n} P_i x^i y^{n-i}.$$

Here $P^h(x, y)$ can be thought as a polynomial on the projective space $\mathbb{P}^1(\mathbb{C})$, so that we can evaluate it also at the infinity. In particular, we have

$$P(x) = P(x, 1), \qquad P(\infty) := P(0, 1).$$

We can define what eigenvalues and eigenvectors are for a homogeneous bivariate polynomial. We do so by requiring that they coincide with the classical notion of eigenvalues and eigenvectors whenever we consider the dehomogenized polynomial $P^h(x, 1) = P(x)$.

Definition 2.5.8. We say that the projective point $(\lambda, \mu) \in \mathbb{P}^1(\mathbb{C})$ is an *eigenvalue* for $P^h(x, y)$ if the matrix $P^h(\lambda, \mu)$ is singular. Moreover, we say that a vector v is a *right eigenvector* corresponding to (λ, μ) if $P^h(\lambda, \mu)v = 0$. In the same way we say that u is a *left eigenvector* if $u^t P^h(\lambda, \mu) = 0$.

We can also define the concept of linearization and of equivalences of matrix polynomials in this context. We have the following.

Definition 2.5.9. We say that two homogeneous bivariate matrix polynomials $P^h(x, y)$ and $Q^h(x, y)$ are *unimodularly equivalent* if, for every eigenvalue (λ, μ) of $P^h(x, y)$, there exist a neighborhood $\mathbb{P}^1(\mathbb{C}) \supseteq U \in (\lambda, \mu)$ and two unimodular analytic functions $E(x, y)$ and $F(x, y)$ defined on U such that

$$E(x, y)P^h(x, y)F(x, y) = Q^h(x, y).$$

In this case we write $P^h(x, y) \sim Q^h(x, y)$.

As in the univariate case, we can also define the extended unimodular equivalence.

Definition 2.5.10. We say that two homogeneous bivariante matrix polynomials $P^h(x, y)$ and $Q^h(x, y)$ are *extended unimodularly equivalent* if there exist r and s such that $\mathrm{diag}(I_r, P^h)$ is unimodularly equivalent to $\mathrm{diag}(I_s, Q^h)$. In this case we write $P^h \asymp Q^h$.

The reader might wonder why we are using the symbol of spectral equivalence here and not the one that we have used for extended unimodular equivalence in the univariate case. This fact is motivated by the following.

Theorem 2.5.11. *Let $P^h(x, y) \asymp Q^h(x, y)$ be two extended unimodularly equivalent homogeneous bivariate matrix polynomials. Then the univariate polynomials $P(x) = P^h(x, 1)$ and $Q(x) = Q^h(x, 1)$ are spectrally equivalent. Moreover, the converse is also true: if $P(x)$ and $Q(x)$ are spectrally equivalent matrix polynomials then their homogeneous version $P^h(x, y) := y^{\deg P} P(xy^{-1})$ and $Q^h(x, y) := y^{\deg Q} Q(xy^{-1})$ are extended unimodularly equivalent.*

Before proving the Theorem, we report the following result that can be found in [2].

Theorem 2.5.12. *If, for any eigenvalue λ of $P(x)$ and $Q(x)$, there exist a neighborhood of $U_\lambda \ni \lambda$ and two invertible analytic functions $E(x)$ and $F(x)$ defined on it such that $P(x) = E(x)Q(x)F(x)$ then $P(x)$ and $Q(x)$ are unimodularly equivalent.*

We are now ready to give the proof of Theorem 2.5.11

Proof of Theorem 2.5.11. We assume that $r = s = 0$. If this is not the case it is sufficient to replace $P^h(x, y)$ with $\mathrm{diag}(I_r, P^h(x, y))$ and $Q^h(x, y)$ with $\mathrm{diag}(I_s, Q^h(x, y))$. We first prove that if $P^h(x, y) \asymp Q^h(x, y)$ then $P(x) \asymp Q(x)$. We know that for every finite eigenvalue of $P^h(x, y)$ there exist two unimodular analytic matrix functions such that $E(x, 1)P^h(x, 1)F(x, 1) = Q^h(x, 1)$ holds locally and this implies that $E(x)P(x)F(x) = Q(x)$ in the same neighborhood. We make use of Theorem 2.5.12 and conclude that $P(x) \sim Q(x)$. Notice now that $P^\#(x) = P^h(1, x)$ and $Q^\#(x) = Q^h(1, x)$. Since the same argument of above can be applied verbatim we have $P^\#(x) \sim Q^\#(x)$. This implies that $P(x) \asymp Q(x)$, proving the first statement. For the other direction we assume that $P^h(1, x) = P^\#(x) \sim Q^\#(x) =$

$Q^h(1,x)$ and $P^h(x,1) = P(x) \sim Q(x) = Q^h(x,1)$ so that there exist unimodular matrices $E_1(x), F_1(x), E_2(x), F_2(x)$ such that

$$E_1(x)P^h(x,1)F_1(x) = Q^h(x,1), \qquad E_2(x)P^h(1,x)F_2(x) = Q^h(1,x).$$

Let (x,y) be an eigenvalue of $P^h(x,y)$ and assume that $x \neq 0$. Then we can rewrite $(x,y) = (1, y/x)$. In particular we have

$$E_2\left(\frac{y}{x}\right) P^h\left(1, \frac{y}{x}\right) F_2\left(\frac{y}{x}\right) = Q^h\left(1, \frac{y}{x}\right)$$

Since $E_2(x)$ and $F_2(x)$ are invertible everywhere the same holds for their evaluation in $\frac{y}{x}$ in a neighborhood of (x,y) such that x does not assume the value 0. Since this holds for every eigenvalue (x,y) and the same can be done when $y \neq 0$ by considering $E_1(x)$ and $F_1(x)$, we have the thesis. □

Definition 2.5.13. We say that a bivariate homogeneous polynomial $L^h(x,y)$ is a *linearization* for $P^h(x,y)$ if $L^h(x,y) \asymp P^h(x,y)$.

Clearly if $L^h(x,y)$ is a linearization for a matrix polynomial $P^h(x,y)$ then $L(x) = L^h(x,1)$ is a strong linearization for $P(x) = P^h(x,1)$. The converse is also true, so a strong linearization can be homogenized to become a homogeneous bivariate linearization.

The previous result implies, in particular, that the Frobenius linearization for a matrix polynomial, being a strong linearization, is also a homogeneous linearization for $P^h(x,y)$. We denote $xF_1^P - yF_0^P$ the linear Frobenius pencil associated with $P^h(x,y)$, as in the previous section.

We prove here a bivariate version of the diagonalization for pencils.

Theorem 2.5.14. *Let $xA - yB$ a homogeneous bivariate regular pencil with semisimple eigenvalues. Then there exist two invertible matrices W and V such that*

$$W^{-1}(xA - yB)V = xD_y - yD_x$$

where D_y and D_x are diagonal matrices such that the pairs (x_i, y_i) with their diagonal elements contain the eigenvalues of $xA - yB$. Moreover, the columns of V and the rows of W^{-1} contain the right and left eigenvectors of the pencil.

Proof. This proof is a generalization of the classical statement for eigenvalues of matrices, that we report here for completeness. First of all, we show that two eigenspaces V_1 an V_2[1] corresponding to two different eigenvalues (x_1, y_1) and (x_2, y_2) are necessarily independent, i.e., we can not find two nonzero vectors γ_1 and γ_2 such that $V_1\gamma_1 = V_2\gamma_2$. Assume, by contradiction, that this is the case. Since we know that

$$x_1 A V_1 = y_1 B V_1, \qquad x_2 A V_2 = y_2 B V_2$$

we can right multiply by γ_1 and γ_2 and we obtain $x_1 At = y_1 Bt$ and $x_2 At = y_2 Bt$ where we have set $t = V_1\gamma_1 = V_2\gamma_2$. We know that either $At \neq 0$ or $Bt \neq 0$, otherwise the pencil would not be regular. We assume that $At \neq 0$. Moreover, one of y_1 and y_2 must not be zero, since

[1] Here we identify an eigenspace corresponding to an eigenvalue (x,y) with a matrix containing the vectors of one of its basis stacked as columns. We have, in particular, that $xAV = yBV$.

the eigenvalues are different. Here we assume that $y_2 \neq 0$, but in the other case it is sufficient to swap the indices. We have

$$x_1 At = y_1 Bt = \frac{y_1}{y_2} y_2 Bt = \frac{x_2 y_1}{y_2} At$$

so that $x_1 y_2 = x_2 y_1$ that implies $(x_1, y_1) = (x_2, y_2)$ (in the projective sense) which leads to a contradiction. This proves that eigenspaces correspondent to different eigenvalues are independent. If the eigenvalues are semisimple the sum of the dimensions of the eigenspaces is equal to n, giving the first part of the thesis. We can then build the matrix W by noting that if $xAv = yBv$ we can define

$$We_i := \frac{Av}{y} = \frac{Bv}{x}.$$

While it might happen that x or y are zero, they cannot be zero at the same time and so at least one of the above is well-defined. Moreover, we can use once again the regularity of the pencil to claim that $We_i \neq 0$. With this definition we have

$$AV = WD_y, \qquad BV = WD_x.$$

Assume now, by contradiction, that there exists a vector $\gamma \neq 0$ such that $W\gamma = 0$. Since the pencil is regular we can find α and β such that $\alpha D_x - \beta D_y$ is invertible and (α, β) is not an eigenvalue of the pencil. We can then write $\hat{\gamma} := (\alpha D_y - \beta D_x)^{-1}\gamma$ so that

$$(\alpha A - \beta B)V\hat{\gamma} = W(\alpha D_y - \beta D_x)\hat{\gamma} = W\gamma = 0$$

which is a contradiction since (α, β) is not in the spectrum of $xA - yB$. As a last step, we have already verified that V contains the eigenvectors as columns, and we can see that W^{-1} contains the eigenvectors as rows by noting that the left eigenvectors of $xD_y - yD_x$ are e_i^t and so from

$$W^{-1}(xA - yB)V = xD_y - yD_x$$

we derive that the left eigenvectors of $xA - B$ are $e_i^t W^{-1}$. This completes the proof. □

We can now characterize the structure of the eigenvectors of the bivariate Frobenius pencil $xF_1^P - yF_0^P$.

Theorem 2.5.15. *Let $L(x) = xF_1^P - yF_0^P$ be the homogeneous Frobenius linearization associated with the polynomial $P^h(x, y)$, so that*

$$xF_1^P - yF_0^P = x \begin{bmatrix} P_n & & & \\ & I & & \\ & & \ddots & \\ & & & I \end{bmatrix} - y \begin{bmatrix} -P_{n-1} & -P_{n-2} & \dots & -P_0 \\ I & & & \\ & \ddots & & \\ & & I & \end{bmatrix}.$$

The right eigenvectors of $L(x)$ are given by

$$\hat{v}_i = \begin{bmatrix} x_i^{n-1} v_i \\ x_i^{n-2} y_i v_i \\ \vdots \\ y_i^{n-1} v_i \end{bmatrix}$$

where v_i is a right eigenvector of $P(x)$ corresponding to (x_i, y_i) and the left ones can be equivalently written as

$$\hat{u}_i = \begin{bmatrix} u_i \\ (\frac{x_i}{y_i}P_n^t + P_{n-1}^t)u_i \\ \vdots \\ ((\frac{x_i}{y_i})^{n-1}P_n^t + \ldots + \frac{x_i}{y_i}P_1^t)u_i \end{bmatrix} = \begin{bmatrix} u_i \\ -((\frac{y_i}{x_i})^{n-1}P_0^t + \ldots + \frac{y_i}{x_i}P_{n-2}^t)u_i \\ \vdots \\ -\frac{y_i}{x_i}P_0^t u_i \end{bmatrix}$$

where v_i is such that $u_i^t P^h(x_i, y_i) = 0$. Moreover, the first form is defined whenever $y_i \neq 0$ and the latter when $x_i \neq 0$.

Proof. We start to analyze the structure of the right eigenvectors. We have that if $\hat{v}$ is a right eigenvector for $xF_1^P - yF_0^P$ then we can partition it in blocks of size m as $\hat{v}^t = \begin{bmatrix} v_1^t & \cdots & v_n^t \end{bmatrix}$. The condition $(xF_1^P - yF_0^P)\hat{v} = 0$ can be rephrased as the system

$$\begin{cases} xP_n v_1 + y\sum_{j=0}^{n-1} P_j v_{n-j} = 0 \\ xv_j + yP_{n-j}v_1 - xv_{j-1} = 0 \qquad \text{for } j = 2, \ldots, n \end{cases}.$$

It is easy to see that $v_j = x_i^{n-j} y_i^{j-1} v$ where v is a right eigenvector for $P^h(x_i, y_i)$ is the unique possible solution (up to scalar multiplication). We can state similar conditions for the left eigenvector $\hat{u}$. Partitioning $\hat{u}^t = [u_1^t \; \cdots u_n^t]$, we can write

$$\begin{cases} u_1^t(xP_n + yP_{n-1}) = yu_2^t \\ u_j^t x + yu_1^t P_{n-j} = yu_{j+1}^t \qquad \text{for } j = 2, \ldots, n-1 \\ xu_n^t = -yu_1^t P_0 \end{cases}.$$

We set $u_1 := u$ and then we can follow two different paths of back-substitution. Either we assume that $y \neq 0$ and we start to compute the elements from u_2 to u_n or we assume that $x \neq 0$ so that we have $u_n^t = -\frac{y}{x}u^t P_0$ and we can compute the other elements from u_{n-1} to u_2. We obtain the following representations for $\hat{u}$:

$$\hat{u} = \begin{bmatrix} u \\ (\frac{x}{y}P_n^t + P_{n-1}^t)u \\ \vdots \\ \left((\frac{x}{y})^{n-1}P_n^t + \ldots + \frac{x}{y}P_1^t\right)u \end{bmatrix} = \begin{bmatrix} u \\ -\left((\frac{y}{x})^{n-1}P_0^t + \ldots + \frac{y}{x}P_{n-2}^t\right)u \\ \vdots \\ -\frac{y}{x}P_0^t u \end{bmatrix}$$

where u has to be a left eigenvector of $P^h(x, y)$. Writing this for an eigenvalue (x_i, y_i) and its corresponding eigenvector yields the thesis. □

We can now generalize the results obtained in Theorem 2.5.3 in order to be able to compute the secular linearization of $P(x)$ relative to $Q(x)$ also in the non-monic case.

Definition 2.5.16. We say that a matrix polynomial $A(x)$ is a *secular linearization relative to* $Q(x)$ for the matrix polynomial $P(x)$ if the homogeneous matrix polynomial $Q^h(x, y)$ is semisimple and V, D_x, D_y and W are such that $W^{-1}(xF_1^Q - yF_0^Q)V = xD_y - yD_x$ with D_x and D_y diagonal matrices,

$$A^h(x, y) = W^{-1}(xF_1^P - yF_0^P)V = xD_y - yD_x + yR.$$

and $A(x) = A^h(x, 1)$.

Lemma 2.5.17. *Let $A(x)$ be the secular linearization of Definition 2.5.16 for a matrix polynomial $P(x)$. Then $A(x)$ has the form*

$$A(x) = xD_y - D_x + UW^t$$

where U is a $nm \times m$ matrix whose rows are the left eigenvectors of $Q^h(x,y)$ and W is a $nm \times m$ matrix such that the columns of W^t are of the form $P^h(x_i, y_i)v_i$ where (x_i, y_i) is an eigenvalue of $Q^h(x,y)$ and v_i is a right eigenvector of $Q^h(x,y)$. In particular, if we set $R := UW^t$ we have

$$R = (r_{ij}), \qquad r_{ij} = \frac{u_i^t P^h(x_j, y_j)v_j}{u_j^t \frac{\partial}{\partial x} Q^h(x,y)v_j} = -\frac{x_j}{y_j} \frac{u_i^t P^h(x_j, y_j)v_j}{u_j^t \frac{\partial}{\partial y} Q^h(x,y)v_j}.$$

where u_i and v_j are left and right eigenvectors relative to (x_i, y_i) and (x_j, y_j), respectively, and $y_j \neq 0$. Moreover, an explicit formula is available when $y_j = 0$ and is here reported in Equation (2.14)

Proof. We prove the lemma by following steps similar to the one of Theorem 2.5.3. We have

$$A^h(x,y) = \hat{W}_Q^{-1}(xF_1^P - yF_0^P)\hat{V}_Q$$

where $\hat{W}$ and $\hat{V}$ are the usual matrices built following the procedure of Theorem 2.5.14. We can rewrite the above as

$$A^h(x,y) = \hat{W}_Q^{-1}(xF_1^Q - yF_0^Q)\hat{V}_Q + \hat{W}_Q^{-1}(yF_0^Q - yF_0^P)\hat{V}_Q = xD_y - yD_y + yR$$

where R has rank at most m. We can compute the elements of R by writing

$$R_{ij} = e_i^t \hat{W}_Q^{-1} \begin{bmatrix} Q_{n-1} - P_{n-1} & \dots & Q_0 - P_0 \\ & & \\ & & \end{bmatrix} \hat{V}_Q e_j$$

The characterization of right and left eigenvectors given in Theorem 2.5.15 tell us that, whenever $y_j \neq 0$ we can write

$$R_{ij} = u_i^t \left(\sum_{k<n} P_k x_j^k y_j^{n-k-1} - Q_k x_j^k y_k^{n-k-1} \right) v_j = u_i^t \frac{1}{y_j} P^h(x_j, y_j) v_j \tag{2.13}$$

where u_i^t and v_j are left and right eigenvectors of $P^h(x,y)$ relative (x_j, y_j) and the missing term of degree n has been added on both sides. We know that, by construction, these eigenvectors have to be normalized so that both the conditions

$$u_j^t F_1^Q v_j = y_j, \qquad u_j^t F_0^Q v_j = x_j$$

are satisfied. Expanding these conditions using the two different representations for u_j we get that

$$\frac{1}{y_j} \frac{\partial}{\partial x} u_j^t P^h(x_j, y_j) v_j = 1, \qquad -\frac{1}{x_j} \frac{\partial}{\partial y} u_j^t P^h(x_j, y_j) v_j = 1$$

which are computable in the cases where $y_j \neq 0$ and $x_j \neq 0$, respectively. Enforcing these conditions by scaling the right eigenvector leads to the following formulas for the element in position (i, j) of R:

$$R_{ij} = \frac{u_i^t P^h(x_j, y_j)v_j}{\frac{\partial}{\partial x} u_j^t Q^h(x_j, y_j)v_j} = -\frac{x_j}{y_j} \frac{u_i^t P^h(x_j, y_j)v_j}{\frac{\partial}{\partial y} u_j^t Q^h(x_j, y_j)v_j}.$$

Unfortunately none of the above formulas are usable int the case where $y_j = 0$. We can make an explicit computation for that particular case relying on the first part of Equation (2.13) and we obtain

$$R_{ij} = \frac{u_i^t x_j \left(\frac{\partial}{\partial y} Q^h(x_j, y_j) - \frac{\partial}{\partial y} P^h(x_j, y_j) \right) v_j}{u_j^t \frac{\partial}{\partial y} Q^h(x_j, y_j) v_j}, \qquad \text{if } y_j = 0. \tag{2.14}$$

$\square$

Remark 2.5.18. Notice that these formulas for the computation of R_{ij} can be rephrased in terms of $P(x)$ and $P^{\#}(x)$, and they end up being more readable. In particular, we have

$$P^h(x_j, y_j) = P(\lambda_j) = P^{\#}(\frac{1}{\lambda_j}), \qquad \lambda_j = \frac{x_j}{y_j}$$

whenever we can $y_j \neq 0$ or $x_j \neq 0$, respectively. In particular, we can choose to evaluate a polynomial at a point that is always included inside the unit circle, thus improving the Horner evaluation by possibly avoiding overflows.

3 QUASISEPARABLE MATRICES

3.1 GENERAL FRAMEWORK AND DEFINITIONS

Exploiting the structure of matrices to speed up computations has been a recurring theme in numerical analysis. Many papers address modified algorithms that are able to exploit the structures of particular problems. In many cases these structures arise in the form of sparse matrices, i.e., with many entries equal to zero, and often the non-zero elements are arranged near the diagonal, so that the resulting matrices are banded. A classical example of this structure are tridiagonal matrices. A matrix T is said to be tridiagonal if its elements t_{ij} are zero whenever $|i-j| > 1$. In general, when a matrix A is such that $|i-j| > k \implies a_{ij} = 0$ we say that it is of bandwidth k.

Several algorithms are available for tridiagonal and, more generally, banded matrices in the literature. The popular numerical library LAPACK also has a particular format for efficient storage of banded matrices [4].

Recently there has been much interest on a more general kind of structure, the so-called *quasiseparable* structure. See for example [47], [49], [50], [89], [90] and [26]. Intuitively, quasiseparable matrices are a generalization of banded matrices where the property of being 0 has been replaced with the one of being low-rank.

3.1.1 *Defining quasiseparable structures*

We give here the formal definitions needed to make the above concept more precise and also to avoid ambiguities. In fact, as often happens when developing a new theory, similar naming are used for different things by different authors. In this work we mostly adhere to the naming scheme used in [89] and [90].

Definition 3.1.1. Given a matrix A we define its *lower* and *upper quasiseparability ranks* as the numbers

$$\operatorname{lr}(A) := \max_{i=1,\ldots,n-1} A[i+1:n, 1:i], \qquad \operatorname{ur}(A) := \max_{i=1,\ldots,n-1} A[1:i, i+1:n],$$

respectively.

The idea of this definition is that whenever we select a lower (resp. upper) offdiagonal submatrix from the matrix A its rank must be bounded by $\operatorname{lr}(A)$ (resp. $\operatorname{ur}(A)$). Figure 3.1 reports a graphical description of the structure.

In the following we often use the notation QS ranks to refer to these two numbers, the lower and upper quasiseparability rank of a matrix A.

Definition 3.1.2. A matrix A is said to be *lower-quasiseparable* (resp. *upper-quasiseparable*) *of rank* k if every submatrix contained in the strictly lower (resp. upper) triangular part of A has rank at most k, that is, if $\operatorname{lr}(A) \leqslant k$ (resp. $\operatorname{ur}(A) \leqslant k$). If A is k_l-lower quasiseparable and k_u-upper quasiseparable we say that A is (k_l, k_u)*-quasiseparable*. If $k = k_l = k_u$ we say that A is k-*quasiseparable*.

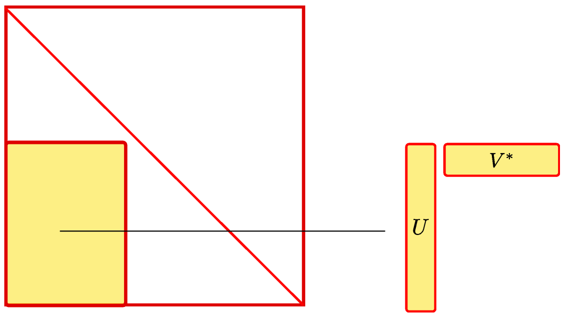


Figure 3.1: Graphics description of the quasiseparable structure. Here in the picture the lower quasiseparability structure is displayed.

Remark 3.1.3. Notice that if a matrix is (k_l, k_u)-quasiseparable then it is also (j_l, j_u)-quasiseparable for any $j_l > k_l$ and $j_u > k_u$.

In the following we will mostly deal with the case where the underlying field is $\mathbb{C}$. For this reason, we denote with QSH_k^n the set of $n \times n$ Hermitian k-quasiseparable matrices with entries in $\mathbb{C}$. We will sometimes omit the superscript n and simply write QSH_k when the dimension is clear from the context. In general, we say that A is quasiseparable if it is (k_l, k_u)-quasiseparable for some nontrivial k_l, k_u.

Remark 3.1.4. It is clear from the definition that banded matrices with bandwidth k are also quasiseparable matrices of rank k. In fact, every off diagonal matrix has at most a corner with non-zero elements and so no more than k rows (and columns) different from zero. On the other hand, while it is immediate to design a method to store and operate on banded matrices efficiently, the same is not true for quasiseparable matrices. Even representing them in an efficient and numerically stable way is a non-trivial issue.

Before introducing the different techniques available in the literature for the representation of quasiseparable matrices we shall point out some of the features that they have. Here we also provide a sketch of proof. Many more details can be found in [89].

Theorem 3.1.5. *Let A be a (k_l, k_u)-quasiseparable matrix and B a (j_l, j_u)-quasiseparable matrix. Then the following statements are true:*

(i) If A is invertible then also A^{-1} is quasiseparable of ranks (k_l, k_u).

(ii) $A + B$ is a $(k_l + j_l, k_u + j_u)$-quasiseparable matrix.

(iii) AB is a $(k_l + j_l, k_u + j_u)$-quasiseparable matrix.

Proof. The property (i) is a direct consequence of the Nullity Theorem [65], that states that the nullity of any off-diagonal block in A^{-1} is the same of the one of the same block in A (and so also the ranks coincide).

The proof of (ii) is also immediate since every off-diagonal submatrix of the sum is the sum of the relative submatrices of A and B. The same argument can also be applied to AB by performing the block multiplication of the two matrices. $\square$

3.1.2 *The relations with linearizations*

We are interested in this kind of structures because it can be found in the coefficients of most linearizations of matrix polynomials. Consider for example the Frobenius linearization for a matrix polynomial $P(x) = \sum_{i=0}^{d} P_i x^i$:

$$P(x) \asymp x \begin{bmatrix} P_d & & & \\ & I_m & & \\ & & \ddots & \\ & & & I_m \end{bmatrix} - \begin{bmatrix} -P_{d-1} & -P_{d-2} & \dots & -P_0 \\ I_m & & & \\ & \ddots & & \\ & & I_m & \end{bmatrix}.$$

Every offdiagonal submatrix of both coefficients has the rank bounded by m. It is important to highlight that the structure in these matrices depends on the degree d of the polynomial. The higher the degree (with respect to the size of the coefficients) the higher the structure available. As an example, if the degree is d and the size $m = 1$, we have the linearization of a scalar polynomial and the quasiseparability rank is 1. If the degree is $d = 1$ the linearization coincide with the polynomial and so no structure is present (at least in general).

Another example of quasiseparable linearizations (and ℓ-ifications) is given by the secular linearization of Theorem 2.2.5. When $\ell = 1$ and so $A(x)$ is a linearization it has the form of a (block) diagonal matrix pencil plus a low rank matrix, and so it is quasiseparable with rank at most $2m$, where m is the size of the blocks. In the even more particular case where the $B_i(x)$ are monic and diagonal we have that $A(x) = xI + D - L$ with L of rank at most m. This implies that $A := D - L$ is m-quasiseparable.

We recall that the usual first step for eigenvalue approximation algorithms is the Hessenberg reduction (or the Hessenberg triangular reduction in case of pencils). This is due to many interesting properties that characterize this structure.

- The Hessenberg structure is preserved under QR iterations (and the Hessenberg-triangular is maintained under QZ). This is interesting because it reduces the cost of an iteration from $O(n^3)$ to $O(n^2)$, lowering the total cost of the method.
- Evaluating the characteristic polynomial of a matrix in Hessenberg form is cheaper than for a full matrix. This is interesting when trying to develop an approximation algorithm using functional iterations such as the secsolve algorithm implemented in `MPSolve` and introduced in Section 1.5 [22].

3.1.3 *Some useful notation*

In the following we will make often use of the MATLAB notation, that is of operators inherited by the commands of MATLAB. More precisely, we will sometimes write $A[i_1 : i_2, j_1 : j_2]$ to mean the submatrix of A obtained by taking the rows with indices from i_1 to i_2 and the columns with indices from j_1 to j_2.

In a similar way we define the operators $\mathrm{tril}(\cdot,\cdot)$ and $\mathrm{triu}(\cdot,\cdot)$, coherently with the corresponding MATLAB functions, such that $L = \mathrm{tril}(A, k)$, $U = \mathrm{triu}(A, k)$ where $A = (a_{i,j})$, $L = (\ell_{i,j})$, $U = (u_{i,j})$ and

$$\ell_{i,j} = \begin{cases} a_{i,j} & \text{if } i \geqslant j - k \\ 0 & \text{otherwise} \end{cases} \qquad u_{i,j} = \begin{cases} a_{i,j} & \text{if } i \leqslant j - k \\ 0 & \text{otherwise} \end{cases}.$$

3.2 REPRESENTATION OF QUASISEPARABLE MATRICES

In this section we briefly survey some possible representations of quasiseparable matrices. In the following we will consistently use k as the rank of the quasiseparable matrices that we want to represent and n as the number of rows and columns. We require the representations to have the following features:

- Use a small amount of storage. Typically, our request is to use only $O(nk)$, but we will also show a representation that uses $O(nk \log n)$.
- Allow for fast operations on the compressed forms. As an example, we want to be able to compute the product Av, where A is quasiseparable, in less than $O(n^2)$. Typically this cost will be $O(nk)$.
- Being able to obtain a quasiseparable representation of the sum and product of quasiseparable matrices.

As we will see, different representation will address the above points with various degrees of success. Each one has its own advantages. In this work we present the following three representations:

(i) The quasiseparable representation introduced by Eidelman and Gohberg. A survey of the available literature for this representation can be found in the books by Eidelman et al. [49, 50]. We will call this kind of representations *generator based representations*.

(ii) The so-called Hierarchical representation or $\mathcal{H}$-matrices introduced by Hackbush [66] and widely used in the context of PDEs (such as [6] and [7]). As we will see this kind of representation has an additional $\log n$ factor cost in many operations (that can be removed by relying on $\mathcal{H}^2$-matrices in some cases [67]) but also has many advantages such as being able to provide adaptive representation without the need of knowing the quasiseparable rank a priori.

(iii) The quasiseparable representation for 1-quasiseparable matrices of Van Barel et al. analyzed in [89] and [90] and its generalization to k-quasiseparable matrices that can be found for example in [88, 40]. We will call these kinds of representations *Givens–Vector representations*. Several papers on this topic are available in the literature by Van Barel and DelVaux including several results on the conservation of rank structures [36, 38, 37, 44, 39, 42, 43].

3.2.1 *Generators based representation*

Generators based representation are a natural extensions of the representation of low-rank matrices as outer products. If A is a matrix of rank k then there exist two $n \times k$ matrices U and V such that $A = UV^*$. Recalling this fact one might be tempted to represent quasiseparable matrices as

$$A = D + \mathrm{tril}(U_l V_l^*, -1) + \mathrm{triu}(U_u V_u^*, 1), \qquad U_l, V_l, U_u, V_u \in \mathbb{C}^{n \times k}. \tag{3.1}$$

Unfortunately, this is not always possible. We can say that, whenever A has such a representation then it is quasiseparable of order k but the converse is not true. A simple counterexample is given by tridiagonal matrices (or banded matrices for any order larger than 1): If we have

$$T = \begin{bmatrix} a_1 & b_1 & & \\ c_1 & \ddots & \ddots & \\ & \ddots & \ddots & b_{n-1} \\ & & c_{n-1} & a_n \end{bmatrix}, \qquad b_i, c_i \neq 0$$

then every submatrix in the lower (resp. upper) triangular part has at most 1 element different from zero, and so it has rank (at most) 1. Moreover, each of these submatrices can be represented as an outer product of two vectors, but we cannot find two vectors u and v such that $\mathrm{tril}(T, -1) = \mathrm{tril}(uv^*, -1)$. In fact, since $b_i \neq 0 \neq c_i$ we have $u_i \neq 0 \neq v_i$ (otherwise we will have zero elements on the row or column with index i). This implies that the lower part of T is dense, with no zero elements, that is not true.

Even when this is not the case it might not be a good idea to write quasiseparable matrices with a representation like (3.1) for numerical reasons. Consider the following example, where we have an "almost" diagonal matrix

$$\tilde{T} = \begin{bmatrix} 1 & \epsilon & \dots & \epsilon^{n-1} \\ \epsilon & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \epsilon \\ \epsilon^{n-1} & \dots & \epsilon & 1 \end{bmatrix}$$

with some ϵ small. This matrix is quasiseparable and it does have a representation like the one of Equation (3.1). The vectors u_l, v_l, u_u, v_u can be written explicitly as

$$u_l = u_u = \begin{bmatrix} \epsilon \\ \epsilon^2 \\ \vdots \\ \epsilon^{n-1} \end{bmatrix}, \qquad v_l = v_u = \begin{bmatrix} 1 \\ \epsilon^{-1} \\ \vdots \\ \epsilon^{-n+2} \end{bmatrix}.$$

We note that when ϵ is near 0, even if the norm of $\tilde{T}$ is small (approximately 1), the norm of the vectors v_l and v_u is not and goes like ϵ^{-n}. This might create some problems for the conditioning of the representation. In fact we can show that a small relative perturbation to v_l or v_u can create a much larger perturbation in the original matrix $\tilde{T}$. As an example, consider the vector γe_1, and suppose that we alter v_l by setting $\hat{v}_l = v_l + \epsilon^{-n+2}\gamma e_1$. Since the norm of v_l is at least ϵ^{-n+2} this perturbation is relatively of the order of γ. If we reconstruct the matrix $\tilde{T}$ by replacing v_l with $\hat{v}_l$ we obtain that the subdiagonal element in position $(2, 1)$ is now $\epsilon(1 + \epsilon^{-n+2}\gamma) \approx \gamma\epsilon^{-n+3}$ so we find a perturbation on the elements of $\tilde{T}$ of the order $O(\epsilon^{-(n-3)})$. This means that the error has grown exponentially in n, thus making the use of this representation not very well suited to this kind of matrices.

Given this introductory example, we want to find a strategy to solve the problem. A possible idea, that is the basis of the generators based representation, is to improve the representation of Equation (3.1) by adding another term that takes care of the relations between the entries of the matrix.

For simplicity we handle the case of symmetric or Hermitian matrices where we only have the need to represent one triangular part.

Definition 3.2.1. A *generators quasiseseparable representation* of a symmetric k-quasiseparable matrix A is a sequence (D, U, V, c_j) where

- D is a diagonal matrix equal to the diagonal of A.
- U and V are matrices of size $n \times k$.
- c_j is a sequence of $k \times k$ matrices for $j = 2, \ldots, n-1$.

and each strictly subdiagonal element of A is equal to

$$a_{ij} = u_i c_{i-1} \ldots c_{j+1} v_j, \quad u_i := e_i^t U, \ v_j^t := e_j^t V.$$

Notice that if we choose $c_j = I$ for every j we obtain a representation of the kind of Equation (3.1) so this representation is more general than the previous one. In particular, we can prove that every quasiseparable matrix has a representation of this kind (see [49]).

A very in-depth analysis of these tools can be found in [49, 50]. We refer to these publications for the details.

3.2.2 *Tracking the rank structure: the $t(\cdot)$ operator*

Even if the proposed representation for quasiseparable matrices in the form of Equation (3.1) is often undesirable from the numerical point of view, it is sometimes useful from the theoretical perspective. In fact, it will be widely used in order to prove rank properties of the partially reduced matrix when computing the Hessenberg form of a matrix A in Section 3.3.

For this reason, we introduce the operator $t(\cdot)$ that eases the use of this kind of representations for Hermitian matrices.

Definition 3.2.2. The *symmetrizing operator* $t(\cdot)$ is defined on the set of $n \times n$ matrices as follows:

$$t(A) = \mathrm{tril}(A, -1) + \mathrm{triu}(A^*, 1), \quad \forall A \in \mathbb{C}^{n \times n}.$$

Lemma 3.2.3. *The operator $t(\cdot)$ enjoys the following properties:*

(i) For any matrix A the matrix $t(A)$ has a zero diagonal.

(ii) For any matrix A the matrix $t(A)$ is Hermitian. Moreover, if A is already a Hermitian matrix then $t(A) = A - D$ where D is the diagonal of A.

(iii) The operator $t(\cdot)$ is linear on $\mathbb{R}$, that is, for any matrices A and B with the same size we have $t(A + B) = t(A) + t(B)$ and for any scalar $\lambda \in \mathbb{R}$ we have $t(\lambda A) = \lambda t(A)$.

(iv) Diagonal congruence can be taken in and out from the operator, that is, for any diagonal matrix D we have $t(DAD^*) = Dt(A)D^*$.

(v) For any block matrix with square diagonal blocks we have

$$t\left(\begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix}\right) = \begin{bmatrix} t(A_{1,1}) & A_{2,1}^* \\ A_{2,1} & t(A_{2,2}) \end{bmatrix}.$$

Proof. We will prove only (iv), since all the other properties can be easily proved from the definition. If D is diagonal we have that right and left multiplication of A are simply a scaling of the columns and rows by some factor. For this reason we have

$$\operatorname{tril}(DA, -1) = D \cdot \operatorname{tril}(A, -1), \qquad \operatorname{tril}(AD^*, -1) = \operatorname{tril}(A, -1) \cdot D^*.$$

This, in particular, implies that $Dt(A)D^*$ and $t(DAD^*)$ have the same strictly lower triangular part. Since also the diagonal of $Dt(A)D^*$ is equal to 0 we have that these two matrices coincide in the whole lower triangular part. Being both of them Hermitian, we have the thesis. □

3.2.3 *Hierarchically quasiseparable matrices*

We introduce another way of representing quasiseparable matrices based on a simple idea: if we split a quasiseparable matrix in 4 blocks

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix}$$

so that the $A_{1,1}$ and $A_{2,2}$ are square we obtain 4 matrices. Two of them are still quasiseparable (the diagonal blocks), the others are low-rank. We have already observed that it is always possible to represent low rank matrices as outer products UV^* so we can do this for $A_{2,1}$ and $A_{1,2}$. Then we can re-apply this procedure recursively on the diagonal blocks $A_{1,1}$ and $A_{2,2}$.

This idea has been exploited in [66] to construct hierarchical representations of quasiseparable matrices. The recursive subdivision can be arbitrary and tailored to the specific problem. In our case, where we have general quasiseparable matrices, we choose to divide the matrices into two (almost) equal parts. A pictorial example of the subdivision process is reported in Figure 3.2.

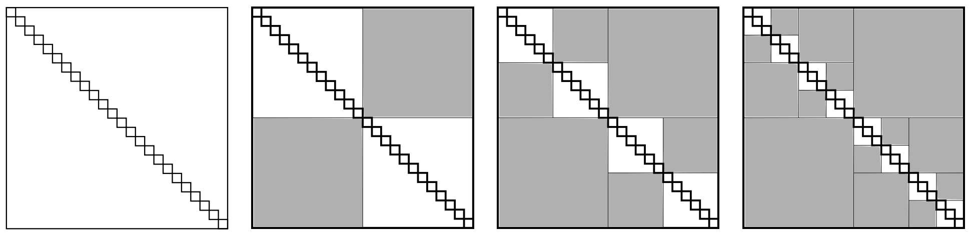


Figure 3.2: Hierarchical subdivision of a matrix used to construct the $\mathcal{H}$-matrix representation. Each gray block in the figures has low rank, while the white blocks are recursively represented using the hierarchical strategy up to an appropriate base case.

In the language of $\mathcal{H}$-matrices the low-rank blocks of Figure 3.2 are called *admissible* blocks. This name is used to mean the blocks that can be represented without the need of further subdivision.

Notice that, if we subdivide a sufficient number of times, the diagonal blocks will become "low-rank", in the sense of rank less than the quasiseparability k, since at some point they will be smaller than k. This guarantees that the recursion can be stopped.

Every $\mathcal{H}$-matrix can be represented as a set of admissible blocks and two trees, called *row* and *column cluster*, respectively, whose nodes are set of indices such that:

- The root of the tree is the set of indices of the matrix $\{1, \dots, n\}$.
- The children of each node are again set of indices that form a partition of the indices of their father.

These trees are called $\mathcal{H}$*-trees*. Given two $\mathcal{H}$-trees I and J we can build a *block* $\mathcal{H}$*-tree* $I \times J$ that is composed by the nodes $\{(i, j) \mid i \in I,\ j \in J,\ \text{depth}(i) = \text{depth}(j)\}$. The set of sons of the vertex (i, j) is given by the cartesian product of the sons of i and the sons of j.

An important concept in the framework of $\mathcal{H}$-matrices is the admissibility condition. We can associate a map $a : I \times J \longrightarrow \{0, 1\}$ to the block $\mathcal{H}$-tree $I \times J$ that takes the value 1 only on a subset of blocks. We call these blocks the *admissible* blocks of the representation.

These blocks correspond to the blocks of the matrix that are explicitly representable, either as a low rank or a full matrix.

In our setting we decided to use the same subdivision for the row indices and the column indices (as depicted in Figure 3.2) but, in general, different choices can be made. Moreover, we say that a block is admissible if and only if its row and column indices i and j are disjoint or if the size of the matrix is smaller than a given threshold. Figure 3.2 reports the admissible blocks of the first kind in gray.

Remark 3.2.4. This representation of quasiseparable matrices has a larger storage requirements than the generator based representation and the Givens–Vector ones (that will be presented in the next subsection). In fact, the representation of low rank admissible blocks at each level accounts for an $O(nk)$ storage but we need to go $\log_2 n$ levels deep, so the total storage need is about $O(nk \log_2 n)$. As we will see, this also reflects on computational costs. However, a variant of $\mathcal{H}$-matrices called $\mathcal{H}^2$-matrices are available [67] and introduce some relationships between the admissible blocks so that the storage can be made again $O(nk)$. In this work we will use $\mathcal{H}$-matrices because, being simpler, they allow a clearer exposition of the techniques used. Notice, however, that the ideas using $\mathcal{H}$-matrices can be reformulated in the new context of $\mathcal{H}^2$-matrices.

The main feature of this representation is the ability to perform matrix operations and compute hierarchical representation of the results without knowing the quasiseparable rank a priori. We refer to [26] for the details and we simply give a short example that shows how this can be done.

Consider two $\mathcal{H}$-matrices A and B, and suppose that they have both quasiseparable rank k. We know by the property of quasiseparable matrices that the sum $A + B$ will have QS rank (at most) $2k$. Here we show how to obtain a representation of $A + B$ as an $\mathcal{H}$-matrix. In order to perform the operation we need A and B to share the same subdivision, i.e., the same row and column cluster. For simplicity we will assume that the chosen is cluster is the binary subdivision cluster, where each set of indices is divided in the first and second half (rounded to the nearest integer). This is exactly the subdivision represented in Figure 3.2.

At the top level we have that A and B are divided in four submatrices. The diagonal blocks will be recursively subdivided, while the offdiagonal ones are low rank, so they admit a representation of the form UV^*. Let $U_A V_A^*$ and $U_B V_B^*$ be two low rank blocks in corresponding position of A and B. The corresponding block of $A + B$ will be $U_A V_A^* + U_B V_B^* = [U_A U_B] \cdot [V_A V_B]^*$ that provides a direct formula for the admissible blocks of the result. This approach can be applied recursively on the diagonal blocks and allows to obtain a quasiseparable representation of order $2k$ of $A + B$.

A reasonable question might be: what if $A + B$ is not only $2k$-quasiseparable but also j-quasiseparable for some $j < 2k$? Can we detect when this happens? The answer is yes, and this is in fact one of the most interesting features of this framework in our context.

Remark 3.2.5. If the resulting matrix $A + B$ is j-quasiseparable then any admissible block of its representation has at least one of the two factors computed in the above way of rank j.

This suggests an algorithm for this compression step:

- We compute the quasiseparable representation in $\mathcal{H}$-matrix form of $A + B$ with QS rank equal to $2k$.
- We compute a QR factorization of each factor of the form $QR = [U_A V_A]$ and $PS = [V_A V_B]$. So then we have
$$[U_A U_B] \cdot [V_A V_B]^* = QRS^*P^*.$$
- We compute a SVD factorization of RS^* (that is a $2k \times 2k$ matrix) and we rewrite $RS^* = \tilde{U}\tilde{V}^*$ by removing the columns and rows of the basis given by the SVD corresponding to zero singular values (or to singular values under a certain threshold, depending on the user's choice).
- We construct the updated factors for the representation of the admissible block as
$$[U_A U_B] \cdot [V_A V_B]^* = Q\tilde{U}(P\tilde{V})^*.$$

The above algorithm has a cost that is cubic in k but only linear in n, since it only requires a QR factorization of $n \times k$ matrices, that costs $O(nk^2)$, and an SVD of a $k \times k$ matrix that costs $O(k^3)$. This makes it very effective when the quasiseparability rank is not large.

Moreover, we can notice that the procedure can be adapted to work in an approximate way: we can choose to approximate the $k \times k$ matrix RS^* with an optimal low rank approximation given by the SVD. Given that Q and P are orthogonal, this will be also an optimal approximation (in the 2-norm sense) for $[U_A U_B] \cdot [V_A V_B]^*$. This allows to find a representation of an $\mathcal{H}$-matrix that approximates $A + B$ with a nearby quasiseparable matrix obtained by truncating small singular values. The truncation epsilon can be chosen arbitrarily and adaptively based on the needs of the user, and this makes the strategy very flexible.

Remark 3.2.6. We have written the compression strategy only for the summation case, but the same can be applied almost verbatim also for the multiplication (by performing block multiplication) and for the inversion. We refer to [26] for an in-depth analysis of these algorithms.

Remark 3.2.7. One of the advantages of the $\mathcal{H}$-matrices and $\mathcal{H}^2$-matrices framework is that there exists a very well-written implementation of these data structures, called HLib [25]. An open source clone (written by the same author) of this is also available on Github under the name of H2Lib [24].

3.2.4 *Handling Givens rotations*

In this subsection we present some preliminary tools before introducing the third kind of representation, called *Givens–Vector representation*. They have been introduced by Mastronardi, Van Barel, Vandebril et al. for 1-quasiseparable matrices and are analyzed in [89] and [90].

As the name suggests, these representations are based on the use of Givens rotations. To be more precise, it is not strictly necessary to consider Givens rotations: any construction that provides 2×2 orthogonal matrices that allow to put zeros in specific places can be used. However, in order to keep the exposition simple, we only refer to Givens rotations.

Given a vector $v = (v_i) \in \mathbb{C}^2$, denote by $G = G(v_1, v_2)$ a 2×2 orthogonal matrix

$$G = \begin{bmatrix} c & s \\ -\bar{s} & c \end{bmatrix}, \quad c \in \mathbb{R}, \quad |s|^2 + c^2 = 1,$$

such that $Gv = \alpha e_1$, where $|\alpha| = \|v\|_2$. Typically, when dealing with bigger matrices, we compute Givens rotations acting only on two consecutive rows, i.e., we consider matrices of the form $G_i = I_{i-1} \oplus G \oplus I_{n-i-1}$. We call also these matrices Givens rotations, in the sense that G_i acts on a matrix A as a rotation applied to the components i and $i+1$ of each column (or on each row, depending if we are left or right multiplying).

Definition 3.2.8. A tuple $\mathcal{G} = (G_i)_{i \in I}$ where G_i are Givens rotations and $(I, \leqslant)$ is some ordered index set is said a *sequence* of Givens rotations. We write $\mathcal{S}_G^n$ to mean the set of Givens sequences built by $n \times n$ Givens rotations. We also define $\prod_{i \in I} G_i$ the product in increasing order while $\prod_{i \in \mathrm{rev}(I)} G_i$ denotes the product in decreasing order with respect to the order defined on I. The following operations on $\mathcal{G}$ are introduced:

- $\mathcal{G}v := \prod_{i \in \mathrm{rev}(I)} G_i v$, for $v \in \mathbb{C}^n$;
- $\mathcal{G}^* v := \prod_{i \in I} G_i^* v$ for $v \in \mathbb{C}^n$;
- for $J \subseteq I$, with the induced order, we call $\mathcal{G}[J] := \left(G_j\right)_{j \in J}$ the *slice* of $\mathcal{G}$ on the indices J;
- for Givens sequences $\mathcal{G} = (G_i)_{i \in I}$, $\mathcal{G}' = \left(G_j'\right)_{j \in J}$, we define the product $\mathcal{G}\mathcal{G}'$ to be the sequence

  $$\mathcal{G}\mathcal{G}' := (E_i)_{i \in I \sqcup J}, \qquad E_i = \begin{cases} G_i \text{ if } i \in I \\ G_i' \text{ if } i \in J \end{cases}$$

  where $\sqcup$ is the disjoint union operator and where the order on $I \sqcup J$ is induced by the ones on I and J and by the agreement that $G_i < G_j'$ for every $i \in I, j \in J$.

The above definitions of the product between a sequence and a vector trivially extend to products between sequences and matrices. For instance we set $\mathcal{G}A := \prod_{i \in \mathrm{rev}(I)} G_i A$.

The sequences of rotations are the basic tool that we use to define Givens Vector representations (in short GV in the following). The definition given above is fairly general and, in our construction, we are interested only in some particular choices for the index sets I. In the cases where the index sets have this particular structure that we require, we give some more specific definitions. We begin by introducing the necessary tools for the representation of 1-quasiseparable matrices and then we extend them to k-quasiseparable matrices for $k > 1$.

Definition 3.2.9. We say that $\mathcal{G}$ is a 1-*sequence* of Givens rotations if $\mathcal{G} = (G_2, \dots, G_{n-1})$.

When $\mathcal{G}$ is not only a sequence, but also a 1-sequence, we can write the actions of $\mathcal{G}$ and $\mathcal{G}^*$ on a vector v in a much more explicit way.

- $\mathcal{G}v := G_{n-1} \dots G_2 v$, for $v \in \mathbb{C}^n$;

- $\mathcal{G}^* v := G_2^* \dots G_{n-1}^* v$, for $v \in \mathbb{C}^n$;
- $\mathcal{G}[i:j] := (G_i, \dots, G_j)$, for $2 \leqslant i < j \leqslant n-1$, is a *slice* of $\mathcal{G}$ from i to j.

Here and in the following we often use the notation $i : j$ to mean the ordered set of integers from i to j, i.e., formally,

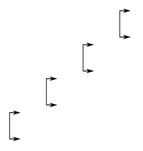
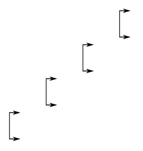
$$i : j := (I, \leqslant), \qquad I = \{i, i+1, \dots, j-1, j\}$$

where $\leqslant$ is the usual order on the integers. We also introduce the following notations:

- $\mathcal{G}[: j] := \mathcal{G}[2 : j]$, i.e., the subsequence of $\mathcal{G}$ with the elements up to the index j.
- $\mathcal{G}[i :] := \mathcal{G}[i : n-1]$, i.e., the subsequence of $\mathcal{G}$ with the elements with an index bigger or equal of i.

In the [88] and some other recent papers such as [5] a nice pictorial notation is used to denote these sequences of rotations. We use it here to make some of the following proofs much more understandable, even if we stick with the formal version of the sequences to verify our claims.

In these works the authors use the symbol to denote a rotation G_i. The index i is determined graphically by the vertical alignment of rotations. As a concrete example, consider the following pictorial representation of the action $\mathcal{G}v$ for a sequence of rotations with $n = 6$:

$$\mathcal{G} = (G_2, \dots, G_{n-1}) =$$

$$, \qquad \mathcal{G}v =$$

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{bmatrix}. \tag{3.2}$$

Here $\mathcal{G}v$ represents the product $G_{n-1} \dots G_2 v$ and the vertical and horizontal alignments of the symbol clarify both the order of the multiplications and the indices where they act.

The 1-sequences are a valid tool to describe quasiseparable matrices of QS rank 1. By joining k 1-sequences we obtain a tool that is useful to parametrize quasiseparable matrices of QS rank k. In order to formally define these objects we will need two indices (intuitively, one to index the vertical alignment of the rotation and the other to index the k-sequence that we have joined together). For this, we introduce the following definition.

Definition 3.2.10. The *standard Givens sequence order* on the set $\mathbb{N}^2$ is the ordering $\leqslant_G$ defined by

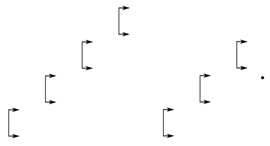
$$(i_1, j_1) \leqslant_G (i_2, j_2) \iff j_1 > j_2 \text{ or } (j_1 = j_2 \text{ and } i_1 \leqslant i_2).$$

With this order on $\mathbb{N}^2$ we can precisely define the concept of k-sequences of Givens rotation.

Definition 3.2.11. A sequence of Givens rotation $\mathcal{G} = (G_{i,j})_{(i,j) \in I}$ is a k-*sequence* if $I = \{(i,j) \in \mathbb{N}^2 \mid i = 2, \dots, n-1, j = 1, \dots, \min(i-1, k)\}$ with the order induced by $\leqslant_G$. With a slight abuse of notation we define the sequence $\mathcal{G}[i_1 : i_2] := (G_{i,j})_{(i,j) \in I'}$, $I' = \{(i,j) \in \mathbb{N}^2 \mid i = i_1, \dots, \min(i_2 + k, n-1), j = \max(1, i - i_2 + 1), \dots, \min(k, i - i_1 + 1), \quad 2 \leqslant i_1 < i_2 \leqslant n-1\}$ to be a *slice* of $\mathcal{G}$ from i_1 to i_2, where the ordering in I' is induced by the ordering $\leqslant_G$ valid on the parent set.

The definition above might seem a little bit involved but, once again, the pictorial representation of the rotations can help to grasp the concept in a more immediate way.

Recall that, intuitively, k-sequences are just k 1-sequences glued together, each with one rotation less than the previous one. For example, for $n = 6$ and $k = 2$ we have $\mathcal{G} = (G_{3,2}, G_{4,2}, G_{5,2}, G_{2,1}, G_{3,1}, G_{4,1}, G_{5,1})$ that is represented by

$\mathcal{G} =$  . (3.3)

Note that for every $i_1 \leqslant i_2 < i_3$, the slices of $\mathcal{G}$ can be factored in the following form:

$$\mathcal{G}[i_1 : i_3] = \mathcal{G}[i_2 + 1 : i_3]\mathcal{G}[i_1 : i_2], \qquad \mathcal{G}^*[i_1 : i_3] = \mathcal{G}^*[i_1 : i_2]\mathcal{G}^*[i_2 + 1 : i_3],$$

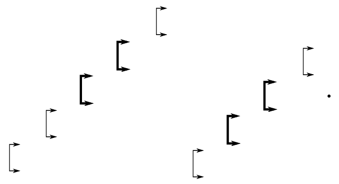
where, for notational simplicity, we set $\mathcal{G}^*[i_1 : i_2] = (\mathcal{G}[i_1 : i_2])^*$. This property is called *slicing of rotations*.

Note that the order $\leqslant_G$ is chosen so that $\mathcal{G}v$ coincides with the product of the rotations by v in the order induced by the above pictorial representation.

Remark 3.2.12. In principle we now have two conflicting definitions for 1-sequences, since both Definition 3.2.9 and Definition 3.2.11 with $k = 1$ can be used. Nevertheless, while they define two formally different objects, we can show that they are isomorphic. In fact, the set of rotations do have the same cardinality and the index sets are $I = \{2, \dots, n-1\}$ in the first case and $I = \{(2, 1), \dots, (n-1, 1)\}$ in the second one. These two sets are clearly isomorphic as ordered sets, and so the ambiguity can be removed.

Remark 3.2.13. It is worth noting that the choice of an ordering that induces the correct order in the multiplications is not unique. In particular, $\leqslant_G$ is only one of the possible choices. In fact, while in general the matrix product is not commutative, the product of Givens rotations is often commutative. More precisely, whenever two rotations act on different set of indices or exactly on the same two indices applying them in every order leads to the same result. This is a very important property from the computational point of view, and will be exploited in the update of Givens–Vector representations.

It is worth highlighting that the operation of slicing a k-sequence is equivalent to removing the heads and tails from the sequences themselves. For example the slice of $\mathcal{G}$ defined by $\mathcal{G}[3 : n-2]$ is obtained by taking only the bold rotations in the following picture, where $n = 7$, which correspond to $G_{4,2}, G_{5,2}, G_{3,1}, G_{4,1}$.

$\mathcal{G} =$  .

As highlighted by Remark 3.2.13 Givens rotation often commute. The only case when this does not happen is when we multiply $G_i G_{i+1}$ or $G_i G_{i-1}$, that is, when the two rotations act on consecutive indices.

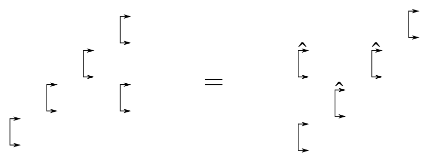
Nevertheless, also in this case we can prove the following lemma, which guarantees that it is still possible to swap rotations if the operation is coupled with an appropriate update.

Lemma 3.2.14 (Turnover)**.** *Let $\mathcal{G}$ be a* 1*-sequence of Givens rotations and* F_i *a Givens rotation acting on the rows* i *and* $i+1$*. Then there exists another sequence* $\hat{\mathcal{G}}$ *and a Givens rotation* $\hat{F}_{i-1}$ *acting on the rows* $i-1$ *and* i *such that*

$$\mathcal{G}F_i = \hat{F}_{i-1}\hat{\mathcal{G}}.$$

Moreover, $\hat{\mathcal{G}}$ *differs from* $\mathcal{G}$ *only in the rotations acting on the rows with indices* $(i-1, i)$ *and* $(i, i+1)$*.*

Proof. A complete proof of this result can be found in [89]. Here we report only a pictorial representation of this fact, that helps to understand what is happening.



On the left-hand side of the equation we have $\mathcal{G}$ and F_i, that is the rightmost rotation. We can move F_i to the left of $\mathcal{G}$ by updating it and updating also the "nearby" rotations as reported in the picture in the right-hand side of the equation. □

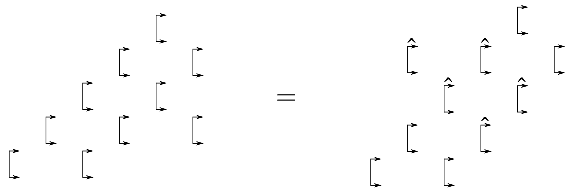
The above result can be extended to k-sequences of rotations for $k > 1$.

Corollary 3.2.15. *Let* $\mathcal{G}$ *be a* k*-sequence of Givens rotations and* F_i *a Givens rotation acting on the rows* i *and* $i+1$*. Then there exists another* k*-sequence* $\hat{\mathcal{G}}$ *and a Givens rotation* $\hat{F}_{i-k}$ *acting on the rows* $i-k$ *and* $i-k+1$ *such that*

$$\mathcal{G}F_i = \hat{F}_{i-k}\hat{\mathcal{G}},$$

where $\hat{F}_{i-k} = I$ *if* $i-k \leqslant 0$*. Moreover,* $\hat{\mathcal{G}}$ *differs from* $\mathcal{G}$ *only in the rotations of indices* $(i-j+1, j)$ *and* $(i-j, j)$ *for* $j = 1, \dots, k$*.*

Again, a pictorial representation of this fact can be useful to figure out the interplay of the rotations. Below, we report the case where $i > k+1$.



It is natural to determine what is the cost of such operations, called turnovers. In fact, they will be often used in the reduction algorithm and so we need to bound their flops count in order to state a bound for the total cost of the reduction algorithm.

As previously said, a turnover on a k-sequence is nothing more than k turnovers on 1-sequences one after the other. So we only need to worry about the cost of a single turnover on a 1-sequence.

In that case we have that only a constant number of rotations needs to be updated (more precisely, only 3 of them) and the cost of computing them is equivalent to the computation of a QR factorization of a 3×3 matrix. This implies that the cost of this operation is $O(1)$ and that the cost of a turnover on a k-sequence is $O(k)$.

3.2.5 *Givens–Vector representations*

We now have all the required tools and notations needed to introduce the concept of Givens–Vector representations, in short GV. In this section we restrict our attention to the case of Hermitian matrices, since they are simpler to handle and they are sufficient for our case. For this reason the following definitions only consider this special case. However, it is very easy to extend these definitions to the more general case by simply adding additional terms to represent the lower and upper part of the matrix separately.

Definition 3.2.16. A *Givens Vector (GV) representation* of rank k for a Hermitian k-quasiseparable matrix A is a triple $(\mathcal{G}, W, D)$ where $\mathcal{G}$ is a k-sequence of Givens rotations, $W \in \mathbb{C}^{k\times(n-1)}$ and D is a diagonal $n \times n$ matrix such that

- D is the diagonal of A;
- for every $i = 1, \dots, n-1$ the subdiagonal elements of the i-th column of A are equal to the last $n-i$ elements of $\mathcal{G}[i+1:]\underline{w}_i$, where we define

$$\underline{w}_i := \begin{bmatrix} 0_i \\ We_i \\ 0_{n-k-i} \end{bmatrix} \text{ if } i < n-k, \qquad \underline{w}_i := \begin{bmatrix} 0_i \\ (We_i)[1:n-i] \end{bmatrix} \text{ otherwise}$$

  where 0_j is the 0 vector of length j if $j > 0$, and is the empty vector otherwise. That is, $\operatorname{tril}(A,-1)e_i = \mathcal{G}[i+1:]\underline{w}_i$.

If the triple $(\mathcal{G}, W, D)$ is a GV representation of the matrix A we write $A = \mathrm{GV}(\mathcal{G}, W, D)$.

We refer to [88, 40] for a detailed analysis of the properties of this representation. We recall here only the following facts:

- If A is k-quasiseparable then there exists a k-sequence $\mathcal{G}$, a matrix $W \in \mathbb{C}^{k\times(n-1)}$ and a diagonal matrix D such that $A = \mathrm{GV}(\mathcal{G}, W, D)$.
- If $A = \mathrm{GV}(\mathcal{G}, W, D)$ for some k-sequence $\mathcal{G}$, $W \in \mathbb{C}^{k\times(n-1)}$ and D diagonal, then A is at most k-quasiseparable.

Remark 3.2.17. Definition 3.2.16 parametrizes the structure of the matrix A by using the columns of W as compressed representation for the lower triangular part of the columns of A and the rotations are needed to parametrize the relation between different columns and to propagate the values in W. However, the same kind of representation can be used to represent the rows of A by storing a $(n-1) \times k$ matrix W and sequences of rotations acting from the right. The two frameworks are completely equivalent and in [89] a $O(n)$ algorithm to swap between the two definitions is given for the case of 1-quasiseparable matrices.

The matrix W is called the *weight matrix*. For this reason these representations are sometimes called Givens–Weight representations. For the sake of clarity in this thesis we will consistently use only the term Givens–Vector representation.

Intuitively, the k-sequence $\mathcal{G}$ represents the column span of the submatrices in the lower part of A, while the columns of W are related to the norm of the columns. In fact, one can note that for every column of $\operatorname{tril}(A,-1)$ we have $\|\operatorname{tril}(A,-1)e_j\| = \|We_j\|$. This can be proved by noting that the strictly lower triangular part of the j-th column of A is obtained by the j-th column of W by multiplication for unitary matrices, that do not alter its norm.

We introduce some results that will help to further characterize these features of $\mathcal{G}$ and W. More precisely, the following lemma allows to check if, given a certain k-sequence $\mathcal{G}$, there exists a GV representation for a matrix A with that k-sequence in it.

Lemma 3.2.18. *Let A be a Hermitian matrix and $\mathcal{G}$ a k-sequence of Givens rotations. Then $B = \mathcal{G}^*A$ is lower banded with bandwidth of k, i.e., $b_{i,j} = 0$ for $i - j > k$, if and only if the matrix A admits a representation of the form $GV(\mathcal{G}, W, D)$ for some $W \in \mathbb{C}^{k\times n}$ and D real diagonal.*

Proof. We first suppose that $A = GV(\mathcal{G}, W, D)$. Recall that, by definition of GV representation, $\operatorname{tril}(A, -1)e_i = \mathcal{G}[i+1:]\underline{w}_i$ for $i = 1, \dots, n-1$. This implies that

$$\mathcal{G}^* \operatorname{tril}(A, -1)e_i = \mathcal{G}^*\mathcal{G}[i+1:]\underline{w}_i = \mathcal{G}^*[:i]\mathcal{G}^*[i+1:]\mathcal{G}[i+1:]\underline{w}_i = \mathcal{G}^*[:i]\underline{w}_i.$$

We also have

$$\mathcal{G}^* \operatorname{triu}(A)e_i = \mathcal{G}^*[:i]\mathcal{G}^*[i+1:]\operatorname{triu}(A)e_i = \mathcal{G}^*[:i]\operatorname{triu}(A)e_i,$$

since $G^*[i+1:]$ is acting on rows that are null. So by decomposing $A = \operatorname{tril}(A, -1) + \operatorname{triu}(A)$ we have

$$\mathcal{G}^*Ae_i = \mathcal{G}^* \operatorname{triu}(A)e_i + \mathcal{G}^* \operatorname{tril}(A, -1)e_i = \mathcal{G}^*[:i](\operatorname{triu}(A)e_i + \underline{w}_i).$$

Now observe that the rotations inside $\mathcal{G}^*[:i]$ only act on the first $i+k$ rows. This implies that, since both $\underline{w}_i$ and $\operatorname{triu}(A)e_i$ have all the components with index strictly bigger than $i+k$ equal to zero, the same must hold for $\mathcal{G}^*[:i](\underline{w}_i + \operatorname{triu}(A)e_i)$, and this completes the proof. The converse is also true. In fact, if $\mathcal{G}^*A$ is lower banded with bandwidth k we can build W by setting $We_i = (\mathcal{G}^*[i+1:]Ae_i)[i+1 : i+k]$ and D equal to the diagonal of A. Then the equation $A = GV(\mathcal{G}, W, D)$ can be verified by direct inspection. □

We have said that the k-sequence $\mathcal{G}$ is related with the column span of the strictly lower submatrices of A. For this reason we say that a k-sequence $\mathcal{G}$ *spans* $U \in \mathbb{C}^{n\times k}$ if there exists $Z \in \mathbb{C}^{k\times k}$ such that $\mathcal{G}^*U = \begin{bmatrix} Z \\ 0 \end{bmatrix}$. This definition is further motivated by the following.

Lemma 3.2.19. *If $\mathcal{G}$ spans $U \in \mathbb{C}^{n\times k}$ then, for every $V \in \mathbb{C}^{n\times k}$, $W \in \mathbb{C}^{k\times(n-1)}$ and D diagonal, the matrix $A_1 = UV^* + GV(\mathcal{G}, W, D)$ is lower k-quasiseparable and $A_2 = t(UV^*) + GV(\mathcal{G}, W, D)$ is k-quasiseparable. In particular, both $\mathcal{G}^*A_1$ and $\mathcal{G}^*A_2$ are lower banded with bandwidth k.*

Proof. For the first part of the lemma it suffices to observe that $\mathcal{G}^*A_1$ is lower banded with bandwidth k. This follows directly by noting that $A = GV(\mathcal{G}, W, D) + UV^*$. Since $\mathcal{G}^*UV^* = \begin{bmatrix} Z \\ 0 \end{bmatrix} V^*$ and $\mathcal{G}^*GV(\mathcal{G}, W, D)$ is lower banded by Lemma 3.2.18, we conclude that also $\mathcal{G}^*A_1$ is lower banded with bandwidth k. Since the strictly lower part of A_2 coincides with the one of A_1 we find that also A_2 is lower k-quasiseparable. Given that A_2 is Hermitian, we conclude that A_2 is also upper k-quasiseparable. To see that also $\mathcal{G}^*A_2$ is lower banded we can write

$$\mathcal{G}^*A_2 = \mathcal{G}^*(A_1 - \operatorname{triu}(UV^*) + \operatorname{triu}(VU^*, 1)) = \mathcal{G}^*A_1 + \mathcal{G}^*R,$$

where R is upper triangular. Since $\mathcal{G}^*$, represented as a matrix, is the product of k upper Hessenberg matrices, it is lower banded with bandwidth k. This implies that also $\mathcal{G}^*R$ is lower banded with bandwidth k and so the same must hold also for $\mathcal{G}^*A_2$. □

Remark 3.2.20. The above lemma shows how the Givens rotations in a GV representation of a matrix in QSH_k are sufficient to determine the column span of the submatrices contained in the lower triangular part. These matrices give the same information obtained by knowing the matrix U in the $D + t(UV^*)$ representation.

Remark 3.2.21. Note that the action of a sequence of Givens rotations $\mathcal{G}$ on vectors and matrices can be represented in matrix form. More precisely, it is possible to define the following operator $\mathcal{M}$

$$\begin{array}{rccl} \mathcal{M} : & \mathcal{S}_G^n & \longrightarrow & \mathbb{C}^{n \times n} \\ & \mathcal{G} & \longmapsto & \mathcal{M}(\mathcal{G}) := \mathcal{G}I \end{array} .$$

We have then that $\mathcal{G}v = \mathcal{M}(G)v$. Notice, in particular, that $\mathcal{M}(\mathcal{G})$ is always a unitary matrix.

We have the following theorem that characterizes the structure of the matrices obtained by applying the operator $\mathcal{M}$ on k-sequences.

Theorem 3.2.22. *If $\mathcal{G}$ is a k-sequence then the matrix $M := \mathcal{M}(\mathcal{G})$ is such that its elements m_{ij} are equal to 0 if $j > i + k$. In particular, if $\mathcal{G}$ is a 1-sequence then M is in lower Hessenberg form and in the general case the matrix M can be factorized as the product of k unitary lower Hessenberg matrices.*

Proof. We prove that, for a 1 sequence, we have that $\mathcal{M}(\mathcal{G})$ is a lower Hessenberg matrix. For this we need to prove that $e_i^t \mathcal{G} e_j = 0$ whenever $j > i + 1$. Notice that, if $j > i + 1$ we have

$$e_i^t \mathcal{G} e_j = e_i^t \mathcal{G}[i+1 : n-1]\mathcal{G}[2 : i]e_j = e_i^t \mathcal{G}[i+1 : n-1]e_j = 0$$

since $\mathcal{G}[2 : i]e_j = e_j$ in view of the fact that it does act only on the first $i + 1 < j$ components. On the other hand $\mathcal{G}[i+1 : n-1]$ does not alter the component of index i, that is 0 in e_j and so we have the thesis. The result on k-sequences can be obtained by recalling that each k-sequence can be seen as a composition of k 1-sequences. □

Given the above result, one may be curious to know if all the unitary lower Hessenberg matrices are obtained as $\mathcal{M}(\mathcal{H})$, i.e., if the map $\mathcal{M}$ is surjective. The next theorem shows that this is almost the case.

Theorem 3.2.23. *Let H be a unitary lower Hessenberg matrix such that $\det(H) = 1$. Then H can be written as $\mathcal{M}(\mathcal{G})$ with $\mathcal{G}$ being a 1-sequence.*

Proof. We prove the result by induction. If $n = 1$ then H is simply a scalar of module 1, and asking that the determinant is 1 implies that $H = 1$. We have then that H can be written as $\mathcal{M}(\emptyset)$, since 1-sequences are made of no rotations when $n = 1$. Assume now that the result holds for $n \times n$ unitary lower Hessenberg matrices and assume $H \in \mathbb{C}^{(n+1)\times(n+1)}$. We compute a rotation G acting on the rows $(n, n+1)$ such that

$$GH = \begin{bmatrix} & & & 0 \\ & \hat{H} & & \vdots \\ & & & 0 \\ 0 & \dots & 0 & 1 \end{bmatrix}.$$

This is always possible since we can compute a rotation G acting on the last two rows that sets the element in position $(n, n+1)$ of H to zero. Notice that for a unitary matrix P we have $Pe_n = e_n \iff P^{-1}e_n = e_n \iff P^*e_n = e_n \iff e_n^*P = e_n^*$ and so since the last column of GH is equal to e_n its last row must be e_n^t. Noting that $\det \hat{H} = 1$ (thanks to $\det G = 1$) we have, by induction, that $\hat{H}$ can be written as a product of rotations $\hat{H} = \hat{G}_{n-1} \dots \hat{G}_2$, so

$$GH = G_{n-1} \dots G_2, \qquad G_i := \hat{G}_i \oplus 1,$$

and setting $G_n = G^{-1}$ and $\mathcal{G} = \{G_n, \dots, G_2\}$ gives us the thesis. □

3.3 HESSENBERG REDUCTION

The main purpose of this section is to develop a Hessenberg reduction algorithm that allows to exploit the quasiseparable structure that we have proven to be present in the linearizations introduced in Section 2.2. Hessenberg reduction is the usual preliminary reduction step of algorithms for eigenvalue computation of classical eigenvalues problems. For this reason we are interested in the case where the secular linearizations are monic, and we only have to deal with the constant coefficient.

In those cases, the zero degree coefficient of the matrix polynomial can be expressed as $A = D + UV^*$, and so it is a very special kind of quasiseparable matrix.

Algorithms for the Hessenberg reduction of quasiseparable matrices are available in the literature, see for example [48] and [56, 41]. These works handle the more general problem of Hessenberg reduction for a generic quasiseparable matrix. Only the minimal assumptions that guarantee that the quasiseparable structure is still available in the Hessenberg matrix H are considered. Nevertheless, the cost of these algorithms is in general $O(n^2k^\alpha)$ with $\alpha > 1$. In [48] the value of α is exactly 3, while in [56] it depends on some choices in the implementation and so it is not fixed, but it is in general $\alpha > 1$. In [41] the optimal complexity $O(n^2k)$ for the reduction is guaranteed in the case of Hermitian matrices, and it is also discussed how to generalize the approach to a Hermitian quasiseparable matrix plus a low rank correction (which is even more general than our assumption of real diagonal plus low rank). However, the numerical experiments are focused on Hermitian matrices (instead of diagonal plus low rank) and we could not perform a direct comparison of the two approaches.

In this section we develop an algorithm with a reduction cost of $O(n^2k)$, so we choose to have $\alpha = 1$. To obtain this result we will show that a certain structure of the starting matrix is preserved during the steps of Hessenberg reduction, and so exactly the same algorithm usually carried out for this process can be executed by using an appropriate parametrization of this structure that yields the gain in the asymptotic complexity. Notice that in this case the complexity is asymptotically lower independently of k, since having $k \leqslant n$ implies $n^2k \leqslant n^3$, the cost of the standard Hessenberg reduction. Clearly since this analysis omits constants it might be that the full Hessenberg reduction is faster if $k \approx n$.

3.3.1 *Rank structure preservation and rank-symmetric matrices*

In this section we show how to use the representations introduced in the previous Section 3.2 in order to reduce a k-quasiseparable matrix A to upper Hessenberg form.

As previously said, we only consider matrices of the form $A = D + UV^*$ where D is real and diagonal and U and V are $n \times k$ matrices. This matrix clearly does have a quasiseparable structure (since any offdiagonal matrix has rank bounded by k) but it is not true that every quasiseparable matrix is of this form.

The reason for which we consider only these matrices is given by the following definition and lemma.

Definition 3.3.1. We say that a set $\mathcal{S} \subseteq \mathbb{C}^{n \times n}$ is *rank symmetric* if for every $A \in \mathcal{S}$ we have $\mathrm{lr}(A) = \mathrm{ur}(A)$, i.e., the lower QS rank of A is equal to the upper one. Moreover, we say that $\mathcal{S}$ is *unitary invariant* if $Q\mathcal{S}Q^* \subseteq \mathcal{S}$ for any unitary matrix Q.

Here we report two important examples of these kinds of sets.

- The set of Hermitian matrices is rank symmetric and unitary invariant. In fact, it is immediate that for any Hermitian matrix A we have $\mathrm{lr}(A) = \mathrm{ur}(A)$ and we also have that if A is Hermitian then QAQ^* is Hermitian for any unitary matrix Q.
- The set of unitary matrices is rank symmetric and unitary invariant. The unitary invariance is obvious, while the rank symmetry property is less trivial, but is a direct consequence of the Nullity Theorem (see [65]), which states that the nullity of the offdiagonal blocks is the same of the corresponding block in the inverse. This, coupled with the fact that $Q^{-1} = Q^*$ leads to the thesis.

Lemma 3.3.2. *Let $A = S + UV^*$ be the sum of a matrix S contained in a rank symmetric and unitary invariant set $\mathcal{S}$ and a rank k one. If $H = QAQ^*$ is one of its upper Hessenberg form obtained with the conjugation by a unitary matrix Q then H is $(1, 2k+1)$-quasiseparable.*

Proof. We can rewrite H as

$$H = QAQ^* = QSQ^* + (QU)(QV)^*.$$

Since $QSQ^* \in \mathcal{S}$ by hypothesis we have $\mathrm{lr}(QSQ^*) \leqslant \mathrm{lr}((QU)(QV)^*) + \mathrm{lr}(H) = k+1$. Since $\mathrm{ur}(QSQ^*) = \mathrm{lr}(QSQ^*)$ we have $\mathrm{ur}(H) \leqslant \mathrm{ur}(QSQ^*) + k \leqslant 2k+1$ which concludes the proof. □

Given that real diagonal matrices are a very particular case of Hermitian matrices, we have that the coefficients of the linearizations of Section 2.2 are in the hypothesis of the above lemma if the nodes $B_i(x)$ are chosen to be real.

The Hessenberg reduction is a very well-understood algorithm in numerical analysis. We analyze what happens to the quasiseparable structure during the execution of the algorithm. Lemma 3.3.2 guarantees that the final matrix H has a quasiseparable structure independently by the procedure chosen, but what about the intermediate steps?

The more widespread implementation of the Hessenberg reduction for general matrices relies on Householder reflections. In order to be able to fully exploit our Givens–Vector representation we instead rely on Givens rotations.

Algorithm 5 Reduction to Hessenberg form by means of Givens rotations

```
1: for $j = 1, \dots, n-2$ do
2:     for $i = n, \dots, j+2$ do
3:         $G \leftarrow \mathtt{givens}(A[i-1, j], A[i, j])$
4:         $A[i-1:i, :] \leftarrow G \cdot A[i-1:i, :]$
5:         $A[:, i-1:i] \leftarrow A[:, i-1:i] \cdot G^*$
6:     end for
7: end for
```

The general algorithm is rather simple and is formulated in Algorithm 5. It requires the computation of $O(n^2)$ Givens rotations. The first $n-2$ rotations are used to set the elements in the first column to zero from the last row to the third one. Then the procedure is repeated on the second column, where only $n-3$ rotations are needed, and so forth.

We call Q_1 the product of the first $n-2$ rotations such that $Q_1 A Q_1^* e_1 = \alpha_1 e_1 + \beta_1 e_2$ for some α_1, β_1. By induction we define Q_i the product of the $n-i-1$ rotations such that $Q_i Q_{i-1} \dots Q_i A Q_1^* \dots Q_{i-1}^* Q_i^* e_i$ has all the elements with row index larger than $j+1$ equal to zero for every column of index $j \leqslant i$.

We are interested in studying the structure of $A_j := Q_j \dots Q_1 A Q_1^* \dots Q_j^*$, that is the partially reduced matrix

$$A_j = Q_j \dots Q_1 (S + UV^*) Q_1^* \dots Q_j^* = \begin{bmatrix} a_{1,1}^{(j)} & \dots & a_{1,j}^{(j)} & \times & \dots & \times \\ a_{2,1}^{(j)} & \ddots & & \vdots & & \vdots \\ & \ddots & a_{j,j}^{(j)} & \times & \dots & \times \\ & & a_{j+1,j}^{(j)} & \star & \dots & \star \\ & & & \vdots & & \vdots \\ & & & \star & \dots & \star \end{bmatrix}. \tag{3.4}$$

3.3.2 *Some technical tools*

The aim of this section is to characterize the structure of the partially reduced matrices obtained at each step of the Hessenberg reduction. As we will see, this will allow to directly give strict bounds for the lower and upper quasiseparability ranks during the procedure. The next lemmas and theorems will provide the necessary tools needed for the result.

Property (iv) of Lemma 3.2.3 is interesting for our purpose. It is natural to ask what happens when we compute $R = t(SAS^*) - St(A)S^*$ where S is not diagonal. In principle we have $R \neq 0$, but we want to better characterize its structure. In our case we will have $S = Q_j$, a matrix version of a sequence of Givens rotations. By looking at the Algorithm 5 we see that Q_j is of the form $\mathcal{M}(\mathcal{G})^*$ where $\mathcal{G}$ is a 1-sequence of Givens rotations. For this reason, in view of Theorem 3.2.22 we have that Q_j is upper Hessenberg. This motivates the next results, that characterize the structure of R when S is upper Hessenberg.

Lemma 3.3.3. *Let $Z \in \mathbb{C}^{k \times k}$ and set $S = Z \oplus I_{n-k}$ where $n > k$. Then for any $A \in \mathbb{C}^{n \times n}$ it holds that*

$$t(SAS^*) - St(A)S^* = W \oplus 0_{n-k},$$

for some $W \in \mathbb{C}^{k \times k}$ where 0_{n-k} is the null matrix of size $n-k$. Similarly, for $S = I_{n-k} \oplus Z$ it holds that $t(SAS^*) - St(A)S^* = 0_{n-k} \oplus W'$, for some $W' \in \mathbb{C}^{k \times k}$. The same properties hold if I_{n-k} is replaced by a diagonal matrix D_{n-k}.

Proof. Concerning the first part, partition A as $A = \left[\begin{smallmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{smallmatrix}\right]$ where $A_{1,1} \in \mathbb{C}^{k \times k}$, so that $SAS^* = \left[\begin{smallmatrix} ZA_{1,1}Z^* & ZA_{1,2} \\ A_{2,1}Z^* & A_{2,2} \end{smallmatrix}\right]$. In view of (v) of Lemma 3.2.3 we have

$$t(SAS^*) = \begin{bmatrix} t(ZA_{1,1}Z^*) & ZA_{2,1}^* \\ A_{2,1}Z^* & t(A_{2,2}) \end{bmatrix}. \tag{3.5}$$

On the other hand,

$$St(A)S^* = S\begin{bmatrix} t(A_{1,1}) & A_{2,1}^* \\ A_{2,1} & t(A_{2,2}) \end{bmatrix} S^* = \begin{bmatrix} Zt(A_{1,1})Z^* & ZA_{2,1}^* \\ A_{2,1}Z^* & t(A_{2,2}) \end{bmatrix}. \tag{3.6}$$

So that, from (3.5) and (3.6) we get $t(SAS^*) - St(A)S^* = W \oplus 0_{n-k}$, with $W = t(ZA_{1,1}Z^*) - Zt(A_{1,1})Z^*$. The second part can be proved similarly. Moreover, if I_{n-k} is replaced by the diagonal matrix D_{n-k} the same properties hold since for a diagonal matrix D one has $t(DAD^*) - Dt(A)D^* = 0$ in view of Property (iv) of Lemma 3.2.3. □

Lemma 3.3.3 characterizes the residual R when the conjugation matrix S only acts on a subset of rows. The lemma guarantees that the residual is different from zero only in the components where the conjugation matrix S is not diagonal. This allows to apply one Givens rotation at a time.

We now state another lemma that shows how composition of conjugation acts on the residual matrix.

Lemma 3.3.4. *Let $S = (Z \oplus I_{n-2})(1 \oplus \hat{S})$ where $Z \in \mathbb{C}^{2\times 2}$, $\hat{S} \in \mathbb{C}^{(n-1)\times(n-1)}$. Then for any matrix A partitioned as $A = \begin{bmatrix} a_{1,1} & u^* \\ v & \hat{A} \end{bmatrix} \in \mathbb{C}^{n\times n}$, where $\hat{A} \in \mathbb{C}^{(n-1)\times(n-1)}$ it holds that*

$$t(SAS^*) - St(A)S^* = W \oplus 0_{n-2} + (Z \oplus I_{n-2})(0 \oplus (t(\hat{S}\hat{A}\hat{S}^*) - \hat{S}t(\hat{A})\hat{S}^*))(Z^* \oplus I_{n-2}).$$

for some $W \in \mathbb{C}^{2\times 2}$.

Proof. Set $B = (1 \oplus \hat{S})A(1 \oplus \hat{S}^*)$ then by Lemma 3.3.3

$$t(SAS^*) = t((Z \oplus I_{n-2})B(Z^* \oplus I_{n-2}) = W \oplus 0_{n-2} + (Z \oplus I_{n-2})t(B)(Z^* \oplus I_{n-2}).$$

On the other hand

$$St(A)S^* = (Z \oplus I_{n-2})(1 \oplus \hat{S})t(A)(1 \oplus \hat{S}^*)(Z^* \oplus I_{n-2}).$$

Thus

$$t(SAS^*) - St(A)S^* = W \oplus 0_{n-2} + (Z \oplus I_{n-2})E(Z^* \oplus I_{n-2}),$$

where $E = t(B) - (1 \oplus \hat{S})t(A)(1 \oplus \hat{S})$. Now, since $B = (1 \oplus \hat{S})A(1 \oplus \hat{S}^*)$, in view of Property (v) of Lemma 3.2.3 we have

$$B = \begin{bmatrix} a_{1,1} & u^*\hat{S}^* \\ \hat{S}v & \hat{S}\hat{A}\hat{S}^* \end{bmatrix}, \quad t(B) = \begin{bmatrix} 0 & v^*\hat{S}^* \\ \hat{S}v & t(\hat{S}\hat{A}\hat{S}^*) \end{bmatrix}.$$

A similar analysis shows that

$$(1 \oplus \hat{S})t(A)(1 \oplus \hat{S}^*) = \begin{bmatrix} 0 & v^*\hat{S}^* \\ \hat{S}v & \hat{S}t(\hat{A})\hat{S}^* \end{bmatrix}.$$

Thus we get

$$t(SAS^*) - St(A)S^* = W \oplus 0_{n-2} + (Z \oplus I_{n-2})(0 \oplus (t(\hat{S}\hat{A}\hat{S}^*) - St(\hat{A})\hat{S}^*))(Z^* \oplus I_{n-2}).$$

□

We want to combine the above lemmas in order to obtain a result usable in the case $S = \mathcal{M}(\mathcal{G})$, that is the one interesting for us. To be more precise, when carrying out the Hessenberg reduction process, we have that, at the step j, $Q_j =: S$ is a product of $n-j-1$ rotations, so that we have $Q_j = G_{j+1} \dots G_{n-1}$. This motivates the following.

Theorem 3.3.5. *Let $A \in \mathbb{C}^{n \times n}$ and $S = G_h \dots G_k$ for some indices $1 \leqslant h \leqslant k \leqslant n-1$. Let c_i and s_i be the parameters defining the rotations G_i and assume that $s_i \neq 0$ for $i = h, \dots, k$. Then the residual matrix $R_n := t(SAS^*) - St(A)S^*$ is of the form $R_n = D + t(ab^*)$ where D is diagonal and a and b are vectors. More precisely we have*

$$a_i = d_i = 0 \ \text{ if } i < h \text{ or } i > k+1, \qquad b_i = \begin{cases} s_h \cdots s_k & \text{if } i = h \\ c_{i-1} s_i \cdots s_k & \text{for } h < i < k+1 \\ 0 & \text{otherwise} \end{cases} .$$

In particular, if $h > 1$ then $R_n e_1 = 0$.

Proof. The matrix S has the form $I_{h-1} \oplus Z_{k-h+2} \oplus I_{n-k-1}$ where Z_{k-h+2} is a unitary Hessenberg matrix of size $k-h+2$. In view of Lemma 3.3.3, we can write $R_n = 0_{h-1} \oplus R_{k-h+2} \oplus 0_{n-k-1}$ and this immediately proves the last statement of the Theorem. Moreover, it follows that $a_i = b_i = d_i = 0$ for $i = 1, \dots, h-1$ and for $i = k+2, \dots, n$ so that it is sufficient to prove the claim for R_{k-h+2}. Equivalently, we may assume that $h = 1$ and $k = n-1$ so that $s_i \neq 0$ for $i = 1, \dots, n-1$. We prove that $R_n \in \mathrm{QSH}_1^n$ by induction on n. For $n = 2$ it holds that $R_2 = \left[\begin{smallmatrix} 0 & \overline{\alpha} \\ \alpha & 0 \end{smallmatrix}\right]$. This way one can choose $a_2 = \alpha / s_1$ and $b_1 = s_1$. For the inductive step, let $n > 1$ and observe that S can be factorized as $S = (Z \oplus I_{n-2})(1 \oplus \hat{S})$ for $Z = \left[\begin{smallmatrix} c & s \\ -\overline{s} & c \end{smallmatrix}\right]$, and $\hat{S} \in \mathbb{C}^{(n-1) \times (n-1)}$, where for notational simplicity we set $s = s_1$, $c = c_1$. Applying Lemma 3.3.4 yields

$$R_n = W \oplus 0_{n-2} + (Z \oplus I_{n-2})(0 \oplus R_{n-1})(Z^* \oplus I_{n-2})$$

for

$$W = t(Z \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix} Z^*) - Zt(\begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix})Z^* =$$
$$= \begin{bmatrix} -c(s a_{2,1} + \overline{s a_{2,1}}) & -s(\overline{c a}_{1,1} + s\overline{a}_{1,2} - \overline{c a}_{2,2} - s\overline{a}_{2,1}) \\ -\overline{s}(c a_{1,1} + \overline{s} a_{1,2} - c a_{2,2} - \overline{s} a_{2,1}) & c(s a_{2,1} + \overline{(s a_{2,1})}) \end{bmatrix},$$

where $R_{n-1} = t(\hat{S}\hat{A}\hat{S}^*) - \hat{S}t(\hat{A})\hat{S}^*$ and $\hat{A}$ is the trailing principal submatrix of A of size $n-1$. A direct inspection shows that

$$R_n = W \oplus 0_{n-2} + \left[\begin{array}{c|c} |s|^2 e_1^T R_{n-1} e_1 & s e_1^T R_{n-1} \tilde{D} \\ \hline \overline{s} \tilde{D} R_{n-1} e_1 & D R_{n-1} D \end{array}\right], \quad \tilde{D} = c \oplus I_{n-2}. \tag{3.7}$$

From the inductive assumption we may write that $R_{n-1} = \mathrm{diag}(\hat{d}) + t(\hat{a}\hat{b}^*)$ for $\hat{a}, \hat{b}, \hat{d} \in \mathbb{C}^{n-1}$, where $\hat{b}_1 = s_2 \cdots s_{n-1} \neq 0$, $\hat{b}_i = c_i s_{i+1} \cdots s_{n-1}$. So that (3.7) turns into

$$R_n = \begin{bmatrix} w_{1,1} & w_{1,2} & & \\ w_{2,1} & w_{2,2} & & \\ & & & \\ & & & \end{bmatrix} + \left[\begin{array}{c|c|cccc} |s|^2 \hat{d}_1 & * & * & \cdots & \dots & * \\ \hline \hat{d}_1 c \overline{s} & c^2 \hat{d}_1 & * & \dots & \dots & * \\ \hline \hat{a}_2 \overline{\hat{b}}_1 \overline{s} & \hat{a}_2 \overline{\hat{b}}_1 c & \hat{d}_2 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \hat{a}_3 \overline{\hat{b}}_2 & \hat{d}_3 & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \ddots & * \\ \hat{a}_{n-1} \overline{\hat{b}}_1 \overline{s} & \hat{a}_{n-1} \overline{\hat{b}}_1 c & \hat{a}_{n-1} \overline{\hat{b}}_2 & \dots & \hat{a}_{n-1} \overline{\hat{b}}_{n-2} & \hat{d}_{n-1} \end{array}\right],$$

where the upper triangular part, denoted with $*$, is determined by symmetry. Thus, it follows that $R_n = \mathrm{diag}(d) + t(ab^*)$ where $d_1 = |s_1|^2\hat{d}_1 + w_{1,1}$, $d_2 = c_1^2\hat{d}_1 + w_{2,2}$, $d_i = \hat{d}_{i-1}$, for $i = 3, \dots, n$; moreover

$$\begin{aligned} a_2 &= \frac{1}{\hat{b}_1}(c\bar{s}\hat{d}_1 + w_{2,1}), \\ a_i &= \hat{a}_{i-1}, \text{ for } i = 3, \dots, n, \\ b_1 &= s_1\hat{b}_1, \ \ b_2 = c_1\hat{b}_1, \\ b_i &= \hat{b}_{i-1}, \text{ for } i = 3, \dots, n-1, \end{aligned}$$

where the condition $s_i \neq 0$ implies that $b_1 \neq 0$. This completes the proof. □

The statement of Theorem 3.3.5 shows that the vector b in the representation of the residual R_n does not depend at all on the matrix A. This allows to prove the next Corollary.

Corollary 3.3.6. *Let A and S be two matrices in the hypothesis of Theorem 3.3.5, so that S is unitary and $S = G_h \dots G_k$ is a product of Givens rotations. Then for any real diagonal matrix D the residual matrix*

$$\tilde{R} = t(SAS^*) - S(D + t(A))S^*$$

is 1-quasiseparable and admits a representation of the form $\tilde{R} = \tilde{D} + t(ab^*)$.

Proof. We can write

$$\tilde{R} = t(SAS^*) - St(A)S^* - SDS^* = D_1 + t(a_1b^*) - SDS^*$$

in view of Theorem 3.3.5. We have then that $SDS^* = SDS^* - St(D)S^* = D_2 + t(a_2b^*)$ since $t(D) = 0$ and b is exactly the same vector as above, given that b only depends on S. Joining these two relations together yields

$$\tilde{R} = D_1 + t(a_1b^*) - D_2 - t(a_2b^*) = D_1 - D_2 + t((a_1 - a_2)b^*)$$

that is in the sought form. □

We can now give an alternative representation of the matrix R_n that will allow to retrieve its Givens–Vector representation used for the reduction algorithm.

Theorem 3.3.7. *Under the assumptions of Theorem 3.3.5 the i-th row of the matrix $R_n = t(SAS^*) - S(D + t(A))S^*$ has the form*

$$e_i^T R = [0, \dots, 0, v_i, d_i, w_i^*]G_{i-2}^*G_{i-3}^* \cdots G_1^* \tag{3.8}$$

where, $v_i, d_i \in \mathbb{C}$, $w_i \in \mathbb{C}^{n-i}$ and $d_i = r_{i,i}$.

Proof. Let us write $S = Q_1Q_2$ where $Q_1 = G_1 \cdots G_{i-2}$, $Q_2 = G_{i-1} \cdots G_{n-1}$, so that (3.8) can be rewritten as $e_i^T RQ_1 = [0, \dots, 0, v_i, d_i, w_i^*]$. This way, it is enough to show that the i-th row of RQ_1 has the first $i-2$ entries equal to zero. In view of Lemma 3.3.3 we have

$$R' := t(Q_2AQ_2^*) - Q_2(D + t(A))Q_2^* = 0_{i-2} \oplus \hat{R} \in \mathrm{QSH}_1^n.$$

Whence

$$Q_2(D + t(A))Q_2^* = t(Q_2AQ_2^*) - 0_{i-2} \oplus \hat{R}. \tag{3.9}$$

Moreover, by definition of R we have

$$RQ_1 = t(Q_1Q_2AQ_2^*Q_1^*)Q_1 - Q_1Q_2(D + t(A))Q_2^*.$$

Setting $B = Q_2AQ_2^*$ and combining the above equation with (3.9) yields

$$RQ_1 = t(Q_1BQ_1^*)Q_1 - Q_1t(B) + Q_1(0_{i-2} \oplus \hat{R}).$$

Now, since $Q_1(0_{i-2} \oplus \hat{R})$ has the first $i-2$ columns equal to zero, it is sufficient to prove that the ith row of $t(Q_1BQ_1^*)Q_1 - Q_1t(B)$ has the first $i-2$ components zero. To this regard, observe that $Q_1 = \tilde{Q}_1 \oplus I_{n-i+1}$, where $\tilde{Q}_1 \in \mathbb{C}^{(i-1)\times(i-1)}$, so that partitioning B as $\begin{bmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{bmatrix}$, where $B_{1,1} \in \mathbb{C}^{(i-1)\times(i-1)}$, by applying again Lemma 3.3.3 we find that $t(Q_1BQ_1^*) - Q_1t(B)Q_1^* = W_{i-1} \oplus 0_{n-i+1}$ for some $W_{i-1} \in \mathbb{C}^{(i-1)\times(i-1)}$. This implies that $t(Q_1BQ_1^*)Q_1 - Q_1t(B)$ has the last $(n-i+1)$ rows equal to zero. This completes the proof. $\square$

3.3.3 *Carrying out the algorithm*

We are now ready to understand what happens during the steps of the reduction algorithm reported in Algorithm 5.

To do this we need to ensure that the matrices Q_j, that can be written as $Q_j = \mathcal{M}(\mathcal{G}_j)$ are such that the rotations in $\mathcal{G}_j$ are different from the identity for indices bigger than $j+1$, in order to satisfy the hypothesis of Theorem 3.3.5. In practice we show that, even if this is not true, we can reduce to a smaller case where the hypothesis are satisfied, and then reconstruct the desired structure by embedding the result in larger matrices.

To this end, we need the following lemma that guarantees that the sequences of Givens rotations do not have "holes".

Lemma 3.3.8. *Let $v \in \mathbb{C}^n$, $v \neq 0$ and consider Givens rotations $G_1, \dots, G_{n-1}$ constructed in such a way that $(G_i \dots G_{n-1})v = (w^{(i)*}, 0, \dots, 0)^*$, where $w^{(i)} \in \mathbb{C}^i$, for $i = 1, \dots, n-1$. If there exists h such that $G_h = I$ then one can choose $G_i = I$ for every $i \geqslant h$, that is, $(G_1 \cdots G_{h-1})v = (w_1^{(1)*}, 0, \dots, 0)^*$.*

Proof. Since $G_i \cdots G_{n-1}$ is a unitary matrix which acts in the last $n-i+1$ components of v, the 2-norm of $v[i:n]$ coincides with the 2-norm of $(G_i \cdots G_{n-1}v)[i:n]$, that is, $|w_i^{(i)}|$. On the other hand, if $G_h = I$ then $(w^{(h)*}, 0, \dots, 0) = (w^{(h+1)*}, 0, \dots, 0)$ so that $w_{h+1}^{(h+1)} = 0$. This implies that $\|v[h+1:n]\| = 0$, whence $v_i = 0$ for $i = h+1, \dots, n$. This way, one can choose $G_i = I$ for $i = h+1, \dots, n-1$. $\square$

Notice that the matrices Q_j are of the form $Q_j = G_{j+1} \dots G_{n-1}$, and they are computed in order to reduce a vector to a multiple of e_1 (if we do not look at the first $j-1$ components). For this reason Lemma 3.3.8 guarantees that if $G_h = I$ then also all the rotation with an index bigger than h are equal to I, so that

$$Q_j = I_j \oplus (\tilde{G}_{j+1} \dots \tilde{G}_{h-1}) \oplus I_{n-h-1}. \tag{3.10}$$

where $\tilde{G}_i$ are restricted version of G_i and h is the minimum index such that $G_h = I$. We have that all the remaining rotations are non-trivial, i.e., $s_i \neq 0$.

We are now ready to characterize the structure of the partially reduced matrices. Recall that we had, by Equation (3.4), the following structure in the matrix A_j:

$$A_j = \begin{bmatrix} a_{1,1}^{(j)} & \dots & a_{1,j}^{(j)} & \times & \dots & \times \\ a_{2,1}^{(j)} & \ddots & & \vdots & & \vdots \\ & \ddots & a_{j,j}^{(j)} & \times & \dots & \times \\ & & a_{j+1,j}^{(j)} & \star & \dots & \star \\ & & & \vdots & & \vdots \\ & & & \star & \dots & \star \end{bmatrix}.$$

We call $\hat{A}_j$ the matrix whose elements are indicated by the $\star$ symbol, that is the submatrix that still need to be reduced to upper Hessenberg form. We want to characterize the structure of this matrix, and to prove that, at any step, there exist $\hat{U}_j$, $\hat{V}_j$, $\hat{W}_j$ and $\hat{S}_j$ such that

$$\hat{A}_j = \hat{U}_j \hat{V}_j^* + t(\hat{U}_j \hat{W}_j^*) + \hat{S}_j$$

where $\hat{U}_j, \hat{V}_j, \hat{W}_j \in \mathbb{C}^{(n-j)\times k}$ and S_j is 1-quasiseparable and Hermitian. We have that the matrix $A_0 = \hat{A}_0 = A = D + UV^*$ clearly has this structure by choosing $S_0 = 0$ and $W_0 = 0$. The next theorem shows that the structure is maintained by the Hessenberg reduction process.

Theorem 3.3.9. *Let* $U, V, W \in \mathbb{C}^{n\times k}$, $S = \mathrm{diag}(d) + t(ab^*) \in \mathrm{QSH}_1^n$ *and define*

$$A = UV^* + t(UW^*) + S.$$

Let G_i, $i = 2, \dots, n-1$ *be Givens rotations acting on the rows* i *and* $i+1$ *such that*

$$QAe_1 = a_{1,1}e_1 + \beta e_2, \quad \textit{where } Q = G_2 \dots G_{n-1}.$$

Then the matrix $\hat{A}$ *obtained by removing the first row and the first column of* QAQ^* *can be written again as*

$$\hat{A} = \hat{U}\hat{V}^* + t(\hat{U}\hat{W}^*) + \hat{S}$$

where $\hat{U}, \hat{W} \in \mathbb{C}^{(n-1)\times k}$, *and* $\hat{S} = \mathrm{diag}(\hat{d} + t(\hat{a}\hat{b}^*)) \in \mathrm{QSH}_1^{n-1}$ *for some vectors* $\hat{d}, \hat{a}, \hat{b} \in \mathbb{C}^{n-1}$. *Moreover,* $\hat{U}$ *and* $\hat{V}$ *are obtained by removing the first row of* QU *and* QV, *respectively.*

Proof. According to Lemma 3.3.8 and to the above remarks, we may assume that in the first step of the process of reduction in Hessenberg form, the parameters s_i satisfy the condition $s_i \neq 0$ for $i = 2, \dots, h$, while $s_i = 0$, for $i = h+1, \dots, n-1$, for some $h \leqslant n-1$. If this is not the case then the matrix Q has the form highlighted in Equation (3.10) and so if we call $\tilde{Q}$ the active block of Q we can apply this theorem to the matrix $\tilde{Q}\tilde{A}\tilde{Q}^*$ where $\tilde{A}$ is the leading square block of A partitioned according to the partitioning of Q. Lemma 3.3.3 provides a way to extend this representation to dimension n. In view of this fact we can assume, without loss of generality, that $h = n-1$. We have

$$QAQ^* = (QU)(QV)^* + F, \qquad F = Q(t(UW^*) + S)Q^*.$$

In view of Corollary 3.3.6 we have $F = t(Q(UW^* + ab^*)Q^*) - R$ for $R \in \mathrm{QSH}_1^n$. Thus

$$QAQ^* = (QU)(QV)^* + t(QU(QW)^*) + t(Qa(Qb)^*) - R. \tag{3.11}$$

Recall from Theorem 3.3.5 that $Re_1 = 0$ and that Q has been chosen so that $QAQ^* e_1 = \alpha e_1 + \beta e_2$. This fact, together with (3.11), implies that the vector $u = t(Qa(Qb)^*)e_1$ is such that $u[3:n]$ is in the span of the columns of $(QU)[3:n,:]$. In view of Property (v) of Lemma 3.2.3 we may write $t(Qa(Qb)^*)[2:n, 2:n] = t(\hat{u}z^*)$ for $\hat{u} = u[2:n]$, and for a suitable $z \in \mathbb{C}^{n-1}$. Applying the linearity properties of Lemma 3.2.3 yields the following representation for the trailing principal submatrix $\hat{A}$ of QAQ^* of size $n-1$:

$$\hat{A} = \hat{U}\hat{V}^* + t(\hat{U}\tilde{W}^* + \hat{u}\hat{z}^*) - \hat{R},$$

where $\hat{U}$, $\hat{V}$ and $\tilde{W}$ are obtained by removing the first row of U, V and W, respectively, while $\hat{R} = R[2:n, 2:n]$. Since $\hat{u}[2:n]$ is in the span of $\hat{U}[2:n,:]$, and since the first row of $\hat{U}$ as well as the first entry of $\hat{u}$ do not play any role in the computation of $t(\hat{U}\tilde{W}^* + \hat{u}\hat{z}^*)$, we may set $\hat{u}_1$ equal to an appropriate value in such a way that $\hat{u}$ is in the span of the columns of $\hat{U}$. This way, the matrix $\hat{U}\tilde{W}^* + \hat{u}\hat{z}^*$ has rank at most k and can be written as $\hat{U}\hat{W}^*$ for a suitable $\hat{W} \in \mathbb{C}^{k\times n}$. Thus we have

$$\hat{A} = \hat{U}\hat{V}^* + t(\hat{U}\hat{W}^*) + \hat{S}$$

for $\hat{S} = -\hat{R}$, that concludes the proof. □

This result concludes the theoretical analysis of the conservations of the rank structure in the partially reduced matrices. Notice that, in fact, if $\hat{A}_j = \hat{U}_h\hat{V}_j^* + t(\hat{U}_j\hat{W}_j^*) + \hat{S}$ as described above we have the following.

Theorem 3.3.10. *Let $\hat{A}_j$ be the submatrix of the matrix A_j obtained at the j-th step of Hessenberg reduction, as per Equation (3.4). Then, if*

$$\hat{A}_j = \hat{U}_j\hat{V}_j^* + t(\hat{U}_j\hat{W}_j^*) + \hat{S}_j$$

with $\hat{U}_j, \hat{V}_j, \hat{W}_j \in \mathbb{C}^{(n-j+1)\times k}$ and $\hat{S}_j \in QSH_k 1$ we have

(i) The QS ranks of $\hat{A}_j$ are bounded by $\mathrm{lr}(\hat{A}_j) \leqslant k+1$ and $\mathrm{ur}(\hat{A}_j) \leqslant 2k+1$.

(ii) The same bounds also holds for the bigger matrix $A_j \in \mathbb{C}^{n\times n}$ so that we have: $\mathrm{lr}(A_j) \leqslant k+1$ and $\mathrm{ur}(A_j) \leqslant 2k+1$.

Proof. We first prove Property (i). Since $\mathrm{rank}(\hat{U}_j\hat{V}_j^*) \leqslant k$ we have $\mathrm{lr}(\hat{U}_j\hat{V}_j^*) \leqslant k$ and $\mathrm{ur}(\hat{U}_j\hat{V}_j^*) \leqslant k$. Moreover, since $\mathrm{ur}(t(\hat{U}_j\hat{W}_j^*)) = \mathrm{lr}(t(\hat{U}_j\hat{W}_j^*) \leqslant k$ and $\mathrm{lr}(\hat{S}_j) = \mathrm{ur}(\hat{S}_j) \leqslant 1$ we have that (by subaddivity of the ranks) $\mathrm{ur}(\hat{A}_j) \leqslant 2k+1$. On the other hand since we have

$$\mathrm{tril}(\hat{U}_j\hat{V}_j^* + t(\hat{U}_j\hat{W}_j^*), -1) = \mathrm{tril}(\hat{U}_j\hat{V}_j^* + \hat{U}_j\hat{W}_j^*, -1) = \mathrm{tril}(\hat{U}_j(\hat{V}_j + \hat{W}_j)^*, -1)$$

the lower rank of $\hat{A}_j$ are bounded by $k+1$. This bound can be trivially extended to the big matrix A_j since every submatrix in its strictly lower triangular part either contains a portion of $\hat{A}_j$ or a part of the subdiagonal elements in the first j positions. For this reasons every lower submatrix must have rank bounded by $k+1$ or 1, and so we have that $\mathrm{lr}(A_j) \leqslant k+1$. To prove the result also for the upper part note that we can write

$$A_j = Q_{j:1}DQ_{j:1}^* + Q_{j:1}U(Q_{j:1}V)^*, \qquad Q_{j:1} := Q_j \dots Q_1.$$

We can note that each submatrix contained in the strictly lower triangular part of $Q_{j:1}DQ_{j:1}$ is of the sum of a rank k matrix obtained by the combination of the low rank factor $-Q_{j:1}U(Q_{j:1}V)^*$

taken to the left-hand side of the above equation, the factors $t(\hat{U}_j \hat{W}_j^*)$ and $\hat{U}_j \hat{V}_j^*$ if the submatrix intersects $\hat{A}_j$ plus a quasiseparable matrix of rank 1 containing the subdiagonal elements or part of $\hat{S}_j$ (again, depending on the choice of the submatrix). The first rank k terms share the same left factor since $\hat{U}_j$ is simply a truncated version of $Q_{j:1} U$ and so their sum is still a rank k matrix. This implies that $\mathrm{lr}(Q_{j:1} D Q_{j:1}^*) \leqslant k+1$. Given that this matrix is Hermitian the same holds also for the upper part and taking the rank k contribution again to the right we have $\mathrm{ur}(A_j) \leqslant 2k+1$, as requested. □

3.3.4 *Operating on matrices using GV representations*

We have presented a theoretical analysis that allows to prove that the ranks in the reduction to Hessenberg form are preserved. Unfortunately, this representation is not very useful as it is, since it might lead to instabilities as highlighted at the start of Section 3.2.1. For this reason we propose to use GV representation to track the structure of the blocks of the form $t(UV^*)$. We have already seen in the previous Section 3.2.5 how it is possible to represent a matrix in this way, but we do not know how to perform the operations that we need efficiently and without degrading the structure.

This section is devoted to cover this topic. We summarize here the operations that we need to able to handle in order to carry out the reduction algorithm described in the previous section. Recall that, at each step, we have to track the structure of the matrix $\hat{A}_j$ that is given in the form

$$\hat{A}_j = \hat{U}_j \hat{V}_j^* + t(\hat{U}_j \hat{W}_j^*) + \hat{S}_j$$

with $\hat{S}_j$ Hermitian and 1-quasiseparable. We propose to use GV representations for $\hat{S}_j$ and for $t(\hat{U}_j \hat{W}_j^*)$. We need to explain how to efficiently perform the following tasks assuming we are given GV representations of a matrix $M = D + t(UV^*) = \mathrm{GV}(\mathcal{G}, W, D) \in \mathrm{QSH}_k$ and of $S = D_S + t(uv^*) \in \mathrm{QSH}_1$, where $U, V \in \mathbb{C}^{n \times k}$, $\mathcal{G}$ spans U, $u, v \in \mathbb{C}^n$ and $u = Ux$ for some vector $x \in \mathbb{C}^k$:

1. Compute a GV representation of rank k of $M + S$. This is guaranteed to exist since the condition $u = Ux$ says that the column span in the lower triangular part of S is contained in the column span in the lower triangular part of M.
2. Given a unitary upper Hessenberg matrix P, compute a GV representation of rank k of $t(PU(PV)^*)$, and a GV representation of rank 1 of $R = PMP^* - t(PU(PV)^*)$. This is needed in order to apply Theorem 3.3.5 in Theorem 3.3.9.

We consider the first problem, of computing a representation of $M + S$. We first note that if we have $M_1 = \mathrm{GV}(\mathcal{G}, W_1, D_1)$ and $M_2 = \mathrm{GV}(\mathcal{G}, W_2, D_2)$ then a representation for $\alpha M_1 + \beta M_2$ is trivially given by $\alpha M_1 + \beta M_2 = \mathrm{GV}(\mathcal{G}, \alpha W_1 + \beta W_2, \alpha D_1 + \beta D_2)$. We want to use this result in order to compute the representation for $M + S$ and we do it by first finding a representation for S of the form $S = \mathrm{GV}(\mathcal{G}, W_S, D_S)$.

It is natural to ask if such a representation is guaranteed to exist. The answer is yes as reported by the following lemma.

Lemma 3.3.11. *Let M and S be two matrices as defined above. Then S admits a representation of the form $S = \mathrm{GV}(\mathcal{G}, W_S, D_S)$, and so $M + S$ can be written as*

$$M + S = \mathrm{GV}(\mathcal{G}, W + W_S, D + D_S).$$

Proof. In view of Theorem 3.2.19 we have $M + S = M + D_S + t(uv^*) = M + D_S + t(U(vx^*)^*)$ is quasiseparable of rank k and $\mathcal{G}^*(M+S)$ is lower banded with bandwidth k and since $\mathcal{G}^*M$ is also lower banded the same holds for $\mathcal{G}^*S$. A straightforward application of Lemma 3.2.18 yields the thesis. □

The above proof is not directly constructive so we need to construct an algorithm for the computation of W_S. In this context we suppose to work with Givens representations only so we assume that $S = \mathrm{GV}(\mathcal{F}, z, D_S)$ is a GV representation of S with quasiseparable rank 1. In order to find W_S we use the fact that, by definition,

$$\mathrm{tril}(S, -1)e_j = \mathcal{G}[j+1:] \begin{bmatrix} 0_j \\ We_j \\ 0_{\max(0, n-j-k)} \end{bmatrix},$$

and we follow these steps (the above equation only holds for the indices up to $n-k$, for the general formula we refer to Definition 3.2.16).

1. We compute the last column of W_S by using Lemma 3.2.18. This is almost free since no rotations are involved, and the only significant element of $W_S e_{n-1}$ is equal to z_{n-1}.

2. We compute $W_S e_i$ starting from $W_S e_{i+1}$; this vector can be computed by using some elements in $\mathcal{G}^*[i+1:]Me_i$. In fact, since we are in the 1-quasiseparable case then $\underline{z}_i = z_i e_{i+1}$. So we have

$$\mathcal{G}^*[i+1:]Me_i = \mathcal{G}^*[i+1:]\mathcal{F}[i+1:]\underline{z}_i = z_i \mathcal{G}^*[i+1:]\mathcal{F}[i+1:]e_i.$$

   In particular, the only relevant quantity that we need to compute to obtain a representation for the i-th column of M is $\Gamma_i := \mathcal{G}^*[i+1:]\mathcal{F}[i+1:]e_i$. To this end we have

$$\begin{aligned} \Gamma_i &= \mathcal{G}^*[i+1]\mathcal{G}^*[i+2:]\mathcal{F}[i+2:]\mathcal{F}[i+1]e_i \\ &= \mathcal{G}^*[i+1]\mathcal{G}^*[i+2:]\mathcal{F}[i+2:](\alpha e_i + \beta e_{i+1}) = \\ &= \mathcal{G}^*[i+1](\beta \Gamma_{i+1} + \alpha e_i) \end{aligned}$$

   for some α, β such that $\alpha^2 + \beta^2 = 1$.

The above procedure allows to compute the representation for S at a very low cost. In fact, the first step costs only $O(1)$ flops, and each subsequent step requires only $O(k)$ rotations on a vector, and so accounts for $O(k)$ flops. In total we have an algorithm with a cost of $O(nk)$ flops for this step.

We now need to handle the computation of the residual matrix provided by Theorem 3.3.5. We notice that Theorem 3.3.7 is a step in the right direction, since it shows that the residual of matrices that are are involved in the process can be obtained directly as Givens–Vector represented matrices. Unfortunately the representation obtained that way is given by rows instead of columns (see Remark 3.2.17). In the book [89] an algorithm for swapping the representation is given and it requires $O(n)$ flops. We rely on that result, and so we assume that we can obtain the residual directly as a GV represented 1-quasiseparable matrix. There is still to clarify how to compute the updated k-quasiseparable matrix $t(PU(PV)^*)$.

Looking at the proof of Theorem 3.3.7 we can see that the matrix R (and its GV representation) can be obtained easily if we are able to compute a single residual $R_i = t(F_i U V F_i^*) -$

$F_i t(UV^*)F_i^*$ obtained by applying a single rotation. For this reason we show how this is possible in the context of GV representations. We will focus on the case where $i < n - k - 1$ so that there is no need to take additional care for border conditions, but the left out cases are completely analogous to this one.

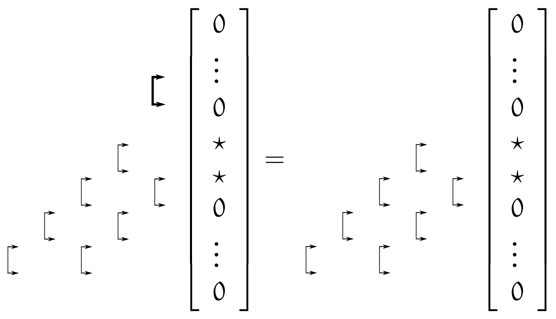
Assume we are given D, U, V, W and $\mathcal{G}$ such that $M = D + t(UV^*) = GV(\mathcal{G}, W, D)$. Note that we can compute an updated $\hat{\mathcal{G}}$ such that $\hat{\mathcal{G}}$ spans F_iU. In fact, we know that $\mathcal{G}^*F_i^*F_iU$ is of the form $\left[\begin{smallmatrix}\times\\0\end{smallmatrix}\right]$ where $\times$ is an appropriate $k \times k$ block. The rotation F_i^* can be passed through the rotations inside $\mathcal{G}^*$ (by properly updating them using the turnover operation – see Lemma 3.2.14) obtaining $\hat{F}_{i+k}^*\hat{\mathcal{G}}^* = \mathcal{G}^*F_i^*$. Then, by $\hat{F}_{i+k}^*\hat{\mathcal{G}}^*F_iU = \left[\begin{smallmatrix}\times\\0\end{smallmatrix}\right]$, we can conclude that also $\hat{\mathcal{G}}^*F_iU = \hat{F}_{i+k}\left[\begin{smallmatrix}\times\\0\end{smallmatrix}\right] = \left[\begin{smallmatrix}\times\\0\end{smallmatrix}\right]$ since $\hat{F}_{i+k}$ is operating on the null rows.

Moreover, it is easy to check that $GV(\hat{\mathcal{G}}, W, D)$ correctly represents the lower part of $t(F_i(D + UV^*)F_i^*)$ on every column but the one with indices $i, i+1$. In fact, the diagonal part of M is left unchanged on the indices different from $i, i+1$. For the rest of the matrix we can distinguish two cases and we do not need to care about D:

- If $j > i+1$, both the left multiplication by F_i and the right multiplication by F_i^* leave unchanged the relevant part of U and V needed for the computation of the portion of the j-th column contained in the lower part of the matrix. Moreover, since in this case $\hat{\mathcal{G}}[j+1:] = \mathcal{G}[j+1:]$ we conclude that the proposed representation for these columns is valid.

- Also when $j < i$ the right multiplication by F_i^* does not change the j-th column at all. However, the left multiplication by F_i does, and we can verify that $\mathrm{tril}(t(F_iUV^*F_i^*), -1)e_j = \mathrm{tril}(F_iUV, -1)e_j$. Recall that, by definition of GV representation, we have
$$\mathrm{tril}(t(UV^*), -1)e_j = \mathrm{tril}(UV^*, -1)e_j = \mathrm{tril}(M, -1)e_j = \mathcal{G}[j+i:]\underline{w_i}.$$
Since $j < i$ we have $F_i\,\mathrm{tril}(UV^*, -1)e_j = \mathrm{tril}(F_iUV^*, -1)e_j$ so that we can write
$$\mathrm{tril}(F_iUV^*, -1)e_j = F_i\mathcal{G}[j+1:]\underline{w}_i = \hat{\mathcal{G}}[j+1:]\hat{F}_{i+k}\underline{w}_j = \hat{\mathcal{G}}[j+1:]\underline{w}_j$$
where the last two equalities follow from the definition of $\hat{\mathcal{G}}$ and from the fact that the components of $\underline{w}_j$ with index bigger than $j+k$ are zero. Thus we have that $GV(\hat{\mathcal{G}}, W, D)$ provides a good representation of the lower part of the j-th column of $t(F_iUV^*F_i^*)$, as requested.

A pictorial representation of these two facts can help to get a better understanding of what is going on (here we are fixing $k = 2$). The rotation F_i on the left is highlighted using the bold font. Equation (3.12) represents the first case, where the rotation F_i does not intersect the indices of the rotations in $\mathcal{G}[j+1:]$, and Equation (3.13) the latter case, where an update of the rotations is necessary.


(3.12)

$$\begin{array}{ccccc} & & & [& \\ \mathbf{[} & & [& & [\\ & [& & [& \\ [& & [& & \end{array} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \star \\ \star \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{array}{ccccc} & & & [& \\ & & \hat{[} & & [\\ & \hat{[} & & \hat{[} & \\ [& & \hat{[} & & \hat{\mathbf{[}} \end{array} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \star \\ \star \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{array}{ccccc} & & & [& \\ & & \hat{[} & & [\\ & \hat{[} & & \hat{[} & \\ [& & \hat{[} & & \end{array} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \star \\ \star \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{3.13}$$

It remains to understand what happens on the rows i and $i+1$. We show how to update D and W in the i an $i+1$ components in order to account for what happens in these indices. Note that these columns of M can be described in the following way (we report the case $k = 3$ for simplicity):

$$M \begin{bmatrix} e_i & e_{i+1} \end{bmatrix} = \mathcal{G}[i+2:] \left(\begin{array}{ccc} [& & \\ & [& \\ & & [\end{array} \begin{bmatrix} d_i & 0 \\ w_{1,i} & 0 \\ w_{2,i} & 0 \\ w_{3,i} & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & \times \\ 0 & d_{i+1} \\ 0 & w_{1,i+1} \\ 0 & w_{2,i+1} \\ 0 & w_{3,i+1} \end{bmatrix} \right).$$

Left and right multiplying by F_i and F_i^* (reported with the bold font), respectively, leads to the following structure:

$$F_i M F_i^* \begin{bmatrix} e_i & e_{i+1} \end{bmatrix} = \mathcal{G}[i+2:] \left(\begin{array}{cccc} \mathbf{[} & & & \\ & [& & \\ & & [& \\ & & & [\end{array} \begin{bmatrix} d_i & 0 \\ w_{1,i} & 0 \\ w_{2,i} & 0 \\ w_{3,i} & 0 \\ 0 & 0 \end{bmatrix} + \mathbf{[} \begin{bmatrix} 0 & \times \\ 0 & d_{i+1} \\ 0 & w_{1,i+1} \\ 0 & w_{2,i+1} \\ 0 & w_{3,i+1} \end{bmatrix} \right) \mathbf{[} .$$

We can explicitly compute the value inside the brackets and then observe that, since $\hat{\mathcal{G}}[i+2:] = \mathcal{G}[i+2:]$, we have a representation of the columns of $F_i M F_i^*$. Now we want to find a Hermitian matrix R of the form $R = \alpha e_{i+1} e_i^t + \overline{\alpha} e_i e_{i+1}^t$, $\hat{w}_{j,i}$, $\hat{w}_{j,i+1}$ for $j = 1, \dots, k$ and $\hat{d}_i$, $\hat{d}_{i+1}$ such that, writing with $\hat{[}$ the rotations taken from $\hat{\mathcal{G}}$, we have

$$\underbrace{\left(\begin{array}{cccc} \mathbf{[} & & & \\ & [& & \\ & & [& \\ & & & [\end{array} \begin{bmatrix} d_i & 0 \\ w_{1,i} & 0 \\ w_{2,i} & 0 \\ w_{3,i} & 0 \\ 0 & 0 \end{bmatrix} + \mathbf{[} \begin{bmatrix} 0 & \times \\ 0 & d_{i+1} \\ 0 & w_{1,i+1} \\ 0 & w_{2,i+1} \\ 0 & w_{3,i+1} \end{bmatrix} \right)}_{C} \mathbf{[} + R = \begin{array}{ccc} \hat{[} & & \\ & [& \\ & & \hat{[} \end{array} \begin{bmatrix} \hat{d}_i & 0 \\ \hat{w}_{1,i} & 0 \\ \hat{w}_{2,i} & 0 \\ \hat{w}_{3,i} & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & \times \\ 0 & \hat{d}_{i+1} \\ 0 & \hat{w}_{1,i+1} \\ 0 & \hat{w}_{2,i+1} \\ 0 & \hat{w}_{3,i+1} \end{bmatrix}. \tag{3.14}$$

Let C be the left matrix in (3.14). Then the elements $\hat{w}_{j,i+1}$ must coincide with the vector $C[3:, 2]$, the diagonal elements $\hat{d}_i$ and $\hat{d}_{i+1}$ are determined by the diagonal of the top 2×2

block of C. It remains to determine the elements $\hat{w}_{j,i}$ and the value α. To find them we can multiply on the left by the inverses of the rotations in $\hat{\mathcal{G}}[i]$. We get the equation

$$\begin{array}{ccc} & & \hat{\lceil} \\ & \hat{\lceil} & \\ \hat{\lceil} & & \end{array} \left(Ce_1 + \begin{bmatrix} 0 \\ \alpha \\ 0 \\ 0 \\ 0 \end{bmatrix} \right) = \begin{bmatrix} \hat{d}_i \\ \hat{w}_{1,i} \\ \hat{w}_{2,i} \\ \hat{w}_{3,i} \\ 0 \end{bmatrix}.$$

We can choose α such that we get a 0 in the last component (which can always be done if the rotations are not trivial – and this is guaranteed by Lemma 3.3.8) and then set the values $\hat{w}_{j,i}$ by back substitution.

3.3.5 *Assembling the reduction algorithm*

Now we have all the necessary tools to formulate the reduction Algorithm 5 in terms of Givens–Vector representations. Recall that our input is a matrix of the form $A = D + UV^*$ so we expect to receive a vector d containing the diagonal elements of D and two $n \times k$ matrices U and V.

The output of our algorithm will be two other $n \times k$ matrices U_{n-2} and V_{n-2} that are the original U and V updated with all the rotations applied on the left. Moreover, we require as output two vectors $\hat{d}$ and $\hat{s}$ with the diagonal and subdiagonal elements of H, the upper Hessenberg form of A that we compute.

This is enough to retrieve the full matrix H and also a structured rank representation of it, as highlighted by the following.

Lemma 3.3.12. *Let* $H = QAQ^*$ *be an upper Hessenberg form of* $A = D + UV^*$*. if* $\hat{U} = QU$ *and* $\hat{V} = QV$ *then* H *can be written as*

$$H = T - t(\hat{U}\hat{V}^*) + \hat{U}\hat{V}^* \tag{3.15}$$

where T *is a Hermitian tridiagonal matrix such that its lower subdiagonal elements coincide with the ones of* H *and the diagonal ones are equal to the ones of* H *minus the ones of* $\hat{U}\hat{V}^*$*.*

Proof. Note that we can write

$$H = QAQ^* = QDQ^* + \hat{U}\hat{V}^*$$

so that, being QDQ^* Hermitian we have $\mathrm{tril}(QDQ^*, -1) = \mathrm{tril}(H, -1) - \mathrm{tril}(\hat{U}\hat{V}^*, -1)$. We can recover the upper part by writing

$$QDQ^* = t(H) - t(\hat{U}\hat{V}^*) + \hat{D}$$

where $\hat{D}$ is chosen diagonal with diagonal elements equal to the ones of H minus the ones of $\hat{U}\hat{V}^*$ in order to satisfy Equation (3.15). The thesis follows by summing $\hat{U}\hat{V}^*$ on both sides of the equation and setting $T = t(H) + \hat{D}$. □

Algorithm 6 reports a pseudocode of the implementation of the reduction algorithm. This sketch of the algorithm does not take into account the representation that we are using, so it is valid both for generators based representations and for Givens–Vector ones. In each case

Algorithm 6 High level reduction process using Givens–Vector representations.

```
1:  function HessenbergReduction(D,U,V)
2:      $A_1 \leftarrow D + UV^*$
3:      $M \leftarrow 0$
4:      $S \leftarrow 0$
5:      $s \leftarrow \texttt{zeros}(1, n-1)$
6:      $d \leftarrow \texttt{zeros}(1, n)$
7:      for $i = 1, \dots, n-2-k$ do
8:          $\mathcal{G} \leftarrow \texttt{cleanColumn}(A_i[:, 1])$
9:          $d[i] \leftarrow (\mathcal{G} A_i[:, 1])[1]$
10:         $s[i] \leftarrow (\mathcal{G} A_i[:, 1])[2]$
11:         $U \leftarrow (\mathcal{G} \cdot U)[2:n, :]$
12:         $V \leftarrow (\mathcal{G} \cdot V)[2:n, :]$
13:         $(R_M, M) \leftarrow \texttt{conjugateAndTruncate}(M, \mathcal{G})$
14:         $(R_S, S) \leftarrow \texttt{conjugateAndTruncate}(S, \mathcal{G})$
15:         $M \leftarrow M + S$
16:         $S \leftarrow R_M + R_S$
17:     end for
18:     $(d[n-1-k:n], s[n-1-k:n-1], U, V) \leftarrow \texttt{reduceTrailingBlock}(A_{n-1-k})$
19: end function
```

the functions conjugateAndTruncate and the sums of lines 15 and 16 have to be implemented according to the representation chosen.

Here we report a brief documentation and explanation for the function called in the pseudocode.

- cleanColumn(v) is a function that takes as input a column vector and returns a sequence of Givens rotations $\mathcal{G}$ such that $\mathcal{G}v = v_1 e_1 + \alpha e_2$ for some α.
- $(R_M, M) \leftarrow$conjugateAndTruncate$(M, \mathcal{G})$ takes as input a quasiseparable Hermitian matrix $M \in \mathrm{QSH}_k$ and a sequence of Givens rotations $\mathcal{G}$. Then it computes a quasiseparable representation for $\mathcal{G}M\mathcal{G}^* - R_M$ where R_M is a matrix in QSH_1. It returns an updated representation of M and the residual matrix R_M.
- $S \leftarrow R_M + R_S$ computes the sum of the matrices $R_M, R_S \in \mathrm{QSH}_1$. This is done by assuming that both have the same sequence of Givens rotations in their representation.
- reduceTrailingBlock(A) reduces the last $k \times k$ block of the matrix using a standard Hessenberg reduction process. This is done because, in the last steps, the trailing block does not have any particular structure anymore and so using the standard reduction algorithm is faster.

Some numerical issues might be encountered in the above version of the algorithm. For instance, some cancellation may happen in the sum $R_M + R_S$, which eventually may affect the Givens rotations of the representation of M.

A technique based on re-orthogonalization can be used to restore better approximations. Recall that the rotations inside the GV representation of M are such that $\mathcal{G}^*U = \left[\begin{smallmatrix} Z \\ 0 \end{smallmatrix}\right]$. Such rotations are not unique but they are essentially unique, that is, their entries can be determined up to a multiplicative constant of modulus 1. Based on this information we can compute rotations in order to obtain $\mathcal{G}^*U$ in the desired form and correct the moduli of the sine and cosine inside $\mathcal{G}$ without altering the signs.

This has shown to be quite effective in practice, leading to better numerical results. The cost of a re-orthogonalization is the cost of a QR factorization of U, thus asymptotically $O(nk^2)$. By performing it every k steps we have a total cost of the modified reduction algorithm that is still $O(n^2k)$, since we need $O(n)$ steps to complete the reduction and $O(nk^2) \cdot \frac{1}{k} O(n) = O(n^2k)$ flops.

3.3.6 *Numerical experiments*

In this section we present some numerical experiments that have been run in order to validate the results obtained. We have implemented Givens–Vector representations and all the routines described above and in Algorithm 6.

The code has been written in the Julia language (see [11]) and have been run on a laptop with an Intel(R) Core(TM) i3-2367M CPU running at 1.40GHz and 4 GB of RAM. Moreover, it will soon be available at `http://numpi.dm.unipi.it/software/` for testing.

The goal of the experiments are the following:

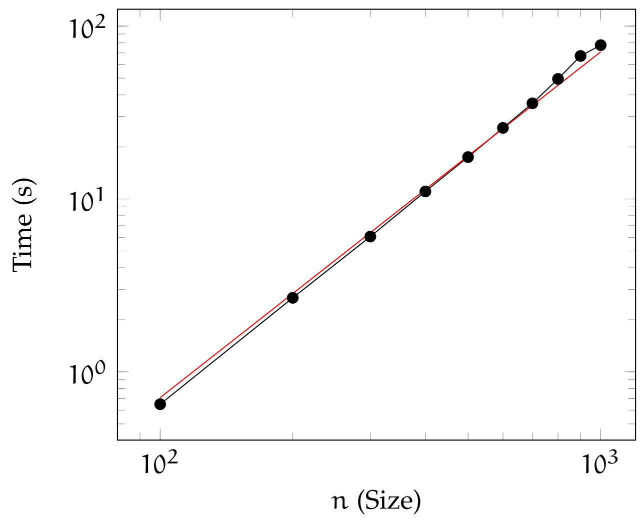
- Validate the theoretical complexity of the algorithm in n, that is expected to be quadratic.
- Validate the theoretical complexity of the algorithm in k, that is expected to be linear. This is the main difference with other Hessenberg reduction algorithm available in the literature.
- Find out what can be inferred on the stability of the algorithm. As we will see, the results obtained are sometimes affected by numerical errors. Some techniques such as the re-orthogonalization can help in these case and has been used in our experiments.

Every experiment has been run 10 times and the mean value of the timings has been taken. In Figure 3.3 we have reported, in log scale, the timings for some experiments with $n = 100 \cdot i$ for $i = 1, \dots, 10$. In Figure 3.4 we have reported the CPU time in the case of matrices of fixed size $n = 400$ with various quasiseparable ranks ranging from 5 to 160.

Looking at the results in Figure 3.4 we see that the complexity in the rank is almost sublinear at the start. This is due to the inefficiency of operations on small matrices and the overhead of these operations in our Julia implementation. The linear trend starts to appear for larger ranks.

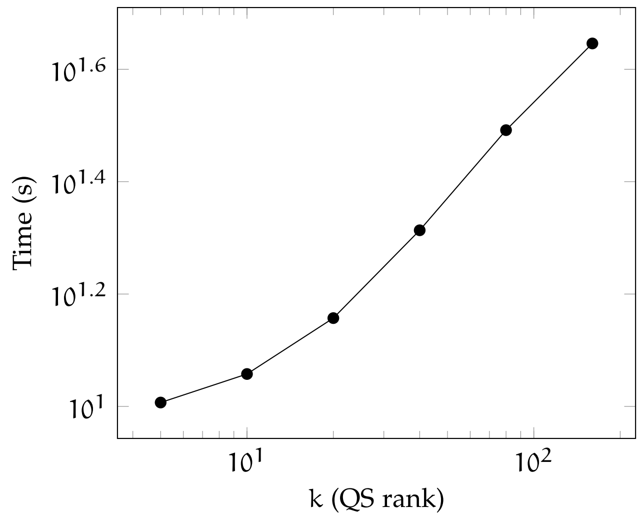
As a last experiment in Figure 3.5 and Figure 3.6 we have reported the absolute and relative errors, respectively, on eigenvalue computations for various sizes and fixed quasiseparable rank. The errors were obtained as differences between the eigenvalues computed from the starting full matrix using the QR algorithm and the QR algorithm applied to the Hessenberg matrix provided by our algorithm. In these examples the re-orthogonalization technique described in Section 3.3.5 has been used, in order to mitigate the errors.

The matrices in these examples have been obtained by using the `randn` function that constructs matrices whose elements are drawn from a $N(0,1)$ Gaussian distribution. This function has been used to construct D, U and V diagonal and $n \times k$, respectively, such that $A = D + UV^*$.



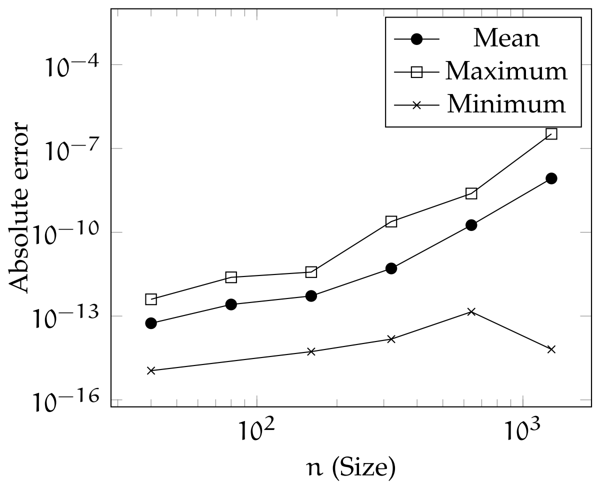
Size	Time (s)
100	0.65
200	2.68
300	6.06
400	11.05
500	17.5
600	25.8
700	35.7
800	49.4
900	67.17
1000	77.56

Figure 3.3: CPU time, in seconds, for the Hessenberg reduction of a diagonal plus rank 10 matrix of size n. Here the line is the plot of γn^2 for an appropriate γ. It is evident the quadratic behavior of the time.



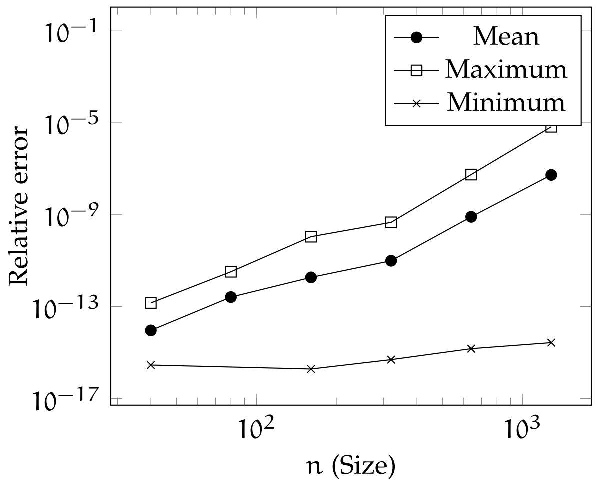
QS Rank	Time (s)
5	10.16
10	11.42
20	14.36
40	20.58
80	31.02
160	44.25

Figure 3.4: CPU time, in seconds, for the Hessenberg reduction of a 400×400 diagonal plus rank k matrix.



Size	Mean	Min	Max
40	5.52e-14	1.11e-15	3.97e-13
80	2.59e-13	0.0	2.43e-12
160	5.23e-13	5.32e-15	3.74e-12
320	5.06e-12	1.49e-14	2.41e-10
640	1.80e-10	1.42e-13	2.43e-9
1280	8.43e-9	6.43e-15	3.32-7

Figure 3.5: Absolute errors on eigenvalues computation for random matrices of quasiseparable rank 30 and variable sizes.



Size	Mean	Min	Max
40	9.13e-15	2.86e-16	1.42e-13
80	2.55e-13	0.0	3.22e-12
160	1.83e-12	1.92e-16	1.07e-10
320	9.66e-12	4.91e-16	4.49e-10
640	7.79e-10	1.47e-15	5.34e-8
1280	5.15e-8	2.69e-15	6.35e-6

Figure 3.6: Relative errors on eigenvalues computation for random matrices of quasiseparable rank 30 and variable sizes.

3.4 HESSENBERG TRIANGULAR REDUCTION

In this section we want to generalize the results of Section 3.3. In that case we had a matrix $A = D + UV^*$ in the form diagonal plus low rank and we wanted to compute its Hessenberg form. The main motivation for this is to solve the eigenvalue problem $xI - A$ that is equivalent to $xI - H$ where $H = QAQ^*$ is upper Hessenberg, and we know that secular linearizations for monic matrix polynomials have this form.

It is natural to ask what happens when the matrix polynomial is not monic and we build a non-monic linearization $xA - B$. It is easy to note that also in this case both A and B have a rank structure. More precisely, they have the form

$$A = I + U_A V_A^* \qquad B = D_B + U_B V_B^*.$$

Similarly to what we have done in the monic case, here we assume D_B to be real. The preprocessing step usually performed before solving eigenvalues problems in form of a pencil is the Hessenberg triangular reduction [80]. This process takes the role of the Hessenberg reduction in the monic case and constructs two unitary matrices Q and Z such that

$$T = QAZ^*, \qquad H = QBZ^*$$

where T is upper triangular and H is upper Hessenberg. The algorithm for the reduction has been introduced in [80] by Moler and Stewart. It can be described with the following steps:

(i) The matrix A is reduced to upper triangular form by computing its QR factorization. If we have $A = PR$ where P is unitary and R upper triangular we can left-multiply both A and B by P^* so that $A_0 := P^*A = R$ and $B_0 := P^*B$.

(ii) The matrix B is reduced to upper Hessenberg form applying Givens rotations from the left. Each rotation also multiplies A and degrades its upper triangular structure. This can be restored by right-multiplying by an appropriate Givens rotations, that does not destroy the partial Hessenberg structure of B. In order to achieve this the lower diagonal elements of B (except the first subdiagonal) are set to 0 starting from the first column and from the bottom to the top.

Remark 3.4.1. Note that, in the first step, it is not strictly necessary to compute a QR factorization of A in order to take it to upper triangular form. In principle, any combination of unitary transformations Q and Z such that QAZ^* is upper triangular can be used. This is important for us since later we will exploit this freedom in order to keep the upper triangular version of A in a structured form that makes it easier to perform efficient operations on it.

3.4.1 *A restriction operator*

In Section 3.3 we proved that a particular rank structure is maintained during the steps of Hessenberg reduction performed with the use of Givens rotations. However, we have shown that the structure is not maintained in the whole matrix (or at least not explicitly) but only in the trailing part that still needs to be reduced. We want to give similar results for the Hessenberg triangular reduction.

To make this notion of considering the trailing submatrix as transparent as possible and to operate easily on these matrices we introduce the following *restriction operator*:

$$\mathcal{R} : \bigcup_{n,k \in \mathbb{N}^+} \mathbb{C}^{n\times k} \longrightarrow \bigcup_{n,k\in\mathbb{N}} \mathbb{C}^{(n-1)\times k}$$

$$\mathcal{R}(A) \longmapsto R_n A, \qquad R_n = \begin{bmatrix} 0 & 1 & & \\ \vdots & & \ddots & \\ 0 & & & 1 \end{bmatrix} \in \mathbb{C}^{(n-1)\times n}$$

For any matrix A the operator $\mathcal{R}(A)$ extracts the submatrix obtained by removing the first row from A.

In the same way we define the operator $\mathcal{R}^*$ as the right multiplication by R_k^* and we use the notation $A\mathcal{R}^*$ to mean $\mathcal{R}^*$ applied to A (according to the syntax for matrix multiplication). In this way we have that $\mathcal{R}A\mathcal{R}^*$ is the trailing principal submatrix of A obtained by removing the first row and the first column. We use the shorthand $\mathcal{R}^m$ and $\mathcal{R}^{*,m}$ to mean the composition of $\mathcal{R}$ and $\mathcal{R}^*$ performed m times.

Note that, in principle, the notation $\mathcal{R}^*A$ does not make sense, since we have only defined the meaning of $A\mathcal{R}^*$. Nevertheless, we agree that $\mathcal{R}^*A := I\mathcal{R}^*A$ where I is the identity matrix of appropriate dimension. The same holds also for $A\mathcal{R}$, and allows us to define also $\mathcal{R}^*\mathcal{R} := I\mathcal{R}^*\mathcal{R}I$ and $\mathcal{R}\mathcal{R}^* = \mathcal{R}I\mathcal{R}^*$.

It is easy to show that these operators enjoy the following properties, most of which are a consequence of thinking of $\mathcal{R}$ and $\mathcal{R}^*$ as projection operators.

- $\mathcal{R}$ and $\mathcal{R}^*$ are linear with respect to $\mathbb{C}$ so that, for any $\mu, \eta \in \mathbb{C}$, $\mathcal{R}(\mu A + \eta B) = \mu\mathcal{R}A + \eta\mathcal{R}B$.
- $(\mathcal{R}A)^* = A^*\mathcal{R}^*$, so that the usage of the * operator on $\mathcal{R}$ is coherent with the one defined on matrices.
- For every triangular and invertible matrix T we have $(\mathcal{R}T\mathcal{R}^*)^{-1} = \mathcal{R}T^{-1}\mathcal{R}^*$. In particular, if T is invertible then $\mathcal{R}T\mathcal{R}^*$ is still invertible.
- $\mathcal{R}\mathcal{R}^* = I$, that is, the composition of these two operators act as the identity. The converse is not true, so that $\mathcal{R}^*\mathcal{R} \neq I$.
- For every upper triangular matrix T the relation $\mathcal{R}^i T \mathcal{R}^{*,i} \mathcal{R}^i = \mathcal{R}^i T$ holds. Note that, in general, $\mathcal{R}A\mathcal{R}^*\mathcal{R}X \neq \mathcal{R}AX$ since, as reported in the previous point, $\mathcal{R}^*\mathcal{R} \neq I$.

3.4.2 *Choosing the initial transformation*

As we have noted in Remark 3.4.1 the preliminary reduction to upper triangular form of the matrix A can be chosen freely. We are interested in choices that preserve the rank structure as much as possible.

Recall that $A = I + U_A V_A^*$, and that we have chosen Q_0 and Z_0 so that $A_0 = Q_0 A Z_0^* = T_0$ is upper triangular. This is the key of our analysis, and we show that not only the upper triangular matrix T_0 is structured, but the same holds for the unitary matrix $Q_0 Z_0^*$.

For this analysis we need to be more precise on the structure of the matrices Q_0 and Z_0. We show two possible strategies to compute them.

Strategy 1 is the one commonly used in the Hessenberg triangular reduction (HTR from now on), while Strategy 2 is the one that we propose to ease the parametrization of the structure.

Reduction strategy 1. We compute a QR factorization of A so that $A = QR$. We set then $Q_0 = Q^*$, $T_0 = Q_0 A = R$ and $B_0 = Q_0 B$. In this strategy Z_0 is chosen to be equal to the identity matrix.

Unfortunately, this choice makes it harder to store the matrices T_i since, even if we can prove that they are quasiseparable (see Lemma 3.4.2), we don't have an explicit expression for the generators of the upper triangular part. For this reason, we present the following alternative strategy where the matrix $T_0 = Q_0 A Z_0^*$ is sparse.

As we will prove in Lemma 3.4.15 the sparsity is also preserved in the trailing submatrices of T_i for $i > 0$, and this will provide a practical way to implement the algorithm.

Reduction strategy 2. We describe the strategy as follows:

- We compute a unitary matrix Z, given as the product of k sequences of ascending Givens rotations such that ZU_A is upper triangular. Then we compute the matrix $ZAZ^* = I + ZU_A(ZV_A)^*$ that has the following structure:

$$ZAZ^* = \begin{bmatrix} X & Y \\ 0 & I_{n-k} \end{bmatrix}, \quad X \in \mathbb{C}^{k\times k},\ Y \in \mathbb{C}^{k\times n-k}.$$

- We compute a $k \times k$ unitary matrix P, again given as a sequence of Givens rotations, such that PX is upper triangular. We have then that we can choose $Q = (P \oplus I_{n-k})Z$ and then $QAZ^* = T$ is upper triangular.

We focus on the analysis of Strategy 2. Nevertheless, note that the hypothesis of Lemma 3.4.2 are satisfied for both the strategies listed above.

Lemma 3.4.2. *Let $A = I + U_A V_A^*$ with U_A, V_A two $n \times k$ matrices and Q_0 and Z_0 unitary matrices such that $Q_0 A Z_0^*$ is upper triangular. Then the upper quasiseparability rank of $Q_0 A Z_0^*$ is bounded by $2k$. Moreover, both the lower and the upper quasiseparability ranks of $Q_0 Z_0^*$ are bounded by k.*

Proof. We have $T_0 = Q_0 A Z_0^* = Q_0 Z_0^* + (Q_0 U_A)(Z_0 V_A)^*$ so that the (strictly) lower triangular part of $Q_0 Z_0^*$ coincides with the stricly lower triangular part of $-(Q_0 U_A)(Z_0 V_A)^*$. In particular, every submatrix contained in the lower part of $Q_0 Z_0^*$ cannot have a rank higher than k, and this proves the bound about the lower quasiseparability rank. Since unitary matrices are rank-symmetric, we can say the same for the upper quasiseparability rank.

The statement about the upper QS rank of T_0 can be derived from that fact that $T_0 = Q_0 Z_0^* + QU_A(Z_0 V_A)^*$, and both summands cannot have an upper QS rank bigger than k. □

We can now show that the matrix B_0 is also structured.

Lemma 3.4.3. *Let $B_0 = Q_0 B Z_0^*$ where Q_0 and Z_0 are unitary matrices as above. Then there exist three $n \times k$ matrices W_0 and S_0, Y_0 and D_0 real diagonal such that*

$$B_0 = T_0(D_0 + t(W_0 S_0^*) + Z_0 U_B (Z_0 V_B)^*) + Q_0 U_A Y_0^*.$$

Proof. In both reduction strategies (1) and (2) we have that the unitary matrix Z can be decomposed as k sequences of Givens rotations. By applying Lemma 3.3.5 repeatedly k times, we obtain that there exist W_0 and S_0 two $n \times k$ matrices such that

$$Z(D_B + U_B V_B^*)Z^* = D_0 + (ZU_B)(ZV_B)^* + t(W_0 S_0^*).$$

We have then that

$$\begin{aligned} B_0 = Q_0 B Z_0^* &= Q_0 Z_0^* Z_0 (D_B + U_B V_B^*) Z_0^* \\ &= (T_0 - Q_0 U_A (Z_0 V_A)^*)(D_0 + Z_0 U_B (Z_0 V_B)^* + t(W_0 S_0^*)) \\ &= T_0 (D_0 + Z_0 U_B (Z V_B)^* + t(W_0 S_0^*)) + Q_0 U_A Y_0^* \end{aligned}$$

for some Y_0. This concludes the proof. □

Remark 3.4.4. Note that in the above representation the term $Q_0 U_A Y_0^*$ is such that all the rows with index larger than k are equal to 0. In particular, this term does not influence the lower QS rank of the matrix B_0.

Lemma 3.4.5. *The lower and upper QS rank of* B_0 *are bounded, respectively, by* $2k$ *and* $4k$.

Proof. Recall that $T_0 = Q_0 Z_0^* + Q_0 U_A (Z_0 V_A)^*$. In particular, we have that, by setting $H_0 = D_0 + Z_0 U_B (Z V_B)^* + t(W_0 S_0^*)$,

$$B_0 = T_0 H_0 + Q_0 U_A Y_0^* = Q_0 Z_0^* H_0 + Q_0 U_A (Y_0 + Z_0 V_A)^*$$

Since H_0 is quasiseparable of order $2k$ and $Q_0 Z_0^*$ is quasiseparable of order k, we have that B_0 has quasiseparable rank at most $4k$. We can make the bound sharper in the lower part by observing that $Q_0 Z_0^* = \hat{Q}_0 \oplus I_{n-k}$ so that the multiplication by $Q_0 Z_0^*$ leaves all the rows with index bigger than k unchanged. It is easy to see that under this assumption the lower QS rank of $Q_0 Z_0^* H_0$ is still bounded by $2k$. Moreover, since $e_j^t Q_0 U_A = 0$ for every $j > k$ the second part of the sum also changes only elements in first k rows, thus leaving the lower QS rank unchanged. In conclusion, the lower QS rank of B_0 can be bounded by $2k$. □

3.4.3 *Analyzing the induction step*

We are now ready to analyze the subsequent steps and to understand how the QS ranks evolve. We introduce some simplified notation for the unitary matrices multiplying on the left and on the right:

$$Q_{i:0} := Q_i \dots Q_0, \qquad Z_{0:i}^* := Z_0^* \dots Z_i^*, \qquad Z_{i:0} := Z_i \dots Z_0$$

We start by looking at T_i, whose structure is easily provided by the following

Lemma 3.4.6. *Let* $T_i = Q_i \dots Q_0 A Z_0^* \dots Z_i^*$ *be the upper triangular matrix obtained at the* i*-th step of the HTR of the pencil* $xA - B$*. Then the upper QS rank of* T_i *can be bounded by* $2k$.

Proof. The result can be obtain by writing $T_i = Q_i \dots Q_0 Z_0^* \dots Z_i^* + \hat{U}_A \hat{V}_A^*$ where

$$\hat{U}_A = Q_i \dots Q_0 U_A, \qquad \hat{V}_A = Z_i \dots Z_0 V_A$$

and the following the same procedure of Lemma 3.4.2. □

Since the analysis of the QS ranks evolution of B_i is a little bit more involved, it is convenient to define some new notation to make the exposition clearer.

We are interested in formulating an hypothesis on the structure of B_i. Here and in the following we will for simplicity assume that A is invertible. In Section 3.4.4 we will show that this is actually not strictly necessary, but rank deficiency in A needs some additional care and might deteriorate the bound on the rank.

For this analysis we will need to keep track of a particular $n \times j$ matrix U. We introduce the following notation, for any choice of the matrix U.

$$U_i = Q_{i:0}U, \quad U_{Z,i} = Z_{i:0}U, \qquad \hat{U}_i = \mathcal{R}^i U_i, \qquad \hat{U}_{Z,i} = \mathcal{R}^i U_{Z,i}. \tag{3.16}$$

Hypothesis 1: The matrix $B_0 = Q_0 B Z_0^*$ has the form

$$B_0 = T_0(D + t(U_{Z,0}S^*) + t(a_0 b_0^*)) + U_0 W^* \tag{3.17}$$

for some S and W $n \times j$ matrices and for a $n \times j$ matrix U such that there exists X_A and X_B with $UX_A = U_A$ and $UX_B = U_B$.

We can note the both choices of an initial transformation lead to a structure of this kind by setting $U = [U_A \ U_B]$ and $X_A^t = [I \ 0]$, $X_B^t = [0 \ I]$.

We can show that this kind of representation is "almost" preserved during the iterations. More precisely, it will not be possible to show that the entire matrix B_i still has a representation of this kind, but we can show that the trailing submatrix of size $(n-i) \times (n-i)$ of B_i still has this form.

In the same spirit of the definition of the matrices of Equation (3.16) we define $\hat{B}_i = \mathcal{R}^i B \mathcal{R}^{*,i}$. We want to show that, even if B_i might not have the same structure of B_i, $\hat{B}_i$ does. We can now state the most important rank conservation theorem of this section.

Theorem 3.4.7. *Let $\hat{B}_i$ be trailing principal submatrix defined as above and obtained from B_i at the i-th step of the HTR. If B_0 has the form given in Equation (3.17) then $\hat{B}_i$ can be represented as*

$$\hat{B}_i = \hat{T}_i(D_i + t(\hat{U}_{Z,i}S_i^*) + t(a_i b_i^*)) + \hat{U}_i W_i^*$$

where D_i is a real diagonal matrix, S_i and W_i are some appropriate $(n-i) \times j$ matrices and a_i and b_i are vectors of length $(n-i)$, where j is the number of columns of U.

We prove now some technical tools that will be needed in order to give the proof of Theorem 3.4.7.

Lemma 3.4.8. *Let A be a $n \times n$ complex matrix, U, V two $n \times k$ matrices and a, b two vectors such that $A = t(UV^*) + t(ab^*)$. Assume that $b_1 \neq 0$ and Q is a unitary matrix of the form $1 \oplus \hat{Q}$ such that $QAe_1 = \alpha e_1 + \beta e_2$ for some $\alpha, \beta \in \mathbb{C}$. Then there exist a vector $x \in \mathbb{C}^k$ and $\hat{\alpha}, \hat{\beta} \in \mathbb{C}$ such that $QUx = Qa + \hat{\alpha} e_1 + \hat{\beta} e_2$.*

Proof. Recall that by definition of the operator $t(\cdot)$ we can write $t(UV^*)e_1 = UV^*e_1 - (e_1^t UV^* e_1)e_1$. This implies the existence of a constant γ such that

$$\alpha e_1 + \beta e_2 = QAe_1 = QUV^*e_1 + Qab^*e_1 + \gamma e_1.$$

The above can be rewritten as $Qab^*e_1 = -QUV^*e_1 + (\alpha - \gamma)e_1 + \beta e_2$ and since, by hypothesis, $b^*e_1 \neq 0$ we can set $x = \frac{-V^*e_1}{b^*e_1}$ and obtain the thesis with $\hat{\alpha} = \frac{\alpha-\gamma}{b^*e_1}$ and $\hat{\beta} = \frac{\beta}{b^*e_1}$. □

We recall here a lemma known as Woodbury matrix inversion formula that will be widely used in the following. The original proof of this result can be found in [93]. This version of the formula is not as general as in the original paper, but is tailored to our needs.

Lemma 3.4.9 (Woodbury matrix inversion)**.** *Let* A*,* U *and* V *be an* $n \times n$ *and two* $n \times k$ *matrices, respectively. If both* A *and* $A + UV^*$ *are invertible then*

$$(A + UV^*)^{-1} = A^{-1} - A^{-1}U(I + V^*A^{-1}U)^{-1}V^*A^{-1}.$$

In the special case where A *is the identity this relation simplifies to*

$$(I + UV^*)^{-1} = I - U(I + V^*U)^{-1}V^*.$$

Lemma 3.4.10. *Using the above notation, let* $T_i = Q_i \dots Q_0(I + U_A V_A^*)Z_0^* \dots Z_i^*$ *and* U_i *and* $U_{Z,i}$ *be the matrices obtained by left multiplying* U *by the sequence of* Q_j *and* Z_j*, respectively. Then there exist two matrices* S *and* $\hat{S}$ *such that*

$$T_i^{-1}U_i = U_{Z,i}S, \qquad \hat{T}_i^{-1}\hat{U}_i = \hat{U}_{Z,i}S.$$

Proof. The first statement can be verified by direct inspection, since

$$T_i^{-1}U_i = Z_i \dots Z_0(I + U_A V_A)^{-1}Q_1^* \dots Q_i^* Q_i \dots Q_1 U = U_{Z,i} - Z_{i:0}U_A\tilde{S}.$$

where $\tilde{S}$ can be obtained by relying on the Woodbury matrix inversion formula of Lemma 3.4.9. S can be chosen as $S = I - X_A\tilde{S}$. Then the statement can be verified directly recalling that $UX_A = U_A$. On the other hand,

$$\hat{T}_i^{-1}\hat{U}_i = \mathcal{R}^i T^{-1}\mathcal{R}^{*,i}\mathcal{R}^i U = \mathcal{R}^i T^{-1}U = \mathcal{R}^i U_{Z,i}S = \hat{U}_{Z,i}S$$

which concludes the proof. □

We now have all the tools needed to prove Theorem 3.4.7.

Proof of Theorem 3.4.7. We prove this result by induction. First, we observe that it is possible to rewrite $\hat{B}_i$ using the restriction operator:

$$\hat{B}_i = \mathcal{R}^i B_i \mathcal{R}^{*,i} = \mathcal{R}^i(Q_i \dots Q_0^*(I + U_B V_B)^* Z_0^* \dots Z_i^*)R^{*,i}$$

and a similar relation also holds for T_i:

$$\hat{T}_i = \mathcal{R}^i T_i \mathcal{R}^{*,i} = \mathcal{R}^i(Q_i \dots Q_0^*(I + U_A V_A)^* Z_0^* \dots Z_i^*)R^{*,i}.$$

By taking advantage of the properties of the restriction operator we can also write the inductive hypothesis on $\hat{B}_{i-1}$:

$$\begin{aligned}\hat{B}_{i-1} &= \hat{T}_{i-1}\left(\hat{D}_{i-1} + t(\hat{U}_{Z,i-1}\hat{S}_{i-1}^*) + t(\hat{a}_{i-1}\hat{b}_{i-1}^*)\right) + \hat{U}_{i-1}\hat{W}_{i-1}^* \\ &= \mathcal{R}^{i-1}T_{i-1}(D_{i-1} + t(U_{Z,i-1}S_{i-1}^*) + t(a_{i-1}b_{i-1}^*))\mathcal{R}^{*,i-1} + \mathcal{R}^{i-1}U_{i-1}(\mathcal{R}^{i-1}W_{i-1})^*.\end{aligned}$$

Here we stress that while some of the matrices above are actually obtained as a truncation of bigger matrices (such as $\hat{U}_{Z,i}$ that is defined as $\mathcal{R}^{i-1}Z_i \dots Z_0 U$, for example), some others are not. More precisely, the matrices $\hat{D}_{i-1}$ and $\hat{S}_{i-1}$ and the vectors $\hat{a}_{i-1}$ and $\hat{b}_{i-1}$ are indeed only defined in their small versions by means of the inductive hypothesis. In the following we will sometimes use bigger versions of them written as S_{i-1}, D_{i-1}, a_{i-1} and b_{i-1}. These are defined simply padding the original small ones with additional zeros at the beginning. More precisely, we define

$$S_{i-1} = (\mathcal{R}_n^*)^{i-1}\hat{S}_{i-1}(\mathcal{R}_n)^{i-1}, \qquad D_{i-1} = (\mathcal{R}_n^*)^{i-1}\hat{D}_{i-1}(\mathcal{R}_n)^{i-1},$$
$$a_{i-1} = (\mathcal{R}_n^*)^{i-1}\hat{a}_{i-1}, \qquad b_{i-1} = (\mathcal{R}_n^*)^{i-1}\hat{b}_{i-1}.$$

We note that the theorem holds for $\hat{B}_0 = B_0$ by hypothesis, since B_0 does have the prescribed structure, so we only need to prove the induction step. Assume that the theorem is satisfied for $i-1$ and consider two unitary matrices Q_i and Z_i, each made of a sequence of rotations, such that $B_i = Q_i B_{i-1} Z_i^*$ has the i-th column in Hessenberg form. Since both Q_i and Z_i are of the form $Q_i = I_i \oplus \hat{Q}_i$ and $Z_i = I_i \oplus \hat{Z}_i$ we have that for every $n \times n$ matrix X the relation $\mathcal{R}^i Q_i X = \hat{Q}_i \mathcal{R}^i X$ holds (and the same can be shown for Z_i). We have

$$\begin{aligned}\mathcal{R}^{i-1}B_i\mathcal{R}^{*,i-1} &= \mathcal{R}^{i-1}Q_iB_{i-1}Z_i^*\mathcal{R}^{*,i-1}\\ &= \mathcal{R}^{i-1}Q_iT_{i-1}Z_i^*Z_i(D_{i-1}+t(U_{Z,i-1}S_{i-1}^*)+t(a_{i-1}b_{i-1}^*))Z_i^*\mathcal{R}^{*,i-1}\\ &+\mathcal{R}^{i-1}Q_iU_{i-1}(\mathcal{R}^{i-1}Z_iW_{i-1})^*.\end{aligned}$$

We can now apply Theorem 3.3.5 to perform the conjugation by Z_i of the internal block. This leads to the following for some a_i and b_i such that the first i component of b_i are equal to 0 (given the structure of Z_i and by applying Lemma 3.3.3):

$$\begin{aligned}\mathcal{R}^{i-1}B_i\mathcal{R}^{*,i-1} &= \mathcal{R}^{i-1}T_i(D_i+t(U_{Z,i}\tilde{S}_i^*)+t(Z_ia_{i-1}b_{i-1}^*Z_i^*)+t(a_ib_i^*))R^{*,i-1}\\ &+\mathcal{R}^{i-1}U_iW_i^*\mathcal{R}^{*,i-1}.\end{aligned}$$

This is almost in the sought form except for an additional term of quasiseparable rank 1 given by $t(Z_ia_{i-1}b_{i-1}^*Z_i^*)$. We show here that this term can be absorbed in the previous one. Recall that $\mathcal{R}^{i-1}T_i\mathcal{R}^{*,i-1}$ is invertible so that we can mutliply on the left by its inverse and obtain

$$\begin{aligned}\mathcal{R}^{i-1}T_i^{-1}B_i\mathcal{R}^{*,i-1} &= \mathcal{R}^{i-1}(D_i+t(U_{Z,i}\tilde{S}_i^*)+t(Z_ia_{i-1}b_{i-1}^*Z_i^*)+t(a_ib_i^*))R^{*,i-1}\\ &+\mathcal{R}^{i-1}U_{Z,i}\tilde{W}_i^*\mathcal{R}^{*,i-1}\end{aligned}$$

thanks to the properties of the $\mathcal{R}$ operator and to Lemma 3.4.10. Since B_i has the i-th column in Hessenberg form and $X\mathcal{R}^{*,i-1}e_2 = Xe_i$ for any matrix X we can right multiply by e_2 and obtain that, for a vector x and two scalars d, f with $f \neq 0$

$$\begin{aligned}\mathcal{R}^{i-1}T_i^{-1}B_i\mathcal{R}^{*,i-1}e_2 &= \alpha e_1 + \beta e_2\\ &= de_1 + \mathcal{R}^{i-1}U_{Z,i}x + R^{i-1}Z_ia_if.\end{aligned}$$

Applying the operator $\mathcal{R}^i$ on both sides of $T_i^{-1}B_i$ yields

$$\mathcal{R}^iT_i^{-1}B_i\mathcal{R}^{*,i}e_1 = \beta e_1 = \mathcal{R}^iU_{Z,i}x + \mathcal{R}^iZ_ia_if$$

so that $\mathcal{R}^iZ_ia_i = \frac{\beta}{f}e_1 - \frac{1}{f}U_{Z,i}x$. For this reason we can write

$$\mathcal{R}^it(Z_aa_ib_i^*Z_i^*)\mathcal{R}^{*,i} = \mathcal{R}^it(-\frac{1}{f}U_{Z,i}xb_i^*Z_i^*)$$

since the first row of A is ignored when computing $t(A)$. In particular, we can obtain the desired representation of $\mathcal{R}^i B_i \mathcal{R}^{*,i}$:

$$\begin{aligned}\mathcal{R}^i B_i \mathcal{R}^{*,i} &= \mathcal{R}\mathcal{R}^{i-1} B_{i-1} \mathcal{R}^{*,i-1} \mathcal{R}^* = \\ &= \mathcal{R}^i T_i \mathcal{R}^{*,i} \mathcal{R}\mathcal{R}^{i-1} T_i^{-1} B_i \mathcal{R}^{*,i-1} \mathcal{R}^* = \\ &= \hat{T}_i \mathcal{R}^i (D_i + t(U_{Z,i} \tilde{S}_i^*) + t(Z_i a_{i-1} b_{i-1}^* Z_i^*) + t(a_i b_i^*)) R^{*,i} + \mathcal{R}^i U_i W_i^* \mathcal{R}^{*,i} \\ &= \hat{T}_i (\hat{D}_i + t(\hat{U}_{Z,i} (\mathcal{R}^i (\tilde{S}_i - \frac{1}{f} Z_i b_{i-1} x^*))^*) + t(\hat{a}_i \hat{b}_i^*)) + \hat{U}_i \hat{W}_i^* \\ &= \hat{T}_i (\hat{D}_i + t(\hat{U}_{Z,i} \hat{S}_i^*) + t(\hat{a}_i \hat{b}_i^*)) + \hat{U}_i \hat{W}_i^*\end{aligned}$$

by setting $\hat{S}_i = \mathcal{R}^i(\tilde{S}_i - \frac{1}{f} Z_i b_{i-1} x^*)$. This matrix is exactly in the form required by the theorem, and so the proof is complete. □

The application of Theorem 3.4.7 completes the analysis of the structures present in the iteration of the HTR. Nevertheless, it is not clear if, in this form, the above formulas provide a bound on the quasiseparability rank of the matrices involved.

We show here that this is the case. Notice that we are still considering only the case of A invertible, since both Theorem 3.4.7 and the following lemma require this. However, we will generalize these results to the non invertible case in Section 3.4.4.

We need some technical lemma before going on:

Lemma 3.4.11. *Let T_i and B_i be two matrices defined as above. Then we have*

$$\mathrm{lr}(B_i) \leqslant j+1, \qquad \mathrm{ur}(B_i) \leqslant 2j+k+1$$

Proof. Notice that we can write

$$\hat{B}_i = \hat{T}_i(\hat{D}_i + t(\hat{U}_{Z,i}\hat{S}_i^*) + t(\hat{a}_i\hat{b}_i^*) + \hat{T}_i^{-1}\hat{U}_i\hat{W}_i^*) = \hat{T}_i(\hat{D}_i + t(\hat{U}_{Z,i}\hat{S}_i^*) + t(\hat{a}_i\hat{b}_i^*) + \hat{U}_{Z,i} S \hat{W}_i^*).$$

This shows that the matrix inside the brackets has QS rank at most $(j+1, 2j+1)$. Multiplying by the upper triangular matrix does not increase the lower QS rank so the bound for the quasiseparability lower rank is $j+1$. Since $\hat{T}_i = \mathcal{R}^i Q_i Z_i^* \mathcal{R}^{*,i} + \hat{U}_i X_A Y^*$ for some Y we have

$$\hat{B}_i = \mathcal{R}^i Q Z^* (D_i + t(U_{Z,i} S_i^*) + t(a_i b_i^*) + U_{Z,i} S W_i^* + U_{Z,i} X_A Y^*) \mathcal{R}^{*,i}$$

and the inner matrix has QS ranks at most $(j+1, 2j+1)$. Given that $Q_{i:0} Z_{0:i}^*$ has rank at most (k, k) we obtain the desired bound. □

3.4.4 *Extension to the singular case*

In the characterization of the ranks involved in the transformations the fact that T is invertible plays an important role. Indeed, this section is devoted to the generalization of the previous results when T is singular and we will see that the rank bound is degraded by the rank deficiency of T. More precisely, in the following we assume that

$$A = I + U_A V_A^*, \qquad B = D_B + U_B V_B^*, \qquad \mathrm{rank}(A) = n - s.$$

Given that A is a rank k perturbation of the identity we must have $s \leqslant k$. In this case we can apply the same procedure for the reduction to upper triangular form of the matrix A. A direct analysis shows that the matrices computed by the algorithm described in Strategy (2) give us a matrix of the form

$$Q_0 A Z_0^* = \begin{bmatrix} T & W^* \\ 0 & I_{n-k} \end{bmatrix}, \qquad W \in \mathbb{C}^{n \times k} \tag{3.18}$$

where T is a $k \times k$ upper triangular matrix. It is clear that T must have rank $k-s$, so that we expect a portion of T to be 0. In order to make the algorithm efficient we want to control the position of these zeros and so we apply the following result.

Lemma 3.4.12. *Let $A = I + U_A V_A^*$ be a matrix of rank $n-s$ with $s \leqslant k$ as above. Then there exist two unitary matrices Q and Z such that $Z = \mathcal{M}(\mathcal{G})$ where $\mathcal{G}$ is a k-sequence of Givens rotations and*

$$\Pi Q A Z^* = \begin{bmatrix} T & W^* \\ 0 & I_{n-k} \end{bmatrix}, \qquad T = \begin{bmatrix} 0 & \dots & 0 & \times & \dots & \times \\ & \ddots & \vdots & \vdots & & \vdots \\ & & 0 & \times & \dots & \times \\ & & & & \hat{T} & \\ & & & & & \end{bmatrix}$$

where $\hat{T}$ is upper triangular and invertible and Π is a permutation matrix.

Proof. We can construct a matrix $Q_{0,1} = \mathcal{M}(\mathcal{G}_1)$ where $\mathcal{G}_1$ is a k-sequence of Givens rotations such that

$$\mathcal{G}_1^* U_A = \begin{bmatrix} R_A \\ 0 \end{bmatrix}$$

where R_A is square and upper triangular. We set $Z_{0,1} := Q_{0,1}$ so we have

$$Q_{0,1} A Z_{0,1}^* = Q_{0,1} A Q_{0,1}^* = \begin{bmatrix} I_k + R_A V_A^* Q_{0,1}^* & \times \\ 0 & I_{n-k} \end{bmatrix} = \begin{bmatrix} X & Y \\ 0 & I_{n-k} \end{bmatrix}.$$

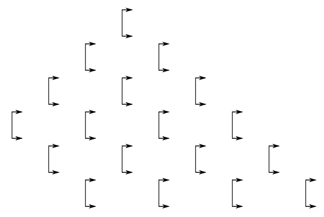
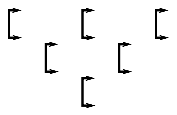
where the symbol $\times$ is used to denote non-zero elements and X and Y are two matrices of appropriate dimensions.

We can now multiply the above matrix on the right by Givens rotations in order to make X antitriangular with all the elements in the bottom-right part. We can start to annihilate the nonzero elements in the top-left position and then proceed row by row. We have then $Z_{0,2} = \hat{Z}_{0,2} \oplus I_{n-k}$ and

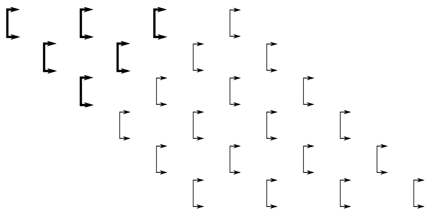
$$Q_{0,1} A Z_{0,1}^* Z_{0,2}^* = \begin{bmatrix} X\hat{Z}_{0,2}^* & Y \\ 0 & I_{n,k} \end{bmatrix}, \qquad X\hat{Z}_{0,2}^* = \begin{bmatrix} 0 & \dots & 0 & \times \\ \vdots & \iddots & \iddots & \vdots \\ 0 & \iddots & & \vdots \\ \times & \dots & \dots & \times \end{bmatrix}.$$

We can then set $\hat{\Pi}$ the permutation matrix associated with the permutation that maps i to $k-i$ and we have that left multiplying by $\Pi = \hat{\Pi} \oplus I_{n-k}$ yields the required structure. Moreover, when performing the antitriangular decomposition of X we can left-multiply by a permutation matrix Π_2 (swapping the rows) in order to make it rank-revealing and so we also have the zeros as depicted in the thesis. We can then show that the matrix QAZ^* has the prescribed structure if $Q = \Pi\Pi_2 Q_{0,1}$ and $Z = Z_{0,2} Z_{0,1}$.

It remains to show that $Z_{0,2}Z_{0,1}$ can be written as $\mathcal{M}(\mathcal{G})$ with $\mathcal{G}$ product of k 1-sequences. We show this fact pictorially, since it describes much better how this works. For this example we fix $k = 4$. We have that, by construction (the order of the rotation can be obtained by the procedure described above)

$Z_{0,1} =$  , $Z_{0,2} =$ 

where the rotations of $Z_{0,2}$ have been printed in bold to distinguish them and the highest rotation of $Z_{0,2}$ acts on the same row of the highest rotation of $Z_{0,1}$. Taking the product yields

$Z_{0,2}Z_{0,1} =$ 

which is in fact the composition of k (in our case 4) sequences of rotations. This completes the proof. □

We can use the above theorem to compute the matrices Q_0 and Z_0 needed for the initial transformation. In this way the structure of $Q_0AZ_0^*$ is the same of Equation (3.18) but the triangular matrix T has s zeros in the first diagonal elements.

We will show in Lemma 3.4.14 that also $Q_0BZ_0^*$ has the a particular structure similar to the one that is present in the non singular case. Moreover, we will show that this structure can be used to prove the same results on rank conservation, even if degraded of a constant factor s.

For this we introduce the following matrix J_s.

Lemma 3.4.13. *Let $J_s^{(n)}$ be the $n \times s$ matrix defined by*

$$J_s^{(n)} = \begin{bmatrix} I_s \\ 0 \end{bmatrix}$$

and let T_i, Q_i and Z_i be the triangular matrix obtained at the i-th step of the Hessenberg triangular reduction and the related unitary transformations. Assume that T_0 has exactly s zeros on the diagonal, all in the top left position. The index n will be omitted whenever it is clear from the context. Then we have

- T_i *has no more than s zeros on the diagonal, and they are packed in the top left positions.*
- *For all $i = 0, \dots, n-2$ there exists a matrix $X_i \in \mathbb{C}^{n \times (s+i)}$ such that* $Q_{i:1}(T_0 + J_sJ_s^*)Z_{1:i}^* = T_i + J_{s+i}X_i^*$.

- *For every* i *the matrix* $T_i + J_s J_s^*$ *is invertible, and there exist two* $n \times k$ *matrices* Y_1, Y_2, Y_3 *such that*

$$(T_i + J_s J_s^*)^{-1} U_i = U_{Z,i} Y_1^* + J_{s+i} Y_2^*, \qquad (T_i + J_s J_s^*)^{-1} T_i = I - J_s Y_3^* = I - J_{s+i} \begin{bmatrix} Y_3 \\ 0 \end{bmatrix}^*$$

- *For every positive integer* s, i *such that* $i < n - s$ *the following relation holds:* $\mathcal{R}^i J_{s+i}^{(n)} \mathcal{R}^{*,i} = J_s^{(n-i)}$.

Proof. Suppose that T_0 has s zeros in the top-left diagonal entries. We prove that if $e_i^t T_0 e_j \neq 0$ then $e_i^t T_i e_j \neq 0$. Consider a pair of rotations (G, H) acting on the same consecutive rows $(k, k+1)$ such that $G T_0 H^*$ is still upper triangular. The diagonal elements of $G T_0 H^*$ that differs from the one of T_0 can be computed by looking at the two rows where the rotations are acting:

$$G \begin{bmatrix} d_k & u_k \\ 0 & d_{k+1} \end{bmatrix} H^* = \begin{bmatrix} \hat{d}_k & \hat{u}_k \\ 0 & \hat{d}_{k+1} \end{bmatrix}.$$

If $d_k d_{k+1} \neq 0$ then the same must hold for $\hat{d}_k \hat{d}_{k+1}$ since the two upper triangular matrix must have the same rank. Moreover, if $d_k = 0$ then the first column is equal to 0 and so the same holds for the first column of the matrix left multiplied by G. Since the right rotation H is computed in order to restore the triangular structure and the matrix is already triangular we have $H = I$, thus we have proved the first statement. Notice that this might not be true for any choice of two matrices G and H in the equation above but it holds for the rotations computed by the HTR algorithm.

Now that we have characterized the structure of the zeros on the diagonal of T_i it follows that $T_i + J_s J_s^*$ is singular since the first s diagonal entries of T_i are the only one being equal to zero, and we are correcting them with the sum of $J_s J_s^*$. Moreover, we can note that

$$\mathcal{R}^s (T_i + J_s J_s^*)^{-1} \mathcal{R}^{*,s} = (\mathcal{R}^s (T_i + J_s J_s^*) \mathcal{R}^{*,s})^{-1} = (\mathcal{R}^s T_i \mathcal{R}^{*,s})^{-1}.$$

This implies that the rows and columns with index bigger than s in the inverse are independent of the shift $J_s J_s^*$ and are exactly equal to the inverse of the relative submatrix in T_i. This implies that

$$(T_i + J_s J_s^*)^{-1} T_i = I - J_s Y_3^*$$

for an appropriate Y_3^*. The results concerning $(T + J_s J_s)^* U$ follows by this fact and the previous lemmas for the invertible case. □

In this context, it is possible to prove a generalization of Theorem 3.4.7 that holds for a singular A. More precisely, we have the following:

Theorem 3.4.14. *Let* B_i *be the matrix obtained at the* i*-th step of the HTR algorithm. Suppose that the matrix* A *is singular, and has rank* $n - s$*, with some* $s \leqslant k$*. Then the matrix* $\hat{B}_i = \mathcal{R}^i B_i \mathcal{R}^{*,i}$ *can be written as*

$$\hat{B}_i = \hat{T}_i (D + t(\hat{U}_{Z,i} \tilde{S}_i^*) + t(a_i b_i^*)) + \hat{U}_i W_i^*$$

where U, S_i *and* W_i *are appropriate* $n \times (2k + s)$ *matrices and* $U_{Z,i}$ *and* U_i *are defined starting from* U *using Equation (3.16), as usual.*

Proof. As in Theorem 3.4.7, we prove the result by induction. First of all, we need to prove that B_0 has the prescribed form. We already know that with the choice of Q_0 and Z_0 of reduction strategy (2) this holds, thanks to Lemma 3.4.3. Recalling that $B_0 = QBZ_0^*$ we can follow the same procedure of Lemma 3.4.3 and we obtain

$$B_0 = Q_0BZ_0^* = Q_0Z_0^*Z_0BZ_0^* = T_0Z_0(D_0 + U_BV_B^*)Z_0^* + U_0W_0^*.$$

Since Z_0 is a product of k sequences of rotations we can perform the conjugation by $\tilde{Z}_0$ applying Lemma 3.4.3 which gives us, by setting $\tilde{U} = [U_A \;\; U_B]$,

$$B_0 = T_0(D_0 + t(Z_0\tilde{U}S_{0,1}^*)) + Q_0\tilde{U}W_0^*.$$

By setting $U = [\tilde{U} \;\; Z_0^{-1}J_s]$ we note that (thanks to Lemma 3.4.13) $U_{Z,i} = [U_{Z,0} \;\; J_s]$ so we can write

$$B_0 = T_0(D_0 + t(U_{Z,0}\tilde{S}_0^*)) + U_iW_i^*.$$

The term J_s is not actually needed in the representation of the matrix B_0, but we will need it in the next steps in order to deal with the rank deficiency of the upper triangular matrix.

Now we prove the inductive step. Let $B_i = Q_iB_{i-1}Z_i^*$ with the usual notation. We reapply the same arguments of Theorem 3.4.7 and we get that

$$\begin{aligned}\mathcal{R}^{i-1}B_i\mathcal{R}^{*,i-1} &= \mathcal{R}^{i-1}T_i(D_i + t(U_{Z,i}\tilde{S}_i^*) + t(Z_ia_{i-1}b_{i-1}^*Z_i^*) + t(a_ib_i^*))R^{*,i-1} \\ &+ \mathcal{R}^{i-1}U_iW_i^*\mathcal{R}^{*,i-1}.\end{aligned}$$

Now we want to show that the term $\mathcal{R}^iZ_ia_{i-1}$ can be written as $\mathcal{R}^iU_{Z,i}x + \alpha e_1$. We can left multiply on the left the above equation by the inverse of $\mathcal{R}^{i-1}(T_i + J_sJ_s^*)\mathcal{R}^{*,i-1}$ and on the right by e_2 so, by applying repeatedly Lemma 3.4.13, we get

$$\begin{aligned}\mathcal{R}^{i-1}(T_i + J_sJ_s^*)^{-1}B_i\mathcal{R}^{*,i-1}e_2 &= d_{i,1}e_1 + \mathcal{R}^{i-1}U_{Z,i}x_1 + \mathcal{R}^{i-1}Z_ia_{i-1}x_2 \\ &+ \mathcal{R}^{i-1}U_{Z,i}x_3 + \mathcal{R}^{i-1}J_{s+i}x_4.\end{aligned}$$

where y_1, x_1, x_3 and x_4 are appropriate vectors and x_2 is a scalar. The above multiplied on the left by the $\mathcal{R}$ operator implies that

$$\mathcal{R}^iZ_ia_{i-1} = J_s\tilde{x}_1 + \hat{U}_{Z,i}\tilde{x}_2.$$

To conclude we just need to show that J_s can be written as $\hat{U}_{Z,i}X$ for some X. This can be proved by induction. In fact, we know that this holds at the step 0 and if there exist X_{i-1} such that $\hat{U}_{Z,i-1}X_{i-1} = J_s$ we can write

$$\hat{U}_{Z,i}X_{i-1} = \mathcal{R}Z_iU_{Z,i-1}X_{i-1} = \mathcal{R}Z_iJ_s.$$

But the matrix Z_iJ_s is a rectangular irreducible upper Hessenberg matrix, since Z_i is a product of non trivial Givens rotations (see Lemma 3.3.8). Then $\mathcal{R}Z_iJ_s$ is an upper triangular matrix of size $(n-i) \times s$ with no zeros on the diagonal. If we call Y_i the top $s \times s$ block we have

$$\hat{U}_{Z,i}X_{i-1}Y_i^{-1} = \mathcal{R}Z_iJ_sY_i^{-1} = J_s.$$

So we can set $X_i = X_{i-1}Y_i^{-1}$ and we can conclude the proof by following the same steps of Theorem 3.4.7. □

The above concludes the analysis in the singular case. We have shown that, even if with some work and in a more involved framework compared to the previous section, it is possible to bound the QS rank increase during the reduction process.

3.4.5 *Working on the upper triangular part*

We have already shown that T_0 has a very sparse structure, since only the first k rows differ from the identity. In this section we prove that $\hat{T}_i$ for $i > 0$ has the same structure.

Lemma 3.4.15. *Let $Q = G_2 \ldots G_{n-1}$ be a unitary matrix that can be written as a product of Givens rotations G_i such that G_i acts on the rows $(i, i+1)$. Let T_0 an upper triangular matrix with all the off-diagonal elements on the rows with index bigger than k equal to 0 and the diagonal elements of modulus 1. If Z is another unitary matrix with the same structure of Q such that $T_1 = QT_0Z^*$ is still upper triangular, then T_1 has the off-diagonal elements with row index bigger than $k+1$ equal to zero.*

Proof. It suffices to note that, for every $i > k$ the rotation G_i multiplying on the left is acting on a 2×2 diagonal and unitary matrix. Let this 2×2 matrix be S. We use the notation G_iS to restrict the product of G_i to this submatrix. If $\tilde{G}_i$ is another Givens rotation such that $G_iS\tilde{G}_i$ is upper triangular then, being unitary, it must be diagonal. This implies that $G_{k+1} \ldots G_nT_0$ still has the same structure of T_0. A direct inspection shows that $G_k \ldots G_nT_0\tilde{G}_n \ldots \tilde{G}_k$ has the property that all the off-diagonal elements with row index bigger than $k+1$ are zero and the same holds also when multiplying by $G_2, \ldots, G_{k-1}$ on the left and $\tilde{G}_2, \ldots, \tilde{G}_{k-1}$ on the right. This proves that T_1 has the desired structure. □

Corollary 3.4.16. *The matrix $\hat{T}_1 = \mathcal{R}QT_0Z^*\mathcal{R}^*$ has the same structure of T_0, that is, all the off-diagonal elements with row index bigger than k are zero.*

Proof. Using the previous Lemma we have that T_1 has all the off-diagonal elements with row index bigger than $k+1$ equal to zero. The property requested by the thesis is obtained if we truncate T_1 removing the first row and column. □

Remark 3.4.17. The above Lemma can be applied as it is to $\hat{T}_1$ to prove that $\hat{T}_2$ has the same sparse structure. We can iterate the process by induction and show that the structure is shared by any $\hat{T}_i$ for $i > 0$.

We consider now the problem of updating T_i to obtain T_{i+1}. To this end, we perform the following block partitioning of T_i:

$$T_i = \begin{bmatrix} T_i^{(1)} & M_{l,i}M_{u,i}^* \\ 0 & I_{n-i-1} \end{bmatrix}, \quad M_{l,i}, M_{u,i} \in \mathbb{C}^{(i+1)\times 2k}, \; T_i^{(1)} \in \mathbb{C}^{(i+1)\times(i+1)}.$$

Recall that $T_{i+1} = Q_iT_iZ_i^*$ and that Q_i and Z_i are of the form $I_i \oplus \hat{Q}_{n-i}$ and $I_i \oplus \hat{Z}_{n-i}$, respectively. In particular, only $O(k)$ rotations act on the block $T_i^{(1)}$ so that its update costs only $O(nk)$ flops. Moreover, even if we have $O(n)$ rotations acting on the rank $2k$ block, each can be applied only with $O(k)$ operations. This leads to a total cost of the update of $O(nk)$ operations. Notice that the above applies to the case where T is invertible, otherwise the complexity grows to $(n(2k+s))$, but since $s \leqslant k$ this is still bounded by $O(nk)$.

3.4.6 *Parametrizing the Hessenberg part*

In this section we show how to parametrize the partial Hessenberg matrix B_i computed at each step of the algorithm. In the previous section we have shown that the quasiseparability rank is bounded during the steps and we have also given a possible factorized representation in Equation (3.17). Here we recall it for the sake of clarity:

$$\hat{B}_i = \hat{T}_i(\hat{D}_i + t(\hat{U}_{Z,i}\hat{S}_i^*) + t(\hat{a}_i\hat{b}_i^*)) + U_i W_i^*.$$

It is worth to recall that this representation only gives the part of the matrix that still need to be reduced to upper Hessenberg form, and ignores the rest. This is enough in order to carry out the algorithm and compute the required unitary rotations, but will not give the final matrix as an output at the end. For this reason, the matrix H needs to be recovered a posteriori. A strategy to carry out this operation is described in Section 3.4.7

In the representation of $\hat{B}_i$ we have three different terms that will be handled differently:

- $\hat{T}_i$ is the trailing part of T_i whose representation has already been described in Section 3.4.5. For this reason it is not necessary to further discuss it here, since we already know how to update it.
- The Hermitian matrix $\hat{D}_i + t(\hat{U}_i\hat{S}_i^* + t(\hat{a}_i\hat{b}_i^*))$ needs to be updated at each step by left and right multiplying it by Z_i and Z_i^*, respectively. For this operation we refer to the algorithm described in Section 3.3, where this operation is performed with the use of Givens–Vector representations. It shall be stressed that the cost of the algorithm presented there is $O(nk)$ flops per step.
- The low-rank part can be updated by simply multiplying its factors by the Givens rotations. Since they are tall and thin $n \times k$ matrices, this costs $O(nk)$ per step, within the desired cost bound of the algorithm.

Note that, at the i-th step, it is possible to store the diagonal and subdiagonal elements of the partial Hessenberg matrix B_i. In fact, these two elements will not change anymore during the algorithm and so they are equal to the diagonal and subdiagonal elements of $H = B_{n-2}$. This will be useful in the next section to recover the full matrix H.

3.4.7 *Recovering the full matrix*

The algorithm that we have presented can track the structure of the trailing principal submatrix that still need to be reduced and carry out the transformation until complete Hessenberg triangular reduction. However, from this formulation it is not completely clear how to recover the Hessenberg matrix QBZ^* at the end of the process.

To this end we consider the matrix QBZ^* obtained at the final reduction step so that we have

$$H = QBZ^* = QZ^*ZD_BZ^* + QU_BV_B^*Z^* = T_{n-2}ZD_BZ^* + QU_BV_BZ^* - QU_AV_AD_BZ^*.$$

It is worth to note that the above formula expresses H as $T_{n-2}S$ plus a low rank correction where S is Hermitian. Moreover, note that is possible to compute the matrix V_AD_B at the start of the algorithm and then perform the update of the low-rank parts at $O(nk)$ cost per step. Thus, at the end, we have that for some V_{n-2},

$$H = T_{n-2}S + U_{n-2}V_{n-2}^* \tag{3.19}$$

where U_j is as defined in the previous Section and has size $n \times 2k$. If T_{n-2} is invertible then it is possible to write

$$T_{n-2}^{-1}H - T_{n-2}^{-1}U_{n-2}V_{n-2}^{*} = S$$

and note that, since $T_{n-2}H$ is upper Hessenberg and its diagonal and subdiagonal elements can be computed in linear time, we have a direct expression for the lower part of S. Being S Hermitian, this provides a direct expression for its upper part, too. This allows to use Equation (3.19) to represent the final matrix H. Note that, since T_{n-2} is quasiseparable, this also provides a structured representation of H.

Moreover, it shall be stressed that the solution of the linear system $T_{n-2}^{-1}U_{n-2}$ can be avoided since

$$T_{n-2}^{-1}U_{n-2} = Z_{n-2} \dots Z_1 T_0^{-1} U_0$$

and this linear system can be easily solved in $O(k^3)$ time given the structure of T_0. Then it is sufficient to update the solution left multiplying by Z_i at each step of the algorithm and thus obtain the desired $T_{n-2}^{-1}U_{n-2}$ at the end.

A slightly more involved strategy must be used when T_{n-2} is singular. Since we cannot invert it directly, we shall left multiply Equation (3.19) by the inverse of $T_{n-2} + J_s J_s^*$.

We get the following relation

$$S = (T_{n-2} + J_s J_s^*)^{-1} H - (T_{n-2} + J_s J_s^*)^{-1} R - J_s Y_3^* S$$

and the computation of $(T_{n-2} + J_s J_s^*)^{-1} R$ only requires $O(n^2 k)$ operation since it requires k back-substitutions. Here we have the solution of a slightly more costly triangular system (given the $O(n^2k)$ complexity instead of the $O(k^3)$ of the previous strategy) but still within the cost bound that we hoped for. Notice that, since Y_3 is not known, the above allows to compute the last $n-k$ rows of S. Given the symmetry of S we can recover the whole matrix but the top-left leading $k \times k$ matrix. To solve this issue we can store the full $k \times k$ leading minor of the matrix H as it is computed. This is not costly since it requires $O(k^3)$ flops and after k steps that part of the matrix is not modified anymore.

3.5 AN ALTERNATIVE HESSENBERG REDUCTION STRATEGY

In this section we present an alternative strategy for the computation of the upper Hessenberg form for a diagonal plus low rank matrix $A = D + UV^*$. We make the following remark.

Remark 3.5.1. Despite the common usage of the article "the" in front of the expression "Hessenberg form", the upper Hessenberg form of a matrix A is not unique. What can be guaranteed is that if the first column of the conjugating matrix Q is fixed then $H = Q^*AQ$ is uniquely determined up to a conjugation by a diagonal matrix D with diagonal elements of modulus 1. This result is known in the literature as the implicit Q theorem and it is the basis of the implicit QR iteration [91].

This result guarantees that as soon as the first column of Q is chosen the matrix H is essentially determined. However, the result can be also seen from another viewpoint: since the Hessenberg form is not unique, we can try to choose the initial transformation in order to provide an easy reduction process.

The process that we introduce in this section is heavily based on Givens rotations so we borrow the syntax of Section 3.3 for Givens rotations and sequences of Givens rotations.

The algorithm is composed of the following parts:

REDUCTION TO BANDED FORM The matrix $A = D + UV^*$ is reduced to banded plus low rank form $Q_1 A Q_1^* = B + U_1 V_1^*$ where B has bandwidth k and

$$U_1 = \begin{bmatrix} X_1 \\ 0 \end{bmatrix}, \qquad X_1 \in \mathbb{C}^{k \times k}.$$

The unitary matrix Q_1 is obtained as $\mathcal{M}(\mathcal{G}^*)$ where $\mathcal{G}$ is a k-sequence of Givens rotations. We show that this transformation can be computed in $O(n^2 k)$ flops.

REDUCTION TO HESSENBERG FORM We reduce the matrix $Q_1 A Q_1^*$ to upper Hessenberg form by annihilating subdiagonal elements one at a time using Givens rotations from the left. This operation degrades the k-banded structure of the matrix so we restore it using other rotations from the right. We can show that this can be carried out in $O(k^2)$ operations, and so the total cost of the reduction is $O(n^2 k)$.

Notice that by applying a preliminary transformation Q_1 to the matrix A we alter the first column and so we have that the final matrix H will be completely different from the one computed by the classical Hessenberg reduction schemes based on Givens rotations or Householder reflectors and by the procedure described in Section 3.3. However, in the context of finding eigenvalues it is not relevant which Hessenberg reduction form is computed as soon as it is obtained through unitary transformations (in order to not deteriorate the conditioning of the eigenvalue problem).

3.5.1 *Reduction to banded form*

In this section we describe the first of the two steps required for the Hessenberg reduction: the reduction of A to banded plus low rank. The algorithm that we report for this task is inspired by Algorithm 2 of [3].

Suppose that $U, V \in \mathbb{C}^{n \times k}$. Assume k is fixed and consider the map ℓ defined by

$$\begin{array}{rccc} \ell(i,j): & \mathbb{N}^2 & \longrightarrow & \mathbb{N} \\ & (i,j) & \longmapsto & k(n+j-i)+j \end{array}.$$

If we set $I = \{1, \dots, n\} \times \{1, \dots, k\} \subseteq \mathbb{N}^2$ we have that $\ell_{|I}$ is an injective map. In fact, we can easily verify that if $\ell(i,j) = \ell(i',j')$ with $(i,j), (i',j') \in I$ then

$$0 = \ell(i,j) - \ell(i',j') = k((j-j') - (i-i')) + j - j'.$$

In particular we have $j \equiv j' \mod k$ and since they are both such that $1 \leqslant j, j' \leqslant k$ then we must have $j = j'$. Then $i = i'$ follows by back substitution. This allows to define an order on I by inheriting the order on $\mathbb{N}$ through the map ℓ. That is, we define the total order $\leqslant_\ell$

$$(i,j) \leqslant_\ell (i',j') \iff \ell(i,j) \leqslant \ell(i',j').$$

Our purpose is to set the elements of U to zero using Givens rotations starting from the elements larger with respect to the order $\leqslant_\ell$. Pictorially we obtain the following pattern for the appearance of the zeros:

$$\begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix} \to \begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ 0 & \times & \times \end{bmatrix} \to \begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \end{bmatrix} \to \begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \end{bmatrix} \to \begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \end{bmatrix} \to \dots \qquad (3.20)$$

We obtain this result by applying Givens rotations on the left. Since we are transforming the whole matrix A the same rotations have to be applied to the diagonal matrix D, thus losing the diagonal structure. Since we are conjugating the matrix A we also need to apply the same transformations on the right. Those will alter the entries of V but this is not a problem since we do not have any requirement on the structure of V. Nevertheless, we want to understand what happens to the matrix $Q_1 D Q_1^*$. Before going on we want to better characterize the rotations in Q_1.

Algorithm 7 Algorithm for the reduction of $A = D + UV^*$ to $Q_1 A Q_1^* = B + U_1 V_1^*$ where B is banded with bandwidth k and $U_1 = [X_1\ 0]^t$ with $X_1 \in \mathbb{C}^{k \times k}$.

```
1: function BandReduction(D,U,V)
2:     $U_1 \leftarrow U$
3:     $V_1 \leftarrow V$
4:     $B \leftarrow D$
5:     for i = n : -1 : 2 do
6:         for j = 1, $\min\{k, n-i+1\}$ do
7:             $l \leftarrow i+j-1$
8:             $G \leftarrow$ GivensRotation($U_1[l-1:l, j]$)
9:             $U_1 \leftarrow G * U_1$
10:            $V_1 \leftarrow G * V_1$
11:            $B \leftarrow G * B * G^*$
12:            for s = l+k : k : n do
13:                $G \leftarrow$ GivensRotation($B[s-1:s, s-k-1]$)
14:                $V_1 \leftarrow G * V_1$        ▷ No need to update $U_1$ since has zeros on these entries
15:                $B \leftarrow G * B * G'$
16:            end for
17:        end for
18:    end for
19:    return $(B, U_1, V_1)$.
20: end function
```

We notice that the rotations can be written as a sequence of rotations in the sense of Definition 3.2.8. In particular, notice that the rotations needed to transform the matrix U in upper triangular form are $G_{i,j}$ with

$$(i,j) \in \tilde{I} := \{(i,j) \in I \mid j = 1, \dots, k, \quad j \leqslant i \leqslant n-1\}.$$

The set $\tilde{I}$ is ordered with the order induced by $\leqslant_\ell$, so that we can define $\mathcal{G} = (G_{i,j})_{(i,j) \in \tilde{I}}$, and we have that $Q_1 = \mathcal{M}(\mathcal{G}^*)$.

However, it is easy to verify that if we perform the algorithm as it is, we end up with U being in the required form but $Q_1 D Q_1^*$ being a full matrix. We propose the following solution, similar to what is found in [3]: we consider the matrix D as a (very special) Hermitian banded matrix with bandwidth k. When applying the rotations we will preserve the banded structure of the matrix D by left multiplying by appropriately chosen rotations that act on the rows where U has already been set to 0. This way we have that both $Q_1 U$ and $Q_1 D Q_1^*$ have the required structure. The pseudocode describing the reduction is reported in Algorithm 7.

$$\begin{bmatrix} \times & \times \\ \times & \times \\ \times & \times \\ \times & \times \\ \times & \times \\ \times & \times \end{bmatrix} \begin{bmatrix} \times & 0 & 0 & 0 & 0 & 0 \\ 0 & \times & 0 & 0 & 0 & 0 \\ 0 & 0 & \times & 0 & 0 & 0 \\ 0 & 0 & 0 & \times & 0 & 0 \\ 0 & 0 & 0 & 0 & \times & 0 \\ 0 & 0 & 0 & 0 & 0 & \times \end{bmatrix} \to \begin{bmatrix} \times & \times \\ \times & \times \\ \times & \times \\ \times & \times \\ \times & \times \\ 0 & \times \end{bmatrix} \begin{bmatrix} \times & 0 & 0 & 0 & 0 & 0 \\ 0 & \times & 0 & 0 & 0 & 0 \\ 0 & 0 & \times & 0 & 0 & 0 \\ 0 & 0 & 0 & \times & 0 & 0 \\ 0 & 0 & 0 & 0 & \times & \times \\ 0 & 0 & 0 & 0 & \times & \times \end{bmatrix} \to \cdots$$

$$\cdots \to \begin{bmatrix} \times & \times \\ \times & \times \\ \times & \times \\ \times & \times \\ 0 & \times \\ 0 & \times \end{bmatrix} \begin{bmatrix} \times & 0 & 0 & 0 & 0 & 0 \\ 0 & \times & 0 & 0 & 0 & 0 \\ 0 & 0 & \times & 0 & 0 & 0 \\ 0 & 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times & \times \end{bmatrix} \to \begin{bmatrix} \times & \times \\ \times & \times \\ \times & \times \\ \times & \times \\ 0 & \times \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \times & 0 & 0 & 0 & 0 & 0 \\ 0 & \times & 0 & 0 & 0 & 0 \\ 0 & 0 & \times & 0 & 0 & 0 \\ 0 & 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times & \times \end{bmatrix} \to \cdots$$

$$\cdots \to \begin{bmatrix} \times & \times \\ \times & \times \\ \times & \times \\ 0 & \times \\ 0 & \times \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \times & 0 & 0 & 0 & 0 & 0 \\ 0 & \times & 0 & 0 & 0 & 0 \\ 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times \end{bmatrix} \to \begin{bmatrix} \times & \times \\ \times & \times \\ \times & \times \\ 0 & \times \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \times & 0 & 0 & 0 & 0 & 0 \\ 0 & \times & 0 & 0 & 0 & 0 \\ 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times \end{bmatrix} \to \cdots$$

$$\cdots \to \begin{bmatrix} \times & \times \\ \times & \times \\ \times & \times \\ 0 & \times \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \times & 0 & 0 & 0 & 0 & 0 \\ 0 & \times & 0 & 0 & 0 & 0 \\ 0 & 0 & \times & \times & \times & 0 \\ 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & 0 & \times & \times & \times \end{bmatrix} \to \begin{bmatrix} \times & \times \\ \times & \times \\ 0 & \times \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \times & 0 & 0 & 0 & 0 & 0 \\ 0 & \times & \times & \times & \times & 0 \\ 0 & \times & \times & \times & \times & 0 \\ 0 & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \\ 0 & 0 & 0 & \times & \times & \times \end{bmatrix} \to \cdots$$

$$\cdots \to \begin{bmatrix} \times & \times \\ \times & \times \\ 0 & \times \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \times & 0 & 0 & 0 & 0 & 0 \\ 0 & \times & \times & \times & \times & 0 \\ 0 & \times & \times & \times & \times & 0 \\ 0 & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \\ 0 & 0 & 0 & \times & \times & \times \end{bmatrix} \to \begin{bmatrix} \times & \times \\ \times & \times \\ 0 & \times \\ 0 & \times \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \times & 0 & 0 & 0 & 0 & 0 \\ 0 & \times & \times & \times & 0 & 0 \\ 0 & \times & \times & \times & \times & 0 \\ 0 & \times & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & 0 & \times & \times & \times \end{bmatrix} \to \cdots$$

$$\cdots \to \begin{bmatrix} \times & \times \\ 0 & \times \\ 0 & \times \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \times & \times & \times & \times & 0 & 0 \\ \times & \times & \times & \times & 0 & 0 \\ \times & \times & \times & \times & \times & 0 \\ \times & \times & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & 0 & \times & \times & \times \end{bmatrix} \to \begin{bmatrix} \times & \times \\ 0 & \times \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \times & \times & \times & \times & 0 & 0 \\ \times & \times & \times & \times & 0 & 0 \\ \times & \times & \times & \times & \times & 0 \\ \times & \times & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & 0 & \times & \times & \times \end{bmatrix} \to \cdots$$

$$\cdots \to \begin{bmatrix} \times & \times \\ 0 & \times \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \times & \times & \times & 0 & 0 & 0 \\ \times & \times & \times & \times & 0 & 0 \\ \times & \times & \times & \times & \times & 0 \\ 0 & \times & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & 0 & \times & \times & \times \end{bmatrix} \to \begin{bmatrix} \times & \times \\ 0 & \times \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \times & \times & \times & 0 & 0 & 0 \\ \times & \times & \times & \times & 0 & 0 \\ \times & \times & \times & \times & \times & 0 \\ 0 & \times & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & 0 & \times & \times & \times \end{bmatrix}.$$

Figure 3.7: Steps needed for the reduction of a matrix $A = D + UV^*$ to banded plus low rank in the case of $n = 6$ and $k = 2$. Here we report the evolution of the matrix U and of the diagonal matrix D. The matrices are displayed side by side so it is easy to apply the rotations at the same time.

The outer for loop iterates over the row indices of the matrix U. For each row the first inner loop walks along the diagonals to clean the elements of U, as specified by the order $\leqslant_\ell$ and pictorially shown by Equation (3.20). The second inner loop, instead, cleans the bulges that have been introduced on the banded matrix B by the rotations. Notice that if $i > n-k$ the latter loop does nothing, since no indices are included in the set $i+k : k : n$, so for the first $k-1$ steps that code is not executed at all. In Figure 3.7 we show the evolution of the matrices U_1 and B in the first steps of the reduction algorithm for $k = 2$. The pictorial representation shows how the zeros in the matrix U_1 allow to recover the banded structure in the matrix B. Even if it is not reported, the matrix V_1 needs to be updated in the same way of U_1.

We note that the second loop has the index s that runs from $i+k$ to n with step k. This is motivated by the fact that when cleaning the element on row $i+k$ on the $(k+1)$-th subdiagonal the multiplication on the right by the same rotations creates a bugle k diagonal entries below. We use the loop to chase all the bulges until the end of the matrix. This step is shown in the last two transformations of the pictorial representation.

Notice that Algorithm 7 consists of $O(nk \cdot \frac{n}{k}))$ rotations since the outer loop is executed $n-1$ times while the inner ones are executed $O(k)$ times and $O(\frac{n}{k})$, respectively. Each rotation has a cost of $O(k)$ flops. In fact this is the cost of left multiplying U_1 and V_1 that have k columns and also the cost of conjugating B that has a band structure that never exceeds bandwidth $k+1$.

This implies that the total cost of the reduction is $O(n^2k)$. Since our purpose is to design an algorithm with total cost bounded by $O(n^2k)$ we are satisfied with this result.

3.5.2 *Reduction to Hessenberg form of a banded plus low rank matrix*

We now show how to perform the actual Hessenberg reduction. We suppose to have $A_1 := Q_1AQ_1^* = B + U_1V_1^*$ computed with the algorithm of the previous subsection and we follow a slightly modified version of the usual reduction algorithm based on Givens rotations.

We perform the following steps:

- We compute a sequence of k rotations $\mathcal{G}$ so that $\mathcal{G}^*A_1e_1 = \alpha e_1 + \beta e_2$ for some α, β. We apply the rotation separately to B and $U_1V_1^*$ so that we compute $\mathcal{G}^*B\mathcal{G}$ and $\mathcal{G}^*U_1, \mathcal{G}^*V_1$.

- At this point $\mathcal{G}^*B$ is no more banded with bandwidth k but has a bulge of order k on the columns from 2 to k and on the rows from $3+k$ to $1+2k$. Notice that this is the same of saying that $\mathcal{G}^*A_1\mathcal{G}$ has the bulge since $\mathcal{G}^*U_1$ has only the first $k+1$ rows different from zero. We compute a sequence of Givens rotations to remove the bulge from the left and we right and left multiply $\mathcal{G}^*A_1\mathcal{G}$ by them. This cleans the bulge but at the same time another one is created k entries below. We continue to chase the bulge until the end of the matrix.

- We repeat the process on the trailing matrix obtained by removing the first row and column. Before continuing with the recursion we store the diagonal and subdiagonal elements of the first column.

The first and the last point of the above algorithm are quite clear, since they are exactly the same of the standard Hessenberg reduction process.

The middle one, however, needs to be studied more carefully. First of all, given the structure of A_1, we note that $\mathcal{G} = \{G_2, \ldots, G_k\}$. Most of the first column, in fact, is already set to 0 and so we only need $O(k)$ rotations to put it in Hessenberg form. We also notice that G_kU_1 makes the $k+1$ row of U_1 different from zero, but all the other rotations do not change the structure of U_1, since they are acting on rows above the $(k+1)$-th. Moreover, since the bulge starts from row $3+k$ the rotations needed to clean it do not alter the zero structure of U_1. For this reason we only need to care about A_1 when chasing the bulge, and consequently update V_1.

We notice that, being the bulge a $k \times k$ matrix, it is not trivial how it can be removed by only $O(k)$ rotations. This is actually possible since the bulge itself has a quasiseparable structure of QS rank 1, inherited by the fact that it is created by a sequence of rotations. However, there is no need to compute a representation for the quasiseparable structure in order to remove it. It is much simpler to avoid its creation in the first place. In practice, we follow this pattern to apply the rotations in $\mathcal{G}$ that clean the first column:

- We apply the last rotation G_k. This creates a bulge in position $(2k+1, k)$.
- We clean the above bulge with a rotation from the left. This step will create another bulge k entries below, and we chase it until the end of the matrix.
- We do the same applying the rotations G_j with $j < k$ that will create a bulge in position $(k+j+1, j)$ that we clean with a rotation from the left and the required bulge chasing.

The above strategy allows to make the bulge chasing operation $O(k^2)$ per bulge, since only $O(k)$ rotations are involved and applying each one only costs $O(k)$ flops. Since we have $O(\frac{n}{k})$ bulges to chase we get a total cost per column of $O(\frac{n}{k}) \cdot O(k^2) = O(nk)$ and so a total cost for this part of the algorithm of $O(n^2 k)$.

As we have said, at each step we do not track the entire structure but we only store the diagonal and subdiagonal elements of the Hessenberg form. Since we are also tracking the low rank factors U_1 and V_1 properly updated with the rotations, in the end we are able to recover the whole matrix H along with its rank structure.

Lemma 3.5.2. *The Hessenberg form H computed through unitary rotations satisfies the following relation:*

$$H - H^* = U_2 V_2^* - V_2 U_2^*$$

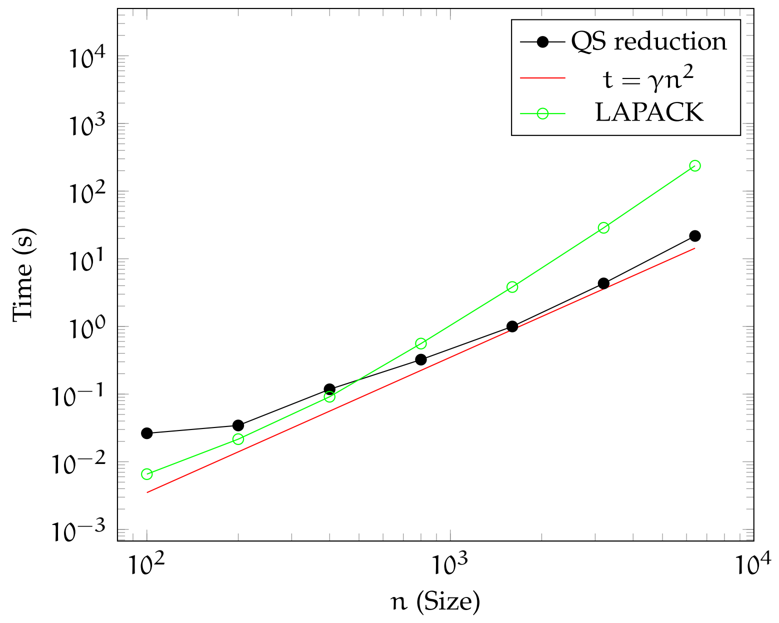
where $U_2 = Q_2 Q_1 U$ and $V_2 = Q_2 Q_1 V$.

Proof. Recall that $H = Q_2 Q_1 (D + UV^*) Q_1^* Q_2^*$. Since $Q_2 Q_1 D Q_1^* Q_2^*$ is Hermitian it is not present in the difference $H - H^*$ and so we have the thesis. □

Using the above result and the fact that we know the lower part of H we can recover the upper part. In fact, we have

$$\operatorname{tril}(H) - \operatorname{tril}(H^*) = \operatorname{tril}(H - H^*) = \operatorname{tril}(U_2 V_2^* - V_2 U_2^*)$$

and since $\operatorname{tril}(H^*) = \operatorname{triu}(H)^*$ we have $\operatorname{triu}(H) = \operatorname{tril}(H)^* - \operatorname{tril}(U_2 V_2^* - V_2 U_2^*)^* = \operatorname{tril}(H)^* - \operatorname{triu}(V_2 U_2^* - U_2 V_2^*)$. This implies that the upper Hessenberg form that we have found is the sum of a tridiagonal matrix and the upper triangular part of a low rank one (which has rank $2k$).



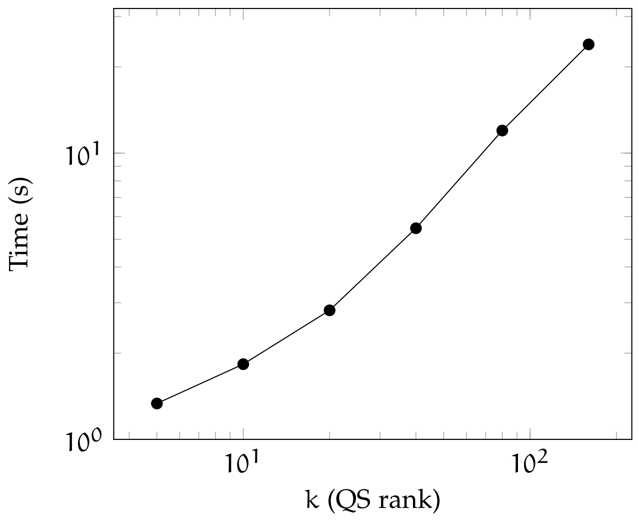
Size	QS (s)	LAPACK (s)
100	0.026	0.006
200	0.034	0.022
400	0.12	0.091
800	0.32	0.56
1600	1.00	3.82
3200	4.32	28.68
6400	21.72	237.12

Figure 3.8: CPU time, in seconds, for the Hessenberg reduction of a diagonal plus rank 10 matrix of size n. Here the line is the plot of γn^2 for an appropriate γ. It is evident the quadratic behavior of the time.

3.5.3 *Numerical experiments*

In this section we report some numerical experiments that we have run in order to validate the approach. The code has been written in the FORTRAN language and exposed to MATLAB and GNU Octave through a MEX file, for easier testing. We have run it on a laptop with an Intel(R) Core(TM) i3-2367M CPU running at 1.40GHz and 4 GB of RAM. Moreover, it will soon be available at `http://numpi.dm.unipi.it/software/` for testing.

We have checked that the complexity is really quadratic in the size and linear in the rank in Figure 3.8 and Figure 3.9, respectively. In Figure 3.8 we compare our algorithm with the standard Hessenberg reduction on a problem with quasiseparable rank equal to 10. The classical algorithm is faster for $n < 800$. From the plot we might guess that the two approaches are comparable at around $n \approx 500$. Using the above results we can estimate that the contant in front of the complexity $O(n^2k)$ of our algorithm is more or less 60 times bigger than the one in fron of the $O(n^3)$ in front of the classical Hessenberg reduction strategy. This can be considered as a good result because the implementation of our method is not blocked and so does not take fully advantage of vectorization. A possible way to improve the result could be to rely in Householder reflections and operate on blocks. We think that this possibility deserves further study. The flops count for Householder-based Hessenberg reduction is just $\frac{10}{3}n^3$, so the constant is very small. In our case, we have to compute and apply about $O(n^2k)$ rotations to at least 3 different matrices. Combining this with the fact that Givens rotations tend to be more expensive than Householder reflections we have a motivation for the fact that our constant is bigger.



QS Rank	Time (s)
5	1.33
10	1.83
20	2.82
40	5.46
80	11.99
160	23.97

Figure 3.9: CPU time, in seconds, for the Hessenberg reduction of a 2000×2000 diagonal plus rank k matrix.

Size	Relative error
160	4.65e-15
320	5.57e-15
640	1.19e-14
1280	8.70e-15

Table 3.1: Relative error (in the 2-norm) of the eigenvalues computed starting from the Hessenberg matrix H obtained by the structured quasiseparable reduction. The eigenvalues have been compared to the ones computed by the function `eig` in GNU Octave.

Figure 3.9 shows that the complexity is linear in the rank as soon as it is large enough, and it exhibits a slight sublinear behavior before. This is likely due to better performance of the algorithm when a more dense linear algebra is performed operating on bigger bulges.

Moreover, we have tested the accuracy of the approaches by computing the eigenvalues of the Hessenberg matrix computed with this strategy and comparing them to the eigenvalues of the full matrix A. The results are reported in Table 3.1. The error is the relative error (in the 2-norm) of the vector containing the eigenvalues. The matrices have been generated using the `randn` function of GNU Octave, that is by writing them as a sum of a random diagonal matrix plus the product of two random $n \times k$ matrices.

3.6 AN APPLICATION TO QUADRATIC MATRIX EQUATIONS

The purpose of this section is to investigate the use of quasiseparable representations in the solution of some special quadratic matrix equations. A much more detailed analysis of these results (along with other extensions) is available in [16].

We consider the problem of finding the invariant probability measure of a QBD (Quasi Birth and Death) Markov chain. This is a stochastic process often found in the probabilistic analysis of queues. It models situations where the space state is divided in levels and each level contains a certain number of substates. Transitions are possible only to adjacent levels, while the internal transitions of the level might be arbitrary. This kind of restriction gives a probability transition matrix of the form

$$P = \begin{bmatrix} \hat{A}_0 & A_1 & & \\ A_{-1} & A_0 & A_1 & \\ & \ddots & \ddots & \ddots \\ & & \ddots & \ddots & \ddots \end{bmatrix}$$

since typically the set of levels is isomorphic to $\mathbb{N}^+$ and the transitions inside the levels and from one to the other are independent of the specific level. Thus we have a block Toeplitz structure with substochastic blocks (except for the top-left block, that needs to be different to account for the fact that we cannot move left since it represents the lowest possible level). The computation of the invariant vector can be rephrased in terms of the solution of the following matrix equation [18]:

$$X = A_{-1} + A_0 X + A_1 X^2, \qquad A_i \in \mathbb{R}^{m \times m}.$$

The minimal non negative solution G of the matrix equation is guaranteed to exist in this context and can be obtained applying the cyclic reduction (see [18, 17]) that defines an iteration on four matrices given by setting $A_i^{(0)} = A_i$ for $i = -1, 0, 1$, $\hat{A}_0^{(0)} := \hat{A}_0$ and defining $\hat{A}_i^{(\ell)}$ and $\hat{A}_0^{(\ell)}$ by the following formulas:

$$\begin{aligned} A_i^{(k+1)} &= \begin{cases} A_i^{(\ell)}(I - A_0^{(\ell)})^{-1} A_i^{(\ell)} & \text{if } i \neq 0 \\ A_0^{(\ell)} + A_{-1}^{(\ell)}(I - A_0^{(\ell)})^{-1} A_1^{(\ell)} + A_1^{(\ell)}(I - A_0^{(\ell)})^{-1} A_{-1}^{(\ell)} & \text{if } i = 0 \end{cases} \\ \hat{A}_0^{(k+1)} &= \hat{A}_0^{(\ell)} + A_1^{(\ell)}(I - A_0^{k})^{-1} A_{-1}^{(\ell)} \end{aligned}$$

At each step an approximation of the solution G can be given as $G^{(\ell)} := (\hat{A}_i^{(\ell)})^{-1} A_1^{(\ell)}$. If the above iteration converges then $G^{(\ell)}$ goes to the solution G as ℓ goes to infinity.

In [18] a thorough analysis of the algorithm is carried out for the solution of Quasi-Birth and Death Markov chains (in short: QBDs). A natural example in this setting is to consider random walks on $\{0, \ldots, n\} \times \mathbb{N}^+$. In this case the states are of the form (i, l) where l is the level and i is the index of the substate. Allowing to move only to adjacent levels and to adjacent substates imposes a tridiagonal structure on the blocks A_i.

The fact that the matrices A_i represent probability measures implies that they are non-negative and the sum $A_{-1} + A_0 + A_1$ is row stochastic, that is,

$$(A_{-1} + A_0 + A_1)e = e, \qquad e = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}. \tag{3.21}$$

We want to deal with situations where this additional structure is available. The case of tridiagonal blocks is theoretically analyzed by Myazawa [79]. The approach that we present here can be easily generalized to the case of k-quasiseparable coefficients, but in order to keep the exposition simple we restrict our attention to tridiagonal matrices. In the general case, we might want to answer the following questions:

- Does the quasiseparable structure of A_i imply that also the solution of the matrix equation is quasiseparable?
- Is the quasiseparable structure maintained during the iterations of the cyclic reduction? If it is, is it possible to exploit this fact in order to speed up the algorithm?

We can answer positively to both questions, provided we make the right assumptions and we look at them in the right framework. In this context we will focus mainly on studying the structure of the matrices A_i since that is the ingredient needed to lower the cost of the iteration. In order to analyze these structures we introduce the matrix polynomial $\varphi(z)$ defined as

$$\varphi(z) := A_{-1} + z(A_0 - I) + z^2 A_1.$$

In the same spirit we define the function $\varphi^{(\ell)}(z)$ for every $\ell \geqslant 0$ by setting

$$\varphi^{(\ell)}(z) := A_{-1}^{(\ell)} + z(A_0^{(\ell)} - I) + z^2 A_1^{(\ell)}.$$

It is immediate to note that $\varphi^{(0)}(z) \equiv \varphi(z)$. We also define $\psi(z) = z\varphi(z)^{-1}$ and $\psi^{(\ell)}(z) = z(\varphi^{(\ell)}(z))^{-1}$, whenever this makes sense[1]. Notice that Equation (3.21) guarantees that 1 is always an eigenvalue of $\varphi(z)$, since $\varphi(1)e = 0$.

We have the following important relationship, proved in [18]:

Theorem 3.6.1. *Let $A_i^{(\ell)}$ be the matrices involved in the CR iteration and $\varphi^{(\ell)}(z)$ and $\psi^{(\ell)}(z)$ defined as above. Then, for every $\ell \geqslant 0$,*

$$\psi^{(\ell+1)}(z^2) = \frac{1}{2}\left(\psi^{(\ell)}(z) + \psi^{(\ell)}(-z)\right). \tag{3.22}$$

Recalling that $\varphi^{(\ell)}(z) = z\psi^{(\ell)}(z)^{-1}$ we can retrieve a representation for $\varphi^{(\ell)}(z)$ as

$$\varphi^{(\ell)}(z^{2^\ell}) = z^{2^\ell}\frac{1}{2^\ell}\left(\sum_{j=0}^{2^\ell-1}\psi(\zeta_{2^\ell}^j z)\right)^{-1} \tag{3.23}$$

where ζ_n is any n-th primitive root of the unity. This fact can be used to give an apparently negative answer to the second question that we have asked.

Remark 3.6.2. Recalling that the QS rank is maintained under inversion, the QS rank of $A_i^{(\ell)}$ can be bounded by $C \cdot 2^k$ where C is an appropriate constant. In fact, we know that by Equation (3.23) the matrix function $\varphi^{(\ell)}(z)$ can be written as the average of 2^k terms that have QS rank 1, and this gives the thesis on the matrix function $\varphi(z)$. Moreover, the coefficients of $A_i^{(\ell)}$ can be retrieved from $\varphi^{(\ell)}(z)$ through the following interpolation formulas:

$$A_{-1}^{(\ell)} = \frac{1}{3}\left(\varphi^{(\ell)}(\zeta_6) + \varphi^{(\ell)}(\zeta_6^5) + \varphi^{(\ell)}(-1)\right) \tag{3.24}$$

$$A_0^{(\ell)} = \frac{\zeta_6^5}{2}\left(\varphi^{(\ell)}(\zeta_6) - \varphi^{(\ell)}(-\zeta_6)\right) \tag{3.25}$$

$$A_1^{(\ell)} = \frac{1}{3}\left(\zeta_6^2\varphi^{(\ell)}(\zeta_6^5) + \zeta_6^4\varphi^{(\ell)}(\zeta_6) + \varphi^{(\ell)}(-1)\right) \tag{3.26}$$

[1] The matrix function $\psi(z)$ can be defined for every z such that $\varphi(z)$ is invertible, that is, for each z that is not part of the spectrum of $\varphi(z)$.

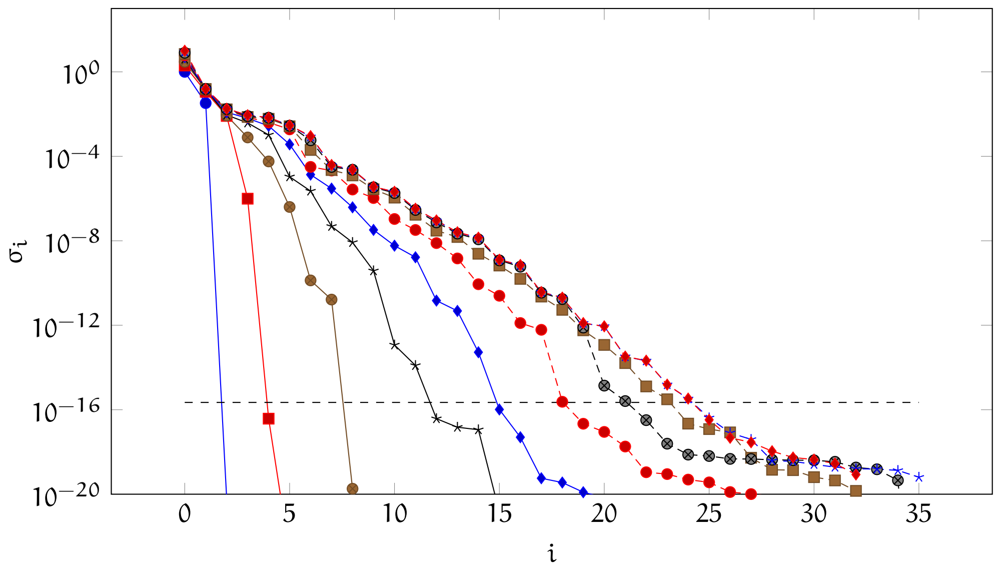


Figure 3.10: Plot of the most significant singular values of an offdiagonal matrix of the middle coefficient $A_0^{(\ell)}$ generated by the CR iteration applied to a problem of size 1600. Each line in the plot shows the most significant singular values for a different value of ℓ.

where, as usual, ζ_6 is a primitive 6-th root of the unity. This implies that if $\varphi^{(\ell)}(z)$ is quasiseparable for any $z \in \mathbb{C}$ the same must hold for its coefficients, even if we might need to put a constant in front of the QS rank.

The above estimate is rather pessimistic, since it provides only an exponential bound to the increase of the QS rank of the coefficients. However, numerical experiments show that this bound is strict, at least in early iterations. Analyzing the QS rank of the subsequent iterations show that the offdiagonal submatrices of $\varphi^{(\ell)}(z)$ are, in general, full rank. Nevertheless, they exhibit an exponential decay in the singular values and so they can be approximated very well by a quasiseparable matrix. In Figure 3.10 we have reported the plot of the most significant singular values of a offdiagonal matrix of $\varphi^{(\ell)}(z)$ for $A_i \in \mathbb{C}^{1600 \times 1600}$.

Figure 3.10 shows that the singular values tend to be arranged on a straight line in the logarithmic scale, and so they exhibit an exponential decay. Moreover, it is clear from this example that the singular values σ_i drop below a threshold comparable to the unit roundoff u at a value of i of about 25. The machine precision is reported as an horizontal line in the plot.

This suggests that even if the off-diagonal matrices have theoretically full rank they are numerically of rank about (at most) 25, at least in this example. The natural question is: how can we exploit this fact?

The quasiseparable representations that we have used for the Hessenberg reduction can handle only the exact QS rank and it is not easy to represent approximations of non quasiseparable matrices. In our experiments we have found hierarchical representations of [26] that we have surveyed in Section 3.1 to be the most convenient for this purpose.

In fact, in hierarchical representations, we have that the off-diagonal blocks are represented as outer products UV^* with U and V $n \times k$ matrices with k larger than the QS rank. As we will see, it is easy to obtain these factors U and V from SVD decompositions that allow to construct QS approximations given a certain truncation threshold.

This section is divided in three parts:

(i) In Section 3.6.1 we prove that the behavior of the singular values that we have found in the numerical experiments is linked with some spectral properties of the matrix function $\varphi(z)$.

(ii) In Section 3.6.2 we show that the characterization of the structure generalizes to the case where a shift is applied to the matrix polynomial, an operation sometimes needed in order to improve the convergence of the method.

(iii) In Section 3.6.3 we show how it is possible to use $\mathcal{H}$-matrices in order to design a fast version of the CR algorithm for this kind of matrices.

3.6.1 *Exponential decay of the singular values*

Here we deal with the following problem: given a class of matrix functions $\psi(z)$ (which in our case will be the inverses of $\varphi(z)$ associated with a QBD process) with the same spectral properties we prove that the singular values of off-diagonal submatrices exhibit an exponential decay that is maintained under CR iterations.

In this context we assume to be working with a non null-recurrent process, that is a kind of regular case for Markov chains. The general analysis of the decay for null-recurrent QBDs is more involved. These results are extended to those cases in [16].

We report the following useful properties of the matrix polynomial $\varphi(z)$. A proof of these results can be found in [17] and [57].

Lemma 3.6.3. *Let $\varphi(z) = A_{-1} + zA_0 - zI + z^2A_1$ be the matrix polynomial associated with the QBD process with blocks equal to A_{-1}, A_0 and A_1. We have that the following properties hold in the non null recurrent case.*

(i) The real number 1 is always an eigenvalue of $\varphi(z)$. Moreover, the matrix polynomial $\varphi(z)$ has another positive real eigenvalue ρ such that one of the two conditions is satisfied.

- *$\rho < 1$ and the annulus $\mathcal{A} = \{z \mid \rho < |z| < 1\}$ does not contain any eigenvalue of $\varphi(z)$. Moreover, ρ is the spectral radius of the minimal nonnegative solution G of the matrix equation $A_{-1} + A_0X + A_1X^2 = X$. In this case we set $\rho_{min} := \rho$ and $\rho_{max} := 1$.*
- *If $\rho > 1$ and the annulus $\mathcal{A} = \{z \mid 1 < |z| < \rho\}$ does not contain any eigenvalue of $\varphi(z)$. Moreover, ρ is inverse of the spectral radius of the minimal nonnegative solution R of the matrix equation $X^2A_{-1} + XA_0 + A_1 = X$. In this case we set $\rho_{min} := 1$ and $\rho_{max} = \rho$.*

(ii) If we call $A(z)$ and $D(z)$ any diagonal block of $\varphi(z)$ in the partitioning

$$\varphi(z) = \begin{bmatrix} A(z) & B(z) \\ C(z) & D(z) \end{bmatrix}$$

then $A(z)$ and $D(z)$ are invertible in the annulus $\mathcal{A}$ defined above.

(iii) The real numbers 1 and ρ are the unique solution to the equation $\rho(rI - \varphi(r)) = r$ where ρ is the spectral radius and $r \in \mathbb{R}^+$. In particular, in the interval (ρ_{min}, ρ_{max}) we have $\rho(rI - \varphi(r)) \leqslant r$ so that $\rho(I - z^{-1}\varphi(z)) \leqslant 1$ for any z contained in the annulus $\mathcal{A}$.

Remark 3.6.4. In order to make the exposition simpler we would like to have the annulus $\mathcal{A}$ of the form

$$\mathcal{A}_t = \left\{ z \mid t \leqslant |z| \leqslant \frac{1}{t} \right\}, \qquad \rho < 1.$$

We can achieve this by setting $\hat{\varphi}(z) := \varphi(z\sqrt{\rho_{max}\rho_{min}})$ so that we have the invertibility of $\hat{\varphi}(z)$ in annulus as defined above by setting $t := \sqrt{\frac{\rho_{min}}{\rho_{max}}}$.

In view of the above remark in the following we will always assume that $\varphi(z)$ is already in the form of $\hat{\varphi}(z)$, without the need of rescaling. In particular, all the results that we prove are independent of the scaling introduced on the matrix function $\varphi(z)$. Moreover, while most of the following results hold in the quasiseparable case, here we only consider the problem of tridiagonal coefficients.

Remark 3.6.5. We notice that the matrix function $\varphi(z)$ that we are interested in has a rather particular form, since it can be written as

$$z^{-1}\varphi(z) = I - (z^{-1}A_{-1} + A_0 + zA_1), \qquad A_i \text{ positive and}$$

for $i = -1, 0, 1$. In particular, $\rho(z^{-1}A_{-1} + A_0 + zA_1) < 1$ for any $z \in \mathcal{A}_t$ so that by setting $K(z) = z^{-1}A_{-1} + A_0 + zA_1$ we can write

$$z\varphi(z)^{-1} = \sum_{j=0}^{\infty} K(z)^j.$$

We can partition $K(z)$ according to the partitioning that we have proposed on $\varphi(z)$ so that we have

$$z^{-1}\varphi(z) = I - K(z) = \begin{bmatrix} I & \\ & I \end{bmatrix} - \begin{bmatrix} A_K(z) & B_K(z) \\ C_K(z) & D_K(z) \end{bmatrix}.$$

$K(z)$ is defined by non negative coefficients so, for any z, $|K(z)| \leqslant K(|z|)$. In particular, for any $z \in \mathcal{A}_t$ we have that both $A_K(z)$ and $D_K(z)$ have a spectral radius smaller than $K(|z|)$. In fact, by relying on the Perron-Frobenius theorem we have

$$\rho(A_K(z)) = \rho\left(\begin{bmatrix} A_K(z) & 0 \\ 0 & 0 \end{bmatrix}\right), \qquad \begin{bmatrix} |A_K(z)| & 0 \\ 0 & 0 \end{bmatrix} \leqslant \begin{bmatrix} A_K(|z|) & 0 \\ 0 & 0 \end{bmatrix} \leqslant K(|z|).$$

that implies $\rho(A_K(z)) \leqslant \rho(K(|z|))$. Since $\rho(K(|z|)) < 1$ in the annulus $\mathcal{A}_t$ we can claim that the same holds for $A_K(z)$ and $D_K(z)$. In particular, both $A(z) = I - A_K(z)$ and $D(z) = I - D_K(z)$ are invertible and it holds

$$|A(z)^{-1}| \leqslant \sum_{j=0}^{\infty} |A_K(z)|^j \leqslant \sum_{j=0}^{\infty} K(|z|)^j = z\varphi(|z|)^{-1},$$

and the same for $D(z)^{-1}$. The monotonicity of the Euclidean matrix norm allows to conclude that

$$\|A(z)^{-1}\|_2 \leqslant \|z\varphi(|z|)^{-1}\|_2, \qquad \|D(z)^{-1}\|_2 \leqslant \|z\varphi(|z|)^{-1}\|_2$$

Lemma 3.6.6. *Let $\varphi(z)$ be defined as above. Assume that $\varphi(z)$ is invertible in the annulus $\mathcal{A}_t$. Then there exist two holomorphic matrix functions $u(z)$ and $v(z)$ of size $n \times k$ such that*

$$\varphi(z) = \begin{bmatrix} A(z) & B(z) \\ C(z) & D(z) \end{bmatrix} \implies z\varphi(z)^{-1} = \begin{bmatrix} \times & \times \\ u(z)v^t(z) & \times \end{bmatrix}.$$

where k *is the quasiseparability rank of* $\varphi(z)$. *Moreover,* $u(z)$ *and* $v(z)$ *satisfy the following constraints:*

$$\|u(z)\|_2 \leqslant \|\psi(z)\|_2, \qquad \|v(z)\|_2 \leqslant \|\psi(z)\|_2.$$

where C *is the constant of Remark 3.6.5.*

Proof. Having that $\varphi(z)$ is tridiagonal we can write $C(z) = f(z)e_{j+1}e_j^t$ for an appropriate value of j. Choosing $\tilde{u}(z) := f(z)e_{j+1}$ and $\tilde{v}(z) := e_j$ allow us to write $C(z) = u(z)v^t(z)$. Moreover, we have that $|f(z)|$ is bounded by $\max\{\rho_{max}, \frac{1}{\rho_{min}}\}$ as a consequence of the non-negative and stochasticity properties. We can then write

$$\begin{bmatrix} A(z) & B(z) \\ \tilde{u}(z)\tilde{v}^t(z) & D(z) \end{bmatrix} \begin{bmatrix} I & \\ -D^{-1}(z)C(z) & I \end{bmatrix} = \begin{bmatrix} S_D(z) & B(z) \\ 0 & D(z) \end{bmatrix}$$

where $S_D(z)$ is the Schur complement. In particular, given the invertibility of $\varphi(z)$ in the annulus we have that both $D(z)$ and $S_D(z)$ are invertible and taking the inverses on both sides yields that the block in the bottom left corner of the inverse $z\varphi^{-1}(z)$ is given by $D^{-1}(z)C(z)S_D^{-1}(z) = D^{-1}(z)\tilde{u}(z)\tilde{v}^t(z)S_D^{-1}(z)$. Setting

$$u(z) := D^{-1}(z)\tilde{u}(z), \qquad v(z) = S_D^{-t}(z)\tilde{v}(z)$$

gives the thesis.

We can now prove that the bounds on the norms are satisfied. Notice that $S_D^{-1}(z)$, being an element of the inverse of $z^{-1}\varphi(z)$, is bounded in norm by $\|\psi(z)\|_2$. Moreover, the bounds on the inverse of $D(z)$ given in Remark 3.6.5 guarantee that

$$\|D^{-1}(z)\|_2 \leqslant z\|\psi(z)\|_2$$

and so we can retrieve the following bound on the norm of $u(z)$ and $v(z)$:

$$\|u(z)\|_2 \leqslant \|\psi(z)\|_2, \qquad \|v(z)\|_2 \leqslant \|\psi(z)\|_2.$$

□

Theorem 3.6.7. *Let* $\psi(z)$ *be the* 1*-quasiseparable matrix function obtained as the inverse of a tridiagonal matrix function* $\varphi(z)$ *on the annulus* $\mathcal{A}_t$ *for some* $t > 1$. *We assume that* $\varphi(z)$ *and* $\psi(z)$ *are associated with a QBD process according to the notation used above. If* $C(z)$ *is an off-diagonal block of* $\psi^{(\ell)}(z)$ *then its singular values* $\sigma_j(C(z))$ *are such that*

$$\sigma_j(C(z)) \leqslant C_t \cdot \sup_{z \in \mathcal{A}_t} \|\psi(z)\|_2^2 \cdot e^{-\alpha(t)j},$$

where C_t *a moderate computable constant which depends on* t, $\alpha(t) = \frac{1}{2}\log(t)$ *and* $\psi^{(\ell)}(z)$ *is defined according to Equation (3.22).*

We want to emphasize that our aim is to derive some decay result that is only dependent on the spectral data of $\psi(z)$, that is, on requiring that its eigenvalues are outside of a given annulus. In particular, notice that if t is chosen in a way that $\mathcal{A}_t$ is the maximum annulus of invertibility $\psi(z)$ will be unbounded on this open set. On the other hand, if t is chosen too small the exponential decay will be slow. As usual with this kind of bounds, the value of t will need to be set looking for a fair trade-off between the decay speed and the size of the constant. The rest of this section is devoted to prove the above result.

Remark 3.6.8. If Theorem 3.6.7 holds then each offdiagonal matrix function $C(z)$ of $\psi(z)$ has numerical rank (with threshold equal to u) bounded by

$$\operatorname{rank}_u(C(z)) \leqslant \frac{\max_{z\in\mathcal{A}_t}\log(\|\psi(z)\|_2) - \log(u) + \log(C_t)}{\alpha(t)}$$

where the above formula is obtained by requiring that $\sigma_j \leqslant u \cdot \sigma_1$ and assuming that in z the norm of $C(z)$ is not too far from the maximum attained on the circle of $\|\psi(z)\|_2$.

Our analysis of the decay of the singular values is strictly connected with the matrix functions $\varphi^{(\ell)}(z)$ and $\psi^{(\ell)}(z)$. For this reason we need to understand the relation of the singular values of the offdiagonal submatrices of a matrix with the ones of its inverse.

Lemma 3.6.9. *Let A and B be two $n \times n$ matrices. Let $\sigma_j(A)$ denote the j-th singular value of A sorted in decreasing order, and the same for B. If B is invertible it holds that*

$$\frac{\sigma_j(A)}{\|B^{-1}\|_2} \leqslant \sigma_j(AB) \leqslant \|B\|_2\sigma_j(A), \qquad \frac{\sigma_j(A)}{\|B^{-1}\|_2} \leqslant \sigma_j(BA^*) \leqslant \|B\|_2\sigma_j(A).$$

Proof. Recall that the singular values of a matrix A are the square roots of the eigenvalues of the Hermitian matrix A^*A. We can write

$$A = U_A\Sigma_A V_A^*, \qquad B = U_B\Sigma_B V_B^*$$

where U_A, V_A, U_B and V_B are unitary matrices and Σ_A and Σ_B are diagonal with the singular values on the diagonal. Since the singular values are invariant for the multiplication by unitary matrices we have that, by setting $Q = V_A^* U_B$, we can write

$$\sigma_j(AB)^2 = \max_{\substack{\dim(V)=j\\ V\subseteq\mathbb{R}^n}} \min_{x\in V} \frac{x^*\Sigma_B^* Q^*\Sigma_A^*\Sigma_A Q\Sigma_B x}{x^*x} = \max_{\substack{\dim(V)=j\\ V\subseteq\mathbb{R}^n}} \min_{x\in V} \frac{(\Sigma_B x)^* Q^*\Sigma_A^2 Q(\Sigma_B x)}{x^*x}$$

which is obtained by writing the j-th eigenvalue of ABB^*A^* by using the Rayleigh quotient. By setting $y = \Sigma_B x$ and recalling that Σ_B must be invertible by hypothesis we have

$$\sigma_j(AB)^2 = \max_{\substack{\dim(V)=j\\ V\subseteq\mathbb{R}^n}} \min_{y\in V} \frac{y^*Q^*\Sigma_A^2 Qy}{y^*y}\cdot\frac{y^*y}{x^*x}$$

so that by using the fact that Q is unitary and that $\frac{x^*x}{\|B^{-1}\|^2} \leqslant y^*y \leqslant \|B\|^2 x^*x$ we obtain

$$\frac{\sigma_j(A)^2}{\|B^{-1}\|_2^2} = \frac{\sigma_j(\Sigma_A)^2}{\|B^{-1}\|_2^2} \leqslant \sigma_j(AB)^2 \leqslant \|B\|_2^2\cdot\sigma_j(\Sigma_A)^2 = \|B\|_2^2\cdot\sigma_j(A)^2.$$

□

We are now ready to show that the singular values of the off-diagonal blocks of A do not change too much under inversion, given reasonable hypothesis on the matrix A.

Lemma 3.6.10. *Let A be an invertible matrix with the following block decompositions:*

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix}, \qquad A^{-1} = \begin{bmatrix} \tilde{A}_{1,1} & \tilde{A}_{1,2} \\ \tilde{A}_{2,1} & \tilde{A}_{2,2} \end{bmatrix}$$

where the corresponding blocks have the same sizes. It holds that *(i) If $A_{2,2}$ is invertible then*

$$\frac{1}{\|A_{2,2}\|_2 \|S_{A_{2,2}}\|_2} \sigma_k(A_{2,1}) \leqslant \sigma_k(\tilde{A}_{2,1}) \leqslant \|A_{2,2}^{-1}\|_2 \cdot \|S_{A_{2,2}}^{-1}\|_2 \cdot \sigma_k(A_{2,1})$$

where $S_{A_{2,2}} = A_{1,1} - A_{1,2}A_{2,2}^{-1}A_{2,1}$ is the Schur complement of $A_{2,2}$.

(ii) If $A_{1,1}$ is invertible then

$$\frac{1}{\|A_{1,1}\|_2 \|S_{A_{1,1}}\|_2} \sigma_k(A_{2,1}) \leqslant \sigma_k(\tilde{A}_{2,1}) \leqslant \|A_{1,1}^{-1}\|_2 \cdot \|S_{A_{1,1}}^{-1}\|_2 \cdot \sigma_k(A_{2,1})$$

where $S_{A_{1,1}} = A_{2,2} - A_{2,1}A_{1,1}^{-1}A_{1,2}$ is the Schur complement of $A_{1,1}$.

Proof. To prove the lemma we write the block inversion formula for A so that we get

$$A^{-1} = \begin{bmatrix} A_{1,1}^{-1} + A_{1,1}^{-1}A_{1,2}S_{A_{1,1}}^{-1}A_{2,1}A_{1,1}^{-1} & -A_{1,1}^{-1}A_{1,2}S_{A_{1,1}}^{-1} \\ S_{A_{1,1}}^{-1}A_{2,1}A_{1,1}^{-1} & S_{A_{1,1}}^{-1} \end{bmatrix}.$$

Since we have $\tilde{A}_{2,1} = S_{A_{1,1}}^{-1}A_{2,1}A_{1,1}^{-1}$ applying Lemma 3.6.9 two times yields

$$\frac{1}{\|A_{1,1}\|_2 \|S_{A_{1,1}}\|_2} \sigma_k(A_{2,1}) \leqslant \sigma_k(\tilde{A}_{2,1}) \leqslant \|A_{1,1}^{-1}\|_2 \cdot \|S_{A_{1,1}}^{-1}\|_2 \cdot \sigma_k(A_{2,1})$$

which is point (ii) of the thesis. The other can be obtained in the same way by just using the other inversion formula involving $S_{A_{2,2}}$. □

In order to obtain the estimates on the singular values we consider a decomposition of the form $A = \sum_{i=0}^{n} u_i v_i^*$. Notice that in the case where the u_i and the v_i form orthogonal basis this is exactly the SVD if we rescale u_i and v_i in order to have unitary norm and we take out a positive scalar σ_i. In practice, we will not be able to directly obtain such a decomposition, but instead we will handle the case where the u_i and v_i are not orthogonal. Nevertheless, it is still possible to bound the singular values, as the following lemma shows.

Lemma 3.6.11. *Let $A = \sum_{j \in \mathbb{Z}} A_j$ where the matrices A_j are of rank k. Assume that $\|A_j\|_2 \leqslant Me^{-\alpha|j|}$ for some positive constant j and $\alpha > 0$. Then we have that the singular values of A are bounded by*

$$\sigma_l(A) \leqslant \frac{2M}{1-e^{-\alpha}} e^{-\alpha \lceil \frac{l-k}{2k} \rceil}.$$

Proof. The idea of the proof is based on the fact that if B is a rank l approximation of A then $\sigma_{l+1} \leqslant \|A - B\|_2$. In fact, we know that the best rank l approximation of A is given by the truncated SVD where only the first l singular values and vectors have been considered. Since B cannot be a better approximation and the residual norm 2 of the truncated SVD is exactly σ_{l+1} we can conclude that the above inequality holds.

Notice that, for any l, if we set $I_{l-1} = \{j \in \mathbb{Z} \mid |j| < \lceil \frac{l-k}{2k} \rceil\}$ the matrix $B = \sum_{j \in I_{l-1}} A_j$ is a rank (at most) $l-1$ approximation to A. For this reason

$$\sigma_l(A) \leqslant \|B - A\|_2 \leqslant \sum_{j \in I_{l-1}^C} \|A_j\|_2 \leqslant M \sum_{k \in I_{l-1}^C} e^{-\alpha|j|} \leqslant \frac{2M}{1-e^{-\alpha}} e^{-\alpha \lceil \frac{l-k}{2k} \rceil}.$$

This concludes the proof. □

We now have all the required tools needed to give a proof of Theorem 3.6.7.

Proof of Theorem 3.6.7. Since $\psi(z)$ is analytic on the annulus $\mathcal{A}_t$ the same must hold for the off-diagonal block $C(z)$. Moreover, given the quasiseparability condition on $\psi(z)$, we know that we can write $C(z) = u(z)v(z)^t$ where $u(z)$ and $v(z)$ are vector functions as per Lemma 3.6.6. According to the Lemma we have the following bounds on their norm:

$$\|u(z)\|_2 \leqslant \|\psi(z)\|_2, \qquad \|v(z)\|_2 \leqslant \|\psi(z)\|_2.$$

Moreover, we know that $u(z)$ and $v(z)$ can be chosen analytic in this context (again thanks to Lemma 3.6.6) so that we can expand them in the Fourier basis:

$$u(z) = \sum_{k\in\mathbb{Z}} \hat{u}_k z^k, \qquad v(z) = \sum_{k\in\mathbb{Z}} \hat{v}_k z^k.$$

Thanks to [68, Theorem 4.4c] we know that the Fourier coefficients of analytic functions have an exponential decay given by their domain of definition, that in our case means

$$\|\hat{u}_j\|_\infty \leqslant \max_{|z|=\{t,\frac{1}{t}\}} \|u(z)\|_\infty e^{-\alpha|j|}, \qquad \|\hat{v}_j\|_\infty \leqslant \max_{|z|=\{t,\frac{1}{t}\}} \|v(z)\|_\infty e^{-\alpha|j|}, \qquad \alpha = \log(t).$$

. Since $\|u(z)\|_\infty \leqslant \|u(z)\|_2 \leqslant \|\psi(z)\|_2$ and, $\|\hat{v}_j\|_\infty \leqslant \|\hat{v}_j\|_2 \leqslant \|\psi(z)\|_2$ we can write

$$|\hat{u}_j| \leqslant \|\psi(z)\|_2 e^{-\alpha|j|}, \quad |\hat{v}_j| \leqslant \|\psi(z)\|_2 e^{-\alpha|j|}.$$

Moreover, recalling that $\psi^{(\ell)}(z)$ can be written as

$$\psi^{(\ell)}(z) = \frac{1}{2^\ell}\sum_{j=1}^{2^\ell} \psi(\zeta_{2^\ell}^j z)$$

the same must hold for $C(z)$ by linearity, and so substituting the Fourier expansion in the above equation yields

$$C^{(\ell)}(z) = \frac{1}{2^\ell}\sum_{j=1}^{2^\ell} C(\zeta_{2^\ell}^j z) = \frac{1}{2^\ell}\sum_{j=1}^{2^\ell}\sum_{l\in\mathbb{Z}} \hat{u}_l z^l \zeta_{2^\ell}^{lj} \sum_{s\in\mathbb{Z}} \hat{v}_s^* z^{sj} \zeta_{2^\ell}^{s}.$$

Swapping the indices in the sum allows to simplify many terms, since the sum over the roots of the unity gives 0:

$$C^{(\ell)}(z) = \frac{1}{2^\ell}\sum_{l\in\mathbb{Z}}\sum_{s\in\mathbb{Z}}\sum_{j=1}^{2^\ell} \hat{u}_l \hat{v}_s^* z^{l+s} \zeta_{2^\ell}^{(l+s)j} = \sum_{l\in\mathbb{Z}} \hat{u}_l \sum_{s\in\mathbb{Z}} \hat{v}_{s2^\ell-l}^* z^{s2^\ell}.$$

If we take $|z| = 1$ we can write $C^{(\ell)}(z)$ as $C^{(\ell)}(z) = \sum_{l\in\mathbb{Z}} X_l$ with

$$X_l = \hat{u}_l \sum_{s\in\mathbb{Z}} \hat{v}_{s2^\ell-l}^* z^{s2^\ell}.$$

being a rank 1 matrix. The decay on $\hat{v}_s$ and $\hat{u}_l$ guarantees that

$$\|X_l\|_2 \leqslant \|\hat{u}_l\|_2 \cdot \sum_{s\in\mathbb{Z}} \|\hat{v}_{s2^\ell-l}^*\|_2 \leqslant C_t \sup_{z\in\mathcal{A}_t} \|\psi(z)\|_2^2 \frac{e^{-\alpha|l|}}{1-e^{-\alpha 2^\ell}}.$$

where C_t is an appropriate constant. Applying Lemma 3.6.11 gives the thesis. ☐

The above theorem guarantees the requested decay on the off-diagonal singular values of $\psi(z)$. We need now to recover the same bound on the off-diagonal singular values of $\varphi(z)$. Recalling that $z\varphi(z)^{-1} = \psi(z)$ we can use Lemma 3.6.10 to obtain the thesis. We have the following

Theorem 3.6.12. *Let $\varphi^{(\ell)}(z)$ be the matrix function associated to the ℓ-th step of the cyclic reduction for a QBD with tridiagonal coefficients and invertible on an annulus $\mathcal{A}_t$. Then, the singular values of each offdiagonal block $C(z)$ are bounded by*

$$\sigma_j(C(z)) \leqslant C_t n \max_{z \in \mathcal{A}_t} \|\psi(z)\|_2^2 e^{-\alpha|j|}$$

where α and C_t are the usual constants of Theorem 3.6.7 and n is the size of $\varphi(z)$.

Proof. Since $z^{-1}\varphi^{(\ell)}(z) = (\psi^{(\ell)})^{-1}(z)$ we can combine Lemma 3.6.10 and Theorem 3.6.7 and we have

$$\sigma_j(C(z)) \leqslant \|A_{1,1}(z)\|_2 \|A_{2,2}(z)\|_2 \sigma_j(\tilde{B}(z))$$

where $\tilde{C}(z)$ is the block corresponding to $C(z)$ in $\psi^{(\ell)}(z)$. Given that $\|\varphi(z)\|_\infty \leqslant 2$ we have

$$\sigma_j(C(z)) \leqslant 4n\sigma_j(\tilde{C}(z)) \leqslant C_t n \|\psi(z)\|_2^2$$

where C_t is an appropriate constant and we have used that $\|\cdot\|_2 \leqslant \sqrt{n}\|\cdot\|_\infty$. □

3.6.2 *Applying the shift*

In many practical situations in order to speed up the convergence of the cyclic reduction it is useful to shift away the eigenvalue 1. In [18] it is shown that we can consider a rank 1 modification of $\varphi(z)$ with the eigenvalue removed. We have the following.

Lemma 3.6.13. *There exist a rank 1 stochastic matrix Q such that the matrix polynomial $\tilde{\varphi}(z) = z^2\tilde{A_1} + z\tilde{A_0} + \tilde{A}_{-1}$ with*

$$\tilde{A}_{-1} = A_{-1} - A_{-1}Q, \quad \tilde{A_0} = A_0 + A_1 Q, \quad \tilde{A_1} = A_1,$$

and $\varphi(z)$ share the same eigenvalues with the only exception of an eigenvalue 1 of $\varphi(z)$ that is moved to 0 in $\tilde{\varphi}(z)$. Moreover, Q is of the form eu^t where e is the vector of ones and $u > 0$ is such that $u^t e = 1$.

We can define $\tilde{\psi}(z)$ as $z\tilde{\varphi}(z)^{-1}$ and we have the following important result that relates the cyclic reduction iterations on the modified problem with the ones on the original one.

Theorem 3.6.14. *Let $\tilde{\varphi}^{(\ell)}(z)$ be the matrix polynomial obtained at the ℓ-step of the CR iteration applied to the shifted $\tilde{\varphi}(z)$. Then we have $\tilde{\varphi}^{(\ell)}(z) = \varphi^{(\ell)}(z) + R^{(\ell)}(z)$ and $\tilde{\psi}^{(\ell)}(z) = \psi^{(\ell)}(z) + T^{(\ell)}(z)$ where $R^{(\ell)}(z)$ and $T^{(\ell)}(z)$ has rank at most 1.*

Proof. Note that

$$\begin{aligned}\tilde{\varphi}^{(0)}(z)^{-1} &= (\varphi^{(0)}(z) + (zA_1 - A_{-1})Q)^{-1} \\ &= (\varphi^{(0)}(z) + (zA_1 - A_{-1})eu^t)^{-1} = \varphi^{(0)}(z)^{-1} - x(z)u^t\end{aligned}$$

for some vector $x(z)$ thanks to the Woodbury matrix inversion formula. Using this fact we can write

$$\tilde{\varphi}^{(\ell)}(z^{2^\ell}) = z^{2^\ell} \left(\frac{1}{2^\ell} \sum_{j=0}^{2^\ell-1} \tilde{\psi}^{(0)}(\zeta_j z) \right)^{-1} = z^{2^\ell} \left(\frac{1}{2^\ell} \sum_{j=0}^{2^\ell-1} \zeta_j z \left(\varphi^{(0)}(\zeta_j z)^{-1} - x(\zeta_j z) u^t \right) \right)^{-1}.$$

Observe that $T^{(\ell)}(z^{2^\ell}) := \left(\frac{1}{2^\ell} \sum_{j=0}^{2^\ell-1} \zeta_j z x(\zeta_j z) \right) u^t$ has rank 1 and so the part of the statement concerning $\tilde{\psi}^{(\ell)}$ is proved.

By setting $\hat{x}(z^{2^\ell}) := \frac{1}{2^\ell z^{2^\ell}} \sum_{j=0}^{2^\ell-1} \zeta_j z x(\zeta_j z)$ we have

$$\tilde{\varphi}^{(\ell)}(z^{2^\ell}) = \left(\varphi^{(\ell)}(z^{2^\ell})^{-1} - \hat{x}(z^{2^\ell}) u^t \right)^{-1} = \varphi^{(\ell)}(z^{2^\ell}) + R^{(\ell)}(z^{2^\ell})$$

where $R(z)$ has rank 1 and the last equality is obtained applying the Woodbury matrix inversion formula again. □

The above guarantees that if we have the decay of the off-diagonal singular values in $\varphi(z)$ then the same must hold also for $\tilde{\varphi}(z)$, since at each step they differ only by a rank 1 update.

3.6.3 *Using $\mathcal{H}$-matrices for the cyclic reduction*

Given the structure of numerical quasiseparability of the matrices $A_i^{(\ell)}$ involved in the CR iteration it is natural to ask how to exploit it. We propose the following strategy. We represent the starting matrices using the $\mathcal{H}$-matrix framework and we carry on the iterations using the matrix operations implemented in the library H2Lib [24]. We have developed a partial MATLAB interface to the library in order to make the implementation of the CR and the comparison of performance easier. It can be found at [75].

As explained in Section 3.1 the library implementing the $\mathcal{H}$-matrices representations perform SVDs at each step to ensure that the rank used for the quasiseparable representation is not too high given a certain threshold. That is, it checks what is the lowest rank that can be used to represent each off-diagonal matrix up to a certain ϵ. Theorem 3.6.7 guarantees that, at each step, this value is bounded by the same constant that can be computed imposing that j is large enough to have $\sigma_j(C(z)) \leqslant u \|C(z)\|_2$ where u is the current unit roundoff and $C(z)$ is an off-diagonal submatrix of $\varphi(z)$, according the notation used in the previous sections. It is important to notice that in this context we are not directly relying on the above bound in order to determine the quasiseparability rank. This has the advantage that whenever the bound obtained by Theorem 3.6.7 is not strict we do not waste computational effort, since the SVD determines the correct numerical rank in any case.

We have carried out experiments on several random QBDs with different values of the threshold ranging from 10^{-8} to 10^{-16} and we have observed that

- The performance improvement is very noticeable. The cost of the algorithm goes from cubic to n times a power of $\log n$, since the QS rank increases as $\log n$ and the operations on $\mathcal{H}$-matrices have generally a cost of n times a small power in $\log n$.
- The accuracy of the results is comparable to the threshold chosen, so the method seems to be very robust from the numerical experiments.

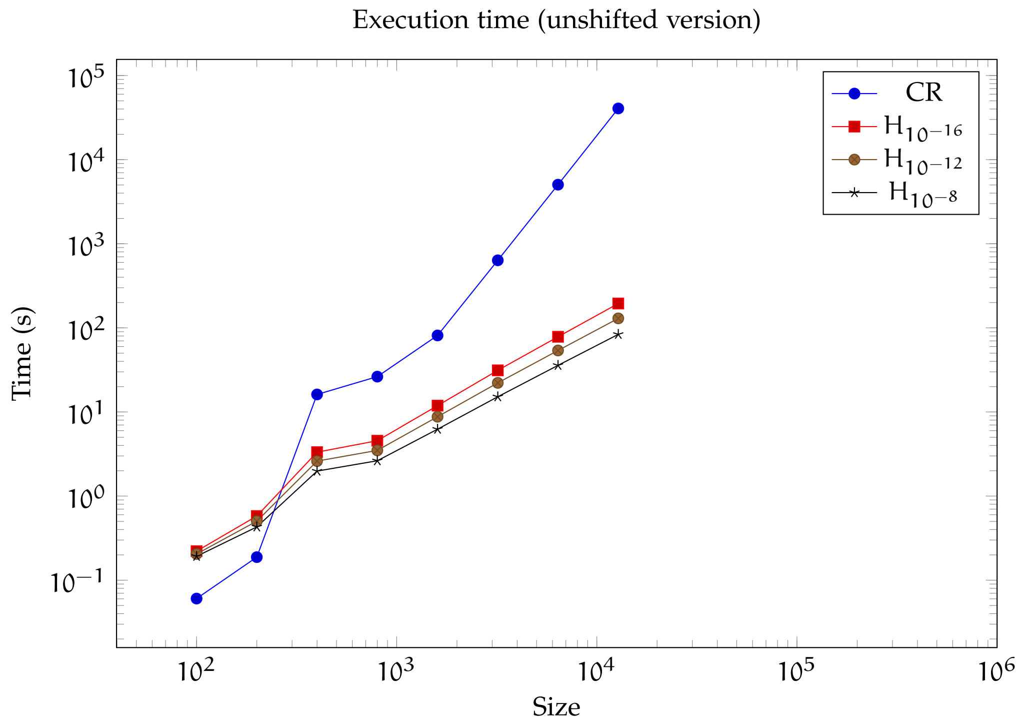
	CR		$H_{10^{-16}}$		$H_{10^{-12}}$		$H_{10^{-8}}$	
Size	Time (s)	Residue	Time (s)	Residue	Time (s)	Residue	Time (s)	Residue
100	$6.04e-02$	$1.91e-16$	$2.21e-01$	$1.79e-15$	$2.04e-01$	$8.26e-14$	$1.92e-01$	$7.40e-10$
200	$1.88e-01$	$2.51e-16$	$5.78e-01$	$1.39e-14$	$5.03e-01$	$1.01e-13$	$4.29e-01$	$2.29e-09$
400	$1.61e+01$	$2.09e-16$	$3.32e+00$	$1.41e-14$	$2.60e+00$	$1.33e-13$	$1.98e+00$	$1.99e-09$
800	$2.63e+01$	$2.74e-16$	$4.55e+00$	$1.94e-14$	$3.49e+00$	$2.71e-13$	$2.63e+00$	$2.69e-09$
1600	$8.12e+01$	$3.82e-12$	$1.18e+01$	$3.82e-12$	$8.78e+00$	$3.82e-12$	$6.24e+00$	$3.39e-09$
3200	$6.35e+02$	$5.46e-08$	$3.12e+01$	$5.46e-08$	$2.21e+01$	$5.46e-08$	$1.51e+01$	$5.43e-08$
6400	$5.03e+03$	$3.89e-08$	$7.83e+01$	$3.89e-08$	$5.38e+01$	$3.89e-08$	$3.58e+01$	$3.87e-08$
12800	$4.06e+04$	$1.99e-08$	$1.94e+02$	$1.99e-08$	$1.29e+02$	$1.99e-08$	$8.37e+01$	$1.97e-08$

Table 3.2: Table reporting the timings for the unshifted runs of the cyclic reduction applied with the standard algorithm (not exploiting the structure) and with the implementation based on $\mathcal{H}$-matrices.

	CR		$H_{10^{-16}}$		$H_{10^{-12}}$		$H_{10^{-8}}$	
Size	Time (s)	Residue	Time (s)	Residue	Time (s)	Residue	Time (s)	Residue
100	$8.99e-02$	$2.21e-16$	$2.52e-01$	$1.17e-15$	$2.27e-01$	$8.58e-14$	$2.16e-01$	$8.60e-10$
200	$4.31e-01$	$2.11e-16$	$6.34e-01$	$9.98e-15$	$5.20e-01$	$1.05e-13$	$4.48e-01$	$1.77e-09$
400	$1.40e+01$	$2.33e-16$	$2.75e+00$	$1.55e-14$	$2.09e+00$	$1.13e-13$	$1.63e+00$	$1.90e-09$
800	$2.89e+01$	$4.28e-16$	$5.60e+00$	$1.51e-14$	$4.06e+00$	$2.79e-13$	$3.15e+00$	$2.35e-09$
1600	$8.25e+01$	$2.63e-14$	$1.48e+01$	$2.66e-14$	$9.63e+00$	$2.93e-13$	$7.03e+00$	$2.81e-09$
3200	$6.46e+02$	$6.92e-08$	$4.11e+01$	$6.92e-08$	$2.44e+01$	$6.92e-08$	$1.72e+01$	$7.18e-08$
6400	$5.11e+03$	$1.25e-10$	$1.10e+02$	$1.25e-10$	$6.00e+01$	$1.25e-10$	$4.11e+01$	$8.06e-09$
12800	$4.13e+04$	$2.03e-08$	$3.06e+02$	$2.03e-08$	$1.48e+02$	$2.03e-08$	$9.89e+01$	$2.02e-08$

Table 3.3: Table reporting the timings for the shifted runs of the cyclic reduction applied with the standard algorithm (not exploiting the structure) and with the implementation based on $\mathcal{H}$-matrices.

The results of the numerical experiments have been reported in Figure 3.11 and in Tables 3.3 and 3.2. In the tables we have reported the timings and the residual norm of the approximation of the solution G of the matrix equation after 15 iterations.



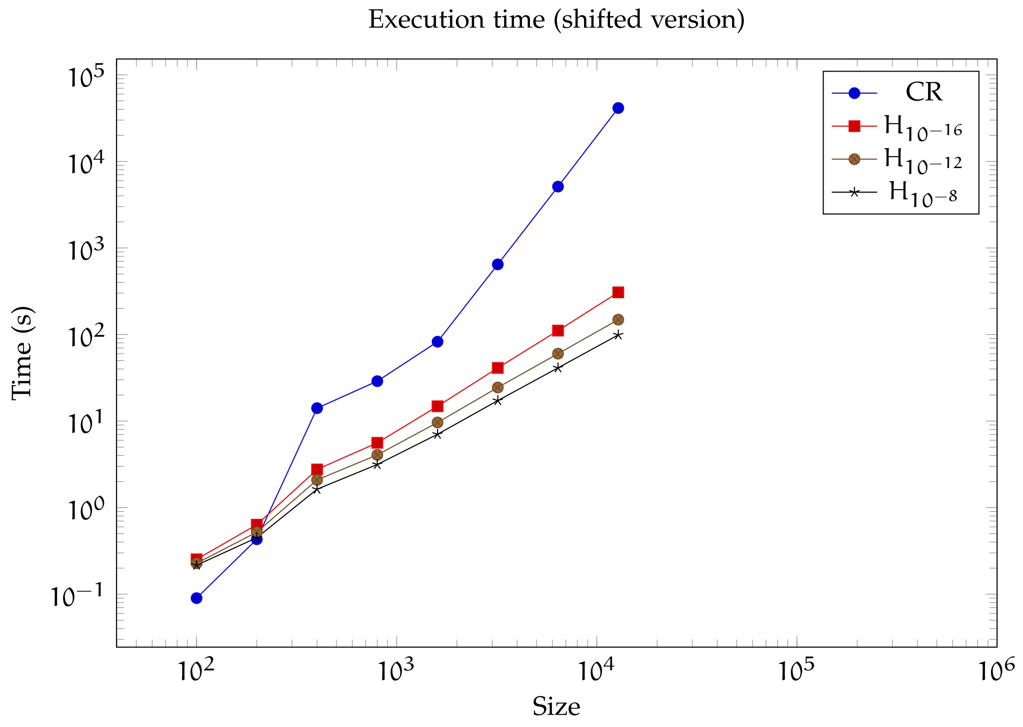


Figure 3.11: Timings of the execution of different implementations of the CR iteration. The iteration is applied to tridiagonal starting matrices of different size. The experiments report the timings for the shifted and unshifted version of the cyclic reduction.

CONCLUSIONS

MAIN CONTRIBUTIONS AND RESULTS

In this chapter we summarize the main results and contributions that have been presented in the thesis. Our main goal was to show that rank structures (in the form of quasiseparability) and matrix polynomials do play well together. We have started our analysis from scalar polynomials, and after briefly reporting the available methods for numerically finding their roots we have provided the tools needed to design the `secsolve` algorithm. This is one of the main contributions of this thesis that has been completed in the first part of the PhD. The algorithm is, as of today, the fastest method available for the approximation of polynomial roots at arbitrary precision in almost any case, as shown by the numerical experiments reported in Table 1.2. Here we have exploited the rank structure of a particular companion matrix of the form $A = D + uv^*$ in order to make the algorithm effective.

In the last part of Chapter 1 we have presented an approximation algorithm for the roots of Mandelbrot polynomials. As far as we know this is the fastest algorithm available at the moment.

The analysis of these tools has been a source of inspiration for the development of a similar framework for matrix polynomials. In Chapter 2 we have turned our attention to the case of matrix polynomials $P(x) = \sum_{i=0}^{n} P_i x^i \in \mathbb{C}^{m\times m}[x]$ and we have developed a novel family of linearizations and ℓ-ifications that mimic the role that the previously mentioned companion matrix had in the scalar case. We have given numerical evidence that the conditioning of the eigenvalue problem associated with these linearizations is low in many cases if good choices for the parameters are made, and we have discussed how the tropical roots can be used to perform these optimal choices. In the last part of Chapter 2 we have also shown a construction that further generalizes this family of linearizations and allows to show how it is possible to build linearizations with an arbitrarily low condition number related to the eigenvalue problem.

To further analyze the role of quasiseparability in Section 3.1 we have briefly reviewed some possible choices to represent quasiseparable matrices. In Section 3.3 and 3.4 we have presented an analysis of the Hessenberg and Hessenberg triangular reduction processes from the point of view of quasiseparability. We have shown that whenever the starting matrix A or the pencil $xA - B$ do have a real diagonal plus low rank structure (a condition more strict than quasiseparability) then the quasiseparable structure is available also in the Hessenberg form. We have shown how it is possible to build an $O(n^2k)$ algorithm where n is the size of the matrices and k is the quasiseparable rank.

In Section 3.5 we have shown an alternative approach for the construction of the Hessenberg form that seems to be more stable and easier to compute. In particular, we have overcome the problem of representing the quasiseparable structure by maintaining a particular banded plus low rank structure. Numerical experiments seem to confirm that this approach is reliable.

ORIGINAL CONTRIBUTIONS

Here we provide a list of the original contributions developed during the PhD that have been published, submitted, or are currently in preparation.

- The approach presented in Chapter 1 for the solution of polynomials and secular equations has been published in [20]. The results have been extended to cover the case of Mandelbrot polynomials and also other classes of functions related to polynomials (such as rational functions). These last extensions are not part of [20] but might be the subject of future work.
- The class of secular linearizations and ℓ-ifications presented in Section 2.2 has been described in the paper [21]. In the paper we show how to construct these ℓ-ifications and also provide numerical evidence of their good properties in many interesting cases.
- The class of secular linearizations associated with a certain matrix polynomial $Q(x)$ is described in a paper that is currently in preparation [81]. In this work we aim, among other things, at giving a solid theoretical base to understand the good conditioning properties found in the linearizations of [21].
- The quasiseparable Hessenberg reduction process of Section 3.3 is described in [22], which has been recently published. In that work the new framework used in this thesis to deal with Givens rotations is introduced. The theoretical results that are presented are of general interest when dealing with quasiseparable structures, and the formalization of the action of rotations aims to find a bridge between easy to follow graphical representations like the arrows ⮡ of [89, 90, 88] and the sometimes involved notation (at least in this context) using classical matrix products.
- A paper describing the Hessenberg reduction process of Section 3.5 is currently in preparation and will be submitted soon.
- The work of Section 3.6 has been recently submitted. The paper contains the results of this thesis and also other extensions to more general quasiseparable matrices.

Here we mention some other papers that have been developed during the course of the PhD. Even though they are not directly related to the subject of this thesis (and in fact they have not been included) they can be seen as a byproduct of this research.

- A paper in collaboration with a group working in the context of random polynomials has been submitted to arXiv in [29]. The work aims at developing a tool, based on `MPSolve`, capable of guaranteeing the number of real roots of random polynomials. The tool has been used to provide new asymptotic data in the study of the distribution of the roots of Cauchy and Gaussian random polynomials.
- A work on the solution of $*$-Sylvester equations by means of periodic Schur decompositions is currently in preparation in collaboration with Fernando De Terán, Bruno Iannazzo and Federico Poloni. We have described a practical algorithm for the solution that is based on the combination of periodic QR and a back-substitution. A FORTRAN implementation of the algorithm has been written and will be released in the near future.

FUTURE RESEARCH

Here we give some ideas of possible extensions of the results that we have presented that could constitute future research focus. Several developments are possible.

- The polynomial approximation package `MPSolve` allows for the analysis and implementation of polynomial solvers in different basis. Part of this work has already been carried out for Chebyshev polynomials and is part of the current `MPSolve` release. Extensions are possible to other kind of orthogonal basis and also to find the eigenvalues of matrix polynomials implicitly represented as the roots of $\det P(x)$. This is one of the reasons that inspired curiosity about the secular linearizations of Section 2.2 in the first place.
- A unified theory for the analysis of the linearizations and ℓ-ifications of Section 2.2 and 2.5 would be nice to have. In particular, we would want to provide explicit constructions for the polynomial $Q(x)$ needed in the context of the linearization of Section 2.5 in a similar way to we have done by setting $Q(x) = \prod_{i=1}^{n}(x - b_i)I_m$, but involving also the eigenvectors.
- The structured Hessenberg reduction of Section 3.5 deserves to be further studied. In particular, we might want to extend this approach, that seems to work very reliably, to the unitary plus low rank case which is very interesting because it would allow to work with classical Frobenius forms and not only Chebyshev and secular linearizations. This analysis will be part of a future work [60].
- The approach of Section 3.6 is interesting since considers approximate quasiseparable structures. More advanced techniques for bounding the off-diagonal decay of singular values are being studied and will be available in [16], which is to be submitted soon. In particular, a bound involving Krylov spaces techniques that allows to measure how much the Fourier coefficient become parallel will be provided. Moreover, it could be interesting the extend these techniques to the infinite problem and see if the theoretical framework admits an extension to this more general setting. This will be part of future works. A further improvement could be obtained by switching to $\mathcal{H}^2$-matrices, instead of $\mathcal{H}$-matrices. However, the software available for the use of $\mathcal{H}^2$-matrices is still in a beta stage, and so we could not make the switch yet. Nevertheless, we plan to investigate the possible performance improvement given by this change.

BIBLIOGRAPHY

[1] Oliver Aberth. Iteration methods for finding all zeros of a polynomial simultaneously. *Mathematics of computation*, 27(122):339–344, 1973.

[2] Amir Amiraslani, Robert M. Corless, and Peter Lancaster. Linearization of matrix polynomials expressed in polynomial bases. *IMA Journal of Numerical Analysis*, 29(1):141–157, 2009.

[3] Gregory S. Ammar and William B. Gragg. $O(n^2)$ reduction algorithms for the construction of a band matrix from spectral data. *SIAM Journal on Matrix Analysis and Applications*, 12(3):426–431, 1991.

[4] Edward Anderson, Zhaojun Bai, Christian Bischof, Susan Blackford, James Demmel, Jack Dongarra, Jeremy Du Croz, Anne Greenbaum, Sven Hammerling, Alan McKenney, et al. *LAPACK Users' Guide*, volume 9. SIAM, 1999.

[5] Jared L. Aurentz, Thomas Mach, Raf Vandebril, and David S. Watkins. Fast and backward stable computation of roots of polynomials. *SIAM J. Matrix Anal. Appl.*, 36(3):942–973, 2015.

[6] Mario Bebendorf. Approximation of boundary element matrices. *Numerische Mathematik*, 86(4):565–589, 2000.

[7] Mario Bebendorf and Sergej Rjasanow. Adaptive low-rank approximation of collocation matrices. *Computing*, 70(1):1–24, 2003.

[8] Timo Betcke, Nicholas J. Higham, Volker Mehrmann, Christian Schröder, and Françoise Tisseur. NLEVP: A collection of nonlinear eigenvalue problems. http://www.mims.manchester.ac.uk/research/numerical-analysis/nlevp.html.

[9] Timo Betcke, Nicholas J. Higham, Volker Mehrmann, Christian Schröder, and Françoise Tisseur. NLEVP: A collection of nonlinear eigenvalue problems. MIMS EPrint 2011.116, Manchester Institute for Mathematical Sciences, The University of Manchester, UK, December 2011.

[10] Timo Betcke, Nicholas J. Higham, Volker Mehrmann, Christian Schröder, and Françoise Tisseur. NLEVP: A collection of nonlinear eigenvalue problems. Users' guide. MIMS EPrint 2011.117, Manchester Institute for Mathematical Sciences, The University of Manchester, UK, December 2011.

[11] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *arXiv preprint arXiv:1411.1607*, 2014.

[12] Dario A. Bini. Numerical computation of polynomial zeros by means of Aberth's method. *Numerical Algorithms*, 13(2):179–200, 1996.

[13] Dario A. Bini, Francesco Daddi, and Luca Gemignani. On the shifted QR iteration applied to companion matrices. *Electronic Transactions on Numerical Analysis*, 18:137–152, 2004.

[14] Dario A. Bini, Yuli Eidelman, Luca Gemignani, and Israel Gohberg. Fast QR eigenvalue algorithms for Hessenberg matrices which are rank-one perturbations of unitary matrices. *SIAM Journal on Matrix Analysis and Applications*, 29(2):566–585, 2007.

[15] Dario A. Bini and Giuseppe Fiorentino. Design, analysis, and implementation of a multi-precision polynomial rootfinder. *Numerical Algorithms*, 23(2-3):127–173, 2000.

[16] Dario A. Bini, Stefano Massei, and Leonardo Robol. Efficient cyclic reduction for QBDs with rank structured sub-matrices. submitted.

[17] Dario A. Bini and Beatrice Meini. The cyclic reduction algorithm: from Poisson equation to stochastic processes and beyond. *Numerical Algorithms*, 51(1):23–60, 2009.

[18] Dario A. Bini, Beatrice Meini, and Guy Latouche. *Numerical methods for structured Markov chains*. Oxford University Press, 2005.

[19] Dario A. Bini, Vanni Noferini, and Meisam Sharify. Locating the eigenvalues of matrix polynomials. *SIAM Journal on Matrix Analysis and Applications*, 34(4):1708–1727, 2013.

[20] Dario A. Bini and Leonardo Robol. Solving secular and polynomial equations: A multi-precision algorithm. *Journal of Computational and Applied Mathematics*, 272:276–292, 2014.

[21] Dario A. Bini and Leonardo Robol. On a class of matrix pencils and ℓ-ifications equivalent to a given matrix polynomial. *Linear Algebra and its Applications*, 2015.

[22] Dario A. Bini and Leonardo Robol. Quasiseparable Hessenberg reduction of real diagonal plus low rank matrices and applications. *Linear Algebra and its Applications*, 2015.

[23] Paola Boito, Yuli Eidelman, Luca Gemignani, and Israel Gohberg. Implicit QR with compression. *Indagationes Mathematicae*, 23(4):733–761, 2012.

[24] Steffen Börm. H2Lib public repository. GitHub repository: `https://github.com/H2Lib/H2Lib`, 2015.

[25] Steffen Börm and Lars Grasedyck. HLib–a library for H-and H2-matrices, 1999.

[26] Steffen Börm, Lars Grasedyck, and Wolfgang Hackbusch. Hierarchical matrices. *Lecture notes*, 21:2003, 2003.

[27] Carsten Carstensen. Inclusion of the roots of a polynomial based on Gerschgorin's theorem. *Numerische Mathematik*, 59(1):349–360, 1991.

[28] Shiv Chandrasekaran, Ming Gu, Jianlin Xia, and Jiang Zhu. A fast QR algorithm for companion matrices. In *Recent advances in matrix and operator theory*, pages 111–143. Springer, 2008.

[29] Joseph Cleveland, Jeffrey Dzugan, Jonathan D. Hauenstein, Ian Haywood, Dhagash Mehta, Anthony Morse, Leonardo Robol, and Taylor Schlenk. Certified counting of roots of random univariate polynomials. *arXiv preprint arXiv:1412.1717*, 2014.

[30] Robert M. Corless. On a generalized companion matrix pencil for matrix polynomials expressed in the Lagrange basis. In *Symbolic-Numeric Computation*, pages 1–15. Springer, 2007.

[31] Robert M. Corless and Piers W. Lawrence. The largest roots of the Mandelbrot polynomials. In *Computational and Analytical Mathematics*, pages 305–324. Springer, 2013.

[32] J. J. M. Cuppen. A divide and conquer method for the symmetric tridiagonal eigenproblem. *Numerische Mathematik*, 36(2):177–195, 1980.

[33] Fernando De Terán, Froilán M. Dopico, and D. Steven Mackey. Fiedler companion linearizations and the recovery of minimal indices. *SIAM Journal on Matrix Analysis and Applications*, 31(4):2181–2204, 2010.

[34] Fernando De Terán, Froilán M Dopico, and D Steven Mackey. Fiedler companion linearizations for rectangular matrix polynomials. *Linear Algebra and its Applications*, 437(3):957–991, 2012.

[35] Fernando De Terán, Froilán M. Dopico, and D. Steven Mackey. Spectral equivalence of matrix polynomials and the index sum theorem. *Linear Algebra and its Applications*, 459:264–333, 2014.

[36] Steven Delvaux and Marc Van Barel. Rank structures preserved by the QR-algorithm: the singular case. *Journal of Computational and Applied Mathematics*, 189(1):157–178, 2006.

[37] Steven Delvaux and Marc Van Barel. Rank structures preserved by the QR-algorithm: the singular case. *Journal of Computational and Applied Mathematics*, 189(1):157–178, 2006.

[38] Steven Delvaux and Marc Van Barel. Structures preserved by Schur complementation. *SIAM journal on matrix analysis and applications*, 28(1):229–252, 2006.

[39] Steven Delvaux and Marc Van Barel. Structures preserved by the QR-algorithm. *Journal of Computational and Applied Mathematics*, 187(1):29–40, 2006.

[40] Steven Delvaux and Marc Van Barel. A Givens-weight representation for rank structured matrices. *SIAM Journal on Matrix Analysis and Applications*, 29(4):1147–1170, 2007.

[41] Steven Delvaux and Marc Van Barel. A Hessenberg reduction algorithm for rank structured matrices. *SIAM Journal on Matrix Analysis and Applications*, 29(3):895–926, 2007.

[42] Steven Delvaux and Marc Van Barel. Eigenvalue computation for unitary rank structured matrices. *Journal of Computational and Applied Mathematics*, 213(1):268–287, 2008.

[43] Steven Delvaux and Marc Van Barel. A QR-based solver for rank structured matrices. *SIAM Journal on Matrix Analysis and Applications*, 30(2):464–490, 2008.

[44] Steven Delvaux and Marc Van Barel. Unitary rank structured matrices. *Journal of Computational and Applied Mathematics*, 215(1):49–78, 2008.

[45] Émile Durand. *Solutions Numériques Des Équations Algébriques*. Masson, 1961.

[46] Louis W. Ehrlich. A modified Newton method for polynomials. *Communications of the ACM*, 10(2):107–108, 1967.

[47] Yuli Eidelman and Israel Gohberg. On generators of quasiseparable finite block matrices. *Calcolo*, 42(3-4):187–214, 2005.

[48] Yuli Eidelman, Israel Gohberg, and Luca Gemignani. On the fast reduction of a quasiseparable matrix to Hessenberg and tridiagonal forms. *Linear Algebra Appl.*, 420(1):86–101, 2007.

[49] Yuli Eidelman, Israel Gohberg, and Iulian Haimovici. *Separable type representations of matrices and fast algorithms. Vol. 1*, volume 234 of *Operator Theory: Advances and Applications*. Birkhäuser/Springer, Basel, 2014. Basics. Completion problems. Multiplication and inversion algorithms.

[50] Yuli Eidelman, Israel Gohberg, and Iulian Haimovici. *Separable type representations of matrices and fast algorithms. Vol. 2*, volume 235 of *Operator Theory: Advances and Applications*. Birkhäuser/Springer Basel AG, Basel, 2014. Eigenvalue method.

[51] Ludwig Elsner. A remark on simultaneous inclusions of the zeros of a polynomial by Gershgorin's theorem. *Numerische Mathematik*, 21(5):425–427, 1973.

[52] Miroslav Fiedler. A note on companion matrices. *Linear Algebra and its Applications*, 372:325–331, 2003.

[53] Steven Fortune. Polynomial root finding using iterated eigenvalue computation. In *Proceedings of the 2001 international symposium on Symbolic and algebraic computation*, pages 121–128. ACM, 2001.

[54] Pierre Fraigniaud. The Durand-Kerner polynomials roots-finding method in case of multiple roots. *BIT Numerical Mathematics*, 31(1):112–123, 1991.

[55] John G. F. Francis. The QR transformation a unitary analogue to the lr transformation—part 1. *The Computer Journal*, 4(3):265–271, 1961.

[56] Katrijn Frederix, Steven Delvaux, and Marc Van Barel. An algorithm for computing the eigenvalues of block companion matrices. *Numerical Algorithms*, 62(2):261–287, 2013.

[57] HR Gail, SL Hantler, and BA Taylor. Matrix-geometric invariant measures for G/M/l type Markov chains. *Communications in statistics. Stochastic models*, 14(3):537–569, 1998.

[58] F. R. Gantmacher. *The Theory of Matrices. Vol. 1*. AMS Chelsea Publishing, Providence, RI, 1998. Translated from the Russian by K. A. Hirsch, Reprint of the 1959 translation.

[59] Stéphane Gaubert and Meisam Sharify. Tropical scaling of polynomial matrices. In *Positive systems*, pages 291–303. Springer, 2009.

[60] Luca Gemignani and Leonardo Robol. Fast Hessenberg reduction of some rank structured matrices. In preparation.

[61] I. Gohberg, P. Lancaster, and L. Rodman. *Matrix polynomials*. Academic Press, Inc. [Harcourt Brace Jovanovich, Publishers], New York-London, 1982. Computer Science and Applied Mathematics.

[62] Gene H. Golub. Some modified matrix eigenvalue problems. *SIAM Review*, 15(2):318–334, 1973.

[63] Ronald L. Graham. An efficient algorith for determining the convex hull of a finite planar set. *Information processing letters*, 1(4):132–133, 1972.

[64] Ming Gu and Stanley C. Eisenstat. A divide-and-conquer algorithm for the symmetric tridiagonal eigenproblem. *SIAM Journal on Matrix Analysis and Applications*, 16(1):172–191, 1995.

[65] William H. Gustafson. A note on matrix inversion. *Linear algebra and its applications*, 57:71–73, 1984.

[66] Wolfgang Hackbusch. A sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part I: Introduction to $\mathcal{H}$-matrices. *Computing*, 62(2):89–108, 1999.

[67] Wolfgang Hackbusch, Boris Khoromskij, and Stefan A Sauter. *On $\mathcal{H}^2$-matrices*. Springer, 2000.

[68] Peter Henrici. *Applied and computational complex analysis. Vol. 1*. Wiley Classics Library. John Wiley & Sons, Inc., New York, 1988. Power series—integration—conformal mapping—location of zeros, Reprint of the 1974 original, A Wiley-Interscience Publication.

[69] Nicholas J. Higham. *Accuracy and stability of numerical algorithms*. SIAM, 2002.

[70] John Hubbard, Dierk Schleicher, and Scott Sutherland. How to find all roots of complex polynomials by Newton's method. *Inventiones mathematicae*, 146(1):1–33, 2001.

[71] Tosio Kato. *Perturbation theory for linear operators*, volume 132. Springer Science & Business Media, 1976.

[72] Immo O. Kerner. Ein gesamtschrittverfahren zur berechnung der nullstellen von polynomen. *Numerische Mathematik*, 8(3):290–294, 1966.

[73] Vera N. Kublanovskaya. On some algorithms for the solution of the complete eigenvalue problem. *USSR Computational Mathematics and Mathematical Physics*, 1(3):637–657, 1962.

[74] D Steven Mackey, Niloufer Mackey, Christian Mehl, and Volker Mehrmann. Vector spaces of linearizations for matrix polynomials. *SIAM Journal on Matrix Analysis and Applications*, 28(4):971–1004, 2006.

[75] Stefano Massei and Leonardo Robol. h2Lib-matlab public repository. GitHub repository: `https://github.com/robol/h2lib-matlab/`, 2015.

[76] Volker Mehrmann and David S. Watkins. Structure-preserving methods for computing eigenpairs of large sparse skew-hamiltonian/hamiltonian pencils. *SIAM Journal on Scientific Computing*, 22(6):1905–1925, 2001.

[77] Aaron Melman. Generalization and variations of Pellet's theorem for matrix polynomials. *Linear Algebra and its Applications*, 439(5):1550–1567, 2013.

[78] Aaron Melman. Implementation of Pellet's theorem. *Numerical Algorithms*, 65(2):293–304, 2014.

[79] Masakiyo Miyazawa. Tail decay rates in double qbd processes and related reflected random walks. *Mathematics of Operations Research*, 34(3):547–575, 2009.

[80] Cleve B. Moler and Gilbert W. Stewart. An algorithm for generalized matrix eigenvalue problems. *SIAM Journal on Numerical Analysis*, 10(2):241–256, 1973.

[81] Vanni Noferini and Leonardo Robol. A class of diagonal plus low rank linearizations with good conditioning properties. In preparation.

[82] Vanni Noferini, Meisam Sharify, and Francoise Tisseur. Tropical roots as approximations to eigenvalues of matrix polynomials. *SIAM Journal on Matrix Analysis and Applications*, 36(1):138–157, 2015.

[83] Leonardo Robol. A rootfinding algorithm for polynomials and secular equations. Master's thesis, University of Pisa, 2012.

[84] Dierk Schleicher and Robin Stoll. Newton's method in practice: finding all roots of polynomials of degree one million efficiently. *arXiv preprint arXiv:1508.02935*, 2015.

[85] Meisam Sharify. *Scaling Algorithms and Tropical Methods in Numerical Matrix Analysis: Application to the Optimal Assignment Problem and to the Accurate Computation of Eigenvalues*. PhD thesis, Ecole Polytechnique X, 2011.

[86] Paolo Tilli. Convergence conditions of some methods for the simultaneous computation of polynomial zeros. *Calcolo*, 35(1):3–15, 1998.

[87] Marc Van Barel, Raf Vandebril, Paul Van Dooren, and Katrijn Frederix. Implicit double shift QR-algorithm for companion matrices. *Numerische Mathematik*, 116(2):177–212, 2010.

[88] Ellen Van Camp. *Diagonal-plus-semiseparable matrices and their use in numerical linear algebra*. PhD thesis, KU Leuven, 2005.

[89] Raf Vandebril, Marc Van Barel, and Nicola Mastronardi. *Matrix Computations and Semiseparable Matrices*, volume 1: Linear systems. Johns Hopkins University Press, Baltimore, MD, 2008.

[90] Raf Vandebril, Marc Van Barel, and Nicola Mastronardi. *Matrix Computations and Semiseparable Matrices*, volume 2: Eigenvalue and singular value methods. Johns Hopkins University Press, Baltimore, MD, 2008.

[91] David S. Watkins. *Fundamentals of matrix computations*, volume 64. John Wiley & Sons, 2004.

[92] Carl Weierstrass. *Neuer Beweis des Satzes, dass jede ganze rationale Function einer Veränderlichen dargestellt werden kann als ein Product aus linearen Functionen derselben Veränderlichen*. Sitzungsberichte der Königlich Preußischen Akademie der Wissenschaften zu Berlin, 1891.

[93] Max A. Woodbury. Inverting modified matrices. *Memorandum report*, 42:106, 1950.