uc3m | Universidad Carlos III de Madrid

TESIS DOCTORAL

Krylov Methods for Large-Scale Modern Problems in Numerical Linear Algebra

Autor:

Javier Andrés González Pizarro

Director:

Froilán César Martínez Dopico

Tutor:

Froilán César Martínez Dopico

INGENIERÍA MATEMÁTICA

Leganés, julio del 2018

Este trabajo ha sido desarrollado en el Departamento de Matemáticas de la Universidad Carlos III de Madrid (UC3M) bajo la dirección del profesor Froilán Martínez Dopico. Inicialmente, se contó con una beca de la UC3M de ayuda al estudio de máster, posteriormente con un contrato predoctoral de la UC3M y finalmente con una beca de la Comisión Nacional de Investigación Científica y Tecnológica (CONICYT) de Chile a través del programa "Beca de Doctorado en el extranjero", número de beca: BCH 72160331. Adicionalmente se recibió ayuda parcial de los proyectos de investigación: "Structured Numerical Linear Algebra: Matrix Polynomials, Special Matrices, and Conditioning" (Ministerio de Economía y Competitividad de España, Número de proyecto: MTM2012-32542) y "Structured Numerical Linear Algebra for Constant, Polynomial and Rational Matrices" (Ministerio de Economía y Competitividad de España, Número de proyecto: MTM2015-65798-P), donde el investigador principal de ambos proyectos fue Froilán Martínez Dopico.

A mi madre

Agradecimientos

En primer lugar, quiero agradecer a mi madre, Olga Pizarro Castillo, por el apoyo incondicional que me ha brindado siempre, porque estar en este momento de mi vida, en esta posición, no hubiese sido posible de no ser por ella. Gracias por todo lo que me has entregado mamá.

También quiero agradecer, de todo corazón, a mi director de tesis, Froilán Martínez Dopico, por la paciencia que ha tenido conmigo, y por todo lo que me ha enseñado. Es increíble la cantidad de aprendizaje que se puede adquirir de él. De él me llevo la frase "A good advisor is your best protection from the horrors". Gracias por permitirme vivir esta increíble experiencia.

Gracias a María José Peláez, por abrirme las puertas para iniciar este viaje. A Diego y Yuliana por ser dos amigos que me han acompañado en todo momento. A los mejores compañeros de oficina que he podido tener: Gustavo, Luis Miguel y Kenet, porque más que compañeros, somos amigos. A Manme y Carla por su ayuda en esta última etapa.

A Mirel y Fredy por abrirme las puertas de su casa y hacerme miembro de su familia. A Daniela, Marta, Fabiola, y sus respectivas familias, que son mis segundas familias.

Al cariño recibido a la distancia de Pilar, Tania, Verónica, Cybill, Paula. A mis nuevas amistades en España, que han sido un pilar fundamental en todo este proceso, Maite, Carlos. A Cris, mi compatriota en Leganés. A Paula (Olivita), María (Nefertiti), Carmen y Carla, por acompañarme durante tanto tiempo. A Natalia, que me ayudó a recuperar la cordura.

A Javier Pérez, mi "hermano mayor", Elizabeth y Nikta, que me ayudaron a crecer como persona y como profesional.

Al departamento de matemáticas de la UC3M, en especial a Julio Moro, Fernando Lledó, Cristina Brändle, Paco Marcellán, Elena Romera y Antonio García. Y por supuesto, al inigualable Alberto Calvo.

A las personas que conforman la cafetería del Edificio Sabatini, donde vas por un café y siempre terminas con una sonrisa.

Y por supuesto, a Luna Macarena. Por todo lo que has hecho por mi madre.

Abstract

Large-scale problems have attracted much attention in the last decades since they arise from different applications in several fields. Moreover, the matrices that are involved in those problems are often sparse, this is, the majority of their entries are zero. Around 40 years ago, the most common problems related to large-scale and sparse matrices consisted in solving linear systems, finding eigenvalues and/or eigenvectors, solving least square problems or computing singular value decompositions. However, in the last years, large-scale and sparse problems of different natures have appeared, motivating and challenging numerical linear algebra to develop effective and efficient algorithms to solve them.

Common difficulties that appear during the development of algorithms for solving modern large-scale problems are related to computational costs, storage issues and CPU time, given the large size of the matrices, which indicate that direct methods can not be used. This suggests that projection methods based on Krylov subspaces are a good option to develop procedures for solving large-scale and sparse modern problems.

In this PhD Thesis we develop novel and original algorithms for solving two large-scale modern problems in numerical linear algebra: first, we introduce the R-CORK method for solving rational eigenvalue problems and, second, we present projection methods to compute the solution of T-Sylvester matrix equations, both based on Krylov subspaces.

The R-CORK method is an extension of the compact rational Krylov method (CORK) [104] introduced to solve a family of nonlinear eigenvalue problems that can be expressed and linearized in certain particular ways and which include arbitrary polynomial eigenvalue problems, but not arbitrary rational eigenvalue problems. The R-CORK method exploits the structure of the linearized problem by representing the Krylov vectors in a compact form in order to reduce the cost of storage, resulting in a method with two levels of orthogonalization. The first level of orthogonalization works with vectors of the same size as the original problem, and the second level works with vectors of size much smaller than the original problem. Since vectors of the size of the linearization are never stored or orthogonalized, R-CORK is more efficient from the point of view of memory and orthogonalization costs than the classical rational Krylov method applied to the linearization. Moreover, since the R-CORK method is based on a classical rational Krylov method, the implementation of implicit restarting is possible and we present an efficient way to do it, that preserves the compact representation of the Krylov vectors.

We also introduce in this dissertation projection methods for solving the T-Sylvester equation, which has recently attracted considerable attention as a consequence of its close relation to palindromic eigenvalue problems and other applications. The theory concerning T-Sylvester equations is rather well understood, and,

before the work in this thesis, there were stable and efficient numerical algorithms to solve these matrix equations for small- to medium- sized matrices. However, developing numerical algorithms for solving large-scale T-Sylvester equations was a completely open problem. In this thesis, we introduce several projection methods based on block Krylov subspaces and extended block Krylov subspaces for solving the T-Sylvester equation when the right-hand side is a low-rank matrix. We also offer an intuition on the expected convergence of the algorithm based on block Krylov subspaces and a clear guidance on which algorithm is the most convenient to use in each situation.

All the algorithms presented in this thesis have been extensively tested, and the reported numerical results show that they perform satisfactorily in practice.

Contents

Chapter 1

Introduction and summary of main results

In recent years, problems with large-scale matrices have attracted the interest of many researchers. Several of these problems arise from applications in different fields, which makes imperative to develop methods to work with large-scale matrices. When a matrix is really large, some problems like the computation of their eigenvalues or the solution of a linear system associated to the matrix can be a very challenging problem and, for example, the computation of its complete spectrum is out of the question. Very large matrices that arise in applications are almost always sparse. This means, the majority of their entries are zero. The most common issues with large matrices are related to storage. For example, consider a square tridiagonal matrix of size 10^6. In the conventional way, if all the entries are stored, it is necessary to store 10^{12} numbers. If, in order to store these numbers, we consider double precision real numbers, we need 8×10^{12} bytes, this is, 8000 gigabytes. However, if we consider a special data structure that only stores the nonzero entries of the matrix, only $3 \times 10^6 - 2$ numbers need to be stored. Then, we have to store only about 24 megabytes. However, these are not all the numbers that need to be stored, for each entry that we store, we have to store its row and column position, in order to know where the number belongs in the matrix. Multiplying by two, we can see that we will need around 48×10^6 bytes or 48 megabytes to store the matrix with this special data structure, which results in an improved and manageable file.

Direct methods can not be used for large-scale and sparse matrices for two main reasons, first, these methods usually need to store the matrices that perform the transformations at each step, which results impractical since we are storing more matrices of large size, and, second, since they are usually based on similarity transformations and, as each similarity transformation causes fill-in, the introduction of nonzeros in positions that previously contained zeros makes that, after few similarity transformations, the matrix becomes completely full, and hence unstorable.

Thus, similarity transformations can not be performed for large-scale and sparse

matrices and it is natural to think in other kind of methods. Since the multiplication of a large-scale and sparse matrix by a vector can be performed very efficiently, because the amount of work is proportional to the number of nonzeros in the matrix, it is natural to think in Krylov subspaces. Let $A \in \mathbb{C}^{n\times n}$ be a large-scale and sparse matrix, and $v \in \mathbb{C}^n$. If we multiply A by v to get Av, and then we multiply A by this vector to get A^2v we can create a Krylov sequence

$$v, Av, A^2v, \ldots$$

using only matrix-vector products.

If we continue this sequence until compute the vector $A^{m-1}v$, we can define the *Krylov subspace*

$$\mathcal{K}_m(A, v) = \text{span}\{v, Av, A^2v, \ldots, A^{m-1}v\}, \tag{1.1}$$

and then, we can use the subspace $\mathcal{K}_m(A, v)$ to obtain approximations for a particular problem. Most of the times, Krylov subspaces are used combined with projection methods. A projection method consists of approximating the exact solution, by a solution that belongs to some subspace $\mathbb{K}$. Projection methods that choose Krylov subspaces as the subspace $\mathbb{K}$ to project the problem, are called *Krylov subspace methods* or directly *Krylov methods*.

For the last 50 years, Krylov methods have been developed to solve several problems in numerical linear algebra. Among these problems, Krylov methods are used to solve four of the most common problems in this field: linear systems [84, 105], classic eigenvalue and eigenvector problems [86], [113, Chapter 9], singular values and SVD decomposition [68], and least square problems [84, Chapter 8]. Also, in the last two decades, Krylov methods have been developed to solve *modern problems in numerical linear algebra* (the adjective "modern" will be used throughout this work to refer to these recent problems; however, it is not habitual to use it in the common bibliography), such as nonlinear eigenvalue problems [8, 63, 75, 100, 71, 104] and matrix equations [89, 90]. In this work, we will focus on two of these problems and we will develop new, novel, and efficient algorithms to solve them.

1.1 First problem: rational eigenvalue problems

For the first problem, we consider the rational eigenvalue problem (REP)

$$R(\lambda)x = 0, \tag{1.2}$$

where $R(\lambda) \in \mathbb{C}(\lambda)^{n\times n}$ is a nonsingular rational matrix, i.e., the entries of $R(\lambda)$ are scalar rational functions in the variable λ with complex coefficients and $\det(R(\lambda)) \not\equiv 0$ is not identically zero, and $x \in \mathbb{C}^n$ is a nonzero vector. More precisely, we consider that $R(\lambda)$ is given as

$$R(\lambda) = P(\lambda) - \sum_{i=1}^{k} \frac{f_i(\lambda)}{g_i(\lambda)} E_i, \tag{1.3}$$

where $P(\lambda) \in \mathbb{C}[\lambda]^{n\times n}$ is a matrix polynomial of degree d in the variable λ, $f_i(\lambda)$, $g_i(\lambda)$ are coprime scalar polynomials of degrees m_i and n_i, respectively, $m_i < n_i$ and $E_i \in \mathbb{C}^{n\times n}$ are constant matrices for $i = 1, \ldots, k$. We emphasize that it is well known that every rational matrix can be written in the form (1.3) [60, 80] (see also [4, Section 2]) and that such form appears naturally in many applications [99].

The REP has attracted considerable interest in recent years since it arises in different applications in some fields such as vibration of fluid-solid structures [108], optimization of acoustic emissions of high speed trains [72], free vibration of plates with elastically attached masses [94], free vibrations of a structure with a viscoelastic constitutive relation describing the behavior of a material [76, 77], and electronic structure calculations of quantum dots [49, 109]. In addition, REPs are often used to approximate other types of nonlinear eigenvalue problems through rational interpolation [45].

A first idea to solve REPs is a brute-force approach, since one can multiply by $\prod_{i=1}^{k} g_i(\lambda)$ to turn the rational matrix (1.3) into a matrix polynomial of degree $d + n_1 + \cdots + n_k$. The common approach to solve a polynomial eigenvalue problem (PEP) is via linearization (see, for instance, [31, 73, 76]), this is, by transforming the PEP into a generalized eigenvalue problem (GEP) and then applying a well-established algorithm to this GEP, as for instance the QZ algorithm in the case of dense medium sized problems [44] or some Krylov subspace method for large-scale problems. However, this brute-force approach is only useful when $n_1 + n_2 + \cdots + n_k$ is small compared with d. So, if k or some n_i are big, then the degree of the matrix polynomial associated to the problem is also big, and this makes the size of the linearization too large, which is impractical for medium to large-scale problems. This drawback has motivated the idea of linearizing directly the REP [99]. The linearization for $R(\lambda)$ in (1.3) constructed in [99] has a size much smaller than the size of the linearization obtained by the brute-force approach. Nonetheless the increase of the size of the problem is still considerable, so for large-scale rational eigenvalue problems, a direct application of this approach, i.e., without taking into account the structure of the linearization, is also impractical. This idea of taking advantage of the structure of the linearization for solving large-scale REPs is closely connected to the intense research effort developed in the last years by different authors for solving large-scale PEPs via linearizations and that is briefly discussed in the next paragraph.

Several methods have been developed to solve large-scale PEPs numerically by applying Krylov methods to the associated GEPs obtained through linearizations [18, 104, 63]. In this approach, the key issues to be solved for using Krylov methods for large-scale PEPs are the increase of the memory cost and the increase of the orthogonalization cost at each step, as a consequence of the increase of the size of the linearization with respect to the size of the original problem. In order to reduce these costs, different representations of the Krylov vectors of the linearizations have been developed. First, the second order Arnoldi method (SOAR) [8] and

the quadratic Arnoldi method (Q-Arnoldi) [75] were developed to solve quadratic eigenvalue problems (QEP), introducing a new representation of the Krylov vectors. However, both methods are potentially unstable as a consequence of performing implicitly the orthogonalization. To cure this instability, the two-level orthogonal Arnoldi process (TOAR) [100, 71] for QEP proposed a different compact representation for the Krylov vectors of the linearization and, combining this representation with the linearization and the Arnoldi recurrence relation, resulted in a memory saving and numerically stable method. Extending the ideas of a compact representation of the Krylov vectors and of the two levels of orthogonalization from polynomials of degree two (TOAR) to polynomials of any degree, the authors of [63] developed a memory-efficient and stable Arnoldi process for linearizations of matrix polynomials expressed in the Chebyshev basis. In 2015, the compact rational Krylov method (CORK) for nonlinear eigenvalue problems (NLEP) was introduced in [104]. CORK considers particular NLEPs that can be expressed and linearized in certain ways, which are solved by applying a compact rational Krylov method to such linearizations. A key feature of the CORK method is that it works for many kinds of linearizations involving a Kronecker structure, as the Frobenius companion form or linearizations of matrix polynomials in different bases (as Newton or Chebyshev, among others [3]). CORK reduces both the costs of memory and orthogonalization by using a generalization of the compact Arnoldi representation of the Krylov vectors of the linearizations used in TOAR [100, 71], and gets stability through two levels of orthogonalization as in TOAR.

In this dissertation, we develop a rational Krylov method that works on the linearization of REPs introduced in [99] to solve large-scale and sparse $n \times n$ REPs. To this aim, we introduce a compact rational Krylov method for REPs (R-CORK).

In the spirit of TOAR and CORK, we will work with two levels of orthogonalization, and, as in CORK, we adapt the classical rational Krylov method [82, 83, 104] on the linearization to a compact representation of the Krylov vectors and to the two levels of orthogonalization.

One of the advantages of rational Krylov subspace (RKS) methods is that different shifts can be chosen at each iteration, in order to compute eigenvalues close to these shifts. Therefore, a shift-and-invert step needs to be computed. We can perform the shift-and-invert step by solving linear systems of size n. To this purpose, the linearization introduced in [99] is preprocessed in a convenient way and, then, an ULP decomposition is used, this is, a decomposition that involves the product of an upper triangular matrix, a lower triangular matrix and a permutation matrix. This decomposition is similar to the one employed in [104] directly on the linearizations of the NLEPs considered there. Once this step is performed, we start with the two levels of orthogonalization.

The first level involves an orthogonalization process with vectors of size n and in the second level of orthogonalization we work with vectors of size much smaller than n, so this level is cheap compared with the first level. As a result, we develop a

stable method that allows us to reduce the orthogonalization cost and the memory cost by exploiting the structure of the matrix pencil that linearizes the REP and using the rational Krylov recurrence relation.

The new method R-CORK for large-scale and sparse REPs is developed in Chapter 4 of this dissertation.

1.2 Second problem: T-Sylvester equations

As we mentioned before, Krylov methods have also been developed to solve large-scale matrix equations, paying particular attention to the most important among these equations, the so called *Sylvester equation*: $AX + XB = C$, which appears in many applications where C has low rank [89, 90]. In this context, as a second problem, we have studied for the first time in the literature the numerical solution of large-scale real square *T-Sylvester equation*

$$AX + X^T B = C, \tag{1.4}$$

where $A, B, C \in \mathbb{R}^{n\times n}$ are given, C has low rank and $X \in \mathbb{R}^{n\times n}$ is the unknown. The study of theoretical properties for this equation goes back to at least 1962, when Taussky and Wielandt [101] analyzed the linear map $X \mapsto AX + X^T B$ for the special case $B = A^T$. Conditions for the existence of solutions in the general case were established by Wimmer in the early '90s [114]. Recently, there has been renewed interest in studying (1.4) see, e.g., [21, 27, 28, 29, 30, 33, 39]. To some extent, this has been sparked by the close relation of (1.4) to palindromic eigenvalue problems of the form $G + \lambda G^T$. For example, the solution of (1.4) is needed to determine the first-order perturbation expansion for a deflating subspace of $G+\lambda G^T$ [16]. In turn, a Newton method for computing such a deflating subspace would require the repeated solution of possibly large T-Sylvester equations, similar to the methods presented in [19, 25] for standard eigenvalue problems. Equations of the form (1.4) also arise as auxiliary problems in a structure-preserving QR algorithm [64] for solving palindromic eigenvalue problems. Applications that involve (1.4) with $B = \pm A^T$ arise from Hamiltonian systems [15], time-varying singular value decompositions [10], and quadratic inverse eigenvalue problems [115].

Finding solutions of (1.4) becomes rather simple for the special case $B = \pm A^T$, $C = \pm C^T$ [15]. For example, if A is invertible then $X = \frac{1}{2}A^{-1}C$ is trivially a solution. In the general case, however, solutions of (1.4) do not admit such a simple expression. For small- to medium-sized matrices, extensions of the Bartels-Stewart algorithm for solving standard Sylvester equations [9] have been proposed for solving numerically T-Sylvester equations in [28, 21, 107]. A whole class of iterative methods can be derived by viewing (1.4) as an $n^2 \times n^2$ linear system in the entries of X and applying an existing iterative solver for linear systems, see [112] for an example. Still, the need for storing all entries of the approximate solution limits these methods to $n \lesssim 10^4$.

In this work, we develop novel projection methods that iteratively construct low-rank approximations to the solution of possibly large-scale T-Sylvester equations with a low-rank right-hand side matrix C. Based on Krylov subspaces, our methods only require matrix-vector multiplications and the solution of linear systems with A, B, which makes them applicable to equations with large and sparse coefficient matrices. Similar projection methods are routinely used for approximating the solution of large Sylvester and Lyapunov equations [90]. As we will see in Chapter 5 of this dissertation, the extension of these existing projection methods to (1.4) is by no means straightforward.

Throughout this dissertation, we restrict our attention to T-Sylvester equations with real coefficient matrices. However, our results and numerical methods can be easily adapted to complex coefficients, for which the transpose in (1.4) is replaced either by the complex transpose or by the conjugate transpose, see [28] and the references therein.

We emphasize that, to the best of our knowledge, the algorithms presented in Chapters 4 and 5 of this dissertation, dealing respectively with large-scale rational eigenvalue problems and T-Sylvester matrix equations, are completely new contributions that solve for the first time these problems in a reliable way without using previous approximations. Moreover, these results can be seen as new contributions to the broad literature on Krylov methods for modern problems in numerical linear algebra.

1.3 Structure and organization of the thesis

This dissertation is organized as follows. Chapter 2 presents some preliminary concepts and it is divided in four parts. Section 2.1 is devoted to summarize matrix eigenvalue problems and it covers from the classic eigenvalue problem up to the two most important nonlinear eigenvalue problems, i.e., polynomial and rational, which share the key property that both can be linearized. So, in this section we also introduce the concept of linearizations for polynomial and rational matrices, and some of the most important properties and examples of these linearizations. In Section 2.2, basic concepts on Krylov methods are presented. Particularly, we introduce the Arnoldi method [7, 86] and the rational Krylov method [82, 83, 104] for solving standard eigenvalue problems (SEPs) and GEPs, respectively. These procedures have allowed to develop memory-efficient algorithms to solve several kind of large-scale problems among the last twenty years. In Section 2.3 we present a brief survey of numerical methods to solve QEPs based on the Arnoldi method. As we mentioned before, we detail some modern methods presented recently in the literature: SOAR [8], Q-Arnoldi [75] and TOAR [100, 71] to solve quadratic eigenvalue problems. TOAR is a method based on the Arnoldi method applied to a linearization and it represents the Krylov vectors in a different way than SOAR and Q-Arnoldi (which

is also based on linearizations), which allows to develop a stable and efficient algorithm. Since the representations of the Krylov vectors in Q-Arnoldi or TOAR can be generalized to linearizations of matrix polynomials of any degree, we present in Section 2.4 the P-Arnoldi method and the TOAR method for polynomial eigenvalue problems, both generalized for matrix polynomials expressed in the Chebyshev basis in [63]. Finally, we present the CORK method [104] for polynomial eigenvalue problems, which is based on the classical rational Krylov method and a compact decomposition of the Krylov vectors.

In Chapter 3 we discuss about matrix equations and we present some basic concepts related to both the Sylvester equation and the $\star$-Sylvester equation, and the conditions for existence and uniqueness of solution for both equations. Section 3.1 presents the most important aspects of the Sylvester equation, and also presents the well-known Bartels-Stewart algorithm [9] to solve the Sylvester equation for small to medium size matrices. Section 3.2 summarizes the most important results for the $\star$-Sylvester equation and presents the algorithm introduced in [28] to solve the $\star$-Sylvester equation, which is based on the generalized Schur decomposition and it is developed (as the Bartels-Stewart algorithm) for small to medium size matrices. Finally, in Section 3.3, we present some extensions of Krylov subspaces: the block and the extended Krylov subspaces, and a brief summary of Krylov methods for solving the Sylvester equation [53, 90].

Chapter 4 is devoted to present our R-CORK method for large-scale and sparse rational eigenvalue problems. R-CORK is a novel method based on the CORK method for polynomial eigenvalue problems and, as CORK, works with a compact representation for rational Krylov subspaces of a linearization which in our case is associated to the rational matrix [99]. In particular, Section 4.1 presents a compact decomposition of the Krylov vectors of this linearization in order to save memory and orthogonalization costs. In this section, we also introduce a ULP decomposition of a preprocessing of the linearization of the rational matrix, and, by using the ULP decomposition, we can perform efficiently the shift-and-invert step in the rational Krylov method. Many of the results presented in this section are closely related to the CORK method, however, some important differences appear from the presence of the strictly rational part in the rational matrix.

Section 4.2 explains in detail the R-CORK method, and we show in this section the development of the so-called two levels of orthogonalization [100, 71, 63, 104]. The first level works with vectors of the same size as the original problem, and the second level works with vectors of a size smaller than the original problem. Also in this section we discuss about computational costs, particularly orthogonalization and memory costs. The implementation of the two levels of orthogonalization is a considerable improvement since the orthogonalization cost of the second level is negligible compared with the cost of the first level while simultaneously avoiding the loss of orthogonality observed in methods like SOAR [8] or Q-Arnoldi [75]. In Section 4.3 we will also show that in terms of memory storage, the R-CORK method

is much more efficient than the classic rational Krylov method applied directly to the linearization. Section 4.4 presents the implementation of implicit restarting for the R-CORK method, which is possible since the R-CORK method is based on a rational Krylov method [32], but require some effort in order to preserve the compact representation of the Krylov vectors after the restarting. The implicit restarting follows the spirit of the Krylov-Schur procedure [98] (see also [97, Section 5.2]) that is used for the TOAR method in [63]. Finally, in Section 4.5, some numerical tests are presented, in order to show the efficiency of the R-CORK method. These tests conclude that, indeed, the R-CORK method is a memory saving method that converges satisfactorily in practice to the targeted eigenvalues. All the original results in this chapter are included in [34].

The goal of Chapter 5 is to present the results obtained for solving the $\star$-Sylvester equation, with $\star = T$, and where A, B in (1.4) are large, sparse, and real coefficient matrices. Section 5.1 presents a general projection framework for the T-Sylvester equation, and, in our case, we suppose that the right-hand side matrix C in (1.4) has low-rank since otherwise is not possible to store the solution. This framework is based on the approach for large-scale Sylvester equations, however, a Petrov-Galerkin condition is considered on a tensor product of low-dimensional subspaces instead of the classic Galerkin condition that is considered for Sylvester equations [13, 55, 87], [90, Section 4.4.1]. Section 5.2 presents the specific block Krylov subspaces we use for solving the T-Sylvester equation. In this section we also bring the algorithmic details of the block Krylov subspace method (BK) and some important details in the implementation that makes that the BK method works efficiently. Since this is a projection method, and at some point of the algorithm we solve a projected reduced T-Sylvester equation, a brief discussion of the solvability of the reduced equation is also presented in this section. Section 5.3 presents a projection method to solve the T-Sylvester equation based on extended Krylov subspaces. Since extended Krylov methods (EK) are used in modern algorithms for solving large-scale Sylvester equations [89], [90, Section 4.4.1], it is natural to extend this idea for the T-Sylvester equation. This section presents the details of the algorithm and develops some important results to implement it in an efficient way. In both methods (block Krylov and extended Krylov) we compute the Frobenius norm of the classic residual matrix, and we show an efficient way to compute it that works only with matrices of a size much smaller than the size of the original problem, reducing in this way the computational costs of the algorithm. In Section 5.4, we present a relation between a T-Stein equation and a fixed point iteration that offers a motivation for choosing the appropriate Krylov subspaces we use, and also proves some convergence results that we expected. These results are confirmed in Section 5.5, which is the section for numerical tests. Since we establish a relation between a certain standard Sylvester equation and the T-Sylvester equation we study, in Section 5.5, we compare our algorithms for the T-Sylvester equation with the extended block Krylov subspace method applied to this standard Sylvester equation [90, Sec-

tion 4.4.1] in several numerical tests. The results obtained in these numerical tests show the good performance of our algorithms, as well as the poor performance of the method based on transforming the problem into a standard Sylvester equation. The original results introduced in this chapter are included in [35].

Chapter 6 discusses the main conclusions and novel results introduced in this PhD Thesis (see Section 6.1), a list of publications that contains the main contributions summarized in this dissertation (see Section 6.2), a list of contribution to conferences where our results were presented (see Section 6.3) and some open problems motivated by the results obtained in this thesis (see Section 6.4).

1.4 Notation and list of acronyms

1.4.1 Notation

We denote vectors by lowercase characters, u, and matrices by capital characters, A. Block vectors and block matrices are denoted by bold face fonts, $\mathbf{u}$, and $\mathbf{A}$, respectively, and the i-th block of $\mathbf{u}$ is represented by $u^{(i)}$. The conjugate transpose of A is denoted as A^*. The $i \times j$ matrix with the main diagonal entries equal to 1 and the rest of entries equal to zero is represented by $I_{i\times j}$. In the particular case of $i = j$ this matrix is the identity matrix and is denoted by I_i. The vector e_j represents the canonical vector associated to the j-th column of the identity matrix and $0_{i\times j}$ represents the zero matrix of size $i \times j$, which in the particular case $i = j$ is denoted simply by 0_j. The matrix U_j represents a matrix with j columns and u_i represents the i-th column of U_j. We omit subscripts when the dimensions of the matrices are clear from the context. We consider the usual Euclidean vector norm

$$\|v\|_2 := \left(\sum_{i=1}^{n} |v_i|^2\right)^{1/2}$$

where $v \in \mathbb{C}^n$ and v_i denotes the i-th coordinate of v, and the spectral and Frobenius matrix norms [51, Ch. 5]:

$$\|A\|_2 = (\rho(A^*A))^{1/2} = \sigma_{max}(A), \quad \|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} |a_{ij}|^2} = \sqrt{\text{trace}(A^*A)},$$

where $A = [a_{ij}] \in \mathbb{C}^{m\times n}$, and, for any square matrix B, $\rho(B)$ denotes the spectral radius of B, i.e.,

$$\rho(B) = \max\{|\lambda| : \lambda \text{ is an eigenvalue of } B\},$$

and $\sigma_{max}(A)$ denotes the largest singular value of A.

The Kronecker product of two matrices is denoted by $A \otimes B$ and its definition can be found in [52]. The set of $n \times n$ complex rational matrices in the variable λ is denoted by $\mathbb{C}(\lambda)^{n\times n}$ and the set of $n \times n$ complex polynomial matrices in the variable λ (or, equivalently, matrix polynomials) is denoted by $\mathbb{C}[\lambda]^{n\times n}$. Also, given a matrix G, the subspace spanned by the columns of G is denoted by range(G). For any subspace $\mathbb{V} \subseteq \mathbb{R}^n$ and any matrix $A \in \mathbb{R}^{n\times n}$, we set $A\,\mathbb{V} := \{Ax : x \in \mathbb{V}\} \subseteq \mathbb{R}^n$. The field of values of a matrix A is given by $\mathcal{F}(A) := \{x^*Ax : x \in \mathbb{C}^n\,,\, x^*x = 1\}$, where x^* is the conjugate-transpose vector of x.

Standard MATLAB notation for submatrices is used, i.e., given a matrix $M \in \mathbb{R}^{m\times n}$, the expression $M(i : j; k : l)$, where $1 \leq i \leq j \leq m$ and $1 \leq k \leq l \leq n$, denotes the submatrix of M consisting of the intersection of rows i up to and including j of M and of columns k up to and including l of M.

1.4.2 List of abbreviations

The following table describes the significance of various abbreviations and acronyms used throughout the thesis. The page on which each one is defined is also given.

Abbreviation	Meaning	Page
BK	Block Krylov Subspace	71
CGS	Classical Gram-Schmidt	22
CORK	Compact Rational Krylov	46
EK	Extended Krylov Subspace	72
GEP	Generalized Eigenvalue Problem	11
HQR	Householder QR	22
MGS	Modified Gram-Schmidt	22
NLEP	Nonlinear Eigenvalue Problem	46
P-Arnoldi	Polynomial Arnoldi	38
PEP	Polynomial Eigenvalue Problem	13
Q-Arnoldi	Quadratic Arnoldi	29
QEP	Quadratic Eigenvalue Problem	25
R-CORK	Compact Rational Krylov method for REPs	75
REP	Rational Eigenvalue Problem	17
RKS	Rational Krylov Subspace	23
SEP	Standard Eigenvalue Problem	11
SOAR	Second Order Arnoldi	26
TOAR	Two-level Orthogonal Arnoldi	32

Table 1.1: List of abbreviations and their meanings

Chapter 2

Preliminaries on eigenvalue problems and Krylov methods

In this chapter, we present a brief summary on eigenvalue problems and also, the most modern methods based on Krylov subspaces to solve them when they are very large.

Section 2.1 introduces some preliminaries concepts related to matrix eigenvalue problems, starting with the standard eigenvalue problem (SEP), which is a particular case of the generalized eigenvalue problem (GEP), and finalizing with two of the most studied problems in linear algebra during the last decades: the polynomial eigenvalue problem (PEP) and the rational eigenvalue problem (REP), which are particular cases of nonlinear eigenvalue problems (NLEP). In this section we also present the most common way for solving numerically PEPs and REPs, which is via linearization. This process constructs a GEP with the same eigenvalues and multiplicities of the original problem. Several linearizations for PEPs have been developed during the past decade [31, 72, 73], and also, more recently, many linearizations for REPs have been studied [2, 4, 36, 99].

In Section 2.2 some basic concepts on Krylov methods are presented. In particular, the Arnoldi method and the rational Krylov (RKS) method are addressed. Both methods compute an orthonormal basis of a Krylov subspace, and then, solve a projected problem which is usually of much smaller size than the original problem. In this section, we are focused on using Krylov methods for solving matrix eigenvalue problems, computing approximate eigenvalues and eigenvectors (usually called Ritz pairs in the literature) of a given problem.

However, the structure of the Krylov vectors computed by either Arnoldi or RKS can be exploited in order to obtain efficient representations of these vectors, which allows to develop memory saving algorithms. Sections 2.3 and 2.4 are devoted to solve numerically QEPs and PEPs, respectively, by exploiting the structure of the Krylov vectors and the structure of the matrices involved in the original problem, resulting in memory efficient procedures.

2.1 Basics on eigenvalue problems

2.1.1 The generalized eigenvalue problem (GEP)

In this section, we consider the GEP

$$Ax = \lambda Bx, \quad x \neq 0$$

where A, $B \in \mathbb{C}^{n\times n}$, $\lambda \in \mathbb{C}$ and $x \in \mathbb{C}^n$. First, we give some basic concepts for the standard eigenvalue problem (SEP), which is the case when $B = I_n$, and then some concepts related to GEPs are introduced.

Definition 2.1. *Let $A \in \mathbb{C}^{n\times n}$. If a scalar $\lambda \in \mathbb{C}$ and a nonzero vector $x \in \mathbb{C}^n$ satisfy the equation*

$$Ax = \lambda x, \quad x \neq 0,$$

then λ is called an eigenvalue of A and x is called an eigenvector of A associated with λ. The pair (λ, x) is an eigenpair for A. The set of all $\lambda \in \mathbb{C}$ that are eigenvalues of A is called the spectrum of A and it is denoted by $\Lambda(A)$.

An important concept related to eigenvalue problems is the characteristic polynomial of a matrix A.

Definition 2.2. *Let $A \in \mathbb{C}^{n\times n}$. The characteristic polynomial of A is*

$$p_A(\lambda) = \det(A - \lambda I_n) \in \mathbb{C}[\lambda].$$

It is important to remark that the characteristic polynomial $p_A(\lambda)$ of each matrix $A \in \mathbb{C}^{n\times n}$ has degree n, and also, $p_A(\lambda) = 0$ if and only if $\lambda \in \Lambda(A)$.

Given $A, B \in \mathbb{C}^{n\times n}$ and $\lambda \in \mathbb{C}$ we define the matrix $A - \lambda B$, which is called a *matrix pencil*. Definition 2.3 introduces the concept of regularity for matrix pencils.

Definition 2.3. *Let $A - \lambda B$ be a matrix pencil with $A, B \in \mathbb{C}^{n\times n}$. If $\det(A - \lambda B)$ is not identically zero, the pencil $A - \lambda B$ is said to be regular; otherwise, it is said to be singular.*

Throughout this work, we will only focus on regular matrix pencils.

Definition 2.4. *A nonzero vector $x \in \mathbb{C}^n$ is a generalized eigenvector of the pair (A, B) with $A, B \in \mathbb{C}^{n\times n}$ if there exists a scalar $\lambda \in \mathbb{C}$, called a generalized eigenvalue, such that*

$$Ax = \lambda Bx, \quad x \neq 0. \tag{2.1}$$

The set of all $\lambda \in \mathbb{C}$ that satisfy (2.1) is denoted by $\Lambda(A, B)$.

When the context is such that no confusion can arise, the adjective "generalized" is usually dropped. Equation (2.1) can be rewritten as

$$(A - \lambda B)x = 0, \quad x \neq 0, \tag{2.2}$$

a square system of homogeneous linear equations. It is immediate to see that if the system (2.2) has a nontrivial solution, then λ is an eigenvalue of (2.1) and the matrix pencil $A - \lambda B$ is singular in that particular λ. Therefore, λ is an eigenvalue of a regular matrix pencil $A - \lambda B$ if and only if $\det(A - \lambda B) = 0$. As with the standard eigenvalue problem, we can define the characteristic polynomial of the pair (A, B) in terms of the determinant of the associated pencil $A - \lambda B$.

Definition 2.5. *Let $A, B \in \mathbb{C}^{n\times n}$. The characteristic polynomial of the matrix pencil $A - \lambda B$ is defined by*

$$p_{A,B}(\lambda) = \det(A - \lambda B) \in \mathbb{C}[\lambda],$$

and the roots of $p_{A,B}(\lambda)$ are the finite eigenvalues of the pair (A, B).

Remark 2.6. *In general, when B is nonsingular, $p_{A,B}(\lambda)$ is a polynomial of degree n, and hence there are n finite eigenvalues associated with the pencil $A - \lambda B$.*

Remark 2.6 shows that, unlike the SEP, a regular matrix pencil $A-\lambda B$ can have $k < n$ finite eigenvalues, with $k = 0, \ldots, n-1$. Example 2.7 illustrates this situation [65, Chapter 12].

Example 2.7. *[65, Chapter 12] Suppose*

$$A = \begin{bmatrix} 1 & 0 \\ 0 & \alpha \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 0 \\ 0 & \beta \end{bmatrix}$$

where $\alpha, \beta \in \mathbb{C}$. Then, the characteristic polynomial of $A - \lambda B$ is

$$p_{A,B}(\lambda) = (1 - \lambda)(\alpha - \beta\lambda),$$

and, by using $p_{A,B}(\lambda)$, we can compute the eigenvalues of the pair (A, B), and there are different cases to consider:

Case 1: *$\alpha \neq 0$, $\beta \neq 0$. There are two finite eigenvalues, 1 and $\frac{\alpha}{\beta}$.*

Case 2: *$\alpha = 0$, $\beta \neq 0$. There are two finite eigenvalues, 1 and 0.*

Case 3: *$\alpha \neq 0$, $\beta = 0$. There is only 1 finite eigenvalue, 1. In this situation it is said that $A - \lambda B$ has also 1 eigenvalue at ∞.*

Case 4: *$\alpha = 0$, $\beta = 0$. In this case the pencil is said to be singular and the eigenvalues cannot be defined through the characteristic polynomial. In fact, the only eigenvalue of $A - \lambda B$ is $\lambda = 1$, since rank$(A - B) <$ rank$(A - \lambda B)$ for $\lambda \neq 1$. Singular pencils are not considered in this work.*

Case 3 in Example 2.7 has helped us to introduce the concept of eigenvalue at ∞. However, this concept, among others, can be generalized for PEPs, and they will be summarized in Section 2.1.2.

2.1.2 The polynomial eigenvalue problem (PEP)

In this section, we study $n \times n$ matrix polynomials written in the form

$$P(\lambda) = \sum_{i=0}^{d} \lambda^i P_i, \quad P_i \in \mathbb{C}^{n\times n}, \quad P_d \neq 0_n, \tag{2.3}$$

where d is called the *degree* of P. In particular, if $\det(P(\lambda))$ is not identically zero for all $\lambda \in \mathbb{C}$, $P(\lambda)$ is called a regular matrix polynomial, otherwise, is called singular. In this thesis, we only consider regular matrix polynomials.

Definition 2.8. *If $\lambda \in \mathbb{C}$ and a nonzero vector $x \in \mathbb{C}^n$ satisfy*

$$P(\lambda)x = 0, \tag{2.4}$$

where $P(\lambda) \in \mathbb{C}[\lambda]^{n\times n}$ is a matrix polynomial, then x is said to be an eigenvector of P corresponding to the eigenvalue λ.

In this thesis, we only consider regular matrix polynomials, for such polynomials the finite eigenvalues are precisely the roots of the scalar polynomial $\det(P(\lambda))$. Sometimes, it is also useful to consider ∞ as an eigenvalue of P, to understand this notion it is necessary to introduce the concept of reversal of matrix polynomials.

Definition 2.9. *For a matrix polynomial $P(\lambda)$ of degree d as in* (2.3)*, the reversal of $P(\lambda)$ is the matrix polynomial*

$$\mathrm{rev}P(\lambda) := \lambda^d P(1/\lambda) = \sum_{i=0}^{d} \lambda^i P_{d-i}.$$

Observe that the nonzero finite eigenvalues of $\mathrm{rev}P$ are the reciprocals of those of P. Definition 2.10 is based on the relation between 0 and ∞ as reciprocals.

Definition 2.10. *Let $P(\lambda)$ be a regular matrix polynomial of degree $d \geq 1$. Then $P(\lambda)$ is said to have an eigenvalue at ∞ with eigenvector x if $\mathrm{rev}P(\lambda)$ has the eigenvalue 0 with eigenvector x.*

The classical approach to solve the polynomial eigenvalue problem (PEP) (2.4), where $P(\lambda)$ is a regular matrix polynomial, is via linearization, this is, the matrix polynomials are converted into matrix pencils (via unimodular matrices) with the same eigenvalues and multiplicities [41, 73], and then, one works with these pencils.

Definition 2.11. *Let $E(\lambda)$ be an $n \times n$ matrix polynomial. $E(\lambda)$ is called a unimodular matrix polynomial if $\det(E(\lambda))$ is a nonzero constant.*

Definition 2.12. *Let $P(\lambda)$ be an $n \times n$ matrix polynomial of degree $d \geq 1$ and $L(\lambda) = \mathbf{A} - \lambda\mathbf{B}$ be an $nd \times nd$ matrix pencil. The pencil $L(\lambda)$ is called a linearization of $P(\lambda)$ if there exist unimodular matrix polynomials $E_1(\lambda)$, $E_2(\lambda)$ such that*

$$\begin{bmatrix} P(\lambda) & 0 \\ 0 & I_{(d-1)n} \end{bmatrix} = E_1(\lambda)(\mathbf{A} - \lambda\mathbf{B})E_2(\lambda).$$

Some linearizations of matrix polynomials of degree d and size $n \times n$, very useful in practice, are called structured linearizations pencils [104, Definition 2.2] and they have the form as the pencils in Definition 2.13.

Definition 2.13. *[104, Definition 2.2] Let $P(\lambda) \in \mathbb{C}[\lambda]^{n\times n}$ be a regular matrix polynomial, i.e., $\det(P(\lambda))$ does not vanish identically, of degree $d \geq 2$ and size $n \times n$. A $dn \times dn$ matrix pencil $L(\lambda)$ of the form*

$$L(\lambda) = \mathbf{A} - \lambda\mathbf{B}, \tag{2.5}$$

where

$$\mathbf{A} = \left[\begin{array}{cccc} A_0 & A_1 & \cdots & A_{d-1} \\ \hline \multicolumn{4}{c}{M \otimes I_n} \end{array}\right], \quad \mathbf{B} = \left[\begin{array}{cccc} B_0 & B_1 & \cdots & B_{d-1} \\ \hline \multicolumn{4}{c}{N \otimes I_n} \end{array}\right] \tag{2.6}$$

and A_i, $B_i \in \mathbb{C}^{n\times n}$, $i = 0, 1, \ldots, d-1$, and M, $N \in \mathbb{C}^{(d-1)\times d}$, is called a structured linearization pencil of $P(\lambda)$ if the following conditions hold

1) $L(\lambda)$ is a linearization of $P(\lambda)$,

2) $M - \lambda N$ has rank $d-1$ for all $\lambda \in \mathbb{C}$, and

3) $(\mathbf{A} - \lambda\mathbf{B})(f(\lambda) \otimes I_n) = e_1 \otimes P(\lambda)$ for some function $f : \mathbb{C} \to \mathbb{C}[\lambda]^d$, $f(\lambda) \neq 0$ for all $\lambda \in \mathbb{C}$, where $e_1 \in \mathbb{C}^d$ is the first vector of the canonical basis of $\mathbb{C}^d$.

Throughout this work, the adjective "structured" refers to matrix pencils with the Kronecker structure that appears in (2.6) and it is used for the CORK method [104], and, then, for the R-CORK method [34]. However, in the literature, this adjective can be used to refer other structures or properties related to the matrix polynomial and/or the associated linearization.

The matrices A_i and B_i that appear in the first block rows in (2.6) are related to the matrix polynomial $P(\lambda)$ and the matrices M and N correspond to the linear relations between the basis functions used in the representation of the matrix polynomial. Some examples for different basis are presented in Table 2.1 [104, Table 1]. Note that in Table 2.1 the function $f(\lambda) = [f_1(\lambda)^T, \ldots, f_d(\lambda)]^T$ satisfies 3) in Definition 2.13 and that M and N do not appear explicitly since they are any two matrices that satisfy 2) in Definition 2.13 and $(M - \lambda N)f(\lambda) = 0$ for 3) in Definition 2.13.

The identity $(\mathbf{A} - \lambda\mathbf{B})(f(\lambda) \otimes I_n) = e_1 \otimes P(\lambda)$ generalizes the identity used in [73] to define certain vector spaces of linearizations. An important property of structured linearization pencils is that their eigenvectors are closely related to the eigenvectors of the matrix polynomial as we can see in Theorem 2.14.

(a) Matrix polynomials of degree d		
Basis	$P(\lambda)$	Basis functions
Monomial	$\sum_{i=0}^{d} \lambda^i P_i$	λ^i
Orthogonal	$\sum_{i=0}^{d} p_i(\lambda) C_i$	$\lambda p_i(\lambda) = \alpha_i p_{i+1}(\lambda) + \beta_i p_i(\lambda) + \gamma_i p_{i-1}(\lambda)$, with $\alpha_i \neq 0$, $\gamma_i > 0$
Newton	$\sum_{i=0}^{d} n_i(\lambda) D_i$	$n_0(\lambda) := 1$, $n_i(\lambda) := \prod_{k=0}^{i-1} (\lambda - \sigma_k)$ for $i > 0$
Lagrange	$\sum_{i=0}^{d} l_i(\lambda) F_i$	$l_i(\lambda) := l(\lambda) \frac{w_i}{\lambda - \sigma_i}$, with $l(\lambda) = (\lambda - \sigma_0)(\lambda - \sigma_1) \cdots (\lambda - \sigma_d)$
(b) A_i and B_i for the matrix pencil $\mathbf{A} - \lambda \mathbf{B}$ in the form (2.6)		
Basis	A_i	B_i
Monomial	P_i, $i = 0, 1, \ldots, d-1$	$\begin{cases} 0 & i < d-1 \\ -P_d & i = d-1 \end{cases}$
Orthogonal	$\begin{cases} C_i & i < d-2 \\ C_{d-2} - \frac{\gamma_{d-1}}{\alpha_{d-1}} C_d & i = d-2 \\ C_{d-1} - \frac{\beta_{d-1}}{\alpha_{d-1}} C_d & i = d-1 \end{cases}$	$\begin{cases} 0 & i < d-1 \\ -\frac{1}{\alpha_{d-1}} C_d & i = d-1 \end{cases}$
Newton	$\begin{cases} D_i & i < d-1 \\ D_{d-1} - \sigma_{d-1} D_d & i = d-1 \end{cases}$	$\begin{cases} 0 & i < d-1 \\ -D_d & i = d-1 \end{cases}$
Lagrange	$\begin{cases} \sigma_{i+1} F_i & i < d-1 \\ \sigma_d F_{d-1} + \sigma_{d-1} \frac{w_d}{w_{d-1}} F_d & i = d-1 \end{cases}$	$\begin{cases} F_i & i < d-1 \\ F_{d-1} + \frac{w_d}{w_{d-1}} F_d & i = d-1 \end{cases}$
(c) f_i and its linear relations between $\mathbf{A} - \lambda \mathbf{B}$ and $P(\lambda)$		
Basis	$f_i(\lambda)$	Linear relations
Monomial	λ^i	$f_{i+1}(\lambda) = \lambda f_i(\lambda)$
Orthogonal	$p_i(\lambda)$	$\alpha_i f_{i+1}(\lambda) = (\lambda - \beta_i) f_i(\lambda) - \gamma_i f_{i-1}(\lambda)$
Newton	$n_i(\lambda)$	$f_{i+1}(\lambda) = (\lambda - \sigma_i) f_i(\lambda)$
Lagrange	$-l_i(\lambda)/(\lambda - \sigma_{i+1})$	$w_i(\lambda - \sigma_{i+2}) f_{i+1}(\lambda) = w_{i+1}(\lambda - \sigma_i) f_i(\lambda)$

Table 2.1: Transformation of matrix polynomials of degree d into the form of Definition 2.6

Theorem 2.14. *([104, Corollary 2.4]) Let $L(\lambda)$ be a structured linearization pencil of $P(\lambda)$ as in Definition 2.13 and let $(\lambda_\star, \mathbf{x})$ be an eigenpair of $L(\lambda)$. Then, the eigenvector $\mathbf{x}$ has the following structure*

$$\mathbf{x} = f(\lambda_\star) \otimes x,$$

where $x \in \mathbb{C}^n$ is an eigenvector of $P(\lambda)$ corresponding to $\lambda_\star$.

The Kronecker structure of $\mathbf{A}$ and $\mathbf{B}$ in (2.6) can be exploited to factorize the matrix pencil $L(\lambda)$ in (2.5) in a way that allows us to solve efficiently linear systems whose coefficient matrix is $L(\mu)$ for different values of μ. This factorization is introduced in Theorem 2.15 and will be used in the future to perform the shift and

invert step in CORK method, and we will adapt it in order to develop the R-CORK method.

Theorem 2.15. *([104, Theorem 2.3]) Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be defined by* (2.6)*. Then, for every* $\mu \in \mathbb{C}$ *there exists a permutation matrix* $\mathcal{P} \in \mathbb{C}^{d\times d}$ *such that the matrix* $(M_1 - \mu N_1) \in \mathbb{C}^{(d-1)\times(d-1)}$ *is invertible with*

$$M =: [m_0 \quad M_1]\mathcal{P}, \quad N =: [n_0 \quad N_1]\mathcal{P}.$$

Moreover, the matrix $L(\mu)$*, i.e., the pencil* $L(\lambda)$ *in* (2.5) *evaluated at* μ*, can be factorized as follows*

$$L(\mu) = \mathbf{A} - \mu\mathbf{B} = \mathcal{U}(\mu)\mathcal{L}(\mu)(\mathcal{P} \otimes I_n),$$

where

$$\begin{aligned} \mathcal{L}(\mu) &= \begin{bmatrix} P(\mu) & 0 \\ (m_0 - \mu n_0) \otimes I_n & (M_1 - \mu N_1) \otimes I_n \end{bmatrix}, \\ \mathcal{U}(\mu) &= \begin{bmatrix} \alpha^{-1} I_n & (\bar{\mathbf{A}}_1 - \mu\bar{\mathbf{B}}_1)((M_1 - \mu N_1)^{-1} \otimes I_n) \\ 0 & I_{(d-1)n} \end{bmatrix}, \end{aligned}$$

with the scalar $\alpha = e_1^T \mathcal{P} f(\mu) \neq 0$ *and*

$$\begin{aligned} [A_0 \quad A_1 \quad \cdots \quad A_{d-1}] &=: [\bar{A}_0 \quad \bar{\mathbf{A}}_1](\mathcal{P} \otimes I_n), \\ [B_0 \quad B_1 \quad \cdots \quad B_{d-1}] &=: [\bar{B}_0 \quad \bar{\mathbf{B}}_1](\mathcal{P} \otimes I_n). \end{aligned}$$

2.1.3 The rational eigenvalue problem (REP)

In this section, we introduce some basic concepts related with the rational eigenvalue problem

$$R(\lambda)x = 0, \quad x \neq 0, \tag{2.7}$$

where $R(\lambda) \in \mathbb{C}^{n\times n}$ is a regular rational matrix, i.e., $\det(R(\lambda)) \not\equiv 0$, written as

$$R(\lambda) = P(\lambda) - \sum_{i=1}^{k} \frac{f_i(\lambda)}{g_i(\lambda)} E_i, \tag{2.8}$$

where $P(\lambda) \in \mathbb{C}[\lambda]^{n\times n}$ is a matrix polynomial of degree d in the variable λ, $f_i(\lambda)$, $g_i(\lambda)$ are coprime scalar polynomials of degrees m_i and n_i respectively, $m_i < n_i$ and $E_i \in \mathbb{C}^{n\times n}$ are constant matrices for $i = 1, \ldots, k$. With a slight lack of rigor, we can say that if the matrices E_i in (2.8) are linearly independent, then the roots of the denominators $g_i(\lambda)$ are the poles of $R(\lambda)$ and that $R(\lambda)$ is not defined in these poles. We emphasize that it is well known that every rational matrix can be written in the form (2.8) [60, 80] (see also [4, Section 2]) and that such form appears naturally in many applications [99].

Definition 2.16. *Let $R(\lambda)$ be a regular rational matrix with the structure of* (2.8)*. A scalar $\lambda \in \mathbb{C}$ that is not a pole of $R(\lambda)$ such that*

$$\det(R(\lambda)) = 0,$$

is referred to as an eigenvalue of R and a corresponding nonzero vector x satisfying (2.7) *is called an eigenvector of R. The pair (λ, x) is referred to as an eigenpair of R.*

Let us denote the matrix polynomial $P(\lambda)$ of degree d in the variable λ as in (2.3)

$$P(\lambda) = \lambda^d P_d + \lambda^{d-1} P_{d-1} + \cdots + \lambda P_1 + P_0,$$

where $P_i \in \mathbb{C}^{n\times n}$, $i = 0, \ldots, d$ are constant matrices. From now on, we assume the generic condition that the leading coefficient matrix P_d is nonsingular in (2.3). As we explained before, we assume that $f_i(\lambda)$ and $g_i(\lambda)$ in (2.8) are coprime, this is, they do not have common factors, and that the rational functions $\dfrac{f_i(\lambda)}{g_i(\lambda)}$ are strictly proper, this is the degree of $f_i(\lambda)$ is smaller than the degree of $g_i(\lambda)$, for $i = 1, \ldots, k$. Under these assumptions, in [99], Su and Bai proposed a linearization to solve the rational eigenvalue problem. With this aim, they first showed that one can find matrices E, F of size $n \times s$, and matrices C, D of size $s \times s$ with $s = r_1 n_1 + r_2 n_2 + \cdots + r_k n_k$ with r_i=rank(E_i) in (2.8), such that

$$R(\lambda) = P(\lambda) - E(C - \lambda D)^{-1} F^T. \tag{2.9}$$

It is important to remark that the representation (2.9) has been previously developed in the literature, also, the manner presented in [99] to express $R(\lambda)$ in the form (2.9) is not the only one, and the interested reader can find more information in [4, 5, 60, 80].

Once the representation (2.9) for the REP is available, the REP $R(\lambda)x = 0$ can be linearized according to [99] as follows

$$(\mathcal{A} - \lambda\mathcal{B})\mathbf{z} = 0, \tag{2.10}$$

where

$$\mathcal{A} = \left[\begin{array}{cccc|c} P_0 & P_1 & \cdots & P_{d-1} & E \\ 0 & \cdots & 0 & -I_n & \\ \vdots & \iddots & \iddots & & \\ 0 & -I_n & & & \\ \hline F^T & & & & C \end{array}\right], \quad \mathcal{B} = -\left[\begin{array}{cccc|c} & & & P_d & \\ & & I_n & & \\ & \iddots & & & \\ I_n & & & & \\ \hline & & & & -D \end{array}\right] \tag{2.11}$$

and

$$\mathbf{z} = \left[\begin{array}{c} x \\ \vdots \\ \lambda^{d-2}x \\ \lambda^{d-1}x \\ \hline y \end{array}\right]. \tag{2.12}$$

Denoting by $\mathbf{A}$ and $\mathbf{B}$ the upper left $nd \times nd$ submatrices of $\mathcal{A}$ and $\mathcal{B}$, we can write $\mathcal{A} - \lambda\mathcal{B}$ as follows

$$\mathcal{A} - \lambda\mathcal{B} = \left[\begin{array}{c|c} \mathbf{A} - \lambda\mathbf{B} & e_1 \otimes E \\ \hline e_1^T \otimes F^T & C - \lambda D \end{array}\right], \tag{2.13}$$

where e_1 is the first column of I_d.

It is important to remark that in many applications, the first step in the process above, i.e., transforming (2.8) into (2.9), is not necessary, since the structure (2.9) appears in a natural way. The size of the matrices $\mathcal{A}$ and $\mathcal{B}$ is $nd+s$ and very often, in practice, $s \ll n$ [99].

The precise definition of linearizations for rational matrices can be found in [4] (see also [2]) and requires the mild assumption that $-E(C-\lambda D)^{-1}F^T$ is a minimal state-space realization of the strictly proper part of $R(\lambda)$. However, we do not need in this thesis all the developments contained in [4]. Theorem 2.17 summarizes the main results that we need for our work.

Theorem 2.17. *([99, Theorem 3.1]) Let $\lambda \in \mathbb{C}$ be such that $det(C-\lambda D) \neq 0$. Then the following statements hold:*

a) If λ is an eigenvalue of the REP (2.9), then it is an eigenvalue of the GEP (2.10).

b) Let λ be an eigenvalue of the GEP (2.10) and $z = [z_1^T, z_2^T, \cdots, z_d^T, y^T]^T$ be a corresponding eigenvector, where z_i are vectors of length n for $i = 1, 2, \ldots, d$. Then $z_1 \neq 0$ and $R(\lambda)z_1 = 0$, namely, λ is an eigenvalue of the REP (2.9) and z_1 is a corresponding eigenvector. Moreover, the algebraic and geometric multiplicities of λ for the REP (2.9) and GEP (2.10) are the same.

2.1.4 Some applications of REPs

In this section, we present some representative examples of REPs that arise from different applications.

- **Loaded elastic string** [14, 94, 99]. The following REP arises from the finite element discretization of a boundary value problem describing the eigenvibration of a string with a load of mass attached by an elastic spring:

$$\left(A - \lambda B + \frac{\lambda}{\lambda - \sigma}E\right)x = 0,$$

where A and B are $n \times n$ real, tridiagonal and symmetric positive definite matrices, $E = e_n e_n^T$, where e_n is the last column of the identity matrix of size n and $\sigma > 0$ is a parameter.

- **Vibration of a fluid-solid structure** [4, 74, 76, 99, 108]. In this example, the REP arises from the simulation of mechanical vibrations of fluid-solid structures. It is of the form

$$\left(A - \lambda B + \sum_{i=1}^{k} \frac{\lambda}{\lambda - \sigma_i} E_i \right) x = 0, \tag{2.14}$$

where the poles σ_i, $i = 1, \ldots, k$ are positive, the $n \times n$ real matrices A and B are nonzero symmetric positive definite and $E_i = C_i C_i^T$, where $C_i \in \mathbb{R}^{n \times r_i}$ has rank r_i for $i = 1, 2, \ldots, k$.

- **Damped vibration of a structure** [4, 76, 99]. This is a REP arising from the free vibrations of a structure if one uses a viscoelastic constitutive relation to describe the behavior of a material. The REP has the form

$$\left(\lambda^2 M + K - \sum_{i=1}^{k} \frac{1}{1 + b_i \lambda} \Delta G_i \right) x = 0, \tag{2.15}$$

where the $n \times n$ real matrices M and K are symmetric positive definite, b_j are relaxation parameters over the k regions, and ΔG_j is an assemblage of element stiffness matrices over the region with the distinct relaxation parameters.

- **Approximation of NLEPs by REPs** [45, 46]. Consider the NLEP

$$A(\lambda) x = 0, \tag{2.16}$$

where $\lambda \in \Sigma$ with a compact target set $\Sigma \subset \mathbb{C}$, $x \in \mathbb{C}^n \backslash \{0\}$, and a family of matrices $A(\lambda) : \Sigma \to \mathbb{C}^{n \times n}$ depending analytically on λ, i.e., each component of $A(\lambda)$ is an analytic function of λ. In order to solve the NLEP (2.16), the matrix $A(\lambda)$ is approximated by a rational function $R_N(\lambda)$, resulting in a REP. The approximation is constructed with a linear rational interpolation procedure.

  Given a sequence of interpolation nodes $\sigma_0, \sigma_1, \ldots,$ and another sequence of nonzero poles $\xi_1, \xi_2, \ldots,$, the sequence of rational basis functions

$$b_j(\lambda) = \frac{1}{\beta_0} \prod_{k=1}^{j} \frac{\lambda - \sigma_{k-1}}{\beta_k (1 - \lambda / \xi_k)}, \quad \text{for } j = 0, 1, \ldots,$$

  is considered, where $\beta_0, \beta_1, \ldots,$ are nonzero scaling parameters. Then, a sequence of matrices $D_j \in \mathbb{C}^{n \times n}$, $j = 0, 1, \ldots,$ is constructed such that for each $N = 0, 1, \ldots,$ the rational eigenvalue problem

$$R_N(\lambda) = b_0(\lambda) D_0 + b_1(\lambda) D_1 + \cdots + b_N(\lambda) D_N$$

interpolates $A(\lambda)$ in Hermite's sense, i.e., counting multiplicities, at the nodes $\sigma_0, \sigma_1, \ldots, \sigma_N$. The matrices D_j are called rational divided difference matrices and they can be computed, when all the interpolation nodes σ_j are distinct, by the recursion

$$D_j = \frac{A(\sigma_j) - b_0(\sigma_j)D_0 - \cdots - b_{j-1}(\sigma_j)D_{j-1}}{b_j(\sigma_j)} = \frac{A(\sigma_j) - R_{j-1}(\sigma_j)}{b_j(\sigma_j)}.$$

2.2 Basics on Krylov methods

In this section, we introduce some basic concepts on Krylov methods. We start in Section 2.2.1 with the well-known Arnoldi method and its shift-and-invert variant [86] and we present in Section 2.2.2 the rational Krylov procedure [82, 83]. As we mentioned in Chapter 1, in (1.1), a Krylov subspace is a vector subspace of the form

$$\mathcal{K}_m(A, v_1) = \text{ span}\{v_1, v_2, v_3, \ldots, v_m\}, \quad v_{i+1} = Av_i, \text{ for } i = 1, \ldots, m-1,$$

where $A \in \mathbb{C}^{n\times n}$ and $v_1 \in \mathbb{C}^n$. These methods are used for solving, for example, matrix equations and matrix eigenvalue problems, and, in this section, we will focus on the latter.

2.2.1 The Arnoldi method

One of the most important method based on Krylov subspaces is the Arnoldi method. This procedure was introduced in 1951 [7] as a way to reduce a dense matrix into Hessenberg form. Arnoldi introduced this method precisely in this manner and he suspected that his process could approximate some eigenvalues if stopped before completion, which is the way is currently used. The procedure starts building an orthogonal basis of the Krylov subspace $\mathcal{K}_m(A, v_1)$. The classical Arnoldi method is presented in Algorithm 2.1, which computes an orthonormal basis of a Krylov subspace. Algorithm 2.1 is combined in Algorithm 2.2 with the computation of Ritz pairs for getting approximations of certain eigenvalues of A.

It is important to remark that Algorithm 2.1 is based on the classical Gram-Schmidt (CGS) orthogonalization process. The reader can see that a new vector $\hat{v}$ is constructed and then orthogonalized against all the previous v_i vectors in each step.

Note that in step 2 of Algorithm 2.1, we have used MATLAB notation for submatrices through block indices. From now on, we introduce the standard notation

$$h_j := H_j(1:j, j), \quad \underline{h}_j := \underline{H}_j(1:j+1, j). \tag{2.17}$$

to define the Hessenberg matrices H_j and $\underline{H}_j$, respectively, obtained by the Arnoldi method.

Algorithm 2.1 Classical Arnoldi method

Input: $A \in \mathbb{C}^{n\times n}$ and a unit vector $v_1 \in \mathbb{C}^n$.

Output: The matrix V_{m+1} whose columns are an orthonormal basis for $\mathcal{K}_{m+1}(A, v_1)$ and the Hessenberg matrix H_m in Proposition 2.20.

Initialize $V_1 = [v_1]$.

for $j = 1, 2, \ldots, m$ **do**

1. Perform matrix-vector multiplication $\hat{v} = Av_j$.
2. Compute $h_j = V_j^* \hat{v}$, where $h_j = H_j(1 : j, j)$.
3. Compute $\tilde{v} = \hat{v} - V_j h_j$.
4. Compute the coefficient $h_{j+1,j} = \|\tilde{v}\|_2$.

if $h_{j+1,j} = 0$ **then**

stop the procedure

end if

5. Compute the next vector $v_{j+1} = \frac{1}{h_{j+1,j}}\tilde{v}$.
6. Update $V_{j+1} = [V_j \quad v_{j+1}]$.

end for

Proposition 2.18. *([86, Proposition 6.5]) The vectors $v_1, v_2, \ldots, v_m$ obtained by Algorithm 2.1 form an orthonormal basis of the subspace $\mathcal{K}_m(A, v_1)$.*

Remark 2.19. *Since the classical Arnoldi method is based on the classical Gram-Schmidt process, it is natural to think that this procedure can be implemented in different ways. There are several alternatives to implement the Arnoldi process, for example, the modified Gram-Schmidt process (MGS) can be used instead of CGS, and then the modified Arnoldi method is obtained [86, Section 6.2]. There is no difference in exact arithmetic between CGS and MGS, both algorithms produce the same orthonormal basis, however, the formulation for the modified Arnoldi method is numerically superior than the classical Arnoldi method because (as occurs for MGS with respect to CGS) with the classical Arnoldi method, the created basis $\{v_i\}_{i=1}^m$ loses orthogonality in finite arithmetic. However, if the reorthogonalization technique is applied, both CGS and MGS algorithms will have a similar performance. Therefore, in practice, it is enough to apply CGS algorithm with reorthogonalization for the Arnoldi procedure (as in ARPACK [67]), and here, by reorthogonalization, we refer to repeat the Gram-Schmidt algorithm one more time. A summary for this technique and its rounding error analysis can be found in [1, 24, 40].*

Another alternative is to perform the Arnoldi method via Householder QR (HQR) algorithm [86, Section 6.2]. HQR is the best of the three options to produce the basis $\{v_i\}_{i=1}^m$ with high orthogonality, but it is also computationally more expensive.

An important result obtained from the Arnoldi process is the well-known Arnoldi recurrence relation. This relation is summarized in Proposition 2.20.

Proposition 2.20. *([86, Proposition 6.6]) Denote by V_m the $n \times m$ matrix with*

columns vectors $v_1, v_2, \ldots, v_m$ obtained by Algorithm 2.1 and by H_m the $m \times m$ Hessenberg matrix (2.17) *whose nonzero entries are defined by Algorithm 2.1. Then the following relations hold:*

$$AV_m = V_m H_m + h_{m+1,m} v_{m+1} e_m^T, \tag{2.18}$$

$$V_m^* A V_m = H_m, \tag{2.19}$$

where e_m represents the m-th column of I_m.

In Algorithm 2.1, if $h_{j+1,j} = 0$ occurs, then the norm of $\tilde{v}$ vanishes at some step j. In this situation, the vector v_{j+1} can not be computed and Algorithm 2.1 stops. In this case, we say that the algorithm breaks down. Proposition 2.22 determines the conditions under which this situation occurs.

Definition 2.21. *The minimal polynomial of a vector $v \in \mathbb{C}^n$ associated to $A \in \mathbb{C}^{n\times n}$ is the nonzero monic polynomial p of lowest degree such that $p(A)v = 0$.*

Proposition 2.22. *[86, Proposition 6.7] The Arnoldi algorithm breaks down at step j (i.e., $\tilde{v} = 0$ in step 3 of Algorithm 2.1) if and only if the minimal polynomial of v_1 associated to A is of degree j. Moreover, in this case the subspace $\mathcal{K}_j(A, v_1)$ is invariant for A and the approximate eigenvalues and eigenvectors computed in Algorithm 2.2 are exact.*

For simplicity, it is assumed in this chapter that breakdown does not occur in the Arnoldi method for all $j = 1, \ldots, m$, and, in this case, the matrix $\underline{H}_j$ is unreduced. In each iteration of the Arnoldi method, we can compute j approximate eigenvalues and eigenvectors of A by solving the small eigenvalue problem

$$H_j t_i = \lambda_i t_i, \quad i = 1, \ldots, j, \tag{2.20}$$

where H_j is the $j \times j$ upper Hessenberg matrix obtained by removing the last row of $\underline{H}_j$. Then, we call $(\lambda_i, x_i = V_j t_i)$ a Ritz pair of A. It is necessary to emphasize that the vectors x_i are not computed in each iteration, because this would be a very expensive step, since V_j is a large dense matrix.

Another important aspect to analyze is a convergence criterion. Since we are projecting our problem into a problem of smaller dimension, it is natural to expect the existence of an inexpensively way to compute the residual norm. For most of the Krylov methods, a cheap residual norm can be computed, and particularly, for the Arnoldi method, this result is summarized in Proposition 2.23.

Proposition 2.23. *[86, Proposition 6.8] Let H_j be the upper $j\times j$ Hessenberg matrix obtained by the Arnoldi method in Algorithm 2.1 after j iterations and by removing the last row of $\underline{H}_j$. Let t_i be an eigenvector of H_j associated with the eigenvalue λ_i and x_i the Ritz approximate eigenvector of A with $x_i = V_j t_i$. Then,*

$$(A - \lambda_i I_n) x_i = h_{j+1,j} e_j^* t_i v_{j+1},$$

and, therefore,

$$\|(A - \lambda_i I_n)x_i\|_2 = h_{j+1,j}|e_j^* t_i|. \tag{2.21}$$

Proposition 2.23 states that the residual norm is equal to the modulus of the last component of the eigenvector t_i multiplied by $h_{j+1,j}$. This residual norm brings a helpful and inexpensive way to derive a stopping criterion. The Arnoldi method to compute approximate eigenvalues of a matrix A is presented in Algorithm 2.2.

Algorithm 2.2 Arnoldi method for solving SEPs

Input: $A \in \mathbb{C}^{n\times n}$ and a unit vector $v_1 \in \mathbb{C}^n$.

Output: A matrix V_{m+1} whose columns form an orthonormal basis for $\mathcal{K}_{m+1}(A, v_1)$ and the Ritz pairs (λ, x) of A corresponding to $\mathcal{K}_m(A, v_1)$.

Initialize $V_1 = [v_1]$.

for $j = 1, 2, \ldots, m$ **do**

1. Perform Algorithm 2.1 until step 5 obtaining h_j and v_{j+1}.
2. Compute the eigenpairs (λ_i, t_i) of H_j and test for convergence (2.21).
3. Update $V_{j+1} = [V_j \quad v_{j+1}]$.

end for

4. Compute the eigenvectors $x_i = V_j t_i$.

The Arnoldi method tends to produce good approximations of the eigenvalues of A with largest absolute values, then it can be improved by introducing the idea of shift-and-invert. Suppose we must compute the p eigenvalues of A closest to $\theta \in \mathbb{C}$. In order to obtain the required eigenvalues, a Krylov subspace for $(A - \theta I_n)^{-1}$ is constructed and denoted by $\mathcal{K}_m(A, v_1, \theta)$, where

$$\mathcal{K}_m(A, v_1, \theta) := \text{span}\{v_1, (A-\theta I_n)^{-1}v_1, (A-\theta I_n)^{-1}v_2, \ldots, (A-\theta I_n)^{-1}v_{m-1}\}, \tag{2.22}$$

$A \in \mathbb{C}^{n\times n}$ and $v_{i+1} = (A - \theta I_n)^{-1}v_i$, for $i = 1, \ldots, m-1$. The shift-and-invert Arnoldi algorithm is detailed in Algorithm 2.3. Note that for the shift-and-invert Arnoldi process it is necessary to solve a linear system of size n in step 1 of Algorithm 2.3, usually by a LU decomposition. Since the shift θ is fixed, the LU decomposition is computed just once at the beginning. The remaining steps proceed as the Arnoldi method.

Several techniques can be implemented to improve the Arnoldi method. When m gets large, the storage cost and computational cost of the orthogonalization of the Arnoldi method can become unacceptably high. For this reason, some form of reduction of the basis is desirable. To achieve this goal, different techniques have been developed. The most important one is called *implicit restarting* and it appeared for the first time in [95]. That implementation of implicit restarting uses QR steps to reduce the basis and throw away a part of the spectrum we are not interested in. A different way to reduce the subspace dimension is *purging* [96]. Here, the idea is: first, consider the Schur factorization of H_m and then purge the undesired part of

Algorithm 2.3 Shift-and-invert Arnoldi method

Input: $A \in \mathbb{C}^{n\times n}$, a unit vector $v_1 \in \mathbb{C}^n$ and a shift $\theta \in \mathbb{C}$.
Output: The matrix V_{m+1} whose columns are an orthonormal basis for $\mathcal{K}_m(A, v_1, \theta)$ and the corresponding Hessenberg matrix H_m.

Initialize $V_1 = [v_1]$.
for $j = 1, 2, \ldots, m$ **do**
1. Compute $\hat{v} = (A - \theta I_n)^{-1} v_j$ by solving a linear system.
2. Compute $h_j = V_j^* \hat{v}$.
3. Compute $\tilde{v} = \hat{v} - V_j h_j$.
4. Compute $h_{j+1,j} = \|\tilde{v}\|_2$.

if $h_{j+1,j} = 0$ **then**
stop the procedure
end if

5. Compute the next vector $v_{j+1} = \frac{1}{h_{j+1,j}} \tilde{v}$.
6. Update $V_{j+1} = [V_j \quad v_{j+1}]$.

end for

this factorization. Finally, another technique is *locking* [57]. The idea of locking is to set some small elements explicitly to zero assuming that the Schur vectors are exact. These techniques can be implemented in methods based on the Arnoldi method (or the rational Krylov method) by adapting the procedure to the representation of the Krylov vectors.

Finally, the Arnoldi method can be improved in order to solve GEPs instead of SEPs and also, several shifts can be incorporated instead of just one. These improvements are developed in the rational Krylov method, which is presented in the next section.

2.2.2 The classical rational Krylov method

We revise in this section the rational Krylov method (RKS) for GEPs since the algorithm R-CORK presented in this PhD Thesis is based on this method. The rational Krylov method [82, 83] is a generalization for computing eigenvalues of matrices and of matrix pencils of the shift-and-invert Arnoldi method. The main differences between these methods are basically two: in rational Krylov methods we can change the shift θ_j at each iteration instead of fixing the shift as in the shift-and-invert Arnoldi method. Also, the information of the approximate eigenvalues is contained in two upper Hessenberg matrices $\underline{H}_j$ and $\underline{K}_j$ instead of in only one matrix. In Algorithm 2.4 we present a basic pseudocode of the rational Krylov method that summarizes its main steps and guides the developments in the rest of this subsection, which are a very brief sketch of the rational Krylov method. The reader can find more details in [82, 83, 104]. This method produces an orthonormal

Algorithm 2.4 Rational Krylov method for solving GEPs

Input: A and B square matrices of size n and an initial vector u_1 with $\|u_1\|_2 = 1$.
Output: The matrix U_{m+1} whose columns are an orthonormal basis of $\mathcal{K}_{m+1}(A, B, u_1, \theta_{1,\ldots,m})$, and the Ritz pairs (λ, x) of $A - \lambda B$, corresponding to the rational Krylov subspace $\mathcal{K}_m(A, B, u_1, \theta_{1,\ldots,m-1})$.
Initialize $U_1 = [u_1]$.
for $j = 1, 2, \ldots, m$ **do**
1. Choose the shift θ_j.
2. Set the continuation combination z_j.
3. Compute $\hat{u} = (A - \theta_j B)^{-1} B w_j$, where $w_j = U_j z_j$.
4. Compute $h_j = U_j^* \hat{u}$.
5. Compute $\tilde{u} = \hat{u} - U_j h_j$.
6. Get the new vector $u_{j+1} = \tilde{u}/h_{j+1,j}$ with $h_{j+1,j} = \|\tilde{u}\|_2$.
7. Update $U_{j+1} = [U_j \quad u_{j+1}]$.
8. Compute the eigenpairs (λ_i, t_i) of (2.27) and test for convergence.

end for
9. Compute the eigenvectors $x_i = U_{j+1} \underline{H}_j t_i$, $i = 1, \ldots, j$.

basis for the subspace

$$\mathcal{K}_m(A, B, u_1, \theta_{1,\ldots,m-1}) := \text{span}\{u_1, (A - \theta_1 B)^{-1} B w_1, \ldots, (A - \theta_{m-1} B)^{-1} B w_{m-1}\} \tag{2.23}$$

with A, $B \in \mathbb{C}^{n \times n}$, $u_1 \in \mathbb{C}^n$ and $w_i = U_i z_i$, for $i = 1, 2, \ldots, m-1$, where w_i is called a *continuation vector*, and U_i denotes the matrix formed by the vectors u_i, this is,

$$U_i = [u_1 \; u_2 \; \ldots \; u_i] \; \in \mathbb{C}^{n \times i}, \tag{2.24}$$

where $u_{i+1} = (A - \theta_i B)^{-1} B w_i$ for $i = 1, 2, \ldots, m-1$ and $z_i \in \mathbb{C}^i$. Usually, the vector z_i is chosen as the canonical vector e_i, and then $w_i = u_i$.

Remark 2.24. *Note that the Krylov vectors in Algorithm 2.4 are represented by u_j, whereas that the Krylov vectors for Arnoldi are represented as v_j in Algorithm 2.1. From now on, we will represent by v_j the Krylov vectors obtained by the Arnoldi method (or its variants) and by u_j the Krylov vectors obtained by the rational Krylov method (or its variants).*

By using the equalities for $\hat{u}$ and $\tilde{u}$ from steps 3 and 5 in Algorithm 2.4 we obtain at the i-th iteration:

$$(A - \theta_i B)^{-1} B w_i = U_{i+1} \underline{h}_i,$$

with $\underline{h}_i = [h_i^* \quad h_{i+1,i}^*]^*$. After j steps of the rational Krylov method, we obtain the classic rational Krylov recurrence relation [83]:

$$A U_{j+1} \underline{H}_j = B U_{j+1} \underline{K}_j, \tag{2.25}$$

where $\underline{H}_j, \underline{K}_j \in \mathbb{C}^{(j+1)\times j}$ are upper Hessenberg matrices and

$$\underline{K}_j = \underline{H}_j \mathrm{diag}(\theta_1, \theta_2, \ldots, \theta_j) + \underline{Z}_j. \tag{2.26}$$

with the triangular matrix

$$\underline{Z}_j = \begin{bmatrix} z_1 & z_2 & \ldots & z_j \\ 0_{j\times 1} & 0_{(j-1)\times 1} & \ldots & 0_{1\times j} \end{bmatrix}$$

built up from the continuation combinations used in step 2 in Algorithm 2.4.

For simplicity, we assume that breakdowns do not occur in the rational Krylov method, this is, $h_{j+1,j} \neq 0$ for all $j = 1, \ldots, m$, and in this case the upper Hessenberg matrix $\underline{H}_j$ is unreduced. We can approximate in each iteration of Algorithm 2.4 the corresponding j eigenvalues and eigenvectors of the pencil $A - \lambda B$ by solving the small generalized eigenvalue problem:

$$K_j t_i = \lambda_i H_j t_i, \quad t_i \neq 0, \tag{2.27}$$

where H_j and K_j are the $j \times j$ upper Hessenberg matrices obtained by removing the last rows of $\underline{H}_j$ and $\underline{K}_j$, respectively. Then, we call $(\lambda_i, x_i = U_{j+1}\underline{H}_j t_i)$ a Ritz pair of (A, B). We emphasize that the approximate eigenvectors x_i are not computed in each iteration, since this would be very expensive, and that the test for convergence in step 8 of Algorithm 2.4 can be performed in an inexpensive way by using only the small vectors t_i, as it is done in most Krylov methods.

2.3 Krylov methods for quadratic eigenvalue problems

In this section, we introduce three Krylov methods to solve the quadratic eigenvalue problem (QEP):

$$(\lambda^2 M + \lambda C + K)x = 0, \quad x \neq 0, \tag{2.28}$$

where M, C, and K are square, large-scale and sparse matrices of size $n \times n$ and x is an eigenvector of size n. These methods are presented in three subsections.

In Section 2.3.1 we introduce the SOAR method [8], which projects the QEP (2.28) into a QEP of smaller size, and then solves the reduced problem via linearization. It is important to remark that the SOAR method considers the matrices M, C and K as real matrices, and it computes a real Krylov subspace. However, since sometimes, the matrices M, C and K arise from applications as the Fourier transformation of the spatial discretization by finite elements of the equation of motion, it is natural to consider these matrices with complex entries. The other methods described in this section, i.e., Q-Arnoldi and TOAR, construct complex Krylov subspaces to solve the problem (2.28) for complex M, C, and K.

In Section 2.3.2 the Q-Arnoldi method [75] is summarized. In this procedure, a new representation for the Krylov vectors of a linearization of the large QEP is presented, resulting in a memory saving method. However, this representation makes the algorithm potentially unstable, and this instability disappears with the representation for the Krylov vectors given in the TOAR method [100, 71], which is presented in Section 2.3.3. Both Q-Arnoldi and TOAR can be generalized for PEPs of any degree, and we will discuss about this in Section 2.4. It is important to remark than SOAR, Q-Arnoldi, and TOAR follow the spirit of the Arnoldi method. From now on, we assume that the leading coefficient M in (2.28) is nonsingular. Also, since both Q-Arnoldi and TOAR are often interested in solving the QEP (2.28) for small eigenvalues, we assume that the matrix K is invertible.

Since both SOAR and Q-Arnoldi are potentially unstable, the TOAR method is preferred in the literature for solving QEPs. The key idea of a compact representation for the Krylov vectors results in a numerically stable and memory saving procedure which can be generalized for PEPs (see Section 2.4) and REPs (see Chapter 4).

2.3.1 The SOAR method

In order to introduce the second-order Arnoldi method (SOAR) it is necessary to define a m-th second order Krylov subspace.

Definition 2.25. *Let A and B be square matrices of size $n \times n$ and let u be a nonzero vector of size n. Then, the sequence*

$$r_0, r_1, \ldots, r_{m-1}, \tag{2.29}$$

where

$$\begin{aligned} r_0 &= u, \quad r_1 = Ar_0, \\ r_j &= Ar_{j-1} + Br_{j-2} \quad \text{for } 2 \leq j \leq m-1, \end{aligned} \tag{2.30}$$

is called a second-order Krylov sequence based on A, B, and u. The space

$$\mathcal{G}_m(A, B, u) = \text{span}\{r_0, r_1, \ldots, r_{m-1}\}$$

is called an m-th second order Krylov subspace.

Remark 2.26. *The subspace $\mathcal{G}_m(A, B, u)$ generalizes the standard Krylov subspace $\mathcal{K}_m(A, u)$ in the way that when B is a zero matrix, the second-order Krylov subspace is a standard Krylov subspace, this is,*

$$\mathcal{G}_m(A, 0_n, u) = \mathcal{K}_m(A, u).$$

Note that the j-th vector r_j defined in (2.30) can be written as

$$r_j = p_j(A, B)u,$$

where u is the starting vector and $p_j(\alpha, \beta)$ are polynomials in α and β, defined by the recurrence

$$p_j(\alpha, \beta) = \alpha p_{j-1}(\alpha, \beta) + \beta p_{j-2}(\alpha, \beta),$$

with $p_0(\alpha, \beta) = 1$ and $p_1(\alpha, \beta) = \alpha$. As we mentioned in Section 2.1.2, the problem (2.28) can be converted into an equivalent GEP

$$\mathbf{A}_S\mathbf{y} = \lambda\mathbf{B}_S\mathbf{y}, \tag{2.31}$$

with

$$\mathbf{A}_S = \begin{bmatrix} -C & -K \\ I_n & 0 \end{bmatrix}, \quad \mathbf{B}_S = \begin{bmatrix} M & 0 \\ 0 & I_n \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} \lambda x \\ x \end{bmatrix}, \tag{2.32}$$

and also, the equation (2.31) can be solved as the standard eigenvalue problem

$$\mathbf{G}_S\mathbf{y} = \lambda\mathbf{y}, \tag{2.33}$$

where

$$\mathbf{G}_S = \mathbf{B}_S^{-1}\mathbf{A}_S = \begin{bmatrix} -M^{-1}C & -M^{-1}K \\ I_n & 0 \end{bmatrix}. \tag{2.34}$$

For simplicity, consider

$$A_S := -M^{-1}C, \quad B_S := -M^{-1}K, \quad \text{and } \mathbf{u} = [u_1^T \quad 0_{n\times 1}^T]^T, \tag{2.35}$$

with $u_1 \in \mathbb{R}^n$, then we can derive immediately that, the second-order Krylov vectors r_j associated to $\mathcal{G}_m(A_S, B_S, u_1)$ defined in (2.29), and the standard Krylov vectors $(\mathbf{G}_S)^j\mathbf{u}$ associated to $\mathcal{K}_m(\mathbf{G}_s, \mathbf{u})$ of length $2n$ are related as follows:

$$\begin{bmatrix} r_j \\ r_{j-1} \end{bmatrix} = (\mathbf{G}_S)^j\mathbf{u}, \quad \text{for } j \geq 1.$$

By this representation, we can conclude that the generalized Krylov sequence r_j defines the entire standard Krylov sequence based on $\mathbf{G}_S$ and $\mathbf{u}$. For this reason, the authors of [8] preferred to work with the subspace $\mathcal{G}_j(A_S, B_S, u_1) \in \mathbb{R}^n$ instead of $\mathcal{K}_j(\mathbf{G}_S, \mathbf{u}) \in \mathbb{R}^{2n}$, since the vectors in $\mathcal{G}_j(A_S, B_S, u_1)$ have smaller size than those in $\mathcal{K}_j(\mathbf{G}_S, \mathbf{u})$. In order to develop a practical method, the authors of [8] created the SOAR procedure, which constructs an orthonormal basis $\{q_1, q_2, \ldots, q_j\}$ for $\mathcal{G}_j(A_S, B_S, u_1)$. The SOAR procedure is presented in Algorithm 2.5. Note that the matrices A_S and B_S defined in (2.35) that appear in step 3 of Algorithm 2.5 are not constructed explicitly. The vector $\hat{q}$ in step 3 of Algorithm 2.5 is computed first by multiplying by the matrices C and K and then, a linear system that involves the matrix M is solved. Lemma 2.27 proves that Algorithm 2.5 generates an orthonormal basis of the second-order Krylov subspace $\mathcal{G}_j(A_S, B_S, u_1)$.

Lemma 2.27. *[8, Theorem 2.3] Let Q_j be the matrix obtained in Algorithm 2.5, with $Q_j = [q_1\, q_2\, \cdots\, q_j]$. If $h_{j+1,j} \neq 0$ in Algorithm 2.5 then the columns of Q_j form an orthonormal basis of the second-order Krylov subspace $\mathcal{G}_j(A_S, B_S, u_1)$, i.e.,*

$$\text{span}\{Q_j\} = \mathcal{G}_j(A_S, B_S, u_1).$$

Algorithm 2.5 SOAR procedure

Input: Coefficient matrices $M, C, K \in \mathbb{R}^{n\times n}$ that define $P(\lambda)$ in (2.28) and $u_1 \in \mathbb{R}^n$, with $u_1 \neq 0$.

Output: The matrix Q_{m+1} whose columns are an orthonormal basis for $\mathcal{G}_{m+1}(A_S, B_S, u_1)$ with A_S and B_S defined as in (2.35).

1. Normalize $q_1 = u_1/\|u_1\|_2$, and initialize $Q_1 = [q_1]$.
2. Set $p_1 = 0_{n\times 1}$, and initialize $P_1 = [p_1]$.

for $j = 1, 2, \ldots, m$ **do**

3. Compute $\hat{q} = A_S q_j + B_S p_j$ as $\hat{q} = -M^{-1}(Cq_j + Kp_j)$.
4. Set $\hat{p} = q_j$.
5. Compute $h_j = Q_j^T \hat{q}$.
6. Compute $\tilde{q} = \hat{q} - Q_j h_j$.
7. Compute $\tilde{p} = \hat{p} - P_j h_j$.
8. Compute $h_{j+1,j} = \|\tilde{q}\|_2$.

if $h_{j+1,j} = 0$ **then**
stop the procedure
end if

10. Compute the new vectors $q_{j+1} = \tilde{q}/h_{j+1,j}$, $p_{j+1} = \tilde{p}/h_{j+1,j}$.
11. Update $Q_{j+1} = [Q_j \quad q_{j+1}]$, $P_{j+1} = [P_j \quad p_{j+1}]$.

end for

Some basic relations between the matrices produced by Algorithm 2.5 can be considered in order to develop a memory saving SOAR process. If $P_j, Q_j \in \mathbb{R}^{n\times j}$ denote the matrices with column vectors $p_1, \ldots, p_j$ and $q_1, \ldots, q_j$, respectively, where $p_i, q_i \in \mathbb{R}^n$ for $i = 1, \ldots, j$, and $\underline{H}_j \in \mathbb{R}^{(j+1)\times j}$ denotes the upper Hessenberg matrix defined in the algorithm, the following relations hold

$$A_S Q_j + B_S P_j = Q_{j+1}\underline{H}_j, \tag{2.36}$$

$$Q_j = P_{j+1}\underline{H}_j. \tag{2.37}$$

Equations (2.36)-(2.37) can be written in the compact form

$$\begin{bmatrix} A_s & B_s \\ I & 0 \end{bmatrix} \begin{bmatrix} Q_j \\ P_j \end{bmatrix} = \begin{bmatrix} Q_{j+1} \\ P_{j+1} \end{bmatrix} \underline{H}_j. \tag{2.38}$$

Relations (2.36)-(2.37) allow to develop a SOAR procedure that reduces the memory cost by almost half.

From now on, some MATLAB notation will be used. Note that, by (2.37) and nothing that $p_1 = 0$ in Algorithm 2.5, we have

$$Q_j = P_{j+1}\underline{H}_j = P_{j+1}(:, 2 : j+1)\underline{H}_j(" : j+1, 1 : j).$$

Then, (2.36) can be rewritten as

$$A_s Q_j + B_s Q_j S_j = Q_j \underline{H}_j, \tag{2.39}$$

where S_j is an $j \times j$ strictly upper triangular matrix of the form

$$S_j = \begin{bmatrix} 0 & (\underline{H}_j(2 : j, 1 : j-1))^{-1} \\ 0 & 0 \end{bmatrix}.$$

Equation (2.39) suggests a method for computing q_{j+1} from $q_1, \ldots, q_j$ without constructing the vectors $p_1 \ldots, p_j$ explicitly. This improvement leads to Algorithm 2.6, which reduces the memory cost of the SOAR procedure in Algorithm 2.5 about a half.

Algorithm 2.6 Memory saving SOAR procedure

Input: Coefficient matrices $M, C, K \in \mathbb{R}^{n\times n}$ that define $P(\lambda)$ in (2.28) and $u_1 \in \mathbb{R}^n$, with $u_1 \neq 0$.

Output: The matrix Q_{m+1} whose columns are an orthonormal basis for $\mathcal{G}_{m+1}(A_S, B_S, u_1)$ with A_S and B_S defined as in (2.35).

1. Normalize $q_1 = u_1/\|u_1\|_2$, and initialize $Q_1 = [q_1]$. Set $f = 0$.

for $j = 1, 2, \ldots, m$ **do**

2. Compute $\hat{q} = A_S q_j + B_S f$ as $\hat{q} = -M^{-1}(Cq_j + Kf)$.
3. Compute $h_j = Q_j^T \hat{q}$.
4. Compute $\tilde{q} = \hat{q} - Q_j h_j$.
5. Compute $h_{j+1,j} = \|\tilde{q}\|_2$.

if $h_{j+1,j} = 0$ **then**

stop the procedure

end if

6. Compute the new vector $q_{j+1} = \tilde{q}/h_{j+1,j}$.
7. Update $f = Q_j(\underline{H}_j(2 : j+1, 1 : j))^{-1} e_j$
8. Update $Q_{j+1} = [Q_j \quad q_{j+1}]$.

end for

More improvements, as deflation, were developed for the SOAR procedure and a detail discussion can be found in [8, pp. 647 - 650].

The SOAR orthogonalization procedure in Algorithm 2.6 can be applied combined with a projection method to solve the QEP (2.28), by using the orthogonal Rayleigh-Ritz approximation procedure. This method approximates a large-scale QEP by a small-scale QEP.

To apply the Rayleigh-Ritz approximation technique based on the subspace $\mathcal{G}_m(A_S, B_S, u_1)$ where A_S, B_S are defined as in (2.35), and $u_1 \in \mathbb{R}^n$, we seek an approximate eigenpair (λ, x), where $\lambda \in \mathbb{C}$ and $x \in \mathcal{G}_m(A_S, B_S, u_1)$, by imposing the orthogonal condition, also called the Galerkin condition:

$$(\lambda^2 M + \lambda C + K)x \quad \perp \quad \mathcal{G}_m(A_S, B_S, u_1),$$

which is equivalent to

$$v^T(\lambda^2 M + \lambda C + K)x = 0, \quad \text{for all } v \in \mathcal{G}_m(A_S, B_S, u_1). \tag{2.40}$$

Since $x \in \mathcal{G}_m(A_S, B_S, u_1)$, x can be expressed as

$$x = Q_m z, \tag{2.41}$$

where Q_m is an $n \times m$ matrix whose columns form an orthonormal basis of $\mathcal{G}_m(A_S, B_S, u_1)$ generated by the SOAR procedure in Algorithm 2.6 and $z \in \mathbb{R}^m$. By (2.40) and (2.41), it yields that λ and z must satisfy the reduced QEP

$$(\lambda^2 M_m + \lambda C_m + K_m)z = 0, \tag{2.42}$$

with

$$M_m = Q_m^T M Q_m, \quad C_m = Q_m^T C Q_m, \quad K_m = Q_m^T K Q_m. \tag{2.43}$$

The eigenpairs (λ, z) define the Ritz pairs (λ, x). It is important to remark that by the definition of the matrices M_m, C_m and K_m, the possible structures of M, C and D appearing in practice (symmetry, positive definite, etc) are preserved. Algorithm 2.7 presents the complete SOAR method for solving QEPs.

Algorithm 2.7 SOAR procedure to solve QEPs

Input: M, C, and K from the QEP (2.28) and $u_1 \in \mathbb{C}^n$.

Output: Approximate eigenpairs (λ, x) of the QEP (2.28).

1. Run the SOAR procedure (Algorithm 2.6) with A_S, B_S defined as in (2.35) and the starting vector u_1 to generate a $n \times m$ orthogonal matrix Q_m whose columns span an orthonormal basis of $\mathcal{G}_m(A_S, B_S, u_1)$.
2. Compute M_m, C_m, and K_m as in (2.43).
3. Solve the reduced QEP (2.42) for (λ, z) via linearization, and obtain the Ritz pairs (λ, x) where $x = Q_m z / \|Q_m z\|_2$.
4. Test the accuracy of the Ritz pairs (λ, x) by using the relative residual [103]:

$$\frac{\|(\lambda^2 M + \lambda C + K)x\|_2}{|\lambda|^2 \|M\|_2 + |\lambda| \|C\|_2 + \|K\|_2}. \tag{2.44}$$

	Arnoldi method	SOAR method
Orthogonalization cost	$(4j^2 + 10j)n$	$(2j^2 + 5j)n$
Memory cost	$2n(j + 1)$	$n(j + 2)$

Table 2.2: Asymptotic orthogonalization and memory costs for classical Arnoldi method and SOAR method after j iterations.

Now, we discuss memory and orthogonalization costs for Algorithm 2.6 and we compare them with the costs obtained by applying Arnoldi to a generic $2n \times 2n$ matrix. Since, in practice, $n \gg j$, where j is the counting iteration, we disregard those terms not containing n. The orthogonalization process is performed in steps 3-4-5-6 of Algorithm 2.6 for SOAR and, in steps 2-3-4-5 of Algorithm 2.1, for Arnoldi. It is well-known that the orthogonalization cost of Arnoldi applied to a generic $2n \times 2n$ matrix at iteration j is $8nj + 6n$ flops, whereas for Algorithm 2.6 the orthogonalization costs for the j-th iteration are: $2nj$ flops for step 3 and 4 each one, $2n$ flops for step 5 and n flops for step 6, resulting in $4nj + 3n$ flops. Therefore, the SOAR method in Algorithm 2.6 reduces the orthogonalization cost in almost a half. Table 2.2 summarizes the orthogonalization cost for each method. It is important to remark that Algorithm 2.6 requires the extra cost of computing f in step 7, which results in $2nj$ flops. However, even considering this computation, Algorithm 2.6 reduces the computational costs with respect to Arnoldi.

In terms of memory cost, Algorithm 2.6 is more efficient than Algorithm 2.1. Arnoldi requires to store $2n(j+1)$ numbers after j iterations, whereas SOAR stores roughly $n(j+2)$ numbers after j iterations (see Table 2.2), reducing in almost a half the storage requirements.

2.3.2 The Q-Arnoldi method

In this section, we present the quadratic Arnoldi algorithm (Q-Arnoldi) [75], which is a Krylov method for the solution of the quadratic eigenvalue problem. This method exploits the structure of the Krylov vectors of a linearization of (2.28), which results in a method that reduce the memory requirements by about a half.

From now on, note that we will change the notation that involves the matrices that contains in their columns the orthonormal Krylov vectors, these matrices will be represented as bold characters and they will be partitioned as follows:

$$\mathbf{V}_{j+1} = [\mathbf{V}_j \quad \mathbf{v}_{j+1}] = \begin{bmatrix} V_j^{(1)} & v_{j+1}^{(1)} \\ V_j^{(2)} & v_{j+1}^{(2)} \end{bmatrix}.$$

Consider the QEP (2.28) where M, C and $K \in \mathbb{C}^{n\times n}$. In order to solve (2.28), the standard procedure is followed, first, an appropriate linearization is constructed

for solving the PEP (2.28)

$$\mathbf{A}_Q \begin{bmatrix} \lambda x \\ x \end{bmatrix} = \lambda \mathbf{B}_Q \begin{bmatrix} \lambda x \\ x \end{bmatrix}, \tag{2.45}$$

and then, by multiplying by the inverse of $\mathbf{A}_Q$ or $\mathbf{B}_Q$, a standard eigenvalue problem is solved. However, since the Q-Arnoldi method is interested in the eigenvalues near to zero, the reversed problem is solved, this is,

$$\mathbf{G}_Q \mathbf{z} = \theta \mathbf{z}, \tag{2.46}$$

where

$$\mathbf{G}_Q := \mathbf{A}_Q^{-1} \mathbf{B}_Q, \quad \theta = \lambda^{-1} \quad \text{and } \mathbf{z} := \begin{bmatrix} x \\ \theta x \end{bmatrix}. \tag{2.47}$$

Since K is invertible, a natural choice for $\mathbf{A}_Q$ and $\mathbf{B}_Q$ is

$$\mathbf{A}_Q = \begin{bmatrix} D & 0 \\ 0 & K \end{bmatrix}, \quad \mathbf{B}_Q = \begin{bmatrix} 0 & D \\ -M & -C \end{bmatrix}, \tag{2.48}$$

where D can be any nonsingular matrix. By using (2.48), we obtain the SEP (2.46) where

$$\mathbf{G}_Q = \begin{bmatrix} 0 & I_n \\ G_1 & G_2 \end{bmatrix}, \quad G_1 := -K^{-1}M, \quad G_2 := -K^{-1}C. \tag{2.49}$$

The use of A_Q and B_Q is justified by Lemma 2.28.

Lemma 2.28. *[75, Lemma 2.1] The pencil $\mathbf{A}_Q - \lambda \mathbf{B}_Q$ is a linearization for (2.28) if and only if D is nonsingular.*

As we mentioned in Section 2.2.1, the problem (2.46) can be solved by using the Arnoldi method with $\mathbf{G}_Q$ and an initial vector $\mathbf{v}_1$. However, given the structure of $\mathbf{G}_Q$ and the Arnoldi recurrence relation (2.18), a new particular representation for the Krylov vectors can be developed.

Let $\mathbf{v}_j \in \mathbb{C}^{2n}$ be the j-th vector produced by the Arnoldi method (Algorithm 2.1) applied to $\mathbf{G}_Q$ and $\mathbf{V}_j \in \mathbb{C}^{2n \times j}$ the matrix that stores the first j Arnoldi vectors in its columns, and consider a partition for $\mathbf{v}_j$ and $\mathbf{V}_j$ as follows

$$\mathbf{V}_{j+1} = [\mathbf{V}_j \quad \mathbf{v}_{j+1}] = \begin{bmatrix} V_j^{(1)} & v_{j+1}^{(1)} \\ V_j^{(2)} & v_{j+1}^{(2)} \end{bmatrix} \tag{2.50}$$

with $V_j^{(1)}$, $V_j^{(2)} \in \mathbb{C}^{n \times j}$ and $v_{j+1}^{(1)}$, $v_{j+1}^{(2)} \in \mathbb{C}^n$. The Arnoldi recurrence relation (2.18) for the matrix $\mathbf{G}_Q$ can now be written as

$$\begin{bmatrix} 0 & I_n \\ G_1 & G_2 \end{bmatrix} \begin{bmatrix} V_j^{(1)} \\ V_j^{(2)} \end{bmatrix} - \begin{bmatrix} V_j^{(1)} \\ V_j^{(2)} \end{bmatrix} H_j = h_{j+1,j} \begin{bmatrix} v_{j+1}^{(1)} \\ v_{j+1}^{(2)} \end{bmatrix} e_j^T, \tag{2.51}$$

from which we deduce that

$$V_j^{(2)} = V_{j+1}^{(1)} \underline{H}_j. \tag{2.52}$$

This implies that only $V_j^{(1)}$, $v_{j+1}^{(1)}$, and $v_{j+1}^{(2)}$ need to be stored to evaluate the recurrence relation. Storing only these vectors results in an important reduction of the memory costs compared with the Arnoldi method applied to a general matrix of size $2n \times 2n$. Based on this key idea, Algorithm 2.8 implements the Q-Arnoldi method.

Algorithm 2.8 Q-Arnoldi method

Input: The matrices M, C and K from (2.28) and an initial vector $\mathbf{v}_1$.

Output: The matrix $V_{m+1}^{(1)}$ defined in (2.50) and the Hessenberg matrix $\underline{H}_m$ obtained by the Arnoldi method.

1. Compute $\mathbf{v}_1 = \mathbf{v}_1/\|\mathbf{v}_1\|_2$ and partition $\mathbf{v}_1$ as in (2.50), obtaining $v_1^{(1)}$ and $y = v_1^{(2)}$. Initialize $\mathbf{V}_1 = [\mathbf{v}_1]$.

for $j = 1, 2, \ldots, m$ **do**

2. Set $\hat{v}^{(1)} = y$ and compute $\hat{v}^{(2)} = G_1 v_j^{(1)} + G_2 y$, with G_1, G_2 as in (2.49)

3. Compute the Arnoldi coefficients

$$h_j = \begin{bmatrix} (V_{j-1}^{(1)})^* \hat{v}^{(1)} + \underline{H}_{j-1}^* ((V_j^{(1)})^* \hat{v}^{(2)}) \\ (v_j^{(1)})^* \hat{v}^{(1)} + y^* \hat{v}^{(2)} \end{bmatrix}.$$

4. Update

$$\begin{aligned} \tilde{v}^{(1)} &= \hat{v}^{(1)} - V_j^{(1)} h_j, \\ \tilde{v}^{(2)} &= \hat{v}^{(2)} - [V_j^{(1)} \quad y] \left(\begin{bmatrix} \underline{H}_{j-1} & 0 \\ 0 & 1 \end{bmatrix} h_j \right). \end{aligned}$$

5. Compute $h_{j+1,j} = \sqrt{\|\tilde{v}^{(1)}\|_2^2 + \|\tilde{v}^{(2)}\|_2^2}$ and normalize $v_{j+1}^{(1)} = \tilde{v}^{(1)}/h_{j+1,j}$, $y = \tilde{v}^{(2)}/h_{j+1,j}$.

6. Update $V_{j+1}^{(1)} = [V_j^{(1)} \quad v_{j+1}^{(1)}]$.

end for

Note that the computation of $\hat{v}^{(1)}$ and $\hat{v}^{(2)}$ is a common step for both Arnoldi and Q-Arnoldi, and $\hat{v}^{(2)}$ is computed by solving a linear system that involves the matrix K, which results in the most expensive step for both algorithms. Now, we compare the costs of the orthogonalization steps for both methods, this is, steps 2-3-4-5 in Algorithm 2.1 applied to a $2n \times 2n$ matrix, and steps 3-4-5 in Algorithm 2.8. Since in practice $n \gg j$, we may disregard those terms not containing n. For the j-th iteration of Algorithm 2.1, it is well-known that the cost of steps 2-3-4-5 of Arnoldi applied to a $2n \times 2n$ matrix is $8nj + 6n$ flops. For the j-th iteration of Algorithm 2.8 we have: the cost of step 3 and 4 is $4nj + 2n$ flops each one, and

for step 5 the cost is $6n$ flops, therefore, the cost of iteration j for Algorithm 2.8 is $8nj + 10n$ flops. These costs are summarized in Table 2.3. As we can see, the computational costs of the Q-Arnoldi method are slightly more expensive than the costs for the Arnoldi method.

However, there is an important improvement in terms of the storage cost. Since only $v_1^{(1)}, v_2^{(1)}, \ldots, v_{j+1}^{(1)}$ and $v_{j+1}^{(2)}$ need to be stored on iteration j in the Q-Arnoldi method, the memory requirements for the storage of the Arnoldi vectors for k iterations is $n(k+2)$ numbers, versus the $2n(k+1)$ numbers that need to be stored in the Arnoldi method.

	Arnoldi method	Q-Arnoldi method
Orthogonalization cost	$(4j^2 + 10j)n$	$(4j^2 + 14j)n$
Memory cost	$2n(j+1)$	$n(j+2)$

Table 2.3: Asymptotic orthogonalization and memory costs for classical Arnoldi method and Q-Arnoldi method after j iterations.

Since the matrix $\underline{H}_j$ appears in the representation of the orthonormal matrix $\mathbf{V}_j$, it introduces potential instability in finite precision arithmetic, which makes the Q-Arnoldi method potentially unstable. In [75], a complete analysis of the numerical stability is presented. Also, a detailed development of the Q-Arnoldi method for other linearizations and implementation of shifts for this method are included in [75].

After j iterations of the Q-Arnoldi method, and assuming that breakdown does not occur, the Ritz vectors corresponding to the Ritz value θ have the form

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} V_j^{(1)} z \\ V_{j+1}^{(1)} \underline{H}_j z \end{bmatrix},$$

where

$$H_j z = \theta z,$$

with H_j the upper Hessenberg matrix obtained by removing the last row of $\underline{H}_j$. When (θ, x) is an eigenpair of $\mathbf{G}_Q$, $x_2 = \theta x_1$. As a Ritz vector of (2.28), the vector x_2/θ or x_1 can be returned. We refer to [75] for a discussion of the best choice of these vectors.

2.3.3 The TOAR method

The TOAR method introduced in [100, 71], where TOAR stands for two-level orthogonal Arnoldi, follows the spirit of Q-Arnoldi in the sense that develops a new representation for the Krylov vectors of a linearization of (2.28). However, this

new representation is more stable than the one presented in the Q-Arnoldi method. The initial steps for the TOAR method are very similar to those of Q-Arnoldi and consider the linearization for the QEP (2.28)

$$\mathbf{A}_T\mathbf{z} = \lambda\mathbf{B}_T\mathbf{z}, \tag{2.53}$$

where

$$\mathbf{A}_T = \begin{bmatrix} I_n & 0 \\ 0 & K \end{bmatrix}, \quad \mathbf{B}_T = \begin{bmatrix} 0 & I_n \\ -M & -C \end{bmatrix}, \quad \mathbf{z} = \begin{bmatrix} \lambda x \\ x \end{bmatrix}. \tag{2.54}$$

Note that the matrices $\mathbf{A}_T$ and $\mathbf{B}_T$ are particular cases of $\mathbf{A}_Q$ and $\mathbf{B}_Q$ in (2.48) for $D = I_n$. Then, we solve the standard eigenvalue problem (2.47)

$$\mathbf{G}_Q\mathbf{z} = \theta\mathbf{z},$$

with $\mathbf{G}_Q$ partitioned as in (2.49). Consider $\mathbf{V}_{j+1}$ obtained by the Arnoldi method applied to $\mathbf{G}_Q$ and partitioned as in (2.50), and the Arnoldi recurrence relation (2.51). Then, a direct implication of (2.52) is

$$\text{span}\{V_j^{(1)}, V_j^{(2)}\} = \text{span}\{V_{j+1}^{(1)}\}. \tag{2.55}$$

Suppose that $Q_j \in \mathbb{C}^{n\times(j+1)}$ is an orthonormal basis for the space spanned by $V_{j+1}^{(1)}$, which is assumed to have dimension $j+1$. It follows from (2.55) that

$$\text{span}\{Q_j\} = \text{span}\{V_j^{(1)}, V_j^{(2)}\}. \tag{2.56}$$

From (2.56) we can represent

$$\mathbf{V}_j = \begin{bmatrix} V_j^{(1)} \\ V_j^{(2)} \end{bmatrix} = \begin{bmatrix} Q_jR_j^{(1)} \\ Q_jR_j^{(2)} \end{bmatrix} = \begin{bmatrix} Q_j & \\ & Q_j \end{bmatrix} \begin{bmatrix} R_j^{(1)} \\ R_j^{(2)} \end{bmatrix}, \tag{2.57}$$

for some matrices $R_j^{(1)}$, $R_j^{(2)} \in \mathbb{C}^{(j+1)\times j}$. Note that $\mathbf{R}_j := \begin{bmatrix} R_j^{(1)} \\ R_j^{(2)} \end{bmatrix}$ has orthonormal columns because V_j and Q_j have both orthonormal columns. By using (2.57), the Arnoldi recurrence relation (2.18) can be written as

$$\begin{bmatrix} 0 & I_n \\ G_1 & G_2 \end{bmatrix} \begin{bmatrix} Q_jR_j^{(1)} \\ Q_jR_j^{(2)} \end{bmatrix} = \begin{bmatrix} Q_jR_j^{(1)} \\ Q_jR_j^{(2)} \end{bmatrix} H_j + h_{j+1,j} \begin{bmatrix} Q_{j+1}r_{j+1}^{(1)} \\ Q_{j+1}r_{j+1}^{(2)} \end{bmatrix} e_j^*. \tag{2.58}$$

where $r_{j+1}^{(1)}$ and $r_{j+1}^{(2)}$ denote the last column of $R_{j+1}^{(1)}$ and $R_{j+1}^{(2)}$, respectively. The relation (2.58) is called the compact Arnoldi decomposition (CARD) of order j, for which only Q_{j+1}, $\mathbf{R}_{j+1}$ and $\underline{H}_j$ need to be stored. In order to develop a memory efficient procedure, Q_{j+1}, $\mathbf{R}_{j+1}$ and $\underline{H}_j$ should be computed without explicitly forming $\mathbf{V}_{j+1}$. The TOAR algorithm is developed to achieve this goal. The TOAR method

is based on two levels of orthogonalization and it will be explained in the following paragraphs.

Consider an initial unit vector $\mathbf{v}_1 \in \mathbb{C}^{2n}$ partitioned as in (2.50)

$$\mathbf{v}_1 = \begin{bmatrix} v_1^{(1)} \\ v_1^{(2)} \end{bmatrix}, \quad v_1^{(1)}, v_1^{(2)} \in \mathbb{C}^n, \tag{2.59}$$

and the QR decomposition

$$[v_1^{(1)} \quad v_1^{(2)}] = Q_1 [r_1^{(1)} \quad r_1^{(2)}], \tag{2.60}$$

with $Q_1 \in \mathbb{C}^{n\times 2}$, $r_1^{(1)}$, $r_1^{(2)} \in \mathbb{C}^2$.

It follows that, by considering the Arnoldi recurrence relation (2.58) and assuming that breakdown does not occur,

$$\text{span}\{Q_{j+1}\} = \text{span}\{Q_j, \hat{v}_j\}, \quad \text{with } \hat{v}_j = G_1 v_j^{(1)} + G_2 v_j^{(2)}. \tag{2.61}$$

Since $\hat{v}_j$ can be written as

$$\hat{v}_j = Q_j x_j + \alpha_j q_{j+1}. \tag{2.62}$$

where q_{j+1} is a unit orthogonal vector to Q_j, we have

$$Q_{j+1} = [Q_j \quad q_{j+1}]. \tag{2.63}$$

This implies that it is better to compute Q_{j+1} before $\mathbf{R}_{j+1}$. This process is called the first level of orthogonalization and it is summarized in Algorithm 2.9.

Algorithm 2.9 First level of orthogonalization for TOAR method

Input: The matrix Q_j with orthonormal columns such that $\text{span}\{Q_j\} = \text{span}\{V_j^{(1)}, V_j^{(2)}\}$, and $\hat{v}_j$ defined in (2.61).

Output: The matrix Q_{j+1}, x_j and α_j defined in (2.62).

1. Compute $x_j = Q_j^* \hat{v}_j$.
2. Compute $\tilde{q}_j = \hat{v}_j - Q_j x_j$.
3. Compute $\alpha_j = \|\tilde{q}_j\|_2$.
4. Reorthogonalize $\tilde{q}_j$ if necessary.

if $\alpha_j = 0$ **then**

5. Keep $Q_{j+1} = Q_j$.

else

6. Compute $q_{j+1} = \tilde{q}_j / \alpha_j$.
7. Extend $Q_{j+1} = [Q_j \quad q_{j+1}]$.

end if

In order to develop the rest of the TOAR method, the Arnoldi procedure is followed to compute $\mathbf{v}_{j+1}$ expressed in the compact form:

$$\mathbf{v}_{j+1} = \begin{bmatrix} v_{j+1}^{(1)} \\ v_{j+1}^{(2)} \end{bmatrix} = \begin{bmatrix} Q_{j+1} r_{j+1}^{(1)} \\ Q_{j+1} r_{j+1}^{(2)} \end{bmatrix}. \tag{2.64}$$

From now on, and w.l.o.g, we consider $\alpha_j \neq 0$ in Algorithm 2.9 (as it is considered in [71]). The first step in the Arnoldi method (Algorithm 2.1) is to compute the vector $\hat{\mathbf{v}}$ which can be computed as

$$\hat{\mathbf{v}} = \mathbf{G}_Q \mathbf{v}_j = \begin{bmatrix} 0 & I \\ G_1 & G_2 \end{bmatrix} \begin{bmatrix} v_j^{(1)} \\ v_j^{(2)} \end{bmatrix} = \begin{bmatrix} v_j^{(2)} \\ \hat{v}_j \end{bmatrix}, \tag{2.65}$$

and then, by (2.62) and the compact representation for $v_j^{(2)}$,

$$\hat{\mathbf{v}} = \begin{bmatrix} Q_j r_j^{(2)} \\ Q_j x_j + \alpha_j q_{j+1} \end{bmatrix}. \tag{2.66}$$

Then, the coefficients h_j can be computed as

$$h_j = \mathbf{V}_j^* \hat{\mathbf{v}} = (R_j^{(1)})^* r_j^{(2)} + (R_j^{(2)})^* x_j = \begin{bmatrix} R_j^{(1)} \\ R_j^{(2)} \end{bmatrix}^* \begin{bmatrix} r_j^{(2)} \\ x_j \end{bmatrix}. \tag{2.67}$$

Continuing with the Arnoldi method,

$$\begin{aligned} \tilde{\mathbf{v}} = \hat{\mathbf{v}} - \mathbf{V}_j h_j &= \begin{bmatrix} Q_j r_j^{(2)} \\ Q_j x_j + \alpha_j q_{j+1} \end{bmatrix} - \begin{bmatrix} Q_j R_j^{(1)} \\ Q_j R_j^{(2)} \end{bmatrix} h_j, \\ &= \begin{bmatrix} Q_{j+1} & 0 \\ 0 & Q_{j+1} \end{bmatrix} \begin{bmatrix} \tilde{s}^{(1)} \\ 0 \\ \tilde{s}^{(2)} \\ \alpha_j \end{bmatrix}, \end{aligned} \tag{2.68}$$

where

$$\tilde{s}^{(1)} = r_j^{(2)} - R_j^{(1)} h_j, \quad \tilde{s}^{(2)} = x_j - R_j^{(2)} h_j. \tag{2.69}$$

By (2.68), the blocks of $\mathbf{v}_{j+1}$ can be computed as

$$v_{j+1}^{(1)} = Q_{j+1} \begin{bmatrix} \tilde{s}^{(1)}/h_{j+1,j} \\ 0 \end{bmatrix}, \quad v_{j+1}^{(2)} = Q_{j+1} \begin{bmatrix} \tilde{s}^{(2)}/h_{j+1,j} \\ \alpha_j/h_{j+1,j} \end{bmatrix} \tag{2.70}$$

with

$$h_{j+1,j} = \|\tilde{v}\|_2 = \left\| \begin{bmatrix} \tilde{s}^{(1)} \\ 0 \\ \tilde{s}^{(2)} \\ \alpha_j \end{bmatrix} \right\|_2. \tag{2.71}$$

With this, we can form $R_{j+1}^{(1)}$ and $R_{j+1}^{(2)}$ as

$$R_{j+1}^{(1)} = \begin{bmatrix} R_j^{(1)} & \tilde{s}^{(1)}/h_{j+1,j} \\ 0 & 0 \end{bmatrix}, \quad R_{j+1}^{(2)} = \begin{bmatrix} R_j^{(2)} & \tilde{s}^{(2)}/h_{j+1,j} \\ 0 & \alpha_j/h_{j+1,j} \end{bmatrix} \tag{2.72}$$

that satisfies

$$\begin{bmatrix} V_{j+1}^{(1)} \\ V_{j+1}^{(2)} \end{bmatrix} = \begin{bmatrix} Q_{j+1}R_{j+1}^{(1)} \\ Q_{j+1}R_{j+1}^{(2)} \end{bmatrix} = \begin{bmatrix} Q_{j+1} & 0 \\ 0 & Q_{j+1} \end{bmatrix} \begin{bmatrix} R_{j+1}^{(1)} \\ R_{j+1}^{(2)} \end{bmatrix}. \tag{2.73}$$

Since $\mathbf{R}_j$ has orthonormal columns, the resulting h_j and $\tilde{\mathbf{s}} := \begin{bmatrix} \tilde{s}^{(1)} \\ \tilde{s}^{(2)} \end{bmatrix}$ will satisfy

$$\begin{bmatrix} r_j^{(2)} \\ x_j \end{bmatrix} = \begin{bmatrix} R_j^{(1)} \\ R_j^{(2)} \end{bmatrix} h_j + \begin{bmatrix} \tilde{s}^{(1)} \\ \tilde{s}^{(2)} \end{bmatrix}, \tag{2.74}$$

where $\tilde{\mathbf{s}} := \begin{bmatrix} \tilde{s}^{(1)} \\ \tilde{s}^{(2)} \end{bmatrix}$ is orthogonal to $\mathbf{R}_j$. This procedure is exactly the classic Gram-Schmidt algorithm without the normalization step. The process of getting $\tilde{\mathbf{s}}$ is the so-called second level of orthogonalization and it is summarized in Algorithm 2.10.

Algorithm 2.10 Second level of orthogonalization for TOAR method

Input: The matrix $\mathbf{R}_j$ defined in (2.57) and x_j obtained in step 1 of Algorithm 2.9.
Output: The vectors $\tilde{\mathbf{s}}$ and h_j.

1. Compute $h_j = \begin{bmatrix} R_j^{(1)} \\ R_j^{(2)} \end{bmatrix}^* \begin{bmatrix} r_j^{(2)} \\ x_j \end{bmatrix}$.
2. Compute $\begin{bmatrix} \tilde{s}^{(1)} \\ \tilde{s}^{(2)} \end{bmatrix} = \begin{bmatrix} r_j^{(2)} \\ x_j \end{bmatrix} - \begin{bmatrix} R_j^{(1)} \\ R_j^{(2)} \end{bmatrix} h_j$.
3. Reorthogonalize $\tilde{\mathbf{s}}$ if necessary.

Finally, the complete two-level orthogonal Arnoldi process (TOAR) for the QEP (2.28) is summarized in Algorithm 2.11.

Now we discuss the computation and memory costs of TOAR and compare them with the costs of Arnoldi applied to an arbitrary $2n \times 2n$ matrix. As occurs for previous methods, the computational costs of Algorithms 2.1 and 2.11 are the sum of the costs for computing $\hat{\mathbf{v}}$ and the orthogonalization steps. The major cost for both methods is the computation of $\hat{\mathbf{v}}$ which requires the solution of a linear system and for Arnoldi is a bigger cost than for TOAR if the structure of the matrix is not considered. If the structure of the $2n \times 2n$ matrix is considered then a linear system of size $n \times n$ need to be solved as for TOAR.

For Arnoldi, the orthogonalization is performed in steps 2-3-4-5 of Algorithm 2.1 which requires at iteration j, as we mentioned in previous section, $8nj + 6n$ flops. Note that we are disregarding the terms that do not contain n since $n \gg j$ for large scale problems. The orthogonalization process for TOAR is performed in steps 3-4-5-6 of Algorithm 2.11. At iteration j, the first level of orthogonalization performed in step 3 costs $4nj + 7n$ flops. Since the second level of orthogonalization

Algorithm 2.11 Two-Level Orthogonal Arnoldi method (TOAR) for the QEP

Input: An initial vector $\mathbf{v}_1 = [v_1^{(1)^*} \quad v_1^{(2)^*}]^* \in \mathbb{C}^{2n}$, with $\|\mathbf{v}_1\|_2 = 1$, and the matrix $\mathbf{G}_Q$ partitioned as in (2.49).

Output: The matrices Q_{m+1} and $\mathbf{R}_{m+1}$ defined in (2.57), and $\underline{H}_{m+1}$.

1. Generate $Q_1, \mathbf{R}_1$ by QR decomposition

$$[Q_1, [r_1^{(1)} \quad r_1^{(2)}]] = qr([v_1^{(1)} \quad v_1^{(2)}]).$$

for $j = 1, 2, \ldots, m$ **do**

2. Compute $\hat{v} = G_1 v_j^{(1)} + G_2 v_j^{(2)}$.
3. Run Algorithm 2.9 with inputs Q_j, $\hat{v}$ and outputs Q_{j+1}, x_j and α_j.
4. Run Algorithm 2.10 with inputs $\mathbf{R}_j$, x_j and outputs $\tilde{\mathbf{s}}$ and h_j.
5. Normalization

$$h_{j+1,j} = \left\| \begin{bmatrix} \tilde{\mathbf{s}} \\ \alpha_j \end{bmatrix} \right\|_2, \begin{bmatrix} s^{(1)} \\ s^{(2)} \end{bmatrix} = \frac{1}{h_{j+1,j}} \begin{bmatrix} \tilde{s}^{(1)} \\ \tilde{s}^{(2)} \end{bmatrix}.$$

6. Update $\mathbf{R}_{j+1}$ by

$$R_{j+1}^{(1)} = \begin{bmatrix} R_j^{(1)} & s^{(1)} \\ 0 & 0 \end{bmatrix}, \quad R_{j+1}^{(2)} = \begin{bmatrix} R_j^{(2)} & s^{(2)} \\ 0 & \beta_j \end{bmatrix}, \quad \text{with } \beta_j = \frac{\alpha_j}{h_{j+1,j}}.$$

8. Generate the new Krylov vector

$$v_{j+1}^{(1)} = Q_j s^{(1)}, \quad v_{j+1}^{(2)} = Q_{j+1} \begin{bmatrix} s^{(2)} \\ \beta_j \end{bmatrix}.$$

end for

	Arnoldi method	TOAR method
Orthogonalization cost	$(4j^2 + 10j)n$	$(2j^2 + 9j)n$
Memory cost	$2n(j + 1)$	$n(j + 2)$

Table 2.4: Asymptotic orthogonalization and memory costs for classical Arnoldi method and TOAR method after j iterations.

computed in step 4 involves matrices of dimension j, the computational step is negligible to the whole process, and the same is true for steps 5 and 6. Then, the total orthogonalization cost is carried out by the first level of orthogonalization. The orthogonalization costs for both methods are summarized in Table 2.4. A final important remark is that TOAR involves the overhead cost of constructing $v_{j+1}^{(1)}$ and $v_{j+1}^{(2)}$ in step 8 of Algorithm 2.11. This step costs $4nj + 4n$ flops at iteration j, which added to the orthogonalization cost of TOAR discussed below would give a cost of the same order of the orthogonalization cost of the Arnoldi method.

In terms of memory cost, TOAR results in a memory efficient algorithm. Since only $Q_{j+1} \in \mathbb{C}^{n\times(j+2)}$ and $\mathbf{R}_{j+1} \in \mathbb{C}^{2(j+2)\times(j+1)}$ need to be stored, the storage requirement is roughly $(j + 2)n$ floating point numbers, when $n \gg j$ as happens in large scale problems. Therefore, the compact representation of the Krylov vectors reduces the storage requirement by about a half (as the Q-Arnoldi method) with respect to classical Arnoldi applied to an arbitrary matrix of size $2n \times 2n$ (see Table 2.4).

The representation of the Krylov vectors for both the Q-Arnoldi method and the TOAR method can be used to solve PEPs, and the following section deals with this problem.

2.4 Krylov methods for polynomial eigenvalue problems

In this section, we present Krylov methods to solve PEPs. As we mentioned in Section 2.3, Q-Arnoldi and TOAR can be generalized for PEPs of any degree. Section 2.4.1 presents a generalization of Q-Arnoldi and this new method is called the P-Arnoldi method. Section 2.4.2 summarizes a generalization of TOAR, resulting in the TOAR method for PEPs. Both methods are developed for matrix polynomials expressed in the Chebyshev basis, which is more stable if the eigenvalues are located in or close to an interval $I \subset \mathbb{R}$ [92]. As occurs for Q-Arnoldi for QEPs, P-Arnoldi presents some instabilities, in particular for Ritz values not close to the interval I which makes it unsuitable for matrix polynomials of larger degree. On the other hand, TOAR for PEPs seems to be numerically stable even for large degrees. The numerical experiments presented in [63] show that both methods are more efficient

in terms of memory requirements and CPU time with respect to the Arnoldi method applied to an arbitrary matrix of the size of the associated linearization.

In Section 2.4.3 we introduce the CORK method for PEPs which is based on the rational Krylov method (see Section 2.2.2). The CORK method works with many other linearizations apart from the Frobenius form or its comrade generalization to the Chebyshev basis and it also follows the spirit of the two levels of orthogonalization.

Since the CORK method have been developed for several linearizations, included the one for matrix polynomials expressed in the Chebyshev basis, and it is based on the rational Krylov method, which allows to choose different shifts at each iteration, it is preferred in the literature for solving large-scale and sparse PEPs. However, keep in mind that all methods presented in Section 2.4 result in efficient memory saving algorithms.

It is usual to express a matrix polynomial in the monomial basis, this is

$$P(\lambda) = \lambda^d P_d + \cdots + \lambda P_1 + P_0, \quad P_i \in \mathbb{C}^{n\times n},\ i = 0, \ldots, d$$

and then, solve the PEP: $P(\lambda)x = 0$. This representation leads to severe numerical difficulties for high degrees, unless the eigenvalues are (nearly) distributed along a circle [92]. In the case when the required eigenvalues are on or close to an interval $I \subset \mathbb{R}$, a non-monomial representation is numerically much more suitable. If we consider w.l.o.g. $I = [-1, 1]$ then a numerically reliable representation [102] is given by

$$P_{Ch}(\lambda) = \tau_d(\lambda)P_d + \cdots + \tau_1(\lambda)P_1 + \tau_0(\lambda)P_0, \tag{2.75}$$

where $\tau_0, \tau_1, \ldots, \tau_d$ denote the Chebyshev polynomials of the first kind, defined by the recurrence

$$\begin{aligned} \tau_0(\lambda) &= 1, \qquad \tau_1(\lambda) = \lambda \\ \tau_{j+1}(\lambda) &= 2\lambda\tau_j(\lambda) - \tau_{j-1}(\lambda), \qquad j = 1, \ldots, d-1. \end{aligned} \tag{2.76}$$

A linearization for the polynomial eigenvalue problem

$$P_{Ch}(\lambda)x = 0 \tag{2.77}$$

is

$$\mathbf{A}_{Ch} = \begin{bmatrix} 0 & I & & & \\ I & 0 & I & & \\ & \ddots & \ddots & \ddots & \\ & & I & 0 & I \\ -P_0 & \cdots & -P_{d-3} & P_d - P_{d-2} & -P_{d-1} \end{bmatrix}, \quad \mathbf{B}_{Ch} = \begin{bmatrix} I & & & & \\ & 2I & & & \\ & & \ddots & & \\ & & & 2I & \\ & & & & 2P_d \end{bmatrix}, \tag{2.78}$$

which yields the generalized eigenvalue problem

$$\mathbf{A}_{Ch}\mathbf{y}_{Ch} = \lambda \mathbf{B}_{Ch}\mathbf{y}_{Ch}. \tag{2.79}$$

If (λ, x) is an eigenpair of the polynomial eigenvalue problem (2.77) then

$$\mathbf{y}_{Ch} = \begin{bmatrix} \tau_0(\lambda)x \\ \tau_1(\lambda)x \\ \vdots \\ \tau_{d-1}(\lambda)x \end{bmatrix}. \tag{2.80}$$

2.4.1 The P-Arnoldi method

In this section, the P-Arnoldi method for matrix polynomials in the Chebysehv basis is developed [63]. P-Arnoldi stands for the extension of the Q-Arnoldi method [75] presented in Section 2.3.2 to polynomials of higher degree. This method basically consists of applying the classical Arnoldi method to the matrix

$$\mathbf{G}_{Ch} = {\mathbf{B}_{Ch}}^{-1}\mathbf{A}_{Ch} = \frac{1}{2}\begin{bmatrix} 0 & 2I & & & \\ I & 0 & I & & \\ & \ddots & \ddots & \ddots & \\ & & I & 0 & I \\ -P_d^{-1}P_0 & \cdots & -P_d^{-1}P_{d-3} & I - P_d^{-1}P_{d-2} & -P_d^{-1}P_{d-1} \end{bmatrix} \tag{2.81}$$

implicitly, such that the Krylov vectors can be represented in a memory-efficient way. In order to develop a compact representation for the Krylov vectors, the partition

$$\mathbf{V}_{j+1} = [\mathbf{V}_j \quad \mathbf{v}_{j+1}] \begin{bmatrix} V_j^{(1)} & v_{j+1}^{(1)} \\ \vdots & \vdots \\ V_j^{(d)} & v_{j+1}^{(d)} \end{bmatrix} \tag{2.82}$$

is considered, where $V_j^{(i)} \in \mathbb{C}^{n\times j}$ and $v_{j+1}^{(i)} \in \mathbb{C}^n$ for $i = 1, \ldots, d$. The goal of the P-Arnoldi method is to perform all computations without explicit reference to the $V_j^{(2)}, \ldots, V_j^{(d)}$ blocks, so that only the $n \times j$ matrix $V_j^{(1)}$ and the $nd \times 1$ vector $\mathbf{v}_{j+1}$, which is updated at each iteration, need to be stored.

By using the structure of $\mathbf{G}_{Ch}$ and the Arnoldi decomposition (2.18) we obtain

$$\begin{aligned} V_j^{(2)} &= [V_j^{(1)} \quad v_{j+1}^{(1)}]\underline{H}_j, \\ V_j^{(i)} &= 2[V_j^{(i-1)} \quad v_{j+1}^{(i-1)}]\underline{H}_j - V_j^{(i-2)}, \quad i = 3, \ldots, d. \end{aligned} \tag{2.83}$$

Lemma 2.29 shows that we can reconstruct all $V_j^{(i)}$ blocks from $V_j^{(1)}$, $\mathbf{v}_{j+1}$, and $\underline{H}_j$.

Lemma 2.29. *[63, Lemma 1] Given an Arnoldi decomposition* (2.18) *for the matrix in* (2.81)*, with* $\mathbf{V}_{j+1}$ *partitioned as in* (2.82)*, the relation*

$$V_j^{(i)} = \check{V}_j^{(i)} W_j^{(i)} \tag{2.84}$$

holds for all $i = 2, \ldots, d$*, where* $\check{V}_j^{(i)} = [V_j^{(1)}, v_{j+1}^{(1)}, \ldots, v_{j+1}^{(i-1)}]$ *and* W_j *is defined by the recurrence*

$$W_j^{(1)} = I_j, \quad W_j^{(2)} = \underline{H}_j, \quad W_j^{(i)} = 2 \begin{bmatrix} W_j^{(i-1)} & 0 \\ 0 & 1 \end{bmatrix} \underline{H}_j - \begin{bmatrix} W_j^{(i-2)} \\ 0 \\ 0 \end{bmatrix}, i = 3, \ldots, d. \tag{2.85}$$

Since for the Arnoldi method in Algorithm 2.1 the operations that need to be performed involved either $\mathbf{V}_j$ or $\mathbf{V}_j^*$, Lemma 2.29 implies that these operations can be computed for the matrix (2.81) without storing $V_j^{(2)}, \ldots, V_j^{(d)}$ and only computing the matrices $W_j^{(i)}$. However, this computation yields an additional cost of order $\mathcal{O}(dj^3)$ which can be avoided as a consequence of Lemma 2.30.

Lemma 2.30. *[63, Lemma 2] Given an Arnoldi decomposition* (2.18) *for* $\mathbf{G}_{Ch}$*, with* $\mathbf{V}_{j+1}$ *partitioned as in* (2.82)*, the relation*

$$V_j^{(i)} = V_j^{(1)} \tau_{i-1}(H_j) + h_{j+1,j} v_{j+1}^{(1)} e_j^* \tilde{\tau}_{i-1}(H_j) + 2h_{j+1,j} \sum_{k=2}^{i-1} v_{j+1}^{(k)} e_j^* \tilde{\tau}_{i-k}(H_j), \tag{2.86}$$

holds for all $i = 2, \ldots, d$*, where* $\tilde{\tau}_0$*,* $\ldots$*,* $\tilde{\tau}_{d-1}$ *denote the Chebyshev polynomials of second kind, defined by the recursion*

$$\tilde{\tau}_0(\lambda) = 0, \quad \tilde{\tau}_1(\lambda) = 1, \quad \tilde{\tau}_{j+1}(\lambda) = 2\lambda \tilde{\tau}_j(\lambda) - \tilde{\tau}_{j-1}(\lambda), \tag{2.87}$$

and H_j *is the Hessenberg matrix obtained by removing the last row of* $\underline{H}_j$*.*

Step 1 in Arnoldi method (see Algorithm 2.1) is a matrix-vector multiplication, which can be performed as in Algorithm 2.12.

Step 2 in Algorithm 2.1 consists in computing the coefficients

$$h_j = \begin{bmatrix} V_j^{(1)} \\ \vdots \\ V_j^{(d)} \end{bmatrix}^* \begin{bmatrix} \hat{v}^{(1)} \\ \vdots \\ \hat{v}^{(d)} \end{bmatrix} = \sum_{i=1}^{d} (V_j^{(i)})^* \hat{v}^{(i)}. \tag{2.88}$$

By using Lemma 2.30 we have

$$h_j = \sum_{i=1}^{d} \tau_{i-1}(H_j^*) {V_j^{(1)}}^* \hat{v}^{(i)} + \overline{h_{j+1,j}} \sum_{i=1}^{d} {(v_{j+1}^{(1)})}^* \hat{v}^{(i)} \tilde{\tau}_{i-1}(H_j^*) e_j$$

Algorithm 2.12 P-Arnoldi: Matrix-vector product with $\mathbf{G}_{Ch}$

Input: Matrices $P_0, P_1, \ldots, P_d \in \mathbb{C}^{n\times n}$ that define $\mathbf{G}_{Ch}$ as in (2.81) and a vector $\mathbf{v}$ partitioned as in (2.82).
Output: The vector $\mathbf{p} = \mathbf{G}_{Ch}\mathbf{v}$ partitioned as in (2.82).
1. Set $p^{(1)} = v^{(2)}$.
for $j = 2, \ldots, d-1$ **do**
2. Compute $p^{(j)} = \frac{1}{2}\left(v^{(j-1)} + v^{(j+1)}\right)$.
end for
3. Compute $p^{(d)} = \frac{1}{2}v^{(d-1)} - \frac{1}{2}P_d^{-1}\sum_{i=1}^{d} P_{i-1}v^{(i)}$.

$$+ \quad 2\overline{h_{j+1,j}}\sum_{i=1}^{d}\sum_{k=2}^{i-1}(v_{j+1}^{(k)})^*\hat{v}^{(i)}\tilde{\tau}_{i-k}(H_j^*)e_j. \tag{2.89}$$

The first sum in (2.89) can be computed by using the Clenshaw's algorithm [23] for evaluating polynomials in the Chebyshev basis and it requires the computation of d matrix-vector products ${V_j^{(1)}}^*\hat{u}^{(i)}$, which can be organized as a single matrix-matrix product ${V_j^{(1)}}^*[\hat{u}^{(1)}, \ldots, \hat{u}^{(d)}]$. The last two sums in (2.89) amount to a matrix-matrix multiplication with the $j \times (d-1)$ matrix

$$T = \overline{h_{j+1,j}}[\tilde{\tau}_{d-1}(H_j^*)e_j, \ldots, \tilde{\tau}_2(H_j^*)e_j, \tilde{\tau}_1(H_j^*)e_j],$$

which can be computed recursively from right to left by using (2.87). An efficient algorithm to compute the coefficients h_j is presented in [63, Algorithm 3] which uses (2.89) and the recurrence relation (2.87).

Finally, step 3 in Algorithm 2.1 requires the matrix-vector multiplication $(V_j^{(i)}h_j)$ which can be also computed by using Lemma 2.30, obtaining

$$V_j^{(i)}h_j = V_j^{(1)}\tau_{i-1}(H_j)h_j + h_{j+1,j}v_{j+1}^{(1)}e_j^*\tilde{\tau}_{i-1}(H_j)h_j + 2h_{j+1,j}\sum_{k=2}^{i-1} v_{j+1}^{(k)}e_j^*\tilde{\tau}_{i-k}(H_j)h_j. \tag{2.90}$$

In [63, Algorithm 4] an efficient algorithm is presented to compute $V_j^{(i)}h_j$. Finally, the P-Arnoldi method is summarized in Algorithm 2.13.

Now, we compare Arnoldi and P-Arnoldi in terms of orthogonalization and memory costs. First, as we mentioned in previous section, it occurs that $n \gg j$ and then we do not consider the computations that do not involve matrices or vectors of size n. Also, in practice, $d < j$, where d represents the degree of the matrix polynomial. Second, the most expensive step for both methods is given by the computation of $\hat{\mathbf{v}}$. If an unstructured solver is used for step 1 in the Arnoldi method instead of Algorithm 2.12, then the computational cost of Arnoldi is much larger than the cost of P-Arnoldi, which solves a linear system of size $n \times n$ instead of size $nd \times nd$.

Algorithm 2.13 The P-Arnoldi method

Input: The coefficient matrices $P_0, P_1, \ldots, P_d$ defining the matrix $\mathbf{G}_{Ch}$ in (2.81) and an initial vector $\mathbf{v}_1$ partitioned as in (2.82).

Output: Matrices $V_m^{(1)}$, $\underline{H}_m$ and vectors $v_{m+1}^{(1)}, \ldots, v_{m+1}^{(d)}$ that define implicitly the vectors of an orthonormal basis of $K_m(\mathbf{G}_{Ch}, \mathbf{v}_1)$ by the recurrence relation (2.83).

for $j = 1, \ldots, m$ **do**

1. Compute $\hat{\mathbf{v}} = \mathbf{G}_{Ch}\mathbf{v}_j$ via Algorithm 2.12.
2. Compute h_j via [63, Algorithm 3].
3. Compute the matrix-vector products $V_j^{(i)} h_j$ via [63, Algorithm 4].
4. Compute $\tilde{\mathbf{v}} = \hat{\mathbf{v}} - \mathbf{V}_j h_j$.
5. Compute $h_{j+1,j} = \|\tilde{\mathbf{v}}\|_2$.

if $h_{j+1,j} = 0$ **then**
stop the procedure
end if

6. Compute the next vector $\mathbf{v}_{j+1} = \dfrac{1}{h_{j+1,j}}\tilde{\mathbf{v}}$.
7. Update $V_{j+1}^{(1)} = [V_j^{(1)}\ v_{j+1}^{(1)}]$.

end for

The P-Arnoldi method results in a memory-efficient procedure that reduces the storage costs from dnj to $(jn+dn)$ numbers with respect to Arnoldi applied directly on $\mathbf{G}_{Ch}$. These memory costs are shown in Table 2.5.

Now we compare the orthogonalization costs for both methods at iteration j, which are steps 2-3-4-5 in Algorithm 2.1 for Arnoldi and steps 2-3-4-5-6 in Algorithm 2.13 for P-Arnoldi. For Arnoldi applied directly on the $nd \times nd$ matrix $\mathbf{G}_{Ch}$ in Algorithm 2.1 we have the well-known cost of $\mathcal{O}(njd)$ flops for the orthogonalization steps. P-Arnoldi requires in Algorithm 2.13: $\mathcal{O}(nd(d+j))$ flops for step 2, $\mathcal{O}(ndj)$ flops for step 3 and $\mathcal{O}(nd)$ flops for steps 4, 5 and 6. Therefore, at iteration j, the orthogonalization cost of P-Arnoldi is $\mathcal{O}(nd(d+j))$ flops. Orthogonalization costs are summarized in Table 2.5. Note that the orthogonalization cost of P-Arnoldi is slightly more expensive than the cost of Arnoldi, however, since in practice $d < j \ll n$ [14, 63], this difference of cost is usually small.

	Arnoldi method	P-Arnoldi method
Orthogonalization cost	$\mathcal{O}(ndj^2)$	$\mathcal{O}(ndj^2 + nd^2j)$
Memory cost	ndj	$(d+j)n$

Table 2.5: Asymptotic orthogonalization and memory costs for classical Arnoldi method and P-Arnoldi method after j iterations.

Techniques like implicit restarting and locking can be implemented and we refer to [63] for more details.

Although P-Arnoldi can be formulated in an elegant way, its instability for Ritz values not close to $[-1, 1]$ makes it unsuitable for solving PEPs with matrix polynomials of large degree. In the next section, we discuss the TOAR method for PEPs which appears to be stable even for comparably large degrees.

2.4.2 TOAR method for polynomial eigenvalue problems

As we mentioned before, it is common that the eigenvalues of interest are close to a given target θ, which is typically in the interior of the spectrum. The shift-and-invert technique addresses this problem by applying the Arnoldi method to the matrix $(\mathbf{G}_{Ch} - \theta I)^{-1}$, where $\mathbf{G}_{Ch}$ is the matrix defined in (2.81). In this section, we focus on this problem for the particular case $\theta = 0$. The interested reader can find the general case $\theta \neq 0$ in [63].

If we consider j steps of the Arnoldi method applied to $\mathbf{G}_{Ch}^{-1}$, we obtain the Arnoldi recurrence relation

$$(\mathbf{G}_{Ch})^{-1} \mathbf{V}_j = \mathbf{V}_j H_j + h_{j+1,j} \mathbf{v}_{j+1} e_j^*, \tag{2.91}$$

which is equivalent to

$$\mathbf{G}_{Ch} \mathbf{V}_j = \mathbf{V}_j H_j^{-1} + h_{j+1,j} \breve{\mathbf{v}}_{j+1} \breve{e}_j^*, \tag{2.92}$$

where $\breve{\mathbf{v}}_{j+1} = -\mathbf{G}_{Ch} \mathbf{v}_{j+1}$ and $\breve{e}_j = H_j^{-*} e_j$. The equation (2.92) has a similar form than the Arnoldi recurrence relation (2.18), therefore, an extension of Lemma 2.30 can be developed. Thus, the relation (2.92) makes it possible to extend the P-Arnoldi method to this situation. However, some difficulties appear due to the fact that the norms of $\tau_j(H_j^{-1})$ grow quickly if H_j^{-1} has eigenvalues too far away from the interval $[-1, 1]$ for larger degree d [63].

This instability is due to the presence of $\tau_j(H_j^{-1})$ in the representation of the orthonormal Arnoldi basis $\mathbf{V}_j$ and motivates the development of a version of TOAR for matrix polynomials with arbitrary degrees expressed in the Chebyshev basis [63].

For TOAR, we use the following representation, that is possible as a consequence of a version of (2.83) adapted to (2.92)

$$\mathbf{V}_j = \begin{bmatrix} V_j^{(1)} \\ V_j^{(2)} \\ \vdots \\ V_j^{(d)} \end{bmatrix} = \begin{bmatrix} Q_j R_j^{(1)} \\ Q_j R_j^{(2)} \\ \vdots \\ Q_j R_j^{(d)} \end{bmatrix} = \begin{bmatrix} Q_j & & & \\ & Q_j & & \\ & & \ddots & \\ & & & Q_j \end{bmatrix} \begin{bmatrix} R_j^{(1)} \\ R_j^{(2)} \\ \vdots \\ R_j^{(d)} \end{bmatrix} = (I_d \otimes Q_j) \mathbf{R}_j, \tag{2.93}$$

where $V_j^{(i)} \in \mathbb{C}^{n\times j}$ for $i = 1, 2, \ldots, d$, $Q_j \in \mathbb{C}^{n\times(d+j)}$ is a matrix with orthonormal columns and

$$\mathbf{R}_j := \begin{bmatrix} R_j^{(1)} \\ R_j^{(2)} \\ \vdots \\ R_j^{(d)} \end{bmatrix}, \tag{2.94}$$

for some matrices $R_j^{(i)} \in \mathbb{C}^{(d+j)\times j}$. Note that $\mathbf{R}_j^*\mathbf{R}_j = I_j$ and that the number of columns of Q_j could be less than $d + j$ in the course of the method. However, in [63] this case is neglected and this possibility is considered in both CORK and R-CORK methods. Observe that $\mathbf{V}_j$ has ndj entries that can be represented by $(n + dj)(d + j)$ parameters. Thus, considering that the dimension j of the Krylov subspace is much smaller than the dimension n of the large-scale problem, and that the degree d of PEPs in applications is a low number [14, 63], we obtain that $j + d < jd \ll n$, obtaining a compact representation for the Krylov vectors.

As for the TOAR method for QEPs, the information needed for the expansion of the basis Q_j is contained in the vector $\hat{v}^{(1)}$. Theorem 2.31 summarizes this result.

Theorem 2.31. *[63, Theorem 4] Suppose that $\mathbf{V}_{j+1}$ satisfies the Arnoldi decomposition (2.18) for $\mathbf{G}_{Ch}^{-1}$ and is partitioned as in (2.82). Let Q_j be an orthonormal basis of $\operatorname{span}\{V_j^{(1)}, V_j^{(2)}, \ldots, V_j^{(d)}\}$, this is $\operatorname{span}\{Q_j\} = \operatorname{span}\{V_j^{(1)}, V_j^{(2)}, \ldots, V_j^{(d)}\}$. Then,*

$$\operatorname{span}\{Q_{j+1}\} = \operatorname{span}\{V_j^{(1)}, v_{j+1}^{(1)}, V_j^{(2)}, v_{j+1}^{(2)}, \ldots, V_j^{(d)}, v_{j+1}^{(d)}\} = \operatorname{span}\{Q_j, \hat{v}^{(1)}\}. \tag{2.95}$$

Next we discuss the implementation of the Arnoldi method applied to $\mathbf{G}_{Ch}^{-1}$ defined in (2.81) by using the representation (2.93).

First, in step 1 of Algorithm 2.3, it is necessary to compute the vector $\hat{\mathbf{v}} = \mathbf{G}_{Ch}^{-1}\mathbf{v}_j$, which can be computed by solving a linear system involving $\mathbf{G}_{Ch}$. Consider the vectors $\hat{\mathbf{v}}$ and $\mathbf{v}_j$ partitioned as in (2.82). Then, for the even superscripts, we obtain the recurrence:

$$\hat{v}^{(2)} = Q_j r_j^{(1)}, \quad \hat{v}^{(2i)} = 2Q_j r_j^{(2i-1)} - \hat{v}^{(2(i-1))}, \quad i = 2, 3, \ldots \tag{2.96}$$

where $r_j^{(i)} \in \mathbb{C}^{d+j}$ denotes the last column of $R_j^{(i)}$ for $i = 1, 2, \ldots, d$, while for odd superscripts

$$\hat{v}^{(2i+1)} = y_{2i} + (-1)^i \hat{v}^{(1)}, \quad i = 1, 2, \ldots \tag{2.97}$$

with

$$y_2 = 2Q_j r_j^{(2)}, \quad y_{2i} = 2Q_j r_j^{(2i)} - y_{2(i-1)}, \quad i = 2, 3, \ldots \tag{2.98}$$

and $\hat{v}^{(1)}$ is obtained by solving the linear system

$$(-P_0 + P_2 - P_4 + \ldots)\hat{v}^{(1)} = (P_1\hat{v}^{(2)} + P_3\hat{v}^{(4)} + P_5\hat{v}^{(6)} + \ldots) + (P_2 y_2 + P_4 y_4 + P_6 y_6 + \ldots). \tag{2.99}$$

Therefore, the LU decomposition of $N = -P_0 + P_2 - P_4 + \ldots$, can be precomputed once and for all and to use it in each iteration to solve (2.99), although other methods can be also used to solve (2.99). Since only the first block $\hat{v}^{(1)}$ of $\hat{\mathbf{v}}$ is needed for the following steps in the Arnoldi method, it is necessary to save it. The details for the computation of $\hat{v}^{(1)}$ are given in Algorithm 2.14.

Algorithm 2.14 TOAR: Matrix-vector product with $\mathbf{G}_{Ch}^{-1}$

Input: The coefficients $P_0, P_1, \ldots, P_d$ defining the matrix $\mathbf{G}_{Ch}$ in (2.81), Q_j obtained by the compact representation of $\mathbf{V}_j$, and vector $\mathbf{r}_j = [(r_j^{(1)})^T, \ldots, (r_j^{(d)})^T]^T$, which is the last column of $\mathbf{R}_j$.

Output: First component $\hat{v}^{(1)}$ of $\hat{\mathbf{v}} = (\mathbf{G}_{Ch})^{-1}\mathbf{v}_j$ where $\mathbf{v}_j = (I_d \otimes Q_j)\mathbf{r}_j$.

1. Set $Y = [r_j^{(1)}, 2r_j^{(2)}]$.

for $i = 3, \ldots, d-1$ **do**

2. Update $Y = [Y, 2r_j^{(i)} - y_{i-2}]$.

end for

3. Compute $Z = Q_j Y$.

4. Compute $\bar{w} = -P_d z_{d-1} + \sum_{i=1}^{d-1} P_i z_i$.

if d is even **then**

Update $\bar{w} = \bar{w} + P_d(2P_d Q_j r_j^{(d)} - z_{d-2})$.

end if

5. Solve the linear system $N\hat{v}^{(1)} = \bar{w}$ with $N = \sum_{i=0}^{\lfloor (d-2)/2 \rfloor} (-1)^{(i+1)} P_{2i}$.

The next step in the shift-and-invert Arnoldi method consists of computing the coefficients $h_j = \mathbf{V}_j^* \hat{\mathbf{v}} = \mathbf{R}_j^*(I_d \otimes Q_j^*)\hat{\mathbf{v}}$. Lemma 2.32 shows analogous recurrences for the block components $\mathbf{p} = (I_d \otimes Q_j^*)\hat{\mathbf{v}}$ by using the presence of Q_j in the recurrences (2.96), (2.97) and (2.98).

Lemma 2.32. *[63, Lemma 3] For given $r_j^{(1)}, \ldots, r_j^{(d)}$ and $p^{(1)} = Q_j^* \hat{v}^{(1)} \in \mathbb{C}^{d+j}$, let the vectors $p^{(i)}$, $i = 2, \ldots, d$ be defined by the recurrences*

$$p^{(2)} = r_j^{(2)}, \quad p^{(2i)} = 2r_j^{(2i-1)} - p^{(2(i-1))}, \qquad i = 2, 3, \ldots, \tag{2.100}$$

$$p^{(2i+1)} = x_{2i} + (-1)^i p^{(1)}, \qquad i = 1, 2, \ldots, \tag{2.101}$$

$$x_2 = 2r_j^{(2)}, \quad x_{2i} = 2r_j^{(2i)} - x_{2(i-1)}, \qquad i = 2, 3, \ldots. \tag{2.102}$$

Then, the relations

1. *$\hat{v}^{(i)} = Q_j p^{(i)}$ and $y_i = Q_j x_i$ hold for all even i;*

2. *$\hat{v}^{(i)} = Q_j p^{(i)} + (-1)^{(1+i)/2}(I - Q_j Q_j^*)\hat{v}^{(1)}$ holds for all odd i:*

where $\hat{v}^{(i)}$ and y_i are defined by the recurrences (2.96), (2.97) *and* (2.98). *In particular, $p^{(i)} = Q^* \hat{v}^{(i)}$ for $i = 2, \ldots, d$.*

After $\hat{v}^{(1)}$ has been computed by Algorithm 2.14, the recurrences (2.100)-(2.102) of Lemma 2.32 allows to develop a cheap way to compute the coefficients h_j by computing the product:

$$h_j = \mathbf{R}_j^* \begin{bmatrix} Q_j & & \\ & \ddots & \\ & & Q_j \end{bmatrix}^* \begin{bmatrix} \hat{v}^{(1)} \\ \vdots \\ \hat{v}^{(d)} \end{bmatrix} = \mathbf{R}_j^* \begin{bmatrix} p^{(1)} \\ \vdots \\ p^{(d)} \end{bmatrix}. \tag{2.103}$$

From Theorem 2.31, we see that Q_j can be expanded with a new vector q_{j+1} computed as

$$\alpha q_{j+1} = \hat{v}^{(1)} - Q_j p^{(1)}, \quad \text{where } p^{(1)} = Q_j^* \hat{v}^{(1)}, \quad \text{and } \alpha = \|\hat{v}^{(1)} - Q_j p^{(1)}\|_2. \tag{2.104}$$

Step 3 in Algorithm 2.3 consists in computing the vector $\tilde{\mathbf{v}}$:

$$\tilde{\mathbf{v}} = \hat{\mathbf{v}} - \mathbf{V}_j h_j = \begin{bmatrix} \hat{v}^{(1)} \\ \vdots \\ \hat{v}^{(d)} \end{bmatrix} - \begin{bmatrix} Q_j & & \\ & \ddots & \\ & & Q_j \end{bmatrix} \begin{bmatrix} R_j^{(1)} \\ \vdots \\ R_j^{(d)} \end{bmatrix} h_j. \tag{2.105}$$

In order to compute this vector efficiently, we first compute the vector $\mathbf{s}$ obtained from the Gram-Schmidt orthogonalization of $\mathbf{p} = (I_d \otimes Q_j^*)\hat{\mathbf{v}}$ against $\mathbf{R}_j$:

$$\mathbf{s} = \begin{bmatrix} s^{(1)} \\ \vdots \\ s^{(d)} \end{bmatrix} = \begin{bmatrix} p^{(1)} \\ \vdots \\ p^{(d)} \end{bmatrix} - \begin{bmatrix} R_j^{(1)} \\ \vdots \\ R_j^{(d)} \end{bmatrix} h_j. \tag{2.106}$$

By using the relations of Lemma 2.32, the blocks with odd superscripts of $\tilde{\mathbf{v}}$ in (2.105) can be represented by

$$\tilde{v}^{(2i-1)} = [Q_j \quad q_{j+1}] \begin{bmatrix} s^{(2i-1)} \\ (-1)^{(1+i)/2}\alpha \end{bmatrix}. \tag{2.107}$$

while the blocks with even superscripts satisfy

$$\tilde{v}^{(2i)} = Q_j s^{(2i)}, \quad \text{for } i = 1, 2, \ldots \tag{2.108}$$

Therefore, $\mathbf{R}_j$ can be expanded by using the vector

$$\tilde{\mathbf{r}} = [(s^{(1)})^*, \alpha, (s^{(2)})^*, 0, (s^{(3)})^*, -\alpha, (s^{(4)})^*, 0, \ldots,]^* \tag{2.109}$$

after normalization. Note that, in particular, $\|\tilde{\mathbf{v}}\|_2 = \|\tilde{\mathbf{r}}\|_2 = h_{j+1,j}$. The TOAR method for PEPs of degree d represented in the Chebyshev basis is summarized in Algorithm 2.15.

Algorithm 2.15 Two-level orthogonalization Arnoldi (TOAR) method for $\mathbf{G}_{Ch}^{-1}$.

Input: Coefficient matrices $P_0, P_1, \ldots, P_d$ defining $\mathbf{G}_{Ch}$ in (2.81) and a starting vector $\mathbf{v}_1$.

Output: Matrices Q_{m+1}, $\mathbf{R}_{m+1}$ such that the matrix $\mathbf{V}_{m+1}$ defined in (2.93) is an orthonormal basis for $K_{m+1}(\mathbf{G}_{Ch}^{-1}, \mathbf{v}_1, 0)$.

1. Normalize $\mathbf{v}_1 = \dfrac{\mathbf{v}_1}{\|\mathbf{v}_1\|_2}$.
2. Compute a QR factorization

$$[v_1^{(1)}, v_1^{(2)}, \ldots, v_1^{(d)}] = Q_1 R_1, \quad \text{with } R_1 = [r_1^{(1)}, r_1^{(2)}, \ldots, r_1^{(d)}] \in \mathbb{R}^{d \times d}.$$

3. Set $\mathbf{r}_1 = [(r_1^{(1)})^*, (r_1^{(2)})^*, \ldots, (r_1^{(d)})^*]^*$.

for $j = 1, 2, \ldots, m$ **do**

4. Compute $\hat{v}^{(1)}$ by using Algorithm 2.14.
5. Compute $p^{(1)} = Q_j^* \hat{v}^{(1)}$ and $\tilde{q} = \hat{v}^{(1)} - Q_j p^{(1)}$.
6. Normalize $\alpha = \|\tilde{q}\|_2$ and compute $q_{j+1} = \tilde{q}/\alpha$.
7. Expand $Q_{j+1} = [Q_j, q_{j+1}]$.
8. Compute $p^{(i)}$ for $i = 2, \ldots, d$ using the recurrences (2.100)-(2.102) and set $\mathbf{p} = [(p^{(1)})^*, (p^{(2)})^*, \ldots, (p^{(d)})^*]^*$.
9. Compute $h_j = \mathbf{R}_j^* \mathbf{p}$ and $\mathbf{s} = \mathbf{p} - \mathbf{R}_j h_j$.
10. Update $R_j^{(i)} = \begin{bmatrix} R_j^{(i)} \\ 0_{j \times 1} \end{bmatrix}$.
11. Set $\tilde{\mathbf{r}} = [(s^{(1)})^*, \alpha, (s^{(2)})^*, 0, (s^{(3)})^*, -\alpha, (s^{(4)})^*, 0, \ldots]^*$.
12. Normalize $\mathbf{r}_{j+1} = \tilde{\mathbf{r}}/h_{j+1,j}$ with $h_{j+1,j} = \|\tilde{\mathbf{r}}\|_2$, and expand $\mathbf{R}_{j+1} = [\mathbf{R}_j, \mathbf{r}_{j+1}]$.

end for

The first and second level of orthogonalization are presented in steps 5 and 9 of Algorithm 2.15, respectively. If it is necessary, reorthogonalization can be implemented.

Now we analyze orthogonalization and memory costs. As for the TOAR method for QEPs, the main orthogonalization cost is concentrated in the first level, which involves only the matrix Q_j, while the cost for the second level is negligible when $n \gg m$. The costs of steps 5-6 in Algorithm 2.15 at iteration j are $\mathcal{O}(n(d+j))$ flops, whereas the orthogonalization cost for Arnoldi is $\mathcal{O}(ndj)$ flops. As occurs for TOAR for QEPs, the TOAR method for PEPs requires the construction of $\mathbf{v}_j$ to perform the shift-and-invert step, and if we add this extra cost to the orthogonalization cost of TOAR for PEPs we would obtain a cost of the same order of the orthogonalization cost of the Arnoldi method. The orthogonalization cost for each method is presented in Table 2.6.

However, the significant reduction of storage requirements, compared to the classic Arnoldi method applied to the linearization, makes the TOAR method a suitable and efficient method since the generated basis consists of vectors of length n instead of the vectors of full length nd, where d represents the degree of the polynomial. Memory costs for both methods are summarized in Table 2.6.

	Arnoldi method	TOAR method for PEPs
Orthogonalization cost	$\mathcal{O}(ndj^2)$	$\mathcal{O}(n(d+j)j)$
Memory cost	ndj	$n(d+j)$

Table 2.6: Asymptotic orthogonalization and memory costs for the Arnoldi method and TOAR for PEPs after j iterations.

2.4.3 The CORK method for polynomial eigenvalue problems

Van Beeumen, Meerbergen, and Michiels in [104] proposed a method based on a compact rational Krylov decomposition, extending the two levels of orthogonalization idea of TOAR from the quadratic eigenvalue problem [100, 71] to arbitrary degree polynomial eigenvalue problems and to other nonlinear eigenvalue problems (NLEPs) that can be linearized similarly to PEPs, including many other linearizations apart from the Frobenius one used in [100, 71], and using the rational Krylov method instead of the Arnoldi method. This method was baptized as CORK in [104] and for simplicity we described it particularized to PEPs of degree d. The key idea in [104] is to apply the rational Krylov method in Algorithm 2.4 to a structured linearization pencil of a matrix polynomial $P(\lambda)$ of degree d (recall Definition 2.13) taking into account that the special structure of these pencils imposes a special structure on the bases of the corresponding rational Krylov subspaces. By using this structure, the authors of [104] reduced both the memory cost and the orthogonali-

zation cost of the classical rational Krylov method applied to an arbitrary pencil of the same size. Considering the matrices $\mathbf{A}$ and $\mathbf{B}$ in (2.6) and the rational Krylov recurrence relation (2.25) for $\mathbf{A}$ and $\mathbf{B}$, the authors of [104] partitioned conformably the matrix $\mathbf{U}_{j+1}$, that stores the first $j+1$ orthonormal "rational" Krylov vectors, as follows

$$\mathbf{U}_{j+1} = [\mathbf{U}_j \quad \mathbf{u}_{j+1}] = \begin{bmatrix} U_j^{(1)} & u_{j+1}^{(1)} \\ U_j^{(2)} & u_{j+1}^{(2)} \\ \vdots & \vdots \\ U_j^{(d)} & u_{j+1}^{(d)} \end{bmatrix},$$

and then, they constructed a matrix $Q_j \in \mathbb{C}^{n \times r_j}$ with orthonormal columns such that

$$\operatorname{span}\{Q_j\} = \operatorname{span}\{U_j^{(1)}, U_j^{(2)}, \ldots, U_j^{(d)}\} \tag{2.110}$$

and $\operatorname{rank}(Q_j) = r_j$. By using the matrix Q_j, the blocks $U_j^{(i)}$ for $i = 1, 2, \ldots, d$ can be represented as follows

$$U_j^{(i)} = Q_j R_j^{(i)}, \quad i = 1, 2, \ldots, d,$$

for some matrices $R_j^{(i)} \in \mathbb{C}^{r_j \times j}$. Then,

$$\mathbf{U}_j = \begin{bmatrix} Q_j R_j^{(1)} \\ Q_j R_j^{(2)} \\ \vdots \\ Q_j R_j^{(d)} \end{bmatrix} = \begin{bmatrix} Q_j & & & \\ & Q_j & & \\ & & \ddots & \\ & & & Q_j \end{bmatrix} \begin{bmatrix} R_j^{(1)} \\ R_j^{(2)} \\ \vdots \\ R_j^{(d)} \end{bmatrix} = (I_d \otimes Q_j)\mathbf{R}_j, \tag{2.111}$$

where

$$\mathbf{R}_j := \begin{bmatrix} R_j^{(1)} \\ R_j^{(2)} \\ \vdots \\ R_j^{(d)} \end{bmatrix}.$$

By using this representation, the rational Krylov recurrence relation (2.25) can be written as follows [104, eq. (4.3)]

$$\mathbf{A}(I_d \otimes Q_{j+1})\mathbf{R}_{j+1}\underline{H}_j = \mathbf{B}(I_d \otimes Q_{j+1})\mathbf{R}_{j+1}\underline{K}_j. \tag{2.112}$$

Observe that $\mathbf{U}_j$ has ndj entries while the representation in (2.111) involves $(n+jd)r_j$ parameters. Therefore, taking into account that in the solution of large-scale PEPs the dimension j of the rational Krylov subspaces is much smaller than the dimension n of the problem and that the degree d of applied PEPs is a low number (for sure smaller than 30, see [63], and often much smaller than 30 [14]), we get that $jd \ll n$

and that the representation (2.111) of $\mathbf{U}_j$ stores approximately nr_j numbers. The fundamental reason why the representation of $\mathbf{U}_j$ in (2.111) is of interest and is indeed compact is because r_j is considerably much smaller than jd for the matrices $\mathbf{A}$ and $\mathbf{B}$ in (2.6). More precisely, the following result is proved in [104].

Theorem 2.33. *[104, Theorems 4.4 and 4.5] Let Q_j be defined as in* (2.110)*. Then*

$$span\{Q_{j+1}\} = span\{Q_j, u_{j+1}^{(p)}\}, \tag{2.113}$$

where $u_{j+1}^{(p)}$ represents the block of the vector $\mathbf{u}_{j+1}$ in a certain p-th position determined in [104]. Also,

$$r_j < j + d. \tag{2.114}$$

Note that Theorem 2.33 shows that Q_j can be expanded to Q_{j+1} by orthogonalizing only one vector of size n at each iteration. Also, $\mathbf{R}_{j+1}$ can be expanded in an easy way, if $u_{j+1}^{(p)} \notin \text{span}\{Q_j\}$ then the blocks $R_{j+1}^{(i)}$, $i = 1, \ldots, d$, can be written as

$$R_{j+1}^{(i)} = \left[\begin{array}{c|c} R_j^{(i)} & r_{j+1}^{(i)} \\ 0_{1\times j} & \end{array}\right], \quad i = 1, \ldots, d,$$

and, if $u_{j+1}^{(p)} \in \text{span}\{Q_j\}$, then $R_{j+1}^{(i)} = \begin{bmatrix} R_j^{(i)} & r_{j+1}^{(i)} \end{bmatrix}$, $i = 1, \ldots, d$. Based on these ideas, the authors of [104] developed CORK, splitting the method into two levels of orthogonalization: the first level is to expand Q_j into Q_{j+1} and the second level is to expand $\mathbf{R}_j$ into $\mathbf{R}_{j+1}$. We can see a basic pseudocode for the CORK method in Algorithm 2.16, whose complete explanation can be found in [104]. For simplicity, we assume that breakdown does not occur in Algorithm 2.16, i.e., $h_{j+1,j} \neq 0$ for all j. Note that the continuation vector z_j in step 2 of Algorithm 2.4 is implicitly considered in step 2 of Algorithm 2.16, where $\hat{u}^{(p)}$ is the p-th block of $\hat{\mathbf{u}}$ obtained by solving the linear system $(\mathbf{A} - \theta_j\mathbf{B})\hat{\mathbf{u}} = \mathbf{B}(I_d \otimes Q_j)\mathbf{R}_j z_j$ via the ULP decomposition presented in Theorem 2.15.

From the discussion above, it is clear that CORK reduces significantly the storage requirements with respect to a direct application of the rational Krylov method to the $(nd) \times (nd)$ GEP $\mathbf{A} - \lambda\mathbf{B}$, since essentially CORK represents $\mathbf{U}_j$ in terms of $n(j + d)$ parameters and, in addition, $n(j + d) \approx nj$ for moderate values of d. Therefore, the memory cost of CORK is approximately the cost of any Krylov method applied to an $n \times n$ GEP. The memory cost of each method is summarized in Table 2.7.

Moreover, it can be seen in [104, Section 5.4] that the orthogonalization cost of CORK is essentially independent of d for moderate values of d, and, so, much lower than the orthogonalization cost of a direct application of rational Krylov to $\mathbf{A} - \lambda\mathbf{B}$, and they are presented for both methods in Table 2.7. With respect to the comparison of the costs of the shift-and-invert steps in CORK (included in step 2 of Algorithm 2.16) and in rational Krylov (step 3 in Algorithm 2.4), we can say that in

Algorithm 2.16 Compact rational Krylov method (CORK)

Input: $Q_1 \in \mathbb{C}^{n\times r_1}$ and $\mathbf{R}_1 \in \mathbb{C}^{dr_1\times 1}$ with $Q_1^*Q_1 = I_{r_1}$ and $\mathbf{R}_1^*\mathbf{R}_1 = 1$, where $r_1 \leq d$.
Output: Approximate eigenpairs $(\lambda, \mathbf{x})$ associated to $\mathbf{A} - \lambda\mathbf{B}$, with $\mathbf{A}$, $\mathbf{B}$ as in (2.6).

for $j = 1, 2, \ldots$ **do**

1. Choose shift θ_j.

First level of orthogonalization:

2. Compute $\hat{u}^{(p)}$ by using the ULP decomposition in Theorem 2.15 with $\mu = \theta_j$ (see [104] for details).
3. Orthogonalize: $\tilde{q} = \hat{u}^{(p)} - Q_j Q_j^* \hat{u}^{(p)}$.
4. If $\tilde{q} \neq 0$ then compute next vector: $q_{j+1} = \tilde{q}/\|\tilde{q}\|_2$ and $Q_{j+1} = [Q_j \quad q_{j+1}]$. Otherwise $Q_{j+1} = Q_j$.

Second level of orthogonalization:

5. If $r_{j+1} > r_j$ then update matrices: $R_j^{(i)} = \begin{bmatrix} R_j^{(i)} \\ 0_{1\times j} \end{bmatrix}$ for $i = 1, \ldots, d$.
6. Compute: $\hat{\mathbf{r}}$ by using the ULP decomposition in Theorem 2.15 (see [104] for details).
7. Compute: $\tilde{\mathbf{r}} = \hat{\mathbf{r}} - \mathbf{R}_j h_j$, where $h_j = \mathbf{R}_j^*\hat{\mathbf{r}}$.
8. Next vector: $\mathbf{r}_{j+1} = \tilde{\mathbf{r}}/h_{j+1,j}$, where $h_{j+1,j} = \|\tilde{r}\|_2$ and $\mathbf{R}_{j+1} = [\mathbf{R}_j \quad \mathbf{r}_{j+1}]$.
9. Compute eigenpairs: (λ_i, t_i) of (2.27) and test for convergence.

end for

10. Compute eigenvectors: $\mathbf{x}_i = (I_d \otimes Q_{j+1})\mathbf{R}_{j+1}\underline{H}_j t_i$.

CORK the particular structure of the pencil $\mathbf{A}-\lambda\mathbf{B}$ together with the fact that only one block of the vector $\hat{\mathbf{u}}$ is needed allow us to perform this step very efficiently by essentially solving just one "difficult" $n \times n$ linear system (see [104, Algorithm 2]). In contrast, in rational Krylov the whole vector $\hat{\mathbf{u}}$ must be computed and there is some extra cost with respect to CORK even in the case the structure of $\mathbf{A} - \lambda\mathbf{B}$ is taken into account for solving the linear system $(\mathbf{A} - \theta_j\mathbf{B})\hat{\mathbf{u}} = \mathbf{B}\mathbf{u}_j$. On the other hand, there is some overhead cost involved in step 2 of Algorithm 2.16, since, in CORK, the actual vector $\mathbf{u}_j$ has to be constructed before solving the linear system associated to the shift-and-invert step. Fortunately, according to (2.111), this computation can be arranged as the single matrix-matrix product $Q_j[r_j^{(1)} \cdots r_j^{(d)}]$, where $r_j^{(1)}, \ldots, r_j^{(d)}$ are the blocks of the last column of $\mathbf{R}_j$, which allows optimal efficiency and cache usage on modern computers (see [63, p. 577]).

	Rational Krylov method	CORK method
Orthogonalization cost	$\mathcal{O}(ndj^2)$	$\mathcal{O}(n(d+j)j)$
Memory cost	ndj	$n(d+j)$

Table 2.7: Asymptotic orthogonalization and memory costs for RKS and CORK method after j iterations.

Inspired in CORK, we will develop in Chapter 4 the new algorithm R-CORK to solve large-scale and sparse rational eigenvalue problems by using a decomposition similar to (2.111) for the bases of the rational Krylov subspaces associated to the linearization (2.10) of the REP and by working in the spirit of the two levels of orthogonalization originally introduced in TOAR [100, 71, 63]. We will see that R-CORK has memory and computational advantages similar to those discussed for CORK in the previous paragraph.

Chapter 3

Preliminaries on matrix equations

3.1 The Sylvester equation

The matrix equation

$$AX + XB = C, \tag{3.1}$$

where $A \in \mathbb{C}^{m\times m}$, $B \in \mathbb{C}^{n\times n}$, $C \in \mathbb{C}^{m\times n}$ and the unknown matrix $X \in \mathbb{C}^{m\times n}$ is called the *Sylvester equation*. This equation is considered the most important matrix equation in linear algebra and it has been studied in both pure and applied mathematics, having several applications in different fields as control theory [5, 26], signal processing [50], model reduction [5, 106], image restoration [17], decoupling techniques for ordinary and partial differential equations [37], implementation of implicit numerical methods for ordinary differential equations [38], block-diagonalization of matrices [44, Chapter 7] and Newton methods for solving NLEPs [46].

A Sylvester equation is called *stable* when both $\Lambda(A)$ and $\Lambda(B)$ lie in the open left half plane. If $\Lambda(A)$ and $\Lambda(B)$ are contained in the open right plane, we say that the Sylvester equation is *anti-stable*.

In order to solve the Sylvester equation (3.1), we introduce some previous results related to the Kronecker product of matrices. The goal is to transform the matrix equation (3.1) into a standard system of linear equations. It is important to remark that the results presented in Section 3.1.1 follow for matrices with entries in any field $\mathbb{F}$ but, for simplicity, we state them for $\mathbb{F} = \mathbb{C}$.

3.1.1 Existence and uniqueness of solutions

In the study of matrix equations, it is often convenient to consider matrices in $\mathbb{C}^{m\times n}$ as vectors by ordering their entries in a conventional way. Definition 3.1 introduces the common convention of stacking columns, left to right.

Definition 3.1. *With each matrix* $A = [a_{ij}] \in \mathbb{C}^{m\times n}$*, we associate the vector* $\mathrm{vec}(A) \in \mathbb{C}^{mn}$ *defined by*

$$\mathrm{vec}(A) := [a_{11}, \ldots, a_{m1}, a_{12}, \ldots, a_{m2}, \ldots, a_{1n}, \ldots, a_{mn}]^T. \tag{3.2}$$

Definition 3.2. *[52, Definition 4.2.1] Let* $A = [a_{ij}] \in \mathbb{C}^{m\times n}$ *and* $B = [b_{ij}] \in \mathbb{C}^{p\times q}$*. The Kronecker product of* A *and* B *is denoted by* $A \otimes B$ *and defined to be the block matrix*

$$A \otimes B = \begin{bmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{bmatrix} \in \mathbb{C}^{mp\times nq}.$$

Theorem 3.3 determines the eigenvalues of the Kronecker product of two square complex matrices.

Theorem 3.3. *[52, Theorem 4.2.12] Let* $A \in \mathbb{C}^{m\times m}$ *and* $B \in \mathbb{C}^{n\times n}$*. If* $\lambda \in \Lambda(A)$ *and* $x \in \mathbb{C}^m$ *is a corresponding eigenvector of* A*, and if* $\mu \in \Lambda(B)$ *and* $y \in \mathbb{C}^n$ *is a corresponding eigenvector of* B*, then* $\lambda\mu \in \Lambda(A \otimes B)$ *and* $x \otimes y \in \mathbb{C}^{mn}$ *is a corresponding eigenvector of* $A \otimes B$*. Every eigenvalue of* $A \otimes B$ *arises as such a product of eigenvalues of* A *and* B*. If* $\Lambda(A) = \{\lambda_1, \ldots, \lambda_m\}$ *and* $\Lambda(B) = \{\mu_1, \ldots, \mu_n\}$*, then* $\Lambda(A \otimes B) = \{\lambda_i\mu_j : i = 1, \ldots, m, j = 1, \ldots, n\}$ *(including algebraic multiplicities in all three cases). In particular,* $\Lambda(A \otimes B) = \Lambda(B \otimes A)$*.*

As we mentioned before, the Kronecker product can be used to obtain a convenient representation for many linear matrix transformations and linear matrix equations. A key representation is given in Lemma 3.4.

Lemma 3.4. *[52, Lemma 4.3.1] Consider the matrix equation*

$$AXB = C, \tag{3.3}$$

where $A \in \mathbb{C}^{m\times n}$*,* $B \in \mathbb{C}^{p\times q}$*, and* $C \in \mathbb{C}^{m\times q}$ *are given, and* $X \in \mathbb{C}^{n\times p}$ *is the unknown matrix. Then, the equation* (3.3) *is equivalent to the linear system of* qm *equations in* np *unknowns given by*

$$(B^T \otimes A)\mathrm{vec}(X) = \mathrm{vec}(C), \tag{3.4}$$

this is, $\mathrm{vec}(AXB) = (B^T \otimes A)\mathrm{vec}(X)$*.*

Now, the equation (3.1) can be written by using the Kronecker product and the $\mathrm{vec}(\cdot)$ notation (3.2), obtaining the system:

$$[(I_n \otimes A) + (B^T \otimes I_m)]\mathrm{vec}(X) = \mathrm{vec}(C), \tag{3.5}$$

where I_n and I_m represent the identity matrices of sizes n and m, respectively. The matrix

$$A \oplus_S B := (I_n \otimes A) + (B^T \otimes I_m) \tag{3.6}$$

is called the *Kronecker sum* of A and B. Since the Kronecker product of A and B has as eigenvalues all possible pairwise products of the eigenvalues of A and B, it is natural to think that the eigenvalues of the Kronecker sum of A and B satisfy a similar relation. This result is summarized in Theorem 3.5.

Theorem 3.5. *Let $A \in \mathbb{C}^{m\times m}$ and $B \in \mathbb{C}^{n\times n}$ be given. If $\lambda \in \Lambda(A)$ and $x \in \mathbb{C}^m$ is a corresponding eigenvector of A, and if $\mu \in \Lambda(B)$ and $y \in \mathbb{C}^n$ is a corresponding eigenvector of B^T, then $\lambda+\mu$ is an eigenvalue of $A \oplus_S B$ and $y \otimes x \in \mathbb{C}^{mn}$ is a corresponding eigenvector. Every eigenvalue of the Kronecker sum arises as such a sum of eigenvalues of A and B. If $\Lambda(A) = \{\lambda_1, \ldots, \lambda_m\}$ and $\Lambda(B) = \{\mu_1, \ldots, \mu_n\}$, then $\Lambda(A \oplus_S B) = \{\lambda_i + \mu_j : i = 1, \ldots, m, j = 1, \ldots, n\}$ (including algebraic multiplicities in all three cases). In particular, $\Lambda(A \oplus_S B) = \Lambda(B \oplus_S A)$.*

Roth proved in 1952 [81] necessary and sufficient conditions for the existence of solution of the Sylvester equation (3.1). The unicity of solution for (3.1) is determined by the eigenvalues of A and B.

Theorem 3.6. *[81, Theorem II] Let $A \in \mathbb{C}^{m\times m}$, $B \in \mathbb{C}^{n\times n}$ and $C \in \mathbb{C}^{m\times n}$. The Sylvester equation $AX + XB = C$ has solution, if and only if, the matrices*

$$M_1 = \begin{bmatrix} A & -C \\ 0 & -B \end{bmatrix}, \quad M_2 = \begin{bmatrix} A & 0 \\ 0 & -B \end{bmatrix} \tag{3.7}$$

are similar.

Theorem 3.7. *[52, Theorem 4.4.6] Let $A \in \mathbb{C}^{m\times m}$ and $B \in \mathbb{C}^{n\times n}$. The Sylvester equation $AX + XB = C$ has a unique solution $X \in \mathbb{C}^{m\times n}$ for each $C \in \mathbb{C}^{m\times n}$, if and only if, $\Lambda(A) \cap \Lambda(-B) = \emptyset$.*

In the following section we present the Bartels-Stewart algorithm to solve the Sylvester equation (3.1) for small to medium size matrices A, B and C.

3.1.2 The Bartels-Stewart algorithm

The Bartels-Stewart algorithm [9] was introduced in 1972 to solve the Sylvester equation (3.1) when the matrices A, B and C are real, dense and have small to medium size. This method is based on the Schur reduction to quasi-triangular form by orthogonal similarity transformations. For historical reasons, we discuss the Bartels-Stewart algorithm for real matrices as appears in [9], however, it is easy to extend the procedure for complex matrices.

Consider the Sylvester equation (3.1) and reduce A to a lower real Schur form A' by an orthogonal similarity transformation U, this is,

$$A' = U^T A U = \begin{bmatrix} A'_{11} & & & \\ A'_{21} & A'_{22} & & \\ \vdots & \vdots & \ddots & \\ A'_{p1} & A'_{p2} & \cdots & A'_{pp} \end{bmatrix} \tag{3.8}$$

where the matrices A'_{ii}, $i = 1, \ldots, p$ are square of size at most 2. In a similar way, B is reduced to upper real Schur form by an orthogonal matrix V

$$B' = V^T B V = \begin{bmatrix} B'_{11} & B'_{12} & \cdots & B'_{1q} \\ & B'_{22} & \cdots & B'_{2q} \\ & & \ddots & \vdots \\ & & & B'_{qq} \end{bmatrix} \tag{3.9}$$

where the matrices B'_{ii}, $i = 1, \ldots, q$ are also square of size at most 2. If we consider

$$C' = U^T C V = \begin{bmatrix} C'_{11} & \cdots & C'_{1q} \\ \vdots & \ddots & \vdots \\ C'_{p1} & \cdots & C'_{pq} \end{bmatrix} \tag{3.10}$$

and

$$X' = U^T X V = \begin{bmatrix} X'_{11} & \cdots & X'_{1q} \\ \vdots & \ddots & \vdots \\ X'_{p1} & \cdots & X'_{pq} \end{bmatrix},$$

partitioned according to the partitions of A' and B', the equation (3.1) is equivalent to

$$A'X' + X'B' = C'. \tag{3.11}$$

Then, if the partitions of A', B', C' and X' are conformal, we have

$$A'_{kk}X'_{kl} + X'_{kl}B'_{ll} = C'_{kl} - \sum_{j=1}^{k-1} A'_{kj}X'_{jl} - \sum_{i=1}^{l-1} X'_{ki}B'_{il}, \quad k = 1, \ldots, p;\ l = 1, \ldots, q. \tag{3.12}$$

These equations are solved successively for $X'_{11}, X'_{21}, \ldots, X'_{p1}, X'_{12}, X'_{22}, \ldots, X'_{p2}, \ldots$ and the solution X of (3.1) is given by $X = UX'V^T$.

Although the solution for X'_{kl} in (3.12) requires to solve a Sylvester equation, the sizes of the matrices A'_{kk} and B'_{ll} are of order at most two, therefore, the solution of (3.12) is obtained by solving a linear system of order at most four. For example, if $A'_{kk} = [a'_{ij}]$ and $B'_{ll} = [b'_{ij}]$ are both of order two, then the system that needs to be solved is given by

$$\begin{bmatrix} a'_{11} + b'_{11} & a'_{12} & b'_{21} & 0 \\ a'_{21} & a'_{22} + a'_{11} & 0 & b'_{21} \\ b'_{12} & 0 & a'_{11} + b'_{22} & a'_{12} \\ 0 & b'_{12} & a'_{21} & a'_{22} + b'_{22} \end{bmatrix} \begin{bmatrix} x'_{11} \\ x'_{21} \\ x'_{12} \\ x'_{22} \end{bmatrix} = \begin{bmatrix} d_{11} \\ d_{21} \\ d_{12} \\ d_{22} \end{bmatrix} \tag{3.13}$$

where x'_{ij} denotes the elements of X'_{lk} and d_{ij} denotes the elements of the right-hand side of (3.12). The Bartels-Stewart algorithm is summarized in Algorithm 3.1.

Algorithm 3.1 Bartels-Stewart algorithm

Input: Coefficient matrices $A \in \mathbb{R}^{m\times m}$, $B \in \mathbb{R}^{n\times n}$ and right-hande side $C \in \mathbb{R}^{m\times n}$.
Output: The solution X of the Sylvester equation $AX + XB = C$.

1. Compute a lower triangular Schur form $A' = U^T AU$ partitioned as in (3.8).
2. Compute an upper triangular Schur form $B' = V^T BV$ partitioned as in (3.9).
3. Compute $C' = U^T CV$ partitioned as in (3.10).

for $l = 1, \ldots, q$ **do**
  for $k = 1, \ldots, p$ **do**
    4. Solve the Sylvester equation for X'_{kl}

$$A'_{kk}X'_{kl} + X'_{kl}B'_{ll} = C'_{kl} - \sum_{j=1}^{k-1} A'_{kj}X'_{jl} - \sum_{i=1}^{l-1} X'_{ki}B'_{il}$$

    by solving a linear system of size at most 4.
  end for
end for

6. Compute $X = UX'V^T$.

The Bartels-Stewart algorithm requires $\mathcal{O}(m^3 + n^3)$ flops and the detailed code appears in [9]. This cost is expensive if m and/or n are large, for this reason, Krylov methods have been developed to solve large-scale Sylvester equations and a brief summary is presented in Section 3.3.

Different direct methods that improve the Bartels-Stewart algorithm at some extent have been developed for solving the Sylvester equation and the interested reader can find more information in [42, 58, 59, 96]. Note that the development of the Bartels-Stewart algorithm for complex A, B and C is analogous to the one presented in this section, with the main difference that the diagonal blocks of the matrices A' and B' are always square of size 1×1.

The Bartels-Stewart algorithm inspired the development of algorithms to solve other matrix equations, and in particular, we present in Section 3.2.2 an extension for solving the ⋆-Sylvester equation presented in [28].

3.2 The Sylvester equation for ⋆-congruence

In this section, we consider the matrix equation

$$AX + X^\star B = C, \tag{3.14}$$

where $A \in \mathbb{C}^{m\times n}$, $B \in \mathbb{C}^{n\times m}$, $C \in \mathbb{C}^{m\times m}$ and the unknown is $X \in \mathbb{C}^{n\times m}$ and the operator $(\cdot)^\star$ denotes either the transpose $(\cdot)^T$ or the conjugate transpose $(\cdot)^*$ of a

matrix. The matrix equation (3.14) is called the $\star$-*Sylvester equation* or *Sylvester equation for* $\star$-*congruence*.

Since one of the goals of this thesis is to solve numerically the Sylvester equation for T-congruence when the rank of C in (3.14) is low, some applications of this matrix equation are discussed. As we mentioned in Section 3.1, it is well-known that standard Sylvester equations with low-rank right-hand sides appear very often in linear system theory and control theory, in particular in the context of model order reduction and as intermediate steps in the solution of continuous-time algebraic Riccati equations (CARE) by iterative methods in linear-quadratic optimal control problems [5, 12]. Analogously, some particular non-standard problems in optimal Hankel-norm model reduction and H_2/H_∞ controls related to nonstandard J-spectral factorization problems lead to T-CARE whose nonlinear and constant terms both have low rank, under the natural assumption that the numbers of inputs and outputs are much smaller than the number of internal states of the system (see [61, 62] and the references therein). For instance the T-CARE appearing in [61, 62] is

$$A^T X + X^T A + X^T R X + Q = 0, \tag{3.15}$$
$$E^T X - X^T E = 0, \tag{3.16}$$

where $E, A, Q, R \in \mathbb{R}^{n\times n}$, E may be singular, $Q = Q^T$, $R = R^T$, Q and R are indefinite, and the ranks of Q and R are much smaller than n under the conditions mentioned above. The problem of interest in applications is to compute a *stabilizing solution* X of (3.15)-(3.16), which roughly speaking is a solution such that the corresponding close-loop matrix has its eigenvalues in the left-hand plane including the extended imaginary axis. Conditions for the existence of such solutions have been established in [62], and their numerical computation is a nontrivial problem considered in [61] and solved satisfactorily in [22] for small- to medium-size matrices. However, the solution of (3.15)-(3.16) for large-scale matrices remains an open problem. The efficient solution of this problem will require iterative methods that will need efficient solvers of large-scale T-Sylvester equations with low-rank right-hand sides, similarly to the solution of large-scale standard CAREs, which requires solvers of large-scale Sylvester equations with low-rank right-hand sides [12]. To realize this point, note that the two equations (3.15)-(3.16) are equivalent to:

$$(A^T + E^T)X + X^T(A - E) + X^T R X + Q = 0, \tag{3.17}$$

because the sum of (3.15) and (3.16) yields (3.17), while (3.17) plus and minus its transpose yields (3.15) and (3.16), respectively [21, Section 2.3]. Clearly, any fixed point iteration or any Newton-based method for solving (3.17) would need the solution of T-Sylvester equations with low-rank right-hand sides. Variants of equations (3.15)-(3.16) where Q and R are low-rank positive semidefinite matrices have also appeared in applications [111], and they can be connected to T-Sylvester

equations exactly in the same way. Finally, the T-Sylvester equation is also used in [56] for the development of a new algorithm for the solution of delay Lyapunov equations.

3.2.1 Existence and uniqueness of solutions

In order to state necessary and sufficient conditions for the existence and uniqueness of a solution X of (3.14), we need to introduce the following notion.

Definition 3.8. *A set of numbers $\{\lambda_1, \lambda_2, \ldots, \lambda_n\} \subset \mathbb{C} \cup \{\infty\}$ is $\star$-reciprocal free if $\lambda_i \neq 1/\lambda_j^\star$ for any $1 \leq i, j \leq n$.*

Note that Definition (3.8) admits 0 and/or ∞ as elements of the set $\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ and by convention $1/0 = \infty$ and $1/\infty = 0$ in Definition 3.8. Also, in the case of $\star = T$, $\lambda_i^T = \lambda_i$ and for $\star = *$, $\lambda_i^* = \overline{\lambda_i}$.

The uniqueness of the solution of the matrix equation (3.14) is discussed in [64, Lemma 8] and presented in Lemma 3.9. In particular, if A and B in (3.14) are both complex square matrices of size n, Lemma 3.9 gives necessary and sufficient conditions for the existence of a unique solution for every right-hand side C in the complex field. Note that if $A \in \mathbb{C}^{m\times n}$ and $B \in \mathbb{C}^{n\times m}$ are rectangular matrices in (3.14), then $X \in \mathbb{C}^{n\times m}$ while $AX + X^\star B \in \mathbb{C}^{m\times m}$. Therefore, the operator $X \mapsto AX + X^\star B$ is never invertible, and then, the equation (3.14) never has a unique solution for every right-hand side C.

Lemma 3.9. *[64, Lemma 8] The $\star$-Sylvester equation $AX + X^\star B = C$ with $A, B \in \mathbb{C}^{n\times n}$ has a unique solution X for every right-hand side $C \in \mathbb{C}^{n\times n}$ if and only if the following two conditions hold:*

1. *The pencil $A - \lambda B^\star$ is regular, and*

2a. *if $\star = T$, $\Lambda(A, B^T) \setminus \{1\}$ is T-reciprocal free and if $1 \in \Lambda(A, B^T)$, then it has algebraic multiplicity 1, or*

2b. *if $\star = *$, $\Lambda(A, B^*)$ is $*$-reciprocal free.*

In the past, some references erroneously replaced condition 2 in Lemma 3.9 simply by $\Lambda(A, B^\star)$ is "reciprocal free" which requires that $1 \notin \Lambda(A, B^\star)$ (see for example [54, Theorem 3] for the case $\star = T$). It is easy to construct examples showing that this requirement is not needed. Consider, for instance, $n = 1$, $A = B = 1$ and $\star = T$. In this case, the only eigenvalue of $A - \lambda B^\star$ is 1, but $AX + X^\star B = C$ has the unique solution $X = C/2$.

The consistency of (3.14) is related to the concept of congruence introduced in Definition 3.10.

Definition 3.10. *Two matrices $A, B \in \mathbb{C}^{n\times n}$ are $\star$-congruent if there exists a non-singular matrix $P \in \mathbb{C}^{n\times n}$ such that $B = P^\star AP$.*

Theorem 3.11 extends the equivalence of conditions (a) and (b) in [114, Theorem 2] which is stated only for matrices over the complex field $\mathbb{C}$ and for the case $\star = *$. Theorem 3.11 establishes a necessary and sufficient condition for the consistency of the Sylvester equation for $\star$-congruence for rectangular matrices with entries in any field of characteristic different from two.

Theorem 3.11. *[28, Theorem 2.3] Let $\mathbb{F}$ be a field of characteristic different from two and let $A \in \mathbb{F}^{m\times n}$, $B \in \mathbb{F}^{n\times m}$, $C \in \mathbb{F}^{m\times m}$ be given. There is some $X \in \mathbb{F}^{n\times m}$ such that*

$$AX + X^{\star}B = C$$

if and only if

$$\begin{bmatrix} C & A \\ B & 0 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 0 & A \\ B & 0 \end{bmatrix} \quad \text{are } \star\text{-congruent.} \tag{3.18}$$

In Chapter 5, we focus in solving numerically the Sylvester equation for T-congruence for square invertible large-scale real matrices A, B and C. With this goal, we present the following theorem, which reveals a relationship between (3.14) for $\star = T$ and a standard Sylvester equation under certain assumptions, and also, for the same case, between (3.14) and a generalized Sylvester equation. To the best of our knowledge, these relations are an original contribution of this PhD Thesis and they appear in [35, Theorem 2.3].

Theorem 3.12. *[35, Theorem 2.3] Let $A, B, C \in \mathbb{R}^{n\times n}$ and assume that A and B are nonsingular. Consider the matrix equations*

$$AX + X^{T}B = C, \tag{3.19}$$

$$\left(B^{-T}A\right)X - X\left(A^{-T}B\right) = B^{-T}C - B^{-T}C^{T}A^{-T}B, \tag{3.20}$$

$$AXA^{T} - B^{T}XB = C - C^{T}A^{-T}B, \tag{3.21}$$

for the unknown $X \in \mathbb{R}^{n\times n}$. Then the following statements hold.

(a) *If X_0 is a solution of the T-Sylvester equation* (3.19) *then X_0 is also a solution of the Sylvester equation* (3.20).

(b) *If the Sylvester equation* (3.20) *has a unique solution X_0 then the T-Sylvester equation* (3.19) *has also a unique solution, which is equal to X_0.*

(c) $\operatorname{rank}\left(B^{-T}C - B^{-T}C^{T}A^{-T}B\right) \leq 2\operatorname{rank}(C)$.

(d) *X_0 is a solution of the generalized Sylvester equation* (3.21) *if and only if (X_0A^T) is a solution of the Sylvester equation* (3.20)*, i.e., there is a bijection between the sets of solutions of* (3.21) *and* (3.20).

Proof. (a) If X_0 satisfies (3.19), then

$$\begin{aligned} AX_0 &= C - X_0^T B, \\ X_0 &= A^{-1}C - A^{-1}X_0^T B, \\ X_0^T &= C^T A^{-T} - B^T X_0 A^{-T}. \end{aligned}$$

Inserting this expression for X_0^T into $AX_0 + X_0^T B = C$, we obtain

$$\begin{aligned} AX_0 + C^T A^{-T} B - B^T X_0 A^{-T} B &= C, \\ AX_0 - B^T X_0 A^{-T} B &= C - C^T A^{-T} B. \end{aligned}$$

Multiplying the latter equation with B^{-T} on the left yields that X_0 satisfies (3.20).
(b) Assume that (3.20) has a unique solution, which is equivalent to assuming that

$$\Lambda\left(B^{-T}A\right) \cap \Lambda\left(A^{-T}B\right) = \emptyset. \tag{3.22}$$

Note that

$$\Lambda\left(\left(B^{-T}A\right)^{-1}\right) = \Lambda\left(\left(A^{-1}B^T\right)^T\right) = \Lambda\left(BA^{-T}\right) = \Lambda\left(A^{-T}B\right).$$

This shows that (3.22) is equivalent to

$$\Lambda\left(B^{-T}A\right) \cap \Lambda\left(\left(B^{-T}A\right)^{-1}\right) = \emptyset.$$

In other words, $\Lambda\left(B^{-T}A\right) = \Lambda\left(A, B^T\right)$ is T-reciprocal free. Thus, Lemma 3.9 shows that (3.19) has a unique solution.

So far, we have established that the unique solvability of (3.20) implies the unique solvability of (3.19), but not yet that the solutions of both equations are the same. This, however, follows directly from part (a), which states that the solution set of (3.19) is included in the solution set of (3.20). Therefore both sets must be identical when they only have one element.

(c) Elementary results on ranks [51, p. 13] yield

$$\operatorname{rank}\left(B^{-T}C - B^{-T}C^T A^{-T}B\right) \leq \operatorname{rank}\left(B^{-T}C\right) + \operatorname{rank}\left(B^{-T}C^T A^{-T}B\right) \leq 2\operatorname{rank}(C).$$

(d) If we multiply equation (3.20) on the left by B^T, then we get the equivalent equation $A(XA^{-T})A^T - B^T(XA^{-T})B = C - C^T A^{-T}B$ and the result follows. □

Note that from Theorem 3.12(c), if the right-hand side of (3.19) has low rank, then the right-hand side of (3.20) has low rank too. Also note that the converse of Theorem 3.12(b) does not hold. The T-Sylvester equation (3.19) may have a unique solution when the Sylvester equation (3.20) does not. To see this, consider again the case $n = 1$ with $A = B = 1$ and arbitrary $C \in \mathbb{R}$. Then (3.19) has the unique solution $X = C/2$, while (3.20) reads $X - X = 0$ and thus every $X \in \mathbb{R}$ is a solution

of (3.20). In addition, note that also every $X \in \mathbb{R}$ is a solution of the generalized Sylvester equation (3.21) in this example. It follows from the proof of Theorem 3.12 that such a situation can only occur when (3.19) has a unique solution and 1 is a simple eigenvalue of $A - \lambda B^T$.

In Section 3.2.2 we discuss a numerical method to solve (3.14), when it has a unique solution for every right-hand side, for small to medium size matrices A, B and C based on the generalized Schur decomposition of a regular matrix pencil. This method can be seen as an extension of the Bartels-Stewart algorithm.

3.2.2 Numerical solution of the $\star$-Sylvester equation via the generalized Schur decomposition

This section is devoted to solve the equation (3.14) for $A, B, C \in \mathbb{F}^{n\times n}$ with $\mathbb{F} = \mathbb{R}$ or $\mathbb{C}$. We will assume that the conditions of Lemma 3.9 hold, that is, we are assuming that the equation (3.14) has a unique solution for every right-hand side C. Note that if $\mathbb{F} = \mathbb{R}$, then the unique solution of (3.14) is real, both for $\star = T$ and $\star = *$. For $\star = T$ this result is trivial, since nonsingular linear systems with real matrix coefficient and real right-hand side vector have a unique real solution. For $\star = *$, if X is a solution of (3.14), then $\overline{X}$ is also a solution and therefore $X = \overline{X}$, implying that X is real. Thus, if $\mathbb{F} = \mathbb{R}$, then we only need to consider $\star = T$.

Using Lemma 3.4, we can write the matrix equation (3.14) for $\star = T$ as a standard linear system for the unknown $\mathrm{vec}(X) \in \mathbb{F}^{n^2}$, obtaining

$$\left[(I_n \otimes A) + (B^T \otimes I_n)\Pi\right] \mathrm{vec}(X) = \mathrm{vec}(C), \tag{3.23}$$

where $\Pi \in \mathbb{R}^{n^2\times n^2}$ is a permutation matrix, that satisfies $\mathrm{vec}(X^T) = \Pi\mathrm{vec}(X)$ for every $X \in \mathbb{F}^{n\times n}$ [52, Theorem 4.3.8]. If we use directly Gaussian elimination with partial pivoting to solve (3.23), the cost is $O(n^6)$ flops, which is prohibitive, except for very small n. Similar techniques allow us to write $AX + X^*B = C$, in the complex case, as a standard real linear system for the unknown $\left[\,(\mathrm{vec}(\mathrm{Re}\, X))^T \;\; (\mathrm{vec}(\mathrm{Im}\, X))^T\,\right]^T \in \mathbb{F}^{2n^2}$, where $\mathrm{Re}\, X$ and $\mathrm{Im}\, X$ are the real and imaginary parts of X. Gaussian elimination with partial pivoting on this linear system leads again to a prohibitive cost of $O(n^6)$ flops.

De Terán and Dopico in [28] developed an algorithm to compute the unique solution of (3.14) with a cost of $O(n^3)$ flops. This algorithm is based on the generalized Schur decomposition of the pair $(A, B^\star)$ and it follows the spirit of the Bartels-Stewart algorithm for the standard Sylvester equations.

First, the generalized Schur decomposition of the pair $(A, B^\star)$ is computed by using the QZ algorithm [44, Section 7.7] obtaining

$$A = URV, \quad B^\star = USV, \tag{3.24}$$

where U, $V \in \mathbb{C}^{n\times n}$ are unitary matrices and R, $S \in \mathbb{C}^{n\times n}$ are upper triangular matrices. In particular, if A and B are real matrices, then a real generalized Schur decomposition can be computed, for which U, $V \in \mathbb{R}^{n\times n}$ are real orthogonal matrices, $S \in \mathbb{R}^{n\times n}$ is upper triangular, but $R \in \mathbb{R}^{n\times n}$ is upper quasi-triangular, that is, block upper triangular with 1×1 or 2×2 diagonal blocks. Defining

$$E := U^* C \, (U^\star)^* \tag{3.25}$$

we have that equation (3.14) is equivalent to

$$RW + W^\star S^\star = E \tag{3.26}$$

where $W = VX \, (U^\star)^*$. Note that the pencils $R - \lambda S$ and $A - \lambda B^\star$ are strictly equivalent, so Lemma 3.9 guarantees that the $\star$-Sylvester equation (3.26) has a unique solution W for every right-hand side E.

The transformed equation (3.26) can be solved in an efficient way proposed in [28]. To cover the possible case of generalized real Schur decompositions in (3.24) when $\mathbb{F} = \mathbb{R}$ (recall that in this case $\star = T$), we consider R and S partitioned into $p \times p$ blocks as

$$R = \begin{bmatrix} R_{11} & R_{12} & \cdots & R_{1p} \\ & R_{22} & & \vdots \\ & & \ddots & R_{p-1,p} \\ & & & R_{pp} \end{bmatrix}, \qquad S = \begin{bmatrix} S_{11} & S_{12} & \cdots & S_{1p} \\ & S_{22} & & \vdots \\ & & \ddots & S_{p-1,p} \\ & & & S_{pp} \end{bmatrix}, \tag{3.27}$$

where $R_{ij}, S_{ij} \in \mathbb{F}^{n_i \times n_j}$ for $1 \le i, j \le p$, and $n_k = 1$ or 2 for $1 \le k \le p$. The diagonal blocks S_{ii} are always upper triangular matrices, but the diagonal blocks R_{ii} may be not if $A, B \in \mathbb{R}^{n\times n}$. If complex generalized Schur decompositions are computed in (3.24), then $p = n$ and $n_k = 1$ for $1 \le k \le n$. The matrices W and the right-hand side E are also partitioned into $p \times p$ blocks as

$$W = \begin{bmatrix} W_{11} & W_{12} & \cdots & W_{1p} \\ W_{21} & W_{22} & & W_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ W_{p1} & W_{p2} & \cdots & W_{pp} \end{bmatrix}, \qquad E = \begin{bmatrix} E_{11} & E_{12} & \cdots & E_{1p} \\ E_{21} & E_{22} & & E_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ E_{p1} & E_{p2} & \cdots & E_{pp} \end{bmatrix}, \tag{3.28}$$

where the sizes of the blocks are $W_{ij}, E_{ij} \in \mathbb{F}^{n_i \times n_j}$, that is, the same sizes as in the partitions (3.27).

The authors of [28] propose to determine first simultaneously the last block column and the last block row of W, then to determine simultaneously the last block column and the last block row of $W(1 : p-1, 1 : p-1) := [W_{ij}]_{i,j=1}^{p-1}$, then to determine simultaneously the last block column and the last block row of $W(1 : p-2, 1 : p-2)$, and, so on until we determine W_{11}. Observe that we have extended in the previous

discussion standard MATLAB notation for submatrices from indices of entries to block-indices, since $W(1:p-1, 1:p-1)$ denotes the submatrix of W consisting of block rows 1 through $p-1$ and block columns 1 through $p-1$. Using this idea, and developing a similar process than the Bartels-Stewart algorithm, they obtain Algorithm 3.2 to solve the matrix equation (3.26).

Algorithm 3.2 Solution of $RW + W^\star S^\star = E$ for (quasi) triangular coefficient matrices

---

Input: Matrices R, $S \in \mathbb{F}^{n\times n}$ with $\mathbb{F} = \mathbb{R}$ or $\mathbb{C}$ obtained in (3.24) and E obtained in (3.25).

Output: $W \in \mathbb{F}^{n\times n}$, which is the unique solution of $RW + W^\star S^\star = E$.

for $j = p : -1 : 1$ **do**

1. Get W_{jj} by solving $R_{jj}W_{jj} + W_{jj}^\star S_{jj}^\star = E_{jj}$.

for $i = j-1 : -1 : 1$ **do**

2. Get W_{ij} and W_{ji} by solving $\left\{ \begin{array}{l} S_{ii}W_{ij} + W_{ji}^\star R_{jj}^\star = E_{ji}^\star - \sum_{k=i+1}^{j} S_{ik}W_{kj} \\ R_{ii}W_{ij} + W_{ji}^\star S_{jj}^\star = E_{ij} - \sum_{k=i+1}^{j} R_{ik}W_{kj} \end{array} \right\}$

end for

3. Update the right-hand side E

$$\begin{aligned} E(1:j-1, 1:j-1) = \; & E(1:j-1, 1:j-1) - R(1:j-1, j)W(j, 1:j-1) \\ & -(S(1:j-1, j)W(j, 1:j-1))^\star \end{aligned}$$

end for

Note that in step 3 of Algorithm 3.2, we have used again MATLAB's notation for submatrices through block indices. Algorithm 3.3 summarizes the process to solve the Sylvester equation for $\star$-congruence (3.14).

Algorithm 3.3 Algorithm to solve $AX + X^\star B = C$

Input: $A, B, C \in \mathbb{F}^{n\times n}$ with $\mathbb{F} = \mathbb{R}$ or $\mathbb{C}$, such that A and B satisfy the conditions 1 and 2 in Lemma 3.9.

Output: The unique solution $X \in \mathbb{F}^{n\times n}$ of $AX + X^\star B = C$.

1. Compute the generalized Schur decomposition of the pair $(A, B^\star)$, $A = URV$, $B^\star = USV$.
2. Compute $E = U^* C\,(U^\star)^*$.
3. Use Algorithm 3.2 to solve the transformed equation $RW + W^\star S^\star = E$.
4. Compute $X = V^* W U^\star$.

Now, we analyze the computational costs of Algorithms 3.2 and 3.3. If we assume that $\mathbb{F} = \mathbb{R}$, the cost of Algorithm 3.2 is $2n^3 + O(n^2)$ flops, if $R_{ii} \in \mathbb{R}^{1\times 1}$ for all i. The cost of the QZ algorithm in Step 1 of Algorithm 3.3 is $66n^3 + O(n^2)$ flops (see [44, p. 385]). In addition, Steps 2 and 4 in Algorithm 3.3 amount to 4 matrix multiplications

of $n \times n$ matrices. Therefore the total cost of Algorithm 3.3 is $76n^3 + O(n^2)$ flops. If $\mathbb{F} = \mathbb{C}$, this cost is multiplied by a factor up to 6. This cost implies that Algorithm 3.3 can only be used for matrices $A, B, C \in \mathbb{C}^{n\times n}$ for small to medium size n. In Chapter 5 we propose a new method to solve Sylvester equations for T-congruence when the matrices A, B and C are large-scale and sparse and the right-hand side matrix C has low rank, based on Krylov methods.

3.3 Galerkin projection methods for the Sylvester equation

In this section, we discuss Galerkin projection methods for solving the Sylvester equation. These methods use Galerkin conditions on the residual to compute an approximate solution of the Sylvester equation and the approximate solution is computed in terms of the solution of a reduced problem. In particular, direct methods are used to solve these reduced problems. As we discussed before, since the original problem is projected into a problem of small size, Krylov methods are used for large-scale coefficient matrices A, B and C in the Sylvester equation. In particular, for Sections 3.3.1 and 3.3.2, we consider the case $A, B, C \in \mathbb{C}^{n\times n}$ for historical reasons. Also note that in both sections m denotes the dimension of the Krylov subspace.

In the literature, a first approach that was used for solving large-scale Sylvester equations was to apply a projection method on the linear system (3.5) which is equivalent to solve (3.1). In order to present this idea, we first introduce in Section 3.3.1 the full orthogonalization method [85] (FOM) which is based on Arnoldi for solving general linear systems of equations. However, FOM does not take into account the structure of the matrix involved in the linear system (3.5), and it results impractical a direct application for large-scale and sparse problems, specially in terms of storage and computational costs. To address this issue, an improved algorithm based on FOM is presented in Section 3.3.2 that replaces the Krylov subspace considered originally in FOM by a new one, which allows to work with the matrices A and B of size $n \times n$ instead of (3.6) of size $n^2 \times n^2$, obtaining a memory saving procedure.

The Sylvester equation arises in several applications, where the right-hand side matrix C has low rank [5]. When this happens, another class of Krylov methods has been developed to project the large-scale Sylvester equation onto another one of smaller size [90]. In Section 3.3.3 we present a brief summary on projection methods for Sylvester equations with low-rank right-hand side. These algorithms are based on the Arnoldi method and they approximate the solution by a low rank matrix. The implementation of these methods depends on the type of Krylov subspace that have been chosen, as block Krylov or extended Krylov, and it allows to develop memory efficient algorithms.

3.3.1 The full orthogonalization method

In this section we discuss the full orthogonalization method (FOM) introduced in [85] which is a method developed for solving general linear systems of equations. Also, a summary of this method is presented in [53, Section 2].

Consider a square linear system

$$Gx = b \tag{3.29}$$

where $G \in \mathbb{C}^{n\times n}$ is invertible and $b \in \mathbb{C}^n$.

The purpose of the full orthogonalization method is to construct an orthonormal basis of a certain Krylov subspace $\mathcal{K}_m(G, u_1)$ by using the Arnoldi method, and then, compute an approximation to the solution of (3.29) by solving a linear system of equations of smaller size by a direct method.

Let $x_0 \in \mathbb{C}^n$ be a given initial approximation of the solution of (3.29), and define the residual vector

$$r_0 := b - Gx_0. \tag{3.30}$$

FOM requires to determine a *correction* $z_0 \in \mathcal{K}_m(G, r_0)$ of x_0 that satisfies $G(x_0 + z_0) \approx b$, or equivalently, $Gz_0 \approx r_0$.

In order to determine a convenient representation of z_0, the Arnoldi process is applied to generate an orthonormal basis of $\mathcal{K}_m(G, r_0)$. Consider the first j iterations of the Arnoldi process for $\mathcal{K}_m(G, r_0)$, and recall the Arnoldi recurrence relation introduced in Proposition 2.20:

$$H_j = V_j^* G V_j, \quad GV_j = V_{j+1}\underline{H}_j, \tag{3.31}$$

where $V_{j+1} = [v_1, v_2, \cdots v_{j+1}] \in \mathbb{C}^{n\times(j+1)}$ contains the orthonormal Krylov vectors as columns, and $\underline{H}_j \in \mathbb{C}^{(j+1)\times j}$ is an upper Hessenberg matrix. Then, we can write

$$z_0 = V_j y_0 \tag{3.32}$$

for some $y_0 \in \mathbb{C}^j$. Since the full orthogonalization method is a Galerkin method, the vector y_0 is required to be such that the residual error

$$r_1 = r_0 - GV_j y_0 \tag{3.33}$$

is orthogonal to $\mathcal{K}_m(G, r_0)$, i.e.,

$$V_j^*(r_0 - GV_j y_0) = 0. \tag{3.34}$$

By using (3.31) and the fact that the first column of V_j is a scaling of r_0, the equation (3.34) can be rewritten as

$$H_j y_0 = \|r_0\|_2 e_1, \tag{3.35}$$

where $e_1 \in \mathbb{R}^j$ represents the first canonical vector. Then, the system (3.35) is solved for y_0, and z_0 can be determined from (3.32). Thus, the new approximate solution can be computed as

$$x_1 = x_0 + z_0, \tag{3.36}$$

and the corresponding residual error as

$$r_1 = r_0 - V_{j+1}\underline{H}_j y_0, \tag{3.37}$$

whose norm can be cheaply computed as $\|r_1\|_2 = \|\|r_0\|_2 e_1 - \underline{H}_j y_0\|_2$, and so, the convergence can be cheaply tested. The full orthogonalization method is summarized in Algorithm 3.4.

Algorithm 3.4 Full orthogonalization method for solving general linear systems of equations

Input: $G \in \mathbb{C}^{n\times n}$ with $\det(G) \neq 0$, $b \in \mathbb{C}^n$ and $x_0 \in \mathbb{C}^n$ an initial approximation of (3.29).

Output: An approximation for the unique solution $x \in \mathbb{C}^n$ of $Gx = b$.

1. Compute $r_0 = b - Gx_0$.

for $j = 1, 2, \ldots, m$ **do**

2. Use the Arnoldi method (Algorithm 2.1) to compute V_{j+1} whose columns form an orthonormal basis of $\mathcal{K}_{j+1}(G, r_0)$ and the upper Hessenberg matrix $\underline{H}_j$.
3. Solve the small linear system $H_j y = \|r_0\|_2 e_1$.
4. Compute $x_0 = x_0 + V_j y$ and test for convergence.

end for

Convergence properties of this method can be found in [85]. If $\|r_1\|_2$ is not sufficiently small for a reasonably small j, we can repeat the computations with $x_0 = x_1$ and $r_0 = r_1$. This process is known as the restarted full orthogonalization method [85].

If FOM wants to be applied for solving the linear system (3.5), the size of $A \oplus_S B$ in (3.6) turns into an issue since we need to store at each iteration a Krylov vector of size n^2. For this reason, it is attractive to replace the Krylov vectors by vectors of smaller size. This technique is developed in Section 3.3.2

3.3.2 An algorithm based on the tensor product of Krylov subspaces for solving the Sylvester equation

In this section we discuss a Krylov method for the unique solution of the Sylvester equation (3.1) with $A, B, C \in \mathbb{C}^{n\times n}$, which consists into project the original problem (3.1) into a Sylvester equation of reduced size, and use the solution of this reduced equation to approximate the solution of the original problem [53].

The full orthogonalization method is not attractive when the matrix $(A \oplus_S B)$ is large, since it is very expensive to store a basis of the Krylov subspace $\mathcal{K}_m(A \oplus_S B, r_0)$. In order to reduce the memory cost, the Krylov subspace $\mathcal{K}_m(A \oplus_S B, r_0)$ is replaced by the subspace $\mathcal{K}_m(B^T, g) \otimes \mathcal{K}_m(A, f)$, for certain vectors $f, g \in \mathbb{C}^n$ where the tensor product of these subspaces is defined by

$$\mathcal{K}_m(B^T, g) \otimes \mathcal{K}_m(A, f) := \text{span}\{w \otimes v : w \in \mathcal{K}_m(B^T, g), v \in \mathcal{K}_m(A, f)\},$$

and, as we will discuss later in this section, f and g are related to r_0.

Suppose, for the moment, f and g arbitrary but fixed vectors in $\mathbb{C}^n$, and use the Arnoldi method in Algorithm 2.1 to generate orthonormal bases $\{v_1, v_2, \ldots, v_{m+1}\}$ and $\{w_1, w_2, \ldots, w_{m+1}\}$ of $\mathcal{K}_{m+1}(A, f)$ and $\mathcal{K}_{m+1}(B^T, g)$, respectively. Using the usual notation $V_m := [v_1, v_2, \ldots v_m]$ and $W_m := [w_1, w_2, \ldots w_m]$ we obtain from the Arnoldi recurrence relation:

$$H_m^{(A)} = V_m^* A V_m, \quad H_m^{(B)} = W_m^* B^T W_m, \tag{3.38}$$

$$AV_m = V_{m+1}\underline{H}_m^{(A)}, \quad B^T W_m = W_{m+1}\underline{H}_m^{(B)}. \tag{3.39}$$

Since the columns of V_m and W_m are orthonormal, the columns of $W_m \otimes V_m$ are also orthonormal and they form a basis of $\mathcal{K}_m(B^T, g) \otimes \mathcal{K}_m(A, f)$.

Consider the linear system

$$(A \oplus_S B)\text{vec}(X) = \text{vec}(C), \tag{3.40}$$

and let $x_0 \in \mathbb{C}^{n^2}$ be an approximate solution of (3.40), and consider the residual vector r_0 as in (3.30) with $G := A \oplus_S B$. As for the full orthogonalization method, we want to compute a correction $z_0 \in \mathcal{K}_m(B^T, g) \otimes \mathcal{K}_m(A, f)$ of x_0 and obtain a new approximate solution $x_1 = x_0 + z_0$ for (3.40) such that the residual vector in (3.33) is orthogonal to $\mathcal{K}_m(B^T, g) \otimes \mathcal{K}_m(A, f)$. Then, the correction z_0 can be written as

$$z_0 = (W_m \otimes V_m)y_0, \tag{3.41}$$

for some vector $y_0 \in \mathbb{C}^{m^2}$. Substituting (3.41) in

$$r_1 = r_0 - (A \oplus_S B)z_0 \tag{3.42}$$

and requiring that $(W_m \otimes V_m)^* r_1 = 0$, the linear system

$$(I_m \otimes H_m^{(A)} + H_m^{(B)} \otimes I_m)y_0 = \bar{r}_0 \tag{3.43}$$

is obtained, where $H_m^{(A)}, H_m^{(B)} \in \mathbb{C}^{m \times m}$ are defined in (3.38) and $\bar{r}_0 := (W_m \otimes V_m)^* r_0 \in \mathbb{C}^{m^2}$. Note that the system (3.43) is equivalent to the Sylvester equation

$$H_m^{(A)} Y_0 + Y_0 \left(H_m^{(B)}\right)^T = \bar{R}_0, \tag{3.44}$$

where $\text{vec}(Y_0) = y_0$ and $\text{vec}(\bar{R}_0) = \bar{r}_0$. Since the size of the Sylvester equation (3.44) is small, it can be solved by using a direct method as the Bartels-Stewart algorithm.

When y_0 is computed, the correction z_0 is computed by using (3.41). Then, $x_1 = x_0 + z_0$ can be computed, and with this, the residual vector r_1 in (3.42). The computation of r_1 can be expensive if A and B are not sparse, for that reason, the residual r_1 can be computed as

$$r_1 = r_0 - (W_m \otimes V_{m+1}\underline{H}_m^{(A)})y_0 - (W_{m+1}\underline{H}_m^{(B)} \otimes V_m)y_0. \tag{3.45}$$

The implementation of the Galerkin algorithm is summarized in Algorithm 3.5.

Algorithm 3.5 Tensor product of Krylov subspaces for solving the Sylvester equation

Input: A, B, $C \in \mathbb{C}^{n\times n}$, such that the Sylvester equation $AX + XB = C$ has a unique solution, $f, g \in \mathbb{C}^n$ with $\|f\|_2 = \|g\|_2 = 1$, and x_0 an initial approximation of the solution of (3.40)

Output: The unique solution $X \in \mathbb{C}^{n\times n}$ of $AX + XB = C$.

1. Compute $r = \text{vec}(C) - (A \oplus_S B)x_0$.

for $j = 1, 2, \ldots, m$ **do**

2. Use the Arnoldi method (Algorithm 2.1) to compute V_{j+1} whose columns form an orthonormal basis of $\mathcal{K}_{j+1}(A, f)$ and the upper Hessenberg matrix $\underline{H}_j^{(A)}$.

3. Use the Arnoldi method (Algorithm 2.1) to compute W_{j+1} whose columns form an orthonormal basis of $\mathcal{K}_{j+1}(B^T, g)$ and the upper Hessenberg matrix $\underline{H}_j^{(B)}$.

4. Compute $\bar{r} = (W_j \otimes V_j)^* r$.

5. Solve the reduced Sylvester equation $H_j^{(A)}Y + Y\left(H_j^{(B)}\right)^T = \bar{R}$, where $\text{vec}(\bar{R}) = \bar{r}$ via the Bartels-Stewart algorithm (Algorithm 3.1).

6. Compute $x_0 = x_0 + (W_j \otimes V_j)y$ where $y = \text{vec}(Y)$ and test for convergence.

7. Compute $r = r - (W_j \otimes V_{j+1}\underline{H}_j^{(A)})y - (W_{j+1}\underline{H}_j^{(B)} \otimes V_j)y$.

end for

8. Define X such that $\text{vec}(X) = x_0$.

If the approximations are not good enough, the Galerkin method can be applied again with new vectors f and g. This process is usually called the restarted Galerkin algorithm. For the selection of the vectors f and g, a brief discussion is presented [53, Section 5]. An ideal option for f and g would be to choose them such that $r_0 \in \mathcal{K}(B^T, g) \otimes \mathcal{K}_m(A, f)$. However, such vectors are difficult to determine. Then, the vectors $f, g \in \mathbb{R}^n$ can be determined so that $\|r_0 - g \otimes f\|_2$ is small, or equivalently, if $\|R_0 - fg^T\|_F$ is small, where $r_0 = \text{vec}(R_0)$ with $R_0 \in \mathbb{R}^{n\times n}$.

If σ_{max} denotes the largest singular value of R_0, and let q_l and q_r be the associated left and right unit singular vectors, then

$$\min_{f,g\in\mathbb{R}^n}\|R_0 - fg^T\|_F = \|R_0 - \sigma_{max} q_l q_r^T\|_F. \tag{3.46}$$

However, the solution of (3.46) requires an extra cost that makes this solution unattractive. Then, an approximate solution is proposed in [53] for computing the vectors f and g. Let $f = [f_1, \ldots, f_n]^T$, $g = [g_1, \ldots, g_n]^T$ and assume that $f \neq 0$, $g \neq 0$. Introduce the functional

$$T(f, g) = \|R_0 - fg^T\|_F^2.$$

Then

$$\frac{\partial T}{\partial f_j} = 0 \quad \forall j \quad \Leftrightarrow \quad f = \frac{R_0 g}{\|g\|_2^2}, \tag{3.47}$$

$$\frac{\partial T}{\partial g_j} = 0 \quad \forall j \quad \Leftrightarrow \quad g = \frac{R_0^T f}{\|f\|_2^2}. \tag{3.48}$$

Therefore, given a vector g, f can be computed by (3.47) that minimizes $f \to T(f, g)$, or conversely, given a vector f, the vector g can be computed from (3.48) such that minimizes $g \to T(f, g)$. Thus, we can summarize that, if $\|R_0\|_1 \geq \|R_0\|_\infty$ then, choose f as the column with largest norm of R_0, and determine g from 3.48. If $\|R_0\|_1 \leq \|R_0\|_\infty$, choose g as the row with largest norm of R_0 and then, determine f by using (3.47).

Now, we discuss and compare the computational and memory costs of Algorithm 3.4 and 3.5, with $G := (A \oplus_S B)$ in Algorithm 3.4. In practice, the dimension n is usually much greater than the iteration number j, so when counting the flops number we may disregard those terms not containing n. First, it is important to remark that multiplications that involve Kronecker products of matrices can be performed without explicitly construct the matrix, for example, consider the matrix-vector multiplication

$$(A \oplus_S B)\mathbf{c}$$

where $(A \oplus_S B) \in \mathbb{C}^{n^2\times n^2}$ and $\mathbf{c} \in \mathbb{C}^{n^2}$ which can be computed as follows: If we partition $\mathbf{c}$ as $\mathbf{c} = [{c^{(1)}}^T, {c^{(2)}}^T, \ldots, {c^{(n)}}^T]^T$ with $c^{(i)} \in \mathbb{C}^n$, for $i = 1, 2, \ldots, n$ and we consider $B = [b_{ij}]_{1\leq i,j\leq n}$ then

$$\begin{aligned}(A \oplus_S B)\mathbf{c} &= (I_n \otimes A + B^T \otimes I_n)\mathbf{c} \\ &= \begin{bmatrix} Ac^{(1)} + b_{11}c^{(1)} + b_{21}c^{(2)} + \cdots + b_{n1}c^{(n)} \\ Ac^{(2)} + b_{12}c^{(1)} + b_{22}c^{(2)} + \cdots + b_{n2}c^{(n)} \\ \vdots \\ Ac^{(n)} + b_{1n}c^{(1)} + b_{2n}c^{(2)} + \cdots + b_{nn}c^{(n)} \end{bmatrix},\end{aligned}$$

and this matrix-vector multiplication can be computed with computational cost at most $\mathcal{O}(n^3)$ flops but much less if A and B are sparse. Anyway, it requires $2n$

matrix-vector products of size n. Since this multiplication appears in the first step in both algorithms, it is a common (and expensive) step and we do not consider it in the computational cost for comparative purposes.

We start with the computational costs for FOM in Algorithm 3.4. Step 2 costs $\mathcal{O}(n^3 + n^2 j)$ flops, where the most expensive step is the computation of $\hat{v}$ in the Arnoldi method, which produces the $\mathcal{O}(n^3)$ cost, but that it can be much less if A and B are sparse. Anyway, as explain above it, it requires $2n$ matrix vector products of size n. The cost of the solution of the linear system in step 3 is $\mathcal{O}(j^3)$ flops and is disposable. Finally, the cost for step 4 is $\mathcal{O}(n^2 j)$ flops. Therefore, the computational cost of the j-th iteration in Algorithm 3.4 is $\mathcal{O}(n^3 + n^2 j)$ flops.

The costs of Algorithm 3.5 are summarized in the following lines. For the computation of steps 2 and 3, $\mathcal{O}(n^2 + nj)$ flops are required for each one. Note that in this case, the computation of the vectors $\hat{v}$ and $\hat{w}$ in steps 2 and 3, respectively, involves matrix-vector multiplications of size n instead of n^2, and, therefore, it costs at most $\mathcal{O}(n^2)$ flops. The computation of $\overline{r}$ in step 4 requires $\mathcal{O}(n^2 j)$ flops. The cost of step 5 is $\mathcal{O}(j^3)$ which is negligible. Finally, for steps 7 and 8 the costs are $\mathcal{O}(n^2 j)$ flops, which results that the computational cost of the j-th iteration is $\mathcal{O}(n^2 + nj + n^2 j) \approx \mathcal{O}(n^2 j)$.

In terms of memory cost, the bigger cost for Algorithm 3.4 lies in storing the matrices $V_{j+1} \in \mathbb{C}^{n^2 \times (j+1)}$ and the vectors $x_0, r \in \mathbb{C}^{n^2}$. After j iterations, this requires to store $(j+1)n^2 + 2n^2$ numbers. For Algorithm 3.5, after j iterations we store the matrices $V_{j+1}, W_{j+1} \in \mathbb{C}^{n \times (j+1)}$ and the vectors $x_0, r \in \mathbb{C}^{n^2}$, which results in a storage cost of $2nj + 2n + 2n^2$ numbers.

In Table 3.1, the comparison of the costs between both algorithms is summarized. As we commented before, Algorithm 3.5 results in a memory saving method. Also, without considering the first step of both methods which has the same cost for both of them, the computational costs are considerably reduced in Algorithm 3.5 with respect to Algorithm 3.4.

	Algorithm 3.4	Algorithm 3.5
Computational cost	$j\times$ (the cost of $2n$ matrix-vector products of size n) $+ \mathcal{O}(n^2 j^2)$	$\mathcal{O}(n^2 j^2)$
Memory cost	$(j+1)n^2 + 2n^2$	$2nj + 2n + 2n^2$

Table 3.1: Computational and memory costs for Algorithms 3.4 and 3.5.

Despite of the relevant improvement of Algorithm 3.5 with respect to Algorithm 3.4, the memory cost of Algorithm 3.5 is very high and it cannot be used to solve modern large-scale problems. The key reason is the term $2n^2$ in the memory cost of Algorithm 3.5, that comes from the fact that Algorithm 3.5 stores the vectors x_0 and r, where $x_0 = \text{vec}(X_0)$ is a full-dense approximation to the solution X. In

modern large-scale problems, n is very large and a dense matrix of size $n \times n$ cannot be stored in the computer. This motivates the development of projection methods that approximate the true solution in terms of a number of parameters much less than n^2 for special Sylvester equations that are important in applications. A family of such projection methods is describe in the next subsection.

3.3.3 Projection methods for the Sylvester equation with a low rank right hand side

In this section, we consider the Sylvester equation

$$AX + XB = C_1 C_2^*, \tag{3.49}$$

where $A \in \mathbb{C}^{m\times m}$, $B \in \mathbb{C}^{n\times n}$ are large and sparse and $C := C_1 C_2^*$ has low rank with $C_1 \in \mathbb{C}^{m\times r}$, $C_2 \in \mathbb{C}^{n\times r}$ and $r = \text{rank}(C)$, with $r \ll \min\{m, n\}$.

An important observation is that, although A and B are sparse, $X \in \mathbb{C}^{m\times n}$ is dense, in general, and that for very large values of m and n, a dense matrix of size $m \times n$ cannot be stored in the computer. Then, the distribution of the singular values of X is a key factor for the development of iterative solution methods. Therefore, a Sylvester equation having solution with exponentially decaying singular values can be well approximated by a low rank matrix [90].

In order to solve (3.49), we introduce the *block Krylov* (BK) subspaces

$$\mathcal{K}_k(G, Z) = \text{span}\{Z, GZ, G^2Z, \ldots, G^{k-1}Z\} \tag{3.50}$$

where $G \in \mathbb{C}^{n\times n}$ and $Z \in \mathbb{C}^{n\times s}$. Note that the Arnoldi method can be straightforwardly adapted to compute an orthonormal basis for a block Krylov subspace $\mathcal{K}_k(G, Z)$ [86, Section 6.5].

Let $\mathcal{V}$ and $\mathcal{W}$ be two subspaces of $\mathbb{C}^m$ and $\mathbb{C}^n$ respectively, and let the columns of $V_k \in \mathbb{C}^{m\times k}$, $W_j \in \mathbb{C}^{n\times j}$ be orthonormal bases of $\mathcal{V}$ and $\mathcal{W}$ respectively, this is,

$$\mathcal{V} = \text{range}(V_k), \quad \mathcal{W} = \text{range}(W_j), \tag{3.51}$$

such that $\mathcal{V}$ is not orthogonal to range(C_1) and $\mathcal{W}$ is not orthogonal to range(C_2). It is well-known [89, 90] that projection methods for Sylvester equations with low-rank right-hand side look for an approximation

$$\tilde{X} = V_k Y W_j^* \approx X \tag{3.52}$$

for a certain matrix $Y \in \mathbb{C}^{k\times j}$ with an associated residual

$$R := C_1 C_2^* - A\tilde{X} - \tilde{X}B. \tag{3.53}$$

Observe that $\tilde{X}$ is represented in terms of $nk + mj + kj$ numbers, and that $nk + mj + kj \ll mn$ if $\max\{k, j\} \ll \min\{m, n\}$. Then, we have

$$\tilde{x} = \text{vec}(\tilde{X}) = (W_j \otimes V_k)\text{vec}(Y), \tag{3.54}$$

where $\tilde{x}$ is an approximate solution of the equivalent linear system (3.5). If a Galerkin condition is imposed to the vector residual $\text{vec}(C) - (A \oplus_S B)\tilde{x}$ in (3.5) with respect to the space spanned by $W_j \otimes V_k$ we have

$$(W_j \otimes V_k)^*(\text{vec}(C) - (A \oplus_S B)\tilde{x}) = 0 \quad \Leftrightarrow \quad V_k^* R W_j = 0. \tag{3.55}$$

If we substitute the residual matrix (3.53) in (3.55), we obtain the small size Sylvester equation

$$V_k^* A V_k Y + Y W_j^* B W_j = V_k^* C_1 (W_j^* C_2)^*. \tag{3.56}$$

The development of the algorithm and the computations of the matrices depend on the selection of the subspaces $\mathcal{V}$ and $\mathcal{W}$. For example, block Krylov subspaces can be chosen, with $\mathcal{V} = \mathcal{K}_k(A, C_1)$ and $\mathcal{W} = \mathcal{K}_k(B^*, C_2)$ or an attractive choice are *extended* Krylov (EK) subspaces, defined as

$$\mathcal{EK}_k(G, Z) = \mathcal{K}_k(G, Z) + \mathcal{K}_k(G^{-1}, G^{-1}Z), \tag{3.57}$$

where $G \in \mathbb{C}^{n\times n}$ is an invertible matrix and $Z \in \mathbb{C}^{n\times s}$. Then, we can choose $\mathcal{V} = \mathcal{EK}_k(A, C_1)$ and $\mathcal{W} = \mathcal{EK}_k(B^*, C_2)$ if we are interested into use extended Krylov subspaces. A very brief summary of this process with $\mathcal{V}$ and $\mathcal{W}$ chosen as block Krylov subspaces is presented in Algorithm 3.6. Note that in Algorithm 3.6 the notation $\mathbf{V}_{j+1} = [V_1, V_2, \ldots, V_{j+1}] = [\mathbf{V}_j, V_{j+1}]$ and $\mathbf{W}_{j+1} = [W_1, W_2, \ldots, W_{j+1}] = [\mathbf{W}_j, W_{j+1}]$ is used since now we are orthogonalizing blocks of vectors instead of a single one at each iteration.

A detailed discussion of the selection of different subspaces is presented in [90]. Also, several other methods to compute a numerical solution of the Sylvester equations, as the ADI iteration, are discussed in [90].

In terms of memory cost, Algorithm 3.6 results in a memory saving procedure. We consider the case $m = n$ to compare the memory costs of Algorithms 3.4, 3.5 and 3.6. Note that for Algorithm 3.6, after j iterations, only two matrices of size $n(j+1)r$, where r is the rank of the right-hand side C, need to be stored, and one of size $j \times j$. As occurs in large-scale problems, $n \gg j$, thus, roughly $2n(j+1)r$ numbers need to be stored, where usually $r < j$ in applications. The memory cost for each method is summarized in Table 3.2. Note that the approximate solution obtained by Algorithm 3.6 depends on a number of parameters much less than n^2, which is a very considerable improvement with respect to Algorithm 3.5 when n is large.

	Algorithm 3.4	Algorithm 3.5	Algorithm 3.6
Memory cost	$(j+1)n^2 + 2n^2$	$2n(j+1) + 2n^2$	$2n(j+1)r$

Table 3.2: Memory costs for Algorithms 3.4, 3.5 and 3.6 after j iterations.

Algorithm 3.6 Block Krylov method for the Sylvester equation with low-rank right-hand side

Input: $A \in \mathbb{C}^{m\times m}, B \in \mathbb{C}^{n\times n}$, $C_1 \in \mathbb{C}^{m\times r}$, $C_2 \in \mathbb{C}^{n\times r}$ such that the Sylvester equation $AX + XB = C_1C_2^*$ has a unique solution.

Output: $\mathbf{V}_k$, Y_k and $\mathbf{W}_k$ such that $X_k = \mathbf{V}_k Y_k \mathbf{W}_k^*$ is an approximate solution of the Sylvester equation $AX + XB = C_1C_2^*$.

1. Orthogonalize the columns of C_1 obtaining $V_1 \in \mathbb{C}^{m\times r}$ and set $\mathbf{V}_1 = [V_1]$.
2. Orthogonalize the columns of C_2 obtaining $W_1 \in \mathbb{C}^{n\times r}$ and set $\mathbf{W}_1 = [W_1]$.

for $k = 1, 2, \ldots, l$ **do**

3. Compute Y_k, solution of

$$(\mathbf{V}_k^* A \mathbf{V}_k)Y_k + Y_k(\mathbf{W}_k^* B \mathbf{W}_k) - \mathbf{V}_k^* C_1 C_2^* \mathbf{W}_k = 0.$$

via the Bartels-Stewart algorithm and test for convergence.

4. Compute $\hat{V}$, $\hat{W}$ for the chosen approximate space, where $\hat{V}$, $\hat{W}$ are block matrices obtained by computing a matrix-matrix multiplication similar to step 1 of Algorithm 2.1 where v_j is a block matrix instead of a vector.

5. Orthogonalize $\hat{V}$ and $\hat{W}$ with respect to $\{V_1, V_2, \ldots, V_k\}$ and $\{W_1, W_2, \ldots, W_k\}$ respectively.

6. Orthogonalize the columns of $\hat{V}$ to get V_{k+1} and columns of $\hat{W}$ to get W_{k+1}.

7. Update $\mathbf{V}_{k+1} = [\mathbf{V}_k \quad V_{k+1}]$, $\mathbf{W}_{k+1} = [\mathbf{W}_k \quad W_{k+1}]$.

end for

In Chapter 5, we present a novel projection method to solve the Sylvester equation for T-congruence with low-rank right-hand side. This method presents some similarities with the Krylov methods presented in this section, however, the use of a Galerkin condition is changed by a Petrov-Galerkin condition given the relation between the matrices A and B in the T-Sylvester equation.

Chapter 4

The R-CORK method

In this chapter, inspired in the methods CORK and TOAR for PEPs explained in Sections 2.4.3 and 2.4.2, we develop the new method R-CORK for the solution of large-scale and sparse rational eigenvalue problems. We use the compact rational Krylov decomposition (2.25) expressed in a compact form (as in (2.112)) for the bases of the rational Krylov subspaces associated to the linearization (2.10) of the REP corresponding to the rational matrix (2.9) and we work in the spirit of the two levels of orthogonalization introduced in [100, 71]. We also discuss the advantages of the R-CORK method in terms of memory and orthogonalization costs which are similar to the costs for the CORK method.

The R-CORK method is based on the rational Krylov method, and it follows the steps introduced in Algorithm 2.4, but exploiting the structure of both the linearization (2.10) and the Krylov vectors.

From now on, we change slightly the notation used in Algorithm 2.4, we denote the Krylov vectors u_j as $\mathbf{u}_j$, this is, we use bold characters and the matrix U_j that contains the Krylov vectors in its columns as $\mathbf{U}_j$, basically because now they represent block vectors and block matrices, respectively. Also, for simplicity, we will consider the continuation vector z_j in step 2 in Algorithm 2.4 as the canonical vector $e_j \in \mathbb{C}^j$ and then, in step 3 in Algorithm 2.4, we have $\mathbf{w}_j = \mathbf{u}_j$ and

$$\hat{\mathbf{u}} = (\mathcal{A} - \theta_j \mathcal{B})^{-1} \mathcal{B} \mathbf{u}_j.$$

The results introduced in Chapter 4 are, as far as we know, novel results that have been accepted for publication and that they appear in [34].

4.1 A compact decomposition for bases of rational Krylov subspaces of $\mathcal{A} - \lambda\mathcal{B}$

Recall the matrices $\mathcal{A}$ and $\mathcal{B}$ introduced in (2.11)

$$\mathcal{A} = \left[\begin{array}{cccc|c} P_0 & P_1 & \cdots & P_{d-1} & E \\ 0 & \cdots & 0 & -I_n & \\ \vdots & \iddots & \iddots & & \\ 0 & -I_n & & & \\ \hline F^T & & & & C \end{array}\right], \quad \mathcal{B} = -\left[\begin{array}{cccc|c} & & & P_d & \\ & & I_n & & \\ & \iddots & & & \\ I_n & & & & \\ \hline & & & & -D \end{array}\right]$$

to linearize the REP written as (2.9) and the rational Krylov recurrence relation (2.25) which is valid for arbitrary pencils.

In this section, we particularize such relation to the matrices $\mathcal{A}$ and $\mathcal{B}$ in (2.11) in order to save memory and orthogonalization costs. For this purpose, we consider a partition of $\mathbf{U}_{j+1}$ conformable to $\mathcal{A}$ and $\mathcal{B}$ as follows

$$\mathbf{U}_{j+1} = [\mathbf{U}_j \quad \mathbf{u}_{j+1}] = \begin{bmatrix} U_j^{(1)} & u_{j+1}^{(1)} \\ U_j^{(2)} & u_{j+1}^{(2)} \\ \vdots & \vdots \\ U_j^{(d)} & u_{j+1}^{(d)} \\ V_j & v_{j+1} \end{bmatrix} \tag{4.1}$$

where $U_j^{(i)} \in \mathbb{C}^{n\times j}$, $u_{j+1}^{(i)} \in \mathbb{C}^n$, for $i = 1, \ldots, d$, $V_j \in \mathbb{C}^{s\times j}$, and $v_{j+1} \in \mathbb{C}^s$. Next, as in CORK for the first d blocks, we define the matrix $Q_j \in \mathbb{C}^{n\times r_j}$ such that the columns of Q_j are orthonormal with

$$\operatorname{span}\{Q_j\} = \operatorname{span}\{U_j^{(1)}, U_j^{(2)}, \ldots, U_j^{(d)}\} \tag{4.2}$$

and $\operatorname{rank}(Q_j) = r_j$. Using (4.2) we can express

$$U_j^{(i)} = Q_j R_j^{(i)}, \quad i = 1, 2, \ldots, d, \tag{4.3}$$

where $R_j^{(i)} \in \mathbb{C}^{r_j\times j}$ for $i = 1, 2, \ldots, d$. Then, by using (4.3), we have

$$\mathbf{U}_j = \begin{bmatrix} Q_j R_j^{(1)} \\ Q_j R_j^{(2)} \\ \vdots \\ Q_j R_j^{(d)} \\ V_j \end{bmatrix} = \begin{bmatrix} Q_j & & & & \\ & Q_j & & & \\ & & \ddots & & \\ & & & Q_j & \\ & & & & I_s \end{bmatrix} \begin{bmatrix} R_j^{(1)} \\ R_j^{(2)} \\ \vdots \\ R_j^{(d)} \\ V_j \end{bmatrix}. \tag{4.4}$$

By introducing the notation

$$\mathbf{Q}_j := \left[\begin{array}{c|c} (I_d \otimes Q_j) & 0_{dn\times s} \\ \hline 0_{s\times dr_j} & I_s \end{array}\right] \in \mathbb{C}^{(dn+s)\times(dr_j+s)} \quad \text{and} \quad \mathbf{R}_j := \left[\begin{array}{c} R_j^{(1)} \\ R_j^{(2)} \\ \vdots \\ R_j^{(d)} \\ V_j \end{array}\right] \in \mathbb{C}^{(dr_j+s)\times j}, \tag{4.5}$$

we have $\mathbf{U}_j = \mathbf{Q}_j\mathbf{R}_j$. Since the columns of $\mathbf{U}_j$ and $\mathbf{Q}_j$ are orthonormal, the matrix $\mathbf{R}_j$ has orthonormal columns too. With this notation, the rational Krylov decomposition (2.25) can be written as the following compact rational Krylov recurrence relation

$$\mathcal{A}\mathbf{Q}_{j+1}\mathbf{R}_{j+1}\underline{H}_j = \mathcal{B}\mathbf{Q}_{j+1}\mathbf{R}_{j+1}\underline{K}_j. \tag{4.6}$$

In order to prove that, as in CORK, we need only one vector to expand Q_j into Q_{j+1} and that, as a consequence, r_j is considerably smaller than jd, i.e., that $\mathbf{Q}_j\mathbf{R}_j$ is indeed a compact representation of $\mathbf{U}_j$, we will prove first Lemmas 4.1 and 4.3. We emphasize the relationship between Lemma 4.1 and Theorem 2.15, but also two differences: the first one is coming from the presence of the strictly proper part $E(C-\lambda D)^{-1}F^T$ of the rational matrix $R(\lambda)$, which motivates the definition of the rational matrix $\mathbf{A}(\lambda)$ in Lemma 4.1, and the second one which is related with the matrices $\mathcal{A}$ and $\mathcal{B}$ in (2.11). Since the matrices P_i are ordered in increasing index order in (2.11), it occurs that the permutation $\mathcal{P}$ in Theorem 2.15 is not needed in Lemma 4.1, therefore, we developed a UL decomposition instead of a ULP decomposition. Apart from these differences, we have stated Lemma 4.1 in an analogous way to Theorem 2.15, with the purpose of stressing the relation with CORK, but note that the simple particular structures of M, $N \in \mathbb{C}^{(d-1)\times d}$, and $\mathbf{B}$ inherited from (2.11)-(2.13) imply that in Lemma 4.1

$$M := [\ m_0 \mid M_1\ ] = \left[\begin{array}{c|ccc} 0 & & & -1 \\ \vdots & & \iddots & \\ 0 & -1 & & \end{array}\right], \quad N := [\ n_0 \mid N_1\ ] = \left[\begin{array}{c|ccc} & & -1 & 0 \\ & \iddots & & \vdots \\ -1 & & & 0 \end{array}\right] \tag{4.7}$$

and that $\bar{\mathbf{B}}_1$ has only one nonzero block. Therefore, the factors $\mathcal{L}(\mu)$ and $\mathcal{U}(\mu)$ in Lemma 4.1 are simpler than the general ones in Theorem 2.15. Note also that Lemma 4.3 is related to [104, Lemma 4.3], although again the strictly proper part of the rational matrix introduces relevant differences.

Lemma 4.1. *Consider a rational matrix*

$$R(\lambda) = P(\lambda) - E(C-\lambda D)^{-1}F^T \in \mathbb{C}(\lambda)^{n\times n},$$

where $P(\lambda) = \sum_{i=0}^{d}\lambda^i P_i$, $P_i \in \mathbb{C}^{n\times n}$ *for* $i = 0,\ldots,d$, E, $F \in \mathbb{C}^{n\times s}$, C, $D \in \mathbb{C}^{s\times s}$, D *is nonsingular, and* $E(C-\lambda D)^{-1}F^T$ *is a minimal realization. Define the rational*

matrix

$$\mathbf{A}(\lambda) = \left[\begin{array}{cccc|c} P_0 - E(C-\lambda D)^{-1}F^T & P_1 & \cdots & P_{d-2} & P_{d-1} \\ \hline \multicolumn{5}{c}{M \otimes I_n} \end{array}\right],$$

and the constant matrix

$$\mathbf{B} = \left[\begin{array}{cccc|c} 0_n & 0_n & \cdots & 0_n & -P_d \\ \hline \multicolumn{5}{c}{N \otimes I_n} \end{array}\right],$$

with M and N defined as in (4.7)*. Then, for every $\mu \in \mathbb{C}$ which is not a pole of $R(\mu)$, i.e., such that $(C-\mu D)$ is nonsingular, we can factorize $\mathbf{A}(\mu)-\mu\mathbf{B}$ as follows*

$$\mathbf{A}(\mu) - \mu\mathbf{B} = \mathcal{U}(\mu)\mathcal{L}(\mu), \tag{4.8}$$

where

$$\begin{aligned} \mathcal{L}(\mu) &= \begin{bmatrix} R(\mu) & 0 \\ (m_0 - \mu n_0) \otimes I_n & (M_1 - \mu N_1) \otimes I_n \end{bmatrix}, \\ \mathcal{U}(\mu) &= \begin{bmatrix} I_n & (\bar{\mathbf{A}}_1 - \mu\bar{\mathbf{B}}_1)((M_1 - \mu N_1)^{-1} \otimes I_n) \\ 0 & I_{(d-1)n} \end{bmatrix}, \end{aligned}$$

and

$$\begin{aligned} [P_0 - E(C-\mu D)^{-1}F^T \quad P_1 \quad \cdots \quad P_{d-1}] &=: [P_0 - E(C-\mu D)^{-1}F^T \quad \bar{\mathbf{A}}_1], \\ [0_n \quad 0_n \quad \cdots \quad -P_d] &=: [0_n \quad \bar{\mathbf{B}}_1]. \end{aligned}$$

Proof. Observe first that the definitions of M and N imply that $M_1 - \mu N_1$ is invertible for every $\mu \in \mathbb{C}$, therefore $M_1 - \mu N_1$ is nonsingular for every $\mu \in \mathbb{C}$. By a direct matrix multiplication, we obtain

$$\begin{aligned} \mathcal{U}(\mu)\mathcal{L}(\mu) &= \begin{bmatrix} R(\mu) + (\bar{\mathbf{A}}_1 - \mu\bar{\mathbf{B}}_1)(((M_1 - \mu N_1)^{-1}(m_0 - \mu n_0)) \otimes I_n) & (\bar{\mathbf{A}}_1 - \mu\bar{\mathbf{B}}_1) \\ (m_0 - \mu n_0) \otimes I_n & (M_1 - \mu N_1) \otimes I_n \end{bmatrix} \\ &= \left[\begin{array}{c} R(\mu) + (\bar{\mathbf{A}}_1 - \mu\bar{\mathbf{B}}_1)(((M_1 - \mu N_1)^{-1}(m_0 - \mu n_0)) \otimes I_n) \quad (\bar{\mathbf{A}}_1 - \mu\bar{\mathbf{B}}_1) \\ \hline (M - \mu N) \otimes I_n \end{array}\right]. \end{aligned}$$

Therefore, we only need to prove that

$$R(\mu) + (\bar{\mathbf{A}}_1 - \mu\bar{\mathbf{B}}_1)(((M_1 - \mu N_1)^{-1}(m_0 - \mu n_0)) \otimes I_n) = P_0 - E(C-\mu D)^{-1}F^T,$$

which is equivalent to prove that

$$P(\mu) + (\bar{\mathbf{A}}_1 - \mu\bar{\mathbf{B}}_1)(((M_1 - \mu N_1)^{-1}(m_0 - \mu n_0)) \otimes I_n) = P_0. \tag{4.9}$$

The proof of (4.9) is a very simple algebraic manipulation as a consequence of the extremely simple structures of m_0 and n_0, M_1 and N_1 in this case. Another proof comes from the observation that (4.9) holds because it is proved for proving the ULP decomposition in Theorem 2.15 (see [104, pp. 823-824]). □

Remark 4.2. *Observe that Theorem 2.15 involves the constant $\alpha = e_1^T \mathcal{P} f(\mu)$, which is not present in Lemma 4.1. The reason is that in Lemma 4.1, this constant is equal to 1 as a consequence of the structure of (2.12) and that $\mathcal{P} = I_d$.*

Lemma 4.3. *Let $\mathcal{A}$ and $\mathcal{B}$ be the matrices defined in (2.11). Consider the linear system*

$$(\mathcal{A} - \mu\mathcal{B})\mathbf{x} = \mathcal{B}\mathbf{w}, \tag{4.10}$$

where $\mathbf{x} = [x^{(1)^T}, x^{(2)^T}, \cdots, x^{(d)^T}, y^T]^T$ and $\mathbf{w} = [w^{(1)^T}, w^{(2)^T}, \cdots, w^{(d)^T}, z^T]^T$, the blocks $x^{(i)}, w^{(i)} \in \mathbb{C}^n$, $i = 1, 2, \ldots, d$, $y, z \in \mathbb{C}^s$, and μ is not a pole of $R(\mu)$, i.e., $(C - \mu D)$ is nonsingular. Then, the block $x^{(1)}$ of $\mathbf{x}$ can be computed by solving the following $n \times n$ linear system whose coefficient matrix is $R(\mu)$ in (2.9):

$$R(\mu)x^{(1)} = -P_d w^{(d)} - E(C - \mu D)^{-1} D z + (\bar{\mathbf{A}}_1 - \mu \bar{\mathbf{B}}_1)((M_1 - \mu N_1)^{-1} \otimes I_n) w^{(d-1,\ldots,1)},$$

where the matrices introduced in Lemma 4.1 are used and

$$w^{(d-1,\ldots,1)} = [w^{(d-1)^T}, \cdots, w^{(1)^T}]^T.$$

The remaining blocks $x^{(i)}$ for $i = 2, \cdots, d$ of $\mathbf{x}$ can be obtained as linear combinations of $x^{(1)}$ and $w^{(i)}$, $i = 1, \ldots, d-1$. Also, $x^{(2,\ldots,d)} = [x^{(2)^T}, \cdots, x^{(d)^T}]^T$ satisfies the linear system

$$x^{(2,\ldots,d)} = -((M_1 - \mu N_1)^{-1} \otimes I_n) \left(w^{(d-1,\ldots,1)} + ((m_0 - \mu n_0) \otimes I_n) x^{(1)} \right).$$

In addition, y can be computed by solving the $s \times s$ linear system

$$(C - \mu D) y = D z - F^T x^{(1)}.$$

Proof. Rewrite the matrix pencil (2.10) as in (2.13)

$$\mathcal{A} - \mu\mathcal{B} = \left[\begin{array}{c|c} \mathbf{A} - \mu\mathbf{B} & e_1 \otimes E \\ \hline e_1^T \otimes F^T & C - \mu D \end{array}\right]$$

with

$$\mathbf{A} = \left[\begin{array}{cccc} P_0 & \cdots & P_{d-2} & P_{d-1} \\ \hline \multicolumn{4}{c}{M \otimes I_n} \end{array}\right], \quad \mathbf{B} = \left[\begin{array}{cccc} 0_n & \cdots & 0_n & -P_d \\ \hline \multicolumn{4}{c}{N \otimes I_n} \end{array}\right],$$

and M and N defined as in (4.7). Then, we can solve the system (4.10) by solving

$$\begin{aligned} (\mathbf{A} - \mu\mathbf{B}) x^{(1,2,\ldots,d)} + (e_1 \otimes E) y &= \mathbf{B} w^{(1,2,\ldots,d)}, & (4.11) \\ (e_1^T \otimes F^T) x^{(1,2,\ldots,d)} + (C - \mu D) y &= D z, & (4.12) \end{aligned}$$

where $x^{(1,2,\ldots,d)} = [x^{(1)^T}, x^{(2)^T}, \cdots, x^{(d)^T}]^T$ and $w^{(1,2,\ldots,d)} = [w^{(1)^T}, w^{(2)^T}, \cdots, w^{(d)^T}]^T$. The second equation is the equation for y in the statement. By replacing $y =$

$(C-\mu D)^{-1}(Dz - F^T x^{(1)})$ from (4.12) in (4.11), and using the notation of Lemma 4.1, we obtain

$$(\mathbf{A}-\mu\mathbf{B})x^{(1,2,\ldots,d)} - (e_1 \otimes E)(C-\mu D)^{-1}F^T x^{(1)} = \mathbf{B}w^{(1,2,\ldots,d)} - (e_1 \otimes E)(C-\mu D)^{-1}Dz,$$

$$\left[\frac{P_0 - E(C-\mu D)^{-1}F^T \quad \cdots \quad P_{d-1}+\mu P_d}{(M-\mu N)\otimes I_n}\right] x^{(1,2,\ldots,d)} = -\left[\frac{P_d w^{(d)} + E(C-\mu D)^{-1}Dz}{w^{(d-1,\ldots,1)}}\right],$$

$$(\mathbf{A}(\mu)-\mu\mathbf{B})x^{(1,2,\ldots,d)} = -\left[\frac{P_d w^{(d)} + E(C-\mu D)^{-1}Dz}{w^{(d-1,\ldots,1)}}\right].$$

By combining the factorization (4.8) in Lemma 4.1 and the equation above, it is immediate to see that the blocks $x^{(i)}$ for $i = 2, \ldots, d$ of $\mathbf{x}$ are linear combinations of $x^{(1)}$ and the blocks $w^{(i)}$, $i = 1, \ldots, d-1$. In addition, some elementary matrix manipulations with the matrices $\mathcal{U}(\mu)$ and $\mathcal{L}(\mu)$ in (4.8) lead to the equations for $x^{(2,\ldots,d)}$ and $x^{(1)}$ in the statement. This finishes the proof. □

As we mentioned before, Lemma 4.3 is the key result that allows us to prove through Theorems 4.5 and 4.6 that only one vector is needed to expand Q_j into Q_{j+1} and, so, that the representation (4.4) for $\mathbf{U}_j$ is indeed compact. Moreover, the equations for $x^{(1)}$, $x^{(2,\ldots,d)}$, and y deduced in Lemma 4.3 lead to the efficient Algorithm 4.1 for solving the linear system (4.10), which is fundamental for performing efficiently the shift-and-invert step in the R-CORK method developed in Section 4.2. Observe that in Algorithm 4.1 a notation similar to that in Lemma 4.3 is used.

Algorithm 4.1 Solver for the linear system $(\mathcal{A}-\mu\mathcal{B})\mathbf{x} = \mathcal{B}\mathbf{w}$, with $\mathcal{A}$ and $\mathcal{B}$ as in (2.11)

Input: $\mathcal{A}, \mathcal{B} \in \mathbb{C}^{(nd+s)\times(nd+s)}$ as in (2.11), $\mu \in \mathbb{C}$ such that $(C-\mu D)^{-1}$ exists and $\mathbf{w} \in \mathbb{C}^{nd+s}$.

Output: The solution $\mathbf{x} = [x^{(1)^T}, \ldots, x^{(d)^T}, y^T]^T \in \mathbb{C}^{nd+s}$ of the linear system.

1. Compute $x = \mathbf{B}w^{(1,2,\ldots,d)} - (e_1 \otimes E)(C-\mu D)^{-1}Dz$ as

$$x = -\left[\frac{P_d w^{(d)} + E(C-\mu D)^{-1}Dz}{w^{(d-1,\ldots,1)}}\right].$$

Solve the block upper triangular system associated to $\mathcal{U}(\mu)$ in (4.8):

2. $x^{(1)} = x^{(1)} - (\bar{\mathbf{A}}_1 - \mu\bar{\mathbf{B}}_1)((M_1-\mu N_1)^{-1} \otimes I_n)x^{(2,\ldots,d)}$.

Solve the block lower triangular system associated to $\mathcal{L}(\mu)$ in (4.8):

3. $x^{(1)} = (R(\mu))^{-1}x^{(1)}$.
4. $x^{(2,\ldots,d)} = ((M_1-\mu N_1)^{-1} \otimes I_n)(x^{(2,\ldots,d)} - ((m_0-\mu n_0)\otimes I_n)x^{(1)})$.

Compute the block y of $\mathbf{x}$

5. $y = (C-\mu D)^{-1}(Dz - F^T x^{(1)})$.

Remark 4.4. *The multiplications by inverses in Algorithm 4.1 have to be understood, in principle, in a similar way than for previous algorithms, this is, as solutions of linear systems and the key observation on Algorithm 4.1 is that all the involved linear systems have sizes smaller than the size* $(nd+s)\times(nd+s)$ *of* $(\mathcal{A}-\mu\mathcal{B})$ *as we discuss in this remark. The only linear system which is always large is the one in step 3 involving* $R(\mu)$ *which has the size* $n\times n$ *of the original REP. Solving the system in step 3 may require just the ability of multiplying by* $R(\mu)$*, if an iterative Krylov method is used, which might be done through the coefficients of* $P(\lambda)$ *and the matrices* E, C, D, F *in* (2.9) *without computing* $R(\mu)$*, or may require to compute* $R(\mu)$*, if a direct method is used. In the case* $(C-\mu D)$ *is large and complicated the computation of* $R(\mu)$ *might be performed more efficiently through* (2.8) *than through* (2.9)*, though this depends on each particular problem. However, we emphasize once again that the matrix* $(C-\mu D)\in\mathbb{C}^{s\times s}$ *is in many applications [76, 99] very small as we commented in Chapter 2, since* $s\ll n$*, and has in addition a very simple structure, which imply that it is often possible just to compute* $(C-\mu D)^{-1}$ *and to perform the corresponding matrix multiplications to construct* $R(\mu)$ *through* (2.9)*. These comments on the size* $s\ll n$ *also apply to the linear systems involving* $(C-\mu D)\in\mathbb{C}^{s\times s}$ *in steps 1 and 5 which are often in practice very small. Finally, the linear systems involving* $(M_1-\mu N_1)\otimes I_n$ *have size* $(d-1)n\times(d-1)n$ *and look very large, but they are block linear systems very easy to solve with cost* $2n(d-2)$ *flops by using a simple two term recurrence relation. More precisely, the solution of* $((M_1-\mu N_1)\otimes I_n)\mathbf{x}=\mathbf{b}$*, taking into account that*

$$M_1-\mu N_1=\begin{bmatrix} & & \mu & -1\\ & \iddots & \iddots & \\ & \iddots & & \\ \mu & \iddots & & \\ -1 & & & \end{bmatrix}\in\mathbb{C}^{(d-1)\times(d-1)},$$

and partitioning the vectors in $(d-1)$ *blocks of size* $n\times 1$*, can be obtained as* $x^{(1)}=-b^{(d-1)}$ *and* $x^{(i)}=\mu\,x^{(i-1)}-b^{(d-i)}$ *for* $i=2,3,\ldots,d-1$*.*

Theorems 4.5 and 4.6 are similar to results obtained in [104, Theorems 4.4 and 4.5].

Theorem 4.5. *Let* Q_j *be defined as in* (4.2)*. Then,*

$$span\{Q_{j+1}\}=span\{Q_j,u_{j+1}^{(1)}\}. \tag{4.13}$$

Proof. From the definition of Q_j in (4.2) we have

$$\begin{aligned}\text{span}\{Q_{j+1}\} &= \text{span}\{U_{j+1}^{(1)},\ldots,U_{j+1}^{(d)}\}\\ &= \text{span}\{U_j^{(1)},\ldots,U_j^{(d)},u_{j+1}^{(1)},\ldots,u_{j+1}^{(d)}\}\end{aligned}$$

$$= \text{span}\{Q_j, u_{j+1}^{(1)}, \ldots, u_{j+1}^{(d)}\}, \tag{4.14}$$

and from Algorithm 2.4, with $z_j = e_j \in \mathbb{R}^j$, by combining steps 3, 5 and 6, we obtain

$$\mathbf{u}_{j+1} = \frac{1}{h_{j+1,j}} \left(\hat{\mathbf{u}} - \mathbf{U}_j h_j\right).$$

Then,

$$u_{j+1}^{(i)} = \frac{1}{h_{j+1,j}} \left(\hat{u}^{(i)} - U_j^{(i)} h_j\right), \quad \text{for } i = 1, \ldots, d.$$

By replacing the latter expression in (4.14), we have

$$\text{span}\{Q_{j+1}\} = \text{span}\{Q_j, \hat{u}^{(1)}, \ldots, \hat{u}^{(d)}\}.$$

Since $(\mathcal{A} - \theta_j \mathcal{B})\hat{\mathbf{u}} = \mathcal{B}\mathbf{u_j}$, by applying Lemma 4.3 with $\mu = \theta_j$ and $\mathbf{w} = \mathbf{u}_j$ it follows

$$\begin{aligned} \text{span}\{Q_{j+1}\} &= \text{span}\{Q_j, \hat{u}^{(1)}, \ldots, \hat{u}^{(d)}\} \\ &= \text{span}\{Q_j, \hat{u}^{(1)}\} \\ &= \text{span}\{Q_j, u_{j+1}^{(1)}\}. \end{aligned}$$

□

Theorem 4.6. *Let Q_j be defined as in (4.2). Then*

$$r_j < d + j. \tag{4.15}$$

Proof. We will prove this theorem by induction. From the definition of Q_j in (4.2), we have that

$$\text{span}\{Q_1\} = \text{span}\{u_1^{(1)}, u_1^{(2)}, \ldots, u_1^{(d)}\},$$

so $r_1 \leq d$. Assuming that the inequality (4.15) is satisfied until $j-1$, then we have by Theorem 4.5 that $r_j \leq r_{j-1} + 1 < d + j$. □

From the fact that r_j increases at most by 1 in each iteration and by considering the inequality (4.15), we show in Lemma 4.7 the possible structures of the expansion of the first d blocks of the matrix $\mathbf{R}_j$ defined in (4.5).

Lemma 4.7. *Let $\mathbf{R}_j \in \mathbb{C}^{(dr_j+s)\times j}$ be defined as in (4.5). Then, the first d blocks of the matrix $\mathbf{R}_{j+1} \in \mathbb{C}^{(dr_{j+1}+s)\times(j+1)}$ can take the following forms:*

- *if $r_{j+1} > r_j$*

$$R_{j+1}^{(i)} = \begin{bmatrix} R_j^{(i)} & r_{j+1}^{(i)} \\ 0_{1\times j} & \end{bmatrix}, \quad i = 1, 2, \ldots, d,$$

  where $r_{j+1}^{(i)} \in \mathbb{C}^{r_{j+1}}$, or

- *if $r_{j+1} = r_j$*

$$R_{j+1}^{(i)} = \begin{bmatrix} R_j^{(i)} & r_{j+1}^{(i)} \end{bmatrix}, \quad i = 1, 2, \ldots, d,$$

  with $r_{j+1}^{(i)} \in \mathbb{C}^{r_{j+1}}$.

4.2 The two levels of orthogonalization

In this section, we will introduce the method to solve large-scale and sparse rational eigenvalue problems based on the compact representation presented in Section 4.1 of the orthonormal bases of the rational Krylov subspaces of the linearization $\mathcal{A} - \lambda\mathcal{B}$ in (2.10). First, we consider an initial vector $\mathbf{u}_1 \in \mathbb{C}^{nd+s}$ with $\|\mathbf{u}_1\|_2 = 1$ partitioned as in (4.1) and then, we express that vector in the compact form:

$$\mathbf{u}_1 = \begin{bmatrix} u_1^{(1)} \\ \vdots \\ u_1^{(d)} \\ v_1 \end{bmatrix} = \begin{bmatrix} Q_1 R_1^{(1)} \\ \vdots \\ Q_1 R_1^{(d)} \\ v_1 \end{bmatrix}$$

where $Q_1 \in \mathbb{C}^{n\times r_1}$ has orthonormal columns such that

$$\operatorname{span}\{Q_1\} = \operatorname{span}\left\{u_1^{(1)}, \cdots, u_1^{(d)}\right\}, \quad r_1 = \operatorname{rank}([u_1^{(1)} \cdots u_1^{(d)}]).$$

Observe that $r_1 = 1$ if and only if $\mathbf{u}_1$ is chosen to have collinear nonzero blocks $u_1^{(1)}, \ldots, u_1^{(d)}$. Now, taking into account the definition of $\mathbf{R}_j$ in (4.5), after j steps the matrices Q_j and $\mathbf{R}_j$ have to be expanded into Q_{j+1} and $\mathbf{R}_{j+1}$ respectively, which results in the so-called two levels of orthogonalization.

First level of orthogonalization. In Theorem 4.5 we have proved that we need to orthogonalize $u_{j+1}^{(1)}$ with respect to Q_j to obtain the last orthonormal column of Q_{j+1}. In addition, it can be easily seen that

$$\operatorname{span}\{Q_{j+1}\} = \operatorname{span}\{Q_j, u_{j+1}^{(1)}\} = \operatorname{span}\{Q_j, \hat{u}^{(1)}\}, \tag{4.16}$$

where $\hat{u}^{(1)}$ is the first block of size n of the vector $\hat{\mathbf{u}}$ obtained by applying the shift-and-invert step to $\mathbf{u}_j$ (step 3 in Algorithm 2.4) when $\hat{\mathbf{u}}$ is partitioned as in (4.1). Therefore, we only need to compute the block $\hat{u}^{(1)}$ of $\hat{\mathbf{u}}$ to compute Q_{j+1}. Thus, we can run Algorithm 4.1 with $\mathbf{w} = \mathbf{u}_j$ and $\mu = \theta_j$ until step 3, saving the resulting vector $x^{(1)} = \hat{u}^{(1)}$. It is important to observe that the first d blocks of $\mathbf{u}_j$ have to be constructed, since the variables in R-CORK are Q_j and $\mathbf{R}_j$, and $\mathbf{u}_j$ is not stored. As in CORK, they are computed as the single matrix-matrix product $Q_j\,[r_j^{(1)} \cdots r_j^{(d)}]$, where $r_j^{(1)}, \ldots, r_j^{(d)}$ are the first d blocks of the last column $\mathbf{r}_j$ of $\mathbf{R}_j$, which is a very efficient computation in terms of cache utilisation on modern computers. Once $\hat{u}^{(1)}$ is computed, by (4.16) we can decompose

$$\hat{u}^{(1)} = Q_j x_j + \alpha_j q_{j+1}, \tag{4.17}$$

where q_{j+1} is a unit vector orthogonal to Q_j and $x_j = Q_j^* \hat{u}^{(1)}$. Note also that since $\hat{u}^{(1)}$ has been already computed, we can compute the last s entries of $\hat{\mathbf{u}}$, denoted by

$\hat{v}$, from step 5 in Algorithm 4.1 without the need of performing step 4. The vector $\hat{v}$ will be used in the second level of orthogonalization. Now, if $\hat{u}^{(1)}$ does not lie in the subspace spanned by the columns of Q_j, i.e., if $\hat{u}^{(1)} - Q_j x_j \neq 0$, we can expand Q_j into Q_{j+1} as follows:

$$Q_{j+1} = [Q_j \quad q_{j+1}], \quad r_{j+1} = r_j + 1.$$

On the other hand, if $\hat{u}^{(1)}$ lies in the subspace spanned by the columns of Q_j, we have $Q_{j+1} = Q_j$ and $r_{j+1} = r_j$. The first level of orthogonalization is summarized in Algorithm 4.2. In step 2, if it is necessary, we can reorthogonalize $\tilde{q}$ to ensure orthogonality. In fact, in our MATLAB code, we perform the classical Gram-Schmidt method twice.

Algorithm 4.2 First level of orthogonalization in R-CORK

Input: The matrix $Q_j \in \mathbb{C}^{n \times r_j}$ and the vector $\hat{u}^{(1)} \in \mathbb{C}^n$ (the first block of $\hat{\mathbf{u}} = (\mathcal{A} - \theta_j \mathcal{B})^{-1} \mathcal{B} \mathbf{u}_j$).

Output: The matrix $Q_{j+1} \in \mathbb{C}^{n \times r_{j+1}}$, the vector x_j, and the scalar α_j.

Expanding Q_j into Q_{j+1}.

1. $x_j = Q_j^* \hat{u}^{(1)}$.
2. $\tilde{q} = \hat{u}^{(1)} - Q_j x_j$.
3. $\alpha_j = \|\tilde{q}\|_2$.

if $\alpha_j \neq 0$ **then**

4a. $Q_{j+1} = [Q_j \quad \tilde{q}/\alpha_j]$.

5a. $r_{j+1} = r_j + 1$.

else

4b. $Q_{j+1} = Q_j$.

5b. $r_{j+1} = r_j$.

end if

Second level of orthogonalization. In Algorithm 2.4, after choosing the shift θ_j and performing the shift-and-invert step, we need to compute the entries of the j-th column of $\underline{H}_j$ in step 4. We will explain how to do it efficiently in the R-CORK method. By using the compact representation of $\mathbf{U}_j$ in (4.4) - (4.5), we have

$$\begin{aligned} h_j &= \mathbf{U}_j^* \hat{\mathbf{u}}, \\ &= (R_j^{(1)})^* Q_j^* \hat{u}^{(1)} + \cdots + (R_j^{(d)})^* Q_j^* \hat{u}^{(d)} + V_j^* \hat{v}, \end{aligned} \tag{4.18}$$

where $\hat{\mathbf{u}}$ has been partitioned in an analogous way to (4.1). Since $(\mathcal{A} - \theta_j \mathcal{B})\hat{\mathbf{u}} = \mathcal{B}\mathbf{u}_j$, and $\mathcal{A}$ and $\mathcal{B}$ have the structures in (2.11), we obtain the following relation between the blocks of size n of $\hat{\mathbf{u}}$ and the blocks of size n of $\mathbf{u}_j$

$$\hat{u}^{(i)} = \theta_j \hat{u}^{(i-1)} + u_j^{(i-1)}, \quad \text{for } i = 2, \ldots, d. \tag{4.19}$$

Motivated by (4.19), we consider the vectors $x_j \in \mathbb{C}^{r_j}$ obtained in step 1 in Algorithm 4.2 and $\hat{v} \in \mathbb{C}^s$ obtained in step 5 in Algorithm 4.1 with $x^{(1)} = \hat{u}^{(1)}$ and $\mu = \theta_j$, and a vector $\hat{\mathbf{p}} \in \mathbb{C}^{dr_j+s}$ partitioned as follows

$$\hat{\mathbf{p}} = \begin{bmatrix} \hat{p}^{(1)} \\ \vdots \\ \hat{p}^{(d)} \\ \hat{v} \end{bmatrix}, \quad \hat{p}^{(i)} \in \mathbb{C}^{r_j}, \quad i = 1, \ldots, d, \tag{4.20}$$

with the blocks defined by the recurrence relation

$$\begin{aligned} \hat{p}^{(1)} &= x_j, \\ \hat{p}^{(i)} &= \theta_j \hat{p}^{(i-1)} + r_j^{(i-1)}, \quad i = 2, \ldots, d, \end{aligned} \tag{4.21}$$

where $r_j^{(i)}$ represents the j-th column of the block $R_j^{(i)}$ in (4.4). If $\alpha_j \neq 0$ in step 3 in Algorithm 4.2, by using $\hat{\mathbf{p}}$, the decomposition (4.17) and the recurrence relation (4.19), the vectors $\hat{u}^{(i)}$, $i = 1, \ldots, d$, corresponding to the partition of $\hat{\mathbf{u}}$ as in (4.1) can be represented as follows

$$\hat{u}^{(i)} = Q_{j+1} \begin{bmatrix} \hat{p}^{(i)} \\ \theta_j^{i-1} \alpha_j \end{bmatrix}, \quad i = 1, \ldots, d, \tag{4.22}$$

whereas that if $\alpha_j = 0$, we can represent the blocks $\hat{u}^{(i)}$, $i = 1, \ldots, d$, as follows

$$\hat{u}^{(i)} = Q_j \hat{p}^{(i)}. \tag{4.23}$$

Then, by using either (4.22) or (4.23) (depending on the value of α_j) in (4.18) and recalling that the columns of Q_{j+1} are orthonormal, we have

$$h_j = \begin{bmatrix} R_j^{(1)} \\ \vdots \\ R_j^{(d)} \\ V_j \end{bmatrix}^* \begin{bmatrix} \hat{p}^{(1)} \\ \vdots \\ \hat{p}^{(d)} \\ \hat{v} \end{bmatrix} = \mathbf{R}_j^* \hat{\mathbf{p}}. \tag{4.24}$$

Thus, after computing $\hat{\mathbf{p}}$ with the recurrence relation (4.21), the vector h_j can be computed by performing a matrix-vector multiplication of size $dr_j + s$, which, according to (4.15), is much smaller than $dn + s$ in large-scale problems and, even more, much smaller than n whenever $s \ll n$ as often happens in applications [76, 99].

Next, in step 5 of Algorithm 2.4, we need to compute the vector $\tilde{\mathbf{u}}$, which means that in R-CORK we need its compact representation. By using the compact representation of $\mathbf{U}_j$ and (4.22), we have, if $\alpha_j \neq 0$,

$$\tilde{\mathbf{u}} = \hat{\mathbf{u}} - \mathbf{U}_j h_j,$$

$$= \begin{bmatrix} Q_{j+1}\begin{bmatrix} \hat{p}^{(1)} \\ \alpha_j \end{bmatrix} \\ \vdots \\ Q_{j+1}\begin{bmatrix} \hat{p}^{(d)} \\ \theta_j^{d-1}\alpha_j \end{bmatrix} \\ \hat{v} \end{bmatrix} - \begin{bmatrix} Q_j R_j^{(1)} \\ \vdots \\ Q_j R_j^{(d)} \\ V_j \end{bmatrix} h_j,$$

$$= \begin{bmatrix} Q_{j+1} & & & \\ & \ddots & & \\ & & Q_{j+1} & \\ & & & I_s \end{bmatrix} \begin{bmatrix} \begin{bmatrix} \hat{p}^{(1)} - R_j^{(1)} h_j \\ \alpha_j \end{bmatrix} \\ \vdots \\ \begin{bmatrix} \hat{p}^{(d)} - R_j^{(d)} h_j \\ \theta_j^{d-1}\alpha_j \end{bmatrix} \\ \hat{v} - V_j h_j \end{bmatrix},$$

and, in a similar way, if $\alpha_j = 0$ we obtain

$$\tilde{\mathbf{u}} = \begin{bmatrix} Q_j & & & \\ & \ddots & & \\ & & Q_j & \\ & & & I_s \end{bmatrix} \begin{bmatrix} \hat{p}^{(1)} - R_j^{(1)} h_j \\ \vdots \\ \hat{p}^{(d)} - R_j^{(d)} h_j \\ \hat{v} - V_j h_j \end{bmatrix}.$$

Defining

$$\tilde{\mathbf{p}} := \begin{bmatrix} \tilde{p}^{(1)} \\ \vdots \\ \tilde{p}^{(d)} \\ \tilde{v} \end{bmatrix}, \quad \tilde{p}^{(i)} := \hat{p}^{(i)} - R_j^{(i)} h_j \in \mathbb{C}^{r_j},\ i = 1, \ldots, d, \quad \tilde{v} := \hat{v} - V_j h_j \in \mathbb{C}^s, \tag{4.25}$$

and taking into account that the columns of Q_{j+1} and Q_j are orthonormal, we can express the step 5 in Algorithm 2.4 as follows: if $\alpha_j \neq 0$, then

$$h_{j+1,j} = \|\tilde{\mathbf{u}}\|_2 = \left\| \begin{bmatrix} \tilde{p}^{(1)} \\ \alpha_j \\ \vdots \\ \tilde{p}^{(d)} \\ \theta_j^{d-1}\alpha_j \\ \tilde{v} \end{bmatrix} \right\|_2 \text{ and } \mathbf{u}_{j+1} = \mathbf{Q}_{j+1} \cdot \frac{1}{h_{j+1,j}} \begin{bmatrix} \tilde{p}^{(1)} \\ \alpha_j \\ \vdots \\ \tilde{p}^{(d)} \\ \theta_j^{d-1}\alpha_j \\ \tilde{v} \end{bmatrix}, \tag{4.26}$$

where the notation in (4.5) is used, while if $\alpha_j = 0$ we proceed as in (4.26) by removing all the entries involving α_j and with $\mathbf{Q}_{j+1} = \mathbf{Q}_j$. From the previous

equations, we can conclude that the first d blocks of size r_{j+1} of the last column of $\mathbf{R}_{j+1}$ in (4.5) are given if $\alpha_j \neq 0$ by

$$r_{j+1}^{(i)} = \frac{1}{h_{j+1,j}} \begin{bmatrix} \tilde{p}^{(i)} \\ \theta_j^{i-1}\alpha_j \end{bmatrix}, \quad i = 1, \ldots, d, \tag{4.27}$$

and if $\alpha_j = 0$ by

$$r_{j+1}^{(i)} = \frac{1}{h_{j+1,j}} \tilde{p}^{(i)}, \quad i = 1, \ldots, d. \tag{4.28}$$

In addition, the last block of size s of the last column of $\mathbf{R}_{j+1}$ is

$$v_{j+1} = \frac{1}{h_{j+1,j}} \tilde{v}. \tag{4.29}$$

Since $\mathbf{R}_j$ has orthonormal columns, from (4.24) and the definitions in (4.20)-(4.21) and (4.25), we have that h_j and $\tilde{\mathbf{p}}$ satisfy

$$\tilde{\mathbf{p}} = \hat{\mathbf{p}} - \mathbf{R}_j h_j, \tag{4.30}$$

where $\tilde{\mathbf{p}}$ is orthogonal to $\mathbf{R}_j$. This process is the Gram-Schmidt process without the normalization step, and it is summarized in Algorithm 4.3.

Algorithm 4.3 Second level of orthogonalization in R-CORK

Input: The matrix $\mathbf{R}_j \in \mathbb{C}^{(dr_j+s)\times j}$ and the vector $\hat{\mathbf{p}} \in \mathbb{C}^{dr_j+s}$ from (4.20)-(4.21).
Output: Vectors $h_j \in \mathbb{C}^j$ and $\tilde{\mathbf{p}} \in \mathbb{C}^{dr_j+s}$.

1. $h_j = \mathbf{R}_j^* \hat{\mathbf{p}}$.
2. $\tilde{\mathbf{p}} = \hat{\mathbf{p}} - \mathbf{R}_j h_j$.

Remark 4.8. *In order to improve orthogonality, a reorthogonalization method can be included in Algorithm 4.3. In our MATLAB code, we use the classical Gram-Schmidt process twice.*

The whole procedure of this new method to solve large-scale and sparse rational eigenvalue problems requires the use of the two levels of orthogonalization described in this section, the first level to expand Q_j into Q_{j+1} and the second level to expand $\mathbf{R}_j$ into $\mathbf{R}_{j+1}$. The complete R-CORK method is summarized in Algorithm 4.4. Note that R-CORK has as inputs the matrix Q_1 and the vector $\mathbf{R}_1$, which have to be computed. As in CORK [104, p. 830], there are two possible ways of computing these inputs: either starting with a random vector $\mathbf{u}_1 \in \mathbb{C}^{nd+s}$ and using an economy-size QR factorization, or emulating the structure of the eigenvectors (2.12) of the linearization in (2.10). We know from Theorem 2.14 that the eigenvectors have a Kronecker structure. Therefore, Algorithm 4.4 can also be started with

$$\mathbf{u}_1 = \begin{bmatrix} f \otimes g \\ t \end{bmatrix}$$

where $f \in \mathbb{C}^n$, $g \in \mathbb{C}^d$ and $g \in \mathbb{C}^s$. This results in

$$Q_1 = g/\|g\|_2 \in \mathbb{C}^n \quad \text{and} \quad \mathbf{R}_1 = \begin{bmatrix} \|g\|_2 f \\ t \end{bmatrix} \in \mathbb{C}^{(d+s)\times 1},$$

with $r_1 = 1$. Recall in Algorithm 4.4 that $\mathbf{r}_j$ denotes the last column of the matrix $\mathbf{R}_j$ in (4.5).

Algorithm 4.4 Compact rational Krylov method for REP (R-CORK)

Input: $Q_1 \in \mathbb{C}^{n\times r_1}$, $\mathbf{R}_1 \in \mathbb{C}^{(dr_1+s)\times 1}$ with $Q_1^* Q_1 = I_{r_1}$ and $\mathbf{R}_1^* \mathbf{R}_1 = 1$.
Output: Approximate eigenpairs $(\lambda, \mathbf{x})$ of $\mathcal{A} - \lambda\mathcal{B}$ with $\mathcal{A}$ and $\mathcal{B}$ as in (2.11).
for $j = 1, 2, \ldots$ **do**
1. Choose shift θ_j.
2. Compute $\mathbf{u}_j = \mathbf{Q}_j \mathbf{r}_j$, obtaining the first d blocks as matrix-matrix product $Q_j\,[r_j^{(1)} \cdots r_j^{(d)}]$.
3. Compute $\hat{u}^{(1)}$ by using Algorithm 4.1 until step 3 applied to $\mathbf{w} = \mathbf{u}_j$ and $\mu = \theta_j$.
4. Compute $\hat{v}$ from step 5 in Algorithm 4.1.

First level of orthogonalization
5. Run Algorithm 4.2 obtaining Q_{j+1}, the scalar α_j and the vector x_j.

Second level of orthogonalization:
6. Compute $\hat{\mathbf{p}}$ in (4.20) via the recurrence relation in (4.21).
7. Run Algorithm 4.3 obtaining $\tilde{\mathbf{p}}$ and h_j.
8. Compute $h_{j+1,j}$ and $\mathbf{r}_{j+1}$ using (4.26)-(4.27)-(4.28)-(4.29) and get $\mathbf{R}_{j+1}$ with Lemma 4.7.
9. Compute eigenpairs: (λ_i, t_i) of (2.27) and test for convergence.

end for
10. Compute eigenvectors: $\mathbf{x}_i = \mathbf{Q}_{j+1}\mathbf{R}_{j+1}\underline{H}_j t_i$.

4.3 Memory and computational costs

In this section, we discuss the memory and the computational costs of R-CORK and compare these costs with those of the classical rational Krylov (RK) method, i.e., Algorithm 2.4, applied directly to the linearization $\mathcal{A} - \lambda\mathcal{B}$ of the REP in (2.10). In order to simplify the results we will take $r_j = j + d$ in R-CORK, which is the upper bound in Theorem 4.6 and that essentially corresponds to start the R-CORK iteration with $Q_1 \in \mathbb{C}^{n\times d}$ ($r_1 = d$) or, equivalently, with a random initial vector $\mathbf{u}_1$ whose first d blocks in the partition (4.1) are linearly independent. If the first d blocks of $\mathbf{u}_1$ are taken to be collinear, then one can take $r_j = j$ and to improve even more the costs of R-CORK. In addition, note that we estimate the costs for any

value of s, where $s \times s$ is the size of the lower-right block $C - \lambda D$ of $\mathcal{A} - \lambda\mathcal{B}$ appearing in the strictly proper part of the REP (2.9). In this way, it will be seen that even if $s \approx n$, R-CORK has considerable advantages with respect to RK in terms of memory and computational costs. However, we emphasize that such advantages are still much more relevant when $s \ll n$, as happens very often in applications [76, 99].

For the memory costs, after j iterations R-CORK stores $Q_j \in \mathbb{C}^{n \times r_j}$ and $\mathbf{R}_j \in \mathbb{C}^{(dr_j+s)\times j}$, which amounts to $(n+dj)(j+d)+sj \approx n(j+d)+sj$ numbers. Note that the approximation $n + dj \approx n$ holds in any reasonable large-scale REP. In contrast, RK stores $\mathbf{U}_j$, which amounts to $(nd + s)j = ndj + sj$ numbers. Since, $(j + d) < dj$ for most reasonable choices of j and degrees d appearing in practice, we see that R-CORK is much more memory-efficient than RK. These memory costs are shown in Table 4.1.

With respect to the computational costs, observe that for both R-CORK and RK the cost is the sum of (i) the shift-and-invert step and (ii) the orthogonalization steps. Let us analyze first the shift-and-invert steps. If the shift-and-invert step in RK, i.e., step 3 in Algorithm 2.4, is performed by applying an unstructured solver to the $(nd + s) \times (nd + s)$ linear system $(\mathcal{A} - \theta_j\mathcal{B})\hat{\mathbf{u}} = \mathcal{B}\mathbf{u}_j$, then the cost of RK is much larger than the cost of R-CORK, since R-CORK solves this system with Algorithm 4.1 (removing step 4) which is much more efficient because it requires the solution of smaller linear systems (essentially, see Remark 4.4, one of size $n \times n$ and two of size $s \times s$, which are very often extremely small). However, one can consider to perform the shift-and-invert step in RK with Algorithm 4.1, but this is still somewhat more expensive than R-CORK, because for RK it is needed to perform step 4 of Algorithm 4.1, with an additional cost of $2n(d - 2)$ flops in each iteration (see Remark 4.4). A final important remark on the shift-and-invert step is that R-CORK involves the overhead cost of constructing $\mathbf{u}_j$ in step 2 of Algorithm 4.4, which in RK is not needed. However, note that, as explained in previous sections, this construction can be performed as in CORK via a single matrix-matrix product, which allows for optimal efficiency and cache utilisation on modern computers [63, p. 577]. Moreover, we emphasize that a traditional construction of $\mathbf{u}_j$ in R-CORK costs $\mathcal{O}(dnr_j) = \mathcal{O}(dn(j + d)) \approx \mathcal{O}(dnj)$ flops at iteration j, which added to the orthogonalization cost of R-CORK discussed below would give a cost of the same order of the orthogonalization cost of RK.

Finally, we discuss the orthogonalization costs of RK and R-CORK. In RK, the orthogonalization is performed in steps 4-5-6 of Algorithm 2.4 and its cost is well-known to be $\mathcal{O}(j(nd + s)) = \mathcal{O}(jnd + js)$ flops at iteration j, which amounts to $\mathcal{O}(j^2nd + j^2s)$ flops in the first j iterations (see Table 4.1). In R-CORK, the orthogonalization is performed in steps 5-6-7-8 of Algorithm 4.4. At iteration j, the cost of step 5 is $\mathcal{O}(r_jn) = \mathcal{O}((j + d)n)$ flops, the cost of step 6 is $\mathcal{O}(r_jd) = \mathcal{O}((j + d)d)$ flops, which is negligible with respect to the cost of step 5, the cost of step 7 is $\mathcal{O}(j(dr_j + s)) = \mathcal{O}(jd(j + d) + js)$ flops, and the cost of step 8 is $\mathcal{O}(dr_j + s) = \mathcal{O}(d(j + d) + s)$ flops. Therefore, the total cost at iteration j of the

orthogonalization in R-CORK is $\mathcal{O}((n+jd)(j+d)+js) \approx \mathcal{O}(n(j+d)+js)$, where we have used again the approximation $n+jd \approx n$, which gives $\mathcal{O}(j^2n+jdn+j^2s)$ flops in the first j iterations (see Table 4.1). Observe that the orthogonalization cost of RK includes the large term j^2nd which is not present in the cost of R-CORK. Therefore, the orthogonalization cost of R-CORK is considerably smaller than the one of RK.

In Table 4.1, the comparison of the costs between R-CORK and RK is summarized.

	Classical rational Krylov method	R-CORK method
Orthogonalization cost	$\mathcal{O}(j^2nd+j^2s)$	$\mathcal{O}(j^2n+jdn+j^2s)$
Memory cost	$ndj+sj$	$n(j+d)+sj$

Table 4.1: Orthogonalization and memory costs for classical rational Krylov method and R-CORK method after j iterations.

4.4 Implicit restarting in R-CORK

Practical implementations of any Krylov-type method for computing eigenvalues of large-scale problems require effective restarting strategies. The goal of this section is to develop an implicit restarting strategy for R-CORK that restarts both Q_j and $\mathbf{R}_j$ in the compact representation of $\mathbf{U}_j$ in (4.4)-(4.5). Since R-CORK shares many of the properties of CORK, the results of this section are similar to those in [104, Section 6], which in turn are based on implicit restarting procedures for classical rational Krylov methods [32] and on the Krylov-Schur restart developed for TOAR in [63, Section 4.2].

Following the Krylov-Schur spirit [98] (see also [97, Section 5.2]), the restarting technique we propose transforms first the matrices $\underline{H}_j$ and $\underline{K}_j$ in (4.6) to (quasi)triangular form (i.e. $\underline{H}_j$ and $\underline{K}_j$ are block upper triangular with 1-by-1 and 2-by-2 blocks on the diagonal), in order to reorder the Ritz values and to preserve the desired ones with a rational Krylov subspace of smaller dimension. Second, by representing the new smaller Krylov subspace in its compact form in an efficient way, the implicit restart of R-CORK is completed. The main difference of the process described below with respect to the implicit restarting in [104, Section 6] is that here we need to add a new block of size $s \times s$ corresponding to the rational part of $R(\lambda)$ in (2.9).

Suppose that after j iterations, we have the rational Krylov recurrence relation in its compact form as in (4.6)

$$\mathcal{A}\mathbf{Q}_{j+1}\mathbf{R}_{j+1}\underline{H}_j = \mathcal{B}\mathbf{Q}_{j+1}\mathbf{R}_{j+1}\underline{K}_j, \tag{4.31}$$

and we want to reduce this representation to a smaller compact rational decomposition of size p, $p < j$, this is

$$\mathcal{A}\mathbf{Q}_{p+1}^{+}\mathbf{R}_{p+1}^{+}\underline{H}_{p}^{+} = \mathcal{B}\mathbf{Q}_{p+1}^{+}\mathbf{R}_{p+1}^{+}\underline{K}_{p}^{+}, \quad p < j.$$

For this purpose, we consider the generalized Schur decomposition:

$$H_j = \begin{bmatrix} Y_p & Y_{j-p} \end{bmatrix} \begin{bmatrix} T_{p\times p}^{(H)} & * \\ 0 & T_{(j-p)\times(j-p)}^{(H)} \end{bmatrix} \begin{bmatrix} Z_p^* \\ Z_{j-p}^* \end{bmatrix}, \tag{4.32}$$

$$K_j = \begin{bmatrix} Y_p & Y_{j-p} \end{bmatrix} \begin{bmatrix} T_{p\times p}^{(K)} & * \\ 0 & T_{(j-p)\times(j-p)}^{(K)} \end{bmatrix} \begin{bmatrix} Z_p^* \\ Z_{j-p}^* \end{bmatrix}, \tag{4.33}$$

where H_j and K_j are the $j \times j$ upper Hessenberg matrices obtained by removing the last row of $\underline{H}_j$ and $\underline{K}_j$ respectively, $Y := \begin{bmatrix} Y_p & Y_{j-p} \end{bmatrix}$, $Z := \begin{bmatrix} Z_p & Z_{j-p} \end{bmatrix} \in \mathbb{C}^{j\times j}$ are unitary matrices with Y_p, $Z_p \in \mathbb{C}^{j\times p}$, Y_{j-p}, $Z_{j-p} \in \mathbb{C}^{j\times(j-p)}$ and $T^{(H)} := \begin{bmatrix} T_{p\times p}^{(H)} & * \\ 0 & T_{(j-p)\times(j-p)}^{(H)} \end{bmatrix}$, $T^{(K)} := \begin{bmatrix} T_{p\times p}^{(K)} & * \\ 0 & T_{(j-p)\times(j-p)}^{(K)} \end{bmatrix} \in \mathbb{C}^{j\times j}$ are upper (quasi) triangular matrices with $T_{p\times p}^{(H)}, T_{p\times p}^{(K)} \in \mathbb{C}^{p\times p}$ and $T_{(j-p)\times(j-p)}^{(H)}, T_{(j-p)\times(j-p)}^{(K)} \in \mathbb{C}^{(j-p)\times(j-p)}$. The $p < j$ Ritz values of interest are the eigenvalues of the pencil $T_{p\times p}^{(K)} - \lambda T_{p\times p}^{(H)}$. By multiplying by Z on the right the recurrence relation (4.31) and using (4.32) and (4.33), and considering the first p columns, we obtain:

$$\mathcal{A}\mathbf{Q}_{j+1}\mathbf{R}_{j+1}\begin{bmatrix} Y_p & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} T_{p\times p}^{(H)} \\ h_{j+1,j}\tilde{z}^* \end{bmatrix} = \mathcal{B}\mathbf{Q}_{j+1}\mathbf{R}_{j+1}\begin{bmatrix} Y_p & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} T_{p\times p}^{(K)} \\ k_{j+1,j}\tilde{z}^* \end{bmatrix}, \tag{4.34}$$

where $\tilde{z}^*$ represents the first p entries of the last row of Z. By introducing the notation:

$$Y_1 := \begin{bmatrix} Y_p & 0 \\ 0 & 1 \end{bmatrix} \in \mathbb{C}^{(j+1)\times(p+1)}, \quad \underline{H}_p^+ := \begin{bmatrix} T_{p\times p}^{(H)} \\ h_{j+1,j}\tilde{z}^* \end{bmatrix}, \quad \underline{K}_p^+ = \begin{bmatrix} T_{p\times p}^{(K)} \\ k_{j+1,j}\tilde{z}^* \end{bmatrix} \in \mathbb{C}^{(p+1)\times p}, \tag{4.35}$$

and defining $\mathbf{W}_{p+1} = \mathbf{R}_{j+1}Y_1$, we obtain

$$\mathcal{A}\mathbf{Q}_{j+1}\mathbf{W}_{p+1}\underline{H}_p^+ = \mathcal{B}\mathbf{Q}_{j+1}\mathbf{W}_{p+1}\underline{K}_p^+. \tag{4.36}$$

Note that with this transformation, we reduce the size of the matrices $\underline{H}_p^+, \underline{K}_p^+$, and $\mathbf{W}_{p+1}$ with respect to $\underline{H}_j, \underline{K}_j$, and $\mathbf{R}_{j+1}$, and remove the Ritz values that are not of interest. However, observe that the large factor $\mathbf{Q}_{j+1}$ remains unchanged. In order to reduce the size of $\mathbf{Q}_{j+1}$, consider

$$\mathbf{W}_{p+1} = \begin{bmatrix} W_{p+1}^{(1)} \\ \vdots \\ W_{p+1}^{(d)} \\ V_{j+1}Y_1 \end{bmatrix}, \quad W_{p+1}^{(i)} \in \mathbb{C}^{r_{j+1}\times(p+1)}, \quad i = 1, \ldots, d,$$

and let ω be the rank of $[W_{p+1}^{(1)} \quad \cdots \quad W_{p+1}^{(d)}]$. The key observation is that although the matrices $\underline{H}_p^+, \underline{K}_p^+$ are no longer in Hessenberg form, the subspace spanned by the columns of $(\mathbf{Q}_{j+1}\mathbf{W}_{p+1})$ is still a rational Krylov subspace corresponding to $\mathcal{A}-\lambda\mathcal{B}$ [32]. Therefore, we can apply Theorem 4.6 to span $\left\{Q_{j+1}W_{p+1}^{(1)}, \ldots, Q_{j+1}W_{p+1}^{(d)}\right\} = Q_{j+1}\,\text{span}\left\{W_{p+1}^{(1)}, \ldots, W_{p+1}^{(d)}\right\}$ to obtain that $\omega \leq d+p < d+j$. Then, the economy singular value decomposition of:

$$[W_{p+1}^{(1)} \quad \cdots \quad W_{p+1}^{(d)}] \;=\; \mathcal{U}\mathcal{S}[\mathcal{V}^{(1)} \quad \cdots \quad \mathcal{V}^{(d)}],$$

is computed, where $\mathcal{U} \in \mathbb{C}^{r_{j+1}\times\omega}$, $\mathcal{S} \in \mathbb{C}^{\omega\times\omega}$ and $\mathcal{V}^{(i)} \in \mathbb{C}^{\omega\times(p+1)}$ for $i = 1, \ldots, d$. Thus, by defining

$$Q_{p+1}^+ = Q_{j+1}\mathcal{U}, \quad \mathbf{R}_{p+1}^+ = \begin{bmatrix} \mathcal{S}\mathcal{V}^{(1)} \\ \vdots \\ \mathcal{S}\mathcal{V}^{(d)} \\ V_{j+1}Y_1 \end{bmatrix}, \quad \mathbf{Q}_{p+1}^+ = \begin{bmatrix} Q_{p+1}^+ & & & \\ & \ddots & & \\ & & Q_{p+1}^+ & \\ & & & I_s \end{bmatrix},$$

we get from (4.36) the compact rational Krylov recurrence relation

$$\mathcal{A}\mathbf{Q}_{p+1}^+\mathbf{R}_{p+1}^+\underline{H}_p^+ = \mathcal{B}\mathbf{Q}_{p+1}^+\mathbf{R}_{p+1}^+\underline{K}_p^+, \tag{4.37}$$

with $p < j$. It is important to emphasize again that the matrices $\underline{H}_p^+$ and $\underline{K}_p^+$ are no longer upper Hessenberg matrices, however, they contain the required Ritz values and the columns of $\mathbf{Q}_{p+1}^+\mathbf{R}_{p+1}^+$ span a corresponding rational Krylov subspace. We continue the process by expanding (4.37) with Algorithm 4.4 until we get a rational Krylov subspace of dimension j. The matrices $\underline{H}_j^+$ and $\underline{K}_j^+$ obtained in this expansion are not in Hessenberg form, although their columns $p+1, \ldots, j$ have a Hessenberg structure (see [97, p. 329]). Then, the restarting process described in this section is applied again to get a new compact relation (4.37) of "size p". This expansion-restarting procedure is cyclicly repeated until the prescribed stopping criterion is satisfied for a certain desired number, less than or equal to p, of Ritz pairs.

4.5 Numerical tests

In this section, we present two large-scale and sparse numerical examples to illustrate the efficiency of the R-CORK method. All reported experiments were performed using Matlab R2013a on a PC with a 2,2 GHz Intel (R) Core (TM) i7 processor, with 16 GB of RAM and DDR3 memory type, and with operating system macOS Sierra, version 10.12.1.

By following [104, Section 8], in the numerical experiments we plot the residuals at each iteration, with and without restarts, obtained by using the R-CORK method,

the dimension of the subspace at each iteration for R-CORK, and the comparison of the memory storages of R-CORK and of the classical rational Krylov method applied directly to the linearization (2.10). We also report on the number of iterations until convergence.

Inspired by the applications in [99, Section 4], we construct numerical experiments with prescribed eigenvalues and poles of a rational matrix $R(\lambda)$ represented as in (2.9). In order to measure the convergence of an approximate eigenpair (λ, x) of $R(\lambda)$, the relative norm residual:

$$E(\lambda, x) = \frac{\|R(\lambda)x\|_2}{(\sum_{i=0}^{d} |\lambda|^i \|P_i\|_F + \|E(C-\lambda D)^{-1}F^T\|_F)\|x\|_2}. \tag{4.38}$$

is considered. Note that the computation of $E(\lambda, x)$ involves matrices and vectors of size n and, so, is expensive. Therefore, in actual practice, we recommend to test first the convergence through a cheap estimation of the residual of the linearized problem, i.e., $\|(\mathcal{A}-\lambda\mathcal{B})\mathbf{z}\|_2$, involving only the small projected problem (2.27), and once such residual is sufficiently small to compute the residual (4.38) every $q > 1$ iterations instead of at each iteration. However, in our examples, we performed the computation of $E(\lambda, x)$ at each iteration for the purpose of illustration.

The computation of (4.38) deserves some comments. Note first that it requires to recover the approximated eigenvector x of $R(\lambda)$ from the approximated eigenvector $\mathbf{z}$ of the linearization $\mathcal{A}-\lambda\mathcal{B}$ in (2.10) computed in step 10 of Algorithm 4.4. This recovery, according to the first equation in (2.12), can be done by taking any of the first d blocks of $\mathbf{z}$ if $\lambda \neq 0$. Since in our numerical examples the moduli of the approximate eigenvalues are larger than 1, we have chosen the d-th block of $\mathbf{z}$ as approximate x. However, we recommend to choose the first block if the moduli of the approximate eigenvalues are smaller than 1. The calculation of the quantities $\|P_i\|_F$, $i = 0, \ldots, d$ needs to be performed only once and it is inexpensive since the matrices P_i are sparse in practice. Finally, to compute the expression $\|E(C-\lambda D)^{-1}F^T\|_F$ on the denominator in (4.38), we use

$$\begin{aligned} \|E(C-\lambda D)^{-1}F^T\|_F^2 &= \operatorname{trace}((E(C-\lambda D)^{-1}F^T)^* E(C-\lambda D)^{-1}F^T), \\ &= \operatorname{trace}((E^*E)(C-\lambda D)^{-1}(F^T\bar{F})(C-\lambda D)^{-*}), \end{aligned}$$

which only involves the matrices E^*E, $F^T\bar{F}$, $(C-\lambda D)^{-1}$ and $(C-\lambda D)^{-*}$ of size $s \times s$. Since in many application $s \ll n$, this computation is usually inexpensive.

Numerical test 4.9. We construct a REP of the type arising from the free vibrations of a structure if one uses a viscoelastic constitutive relation to describe the behavior of a material [76, 99]. The REPs of this type have the following structure:

$$R(\lambda)x = \left(\lambda^2 M + K - \sum_{i=1}^{k} \frac{1}{1+b_i\lambda}\Delta G_i\right) x = 0, \tag{4.39}$$

where the mass and stiffness matrices M and K are real symmetric and positive definite, b_j are relaxation parameters over k regions, and ΔG_j is an assemblage of element stiffness matrices over the region with the distinct relaxation parameters. As in [99], we consider the case where $\Delta G_i = E_i E_i^T$ and $E_i \in \mathbb{R}^{n\times s_i}$. By defining

$$E = [E_1, E_2, \ldots, E_k], \quad D = \operatorname{diag}(b_1 I_{s_1}, b_2 I_{s_2}, \ldots, b_k I_{s_k}),$$

the REP (4.39) can be written in the form (2.9):

$$(\lambda^2 M + K - E(I + \lambda D)^{-1} E^T)x = 0.$$

In our particular example, we consider the case with one region and one relaxation parameter $b_1 = -1$. The construction of the matrices M and K in our example proceeds as follows: construct first $R_1(\lambda) = \lambda^2 A_2 + A_0 - e_{10000}(1-\lambda)^{-1}(e_{10000})^T$, with $A_2, A_0 \in \mathbb{R}^{10000\times 10000}$ diagonal and positive definite matrices where $A_2 = I_{10000}$, $A_0(i,i) = i^2$ and e_{10000} the last column of I_{10000}. This structure allows to prescribe easily the eigenvalues for $R_1(\lambda)$. Then, we consider the following invertible tridiagonal matrix P

$$P = \begin{bmatrix} 1 & \frac{1}{2} & & \\ \frac{1}{3} & 1 & \ddots & \\ & \ddots & \ddots & \frac{1}{2} \\ & & \frac{1}{3} & 1 \end{bmatrix},$$

and finally construct $R(\lambda) = PR_1(\lambda)P^T$. Since P is invertible, the eigenvalues of $R(\lambda)$ and $R_1(\lambda)$ are the same. By using this procedure, we have constructed the REP

$$R(\lambda)\, x = \left(\lambda^2 M + K - p_{10000}(1-\lambda)^{-1}(p_{10000})^T\right)\, x = 0, \tag{4.40}$$

where $M := PA_2P^T$, $K := PA_0P^T \in \mathbb{R}^{10000\times 10000}$ are symmetric, positive definite, and pentadiagonal matrices, and $p_{10000} \in \mathbb{R}^{10000}$ represents the last column of the matrix P.

In this example we are interested in computing the 20 eigenvalues of (4.40) with negative imaginary part and with largest absolute value of the negative imaginary part. To aim our goal, we use 3 cyclically repeated shifts in the rational Krylov steps and a random unit real vector as an initial vector. The reader can see the approximate eigenvalues computed by R-CORK and the chosen shifts in Figure 4.1(a). We first solve the REP (4.40) by using Algorithm 4.4 without restart, and after 85 iterations, we find the required eigenvalues with a tolerance (4.38) of 10^{-10}. The convergence history is shown in Figure 4.1(b). In Figure 4.1(d), we plot r_j, the rank of Q_j at the iteration j, and j, the dimension of the Krylov subspace. Since we did not perform restart, we can see that both, r_j and j increases with the iteration count j and that $r_j = j+1$, as expected since the degree of the polynomial part of (4.40) is $d = 2$. Figure 4.1(f) displays the comparison between the cost

of memory storage of both the R-CORK method, by using Algorithm 4.4, and the classical rational Krylov method, by using Algorithm 2.4. From this figure, we can see that the R-CORK method requires approximately half of the memory storage that the classical rational Krylov method, which is consistent with the degree 2 of the polynomial part of (4.40).

Next, we apply Algorithm 4.4 to the REP (4.40) combined with the implicit restarting introduced in Section 4.4. We choose the maximum dimension of the subspace $m = 45$, which is reduced after each restart to dimension $p = 30$ to compute the 20 required eigenvalues. The convergence history of the eigenpairs (λ, x) computed by this restarted R-CORK method is shown in Figure 4.1(c). After 3 restarts and 81 iterations, the 20 required eigenvalues have been found with a tolerance (4.38) of 10^{-10}. In Figure 4.1(e) the reader can see the rank of Q_j at the j-th iteration and it can be seen that with restart, the relation between j and r_j continues the same. Finally, in Figure 4.1(g) we plot the memory storage for R-CORK and classical rational Krylov, and it can be observed that the memory cost for the R-CORK method is a factor close to 2 smaller than the memory cost obtained by the classical rational Krylov method.

As a final comment, note that $R(\lambda)$ has prescribed eigenvalues, therefore, we can compare both approximate and exact eigenvalues. If $\hat{\lambda}$ denotes the approximations and λ the exact eigenvalues, our numerical results show that, in practice,

$$\frac{|\hat{\lambda} - \lambda|}{\lambda} < 10^{-10}, \tag{4.41}$$

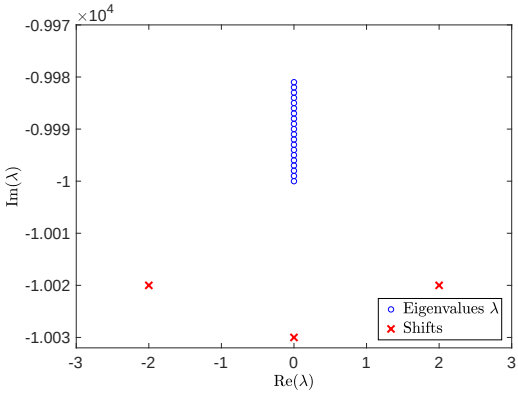
for every approximate eigenvalue $\hat{\lambda}$.

Numerical test 4.10. For this numerical example, we consider an academic REP of size 5000×5000 and with the degree of its polynomial part equal to 3, i.e., a REP of the form

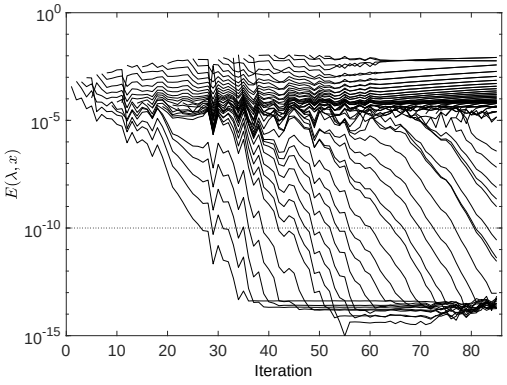
$$R(\lambda) = \lambda^3 A_3 + \lambda^2 A_2 + \lambda A_1 + A_0 - E(C - \lambda D)^{-1} F^T. \tag{4.42}$$

The coefficient matrices of $R(\lambda)$ in (4.42) were constructed in a similar way as in the numerical experiment 4.9: first, we consider a rational matrix $R_2(\lambda) = \lambda^3 P_3 + \lambda^2 P_2 + \lambda P_1 + P_0 - E_0(C - \lambda D)^{-1} F_0^T$ with prescribed eigenvalues, where $P_i \in \mathbb{R}^{5000\times 5000}$ are diagonal matrices, $E_0 = [e_1 + e_2, \quad e_5 + e_6]$, $\quad F_0 = [e_{4997} + e_{4998}, \quad e_{4999} + e_{5000}] \in \mathbb{R}^{5000\times 2}$, with e_i the ith canonical vector of size 5000, and

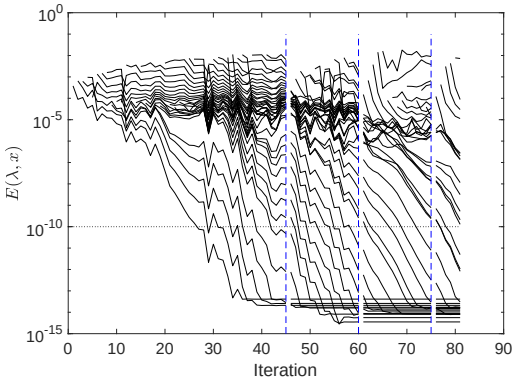
$$C = \begin{bmatrix} 105 & 0 \\ 0 & -105 \end{bmatrix}, \quad D = I_2,$$



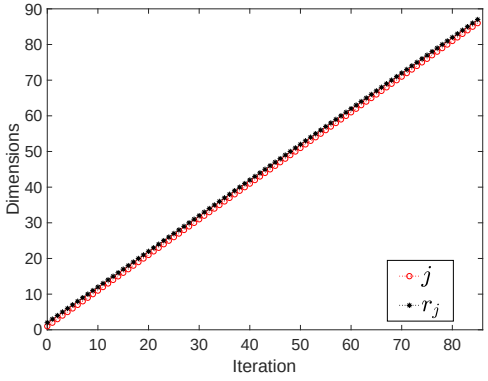
(a) Eigenvalues.



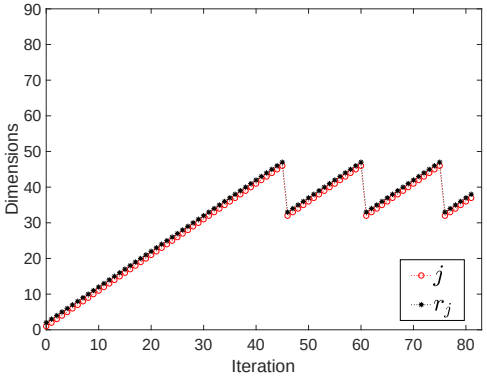
(b) Convergence history without restart.



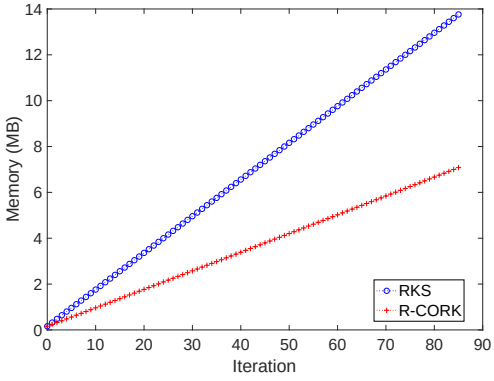
(c) Convergence history with restart.



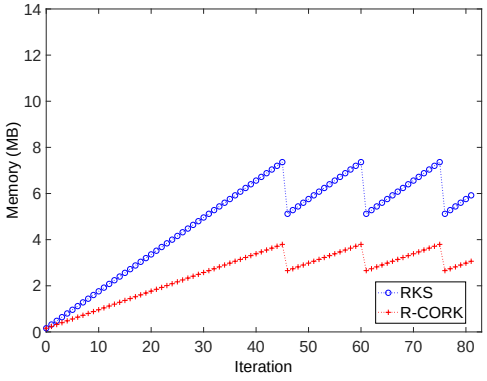
(d) Dimension of the subspace without restart.



(e) Dimension of the subspace with restart.



(f) Memory cost without restart.



(g) Memory cost with restart.

Figure 4.1: Numerical experiment 4.9.

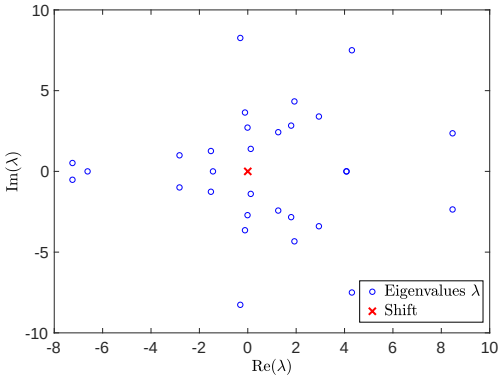
and then we define $R(\lambda) = PR_2(\lambda)Q$, where

$$P = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & & \\ -\frac{1}{4} & \ddots & \ddots & \ddots & \\ -\frac{1}{5} & \ddots & \ddots & \ddots & \frac{1}{3} \\ & \ddots & \ddots & \ddots & \frac{1}{2} \\ & & -\frac{1}{5} & -\frac{1}{4} & 1 \end{bmatrix}, \quad Q = \begin{bmatrix} -1 & -\frac{1}{3} & & \\ \frac{1}{2} & \ddots & \ddots & \\ & \ddots & \ddots & -\frac{1}{3} \\ & & \frac{1}{2} & -1 \end{bmatrix} \in \mathbb{R}^{5000\times 5000}.$$

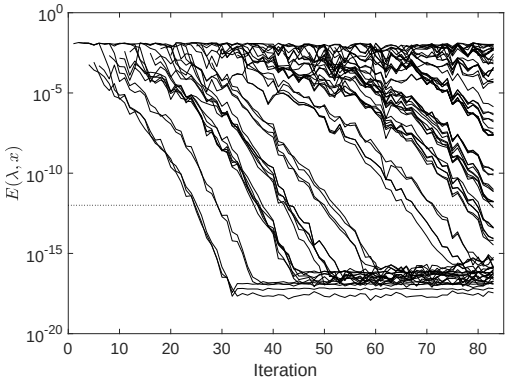
The goal of this example is to compute the 30 eigenvalues closest to zero. In this situation, it is natural to choose the origin as a fixed shift. In Figure 4.2(a), the approximate eigenvalues computed by R-CORK are displayed. By starting with a random unit complex vector, first we apply R-CORK without restarting, and after 83 iterations, the desired eigenvalues are obtained with a tolerance (4.38) of 10^{-12}. The convergence history can be seen in Figure 4.2(b). In Figure 4.2(d), we see that the relation $r_j < j + d$ with j the number of iterations also holds in this example, though in this case with $d = 3$ since this is the degree of the polynomial part in (4.42). Figure 4.2(f) shows the memory costs of R-CORK and classical rational Krylov. It is observed that the reduction in cost of R-CORK is approximately a factor of 3, i.e., the degree of the polynomial part of (4.42).

As a final example, we solve (4.42) by using R-CORK combined with restarting and taking a maximum subspace dimension $m = 60$ which is reduced to $p = 40$ after every restart. The convergence history is shown in Figure 4.2(c), where it is observed that after 91 iterations and 2 restarts, the 30 eigenvalues closest to the origin have been found with a tolerance (4.38) of 10^{-12}. Despite the fact that a few more iterations are needed with restart than without restart, we see in Figure 4.2(e) that we are using a subspace of much smaller dimension to compute the eigenpairs, and, particularly for this example, $r_j < j$ after the restart. Finally, the comparison of the memory costs for the R-CORK and for the classical rational Krylov methods is plotted in Figure 4.2(g), where we see again that the cost of R-CORK is a factor $d = 3$ smaller. Finally, relation (4.41) also holds for the approximate eigenvalues $\hat{\lambda}$.

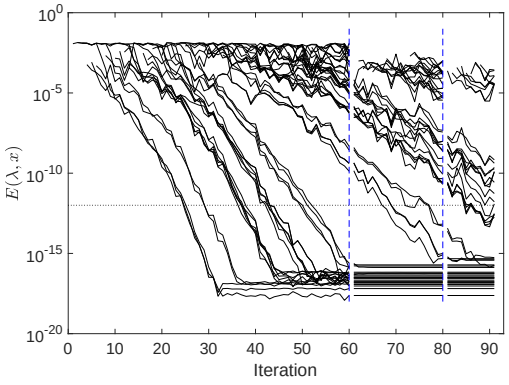
As we can see, the numerical experiments confirm all the good properties of the R-CORK method that we mentioned along this chapter. The combined use of the compact representation of rational Krylov subspaces and the two levels of orthogonalization in R-CORK reduces significantly the orthogonalization and the memory costs with respect to a direct application of the classical rational Krylov method to the linearization (2.10) of $\mathcal{A} - \lambda\mathcal{B}$. More conclusions are presented in Chapter 6.



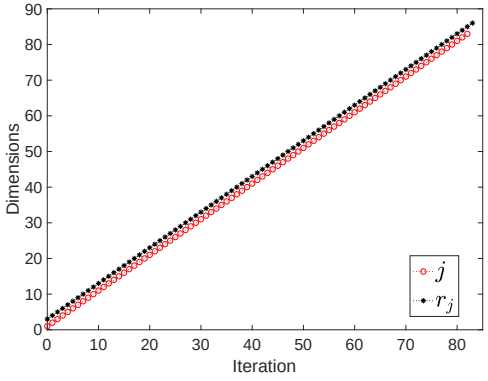
(a) Eigenvalues.



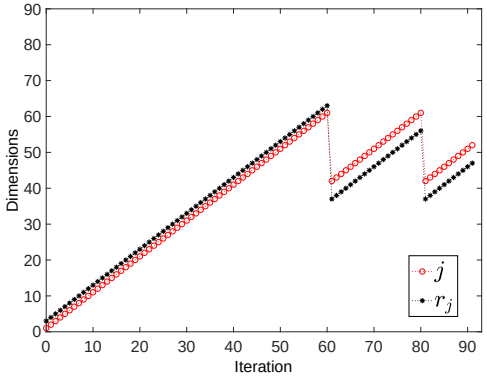
(b) Convergence history without restart.



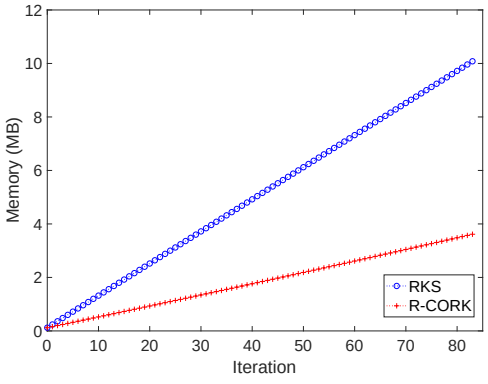
(c) Convergence history with restart.



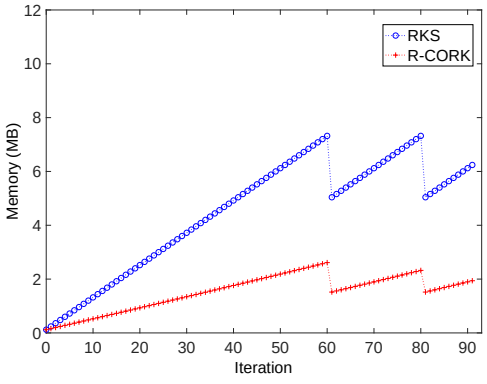
(d) Dimension of the subspace without restart.



(e) Dimension of the subspace with restart.



(f) Memory cost without restart.



(g) Memory cost with restart.

Figure 4.2: Numerical experiment 4.10.

Chapter 5

Projection methods for T-Sylvester equations

This chapter is devoted to the numerical solution of the T-Sylvester equation

$$AX + X^T B = C_1 C_2^T \tag{5.1}$$

where $A, B \in \mathbb{R}^{n\times n}$ are large and sparse coefficient matrices, the right-hand side matrix $C := C_1 C_2^T$ has low rank, this is, $C_1, C_2 \in \mathbb{R}^{n\times r}$ with $r \ll n$ and $X \in \mathbb{R}^{n\times n}$ is the unknown. Since our methods require the inversion of A and/or B, we also assume the generic condition that A and B are invertible. Moreover, (5.1) is assumed to admit a unique solution, this is, the conditions of Lemma 3.9 hold.

As we mentioned in Chapter 1, our goal is to develop projection methods that construct low-rank approximations of the solution for T-Sylvester equations. To achieve our goal, we will use subspaces based on Krylov spaces, as block Krylov or extended block Krylov subspaces.

All results presented in Chapter 5 are, at the best of our knowledge, original results and they have been published by the author of this dissertation in [35].

5.1 A general projection framework for the T-Sylvester equation

As occurs for projection methods for the Sylvester equation, although A and B are sparse, the solution matrix X is full, in general, so the storage of X requires excessive memory allocations for large-scale problems. For this reason, we look for low-rank approximations to X. This strategy is strongly supported by the link established in Theorem 3.12(b) between a T-Sylvester equation and a Sylvester equation both with low-rank right-hand sides (see Theorem 3.12(c)), together with existing results on the singular value decay of solutions to Sylvester equations with low-rank right-

hand side [43, 79], which suggests that the solution X of (5.1) can often be well approximated by a low-rank matrix.

As we discuss in Section 3.3, there exist different approaches to construct low-rank approximations to the solution of large-scale Sylvester equations with low-rank right-hand sides and often they proceed by imposing a Galerkin condition on a tensor product of low-dimensional subspaces; see [13, 55, 87, 90]. In this thesis, we follow a similar strategy for T-Sylvester equations.

Our starting point is therefore to consider approximations of the form

$$X \approx X_m = \mathbf{V}_m Y_m \mathbf{W}_m^T \in \mathbb{R}^{n\times n} \tag{5.2}$$

where the columns of $\mathbf{V}_m, \mathbf{W}_m \in \mathbb{R}^{n\times p_m}$ form orthonormal bases of subspaces $\mathbb{V}_m, \mathbb{W}_m \subset \mathbb{R}^n$, respectively:

$$\mathbb{V}_m = \text{range}(\mathbf{V}_m) \quad \text{and} \quad \mathbb{W}_m = \text{range}(\mathbf{W}_m). \tag{5.3}$$

Note that we are using bold characters to represent $\mathbf{V}_m$ and $\mathbf{W}_m$ since in the following sections we will consider them as block matrices.

The relation (5.2) states that $X_m \in \mathbb{V}_m \otimes \mathbb{W}_m$, where the tensor product of these subspaces is defined as

$$\mathbb{V}_m \otimes \mathbb{W}_m := \text{span}\{v \otimes w : v \in \mathbb{V}_m, w \in \mathbb{W}_m\}.$$

Note that the set $\mathbb{V}_m \otimes \mathbb{W}_m$ is the set of all matrices of the form (5.2). The matrix $Y_m \in \mathbb{R}^{p_m\times p_m}$, which contains the coefficients of X_m in the chosen bases, is determined by imposing a Petrov-Galerkin condition: We require the residual

$$R_m = AX_m + X_m^T B - C_1 C_2^T = A(\mathbf{V}_m Y_m \mathbf{W}_m^T) + (\mathbf{V}_m Y_m \mathbf{W}_m^T)^T B - C_1 C_2^T \tag{5.4}$$

to be orthogonal to $\mathbb{W}_m \otimes \mathbb{W}_m$ in the matrix inner product [51, Section 5.2], $R_m \perp \mathbb{W}_m \otimes \mathbb{W}_m$. Using the orthonormal basis $\mathbf{W}_m$, this becomes equivalent to requiring

$$\mathbf{W}_m^T R_m \mathbf{W}_m = 0. \tag{5.5}$$

Replacing (5.4) into (5.5) yields the small-scale T-Sylvester equation

$$(\mathbf{W}_m^T A \mathbf{V}_m) Y_m + Y_m^T (\mathbf{V}_m^T B \mathbf{W}_m) = (\mathbf{W}_m^T C_1)(\mathbf{W}_m^T C_2)^T. \tag{5.6}$$

If we assume that equation (5.6) admits a unique solution, it can be solved within $\mathcal{O}(p_m^3)$ operations using Algorithm 3.3 discussed in Section 3.2.2.

It is important to emphasize that most of the existing projection methods for the standard Sylvester equation $FX + XG = C_1 C_2^T$ use a Galerkin technique instead of a Petrov-Galerkin technique. This means that the same tensorized subspace $\mathbb{V}_m \otimes \mathbb{W}_m$ is used for searching the approximate solution and testing the residual:

$$X_m \in \mathbb{V}_m \otimes \mathbb{W}_m \quad \text{and} \quad (FX_m + X_m G - C_1 C_2^T) \perp \mathbb{V}_m \otimes \mathbb{W}_m. \tag{5.7}$$

In contrast, our projection framework for T-Sylvester equations involves two different tensorized subspaces $\mathbb{V}_m \otimes \mathbb{W}_m$ and $\mathbb{W}_m \otimes \mathbb{W}_m$ as search and test spaces, respectively. This is needed to ensure that the compressed equation (5.6) is again a T-Sylvester equation, which allows for its inexpensive solution. In the language of projection methods [84, 86], our framework for T-Sylvester equations yields *oblique projection methods*, while the one for Sylvester equations yields *orthogonal projection methods*.

The choice of the subspaces $\mathbb{V}_m$ and $\mathbb{W}_m$ is decisive and it determines the quality of the approximation X_m obtained from (5.2) and (5.6). In this thesis two different choices adapted to the structure of the T-Sylvester equation will be studied in detail in Sections 5.2 and 5.3.

5.2 Block Krylov subspaces for the T-Sylvester equation

In order to motivate the choice of adequate subspaces $\mathbb{V}_m$ and $\mathbb{W}_m$ for the projection framework from Section 5.1, we will consider the generalized Schur decomposition of the matrix pencil $A - \lambda B^T$

$$A = WT_AV^T, \quad B^T = WT_BV^T$$

where $V, W \in \mathbb{R}^{n\times n}$ are orthogonal matrices, $T_A \in \mathbb{R}^{n\times n}$ is quasi-triangular, i.e., T_A is block upper triangular with 1-by-1 and 2-by-2 blocks on the diagonal and $T_B \in \mathbb{R}^{n\times n}$ is triangular, which implies that

$$B^{-T}A = VT_B^{-1}T_AV^T \quad \text{and} \quad B^TV = WT_B. \tag{5.8}$$

If we denote by V_p the first $p < n$ columns of V, we have that the columns of V_p represents an orthonormal basis for an invariant subspace of $B^{-T}A$, provided that the subdiagonal entry $(p+1, p)$ of the upper quasi-triangular matrix T_A is zero. Moreover, if we denote by W_p the first p columns of W, we have

$$\text{range}(W_p) = \text{range}(B^TV_p) = B^T\text{range}(V_p). \tag{5.9}$$

This suggests we choose a subspace $\mathbb{V}_m$ that contains good approximations to invariant subspaces of $B^{-T}A$ and set $\mathbb{W}_m = B^T\mathbb{V}_m$. Since Krylov subspaces are known to often contain excellent approximations to invariant subspaces [86] it is natural to choose $\mathbb{V}_m$ as a Krylov subspace for the matrix $B^{-T}A$. Also, in view of (5.6), it is important to ensure that $\text{range}(C_1) \cup \text{range}(C_2) \subset \mathbb{W}_m$, in order to fully preserve the information from the right-hand side during the projection. Taking these considerations into account leads us to choose the block Krylov subspaces

$$\mathbb{V}_m = \mathcal{K}_m(B^{-T}A, B^{-T}[C_1, C_2]), \tag{5.10}$$
$$\mathbb{W}_m = B^T\mathbb{V}_m. \tag{5.11}$$

By using the definition of block Krylov subspace and the algebraic identity

$$B^T(B^{-T}A)^k B^{-T} = (AB^{-T})^k, \quad k \in \mathbb{Z},$$

we have

$$\mathbb{W}_m = \mathcal{K}_m(AB^{-T}, [C_1, C_2]), \tag{5.12}$$

which shows that, in fact, $\mathbb{W}_m$ contains the columns of C_1 and C_2. Note that the computational cost can be reduced if $\text{range}(C_1) = \text{range}(C_2)$. Indeed, in this case it holds that

$$\mathbb{V}_m = \mathcal{K}_m(B^{-T}A, B^{-T}[C_1, C_2]) = \mathcal{K}_m(B^{-T}A, B^{-T}C_1),$$

yielding a block Krylov subspace of half the dimension. In Section 5.2.2 we will discuss algorithms to construct orthonormal bases for $\mathbb{V}_m$ and $\mathbb{W}_m$.

Remark 5.1. *Note that the roles of A and B can be reversed. By transposing* (5.1)*, we obtain the T-Sylvester equation*

$$B^T X + X^T A^T = C_2 C_1^T, \tag{5.13}$$

where A is replaced by B^T, B by A^T, C_1 by C_2 and C_2 by C_1. Assuming that A is invertible, we thus arrive at the block Krylov subspaces

$$\mathbb{V}'_m = \mathcal{K}_m(A^{-1}B^T, A^{-1}[C_1, C_2]), \tag{5.14}$$
$$\mathbb{W}'_m = A\mathbb{V}'_m. \tag{5.15}$$

In Section 5.4, we will show that we can expect fast convergence of the projection method applied to (5.13) with $\mathbb{V}'_m$ and $\mathbb{W}'_m$ if $\rho(A^{-1}B^T) < 1$ and if this quantity is not too close to one, this is, all eigenvalues of $B^{-T}A$ should be located well outside the unit circle.

Therefore, we can choose between two methods depending on the location of eigenvalues of $B^{-T}A$:

1. when all eigenvalues are well inside the unit circle, use the projection method with the block Krylov subspaces $\mathbb{V}_m$, $\mathbb{W}_m$ defined in (5.10)-(5.11);

2. when all eigenvalues are well outside the unit circle, use the projection method with the block Krylov subspaces $\mathbb{V}'_m$, $\mathbb{W}'_m$ defined in (5.14)-(5.15).

In order to estimate $\rho(B^{-T}A)$ and $\rho(A^{-1}B^T)$, the power method can be used and guide the decision between 1 and 2. In Section 5.3, a variant that combines both approaches and does not require such decision will be discussed.

For simplicity, in the rest of this section, we will focus on the equation (5.1) and the block Krylov subspaces (5.10)-(5.11), but it should be kept in mind that the same developments can be applied to the transposed equation.

5.2.1 Solvability of the reduced equation

When the projection method from Section 5.1 is performed, it is required that the compressed equation (5.6) admits a unique solution. However, the unique solvability of the original equation (5.1) does not guarantee that the reduced equation has also a unique solution. Similar difficulties arise in projection methods for the solution of the Sylvester equation $FX + XG = C_1C_2^T$, where this problem is addressed by imposing the more restrictive condition that the fields of values [52, Definition 1.1.1] (instead of the spectra) of F and $-G$ are disjoint [90]. In fact, the assumption $\mathcal{F}(F)\cap\mathcal{F}(-G) = \emptyset$ is a key point to prove rigorous error bounds for rational Galerkin projection methods for the Sylvester equation [11].

Theorem 5.2 proposes an analogous condition for the T-Sylvester equation. Note that the result of Theorem 5.2 does not hold only for block Krylov subspaces, but for any pair of subspaces $\mathbb{V}_m$, $\mathbb{W}_m$ that satisfy $\mathbb{W}_m = B^T\mathbb{V}_m$. Therefore, Theorem 5.2 is also applicable to the extended block Krylov subspaces introduced in (5.20) and (5.21) and that will be used in Section 5.3.

Theorem 5.2. *Let A, $B \in \mathbb{R}^{n\times n}$ and assume that B is nonsingular. Let the columns of $\mathbf{V}_m, \mathbf{W}_m \in \mathbb{R}^{n\times p_m}$ form orthonormal bases of the two subspaces $\mathbb{V}_m$ and $\mathbb{W}_m \subset \mathbb{R}^n$, respectively, which satisfy $\mathbb{W}_m = B^T\mathbb{V}_m$. If $\mathcal{F}(AB^{-T})$ is inside the open unit disk, then*

(a) The T-Sylvester equation $AX + X^TB = C$ has a unique solution for every right-hand side C, and

(b) The T-Sylvester equation $(\mathbf{W}_m^TA\mathbf{V}_m)Y_m + Y_m^T(\mathbf{V}_m^TB\mathbf{W}_m) = \tilde{C}$ has a unique solution for every right-hand side $\tilde{C}$.

Proof. (a) Since $\mathcal{F}(AB^{-T})$ is inside the open unit disk, all eigenvalues of AB^{-T} are inside the open unit disk, therefore, the spectrum of $A - \lambda B^T$ is T-reciprocal free. Thus, part (a) follows from Lemma 3.9.

(b) Since $\mathbb{W}_m = B^T\mathbb{V}_m$, there exists a nonsingular matrix $Q_m \in \mathbb{R}^{p_m\times p_m}$ such that $B^T\mathbf{V}_m = \mathbf{W}_mQ_m$. Then,

$$\begin{aligned}\mathbf{W}_m^TA\mathbf{V}_m &= \mathbf{W}_m^TAB^{-T}\mathbf{W}_mQ_m,\\ (\mathbf{V}_m^TB\mathbf{W}_m)^T &= \mathbf{W}_m^TB^T\mathbf{V}_m = \mathbf{W}_m^T\mathbf{W}_mQ_m = Q_m.\end{aligned}$$

Thus, the pencil $(\mathbf{W}_m^TA\mathbf{V}_m) - \lambda(\mathbf{V}_m^TB\mathbf{W}_m)^T$ is strictly equivalent to the regular pencil $\mathbf{W}_m^TAB^{-T}\mathbf{W}_m - \lambda I_{p_m}$, and both pencils have the same eigenvalues. Since $\mathcal{F}(\mathbf{W}_m^TAB^{-T}\mathbf{W}_m) \subseteq \mathcal{F}(AB^{-T})$, all eigenvalues of $\mathbf{W}_m^TAB^{-T}\mathbf{W}_m$ are inside the open unit disk. Therefore, part (b) also follows from Lemma 3.9.

□

The condition that $\mathcal{F}(AB^{-T})$ is inside the open unit disk is sufficient but by no means necessary for $\rho(AB^{-T}) = \rho(B^{-T}A) < 1$ and the unique solvability of the compressed equation. As we will see in Section 5.5, in our numerical experience, we observe fast convergence of the projection method in situations where $\rho(B^{-T}A) < 1$, but $\mathcal{F}(AB^{-T})$ is much larger than the open unit disk.

5.2.2 Algorithmic details of the block Krylov subspace method

In this section, the algorithm associated with the Petrov-Galerkin projection onto the block Krylov subspaces (5.10)-(5.11) is described. Two different implementations can be chosen:

(i) We can use the block Arnoldi method [86] to compute an orthonormal basis $\mathbf{V}_m$ of $\mathbb{V}_m$ and then, an orthonormal basis $\mathbf{W}_m$ of $\mathbb{W}_m$ by orthonormalizing $B^T\mathbf{V}_m$; or

(ii) Use (5.12) to first compute an orthonormal basis $\mathbf{W}_m$ of $\mathbb{W}_m$ with the block Arnoldi method, and then, an orthonormal basis $\mathbf{V}_m$ of $\mathbb{V}_m$ by orthonormalizing $B^{-T}\mathbf{W}_m$.

Observe that the implementation (ii) does not require more linear system solves than (i), since the linear solves needed for computing $B^{-T}\mathbf{W}_m$ can be re-used by the Arnoldi method in computing $\mathbf{W}_{m+1}$. In fact, (ii) is slightly cheaper than (i), since the matrix-vector products $B^T\mathbf{V}_m$ in (i) cannot be re-used by block Arnoldi to compute $\mathbf{V}_{m+1}$. In this thesis, we will describe in detail only the implementation (i), which is similar to the one in Section 5.3 and the implementation (ii) is left to the interested reader. Also note that applying block Arnoldi twice for constructing $\mathbf{V}_m$ and $\mathbf{W}_m$ independently requires more work and does not provide us with the connection between both bases, which is needed to cheaply compute the coefficients of the compressed equations (see Proposition 5.3).

As occurs for the Arnoldi method, after m steps of the block Arnoldi method, the following block Arnoldi recurrence relation holds (see, e.g., [86, Section 6.5]):

$$B^{-T}A\mathbf{V}_m = \mathbf{V}_m\mathbf{H}_m + V_{m+1}H_{m+1,m}E_m^T = \mathbf{V}_{m+1}\underline{\mathbf{H}}_m, \tag{5.16}$$

where $\mathbf{V}_m = [V_1, V_2, \ldots, V_m] \in \mathbb{R}^{n\times 2mr}$ with $V_j \in \mathbb{R}^{n\times 2r}$, for $j = 1, \ldots, m$, is an orthonormal basis of $\mathbb{V}_m = \mathcal{K}_m(B^{-T}A, B^{-T}[C_1, C_2])$, the matrix E_m denotes the last $2r$ columns of the $2mr \times 2mr$ identity matrix, and $\mathbf{H}_m \in \mathbb{R}^{2mr\times 2mr}$, $\underline{\mathbf{H}}_m \in \mathbb{R}^{2(m+1)r\times 2mr}$ are block Hessenberg matrices with

$$\underline{\mathbf{H}}_m = \begin{bmatrix} \mathbf{H}_m \\ H_{m+1,m}E_m^T \end{bmatrix} \quad \text{where } H_{m+1,m} \in \mathbb{R}^{2r\times 2r}.$$

The matrix $\mathbf{W}_m$ can be obtained from $B^T\mathbf{V}_m$ by computing the "skinny" QR decomposition

$$\mathbf{W}_m\mathbf{Z}_m = B^T\mathbf{V}_m, \quad \mathbf{Z}_m \in \mathbb{R}^{2mr\times 2mr}, \tag{5.17}$$

where $\mathbf{W}_m = [W_1, W_2, \ldots, W_m] \in \mathbb{R}^{n\times 2mr}$ with $W_j \in \mathbb{R}^{n\times 2r}$, for $j = 1, \ldots, m$, is an orthonormal basis of $\mathbb{W}_m = B^T\mathbf{V}_m$ and $\mathbf{Z}_m$ is upper triangular. Then, the matrices

$$\mathbf{H}_{A,m} := \mathbf{W}_m^T A\mathbf{V}_m, \quad \mathbf{H}_{B,m} := \mathbf{V}_m^T B\mathbf{W}_m$$

can be cheaply obtained during the generation of the bases $\mathbf{V}_m$ and $\mathbf{W}_m$ by using (5.16) and (5.17). This means that the matrices $\mathbf{H}_{A,m}$ and $\mathbf{H}_{B,m}$ can be computed without performing explicitly the multiplication that involves the large matrices A and B, and then, they can be obtained without extra computational costs. In particular, both matrices $\mathbf{H}_{A,m}$, $\mathbf{H}_{B,m}$ can be expanded as the iteration proceeds.

Proposition 5.3. *With the notation introduced above, the following relations hold:*

$$\mathbf{H}_{A,m} = [I_{2mr},\, 0_{2mr\times 2r}]\mathbf{Z}_{m+1}\underline{\mathbf{H}}_m, \quad \textit{and} \quad \mathbf{H}_{B,m} = \mathbf{Z}_m^T. \tag{5.18}$$

Proof. Using (5.16) and (5.17) for $m+1$ we have

$$\begin{aligned}
\mathbf{H}_{A,m} &= \mathbf{W}_m^T A\mathbf{V}_m \\
&= \mathbf{W}_m^T B^T\mathbf{V}_{m+1}\underline{\mathbf{H}}_m \\
&= \mathbf{W}_m^T\mathbf{W}_{m+1}\mathbf{Z}_{m+1}\underline{\mathbf{H}}_m \\
&= [I_{2mr},\, 0_{2mr\times 2r}]\mathbf{Z}_{m+1}\underline{\mathbf{H}}_m
\end{aligned}$$

and, using (5.17), we have

$$\begin{aligned}
\mathbf{H}_{B,m} &= \mathbf{V}_m^T B\mathbf{W}_m \\
&= \mathbf{Z}_m^T\mathbf{W}_m^T\mathbf{W}_m \\
&= \mathbf{Z}_m^T.
\end{aligned}$$

□

Algorithm 5.1 gives an overview of our proposed block Krylov subspace method for solving large-scale T-Sylvester equations, which amounts to the projection method from Section 5.1 with the subspaces $\mathbb{V}_m$ and $\mathbb{W}_m$ defined as in (5.10)-(5.11).

Some comments related to the implementation of Algorithm 5.1 are listed below:

Step 2. The orthogonalization of B^TV_m with respect to the columns of $\mathbf{W}_{m-1}$ in Step 2 is performed by applying the classical Gram-Schmidt process twice.

Step 4. The matrices $\mathbf{H}_{A,m} = \mathbf{W}_m^T A\mathbf{V}_m$ and $\mathbf{H}_{B,m} = \mathbf{V}_m^T B\mathbf{W}_m$ in Step 4 are cheaply updated using the expressions presented in Proposition 5.3. Also note that (5.12) implies range$([C_1, C_2]) = \mathbb{W}_1 =$ range(W_1), and hence $\mathbf{W}_m^T C_i = [(W_1^T C_i)^T, 0, \ldots, 0]^T$ for $i = 1, 2$.

Algorithm 5.1 Block Krylov method for solving T-Sylvester equation

Input: $A, B \in \mathbb{R}^{n\times n}$ and $C_1, C_2 \in \mathbb{R}^{n\times r}$.
Output: Factors of approximate solution X_m in (5.2) of $AX + X^TB = C_1C_2^T$.
Compute V_1 orthonormal basis of range$(B^{-T}[C_1, C_2])$ and set $\mathbf{V}_0 = \mathbf{W}_0 = \emptyset$.
for $m = 1, 2, \ldots$ **do**
1. $\mathbf{V}_m = [\mathbf{V}_{m-1}, V_m]$.
2. Orthonormalize the columns of B^TV_m with respect to $\mathbf{W}_{m-1}$ to obtain W_m.
3. $\mathbf{W}_m = [\mathbf{W}_{m-1}, W_m]$.
4. Expand $\mathbf{H}_{A,m} = \mathbf{W}_m^TA\mathbf{V}_m$, $\mathbf{H}_{B,m} = \mathbf{V}_m^TB\mathbf{W}_m$ by using Proposition 5.3 and $\tilde{C}_m = (\mathbf{W}_m^TC_1)(\mathbf{W}_m^TC_2)^T$.
5. Compute Y_m solution of $\mathbf{H}_{A,m}Y_m + Y_m^T\mathbf{H}_{B,m} = \tilde{C}_m$ via Algorithm 3.3 with $\star = T$.
if converged **then**
Return $\mathbf{V}_m, Y_m, \mathbf{W}_m$ such that $X_m = \mathbf{V}_mY_m\mathbf{W}_m^T$ and stop.
end if
6. Orthonormalize the columns of $B^{-T}AV_m$ with respect to $\mathbf{V}_m$ to obtain V_{m+1}.
end for

Step 6. One step of the standard block Arnoldi method presented in [86, Algorithm 6.8] is used in Step 6: it requires $2r$ matrix-vector products with A, followed by $2r$ solves with B^T. If B is a sparse matrix, the latter can be implemented efficiently by computing and storing a sparse LU factorization of B^T beforehand. Solves with B^T then only require forward/backward substitutions with the sparse LU factors. For simplicity, we assume that breakdowns do not occur in the block Arnoldi method, and thus the matrices $\mathbf{V}_m, \mathbf{W}_m \in \mathbb{R}^{n\times 2mr}$ are of full rank and have orthonormal columns.

Stopping criterion. To check convergence, the stopping criterion

$$\frac{\|AX_m + X_m^TB - C_1C_2^T\|_F}{(\|A\|_F + \|B\|_F)\|Y_m\|_F + \|C_1C_2^T\|_F} < \texttt{tol} \tag{5.19}$$

is used, where `tol` is a fixed user-specified tolerance, $X_m = \mathbf{V}_mY_m\mathbf{W}_m^T$, and $\|X_m\|_F = \|Y_m\|_F$ has been used. Analogous stopping criteria are used in other algorithms for matrix equations [89, p. 1275], [48, Chapter 16].

The residual norm $\|AX_m + X_m^TB - C_1C_2^T\|_F$ is computed inexpensively via the result obtained in Proposition 5.4, which uses the information obtained in Steps 2 and 6. Observe that the computation of the quantities $\|A\|_F$ and $\|B\|_F$ in (5.19) needs to be performed only once and is inexpensive if A and B are sparse; also notice that to compute $\|C_1C_2^T\|_F$ we use the expression

$$\|C_1C_2^T\|_F^2 = \text{trace}\left((C_1C_2^T)^TC_1C_2^T\right) = \text{trace}\left((C_1^TC_1)(C_2^TC_2)\right),$$

which only involves the small $r \times r$ matrices $C_1^T C_1$ and $C_2^T C_2$.

From the expressions in (5.18) we can construct $\mathbf{H}_{A,m}$ at the cost of a small matrix multiplication and $\mathbf{H}_{B,m}$ at no additional cost. Also note that, in (5.18), the matrix $\mathbf{H}_{A,m}$ is a block Hessenberg matrix and $\mathbf{H}_{B,m}^T$ is upper triangular. To a certain extent, this structure can be exploited to reduce the cost of the QZ algorithm [44, Section 7.7] for computing the generalized Schur decomposition needed when solving the compressed equation. A minor complication in (5.18) for $\mathbf{H}_{A,m}$ is that it requires $\mathbf{Z}_{m+1}$, which only becomes available after the orthonormalization of $B^T\mathbf{V}_{m+1}$ has been performed. This issue, however, it is easily addressed by slightly reorganizing Algorithm 5.1.

Proposition 5.4 gives an expression for the residual norm that requires the computation of the Frobenius norm for a small $2r \times 2mr$ matrix only, in the spirit of a corresponding result for Lyapunov equations [89, Theorem 2.1].

Proposition 5.4. *With the notation introduced in Proposition 5.3, the following relation holds for the residual norm:*

$$\|AX_m + X_m^T B - C_1 C_2^T\|_F = \|Z_{m+1,m+1} H_{m+1,m} E_m^T Y_m\|_F,$$

where $Z_{m+1,m+1}$ is the trailing $2r \times 2r$ principal submatrix of $\mathbf{Z}_{m+1}$.

Proof. Set $R_m := AX_m + X_m^T B - C_1 C_2^T = A\mathbf{V}_m Y_m \mathbf{W}_m^T + \mathbf{W}_m Y_m^T \mathbf{V}_m^T B - C_1 C_2^T$ and let the columns of $\mathbf{W}_{m,\perp}$ form an orthonormal basis for $\mathbb{W}_m^\perp$, such that the first $2r$ columns of $\mathbf{W}_{m,\perp}$ coincide with W_{m+1}. Then we have $\mathbf{W}_{m,\perp}^T \mathbf{W}_m = 0$, $\mathbf{W}_{m,\perp}^T C_1 = \mathbf{W}_{m,\perp}^T C_2 = 0$ and the relations (5.16)-(5.17) imply:

$$\begin{aligned}
\|\mathbf{W}_{m,\perp}^T R_m \mathbf{W}_m\|_F &= \|\mathbf{W}_{m,\perp}^T A \mathbf{V}_m Y_m\|_F \\
&= \|\mathbf{W}_{m,\perp}^T B^T \mathbf{V}_{m+1} \underline{\mathbf{H}}_m Y_m\|_F \\
&= \|\mathbf{W}_{m,\perp}^T \mathbf{W}_{m+1} \mathbf{Z}_{m+1} \underline{\mathbf{H}}_m Y_m\|_F \\
&= \| E_{m+1}^T \mathbf{Z}_{m+1} \underline{\mathbf{H}}_m Y_m\|_F \\
&= \|Z_{m+1,m+1} H_{m+1,m} E_m^T Y_m\|_F.
\end{aligned}$$

In a similar way,

$$\begin{aligned}
\|\mathbf{W}_m^T R_m \mathbf{W}_{m,\perp}\|_F &= \|Y_m^T \mathbf{V}_m^T B \mathbf{W}_{m,\perp}\|_F \\
&= \|Y_m^T \mathbf{Z}_m^T \mathbf{W}_m^T \mathbf{W}_{m,\perp}\|_F \\
&= 0
\end{aligned}$$

and $\|\mathbf{W}_{m,\perp}^T R_m \mathbf{W}_{m,\perp}\|_F = 0$. Using that the Petrov-Galerkin condition (5.5) implies that $\|\mathbf{W}_m^T R_m \mathbf{W}_m\|_F = 0$, we obtain

$$\|R_m\|_F^2 = \|\mathbf{W}_m^T R_m \mathbf{W}_m\|_F^2 + \|\mathbf{W}_{m,\perp}^T R_m \mathbf{W}_m\|_F^2 + \|\mathbf{W}_m^T R_m \mathbf{W}_{m,\perp}\|_F^2$$

$$
\begin{aligned}
& +\|\mathbf{W}_{m,\perp}^T R_m \mathbf{W}_{m,\perp}\|_F^2 \\
= \; & \|\mathbf{W}_{m,\perp}^T R_m \mathbf{W}_m\|_F^2 \\
= \; & \|Z_{m+1,m+1} H_{m+1,m} E_m^T Y_m\|_F^2
\end{aligned}
$$

which completes the proof. □

5.3 Extended Krylov subspaces for the T-Sylvester equation

Extended Krylov subspaces are often used in modern algorithms for solving large-scale Sylvester equations (see [89, 90] and the references therein). A motivation to use these subspaces is to attain spaces that have smaller dimension compared to standard Krylov spaces but contain equally good approximations of the solution. This results in decreased storage requirements for the dense matrices $\mathbf{V}_m$ and $\mathbf{W}_m$ needed to get the approximate solution (5.2). In addition, in the case of standard Sylvester equations, it has been observed that these enriched spaces exhibit an impressive performance despite the fact that multiplication by inverses, i.e., solution of large linear systems, is involved in their construction, in contrast to standard Krylov spaces. In the case of the T-Sylvester equation $AX + X^T B = C_1 C_2^T$ the use of extended Krylov subspaces is even more natural, since the standard block Krylov spaces (5.10) and (5.11) adapted to this equation already involve inverses. Other methods using enriched Krylov subspaces suggest themselves, such as Rational Krylov space and ADI methods (see, e.g., [90] and also the discussion in Section 5.4).

Following [89, Section 3], in this section we propose the use of extended block Krylov subspaces to implement the projection method in Section 5.1:

$$
\begin{aligned}
\mathbb{V}_m &= \mathcal{K}_m(B^{-T}A, B^{-T}[C_1, C_2]) + \mathcal{K}_{m+1}((B^{-T}A)^{-1}, B^{-T}[C_1, C_2]), && (5.20)\\
\mathbb{W}_m &= B^T \mathbb{V}_m. && (5.21)
\end{aligned}
$$

Let $\mathbf{C}_B = B^{-T}[C_1, C_2] \in \mathbb{R}^{n\times 2r}$. The subspace $\mathbb{V}_m$ is iteratively expanded with two blocks of vectors at the time by an Arnoldi-type process as

$$
[\mathbf{C}_B, (B^{-T}A)^{-1}\mathbf{C}_B], [(B^{-T}A)\mathbf{C}_B, (B^{-T}A)^{-2}\mathbf{C}_B], [(B^{-T}A)^2\mathbf{C}_B, (B^{-T}A)^{-3}\mathbf{C}_B], \ldots
$$

where for each pair, the first block of vectors expands the space in $B^{-T}A$, while the second block expands the space in $(B^{-T}A)^{-1} = A^{-1}B^T$.The actual basis is computed by orthogonalizing the newly built vectors by means of the Gram-Schmidt process [89]. Observe that $2r + 2r$ vectors are added to the basis during each iteration. After m iterations, the matrix $\mathbf{V}_m$ whose columns form an orthonormal basis of $\mathbb{V}_m$ is iteratively generated:

$$
\mathbf{V}_m = [\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_m] \in \mathbb{R}^{n\times 4mr}, \quad \mathcal{V}_j = [V_j^{(1)}, V_j^{(2)}] \in \mathbb{R}^{n\times 4r}, \tag{5.22}
$$

where the "odds" $n \times 2r$ matrices $V_j^{(1)}$ originally come from multiplications with $B^{-T}A$, while the "even" $n \times 2r$ matrices $V_j^{(2)}$ come from multiplications with $A^{-1}B^T$. Keep in mind that, in the following, we shall heavily rely on this type of partitioning to build key recurrence results.

As occurs for the block Krylov subspace method, by using the definition of block Krylov subspaces and the matrix identity $B^T(B^{-T}A)^kB^{-T} = (AB^{-T})^k, k \in \mathbb{Z}$, we obtain

$$\mathbb{W}_m = \mathcal{K}_m(AB^{-T}, [C_1, C_2]) + \mathcal{K}_{m+1}((AB^{-T})^{-1}, [C_1, C_2]). \tag{5.23}$$

From (5.23) it is immediate to see that

$$AB^{-T}\mathbb{W}_m \subseteq \mathbb{W}_{m+1}. \tag{5.24}$$

These relations show that there is nothing special about the matrix equation (5.21), since the same properties could be obtained by explicitly building the two extended subspaces $\mathbb{V}_m$ and $\mathbb{W}_m$ as in (5.20) and (5.23), respectively. On the other hand, as already mentioned, using (5.21) allows one to avoid extra system solves and provides us with the connection between orthonormal bases of both spaces to cheaply compute the coefficients of the reduced equations. Note that (5.23) allows one to construct first an orthonormal basis $\mathbf{W}_m$ of $\mathbb{W}_m$ and then use $\mathbb{V}_m = B^{-T}\mathbb{W}_m$ for computing an orthonormal basis of $\mathbb{V}_m$. However, this approach requires more system solves than the approach we follow below, since only half of the solves in $B^{-T}\mathbf{W}_m$ can be re-used for expanding $\mathbb{W}_m$ to $\mathbb{W}_{m+1}$.

We denote by $\mathbf{V}_m \in \mathbb{R}^{n \times 4mr}$ the orthonormal basis of $\mathbb{V}_m$ in (5.22), and we compute an orthonormal basis $\mathbf{W}_m \in \mathbb{R}^{n \times 4mr}$ of $\mathbb{W}_m$ by using again the "skinny" QR decomposition

$$\mathbf{W}_m\mathbf{Z}_m = B^T\mathbf{V}_m, \tag{5.25}$$

where we can see that

$$\mathbf{H}_{B,m} := \mathbf{V}_m^T B\mathbf{W}_m = \mathbf{Z}_m^T.$$

As occurred for the block Krylov subspace method, the derivation of an updating expression for $\mathbf{H}_{A,m} := \mathbf{W}_m^T A\mathbf{V}_m$ which does not involve multiplications with A may be obtained as follows: From (5.20), we obtain that $B^{-T}A\mathbb{V}_m \subseteq \mathbb{V}_{m+1}$, and this implies that there exists a $4(m+1)r \times 4mr$ block Hessenberg matrix $\underline{\mathbf{K}}_m$ such that $B^{-T}A\mathbf{V}_m = \mathbf{V}_{m+1}\underline{\mathbf{K}}_m$. Note that the matrix $\underline{\mathbf{K}}_m$ is not obtained directly from the construction of the extended Krylov space $\mathbb{V}_m$, in constrast with the situation for standard Krylov spaces, however $\underline{\mathbf{K}}_m$ can be computed without extra large matrix-vector multiplications and system solves via the recursion presented in [89, Proposition 3.2]. Therefore, with (5.25), we have

$$\begin{aligned} \mathbf{H}_{A,m} &= \mathbf{W}_m^T A\mathbf{V}_m \\ &= \mathbf{W}_m^T B^T(B^{-T}A\mathbf{V}_m) \\ &= \mathbf{W}_m^T B^T\mathbf{V}_{m+1}\underline{\mathbf{K}}_m \end{aligned}$$

$$\begin{aligned} &= \mathbf{W}_m^T \mathbf{W}_{m+1} \mathbf{Z}_{m+1} \underline{\mathbf{K}}_m \\ &= [I_{4mr},\, 0] \mathbf{Z}_{m+1} \underline{\mathbf{K}}_m. \end{aligned}$$

This method for computing $\mathbf{H}_{m,A}$ is very efficient since it only involves a small matrix product and allows one to easily expand $\mathbf{H}_{A,m}$ as the iteration proceeds. Note also that the expression above shows that $\mathbf{H}_{A,m}$ is block upper Hessenberg.

However, we develop another method for computing $\mathbf{H}_{A,m}$ which is slightly more efficient, since it adapts the recursion in [89, Proposition 3.2] to computing directly $\mathbf{H}_{A,m}$ instead of $\underline{\mathbf{K}}_m$. In order to explain this method in detail, we need to introduce some additional notation. First, we partition the upper triangular matrix $\mathbf{Z}_m = (Z_{ij})_{1\le i,j\le m} \in \mathbb{R}^{4mr\times 4mr}$ with $Z_{ij} \in \mathbb{R}^{4r\times 4r}$ as $Z_{ij} = [Z_{ij}^{(1)}, Z_{ij}^{(2)}]$, with $Z_{ij}^{(1)}, Z_{ij}^{(2)} \in \mathbb{R}^{4r\times 2r}$, and define the two block upper triangular matrices

$$\mathbf{Z}_m^{(1)} = (Z_{ij}^{(1)})_{1\le i,j\le m}, \quad \mathbf{Z}_m^{(2)} = (Z_{ij}^{(2)})_{1\le i,j\le m} \in \mathbb{R}^{4mr\times 2mr}. \tag{5.26}$$

Second, we observe that the orthonormalization process associated with the construction of $\mathbf{V}_m$ generates the block upper Hessenberg matrix

$$\underline{\mathbf{H}}_m = (H_{ij}) \in \mathbb{R}^{4(m+1)r\times 4mr}, \quad \text{where } H_{ij} \in \mathbb{R}^{4r\times 4r}, \tag{5.27}$$

which satisfies

$$[B^{-T}AV_1^{(1)}, (B^{-T}A)^{-1}V_1^{(2)}, \ldots, B^{-T}AV_m^{(1)}, (B^{-T}A)^{-1}V_m^{(2)}] = \mathbf{V}_{m+1}\underline{\mathbf{H}}_m. \tag{5.28}$$

The subdiagonal blocks $H_{j+1,j}$ are upper triangular, and we consider them partitioned as follows:

$$H_{j+1,j} = \begin{bmatrix} \chi_{11}^{(j)} & \chi_{12}^{(j)} \\ 0 & \chi_{22}^{(j)} \end{bmatrix}, \quad \text{with } \chi_{11}^{(j)}, \chi_{12}^{(j)}, \chi_{22}^{(j)} \in \mathbb{R}^{2r\times 2r}. \tag{5.29}$$

Submatrices $\underline{\mathbf{H}}_m^{(1)} = (H_{ij}^{(1)})$ and $\underline{\mathbf{H}}_m^{(2)} = (H_{ij}^{(2)})$ of $\underline{\mathbf{H}}_m$ analogous to those in (5.26) will also be used. This method is summarized in Proposition 5.5.

Proposition 5.5. *Let $\mathbf{H}_{A,m} := \mathbf{W}_m^T A \mathbf{V}_m \in \mathbb{R}^{4mr\times 4mr}$ be partitioned in blocks as $\mathbf{H}_{A,m} = (\mathtt{h}_{ij})_{1\le i,j\le m}$, where $\mathtt{h}_{ij} \in \mathbb{R}^{4r\times 4r}$. Let each block $\mathtt{h}_{ij}$ be partitioned as $\mathtt{h}_{ij} = \left[\mathtt{h}_{ij}^{(1)},\ \mathtt{h}_{ij}^{(2)}\right]$, where $\mathtt{h}_{ij}^{(1)}, \mathtt{h}_{ij}^{(2)} \in \mathbb{R}^{4r\times 2r}$, and define the following submatrices of $\mathbf{H}_{A,m}$ (separating the odd and even block columns of $\mathbf{H}_{A,m}$)*

$$\mathtt{h}^{(1)} = \left(\mathtt{h}_{ij}^{(1)}\right)_{1\le i,j\le m} \in \mathbb{R}^{4mr\times 2mr} \qquad \text{and} \qquad \mathtt{h}^{(2)} = \left(\mathtt{h}_{ij}^{(2)}\right)_{1\le i,j\le m} \in \mathbb{R}^{4mr\times 2mr}.$$

By using the matrices defined in (5.22)-(5.25)-(5.27), the following results hold.

(a) $\mathtt{h}^{(1)} = Z_{1:m,1:(m+1)}\, \underline{\mathbf{H}}_m^{(1)}$.

(b) *With* $E_1 = [I_{2r}; 0]^T \in \mathbb{R}^{4mr\times 2r}$, *let* $w_1 = \mathbf{W}_m E_1$ *be the matrix of the first* $2r$ *columns of* $\mathbf{W}_m$, *and let*

$$[B^{-T}[C_1, C_2]\,,\; A^{-1}[C_1, C_2]] = \mathcal{V}_1 \begin{bmatrix} \chi_{11}^{(0)} & \chi_{12}^{(0)} \\ 0 & \chi_{22}^{(0)} \end{bmatrix}, \qquad \chi_{11}^{(0)},\, \chi_{12}^{(0)},\, \chi_{22}^{(0)} \in \mathbb{R}^{2r\times 2r} \tag{5.30}$$

be the QR decomposition of the matrix in the left-hand side. Then

$$\mathtt{h}_{1:m,1}^{(2)} = \left(E_1\, w_1^T\,[C_1,\, C_2] \;-\; \mathtt{h}_{1:m,1}^{(1)}\, \chi_{12}^{(0)}\right) \left(\chi_{22}^{(0)}\right)^{-1}, \tag{5.31}$$

$$\mathtt{h}_{1:m,k+1}^{(2)} = \left(Z_{1:m,k}^{(2)} \;-\; \mathtt{h}_{1:m,1:k}\, H_{1:k,k}^{(2)} \;-\; \mathtt{h}_{1:m,k+1}^{(1)}\, \chi_{12}^{(k)}\right) \left(\chi_{22}^{(k)}\right)^{-1}, \tag{5.32}$$

$$k = 1, \ldots, m-1. \tag{5.33}$$

Proof. To prove (a), let $\mathbf{V}_m^{(1)} := [V_1^{(1)}, \ldots, V_m^{(1)}]$ and notice that (5.28) implies

$$B^{-T} A\, \mathbf{V}_m^{(1)} = \mathbf{V}_{m+1}\, \underline{\mathbf{H}}_m^{(1)} \quad \text{and} \quad A\, \mathbf{V}_m^{(1)} = B^T \mathbf{V}_{m+1}\, \underline{\mathbf{H}}_m^{(1)}.$$

Therefore, $\mathtt{h}^{(1)} = \mathbf{W}_m^T A \mathbf{V}_m^{(1)} = \mathbf{W}_m^T B^T \mathbf{V}_{m+1} \underline{\mathbf{H}}_m^{(1)} = \mathbf{W}_m^T \mathbf{W}_{m+1} \mathbf{Z}_{m+1} \underline{\mathbf{H}}_m^{(1)}$, by using (5.25) for $m+1$. So $\mathtt{h}^{(1)} = [I_{4mr},\, 0_{4mr\times 4r}] \mathbf{Z}_{m+1} \underline{\mathbf{H}}_m^{(1)}$, and the result follows.

Let us prove (b). To obtain (5.31), we start by equating the second block columns in (5.30) to get $A^{-1}[C_1,\, C_2] = V_1^{(1)} \chi_{12}^{(0)} + V_1^{(2)} \chi_{22}^{(0)}$, which implies

$$A V_1^{(2)} = \left([C_1,\, C_2] - A V_1^{(1)} \chi_{12}^{(0)}\right) \left(\chi_{22}^{(0)}\right)^{-1}.$$

Therefore

$$(\mathtt{h}^{(2)})_{1:m,1} = \mathbf{W}_m^T A V_1^{(2)} = (\mathbf{W}_m^T [C_1,\, C_2] - \mathbf{W}_m^T A V_1^{(1)} \chi_{12}^{(0)})\, (\chi_{22}^{(0)})^{-1}.$$

Equation (5.31) follows from observing that $\mathbf{W}_m^T A V_1^{(1)} = (\mathtt{h}^{(1)})_{1:m,1}$ and that, from (5.25), $\mathrm{range}(w_1) = B^T \mathrm{range}(V_1^{(1)}) = \mathrm{range}([C_1,\, C_2])$. Next, we focus on (5.32). From (5.28), we obtain

$$\begin{aligned} (B^{-T}A)^{-1} V_k^{(2)} &= \mathbf{V}_{m+1} H_{1:(m+1),k}^{(2)} = \mathbf{V}_k H_{1:k,k}^{(2)} + V_{k+1} H_{k+1,k}^{(2)} \\ &= \mathbf{V}_k H_{1:k,k}^{(2)} + [V_{k+1}^{(1)}\; V_{k+1}^{(2)}] \begin{bmatrix} \chi_{12}^{(k)} \\ \chi_{22}^{(k)} \end{bmatrix}, \end{aligned}$$

where (5.29) has been used. This equation implies, after some manipulations,

$$\begin{aligned} A\, V_{k+1}^{(2)} &= (B^T\, V_k^{(2)} - A\, \mathbf{V}_k H_{1:k,k}^{(2)} - A\, V_{k+1}^{(1)}\, \chi_{12}^{(k)})\, (\chi_{22}^{(k)})^{-1} \\ &= (\mathbf{W}_m\, Z_{1:m,k}^{(2)} - A\, \mathbf{V}_k H_{1:k,k}^{(2)} - A\, V_{k+1}^{(1)}\, \chi_{12}^{(k)})\, (\chi_{22}^{(k)})^{-1}, \end{aligned} \tag{5.34}$$

where we used again (5.25). Equation (5.32) follows from combining (5.34) with $\mathtt{h}_{1:m,k+1}^{(2)} = \mathbf{W}_m^T A V_{k+1}^{(2)}$, and the definition of $\mathbf{H}_{A,m}$. □

The method presented in Proposition 5.5 is the one we have used in all numerical tests presented in Section 5.5. Algorithm 5.2 presents the projection algorithm in Section 5.1 for the subspaces defined in (5.20) and (5.21) and the part concerning with the construction of the extended Krylov subspace $\mathbb{V}_m$ is based on the algorithm introduced in [89, Section 3]. Technical comments similar to those made after Algorithm 5.1 apply likewise to Algorithm 5.2. In particular, the same stopping criterion (5.19) is used here. Observe that the product $(B^{-T}A)^{-1}V_m^{(2)} = A^{-1}B^TV_m^{(2)}$ in step 6 is computed by first multiplying the columns of $V_m^{(2)}$ with B^T and then $2r$ linear systems with A are solved.

Algorithm 5.2 Extended block Krylov method for solving T-Sylvester equation

Input: $A, B \in \mathbb{R}^{n\times n}$ and $C_1, C_2 \in \mathbb{R}^{n\times r}$.
Output: Factors of approximate solution X_m of $AX + X^TB = C_1C_2^T$.
Compute V_1 orthonormal basis of range($[B^{-T}[C_1, C_2]\,,\ A^{-1}[C_1, C_2]]$), and set $\mathbf{V}_0 = \mathbf{W}_0 = \emptyset$.
for $m = 1, 2, \ldots,$ **do**
1. $\mathbf{V}_m = [\mathbf{V}_{m-1},\ V_m]$.
2. Orthonormalize the columns of B^TV_m with respect to $\mathbf{W}_{m-1}$ to obtain W_m.
3. $\mathbf{W}_m = [\mathbf{W}_{m-1},\ W_m]$.
4. Set $\mathbf{H}_{A,m} = \mathbf{W}_m^TA\mathbf{V}_m$, $\mathbf{H}_{B,m} = \mathbf{V}_m^TB\mathbf{W}_m$, and $\tilde{C}_m = (\mathbf{W}_m^TC_1)(\mathbf{W}_m^TC_2)^T$.
5. Compute Y_m solution of $\mathbf{H}_{A,m}Y_m + Y_m^T\mathbf{H}_{B,m} = \tilde{C}_m$ via Algorithm 3.3 with $\star = T$.
if converged **then**
Return $\mathbf{V}_m, Y_m, \mathbf{W}_m$ such that $X_m = \mathbf{V}_mY_m\mathbf{W}_m^T$ and stop.
end if
6. $V'_{m+1} = [B^{-T}A\,V_m^{(1)}\,,\ (B^{-T}A)^{-1}V_m^{(2)}]$ where $V_m = [V_m^{(1)}, V_m^{(2)}]$.
7. Orthonormalize the columns of V'_{m+1} with respect to $\mathbf{V}_m$ to obtain V_{m+1}.
end for

Proposition 5.6 shows that, in a similar manner as for the methods in Section 5.2, the residual norm can be computed without explicitly storing the residual matrix, and, thus, without performing any multiplication by the large matrices A and B.

Proposition 5.6. *Let $\mathbf{H}_{A,m} := \mathbf{W}_m^TA\mathbf{V}_m \in \mathbb{R}^{4mr\times 4mr}$ be partitioned in blocks as $\mathbf{H}_{A,m} = (\mathtt{h}_{ij})_{1\le i,j\le m}$, where $\mathtt{h}_{ij} \in \mathbb{R}^{4r\times 4r}$, and let E_m be the matrix of the last $4r$ columns of I_{4mr}. Then, with the notation in Algorithm 5.2, the residual matrix $R_m = AX_m + X_m^TB - C_1C_2^T$ satisfies*

$$\|R_m\|_F = \|\mathtt{h}_{m+1,m}\,E_m^T\,Y_m\|_F.$$

Proof. We have $R_m = A\mathbf{V}_mY_m\mathbf{W}_m^T + \mathbf{W}_mY_m^T\mathbf{V}_m^TB - C_1C_2^T$. Combining (5.24) and (5.25) it follows that range($A\mathbf{V}_m$) $\subseteq \mathbb{W}_{m+1}$. In addition, range($\mathbf{W}_m$) $\subseteq \mathbb{W}_{m+1}$

and range$(C_1) \subseteq \mathbb{W}_{m+1}$ by (5.23). Therefore, range$(R_m) \subseteq \mathbb{W}_{m+1}$. An analogous argument shows that range$(R_m^T) \subseteq \mathbb{W}_m$, by using again (5.25) and (5.23). As a consequence we can write $R_m = \mathbf{W}_{m+1}\widehat{R}_m\mathbf{W}_m^T$. Thus, $\widehat{R}_m = \mathbf{W}_{m+1}^T R_m \mathbf{W}_m$ and by (5.5)

$$\widehat{R}_m = \begin{bmatrix} \mathbf{W}_m^T R_m \mathbf{W}_m \\ W_{m+1}^T R_m \mathbf{W}_m \end{bmatrix} = \begin{bmatrix} 0 \\ W_{m+1}^T R_m \mathbf{W}_m \end{bmatrix}.$$

Since $W_{m+1}^T \mathbf{W}_m = 0$ and $W_{m+1}^T C_1 = 0$, because range$(C_1) \subseteq \mathbb{W}_1 =$ range(W_1) by (5.23), we have $W_{m+1}^T R_m \mathbf{W}_m = W_{m+1}^T A \mathbf{V}_m Y_m = \mathtt{h}_{m+1,1:m} Y_m$. Besides, $\mathbf{H}_{A,m+1}$ is block upper Hessenberg, which implies that $\mathtt{h}_{m+1,1:m} = \mathtt{h}_{m+1,m} E_m^T$, which proves the result, since $\|R_m\|_F = \|\widehat{R}_m\|_F$. □

5.4 Relation to a T-Stein equation and a fixed point iteration

In this section, another motivation for choosing the Krylov subspaces (5.10)-(5.11) is provided, which offers some intuition on the expected convergence of Algorithm 5.1. This intuition is fully confirmed by the numerical tests presented in Section 5.5.

Since we assume that B is nonsingular, the T-Sylvester equation (5.1) is equivalent to

$$X^T = C_1 C_2^T B^{-1} - A X B^{-1}.$$

Transposing this equation yields the fixed point equation

$$X = B^{-T} C_2 C_1^T - B^{-T} X^T A^T. \tag{5.35}$$

Matrix equations of this form are usually called T-Stein equations [20]. The following theorem shows that the iterates produced by the fixed point iteration applied to (5.35) are contained in the tensor product $\mathbb{V}_m \otimes \mathbb{W}_m$ of the block Krylov subspaces in (5.10) and (5.11).

Theorem 5.7. *Let $A, B \in \mathbb{R}^{n\times n}$, $C_1, C_2 \in \mathbb{R}^{n\times r}$, and assume that B is nonsingular. Consider the sequence of matrices $\{\tilde{X}_m\}_{m=0}^{\infty}$ defined by the fixed point iteration*

$$\tilde{X}_0 = 0, \qquad \tilde{X}_{m+1} = B^{-T} C_2 C_1^T - B^{-T} \tilde{X}_m^T A^T, \quad \text{for } m = 0, 1, 2, \ldots \tag{5.36}$$

Then it holds that $\tilde{X}_{m+1} \in \mathbb{V}_m \otimes \mathbb{W}_m$ for $m = 2, 3, \ldots$, where

$$\mathbb{V}_m = \mathcal{K}_m(B^{-T}A, B^{-T}[C_1, C_2]) \quad \text{and} \quad \mathbb{W}_m = \mathcal{K}_m(AB^{-T}, [C_1, C_2]).$$

Proof. We will show by induction that range$(\tilde{X}_{m+1}) \subseteq \mathbb{V}_m$ and range$(\tilde{X}_{m+1}^T) \subseteq \mathbb{W}_m$; these two conditions taken together are equivalent to $\tilde{X}_{m+1} \in \mathbb{V}_m \otimes \mathbb{W}_m$.

The first three iterates of (5.36) are given by

$$\tilde{X}_1 = B^{-T} C_2 C_1^T,$$

$$\tilde{X}_2 = B^{-T}C_2C_1^T - B^{-T}C_1C_2^T(AB^{-T})^T\,,$$
$$\tilde{X}_3 = B^{-T}C_2C_1^T - B^{-T}C_1C_2^T(AB^{-T})^T - (B^{-T}A)\,B^{-T}C_2C_1^T(AB^{-T})^T\,.$$

By inspection, $\operatorname{range}(\tilde{X}_3) \subseteq \mathbb{V}_2$ and $\operatorname{range}(\tilde{X}_3^T) \subseteq \mathbb{W}_2$.

Suppose now that $\operatorname{range}(\tilde{X}_m) \subseteq \mathbb{V}_{m-1}$ and $\operatorname{range}(\tilde{X}_m^T) \subseteq \mathbb{W}_{m-1}$ hold for some $m \geq 3$. Together with (5.36), this implies

$$\operatorname{range}(\tilde{X}_{m+1}^T) \subseteq \operatorname{range}(C_1) + \operatorname{range}(A\tilde{X}_m) \subseteq \operatorname{range}(C_1) + A\,\mathbb{V}_{m-1}.$$

Using

$$A\,\mathbb{V}_{m-1} = \operatorname{range}\big(\big[\,AB^{-T}[C_1,C_2]\,,\,(AB^{-T})^2[C_1,C_2],\ldots,(AB^{-T})^{m-1}[C_1,C_2]\,\big]\big) \subseteq \mathbb{W}_m,$$

we therefore obtain

$$\operatorname{range}(\tilde{X}_{m+1}^T) \subseteq \mathbb{W}_1 + \mathbb{W}_m = \mathbb{W}_m.$$

Analogously,

$$\begin{aligned}\operatorname{range}(\tilde{X}_{m+1}) &\subseteq \operatorname{range}(B^{-T}C_2) + \operatorname{range}(B^{-T}\tilde{X}_m^T)\\ &\subseteq \mathbb{V}_1 + B^{-T}\mathbb{W}_m = \mathbb{V}_1 + \mathbb{V}_m = \mathbb{V}_m,\end{aligned}$$

where we have used (5.11). We have thus shown that the statement of the theorem holds for $m+1$, which completes the induction proof. □

Theorem 5.7 shows that there is a $2mr \times 2mr$ matrix $\tilde{Y}_m$ such that

$$\tilde{X}_{m+1} = \mathbf{V}_m\tilde{Y}_m\mathbf{W}_m^T,$$

for orthonormal bases $\mathbf{V}_m, \mathbf{W}_m$ of $\mathbb{V}_m, \mathbb{W}_m$ respectively. In particular, $\tilde{X}_{m+1}$ has rank at most $2mr$. Although the iteration (5.36) itself operates with full $n \times n$ matrices, it is certainly possible to develop a low-rank variant, similar to the low-rank Smith method [78] for Lyapunov equations. More importantly, $\tilde{X}_{m+1}$ is in general different from the approximation X_m obtained when applying the projection method from Section 5.1 with the same block Krylov subspaces $\mathbb{V}_m, \mathbb{W}_m$. Although, $\tilde{X}_{m+1}$ and X_m are both contained in $\mathbb{V}_m \otimes \mathbb{W}_m$, the Petrov-Galerkin condition (5.5) will usually pick an approximation different from $\tilde{X}_{m+1}$. Still we believe that the established link to the fixed point iteration (5.36) also offers some intuition on the expected convergence of the projection method.

Establishing sufficient conditions that guarantee fast convergence of the projection method is likely a difficult problem and it is not considered in this thesis. However, the convergence analysis of the fixed point iteration (5.36) is very simple. Observe that although in Theorem 5.8 the spectral norm is used, any other submultiplicative matrix norm may also be considered.

Theorem 5.8. *Let $A, B \in \mathbb{R}^{n\times n}$, $C_1, C_2 \in \mathbb{R}^{n\times r}$, and assume that B is nonsingular and $\rho(B^{-T}A) < 1$. Then the following statements hold:*

(a) *the T-Sylvester equation* (5.1) (*or, equivalently, the T-Stein equation* (5.35)) *has a unique solution* X*;*

(b) *the sequence of matrices* $\{\tilde{X}_m\}_{m=0}^{\infty}$ *defined by the fixed point iteration* (5.36) *converges to* X*;*

(c) *the error matrix* $E_m = X - \tilde{X}_m$ *satisfies*

$$\|E_{2\ell}\|_2 \le \left\|\left(B^{-T}A\right)^{\ell}\right\|_2 \left\|\left(B^{-1}A^T\right)^{\ell}\right\|_2 \|X\|_2, \quad \text{for } \ell = 0, 1, 2, \ldots.$$

Proof. Part (a) follows immediately from Theorem 3.9; the condition $\rho(B^{-T}A) < 1$ implies that the spectrum of $A - \lambda B^T$ is T-reciprocal free.

By the definition (5.36) of $\tilde{X}_m$, we get $E_0 = X$ and $E_{m+1} = -B^{-T}E_m^T A^T$. Then a simple induction argument shows that

$$E_{2\ell} = (B^{-T}A)^{\ell} X (B^{-1}A^T)^{\ell} \quad \text{and} \quad E_{2\ell+1} = -B^{-T}\,(AB^{-T})^{\ell} X^T (A^T B^{-1})^{\ell}\, A^T, \tag{5.37}$$

for $\ell = 0, 1, 2, \ldots$. Observe that $\rho(B^{-T}A) = \rho(AB^{-T}) = \rho(B^{-1}A^T) = \rho(A^T B^{-1})$. Hence, $\rho(B^{-T}A) < 1$ implies that

$$\lim_{\ell\to\infty} (B^{-T}A)^{\ell} = \lim_{\ell\to\infty} (AB^{-T})^{\ell} = \lim_{\ell\to\infty} (B^{-1}A^T)^{\ell} = \lim_{\ell\to\infty} (A^T B^{-1})^{\ell} = 0,$$

see [51, Theorem 5.6.12], and $\lim_{m\to\infty} E_m = 0$. This proves (b). Part (c) follows from the first equation in (5.37). □

Standard properties of the spectral radius [51, Corollary 5.6.14] and Theorem 5.8(c) show that asymptotically

$$\|E_{2\ell}\|_2 \lesssim \left(\rho(B^{-T}A)\right)^{2\ell} \|X\|_2, \qquad \ell \to \infty\,. \tag{5.38}$$

Therefore fast convergence of the fixed point iteration (5.36) for the T-Stein equation is expected if $\rho(B^{-T}A) < 1$ and this spectral radius is not too close to one. By Theorem 5.7, this implies that $\mathbb{V}_m \otimes \mathbb{W}_m$ contains an approximation to X that rapidly becomes more accurate as m increases. Numerically, we have observed this behavior as well for the approximation X_m picked by the Petrov-Galerkin condition; the experiments in Section 5.5 demonstrate that the projection method with the block Krylov subspaces $\mathbb{V}_m, \mathbb{W}_m$ in (5.10)-(5.11) converges always quickly when $\rho(B^{-T}A) < 1$, although, sometimes, it converges also in other situations.

5.5 Numerical tests

In this section we report on numerical experiments with the new Algorithms 5.1 and 5.2. All reported experiments were performed using Matlab R2012a on a PC with

processor Intel (R) Core (TM) 2 Quad CPU Q9400 @2.66GHz (4CPUs), with 4096 MB of RAM, and with operating system Windows 7 Enterprise 64 bits. Both CPU time (in seconds) and the dimension of the approximation space $\mathbb{V}_m$ (equal to the dimension of $\mathbb{W}_m$) are used to measure the cost of the different methods. We also report on the number of iterations for completeness.

Our algorithms are compared with the extended block Krylov subspace method applied to the standard Sylvester equation (3.20) (see Section 3.3.3, or [90] and the references therein for details). Some remarks are in order concerning this comparison. In view of the eventual goal, we use the relative residual defined in (5.19) for the T-Sylvester equation (instead of the relative residual of the Sylvester equation) for monitoring the convergence of the approximate solution $\widehat{X}_m \approx \widehat{\mathbf{V}}_m \widehat{Y}_m \widehat{\mathbf{W}}_m^T$ of (3.20) provided by the extended block Krylov subspace method. To compute this residual norm, we note that

$$\widehat{R}_m = A\widehat{X}_m + \widehat{X}_m^T B - C_1 C_2^T = \begin{bmatrix} A\widehat{\mathbf{V}}_m & \widehat{\mathbf{W}}_m & C_1 \end{bmatrix} \begin{bmatrix} \widehat{Y}_m & & \\ & \widehat{Y}_m^T & \\ & & I_r \end{bmatrix} \begin{bmatrix} \widehat{\mathbf{W}}_m^T \\ \widehat{\mathbf{V}}_m^T B \\ -C_2^T \end{bmatrix}.$$

Therefore, if $[A\widehat{\mathbf{V}}_m,\ \widehat{\mathbf{W}}_m,\ C_1] = Q_m S_m$ and $[\widehat{\mathbf{W}}_m,\ B^T\widehat{\mathbf{V}}_m,\ -C_2] = U_m G_m$ are two "skinny" QR factorizations then $\|\widehat{R}_m\|_F = \|S_m \operatorname{diag}(\widehat{Y}_m, \widehat{Y}_m^T, I_r)\, G_m^T\|_F$. Although this computation is cheap, it is far more expensive than the methods for computing residual norms presented in Propositions 5.4 and 5.6 for Algorithms 5.1 and 5.2, respectively. To take this into account, we will report CPU times with and *without* the computation of the relative residual in every iteration.

One motivation for reporting both times is that it usually suffices to verify the stopping criterion (5.19) only every few (say 5 or 10) iterations and hence the actual time will be in between.

The following table summarizes the algorithms that will be compared.

BK	Algorithm 5.1
BK-TR	Algorithm 5.1 applied to the transposed equation (5.13)
EK	Algorithm 5.2
EK-woR	EK without time for residual computation
EK-SYLV	extended block Krylov subspace method applied to Sylvester equation (3.20)
EK-SYLV-woR	EK-SYLV without time for residual computation

Before presenting the numerical tests in detail, let us summarize the main conclusions we have obtained, since they offer a clear guidance on the selection of algorithm for solving a given problem. EK is the most reliable method as it succeeds to converge more often than any other method; indeed, we have not found any example where EK does not converge but one of the other methods does. However, for problems satisfying $\rho(B^{-T}A) < 1$ we recommend the use of BK, which we have observed to be much faster than the other methods in this situation. On the other

hand, if $\rho(A^{-1}B^T) < 1$ then BK-TR is the fastest method. This is in accordance with the discussions in Section 5.4 and in the paragraphs after Remark 5.1. The method EK-SYLV based on the standard Sylvester equation (3.20) is not recommended since it fails to converge in situations where other methods succeed and even when it works, it is almost always slower than EK.

In particular, we only show results for right-hand sides $C_1C_2^T$ with vectors $C_1, C_2 \in \mathbb{R}^{n\times 1}$ generated by the command `rand` in Matlab, although we have also performed tests with random matrices $C_1, C_2 \in \mathbb{R}^{n\times r}$ with $r = 2.\ldots, 7$.

In this thesis, we present three types of numerical tests. In the first type of tests (5.9–5.11) the coefficient matrices A and B are finite difference discretizations of certain differential operators. In the second type of tests (5.12–5.17), we construct structured matrices A and B with prescribed eigenvalues for $B^{-T}A$. Finally, we discuss tests with block diagonal matrices built from the previous tests; see (5.39) and the numerical test 5.18.

Numerical test 5.9. A and B are $10^4 \times 10^4$ matrices obtained by finite-difference discretizations in $[0,1] \times [0,1]$ of the differential operators

$$a(u) = -u_{xx} - u_{yy} + y(1-x)u_x + \gamma u\,,$$
$$b(u) = -u_{xx} - u_{yy}\,,$$

respectively, where $\gamma = 10^4$. The vectors C_1, C_2 have been multiplied by 10^4 to match the magnitude of the entries of the matrices A and B. The following table displays the obtained results.

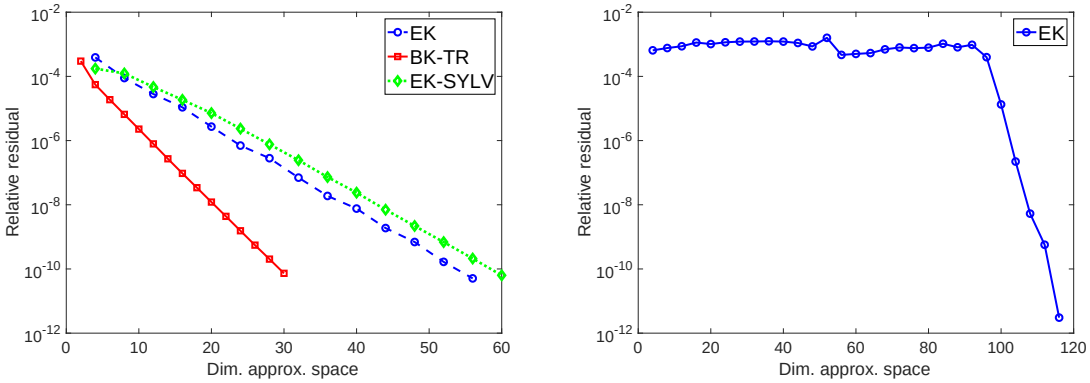
`tol` $= 10^{-10}$	EK	BK	BK-TR	EK-SYLV	EK-SYLV-woR	EK-woR
iterations	14	70	15	15	15	14
dim. approx. space	56	140	30	60	60	56
time (seconds)	4.277	50.856	1.684	6.255	2.901	3.666

The convergence history is shown in Figure 5.1(a). In this test, all eigenvalues of $B^{-T}A$ are well outside the unit circle; the smallest absolute value of the eigenvalues is approximately equal to 1.1226. All methods converge, with BK-TR being the fastest by far and BK the slowest. In this test, the use of extended Krylov spaces in EK does not add any essential information compared to BK-TR and therefore EK wastes half of the space.

Numerical test 5.10. A and B are $10^4 \times 10^4$ matrices obtained by finite-difference discretizations in $[0,1] \times [0,1]$ of the differential operators

$$a(u) = (-\exp(-xy)\,u_x)_x + (-\exp(xy)\,u_y)_y + 100\,x\,u_x + \gamma\,u$$
$$b(u) = -u_{xx} - u_{yy},$$

where $\gamma = 5 \cdot 10^4$. The vectors C_1, C_2 have been multiplied by 10^4 to match the magnitude of the entries of the matrices A and B.



(a) Numerical test 5.9. (b) Numerical test 5.11.

Figure 5.1: Convergence histories for numerical tests 5.9 and 5.11.

tol $= 10^{-10}$	EK	BK	BK-TR	EK-SYLV	EK-SYLV-woR	EK-woR
iterations	8	83	8	8	8	8
dim. approx. space	32	166	16	32	32	32
time (seconds)	1.716	58.175	0.764	2.496	1.784	1.6160

The results are similar to the numerical test 5.9, only that the convergence (except for BK) is faster due to the fact that the eigenvalues of $B^{-T}A$ are even further outside the unit circle. The eigenvalue of smallest absolute value is approximately equal to 1.6159.

Numerical test 5.11. A and B are $10^4 \times 10^4$ matrices obtained via finite-difference discretizations in $[0,1] \times [0,1]$ of the differential operators

$$a(u) = (-\exp(-xy)\, u_x)_x + (-\exp(xy)\, u_y)_y + 100\, x\, u_x + \gamma u\,,$$
$$b(u) = -u_{xx} - u_{yy} + 100\, x\, u_x\,,$$

where $\gamma = 5 \cdot 10^4$. The vectors C_1, C_2 have been multiplied by 10^4 to match the magnitude of the entries of the matrices A and B. It turns out that only EK converges for this example. BK, BK-TR, and EK-SYLV do not converge within 100 iterations and their relative residuals remained essentially constant around 10^{-3}. This lack of convergence is marked with a star in the following table.

tol $= 10^{-10}$	EK	BK*	BK-TR*	EK-SYLV*	EK-SYLV-woR*	EK-woR
iterations	29	100	100	100	100	29
dim. approx. space	116	200	200	400	400	116
time (seconds)	10.920	70.715	63.835	521.214	71.807	9.158

The convergence history of EK is plotted in Figure 5.1(b), which shows stagnation until the dimension of the approximation subspace is 90, after which quick convergence sets in. A key difference to the previous tests 5.9 and 5.10 is that the eigenvalues of $B^{-T}A$ are now located inside *and* outside the unit circle, with the magnitudes of the eigenvalues varying between 0.8679 and 1.4563. This fact has drastic effects on the convergence of the methods making EK the only valid method.

Tests 5.9, 5.10, and 5.11 clearly show that the eigenvalue distribution of $B^{-T}A$ plays an important role in the behavior of the algorithms. To investigate this further, we prescribe the eigenvalues of $B^{-T}A$ in the numerical tests 5.12-5.17. We construct these matrices as follows: let A_1 be a block diagonal real matrix with 1×1 or 2×2 blocks, whose eigenvalues are, respectively, the real and complex eigenvalues that are prescribed for $B^{-T}A$. We consider in addition the following tridiagonal matrices P and Q

$$P = \begin{bmatrix} 1 & \frac{1}{3} & & \\ \frac{1}{2} & 1 & \ddots & \\ & \ddots & \ddots & \frac{1}{3} \\ & & \frac{1}{2} & 1 \end{bmatrix}, \qquad Q = \begin{bmatrix} 1 & \frac{1}{6} & & \\ \frac{1}{4} & 1 & \ddots & \\ & \ddots & \ddots & \frac{1}{6} \\ & & \frac{1}{4} & 1 \end{bmatrix}.$$

Next, we define the matrices $A = PA_1Q$ and $B = Q^TP^T$. The eigenvalues of A_1 and $B^{-T}A$ are equal because

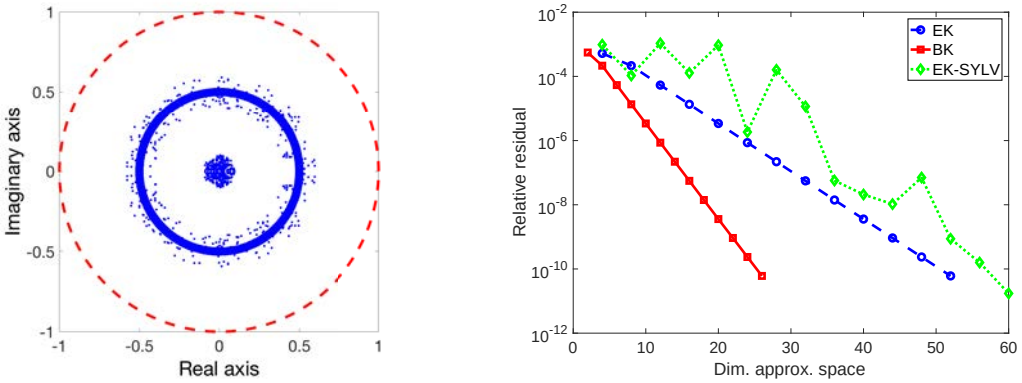
$$B^{-T}A = (Q^TP^T)^{-T}PA_1Q = (PQ)^{-1}PA_1Q = Q^{-1}P^{-1}PA_1Q = Q^{-1}A_1Q.$$

Note that B is pentadiagonal and that A has at most 7 nonzero diagonals.

Numerical test 5.12. A and B are $10^5 \times 10^5$ matrices such that the eigenvalues of $B^{-T}A$ are distributed as in Figure 5.2(a). Observe that all eigenvalues of $B^{-T}A$ are well inside the unit circle. BK-TR did not converge in 70 iterations; it was not even close to.

$\texttt{tol} = 10^{-10}$	EK	BK	BK-TR*	EK-SYLV	EK-SYLV-woR	EK-woR
iterations	13	13	70	15	15	13
dim. approx. space	52	26	140	60	60	52
time (seconds)	14.227	6.474	179.416	107.796	15.568	11.776

The convergence history is shown in Figure 5.2(b). BK is the fastest method by far, which is in full agreement with the discussion in Section 5.4. The behaviors of BK and BK-TR are opposite to what has been observed in the tests 5.9 and 5.10, while EK and EK-SYLV behave the same.



(a) Eigenvalues of $B^{-T}A$ and unit circle (dashed line).

(b) Convergence history.

Figure 5.2: Distribution of eigenvalues of $B^{-T}A$ and convergence history for numerical test 5.12.

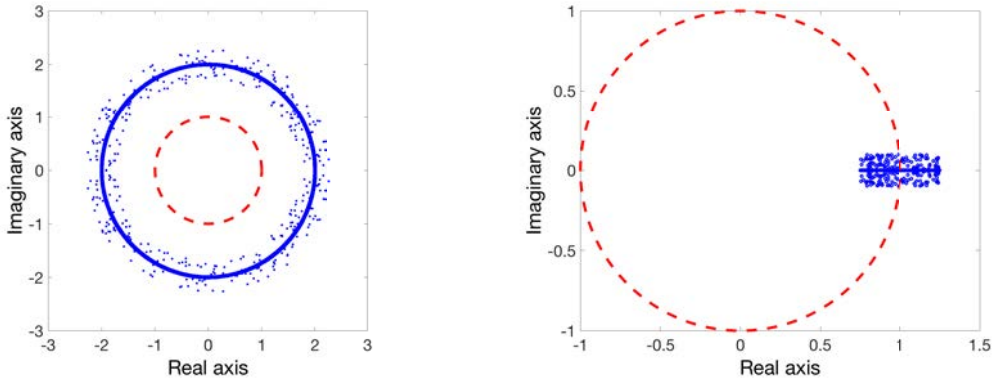
Numerical test 5.13. A and B are $10^5 \times 10^5$ matrices such that the eigenvalues of $B^{-T}A$ are distributed as in Figure 5.3(a). Observe that all eigenvalues of $B^{-T}A$ are well outside the unit circle. BK did not converge in 70 iterations; it was not even close to.

$\texttt{tol} = 10^{-10}$	EK	BK*	BK-TR	EK-SYLV	EK-SYLV-woR	EK-woR
iterations	13	70	13	13	13	13
dim. approx. space	52	140	26	52	52	52
time (seconds)	14.024	168.041	6.489	73.492	12.448	11.965

All results are very similar to the ones in test 5.9, except that the CPU times increase as a consequence of the larger size. However, we emphasize that the distributions of eigenvalues of $B^{-T}A$ are very different in both cases, in particular, for test 5.9 the eigenvalues are either real or distributed along an arc, even though the eigenvalues are well outside the unit circle in both cases.

Numerical test 5.14. A and B are $10^5 \times 10^5$ matrices such that the eigenvalues of $B^{-T}A$ are distributed as in Figure 5.3(b). Observe that $B^{-T}A$ has eigenvalue inside and outside the unit circle and near 1. The behaviors of the methods are as in test 5.11, with EK being the only converging method. All other methods were not close to convergence in 70 iterations.

$\texttt{tol} = 10^{-10}$	EK	BK*	BK-TR*	EK-SYLV*	EK-SYLV-woR*	EK-woR
iterations	21	70	70	70	70	21
dim. approx. space	84	140	140	280	280	84
time (seconds)	35.427	178.722	192.536	3041.208	296.339	32.569



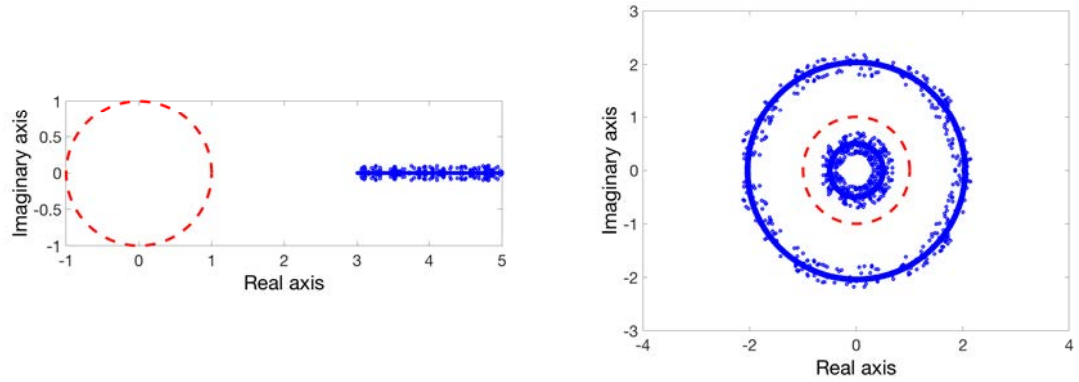
(a) Numerical test 5.13 (unit circle-dashed line). (b) Numerical test 5.14 (unit circle-dashed line).

Figure 5.3: Distribution of eigenvalues of $B^{-T}A$ for numerical tests 5.13 and 5.14.

Numerical test 5.15. A and B are $10^5 \times 10^5$ matrices such that the eigenvalues of $B^{-T}A$ are distributed as in Figure 5.4(a). In this test, the eigenvalues of $B^{-T}A$ are outside the unit circle and close to the real interval $[3, 5]$. This test illustrates a situation that we have not seen before: both BK-TR and BK exhibit fast convergence, which is surprising in the light of the discussion in Section 6 and the results of the previous tests. As expected, BK-TR is the fastest method, but the difference is not as large as in the other tests.

$\texttt{tol} = 10^{-10}$	EK	BK	BK-TR	EK-SYLV	EK-SYLV-woR	EK-woR
iterations	3	9	4	3	3	3
dim. approx. space	12	18	8	12	12	12
time (seconds)	1.435	8.190	1.092	3.135	1.544	1.382

Numerical test 5.16. The tests presented so far may give the impression that EK converges well for any real T-Sylvester equation; however, this is not true. In fact, when $B^{-T}A$ has eigenvalues inside and outside the unit circle, often none of the methods presented in this paper can be expected to work well. This test is an example of such a situation. A and B are $10^5 \times 10^5$ matrices such that the eigenvalues of $B^{-T}A$ are distributed as in Figure 5.4(b). With $\texttt{tol} = 10^{-10}$ none of the methods converged in 200 iterations, at which they reached relative residuals in the order of $10^{-2} - 10^{-3}$. We have performed other tests with eigenvalues of $B^{-T}A$ distributed as in Figure 5.4(b) but with matrices of size 1000×1000. These small sized cases can be solved directly by using the Algorithm 3.3 and we have observed that the solution is very far from having low-rank. As a consequence, the relative residuals for all methods remain constant around 10^{-3} until the dimensions of the approximation spaces are equal to 1000, i.e., they are the whole space.



(a) Numerical test 5.15 (unit circle-dashed line). (b) Numerical test 5.16 (unit circle-dashed line).

Figure 5.4: Distribution of eigenvalues of $B^{-T}A$ for numerical tests 5.15 and 5.16.

Numerical test 5.17. In this example, the eigenvalues of $B^{-T}A$ are prescribed to lie all inside the unit circle, but some of them are close to 1, see Figure 5.5(a). This test illustrates a situation where EK-SYLV does not converge in 100 iterations, while EK and BK do converge. The failure of BK-TR is expected from the discussion in Section 5.4 and the failure of EK-SYLV from the discussion after Theorem 3.12. After 100 iterations BK-TR and EK-SYLV were far from satisfying the stopping criterion. All methods have difficulties with this example and the convergence history for EK and BK differs from the other examples, see Figure 5.5(b).

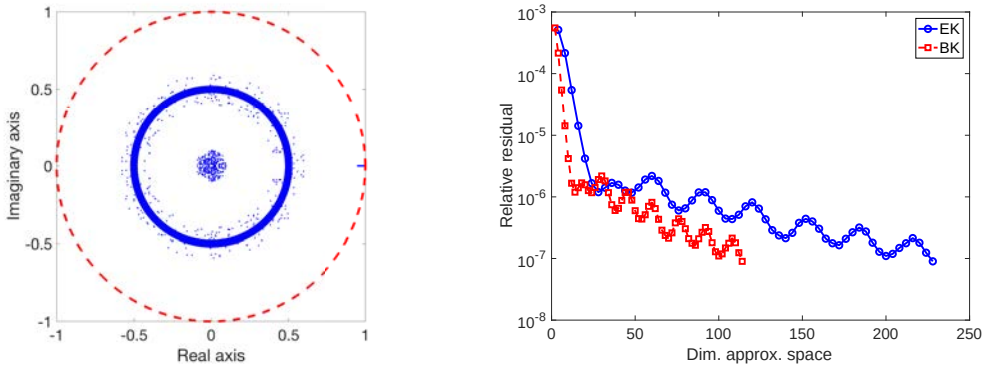
$\texttt{tol} = 10^{-7}$	EK	BK	BK-TR*	EK-SYLV*	EK-SYLV-woR*	EK-woR
iterations	57	57	100	100	100	57
dim. approx. space	228	114	200	400	400	228
time (seconds)	261.566	112.351	336.915	6785.715	567.110	217.339

Finally, we have considered matrices A and B from the previous tests such that BK or BK-TR performs well but the other methods do not, and then we construct the new pair of matrices

$$\tilde{A} = \mathrm{diag}(A, B) \qquad \text{and} \qquad \tilde{B} = \mathrm{diag}(B, A)\,. \tag{5.39}$$

When proceeding in this way for the matrices from tests 5.9, 5.10, and 5.12, only EK is observed to perform well. For brevity we report only the case coming from test 5.12.

Numerical test 5.18. We construct matrices A and B of size $(2 \cdot 10^5) \times (2 \cdot 10^5)$ applying (5.39) to the matrices from test 5.12. BK, BK-TR and EK-SYLV do not



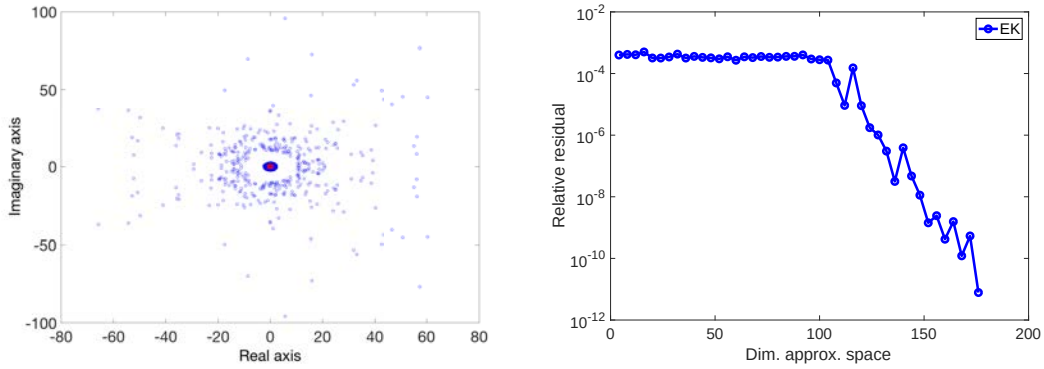
(a) Eigenvalues of $B^{-T}A$ and unit circle (dashed line).

(b) Convergence history.

Figure 5.5: Distribution of eigenvalues of $B^{-T}A$ and convergence history for numerical test 5.17.

converge in 70 iterations and they are by no means close to satisfying the stopping criterion. In contrast, EK converges in 44 iterations, see also Figure 5.6.

$\texttt{tol} = 10^{-10}$	EK	BK*	BK-TR*	EK-SYLV*	EK-SYLV-woR*	EK-woR
iteration	44	70	70	70	70	44
dim. approx. space	176	140	140	280	280	176
time (seconds)	305.621	352.733	372.358	6750.553	675.375	289.663



(a) Eigenvalues of $B^{-T}A$.

(b) Convergence history.

Figure 5.6: Distribution of eigenvalues of $B^{-T}A$ and convergence history for numerical test 5.18.

We can see that our numerical experiments show that the new extended Krylov method we have introduced in Algorithm 5.2 works very well in many situations and that, if it fails, then all the other methods fail too. However, if we have some previous information about the behavior of the eigenvalues of $B^{-T}A$ and they are either well inside the unit circle, or well outside the unit circle, then the new block Krylov method introduced in Algorithm 5.1 is much faster. More conclusions and open problems will be discussed in Chapter 6.

Chapter 6

Conclusions, publications and open problems

In this chapter, we discuss the main conclusions and original results of this PhD Thesis. We also present a list of all papers that include the results developed in this dissertation, the conferences where they have been presented, and we discuss some future work motivated by the results obtained in this thesis.

6.1 Conclusions and original contributions

In this section, we provide a summary of the main original results introduced in this PhD Thesis.

Chapter 3: As far as we know, the connection we have established between T-Sylvester equations and Sylvester equations in Theorem 3.12 is a new result in the literature. A priori, these relations suggest that a large-scale T-Sylvester equation can be solved numerically by using a projection method on the related Sylvester equation stated in Theorem 3.12(b). However, the numerical experiments show that this is not a good approach, and projection methods developed specifically for T-Sylvester equations perform much better than the ones for the related Sylvester equation.

Chapter 4: We have introduced the R-CORK method for solving large-scale rational eigenvalue problems $R(\lambda)x = 0$, with $R(\lambda)$ represented as in (2.9), this is, $R(\lambda) = P(\lambda) - E(C - \lambda D)^{-1}F^T$, where $P(\lambda)$ is the polynomial part and $E(C - \lambda D)^{-1}F^T$ is the strictly proper part of the rational matrix $R(\lambda)$. Note that, the first key idea is that R-CORK solves the generalized (linear) eigenvalue problem $(\mathcal{A} - \lambda\mathcal{B})\mathbf{z} = 0$, defined in (2.10), for $\mathcal{A}, \mathcal{B}$ and $\mathbf{z}$ as in (2.11)-(2.12).

The second key idea is that R-CORK is a structured version of the classical rational Krylov method for solving generalized eigenvalue problems that takes

advantage of the particular structure of the linearization (2.10). This structure allows us to represent the orthonormal bases of the rational Krylov subspaces of the linearization in a compact form involving less parameters than the bases of rational Krylov subspaces of the same dimension corresponding to unstructured generalized eigenvalue problems of the same size as the considered linearization. In addition, this compact form can be efficiently and stably updated in each rational Krylov iteration by the use of two levels of orthogonalization in the spirit of the TOAR [100, 71, 63] and the CORK [104] methods for large-scale polynomial eigenvalue problems.

The combined use of the compact representation of rational Krylov subspaces and the two levels of orthogonalization in R-CORK reduces significantly the orthogonalization and the memory costs with respect to a direct application of the classical rational Krylov method to the linearization $\mathcal{A}-\lambda\mathcal{B}$. If we consider as $n \times n$ the size of $R(\lambda)$, j the maximum dimension of the Krylov subspaces of $\mathcal{A} - \lambda\mathcal{B}$, d the degree of $P(\lambda)$, and $s \times s$ the size of the pencil $(C - \lambda D)$, then the reduction in costs of R-CORK is appreciable in the case when $jd \ll n$, which always happens in large scale problems, and very considerable if also $s \ll n$ and $d < j$, which occurs in most applications of rational eigenvalue problems. In this case, after j iterations, the orthogonalization cost of R-CORK is $\mathcal{O}(j^2 n)$, while the cost of the classical rational Krylov is $\mathcal{O}(j^2 nd)$, and the memory cost of R-CORK is approximately nj numbers, while the one of classical rational Krylov is ndj. These reductions can be combined with an structured implementation of a Krylov-Schur implicit restarting adapted to the compact representation used by R-CORK, which allows us to keep the dimension of the Krylov subspaces moderate without essentially increasing the number of iterations until convergence. Our numerical experiments confirm all these good properties of the R-CORK method.

Chapter 5: Two new projection algorithms based on Krylov subspaces to solve real, large-scale and sparse T-Sylvester equations, $AX + X^T B = C$, with low-rank right-hand side were developed and tested extensively in many different situations. In our numerical experiments, we have compared the new methods with the extended Krylov method applied to the equivalent standard Sylvester equation: $(B^{-T}A)X - X(A^{-T}B) = B^{-T}C - B^{-T}C^T A^{-T}B$.

Our experiments show that the first method, based on block Krylov subspaces, works efficiently if the eigenvalues of $B^{-T}A$ are inside the unit circle, in the sense of CPU time and reliability. Note that similar comments hold if the eigenvalues of $B^{-T}A$ are outside the unit circle and the block Krylov method is applied to the transposed of the given T-Sylvester equation, this is, if it is applied to $B^T X + X^T A^T = C^T$.

The second procedure, based on extended block Krylov subspaces, works very well in many situations and we can see, based on our numerical experience, that

if it fails, then all the other methods fail.

The implementation for the new procedures that we have introduced for the solution of the T-Sylvester equation projects the large-scale problem into a small-scale T-Sylvester equation. We showed that the construction of the matrices for the reduced equation can be performed without extra costs, resulting in fast and efficient algorithms.

Note that, the extended Krylov method applied to the equivalent standard Sylvester equation fails more often than the new methods, and also that, when it converges, it is always considerably smaller than the new algorithms. For these reasons, we do not recommend to use it.

The relation between T-Sylvester equations and T-Stein equations give us some intuition on the expected convergence of the projection method, in particular when $\rho(B^{-T}A) < 1$, where the block Krylov method converges always quickly.

With the results in Chapter 5, we solve a relevant open problem concerning the T-Sylvester equation, namely, its numerical solution in the large-scale setting.

6.2 Publications

The results in Chapter 4 are contained in:

Dopico, F. M., González-Pizarro, J., *A compact rational Krylov method for large-scale rational eigenvalue problems*, accepted in Numer. Linear Algebra Appl., arXiv:1705.06982, 2018.

Theorem 3.12 in Section 3.2 and the results in Chapter 5 are contained in:

Dopico, F. M., González-Pizarro, J., Kressner, D., Simoncini, V., *Projection methods for large-scale T-Sylvester equations*, Math. Comp. 85:2427-2455, DOI: https://doi.org/10.1090/mcom/3081, 2016.

6.3 Contributions to conferences

The results developed in this PhD Thesis were presented by its author in several conferences. Among these conferences, we find some of the most relevant international conferences in the area of Linear Algebra: The International Linear Algebra Society (ILAS) conference and the Society for Industrial and Applied Mathematics (SIAM) conference. The Krylov methods for solving T-Sylvester equations developed in Chapter 5 were presented in the following conferences:

- *Projection methods for large T-Sylvester equations.* Presented as a contributed talk in the joint meeting organized by the spanish thematic network of Linear Algebra, Matrix Analysis and Applications (ALAMA) and the german activity group of the International Association of Applied Mathematics and Mechanics (GAMM) on Applied and Numerical Linear Algebra (ANLA), (ALAMA-GAMM/ANLA'2014 meeting), Polytechnic University of Catalonia, Barcelona, Spain, July 14-16, 2014.

- *Projection methods for large T-Sylvester equations.* Also presented as a contributed talk, in the 19th ILAS Conference, Sungkyunkwan University, Seoul, South Korea, August 6-9, 2014.

- *Projection methods for large T-Sylvester equations.* Presented as a contributed talk in the Workshop on Matrix Equations and Tensor Techniques, University of Bologna, Bologna, Italy, September 21-22, 2015.

Whereas that the results related to the R-CORK method developed in Chapter 4 were presented in:

- *Two-Level orthogonal Arnoldi method for large rational eigenvalue problems.* Presented as a contributed presentation in the session of "Eigenvalue and SVD Problems" in the SIAM Conference on Applied Linear Algebra (LA15), Hyatt Regency Atlanta Hotel, Atlanta, United States of America, October 26-30, 2015.

- *Two-level orthogonal Arnoldi method for large rational eigenvalue problems.* Presented in the Young Researchers Sessions in the ALAMA'2016 Meeting, Real Colegiata San Isidoro Hotel, León, Spain, June 20-22, 2016.

- *A compact rational Krylov method for large-scale rational eigenvalue problems.* Presented in the minisymposium "Polynomial and Rational Eigenvalue Problems" in the 20th Conference of the International Linear Algebra Society (ILAS), Catholic University of Leuven, Leuven, Belgium, July 11-15, 2016.

6.4 Future work and open problems

Finally, we present some open problems and future research motivated by the results obtained in this dissertation.

Problem 1: Backward error analysis of the R-CORK method. Consider the Arnoldi method with the Arnoldi recurrence relation (2.18) written as

$$AV_m = V_{m+1}\underline{H}_m.$$

It is well known that if an implementation in floating point arithmetic of the Arnoldi method is performed, with special care in the orthogonalization process, it results in a backward stable algorithm [6, 91] in the sense that computed the matrices $\hat{V}_{m+1}$ and $\underline{\hat{H}}_m$ satisfy an exact Arnoldi recurrence relation for a slightly perturbed matrix, this is,

$$(A + \Delta A)\hat{V}_m = \hat{V}_{m+1}\underline{\hat{H}}_m,$$

with $\|\Delta A\|_2 = \mathcal{O}(\epsilon)\|A\|_2$ and ϵ the unit roundoff of the computer.

This means that a basis of an exact Krylov subspace corresponding to a nearby matrix to A is computed. A similar backward error analysis can be performed for the shift-and-invert Arnoldi algorithm [88]. This analysis shows that an implementation in floating point arithmetic of the Arnoldi method applied to A^{-1} yields computed matrices $\hat{V}_{m+1}$ and $\underline{\hat{H}}_m$ such that

$$(A + \Delta A)^{-1}\hat{V}_m = \hat{V}_{m+1}\underline{\hat{H}}_m,$$

again with $\|\Delta A\|_2 = \mathcal{O}(\epsilon)\|A\|_2$.

As a future work, we propose to develop a backward error analysis of the R-CORK method. First, we are interested into prove that the algorithm is backward stable for the linearized problem, and then, establish relations between this analysis and the "coefficients" of the rational matrix in the representation (2.9). However, the R-CORK method is a structured algorithm based on the rational Krylov method. For this reason, as a starting point, this analysis should be performed first for a compact Arnoldi method applied to the linearization (2.10) and be continued to the shift-and-invert compact Arnoldi algorithm with a fixed shift at the origin, to finalize with the implementation of the compact rational Krylov method with different shifts at each iteration, i.e., R-CORK. This work is already being addressed by the author and is a work in progress.

Problem 2: Extend R-CORK to other linearizations, in particular to linearizations that express the polynomial part in other bases. In this work, we have considered the R-CORK method for solving the REP $R(\lambda)x = 0$, with $R(\lambda)$ defined as in (2.9), i.e.,

$$R(\lambda) = P(\lambda) - E(C - \lambda D)^{-1}F^T,$$

where $P(\lambda)$ is expressed in the monomial basis. Since matrix polynomials can be expressed in several different bases, it is natural to consider the rational matrix with the polynomial part expressed as

$$P(\lambda) = \sum_{i=0}^{d} b_i(\lambda)P_i,$$

where $b_i(\lambda)$, $i = 0, \ldots, d$, is a polynomial of degree i. The authors of [36] developed linearizations for rational matrices with polynomial part expressed in orthogonal bases. Some of these linearizations have the structure of the structured pencils in Definition 2.13 augmented with the matrices E, C, D and F in (2.9), which allows easily to extend R-CORK for them. However, there are linearizations in [36] that do not have this structure. For that reason, it is of interest to analyze if R-CORK can be developed by considering linearizations that are not structured-like pencils. In [70] this problem is addressed for nonlinear matrices that are approximated by rational matrices, however, these rational matrices are not expressed as in (2.9) and the linearizations that they use are not strong linearizations, unlike the linearizations introduced in [36]. In this way, both methods could be compared and the advantages or disadvantages of them could be analyzed.

Problem 3: Develop a compact two-sided Krylov method for REPs expressed as the sum of a polynomial part and a strictly proper rational part. As we mentioned in Section 2.4.3, CORK is a method that exploits the structure of the linearization for constructing the Krylov vectors in a compact form. However, CORK approximates eigenvalues and their corresponding right eigenvectors but is not suitable in its current form for the computation of left eigenvectors. For this reason, two-sided compact rational Krylov methods are introduced in [69]. These methods are a generalization of the CORK method and they are based on a class of Kronecker structured pencils that includes many linearizations. This class of structured pencils facilitates the development of a general framework for the computation of both right and left-sided Krylov subspaces in compact form, which allows to compute both right and left eigenvectors. However, these linearizations do not consider the case when the rational matrix is expressed as in (2.9), this is, as the sum of a polynomial part and a strictly proper rational part. For this reason, an extension of the R-CORK method would be useful to develop a compact two-sided Krylov method for REPs expressed as in (2.9).

Problem 4: Develop low-rank Smith methods for solving large-scale and sparse T-Sylvester equations with low-rank right-hand side. Since the projection methods that we introduced in this dissertation are, as far as we know, the first methods for solving large-scale T-Sylvester equations with low-rank right-hand side, it is natural to think about the idea of developing other procedures to solve this matrix equation. As we saw in Section 5.4, the T-Sylvester equation $AX + X^TB = C_1C_2^T$ is equivalent to the T-Stein equation $X = B^{-T}C_2C_1^T - B^{-T}X^TA^T$ which might be solved by developing a low-rank Smith method [78]. Smith methods were developed many years ago for solving Sylvester equations [93] and, then, they were used for solving large-scale Lyapunov equations with low-rank right-hand side [78] by creating

a procedure that accelerates the convergence of the fixed point iteration to the solution and by approximating the solution by a low-rank matrix in order to have low storage requirements. They use an expression of the iterates, that could be extended for T-Sylvester equations, to compute iterative subsequences of the fixed point iteration which results in algorithms that converges fast. The development of this procedure could result in an efficient method for solving large-scale T-Sylvester equations and it would allow us to compare the results with the ones obtained by projection methods.

Problem 5: An error analysis of the projection methods for the T-Sylvester equation. An error analysis of projection methods for Sylvester equations has been discussed in [11]. As we mentioned in Section 3.3, projection methods for solving the Sylvester equation are based on Galerkin conditions, which is an important difference with our projection methods for the T-Sylvester equation, which are based on Petrov-Galerkin conditions. For instance, if we consider the Sylvester equation $AX + XB = C$ with $C := C_1 C_2^T$ of rank 1 and A and B having disjoint field of values, an estimate for the Galerkin residual can be obtained [11, Theorem 2.3]. As far as we know, it does not exist a similar result for projection methods for the T-Sylvester equation and these results are far from being trivial. The development of this error analysis will be the subject of future work.

Bibliography

[1] Abdelmalek, N.N., *Round off error analysis for Gram-Schmidt method and solution of linear least square problems*, BIT, 11:345-367, 1971.

[2] Alam, R., Behera, N., *Linearizations for rational matrix functions and Rosenbrock system polynomials*, SIAM J. Matrix Anal. Appl., 37(1):354-380, 2016.

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

[4] Amparan, A., Dopico, F.M., Marcaida, S., Zaballa, I., *Strong linearizations of rational matrices*, MIMS EPrint 2016.51, Manchester Institute for Mathematical Sciences, The University of Manchester, UK, 2016.

[5] Antoulas, A.C., *Approximation of Large-Scale Dynamical Systems*, Society for Industrial and Applied Mathematics, Philadelphia, 2005.

[6] Arioli, M., Fassino, C., *Roundoff error analysis of algorithms based on Krylov subspace methods*, BIT, 36(2):189-206, 1996.

[7] Arnoldi, W.E., *The principle of minimized iteration in the solution of the matrix eigenvalue problem*, Quart. Appl. Math., 9:17-29, 1951.

[8] Bai, Z., Su, Y., *SOAR: A second-order Arnoldi method for the solution of the quadratic eigenvalue problem*, SIAM J. Matrix Anal. Appl., 26:640-659, 2005.

[9] Bartels, R.H., Stewart, G.W., *Algorithm 432: Solution of the matrix equation $AX + XB = C$*, Comm. ACM, 15(9):820-826, 1972.

[10] Baumann, M., Helmke, U., *Singular value decomposition of time-varying matrices*, Future Generation Computer Systems, 19(3):353-361, 2003.

[11] Beckermann, B., *An error analysis for rational Galerkin projection applied to the Sylvester equation*, SIAM J. Numer. Anal., 49(6):2430-2450, 2011.

[12] Benner, P., Li, J.R., Penzl, T., *Numerical solution of large-scale Lyapunov equations, Riccati equations, and linear-quadratic optimal control problems*, Numer. Linear Algebra Appl., 15:755-777, 2008.

[13] Benner, P., Li, R.C., Truhar, N., *On the ADI method for Sylvester equations*, J. Comput. Appl. Math., 233(4):1035-1045, 2009.

[14] Betcke, T., Higham, N.J., Mehrmann, V., Schröder, C., Tisseur, F., *NLEVP: a collection of nonlinear eigenvalue problems*, ACM Trans. Math. Software, 39(2):Art.7, 28, 2013.

[15] Braden, H.W., *The equations* $A^T X \pm X^T A = B$, SIAM J. Matrix Anal. Appl., 20(2):295-302 (electronic), 1999.

[16] Byers, R., Kressner, D., *Structured condition numbers for invariant subspaces*, SIAM J. Matrix Anal. Appl., 28(2):326-347, 2006.

[17] Calvetti, D., Reichel, L., *Application of ADI iterative methods to the restoration of noisy images*, SIAM J. Matrix Anal. Appl., 28:326:347, 2006.

[18] Campos, C., Roman, J.E., *Parallel Krylov solvers for the polynomial eigenvalue problem in SLEPc*, SIAM J. Sci. Comput., 38(5):S385-S411, 2016.

[19] Chatelin, F., *Simultaneous Newton's iteration for the eigenproblem*, In *Defect correction methods (Oberwolfach, 1983)*, volume 5 of Comput. Suppl., pages 67-74. Springer, Vienna, 1984.

[20] Chiang, C.Y., *A note on the T-Stein matrix equation*, Abstr. Appl. Anal. Art. ID 824641, 8, 2013.

[21] Chiang, C.Y., Chu, E.K.W, Lin, W.W., *On the* $\star-$*Sylvester equation* $AX \pm X^\star B^\star = C$, Appl. Math. Comput., 218(17):8393-8407, 2012.

[22] Chu, D., Lin, W.W., Tan, R.C.E., *A numerical method for a generalized algebraic Riccati equation*, SIAM J. Control Optim., 45:1222-1250, 2006.

[23] Clenshaw, C.W., *A note on the summation of Chebyshev series*, Math. Comp., 9(51):118-120, 1955.

[24] Daniel, J.W., Gragg, W.B., Kaufman, L., Stewart, G.W., *Reorthogonalization and stable algorithms for updating the Gram-Schmidt factorization*, Math. Comput., 30:772-795, 1976.

[25] Demmel, J.W., *Three methods for refining estimates of invariant subspaces*, Computing, 38:43-57, 1987.

[26] De Souza, E., Bhattacharyya, S.P., *Controllability and observability and the solution of* $AX - XB = C$, Linear Alg. Appl., 39:167-188, 1981.

[27] De Terán, F., *The solution of the equation* $AX + BX^\star = 0$, Linear Multilinear Algebra, 61:1605-1628, 2013.

[28] De Terán, F., Dopico, F.M., *Consistency and efficient solution of the Sylvester equation for* $\star-$*congruence*, Electron. J. Linear Algebra, 22:849-863, 2011.

[29] De Terán, F., Dopico, F.M., *The solution of the equation* $XA + AX^T = 0$ *and its application to the theory of orbits*, Linear Algebra Appl., 434(1):44-67, 2011.

[30] De Terán, F., Dopico, F.M., Guillery, N., Montealegre, D., Reyes, N., *The solution of the equation* $AX + X^\star B = 0$, Linear Algebra Appl., 438(7):2817-2860, 2013.

[31] De Terán, F., Dopico, F.M., Mackey, D.S., *Fiedler companion linearizations and the recovery of minimal indices*, SIAM J. Matrix Anal. Appl., 31:2181-2204, 2010.

[32] De Samblanx, G., Meerbergen, K., Bultheel, A., *The implicit application of a rational filter in the RKS method*, BIT, 37(4):925-947, 1997.

[33] Dmytryshyn, A., Kågström, B., Sergeichuk, V.V., *Symmetric matrix pencils: codimension counts and the solution of a pair of matrix equations*, Electron. J. Linear Algebra, 27:1-18, 2014.

[34] Dopico, F.M., González-Pizarro, J., *A compact rational Krylov method for large-scale rational eigenvalue problems*, submitted, 2017.

[35] Dopico, F.M., González-Pizarro, J., Kressner, D., Simoncini, V., *Projection methods for large-scale T-Sylvester equations*, Math. Comp., 85:2427-2455, 2016.

[36] Dopico, F.M., Marcaida, S., Quintana, M.C., *Strong linearizations of rational matrices with polynomial part expressed in an orthogonal basis*, submitted, 2018.

[37] Enright, W.H., *Improving the efficiency of matrix operations in the numerical solution of stiff ordinary differential equations*, ACM Trans. Math. Softw., 4:127-136, 1978.

[38] Epton, M., *Methods for the solution of $ADXD-BXC = E$ and its application in the numerical solution of implicit ordinary differential equations*, BIT, 20:341-345, 1980.

[39] Garcia, S.R., Shoemaker, A.L., *On the matrix equation $XA+AX^T = 0$*, Linear Algebra Appl., 438(6):2740-2746, 2013.

[40] Giraud, L., Langou, J., Rozloznik, M., Van Den Eshof, J., *Rounding error analysis of the classical Gram-Schmidt orthogonalization process*, Numer. Math., 101:87-100, 2005.

[41] Gohberg, I., Lancaster, P., Rodman, L., *Matrix Polynomials*, Academic Press, New York, 1982.

[42] Golub, G.H., Nash, S., Van Loan, C.F., *A Hessenberg-Schur method for the problem $AX + XB = C$*, IEEE Trans. Automat. Control, 24:909-913, 1979.

[43] Grasedyck, L., *Existence of a low rank or $\mathcal{H}$-matrix approximant to the solution of a Sylvester equation*, Numer. Linear Algebra Appl., 11:371-389, 2004.

[44] Golub, G.H., Van Loan, C.F., *Matrix Computations, 4th ed.*, Johns Hopkins University Press, Baltimore, MD, 2013.

[45] Güttel, S., Van Beeumen, R., Meerbergen, K., Michiels, W., *NLEIGS: A class of robust fully rational Krylov methods for nonlinear eigenvalue problems*, SIAM J. Sci. Comput., 36(6):A2842-A2864, 2013.

[46] Güttel, S., Tisseur, F., *The nonlinear eigenvalue problem*, Acta Numerica, 26:1-94, 2017.

[47] Hayami, K., Morikuni, K., *Inner-iteration Krylov subspace methods for least squares problems*, SIAM J. Matrix Anal. Appl., 34:1-22, 2013.

[48] Higham, N. J., *Accuracy and Stability of Numerical Algorithms, 2nd ed.*, Society for Industrial and Applied Mathematics, Philadelphia, 2002.

[49] Hwang, T., Lin, W., Liu, J., Wang, W., *Numerical simulation of a three dimensional quantum dot*, J. Comput. Phys., 196:208-232, 2004.

[50] Hodel, A.S., Misra, P., *Solution of under-determined Sylvester equations in sensor array signal processing*, Linear Alg. Appl., 249:1-14, 1996.

[51] Horn, R., Johnson, C., *Matrix Analysis, 2nd ed.*, Cambridge University Press, New York, 2012.

[52] Horn, R., Johnson, C., *Topics in Matrix Analysis*, Cambridge University Press, New York, 1991.

[53] Hu, D.Y., Reichel, L., *Krylov subspace methods for the Sylvester equation*, Linear Alg. Appl., 172:283-313, 1992.

[54] Ikramov, Kh. D., *On conditions for the unique solvability of the matrix equation $AX+X^TB=C$*, (Russian), Dokl.Akad. Nauk, 430:444-447, 2010; English transl., Dokl. Math. 81:63-65, 2010.

[55] Jaimoukha, I.M., Kasenally, E.M., *Krylov subspace methods for solving large Lyapunov equations*, SIAM J. Numer. Anal., 31:227-251, 1994.

[56] Jarlebring, E., Poloni, F., *Iterative methods for the delay Lyapunov equation with T-Sylvester preconditioning*, submitted, 2015.

[57] Jennings, A., Stewart, W.J., *A simultaneous iteration algorithm for real matrices*, ACM Trans. of Math. Software, 7:184-198, 1981.

[58] Jonsson, I., Kågström, B., *Recursive blocked algorithms for solving triangular systems, Part I: One-sided and coupled Sylvester-type matrix equations*, ACM Trans. Math. Software, 28:392-415, 2002.

[59] Jonsson, I., Kågström, B., *Recursive blocked algorithms for solving triangular systems, Part II: Two-sided and generalized Sylvester and Lyapunov matrix equations*, ACM Trans. Math. Software, 28:416.435, 2002.

[60] Kailath, T., *Linear Systems*, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1980.

[61] Kawamoto, A., Katayama, T., *The semi-stabilizing solution of generalized algebraic Riccati equation for descriptor systems*, Automatica J. IFAC, 38:1651-1662, 2002.

[62] Kawamoto, A., Takaba, K., Katayama, T., *On the generalized algebraic Riccati equation for continuous-time descriptor systems*, Linear Algebra Appl., 296:1-14, 1999.

[63] Kressner, D., Roman, J.E., *Memory-efficient Arnoldi algorithms for linearizations of matrix polynomials in Chebyshev basis*, Numer. Linear Algebra Appl., 21(4):569-588, 2014.

[64] Kressner, D., Schröder, C., Watkins, D.S., *Implicit QR algorithms for palindromic and even eigenvalue problems*, Numer. Algorithms, 51(2):209-238, 2009.

[65] Laub, A.J., *Matrix Analysis for Scientists & Engineers*, Society for Industrial and Applied Mathematics, Philadelphia, 2004.

[66] Lehoucq, R.B., Sorensen, D.C., *Deflation techniques within an implicitly restarted Arnoldi iteration*, SIAM J. Matrix Anal. Appl., 17:789-821, 1996.

[67] Lehoucq, R.B., Sorensen, D.C., Yang, C., *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*, Society for Industrial and Applied Mathematics, Philadelphia, 2004.

[68] Liang, Q., Ye, Q., *Computing singular values of large matrices with an inverse-free preconditioned Krylov subspace method*, Electron. Trans. Numer. Anal., 42:197-221, 2014.

[69] Lietaert, P., Meerbergen, K., Tisseur, F., *Compact two-sided Krylov methods for nonlinear eigenvalue problems*, Report TW681, 2017.

[70] Lietaert, P., Pérez, J., Vandereycken, B., Meerbergen, K., *Automatic rational approximation and linearization of nonlinear eigenvalue problems*, submitted, 2018.

[71] Lu, D., Su, Y., Bai, Z., *Stability analysis of the two-level orthogonal Arnoldi procedure*, SIAM J. Matrix Anal. Appl., 37(1):192-214, 2016.

[72] Mackey, D.S., Mackey, N., Mehl, C. Mehrmann, V., *Structured polynomial eigenvalue problems: Good vibrations from good linearizations*, SIAM J. Matrix Anal. Appl., 28:1029-1051, 2006.

[73] Mackey, D.S., Mackey, N., Mehl, C. Mehrmann, V., *Vector spaces of linearizations for matrix polynomials*, SIAM J. Matrix Anal. Appl., 28:971-1004, 2006.

[74] Mazurenko, L., Voss, H., *Low rank rational perturbations of linear symmetric eigenproblems*, Z. Angew. Math. Mech., 86:606-616, 2006.

[75] Meerbergen, K., *The quadratic Arnoldi method for the solution of the quadratic eigenvalue problem*, SIAM J. Matrix Anal. Appl., 30(4):1463-1482, 2008.

[76] Mehrmann, V., Voss, H., *Nonlinear eigenvalue problems: A challenge for modern eigenvalue methods*, GAMM-Reports, 27:121-152, 2004.

[77] Mohammmadi, S.A., Voss, H., *Variational characterization of real eigenvalues in linear viscoelastic oscillators*, Technical report, submitted, 2016.

[78] Penzl, T., *A cyclic low-rank Smith method for large sparse Lyapunov equations*, SIAM J. Sci. Comput., 21(4):1401-1418, 1999.

[79] Penzl, T., *Eigenvalue decay bounds for solutions of Lyapunov equations: the symmetric case*, Systems Control Lett., 40:139-144, 2000.

[80] Rosenbrock, H.H., *State-space and Multivariable Theory*, Thomas Nelson & Sons, London, 1970.

[81] Roth, W.E., *The equations $AX - YB = C$ and $AX - XB = C$ in matrices*, Proc. Amer. Math. Soc., 3:392-392, 1952.

[82] Ruhe, A., *Rational Krylov sequence methods for eigenvalue computation*, Linear Algebra Appl., 58:391-405, 1984.

[83] Ruhe, A., *A practical algorithm for large sparse nonsymmetric matrix pencils*, SIAM J. Sci. Comput., 19:1535-1551, 1998.

[84] Saad, Y., *Iterative Methods for Sparse Linear Systems, 2nd ed.*, Society for Industrial and Applied Mathematics, Philadelphia, 2003.

[85] Saad, Y., *Krylov subspace methods for solving large unsymmetric linear systems*, Math. Comp., 37:105-126, 1981.

[86] Saad, Y., *Numerical Methods for Large Eigenvalue Problems, Rev. ed.* , Society for Industrial and Applied Mathematics, Philadelphia, 2011.

[87] Saad, Y., *Numerical solution of large Lyapunov equations*, in *Signal processing, scattering and operator theory, and numerical methods (Amsterdam, 1989)*, volume 5 of *Progr. Systems Control Theory*, pages 503-511, Birkhäuser Boston, Boston, MA, 1990.

[88] Schröder, C., Taslaman, L., *Backward error analysis of the shift-and-invert Arnoldi algorithm*, Numer. Math., 133:819-843, 2015.

[89] Simoncini, V., *A new iterative method for solving large-scale Lyapunov matrix equations*, SIAM J. Sci. Comput., 29(3):1268-1288, 2007.

[90] Simoncini, V., *Computational methods for linear matrix equations*, SIAM Rev., 58(3):377-441, 2016.

[91] Simoncini, V., Szyld, D.B., *Theory of inexact Krylov subspace methods and applications to scientific computing*, SIAM J. Sci. Comput., 25(2):454-477, 2003.

[92] Sitton, G.A., Burrus, C.S., Fox, J.W., Treitel, S., *Factoring very-high-degree polynomials*, IEEE Signal Processing Magazine, 20(6):27-42, 2003.

[93] Smith, R.A., *Matrix equation* $XA + BX = C$, SIAM J. Appl. Math., 16(1):198-201, 1968.

[94] Solov'ëv, S., *Preconditioned iterative methods for a class of nonlinear eigenvalue problems*, Linear Algebra Appl., 415:210-229, 2006.

[95] Sorensen, D., *Implicit application of polynomial filters in a k-step Arnoldi method*, SIAM J. Matrix Anal. Appl., 13:375-385, 1992.

[96] Sorensen, D., Zhou, Y., *Direct methods for matrix Sylvester and Lyapunov equations*, J. Appl. Math., 2003:277-303, 2003.

[97] Stewart, G.W., *Matrix Algorithms. Vol. II: Eigensystems*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2001.

[98] Stewart, G.W., *A Krylov-Schur algorithm for large eigenproblems*, SIAM J. Matrix Anal. Appl., 23(3):601-614, 2001/02.

[99] Su, Y., Bai, Z., *Solving rational eigenvalue problems via linearization*, SIAM J. Matrix Anal. Appl., 32(1):201-216, 2011.

[100] Su, Y., Zhang, J., Bai, Z., *A compact Arnoldi algorithm for polynomial eigenvalue problems*, In Recent Advances in Numerical Methods for Eigenvalue Problems (RANMEP2008), January 2008.

[101] Taussky, O., Wielandt, H., *On the matrix function* $AX + X'A'$, Arch. Rational Mech. Anal., 9:93-96, 1962.

[102] Trefethen, L.N., *Approximation Theory and Approximation Practice*, Society for Industrial and Applied Mathematics, Philadelphia, 2012.

[103] Tisseur, F., *Backward error and condition of polynomial eigenvalue problems*, Linear Algebra Appl., 309:339-361, 2000.

[104] Van Beeumen, R., Meerbergen, K., Michiels W., *Compact rational Krylov methods for nonlinear eigenvalue problems*, SIAM J. Matrix Anal. Appl., 36(2):820-838, 2015.

[105] Van Der Vorst, H. A., *Iterative Krylov Methods for Large Linear Systems*, Cambridge University Press, Cambridge, 2003.

[106] Vasilev, D., *Theoretical and practical aspects of linear and nonlinear model order reduction techniques*, PhD Thesis, Department of Electrical Engineering and Computer Science, MIT, 2008.

[107] Vorontsov, Y.O., Ikramov, Kh. D., *A numerical algorithm for solving the matrix equation* $AX + X^TB = C$, C. Zh. Vychisl. Mat. Mat. Fiz., 51(5):739-747, 2011.

[108] Voss, H., *A rational spectral problem in fluid-solid vibration*, Electron. Trans. Numer. Anal., 16:94-106, 2003.

[109] Voss, H., *Iterative projection methods for computing relevant energy states of a quantum dot*, BIT, 44:387-401, 2004.

[110] Walker, H.F., *Implementation of the GMRES method using Householder transformations*, SIAM J. Sci. Comput., 9:152-163, 1988.

[111] Wang, H.S., Yung, C.F., Chang, F.R., *Bounded real lemma and* H_∞ *control for descriptor systems*, Proceedings of the American Control Conference, Alburquerque, New Mexico, 2115-2119, 1997.

[112] Wang, L., Bo, Y., Chu, M.T., *A computational framework of gradient flows for general linear matrix equations*, Numer. Algorithms, 68(1):121-141, 2015.

[113] Watkins, D.S., *The Matrix Eigenvalue Problem: GR and Krylov Subspace Methods*, Society for Industrial and Applied Mathematics, Philadelphia, 2007.

[114] Wimmer, H.K., *Roth's theorems for matrix equations with symmetry constraints*, Linear Algebra Appl., 199:357-362, 1994.

[115] Yuan, Y., Dai, H., *The direct updating of damping and gyroscopic matrices*, J. Comput. Appl. Math., 231(1):255-261, 2009.