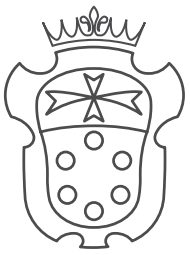


SCUOLA
NORMALE
SUPERIORE

Classe di Scienze

Corso di perfezionamento in
Computational Methods and Mathematical Models
for Sciences and Finance

XXXV ciclo

Block rational Krylov methods for matrix equations and matrix functions

Settore Scientifico Disciplinare **MAT/08**

Candidato
dr. Angelo Alberto Casulli

Relatore
Prof. Leonardo Robol

Supervisione interna
Prof. Michele Benzi

Anno accademico 2023–2024

Contents

List of Figures

List of Tables

Abstract

In this thesis, we describe block rational Krylov subspaces highlighting their correspondence with rational matrices and generalizing important properties that hold for the non-block case. We develop procedures based on block rational Krylov subspaces for the solution of Sylvester and tensor Sylvester equations, for the computation of functions of Hermitian hierarchically semiseparable (HSS) matrices and the action of functions of Hermitian matrices on block vectors.

In the context of Sylvester equations with low-rank right-hand sides, solvers based on Krylov subspaces are widely employed. By exploiting the correspondence between rational matrices and block rational Krylov subspaces, we develop a new formulation of the residual obtained when projection methods based on block rational Krylov subspaces are utilized. This formulation explicitly depends on the poles and represents a non-trivial extension of the result provided by Beckermann for Sylvester equations with rank-one right-hand sides. This extension establishes a connection between the convergence of block rational Krylov methods and the task of minimizing the norm of a small rational matrix across the spectrum or field-of-values of the involved matrices. In contrast to the rank-one scenario, where the minimization problem is scalar, here it becomes matrix-valued. Substituting the norm of the objective function with a more computationally tractable function leads to various adaptive pole selection strategies, offering a theoretical analysis for established heuristics as well as effective novel techniques. We also provide a procedure based on pole reordering to efficiently compute the residual.

A natural extension is represented by tensor Sylvester equations, which involve d summands. In these equations, both the unknown and the right-hand side are d-dimensional tensors. Specifically, for each i, the ith summand on the left-hand side multiplies the unknown by a matrix in the ith mode. Methods based on projection onto Krylov subspaces have previously been utilized in this context, but they typically assume a right-hand side of rank one. In this thesis, we demonstrate how to apply block rational Krylov methods to handle right-hand sides with low multilinear or tensor train rank. By extending the results established for Sylvester equations, we present a formulation of the residual based on rational matrices for the tensor case as well. This extension enables us to generalize the pole selection strategies and the techniques for computing the residual developed for classical Sylvester equations. An efficient computation of the residual becomes crucial in this scenario since explicitly constructing the residual is a highly expensive operation, which becomes impossible when dealing with a large tensor that cannot be fully stored.

Both Sylvester and tensor Sylvester solvers, employing block rational Krylov subspaces with adaptive pole selection, are tested on model problems derived from the discretization of partial differential equations. This testing demonstrates the efficiency of the proposed techniques compared to other strategies available in the literature. Particularly noteworthy is the effectiveness of rational Krylov subspaces in the tensor case. As the solution of the projected problem tends to be the most computationally demanding aspect, utilizing block rational Krylov subspaces enables the handling of smaller projected tensor Sylvester equations compared to their polynomial counterparts, leading to significant gains in terms of computational time and memory requirements.

In the context of matrix functions, we utilize block rational Krylov methods in an unconventional manner. Our aim is to leverage the rapid convergence of block rational Krylov subspaces while circumventing costly operations such as solving large linear systems.

A significant rank structure frequently encountered in various applications, particularly in the context of discretized (fractional) differential and integral operators, is known as Hierarchically semiseparability.

HSS matrices possess numerous appealing properties that aid in the creation of efficient algorithms. In this study, we present a fast algorithm designed to approximate $f(A)$, where A represents a Hermitian HSS matrix. We use a telescopic decomposition of A, inspired by the recent work of Levitt and Martinsson that allows us to approximate $f(A)$ by recursively performing low-rank updates with block rational Krylov subspaces while keeping the size of the matrices involved in the rational Krylov subspaces small. In particular, no large-scale linear system needs to be solved, which yields favorable complexity estimates and reduced execution times compared to existing methods, including an existing divide-and-conquer strategy. The advantages of our newly proposed algorithm are demonstrated in several examples from the literature, featuring the exponential, the inverse square root, and the sign function of a matrix. Even for matrix inversion, our algorithm exhibits superior performance, even if not specifically designed for this task.

Finally, the concepts used in computing functions of HSS matrices are extended to develop a memory-efficient algorithm for computing $f(A)C$, where A is a Hermitian matrix (not necessarily HSS), and C is a block vector. Our approach combines a block Lanczos algorithm with a basis compression technique based on block rational Krylov subspaces involving only small matrices. The computational effort required for the compression steps is minimal compared to the block Lanczos algorithm. This approach enables us to avoid storing the entire Lanczos basis, leading to significant reductions in memory usage. The method is particularly effective when the block Lanczos algorithm requires numerous iterations to converge. Theoretical results demonstrate that, for a wide variety of functions, the proposed algorithm differs from block Lanczos by an error term that is typically negligible. Comparisons with other low-memory Krylov methods from the literature on various test problems reveal competitive performance.

Chapter 1

Introduction

Over the past few decades, handling computations involving large-scale matrices and tensors has emerged as a core challenge in numerical linear algebra, aimed at solving problems stemming from science and engineering, such as electronic structure calculations [5, 16, 32, 115], micromagnetics [50, 51], control problems [3, 14], and fluid dynamics [47].

In this thesis, our focus is on the use of block Krylov methods, specifically their rational variants, to address significant tasks in numerical linear algebra: solving Sylvester and tensor Sylvester equations, as well as computing matrix functions.

To illustrate the significance of these tasks, we can consider a basic model problem involving the solution of the partial differential equation:

$$\begin{cases} \frac{\partial u(\mathbf{x},t)}{\partial t} = \Delta u(\mathbf{x},t) + f(\mathbf{x}) & \text{in } \Omega \times [0,\infty) \\ u(\mathbf{x},t) = 0 & \text{on } \partial\Omega \times [0,\infty) \\ u(\mathbf{x},0) = u_0(\mathbf{x}) & \text{in } \Omega \end{cases} \tag{1.1}$$

where Δ represents the Laplace operator, and $\Omega = [0,1]^d$. This problem is a classical model describing the temperature $u(\mathbf{x},t)$ within a hypercube at time t. Initially, the temperature distribution is determined by $u_0(\mathbf{x})$, and it evolves over time under the influence of an external heat source denoted by $f(\mathbf{x})$, while the boundary temperature remains constant at zero. When pursuing the steady-state solution (where $\frac{\partial u}{\partial t} = 0$), the problem transforms into:

$$\begin{cases} -\Delta u(\mathbf{x}) = f(\mathbf{x}) & \text{in } \Omega \\ u(\mathbf{x}) = 0 & \text{on } \partial\Omega, \end{cases} \tag{1.2}$$

which, through numerical discretization, reduces the problem to solving a (tensor) Sylvester equation. Moreover, the discretized solution of (1.1) for a specific time t can be obtained by additionally computing the product between a matrix function and a vector.

Krylov subspace methods are an essential tool in addressing matrix equations and matrix functions associated with large matrices [40, 41, 111, 113]. The primary concept behind Krylov subspace methods is projecting a large-scale problem onto a Krylov subspace, converting it into a smaller-scale problem that is easier to solve. Indeed, linear algebra problems involving small-sized matrices have been deeply studied, and several methods for solving a wide class of problems have been developed. In particular, due to the small size of the problem, techniques that do not leverage any specific matrix structure are often employed, usually referred to as dense methods.

In the context of Krylov methods, the most classical variants are the polynomial Krylov methods [55, 76, 88, 113, 114]. These methods are founded on the idea of constructing a subspace spanned by iteratively applying a matrix to a given vector, thereby providing an efficient framework for solving matrix equations and computing the action of a matrix function on a vector. Despite their remarkable success, polynomial Krylov methods are not without limitations. Specifically, their convergence can be attributed to the quality of polynomial approximation for functions relevant to the problem. For many problems of interest, the

associated function is poorly approximated by polynomials; therefore, polynomial Krylov methods may exhibit slow convergence rates or even fail to converge altogether. To address these challenges, researchers have turned to the rational Krylov methods, which offer a promising alternative. Rational Krylov methods [40, 66, 108, 110], extend the concept of polynomial Krylov methods by integrating the matrix-vector products with shifted linear systems. This flexibility allows for establishing a connection between the convergence of rational Krylov methods and rational approximations of functions relevant to the problem, resulting in faster convergence rates compared to their polynomial counterparts. Given a matrix $A \in \mathbb{C}^{n\times n}$, a vector $\mathbf{c} \in \mathbb{C}^n$ and a sequence of poles $\boldsymbol{\xi}_k = \{\xi_0, \ldots, \xi_{k-1}\} \subseteq \mathbb{C} \cup \{\infty\}$ that are not eigenvalues of A, the associated rational Krylov subspace $\mathcal{Q}_k(A, \mathbf{c}, \boldsymbol{\xi}_k)$ is given by

$$\text{span}\{(A-\xi_0 I)^{-1}\mathbf{c}, (A-\xi_1 I)^{-1}(A-\xi_0 I)^{-1}A\mathbf{c}, \ldots, (A-\xi_{k-1}I)^{-1}\cdots(A-\xi_0 I)^{-1}A^{k-1}\mathbf{c}\},$$

where I represents the identity matrix and with the convention that if a pole is equal to infinity, the corresponding shifted inverse of A reduces to the identity matrix. While rational Krylov methods frequently enable a satisfactory approximate solution using a smaller dimensional subspace, their construction requires solving shifted linear systems that are more computationally expensive compared to the matrix-vector products employed in the polynomial Krylov version.

Another notable variant of polynomial Krylov subspaces, which has garnered significant attention in recent years, is their block version employed, for instance, for the computation of the action of a matrix function on multiple vectors, see [93] and the references therein. In this setting, the matrix-vector product is replaced by a matrix-matrix product, involving a large square matrix and a block vector, that is, a tall and skinny matrix. An essential aspect for the effectiveness of block polynomial Krylov methods is their utilization of BLAS level 3 operations. BLAS (Basic Linear Algebra Subprograms) level 3 operations, optimized for matrix-matrix multiplications, provide a significant performance boost by exploiting parallelism and cache locality. By efficiently utilizing these operations, block polynomial Krylov methods can achieve substantial improvements in computational efficiency, particularly for large-scale problems involving multiple vectors.

It is important to note that block Krylov subspaces have been introduced in various forms, such as the *global* [22,45,73,78] and *loop-interchange* [107] variants. In this work, we focus on the *classical* block Krylov subspaces [34, 60, 112], which are larger than their variants mentioned above, thus usually allow for more accurate approximate solutions.

While block polynomial Krylov methods have been studied and applied in various contexts, their rational generalization remains relatively unexplored. The central aim of this thesis is to utilize block rational Krylov methods in the context of matrix equations and matrix functions.

In the solution of matrix equations, block rational Krylov methods offer broader applicability compared to traditional rational Krylov methods. Specifically, for a certain class of matrix equations, we demonstrate that when combined with a suitable pole selection strategy, they facilitate faster convergence compared to block polynomial methods. By advancing theoretical understanding regarding the convergence of block polynomial Krylov methods, we derive efficient methods for selecting poles, thus providing effective algorithms for the task.

In the context of Hermitian matrix functions, we employ block rational Krylov subspaces for the development of new procedures for the computation of functions of rank-structured matrices, often outperforming existing methods in computational complexity and execution time. Additionally, when only the action of a matrix function on a block vector is needed, we enhance the efficiency of the block Lanczos algorithm [88,112] by reducing memory requirements through a compression procedure based on block rational Krylov subspaces.

In Chapter 2, we introduce block rational Krylov subspaces and the necessary tools from the theory of matrix polynomials and rational functions. Furthermore, we extend the core principles of rational Krylov methods to the block framework, leveraging the advantages of rational approximation and block structure.

An important application of block rational Krylov methods regards the solution of the Sylvester equations

$$AX - XB = C, \tag{1.3}$$

in which the coefficient matrices A and B are large square matrices, and the right-hand side is low rank, that is, $C = C_1 C_2^H$ where C_1 and C_1 are tall and skinny matrices [121]. Sylvester equations commonly arise in control theory [3, 14] and in the numerical solution of PDEs on tensorized domains [103], such as the model problem in (1.2) with $\Omega = [0, 1]^2$.

Numerous authors have shown interest in the numerical solution of Sylvester equations, especially focusing on their variant, the Lyapunov equation, where $B = -A^H$, since it is frequently encountered in control problems [14]. When dealing with moderately sized matrices A and B, dense methods such as the Bartels-Stewart [7] and the Hessenberg Schur [59] algorithms are usually employed. However, dense solvers may become impractical for large coefficient matrices due to memory limitations and their computational cost, which typically grows cubically with the size of the matrices, thus, it becomes crucial to utilize the sparsity and data-sparsity structures of the involved matrices [8,82,104]. In cases where the right-hand side C is a low-rank matrix, exponential decay of the singular values of the solution X is frequently observed [13, 125], therefore, X can be well approximated by a low-rank matrix.

A widely used approach for solving Sylvester equations with a low-rank right-hand side is known as the Alternating Direction Implicit (ADI) iteration [15, 46]. This method entails generating a sequence of matrices that progressively converge towards the solution. The ADI method requires solving numerous linear systems with varying shifts, the selection of which significantly influences its convergence. As extensively investigated by Sabino in his thesis [116], theoretical insights into the convergence of the ADI method enable the development of techniques for determining a priori shifts. However, these techniques are often not easily applicable, or they result in slow convergence. Consequently, efficient techniques based on the adaptive determination of shifts during the algorithm have been developed; for a broader discussion on this topic, we refer to [121, Section 5.2.2].

When solving Sylvester equations, ADI and rational Krylov methods are closely linked, as discussed in [42, 91]. Specifically, Beckermann in [9] demonstrated that when dealing with a right-hand side of rank one, the ADI method cannot produce notably superior approximations compared to rational Krylov methods that employ the shifts used in the ADI iterations as poles.

The fundamental concept behind Krylov methods for solving Sylvester equations is that block rational Krylov subspaces involving A and B^H, with suitable poles, enable a highly accurate approximation of the spaces spanned by the columns and rows of X. Thus, denoting by U and V tall and skinny matrices, whose columns are orthonormal bases of the block rational Krylov subspaces associated with A and B^H, respectively, X can be well approximated by UYV^H, where Y is the solution of the Sylvester equation obtained from (1.3) by replacing A, B, and C with their projections into the block rational Krylov subspaces, that is $U^H AU$, $V^H BV$, and $U^H CV$. Therefore, the problem is transformed into a new Sylvester equation with smaller coefficient matrices that can be efficiently solved using dense methods. The accuracy of the approximate solution is heavily influenced by the selection of poles. Specifically, when the right-hand side of the original Sylvester equation has rank one, rational Krylov subspaces can be used instead of their block counterparts. In such instances, Beckermann developed a formulation of the residual that explicitly depends on the chosen poles [9].

Chapter 3 is devoted to proving a non-trivial extension of this formulation to the case of Sylvester equations with a low-rank right-hand side, employing block rational Krylov methods. It presents a new formulation of the residual based on a small rational matrix, enabling the design of an adaptive pole selection algorithm. In this algorithm, pole selection relies on minimizing the norm of a parameter-dependent matrix. Due to the numerical challenges associated with such minimization, it is natural to substitute the matrix with a simpler surrogate. We explore various options and demonstrate that one of these aligns with the heuristic proposed by Druskin and Simoncini in [42], providing a theoretical justification for this choice. Furthermore, we show that alternative choices for the surrogate function are viable. Specifically, we introduce an adaptive technique for pole selection that marginally enhances the approach proposed in [42]. The proposed results enable the development of an algorithm for solving Sylvester equations with a low-rank right-hand side based on projecting onto block rational Krylov subspaces, adaptively determining the poles. Furthermore, we demonstrate how to practically compute the residual with a low computational effort by appropriately reordering the poles of the block rational Krylov subspace.

Chapter 4 is dedicated to extending the results developed in Chapter 3 to tensor Sylvester equations

$$\mathcal{X} \times_1 A_1 + \mathcal{X} \times_2 A_2 + \cdots + \mathcal{X} \times_d A_d = \mathcal{C},$$

where, for each i, $\times_i$ denotes the ith mode product of tensors [81]. Here, the coefficient matrices A_i are large and square, while the right-hand side $\mathcal{C}$ and the unknown $\mathcal{X}$ are d-dimensional tensors. Such tensor equations arise, for example, in the solution of d-dimensional PDEs on tensorized domains [52, 63, 123], such as the model problem in (1.2) for general d.

The problem of solving tensor Sylvester equations can be reformulated as the solution of a linear system, which size grows exponentially in d. To address this challenge, several techniques have been developed to manage linear systems involving tensors, including TT-GMRES [36] and the AMEn method [37]. However, the convergence of these methods is not always clear. Specifically for solving tensor Sylvester equations, a generalization of the Bartels-Stewart algorithm has been provided in [30]. Nevertheless, its computational complexity is cubic in the size of the coefficient matrices, and the required memory storage scales exponentially in d, limiting its applicability to small problems. Similar to classical Sylvester equations, exploiting the structure of large matrices becomes essential for their tensor counterparts [52, 96]. When the right-hand side $\mathcal{C}$ has rank one, Krylov methods, specifically in their polynomial variant, have been previously employed by Kressner and Tobler [84]. Our contribution can be seen as a generalization of such work for cases where the right-hand side has low multilinear or tensor train rank, achieved by employing block rational Krylov subspaces. The formulation of the residual in terms of poles allows us to generalize adaptive pole selection strategies introduced in Chapter 3. Moreover, the development of an efficient way to compute the residual norm, using a lower amount of memory, is fundamental because the residual is typically a large tensor that may not even be stored explicitly.

The application of block rational Krylov methods in solving tensor Sylvester equations underscores the efficacy of our approach. While Sylvester equations with low-rank right-hand sides can be decomposed into a small number of Sylvester equations with right-hand side of rank one solvable with rational Krylov methods, extending this procedure to the tensor case is often impractical. This is because representing a tensor with low multilinear or tensor train rank as the sum of rank-one tensors typically requires a large number of summands, and determining the smallest number of terms is an NP-Hard problem [72]. Block rational Krylov methods effectively address this challenge by managing the right-hand side in Tucker or tensor train format. Additionally, solving the projected tensor Sylvester equations often constitutes the most computationally demanding aspect of the procedure, rendering the computational effort needed for solving shifted linear systems involving the matrices A_i relatively small. By selecting suitable poles, it becomes feasible to solve a projected tensor Sylvester equation with significantly smaller coefficient sizes compared to those generated by block polynomial Krylov methods. Consequently, the block rational variant yields significantly faster solvers.

In Chapters 5 and 6, we employ block rational Krylov methods in the setting of Hermitian matrix functions. Consider a Hermitian matrix $A \in \mathbb{C}^{n\times n}$ with spectral decomposition $A = V\Lambda V^H$, with the orthogonal matrix V and the diagonal matrix $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$ containing the eigenvalues of A. Given a scalar function f well defined on the eigenvalues of A, the matrix function $f(A) \in \mathbb{C}^{n\times n}$ is defined as $Vf(\Lambda)V^H$, where $f(\Lambda) := \operatorname{diag}(f(\lambda_1), \ldots, f(\lambda_n))$. Popular examples include the matrix inverse, the matrix exponential, the sign function, and the (inverse) matrix square root; see the monograph [74] for an overview. Matrix functions find significant applications in the discretized solution of partial differential equations [39, 89] and network analysis [49]. In particular, the solution of the numerical discretization of the model problem in (1.1) is given by $\exp(tA)(\mathbf{u}_0 + \mathbf{f}) - A^{-1}\mathbf{f}$, where A, $\mathbf{u}_0$ and $\mathbf{f}$ are discretizations of the Laplace operator, $u_0(\mathbf{x})$ and $f(\mathbf{x})$, respectively.

In the general case, for any (block) vector C, computing $f(A)C$ using its definition requires a computational effort that is cubic in the size of A, making it unfeasible for large matrices. In such cases, both polynomial and rational Krylov methods, in both standard and block settings, find extensive application (see [66, 93]). Similar to the context of matrix equations, these methods involve projecting the problem into a Krylov subspace generated by A and C, thereby reducing it to computing the action of the small matrix function $f(U^HAU)$ on the (block) vector U^HC, where U denotes an orthonormal basis of the Krylov

subspace.

The accuracy of the approximation achieved by polynomial and rational Krylov subspaces depends on the approximation error of f across the spectrum of A, using polynomial and rational functions, respectively, whose degree is proportional to the dimension of the Krylov subspace. Consequently, rational Krylov methods have the potential to yield superior approximations using smaller subspaces. However, it is important to stress once again that they require solving shifted linear systems with A, which can render them impractical or potentially reduce their computational advantage, the extent of which is not always clear. Therefore, several authors have studied techniques to employ polynomial Krylov methods, circumventing the need to store the complete basis of the Krylov subspace, to handle potentially large Krylov subspaces. These techniques include the low-memory Lanczos method [21, 57], the multi-shift conjugate gradient method [55, 127], and strategies based on restarting [43, 53, 54]. For a more comprehensive treatment of the topic, we refer to the surveys [68, 69].

In this thesis, we utilize block rational Krylov methods within the context of matrix functions in an unconventional manner. Our objective is to exploit the "anticipating the future" feature of block rational Krylov methods, thereby circumventing the challenge of solving large linear systems. We achieve this by leveraging block rational Krylov subspaces associated with smaller matrices.

When A is a large matrix, exploiting the structure of A becomes crucial for the development of efficient algorithms for the computation of $f(A)$, as demonstrated in works such as [33, 56, 105]. In Chapter 5 we develop a fast algorithm for approximating the whole matrix function $f(A)$ of a square matrix A that is Hermitian and has hierarchically semiseparable (HSS) structure, with f well approximable by a rational function, fully exploiting the data-sparsity of A.

A matrix $A \in \mathbb{C}^{n\times n}$ is hierarchically semiseparable if it can be recursively partitioned as

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

where the matrices A_{12} and A_{21} are of low rank, and A_{11} and A_{22} are square matrices that can be recursively partitioned in the same manner. Furthermore, the low-rank factors representing the off-diagonal blocks at different levels of recursion are nested. For a more detailed description, refer to [94, 97, 129]. Notable examples include banded matrices, moreover, rational functions of HSS matrices remain HSS [33]. Consequently, this structure synergizes effectively with functions that can be well-approximated by rational functions, allowing the development of several methods to manage linear algebra tasks [33, 64, 85].

We introduce a general framework for telescopic decomposition for HSS matrices, which generalizes the data sparse representation employed by Levitt and Martinsson in [90] for approximating an HSS matrix using matrix-vector products with a few random vectors. In practice, we decompose A into the sum of a block diagonal and a low-rank matrices, where the latter one involves a smaller HSS matrix which is recursively defined. This allows us to approximate $f(A)$ by computing functions of low-rank updates of matrices using block rational Krylov methods as described in [10]. Additionally, the specific structure of the matrices enables us to construct block rational Krylov subspaces only involving small matrices, thus avoiding the solution of large linear systems.

Although not all matrices exhibit the HSS structure, any Hermitian matrix can be transformed into a (block) tridiagonal matrix via unitary similarity employing direct methods [128, Section 5.5]. Since block tridiagonal matrices exhibit the HSS property, this provides a theoretical framework for extending the procedure introduced in Chapter 5 to arbitrary matrices. This observation is purely of theoretical interest, as the computational cost of the Hessenberg reduction scales cubically with the size of the involved matrix, rendering it impractical for large matrices. However, in the context of computing the action of a matrix function on a block vector, the block Lanczos algorithm [88, 112] behaves similarly to a Hessenberg reduction, transforming a Hermitian matrix into a block tridiagonal one. This concept forms the core idea driving the development of the contents in Chapter 6.

In Chapter 6, we propose a memory-efficient algorithm for computing $f(A)C$, where A is a Hermitian matrix and C is a block vector. Our method combines an outer block Lanczos algorithm with a procedure based on inner block rational Krylov subspaces for compressing the Lanczos basis, thereby avoiding

the storage of the entire Lanczos basis. Specifically, the construction of the inner block rational Krylov subspace involves only small matrices, making the computational effort needed for basis compression negligible compared to the cost of the block Lanczos algorithm. The approximation provided by the proposed procedure differs from that of the block Lanczos algorithm by an error that is negligible if f can be effectively approximated by a rational function across the spectrum of A. This applies to important functions, such as the exponential and the (inverse) square root.

The proposed algorithm exhibits performance that is comparable to or better than other low-memory methods proposed in the literature, such as the two-pass Lanczos method [21, 57], techniques based on multishift conjugate gradient [55, 127], and restarted Krylov methods [43, 53, 54].

Each of the following chapters, except for Chapter 2, concludes with a section on numerical experiments. In these experiments, the proposed algorithms are systematically compared with other methods available in the literature using various test problems.

1.1 Notation

We utilize a variety of scripts and boldface styles, as detailed in Table 1.1, to distinguish between different objects. We maintain strict adherence to these choices whenever possible, ensuring that an object's usage can be determined from its typeface. The field of values (or numerical range) of a matrix A, denoted by $\mathbb{W}(A)$, is defined as the set $\{\mathbf{x}^H A\mathbf{x} : \mathbf{x}^H\mathbf{x} = 1\}$. While determining the field of values of a matrix is generally challenging, there are cases where it can be easily computed. For example, if A is normal, its field of values is the convex hull of its spectrum. Specifically, if A is Hermitian, its field of values is the interval $[\lambda_{\min}, \lambda_{\max}]$, where $\lambda_{\min}$ and $\lambda_{\max}$ are the minimum and maximum eigenvalues of A. We employ a Matlab-like notation for submatrices, for instance, given $A \in \mathbb{C}^{m\times n}$ the matrix $A_{i_1:i_2,j_1:j_2}$ is the submatrix obtained selecting only rows from i_1 to i_2 and columns from j_1 to j_2 (extrema included). Given two vectors, $\mathbf{k}$, $\mathbf{h}$ with d components, the notation $\mathbf{h} \leq \mathbf{k}$ means that $\mathbf{h}$ is component-wise smaller than $\mathbf{k}$. We often use the terminology "block vectors", to indicate tall and skinny matrices and we usually employ b to denote the block size, that is, the number of columns of block vectors. To simplify the notation we use bold letters to denote block indices, that is, we use $\mathbf{s}$ to denote the set of indices $b(s-1)+1 : bs$. Given a list of matrices A_i, we denote the block diagonal matrix containing these matrices A_i as diagonal blocks by $\mathrm{blkdiag}(A_i)$. It's worth noting that in our scenario, the matrices A_i might be rectangular, and consequently, so is $\mathrm{blkdiag}(A_i)$. For a compact set $\mathcal{S} \subset \mathbb{C}$ and a continuous function $f : \mathcal{S} \to \mathbb{C}$, we denote by $\|f\|_{\mathcal{S}}$ the supremum norm of the function f on $\mathcal{S}$. When the domain of the function f is evident from the context, we denote the supremum norm of f on its domain of definition as $\|f\|_\infty$.

Table 1.1: Symbol Glossary

Symbol	Description	Example
$\mathbb{C}$	space of complex numbers	
$\overline{\mathbb{C}}$	space of extended complex numbers (i.e. $\mathbb{C} \cup \{\infty\}$)	
$\mathbb{R}$	space of real numbers	
$\mathbb{N}$	space of natural numbers	
$\mathcal{P}$	space of polynomials with complex coefficients	
$\mathcal{P}_d$	space of polynomials with complex coefficients of degree bounded by d	

Continued on next page

Table 1.1 Continued from previous page

Symbol	Description	Example
$\mathcal{P}(\mathbb{C}^{b\times b})$	space of matrix polynomials with coefficients in $\mathbb{C}^{b\times b}$	
$\mathcal{P}_d(\mathbb{C}^{b\times b})$	space of matrix polynomials with coefficients in $\mathbb{C}^{b\times b}$ of degree bounded by d	
Lower case Latin letters	natural numbers and scalar polynomials	$n, m, b, d, k \in \mathbb{N}$, $q \in \mathcal{P}$
Upper case Latin letters	matrices, block vectors (i.e., tall and skinny matrices), matrix polynomials and rational matrices	$A \in \mathbb{C}^{n\times n}, C \in \mathbb{C}^{n\times b}$, $P \in \mathcal{P}(\mathbb{C}^{b\times b})$, $R \in \mathcal{P}(\mathbb{C}^{b\times b})/q$
Uppercase Greek letters	small square matrices (typically coefficients of matrix polynomials)	Γ, Δ
Bold lowercase latin letters	vectors of complex or natural numbers	$\mathbf{c} \in \mathbb{C}^n, \mathbf{k} \in \mathbb{N}^d$
Bold lowercase greek letters	sequence of extended complex numbers (usually poles of rational Krylov subspaces)	$\boldsymbol{\xi}_k \in \overline{\mathbb{C}}^k$
Uppercase calligraphic Latin letters	tensors	$\mathcal{X}, \mathcal{C} \in \mathbb{C}^{n_1\times\cdots\times n_d}$
Superscripts $\overline{\ }$, ^{H} and ^{T}	conjugate, conjugate transpose and transpose respectively	$\overline{A}, A^H, A^T$ where $A \in \mathbb{C}^{n\times n}$
$\otimes$	Kronecker product	
$\oplus$	Kronecker sum	
$\mathrm{vec}(\cdot)$	transforms a matrix or a tensor into a vector ordering its entry lexicographically	if $C \in \mathbb{C}^{n\times b}$, then $\mathrm{vec}(C) \in \mathbb{C}^{nb}$
$\Lambda(\cdot)$	spectrum of a matrix	$\Lambda(A)$, where $A \in \mathbb{C}^{n\times n}$
$\mathcal{P}_{\mathbf{k}}$	space of multivariate polynomials with complex coefficients and degrees bounded by the vector $\mathbf{k}$	
$\mathcal{P}_{\mathbf{k}}(\mathbb{C}^{b\times b})$	space of multivariate matrix polynomials with coefficients in $\mathbb{C}^{b\times b}$ and degrees bounded by the vector $\mathbf{k}$	
$\mathbb{W}(\cdot)$	field-of-values of a matrix	$\mathbb{W}(A)$, where $A \in \mathbb{C}^{n\times n}$
$\sigma(\cdot)$	set of singular values of a matrix	$\sigma(A)$, where $A \in \mathbb{C}^{n\times n}$
I	identity matrix (sometimes with a subscript that denotes the dimension)	$I_b \in \mathbb{C}^{b\times b}$

Continued on next page

Table 1.1 Continued from previous page

Symbol	Description	Example
E_i	block vector defined as $\mathbf{e}_i \otimes I_b$, where $\mathbf{e}_i$ is the ith element of the canonical basis	
$\|\cdot\|_F$	Frobenius norm	
$\|\cdot\|_2$	Euclidean norm	

Chapter 2

Block rational Krylov methods

In the last decades, Krylov methods have emerged as a fundamental tool for solving matrix equations [121] and computing the action of matrix functions on (block) vectors [66]. Following the success of polynomial Krylov methods, their rational counterparts were introduced, offering the distinct advantage of faster convergence in terms of required iterations, with the trade-off of solving potentially large linear systems. While the block generalization of polynomial Krylov methods has been extensively studied (see [93] and the references therein), the block variant of rational Krylov methods remains relatively underexplored in the literature. Although some progress has been made, as evidenced in [48], the available literature lacks the breadth and depth found in its polynomial counterpart.

This chapter aims to address this disparity by introducing block rational Krylov subspaces and elucidating their principal properties. Our exploration begins with an investigation into matrix polynomials and their associated rational matrices, providing the foundational knowledge necessary to relate block rational Krylov subspaces and rational functions. Additionally, we aim to generalize important properties from rational Krylov subspaces to their block counterparts, such as the exactness on rational functions.

2.1 Matrix polynomials and rational functions

In this section, we provide some definitions and properties about matrix polynomials that we use in the thesis.

Matrix polynomials can be equivalently interpreted as polynomials with a scalar variable and matrix coefficients or as a matrix with polynomial entries. Both interpretations can be useful for proving different results.

Definition 2.1.1 ([58]). *If $\Gamma_0, \dots, \Gamma_d$ are $b \times b$ complex matrices and Γ_d is nonzero, the matrix-valued function defined on the complex numbers by*

$$P(z) := \sum_{i=0}^{d} z^i \Gamma_i,$$

is called matrix polynomial of degree d. *When Γ_d is the identity matrix, the matrix polynomial is said to be monic.*

Formally, we will denote by $\mathcal{P}(\mathbb{C}^{b\times b})$ the space of $b \times b$ matrix polynomials, with coefficients in $\mathbb{C}^{b\times b}$. We use the notation $\mathcal{P}_d(\mathbb{C}^{b\times b})$ to denote the set of matrix polynomials of degree less than or equal to d.

In order to analyze block Krylov methods, we associate a matrix polynomials with a linear operator that acts on block vectors. More precisely, we define an operator $\circ$ as a function from $\mathbb{C}^{n\times n} \times \mathbb{C}^{n\times b}$ to $\mathbb{C}^{n\times b}$ as follows: given two matrices $A \in \mathbb{C}^{n\times n}$ and $C \in \mathbb{C}^{n\times b}$, we set

$$P(A) \circ C := \sum_{i=0}^{d} A^i C \Gamma_i.$$

This notation has already been used in [79,119,122], and has been exploited in [93] for the analysis of block Krylov subspaces. If the matrix A is fixed, the map $C \mapsto P(A) \circ C$ is a linear function from $\mathbb{C}^{n\times b}$ to $\mathbb{C}^{n\times b}$. When dealing with rational Krylov methods, it will often be useful to apply the inverse of the operator, that is given a generic vector C finding another block vector Z such that $P(A) \circ Z = C$. Since the operator is linear in Z, this is equivalent to solving a linear system. A formal definition can be given as follows.

Definition 2.1.2. *Given a matrix $A \in \mathbb{C}^{n\times n}$, a block vector $C \in \mathbb{C}^{n\times b}$ and a matrix polynomial $P(z) = \sum_{i=0}^{d} z^i \Gamma_i \in \mathcal{P}(\mathbb{C}^{b\times b})$, such that $det(P(\lambda)) \neq 0$ for each λ eigenvalue of A, we define $P(A) \circ^{-1} C$ as the block vector $Z \in \mathbb{C}^{n\times b}$, such that $P(A) \circ Z = C$.*

Since Z is implicitly defined as the solution of a linear system, we shall check that the system is invertible to ensure that the definition is well-posed.

Lemma 2.1.1. *Given a matrix $A \in \mathbb{C}^{n\times n}$, a block vector $C \in \mathbb{C}^{n\times b}$, and a matrix polynomial $P(z)$ as above such that $\det(P(\lambda)) \neq 0$ for $\lambda \in \Lambda(A)$, there is a unique $Z \in \mathbb{C}^{n\times b}$ verifying $P(A) \circ Z = C$.*

Proof. The relation $P(A) \circ Z = C$ can be rewritten as $\mathrm{vec}(P(A) \circ Z) = \mathrm{vec}(C)$, in addition, we note that

$$\mathrm{vec}(P(A) \circ Z) = \left(\sum_{i=0}^{d} \Gamma_i^T \otimes A^i \right) \mathrm{vec}(Z),$$

where $\otimes$ denotes the Kronecker product, and we used the standard Kronecker relation $\mathrm{vec}(AXB) = (B^T \otimes A)\mathrm{vec}(X)$. We now prove that the matrix $\sum_{i=0}^{d} \Gamma_i^T \otimes A^i$ is invertible, which implies the sought claim, since Z can be defined as

$$Z = \mathrm{vec}^{-1} \left(\left(\sum_{i=0}^{d} \Gamma_i^T \otimes A^i \right)^{-1} \mathrm{vec}(C) \right).$$

Let $A = UTU^H$ be a Schur decomposition of A, with T upper triangular, then

$$\sum_{i=0}^{d} \Gamma_i^T \otimes A^i = (I_b \otimes U) \left(\sum_{i=0}^{d} \Gamma_i^T \otimes T^i \right) \left(I_b \otimes U^H \right).$$

There exists a permutation matrix $S \in \mathbb{C}^{nb\times nb}$ (the "perfect shuffle", see [61]), such that

$$\sum_{i=0}^{d} \Gamma_i^T \otimes T^i = S \left(\sum_{i=0}^{d} T^i \otimes \Gamma_i^T \right) S^H.$$

Hence, it is sufficient to prove the invertibility of $\sum_{i=0}^{d} T^i \otimes \Gamma_i^T$ that is a block triangular matrix with block diagonal matrices given by $P(\lambda_1)^T, \ldots, P(\lambda_n)^T$, where λ_i are the eigenvalues of A. Therefore, the assumption $\det(P(\lambda)) \neq 0$ for each λ eigenvalue of A yields the claim. □

Remark 2.1.1. *The proof of well-posedness of Definition 2.1.2 also gives us an explicit representation of $P(A)\circ^{-1}$: for any $C \in \mathbb{C}^{n\times b}$*

$$P(A) \circ^{-1} C = \mathrm{vec}^{-1} \left(\left(\sum_{i=0}^{d} \Gamma_i^T \otimes A^i \right)^{-1} \mathrm{vec}(C) \right).$$

In particular, the hypothesis $\det(P(\lambda)) \neq 0$ for $\lambda \in \Lambda(A)$ is necessary to guarantee the invertibiliy of $\sum_{i=0}^{d} \Gamma_i^T \otimes A^i$.

The previous definitions and results essentially deal with matrix polynomials; for rational Krylov methods, we will need a way to incorporate rational functions into the picture. In practice, it will be sufficient to consider objects of the form $q(z)^{-1}P(z)$, where $q(z)$ is a scalar polynomial, and $P(z)$ a matrix polynomial. It is immediate to check that any rational matrix (i.e., a matrix with rational entries) can be always written in this form.

The following lemma suggests a way to extend the operators $\circ$ and $\circ^{-1}$ to rational matrix polynomials with scalar denominator.

Lemma 2.1.2. *Let $P(z) = \sum_{i=0}^{d} \Gamma_i z^i \in \mathcal{P}_d(\mathbb{C}^{b\times b})$ and let $q(z) \in \mathcal{P}_k(\mathbb{C})$ be a scalar polynomial. Denoting by $\widetilde{P}(z) = q(z)P(z) = \sum_{i=0}^{d+k} \Delta_i z^i$, it holds*

$$q(A) \cdot (P(A) \circ C) = \widetilde{P}(A) \circ C \quad \text{and} \quad q(A)^{-1} \cdot (P(A) \circ^{-1} C) = \widetilde{P}(A) \circ^{-1} C,$$

where in the second equality we assume $\det(\widetilde{P}(\lambda)) \neq 0$ for each $\lambda \in \Lambda(A)$.

Proof. To derive the first equality it is sufficient to prove the case of $q(z) = z - \alpha$ for $\alpha \in \mathbb{C}$, since we can factor $q(z)$ as the product of linear terms. By definition of $\widetilde{P}(z)$,

$$\widetilde{P}(z) = (z-\alpha)P(z) = \sum_{i=0}^{d+1} (\Gamma_{i-i} - \alpha\Gamma_i) z^i,$$

with the convention that $\Gamma_{-1} = \Gamma_{d+1} = 0$. In particular $\Delta_i = \Gamma_{i-1} - \alpha\Gamma_i$. Hence,

$$\begin{aligned}\widetilde{P}(A) \circ C =& \sum_{i=0}^{d+1} A^i C \Delta_i = \sum_{i=0}^{d} A^{i+1} C \Gamma_i - \alpha \sum_{i=0}^{d} A^i C \Gamma_i \\ =& A \cdot P(A) \circ C - \alpha P(A) \circ C = (A - \alpha I_n) \cdot (P(A) \circ C) = q(A) \cdot (P(A) \circ C).\end{aligned}$$

For the second identity, it is sufficient to prove that $\widetilde{P}(A) \circ (q(A)^{-1} Z) = C$, where $Z = P(A) \circ^{-1} C$. Using the first identity,

$$\widetilde{P}(A) \circ (q(A)^{-1} Z) = q(A) \cdot (P(A) \circ (q(A)^{-1} Z)) = q(A) \sum_{i=0}^{d} A^i Q(A)^{-1} Z \Gamma_i.$$

Since $q(A)$ commutes with the powers of A, this can be reduced to

$$\widetilde{P}(A) \circ (q(A)^{-1} Z) = P(A) \circ Z.$$

By definition of Z it follows that $P(A) \circ Z = C$, that concludes the proof. □

In view of the previous result, we can extend the action of a matrix polynomial $P(A) \circ C$ to the case of rational matrices with prescribed poles.

Definition 2.1.3. *Let $q(z) \in \mathcal{P}(\mathbb{C})$ and let $R(z) \in \mathcal{P}(\mathbb{C}^{b\times b})/q(z)$, that is there exists $P(z) \in \mathcal{P}(\mathbb{C}^{b\times b})$ such that $R(z) = P(z)/q(z)$. Given $A \in \mathbb{C}^{n\times n}$ such that $q(A)$ is invertible and $C \in \mathbb{C}^{n\times b}$, we define*

$$R(A) \circ C = q(A)^{-1} (P(A) \circ C) \quad \text{and} \quad R(A) \circ^{-1} C = q(A) \left(P(A) \circ^{-1} C\right).$$

The expression of a rational matrix in the form $R(z) = P(z)/q(z)$ is not unique; however the previous definition does not depend on the representation, indeed if $R(z) = P(z)/q(z) = \widetilde{P}(z)/\widetilde{q}(z)$, then $q(z)\widetilde{P}(z) = \widetilde{q}(z)P(z)$, hence by Lemma 2.1.2,

$$q(A) \cdot (\widetilde{P}(A) \circ C) = \widetilde{q}(A) \cdot (P(A) \circ C) \tag{2.1}$$

and

$$q(A)^{-1} \cdot (\widetilde{P}(A) \circ^{-1} C) = \widetilde{q}(A)^{-1} \cdot (P(A) \circ^{-1} C). \tag{2.2}$$

Multiplying both sides of (2.1) on the left by $q(A)^{-1} \cdot \widetilde{q}(A)^{-1}$ we obtain the well-posedness of the map $C \mapsto R(A) \circ C$, and multiplying both sides of (2.2) on the left by $q(A)\widetilde{q}(A)$ we have the well-posedness of the map $C \mapsto R(A) \circ^{-1} C$.

Remark 2.1.2. *If the matrix A is fixed, both operators*

$$C \mapsto R(A) \circ C \quad \text{and} \quad C \mapsto R(A) \circ^{-1} C$$

are linear. As in the polynomial case, the latter is only defined if $R(z)$ is nonsingular over all the eigenvalues of A.

Lemma 2.1.3. *If $A, B \in \mathbb{C}^{n\times n}$ commute, then for every rational matrix $R(z) = P(z)/q(z)$, where $P(z) \in \mathcal{P}(\mathbb{C}^{b\times b})$ and $q(z) \in \mathcal{P}(\mathbb{C})$,*

$$B \cdot R(A) \circ C = R(A) \circ (BC),$$

moreover, if $det(P(\lambda)) \neq 0$ for each $\lambda \in \Lambda(A)$,

$$B \cdot R(A) \circ^{-1} C = R(A) \circ^{-1} (BC).$$

Proof. Let $P(z) = \sum_{i=1}^{d} z^i \Gamma_i \in \mathcal{P}_d(\mathbb{C}^{b\times b})$ and $q(z) \in \mathcal{P}(\mathbb{C})$, such that $R(z) = P(z)/q(z)$. Then

$$B \cdot R(A) \circ C = BQ(A)^{-1} \sum_{i=1}^{d} A^i C \Gamma_i = q(A)^{-1} \sum_{i=1}^{d} A^i BC\Gamma_i = R(A) \circ (BC),$$

and therefore

$$\begin{aligned}\operatorname{vec}(B \cdot R(A) \circ^{-1} C) &= (I_b \otimes B)(I_b \otimes q(A)) \left(\sum_{i=1}^{d} \Gamma_i^T \otimes A^i\right)^{-1} \operatorname{vec}(C) \\ &= (I_b \otimes q(A)) \left(\sum_{i=1}^{d} \Gamma_i^T \otimes A^i\right)^{-1} (I_b \otimes B)\operatorname{vec}(C) = \operatorname{vec}(R(A) \circ^{-1} (BC)).\end{aligned}$$

□

Given a matrix polynomial $P(z) = \sum_{i=0}^{d} z^i \Gamma_i$, we denote by $P^H(z)$ the matrix polynomial $P^H(z) := \sum_{i=0}^{d} z^i \Gamma_i^H$. Similarly, we denote by $\bar{P}(z)$ the matrix polynomial with complex conjugate (but not transposed) coefficients. Given a function $R(z) = P(z)/q(z)$, we denote by $\bar{R}(z)$ and $R^H(z)$ the rational functions $\bar{P}(z)/\bar{q}(z)$ and $P^H(z)/\bar{q}(z)$, respectively.

Lemma 2.1.4. *Given $C \in \mathbb{C}^{n\times b}$ and $Z \in \mathbb{C}^{m\times b}$, the following identities hold:*

$$R(zI_n) \circ^{-1} C = C(R(z))^{-1} \qquad R(zI_n) \circ^{-1} CZ^H = C(R^H(\bar{z}I_m) \circ^{-1} Z)^H.$$

Proof. Let $P(z) = \sum_{i=1}^{d} z^i \Gamma_i \in \mathcal{P}_d(\mathbb{C}^{b\times b})$ and $q(z) \in \mathcal{P}(\mathbb{C})$, such that $R(z) = P(z)/q(z)$. It holds

$$\begin{aligned}\operatorname{vec}\left(R(zI_n) \circ^{-1} C\right) &= q(z) \left(\sum_{i=0}^{d} \Gamma_i^T \otimes z^i I_n\right)^{-1} \operatorname{vec}(C) \\ &= \left(\left(R^T(z)\right)^{-1} \otimes I_n\right) \operatorname{vec}(C) = \operatorname{vec}\left(\mathbf{v}(R(z))^{-1}\right),\end{aligned}$$

from which follows the first equality. For the second identity notice that

$$R(zI_n) \circ^{-1} CZ^H = C(R(z))^{-1} Z^H = C(Z(R^H(\bar{z}))^{-1})^H = C(R^H(\bar{z}I_m) \circ^{-1} Z)^H.$$

□

The following theorem is a generalization of the Cauchy integral formula to the action of rational matrices.

Theorem 2.1.5. *Let $A \in \mathbb{C}^{n\times n}$, $C \in \mathbb{C}^{n\times b}$ and let γ be a compact contour that encloses once the eigenvalues of A with positive orientation. Then, for any $R(z) \in \mathcal{P}(\mathbb{C}^{b\times b})/q(z)$, such that $\det(R(z)) \neq 0$ for each z in the compact set enclosed by γ, it holds*

$$\frac{1}{2\pi i}\int_\gamma R(zI_n) \circ^{-1} \left[(zI_n - A)^{-1}C\right]\, dz = R(A) \circ^{-1} C.$$

Proof. Let $P(z) = \sum_{i=1}^d z^i\Gamma_i$, be such that $R(z) = P(z)/q(z)$. Then

$$\begin{aligned}
&\operatorname{vec}\left(\int_\gamma R(zI_n) \circ^{-1} \left[(zI_n - A)^{-1}C\right]\, dz\right)\\
&= \left(\int_\gamma q(z)\left(\sum_{i=0}^d \Gamma_i^T \otimes z^i I_n\right)^{-1} \cdot \left(I_n \otimes (zI_n - A)^{-1}\right) dz\right)\operatorname{vec}(C)\\
&= \left(\int_\gamma q(z)\left(\sum_{i=0}^d \Gamma_i^T z^i\right)^{-1} \otimes (zI_n - A)^{-1}\, dz\right)\operatorname{vec}(C).
\end{aligned}$$

For each $s,t \in \{1,\dots,b\}$, let $f_{s,t}(z)$ be the function that maps z in the entry in position (s,t) of the rational matrix $q(z)\left(\sum_{i=0}^d \Gamma_i^T z^i\right)^{-1}$. Since for each z inside the compact set bounded by γ it holds $\det(R(z)) \neq 0$, the functions $f_{s,t}(z)$, are holomorphic on such set. Then for the Cauchy integral formula, we have

$$\frac{1}{2\pi i}\int_\gamma q(z)\left(\sum_{i=0}^d \Gamma_i^T z^i\right)^{-1}_{s,t} \otimes (zI_n - A)^{-1}\, dz = \frac{1}{2\pi i}\int_\gamma f_{s,t}(z)\cdot(zI_n - A)^{-1}\, dz = f_{s,t}(A).$$

Then, if we denote by $F \in \mathbb{C}^{nb\times nb}$ the block matrix for which the block in position (s,t) is defined by $f_{s,t}(A)$, we have the equivalence

$$F = \frac{1}{2\pi i}\int_\gamma q(z)\left(\sum_{i=0}^d \Gamma_i^T z^i\right)^{-1} \otimes (zI_n - A)^{-1}\, dz.$$

We now claim that $F = (I_b \otimes q(A))\left(\sum_{i=0}^d \Gamma_i^T \otimes A^i\right)^{-1}$, which implies the sought results, since

$$\begin{aligned}
\operatorname{vec}\left(\frac{1}{2\pi i}\int_\gamma R(zI_n) \circ^{-1} (zI_n - A)^{-1}C\, dz\right) &= F\cdot\operatorname{vec}(C)\\
&= (I_b \otimes q(A))\left(\sum_{i=0}^d \Gamma_i^T \otimes A^i\right)^{-1}\operatorname{vec}(C) = \operatorname{vec}\left(R(A) \circ^{-1} C\right).
\end{aligned}$$

Hence in the following we prove that $\left(\sum_{i=0}^d \Gamma_i^T \otimes A^i\right)\cdot F = I_b \otimes q(A)$.

For any $s,t \in \{1,\dots,b\}$, let us define $g_{s,t}(z) = \left(\sum_{i=0}^d \Gamma_i^T z^i\right)_{s,t}$. Since

$$\left(\sum_{i=0}^d \Gamma_i^T z^i\right)\cdot\left[q(z)\left(\sum_{i=0}^d \Gamma_i^T z^i\right)^{-1}\right] = q(z)I_b,$$

it holds

$$q(z)\delta_{s,t} = \sum_{r=1}^b g_{s,r}(z)f_{r,t}(z), \tag{2.3}$$

where $\delta_{s,t}$ denotes the Kronecker delta.

To simplify the notation, for any integer $r \in \{1, \dots, b\}$, we define $ix(r)$ as the set of indices $n(r-1)+1 : nr$. For any $s, t \in \{1, \dots, b\}$ we have

$$\left(\left(\sum_{i=0}^{d} \Gamma_i^T \otimes A^i\right) \cdot F\right)_{ix(s),ix(t)} = \sum_{r=1}^{b} \left(\sum_{i=0}^{d} (\Gamma_i^T)_{s,r} \cdot A^i\right) f(A)_{r,t} = \sum_{r=1}^{b} g_{s,r}(A) f_{r,t}(A) = \delta_{s,t} q(A),$$

where the last equality follows from (2.3).

□

Let us now recall the concept of divisibility for matrix polynomials and the definition of block characteristic polynomial. We use the term *regular* to identify matrix polynomials whose determinant is not identically zero over $\mathbb{C}$. The following results, including proofs of theorems, can be found in [93, Section 2.5] or in the more classical reference [58, Section 7.7].

The results extend the familiar concept of Euclidean division to matrix polynomials. Matrix polynomials form a non-commutative ring, so we need to differentiate between left and right divisors. However, the underlying idea of dividing $P(z)$ by $D(z)$ is still the same: we want to write $P(z)$ as a multiple of $D(z)$ plus an additional remainder term, which should be of lower degree than $D(z)$.

Definition 2.1.4. *Let $P(z), K(z), R(z)$ and $D(z)$ be matrix polynomials, where $P(z)$ has degree d, $D(z)$ is regular with degree less than d, and $R(z)$ has degree less than $\deg D(z)$. $K(z)$ is defined as "left quotient" and $R(z)$ as the "left remainder" of $P(z)$ divided by $D(z)$ if*

$$P(z) = D(z)K(z) + R(z).$$

If $R(z) = 0$, we say that $P(z)$ is left divisible by $D(z)$.

A natural question arises: given $P(z)$ and a lower degree polynomial $D(z)$, can we easily check if $D(z)$ divides $P(z)$ (i.e., if the remainder of the left or right division is zero)?

For a scalar polynomial $p(\lambda)$ and a linear divisor $\lambda - s$, this amounts to check if $p(s) = 0$. A similar result holds for matrix polynomials as well.

Theorem 2.1.6. *[93, Theorem 2.17] The matrix polynomial $P(z) \in \mathcal{P}(\mathbb{C}^{b\times b})$ is left divisible by $zI_b - S$, where $S \in \mathbb{C}^{b\times b}$ if and only if $P(S) = 0$.*

Definition 2.1.5. *Let $P(z)$ be a matrix polynomial. A matrix $S \in \mathbb{C}^{b\times b}$ is called a left solvent of $P(z)$ if $P(S) = 0$.*

In the following, we omit "left" when referring to quotients, divisibility and solvents.

We remark that solvents are important tools in the analysis of matrix polynomials. They can be used to compute a part of the spectrum [87], and are closely related to the solution of one-sided matrix equation that arises, for instance, in some Markov chains (see [20] and the references therein).

We now present a possible way to construct a monic block characteristic polynomial. In the scalar case, we may think of building the characteristic polynomial of a matrix A by computing its eigenvalues $s_1, \dots, s_n$, and then taking the product of the linear factors $p(\lambda) = (\lambda - s_1) \dots (\lambda - s_n)$. The next theorem presents the extension of this idea to the block case, where the eigenvalues are replaced by blocks in a block diagonal matrix similar to the original one, and solvents play the role of the roots.

Definition 2.1.6. *Let $A \in \mathbb{C}^{db\times db}$ and $C \in \mathbb{C}^{db\times b}$. A monic block characteristic polynomial of A with respect to C is a matrix polynomial $P(z) = \sum_{i=0}^{d} z^i \Gamma_i \in \mathcal{P}_d(\mathbb{C}^{b\times b})$, with $\Gamma_d = I_d$, such that*

$$P(A) \circ C = 0.$$

A sufficient condition for the existence of a monic block characteristic polynomial of $A \in \mathbb{C}^{db\times db}$ with respect to $C \in \mathbb{C}^{db\times b}$, is the invertibility of the matrix

$$[C, AC, \dots, A^{d-1}C].$$

For further details, refer to [93, Section 2.2.5]. For simplicity, throughout the thesis, we will assume this condition whenever monic block characteristic polynomials are employed.

Theorem 2.1.7. *[93, Theorem 2.24] Let $A \in \mathbb{C}^{db\times db}$ and $C \in \mathbb{C}^{db\times b}$. Let $P(z)$ be a monic block characteristic polynomial of A with respect to C. Assuming that there exists a block diagonal matrix*

$$T = \begin{bmatrix} \Theta_1 & & \\ & \ddots & \\ & & \Theta_d \end{bmatrix},$$

with $\{\Theta_i\}_{i=1:d} \subseteq \mathbb{C}^{b\times b}$ and an invertible matrix $U \in \mathbb{C}^{db\times db}$ such that

$$A = UTU^{-1},$$

and letting $[\Delta_1, \dots, \Delta_d]^T = U^{-1}C$, with $\{\Delta_i\}_{i=1}^d \subseteq \mathbb{C}^{b\times b}$, then if Δ_i is invertible for each i, it holds that

1. *$S_i = \Delta_i^{-1}\Theta_i\Delta_i$ are solvents of P(z);*
2. *if $S_i - S_j$ is nonsingular for each $i \neq j$ then*

$$P(z) = (zI_b - S_1)\cdot\cdots\cdot(zI_b - S_d).$$

2.2 Block rational Krylov subspaces

Given a matrix $A \in \mathbb{C}^{n\times n}$, a block vector $C \in \mathbb{C}^{n\times b}$ and a sequence of poles $\boldsymbol{\xi}_k = \{\xi_j\}_{j=0}^{k-1} \subseteq \overline{\mathbb{C}} \setminus \Lambda(A)$ the kth block rational Krylov subspace is defined as

$$\mathcal{Q}_k(A, C, \boldsymbol{\xi}_k) = \left\{ R(A)\circ C : R(z) = \frac{P(z)}{q_k(z)},\ \text{with } P(z) \in \mathcal{P}_{k-1}(\mathbb{C}^{b\times b}) \right\}, \tag{2.4}$$

where

$$q_k(z) = \prod_{\xi_j \in \boldsymbol{\xi}_k, \xi_j \neq \infty} (z - \xi_j). \tag{2.5}$$

For simplicity, we sometimes denote such space by $\mathcal{Q}_k(A, C)$ omitting poles. If all the poles are equal to infinity, (2.4) is usually referred as block *polynomial* Krylov subspace, and it is denoted by $\mathcal{K}_k(A, C)$.

The dimension of $\mathcal{Q}_k(A, v, \boldsymbol{\xi}_k)$ is defined as the rank of the matrix[1]

$$[(A-\xi_0)^{-1}C, (A-\xi_1)^{-1}(A-\xi_0)^{-1}AC, \dots, (A-\xi_{k-1})^{-1}\cdots(A-\xi_0)^{-1}A^{k-1}C]. \tag{2.6}$$

To simplify the notation we will always assume that (2.6) is full rank, that is, the dimension of $\mathcal{Q}_k(A, C)$ is equal to kb.

An orthonormal block basis of $\mathcal{Q}_k(A, C)$ (for simplicity, we will often just say "orthonormal basis") is defined as a matrix $V_k = [V^{(1)}, \dots, V^{(k)}]$ with orthonormal columns, where $V^{(1)}, \dots, V^{(k)} \in \mathbb{C}^{n\times b}$, such that every block vector $Z \in \mathcal{Q}_k(A, C)$ can be written as $Z = \sum_{i=1}^k V^{(i)}\Gamma_i$, for $\Gamma_i \in \mathbb{C}^{b\times b}$.

Key components in utilizing Krylov subspaces involve calculating a block orthonormal basis and performing the corresponding projection of A. If an orthonormal basis V_k is known, than the projected matrix is given by $A_k = V_k^H A V_k$.

[1] Where the matrix $(A-\xi)^{-1}$ is replaced by the identity matrix if $\xi = \infty$.

The matrix V_k can be computed by a block rational Arnoldi Algorithm[2] 1, that iteratively computes the block columns of V_k and two matrices $\underline{K}_{k-1}, \underline{H}_{k-1} \in \mathbb{C}^{bk \times b(k-1)}$ in block upper Hessenberg form such that

$$AV_k\underline{K_{k-1}} = V_k\underline{H_{k-1}}. \tag{2.7}$$

We refer to the portion of Algorithm 1 enclosed between lines 4 and 10 as *Arnoldi iteration*.

Algorithm 1 Block Rational Arnoldi

Input: $A \in \mathbb{C}^{n\times n}, C \in \mathbb{C}^{n\times b}, \boldsymbol{\xi}_k = \{\xi_0, \dots, \xi_{k-1}\}$
Output: $V_k \in \mathbb{C}^{n\times bk}, \underline{H}_{k-1}, \underline{K}_{k-1} \in \mathbb{C}^{bk\times b(k-1)}$

1: $Z \leftarrow (I - A/\xi_0)^{-1}C$ ▷ with the convention $A/\infty = 0$
2: $[V^{(1)}, \sim] \leftarrow \mathsf{qr}(Z)$ ▷ compute a thin QR decomposition
3: **for** $j = 1, \dots, k-1$ **do**
4: Compute $Z = (I - A/\xi_j)^{-1}AV^{(j)}$
5: **for** $i = 1, \dots, j$ **do**
6: $(\underline{H}_{k-1})_{\mathbf{i},\mathbf{j}} \leftarrow (V^{(i)})^H Z$ ▷ where $\mathbf{i}$ and $\mathbf{j}$ are block indices
7: $Z \leftarrow Z - V^{(i)}(\underline{H}_{k-1})_{\mathbf{i},\mathbf{j}}$
8: **end for**
9: $[V^{(j+1)}, (\underline{H}_{k-1})_{\mathbf{j+1},\mathbf{j}}] \leftarrow \mathsf{qr}(Z)$ ▷ compute a thin QR decomposition
10: $(\underline{K}_{k-1})_{\mathbf{i},1:(j+1)b} \leftarrow (\underline{H}_{k-1})_{\mathbf{i},1:(j+1)b}/\xi_j - E_j,$ ▷ where $E_j = [0, \dots, 0, 1, 0]^T \otimes I_b$
11: **end for**
12: $V_k \leftarrow [V^{(1)}, \dots, V^{(k)}]$

Relation (2.7) completely determines the rational Krylov subspace, and encodes all the information regarding poles and column span of the starting block vector. The following definition given by Elsworth and Güttel in [48] generalizes relation (2.7).

Definition 2.2.1 ([48]). *Let $A \in \mathbb{C}^{n\times n}$. A relation of the form*

$$AV_k\underline{K_{k-1}} = V_k\underline{H_{k-1}}$$

is called orthonormal block rational Arnoldi decomposition (BRAD), if the following conditions are satisfied[3]*:*

1. *$V_k \in \mathbb{C}^{n\times bk}$ has orthonormal columns;*
2. *$\underline{K_{k-1}}$ and $\underline{H_{k-1}}$ are $bk \times b(k-1)$ block upper Hessenberg matrices such that for each i either $(\underline{K}_{k-1})_{\mathbf{i+1},\mathbf{i}}$ or $(\underline{H}_{k-1})_{\mathbf{i+1},\mathbf{i}}$ (or both) are invertible;*
3. *for any i, there exist two scalars $\mu_i, \nu_i \in \mathbb{C}$, with at least one different from zero, such that*

$$\mu_i(\underline{K}_{k-1})_{\mathbf{i+1},\mathbf{i}} = \nu_i(\underline{H}_{k-1})_{\mathbf{i+1},\mathbf{i}};$$

4. *the numbers $\xi_i = \mu_i/\nu_i$ above, called poles of the BRAD, are outside the spectrum of A.*

Remark 2.2.1. *The relation* (2.7) *produced by the block rational Arnoldi algorithm is a block rational Arnoldi decomposition, see [48, Section 2].*

Remark 2.2.2. *The matrices $\underline{H}_{k-1}$ and $\underline{K}_{k-1}$ of a block rational Arnoldi decomposition are both full rank. This follows from [48, Lemma 3.2].*

The following theorem relates rational Arnoldi decompositions with rational Krylov subspaces.

[2] For simplicity we describe a version of the algorithm that does not allow poles equal to zero. For a more complete version of the algorithm, we refer to [48].

[3] The bold subscripts denote block indices. For instance $\mathbf{i} = [(i-1)b + 1, \dots, ib]$.

Theorem 2.2.1 ([48]). *Let $A \in \mathbb{C}^{n\times n}$, $C \in \mathbb{C}^{n\times b}$, $\boldsymbol{\xi}_k = \{\xi_0, \dots \xi_{k-1}\}$ and let $\mathcal{Q}_k(A, C)$ be the block rational Krylov subspace with poles $\boldsymbol{\xi}_k$. Let*

$$AV_k\underline{K_{k-1}} = V_k\underline{H_{k-1}}$$

be a BRAD with poles $\{\xi_1, \dots \xi_{k-1}\}$, such that the first block column of V_k is an orthonormal basis of the space spanned by the columns of $(I - A/\xi_0)^{-1}C$. Then V_k is an orthonormal block basis of $\mathcal{Q}_k(A, C)$. Moreover, the matrix obtained by taking the first j block columns of V_k is an orthonormal block basis for $\mathcal{Q}_j(A, C, \{\xi_i\}_{i=0}^{j-1})$ for each $j \leq k$.

For the proof of the theorem and a more detailed description of block rational Arnoldi decompositions we refer to [48].

Let V_{k-1} be the matrix obtained by taking the first $b(k-1)$ columns of V_k. The computation of the projected matrix $A_{k-1} = V_{k-1}^H A V_{k-1}$ by using the formula is usually expensive if the dimension of the matrix A is large. For the case of Hermitian A, several methods that exploit the structure of A_{k-1} have been developed to avoid expensive operations for the computation, see for instance [27, 102]. In the non-Hermitian case, it is more difficult to exploit a structure of A_{k-1}. However, if the last pole of the associated BRAD is equal to infinity the projected matrix can be easily computed as $A_{k-1} = H_{k-1}K_{k-1}^{-1}$, where K_{k-1} and H_{k-1} are the head $(k-1)b \times (k-1)b$ principal submatrix of $\underline{K}_{k-1}$ and $\underline{H}_{k-1}$ respectively. To prove this, notice that if the last pole is equal to infinity then the last block row of $\underline{K}_{k-1}$ has to be zero, then since $\underline{K}_{k-1}$ is full rank, K_{k-1} is invertible, hence multiplying both the terms of the block rational Arnoldi decomposition (2.7) on the left by V_{k-1}^H and on the right by K_{k-1}^{-1} we obtain $A_{k-1} = H_{k-1}K_{k-1}^{-1}$.

A technique that is often used to compute A_k is to add a pole equal to infinity every time we want to compute a new projected matrix. However, this would significantly increase the size of the block rational Krylov subspace in an artificial manner. In the next section, we describe a way to ensure that the last pole is always equal to infinity, avoiding these additional steps.

2.2.1 Reordering poles

Let us assume to know a block rational Arnoldi decomposition

$$A\widehat{V}_{k+1}\underline{\widehat{K}_k} = \widehat{V}_{k+1}\underline{\widehat{H}_k} \tag{2.8}$$

with poles $\{\xi_1, \dots, \xi_{k-2}, \infty, \xi_{k-1}\}$. In the following, we introduce a practical way to swap the last two poles by using unitary transformations producing a block rational Arnoldi decomposition associated with the sequence of poles $\{\xi_1, \dots, \xi_{k-2}, \xi_{k-1}, \infty\}$. This technique has been already described for the non-block case in [66]. By Definition 2.2.1, since the second last pole is equal to infinity, the submatrix $(\underline{\widehat{K}_k})_{\mathbf{k},\mathbf{k-1}}$ is equal to zero. Moreover, to produce a new block rational Arnoldi decomposition that has the last pole equal to infinity it is sufficient to annihilate the submatrix $(\underline{\widehat{K}_k})_{\mathbf{k+1},\mathbf{k}}$, keeping the block Hessenberg structure of the two matrices. This can be done by employing unitary transformations. Let

$$Q_1R_1 = \begin{bmatrix} (\underline{\widehat{K}_k})_{\mathbf{k},\mathbf{k}} \\ (\underline{\widehat{K}_k})_{\mathbf{k+1},\mathbf{k}} \end{bmatrix}$$

be a thin QR decomposition and let R_2Q_2 be an RQ decomposition[4] for the last block row of

$$Q_1^H \begin{bmatrix} (\underline{\widehat{H}_k})_{\mathbf{k},\mathbf{k-1}} & (\underline{\widehat{H}_k})_{\mathbf{k},\mathbf{k}} \\ 0 & (\underline{\widehat{H}_k})_{\mathbf{k+1},\mathbf{k}} \end{bmatrix}.$$

Then, the matrices

$$Q_1^H \begin{bmatrix} 0 & (\underline{\widehat{K}_k})_{\mathbf{k},\mathbf{k}} \\ 0 & (\underline{\widehat{K}_k})_{\mathbf{k+1},\mathbf{k}} \end{bmatrix} Q_2^H \quad \text{and} \quad Q_1^H \begin{bmatrix} (\underline{\widehat{H}_k})_{\mathbf{k},\mathbf{k-1}} & (\underline{\widehat{H}_k})_{\mathbf{k},\mathbf{k}} \\ 0 & (\underline{\widehat{H}_k})_{\mathbf{k+1},\mathbf{k}} \end{bmatrix} Q_2^H$$

[4] An RQ decomposition consists of writing a matrix as the product of an upper triangular matrix times a unitary matrix. It can be computed by using the same techniques involved in the computation of a QR decomposition.

are block upper triangular and the last block row of the first one is equal to zero.

If we let

$$\begin{aligned} V_{k+1} &= \widehat{V}_{k+1}(I_{b(k-1)} \oplus Q_1), \\ \underline{K_k} &= (I_{b(k-1)} \oplus Q_1^H)\underline{\widehat{K}_k}(I_{b(k-2)} \oplus Q_2^H) \\ \underline{H_k} &= (I_{b(k-1)} \oplus Q_1^H)\underline{\widehat{H}_k}(I_{b(k-2)} \otimes Q_2^H), \end{aligned}$$

where $\oplus$ denotes the Kronecker sum, the relation

$$AV_{k+1}\underline{K_k} = V_{k+1}\underline{H_k}$$

is a new block rational Arnoldi decomposition with poles $\{\xi_1, \dots, \xi_{k-2}, \xi_{k-1}, \infty\}$.

The procedure described above can be used to efficiently compute the projected matrix during the block rational Arnoldi algorithm. It is sufficient to start by performing the block rational Arnoldi algorithm (Algorithm 1) using poles $\{\xi_0, \infty\}$ to compute the matrices $V_2, \underline{K}_1, \underline{H}_1$ that form a block rational Arnoldi decomposition. Then, by performing a new Arnoldi iteration, the pole ξ_1 can be added, and subsequently the produced block rational Arnoldi decomposition can be transformed into a new one $V_3, \underline{K}_2, \underline{H}_2$ by swapping the last two poles. In particular, since the last pole is equal to infinity, the projected matrix satisfies $A_2 = H_2K_2^{-1}$. This strategy can be iterated to enlarge the Krylov subspace by adding all the remaining poles.

Observe that the computational cost of the procedure to swap the poles is independent of the size of A, therefore it is usually negligible compared to the computational cost of an iteration of the block rational Arnoldi algorithm (Algorithm 1).

Remark 2.2.3. *When we transform the matrix $\widehat{V}_k$ in V_k we only perform a linear combination between the last two block columns. For this reason the top-left principal $b(k-1) \times b(k-1)$ submatrix of A_k is equal to A_{k-1} . Hence, to compute A_k it is sufficient to determine its last block row and column. This can be done using the relation $A_k = H_kK_k^{-1}$ and so*

$$A_kE_k = H_kK_k^{-1}E_k \quad \text{and} \quad E_k^TA_k = E_k^TH_kK_k^{-1},$$

with $E_k = \mathbf{e}_k \otimes I_b$, where $\mathbf{e}_k \in \mathbb{C}^k$ is the last vector of the canonical basis.

2.3 Properties of block rational Krylov subspaces

One of the key characteristics of rational Krylov subspaces lies in the "exactness" on rational functions (see [66, Lemma 4.6]). We extend this property to the block rational Krylov framework.

Proposition 2.3.1 (Block exactness)**.** *Let $A \in \mathbb{C}^{n\times n}$, $C \in \mathbb{C}^{n\times b}$ and let $\boldsymbol{\xi}_k$ be a set of poles. Denoting by V_k an orthonormal basis of $\mathcal{Q}_k(A, C, \boldsymbol{\xi}_k)$, for any matrix U with orthonormal columns such that $\operatorname{span}(V_k) \subseteq \operatorname{span}(U)$, we have*

$$UU^HR(A) \circ C = UR(U^HAU) \circ U^HV,$$

for any $R(z) \in \mathcal{P}_k(\mathbb{C}^{b\times b})/q(z)$ with $q(z) = \prod_{\xi\in\boldsymbol{\xi}_k, \xi\neq\infty}(z-\xi)$. In particular, if $R(z) \in \mathcal{P}_{k-1}(\mathbb{C}^{b\times b})/q(z)$ it holds

$$R(A) \circ C = UR(U^HAU) \circ U^HC.$$

As a consequence, we have

$$U\mathcal{Q}_k(U^HAU, U^HC, \boldsymbol{\xi}_k) = \mathcal{Q}_k(A, C, \boldsymbol{\xi}_k).$$

Proof. The proof is composed of two parts. First, we suppose that the poles are all equal to infinity, i.e., $q(z) = 1$. Then, we extend the proof for a generic choice of poles.

For the first part, by linearity, it is sufficient to prove the equalities for $R(z) = z^j I_b$ with $j \leq h$. We proceed by induction on j. If $j = 0$ there is nothing to prove. For $R(z) = z^{j+1} I_b$, by the inductive hypothesis we have

$$UU^H A^{j+1} C = UU^H AA^j C = UU^H AU(U^H AU)^j U^H C = U(U^H AU)^{j+1} U^H C.$$

Moreover, if $j + 1 \leq k - 1$, $A^{j+1} C \in \mathcal{K}_k(A, C)$, hence $UU^H A^{j+1} C = A^{j+1} C$.

Let now $R(z) = P(z)/q(z)$ with $P(z) \in \mathcal{P}(\mathbb{C}^{b\times b})$. Using the commutativity property of Lemma 2.1.3, we have that

$$R(A) \circ C = q(A)^{-1} P(A) \circ C = P(A) \circ (q(A)^{-1} C).$$

Hence, if we let $\widetilde{C} = q(A)^{-1} C$, from the result of the first step we have

$$UU^H R(A) \circ C = UU^H P(A) \circ \widetilde{C} = UP(U^H AU) \circ (U^H \widetilde{C}),$$

and, if $R(A) \in \mathcal{P}_{k-1}(\mathbb{C}^{b\times b})/q(A)$, we have

$$R(A) \circ C = P(A) \circ \widetilde{C} = UP(U^H AU) \circ (U^H \widetilde{C}).$$

To conclude it is sufficient to prove that $U^H \widetilde{C} = q(U^H AU)^{-1} U^H C$. Since $C = q(A)\widetilde{C}$, or analogously $C = Q(A) \circ \widetilde{C}$ where $Q(z) = q(z) I_b$, by the first step of the proof we have

$$UU^H C = UU^H Q(A) \circ \widetilde{C} = UQ(U^H AU) \circ (U^H \widetilde{C}) = Uq(U^H AU) U^H \widetilde{C}.$$

Since $U^H U = I$, multiplying both sides on the left by $q(U^H AU)^{-1} U^H$ we get

$$q(U^H AU)^{-1} U^H C = U^H \widetilde{C},$$

that concludes the proof. □

Corollary 2.3.2. *Let $\chi(z) \in \mathcal{P}_k(\mathbb{C}^{b\times b})$ be a monic block characteristic polynomial for $V_k^H A V_k$ with respect to $V_k^H C$. Denoting by $R^G(z) = \chi(z)/q(z)$, it holds*

$$V_k^H R^G(A) \circ C = 0,$$

moreover $R^G(A) \circ C$ minimizes $\|R(A) \circ C\|_F$ over all the $R(z) \in \mathcal{P}_k(\mathbb{C}^{b\times b})/q(z)$ such that $R(z) = P(z)/q(z)$ where $P(z)$ is a monic matrix polynomial of degree k.

Proof. By Proposition 2.3.1 it holds

$$V_k V_k^H R^G(A) \circ C = V_k R^G(V_k^H A V_k) \circ V_k^H C = 0.$$

Since $V_k^H V_k = I_{bk}$, multiplying on the left by V_k^H we obtain the first equivalence.

The problem of minimizing $\|R(A) \circ C\|_F$ over all the $R(z) \in \mathcal{P}_k(\mathbb{C}^{b\times b})/q(z)$ with monic numerator can be rewritten as

$$\begin{aligned}
&\min_{\widehat{R}(z)\in\mathcal{P}_{k-1}(z)/q(z)} \left\| q(A)^{-1} A^k C - \widehat{R}(z) \circ C \right\|_F \\
&= \min_{Y\in\mathbb{C}^{bk\times b}} \left\| q(A)^{-1} A^k C - V_k Y \right\|_F \\
&= \min_{Y\in\mathbb{C}^{bk\times b}} \left\| (I_b \otimes q(A)^{-1} A^k)\mathrm{vec}(C) - (I_b \otimes V_k)\mathrm{vec}(Y) \right\|_2 .
\end{aligned}$$

The solution of the least square problem is given by the matrix Y such that

$$(I_b \otimes V_k)^H \left((I_b \otimes q(A)^{-1} A^k)\mathrm{vec}(C) - (I_b \otimes V_k)\mathrm{vec}(Y) \right) = 0,$$

that is analogue to ask that $V_k^H (q(A)^{-1} A^k C - V_k Y) = 0$, that is, the solution of the minimization problem sathisfies $V_k^H (R(A) \circ C) = 0$, hence the function $R^G(z)$ is the solution. □

When working on developing algorithms for computing matrix functions, we frequently encounter Krylov subspaces that involve block diagonal matrices. Proposition 2.3.3 is a fundamental yet straightforward result essential for our purposes.

Proposition 2.3.3. *Let $\mathbf{A} := \operatorname{blkdiag}(A_i)$ and $\mathbf{C} := \operatorname{blkdiag}(C_i)$ with $A_i \in \mathbb{C}^{n_i \times n_i}$ and $C_i \in \mathbb{C}^{n_i \times b_i}$ and let $\boldsymbol{\xi}_k$ be a sequence of poles. Then an orthonormal basis of $\mathcal{Q}_k(\mathbf{A}, \mathbf{C}, \boldsymbol{\xi}_k)$ is given by $\mathbf{V} := \operatorname{blkdiag}(V_i)$, where V_i is an orthonormal basis of $\mathcal{Q}_k(A_i, C_i, \boldsymbol{\xi}_k)$ for each i.*

Proof. By definition, every element of $\mathcal{Q}_k(\mathbf{A}, \mathbf{C}, \boldsymbol{\xi}_k)$ takes the form

$$q_k(\mathbf{A})^{-1} \sum_{j=0}^{k-1} \mathbf{A}^j \mathbf{C} \Gamma_j = \sum_{j=0}^{k-1} \operatorname{blkdiag}\big(q_k(A_i)^{-1} A_i^j C_i\big) \Gamma_j,$$

with q_k defined in (2.5). Because $q_k(A_i)^{-1} A_i^j C_i \in \mathcal{Q}_k(A_i, C_i, \boldsymbol{\xi}_k)$ for every i, j, we have $\mathcal{Q}_k(\mathbf{A}, \mathbf{C}, \boldsymbol{\xi}_k) = \operatorname{span}(\operatorname{blkdiag}(V_i))$. □

2.4 Low-rank updates of Hermitian matrix functions

The following result from [10] shows how to approximate a low-rank update of a Hermitian matrix function, using block rational Krylov subspaces.

Theorem 2.4.1 ([10, Theorem 4.5]). *Let $A = B + C\Delta C^H$, where $B \in \mathbb{C}^{n \times n}$, $\Delta \in \mathbb{C}^{b \times b}$ are Hermitian and $C \in \mathbb{C}^{n \times b}$. Let $[a, b]$ be an interval that contains the spectra of A and Δ, and let $\boldsymbol{\xi}_k = \{\xi_0, \dots, \xi_{k-1}\} \subseteq \overline{\mathbb{C}}$, be a sequence of poles closed under complex conjugation. For a function f analytic on $[a, b]$, we consider the approximation*

$$f(A) \approx \widehat{F} := f(B) + V_k\big(f(V_k^H A V_k) - f(V_k^H B V_k)\big) V_k^H, \tag{2.9}$$

where V_k is an orthonormal basis of $\mathcal{Q}_k(B, C, \boldsymbol{\xi}_k)$. Then the approximation error $E(f) := \widehat{F} - f(A)$ satisfies

$$\|E(f)\|_2 \le 4 \min_{r \in \mathcal{P}_{k-1}/q_k} \|f - r\|_\infty ,$$

where $q_k(z) = \prod_{\xi \in \boldsymbol{\xi}_k, \xi \neq \infty} (z - \xi)$ and $\|\cdot\|_\infty$ denotes the supremum norm on $[a, b]$. In particular, the approximation (2.9) is exact if $f \in \mathcal{P}_{k-1}/q_k$.

Theorem 2.4.1 can be extended to the case where V_k is replaced by a matrix U such that $\operatorname{span}(U) \supseteq \mathcal{Q}_k(B, C, \boldsymbol{\xi}_k)$ as shown in Proposition 2.4.2.

Proposition 2.4.2. *Using the notation of Theorem 2.4.1, we have*

$$f(A) \approx \widehat{F} := f(B) + U\big(f(U^H A U) - f(U^H B U)\big) U^H, \tag{2.10}$$

where U is matrix with orthonormal columns such that $\operatorname{span}(U) \supseteq \mathcal{Q}_k(B, C, \boldsymbol{\xi}_k)$. Then the approximation error $E(f) := \widehat{F} - f(A)$ satisfies

$$\|E(f)\|_2 \le 4 \min_{r \in \mathcal{P}_{k-1}/q_k} \|f - r\|_\infty .$$

Proof. If we assume that the approximation (2.10) is exact when $f \in \mathcal{P}_{k-1}/q_k$, then we can prove the statement using the same proof as [10, Theorem 4.5]. So, we just need to show that if $f \in \mathcal{P}_{k-1}/q_k$, then the approximation (2.10) is exact.

First of all, we note that, since $\operatorname{span}(U) \supseteq \mathcal{Q}_k(B, C, \boldsymbol{\xi}_k)$ we have $UU^H V_k = V_k$, where V_k is an orthonormal basis of $\mathcal{Q}_k(B, C, \boldsymbol{\xi}_k)$. Moreover, by Theorem 2.4.1 we have

$$r(B + C\Delta C^H) - r(B) = V_k Y_k(r) V_k^H = UU^H V_k Y_k(r) V_k^H UU^H,$$

with

$$Y_k(r) := r(V_k^H A V_k) - r(V_k^H B V_k).$$

To conclude it is sufficient to prove that

$$U^H V_k Y_k(r) V_k^H U = r(U^H A U) - r(U^H B U).$$

It follows from Proposition 2.3.1 that $U^H V_k$ is an orthonormal basis of the subspace

$$\mathcal{Q}_k(U^H A U, U^H C, \boldsymbol{\xi}_k) = U^H \mathcal{Q}_k(A, C, \boldsymbol{\xi}_k),$$

and by Theorem 2.4.1 we obtain

$$\begin{aligned} & r(U^H A U) - r(U^H B U) \\ & \quad = U^H V_k \left(r(V_k^H U U^H A U U^H V_k) - r(V_k^H U U^H B U U^H V_k) \right) V_k^H U \\ & \quad = U^H V_k (r(V_k^H A V_k) - r(V_k^H B V_k)) V_k^H U \\ & \quad = U^H V_k Y_k(r) V_k^H U. \end{aligned}$$

□

Chapter 3

Sylvester equations

The contents of this chapter result from joint work with Leonardo Robol, as detailed in [26].

We are concerned with the solution of Sylvester equations of the form

$$AX - XB = C_1 C_2^H, \quad \text{where} \quad C_1 \in \mathbb{C}^{n\times b},\ C_2 \in \mathbb{C}^{m\times b}, \tag{3.1}$$

and A, B are square matrices of sizes $n \times n$ and $m \times m$, respectively. The matrices C_1, C_2 are block vectors with a few b columns, with $b \ll n, m$. If A and B have disjoint spectra, the solution is unique and can be expressed in the integral form

$$X = \frac{1}{2\pi i} \int_\gamma (zI_n - A)^{-1} C_1 C_2^H (zI_m - B)^{-1} dz \tag{3.2}$$

where γ is a compact contour that encloses once, in positive orientation, the eigenvalues of A, but not the eigenvalues of B [86].

Sylvester equations arise often in control theory [3, 14], and in the solution of 2D PDEs on tensorized domains [103, 124]. In this setting, the matrices involved are often of large size, and exploiting the low-rank structure in the right-hand side is essential. For problems arising from control theory, the rank is linked with the number of inputs and outputs in the system, so b is typically moderate and related to the analysis of MIMO systems [3]. For PDEs, the low-rank property holds in an approximate sense and is related to the regularity of the problem under consideration.

When the spectra of A and B are well-separated, one can show that the matrix X that solves (3.1) has exponentially decaying singular values [13], and can be approximated as a low-rank matrix [121]. If X is close to a low-rank matrix, i.e., we can write it as $X = UYV^H + E$ where U, V are matrices with a few orthogonal columns, and E is a small error, then the Sylvester equation can be approximately solved by computing the exact solution of the projected equation $(U^H AU)Y - Y(V^H BV) = U^H C_1 C_2^H V$. This is the core idea of projection methods. The main difficulty lies in identifying good bases U and V to use for projecting the equation.

A common choice is to take U and V as orthonormal basis of polynomial or rational Krylov subspaces; progressively increasing the dimensions of the subspaces produce a sequence of approximate solutions.

When C_1, C_2 are vectors, the characterization of Krylov subspaces through polynomials or rational functions allow us to link the convergence of the method with a polynomial (resp. rational) approximation problem, which allows us to state explicit results (at least in the case of normal matrix coefficients) [9]. The rational methods are inherently more complex to analyze because a choice of poles is involved, and the convergence is dependent on the quality of these poles.

When C_1 and C_2 are block vectors an analogous construction can be made. The results in the literature focus mostly on the non-block case and are more scarce for this setting. One of the contributions of this chapter is to extend the convergence analysis for rational Krylov found in [9] to this more general setting. This is done by exploiting the notation for rational matrices introduced in Section 2.1.

If $X = UYV^H$ with U, V bases of a Krylov subspace of order ℓ, then the residual $AX - XB - C$ belongs to the Krylov subspace of order $\ell + 1$ [121]. This property can be exploited to compute the

residual error almost for free at each step. For rational Krylov subspaces, the analogous result tells us that the residual belongs to a larger subspace obtained by adding an infinity pole. However, if infinity poles are periodically injected into the space for residual computation rather than necessity, we risk artificially inflating the size of the projected problem. In this chapter, we show how one can exploit the techniques for pole reordering described in Section 2.2.1 to maintain a single infinity pole in the definition of the block rational Krylov subspace, precisely with the aim of checking the residual.

Then, the convergence analysis, which extends the results provided by Beckermann in [9], is utilized to design an adaptive-pole-selection algorithm. Since the objective function is now matrix-valued, instead of scalar, the problem is much richer. In particular, the minimization of its norm is numerically challenging, and it is natural to replace the objective function with a simpler surrogate. We present various options, and we show that one of these leads to the same heuristic proposed by Druskin and Simoncini in [42] generalizing the rank 1 case. Hence, our theory provides a theoretical analysis of the convergence of this choice. Then, we show that other choices for the surrogate function are possible; in particular, we provide an adaptive technique of pole selection that slightly improves the one proposed in [42].

3.1 Rational Krylov for Sylvester equations

Krylov subspace methods are one of the most popular methods for solving the Sylvester equation (3.1) where $A \in \mathbb{C}^{n\times n}, B \in \mathbb{C}^{m\times m}$ are large size matrices and $C_1 \in \mathbb{C}^{n\times b}, C_2 \in \mathbb{C}^{m\times b}$ are tall and skinny. In such a case, the solution can be approximated by a low-rank matrix to avoid storing the complete solution which is prohibitive for large n and m. We refer to [121, Section 4.4] for a more complete discussion about the topic.

The technique described in Section 2.2.1 can be employed in the context of solving Sylvester equations: let U_{h+1} and V_{k+1} be orthonormal block basis for $\mathcal{Q}_{h+1}(A, C_1)$ and $\mathcal{Q}_{k+1}(B^H, C_2)$ respectively, generated by the block rational Arnoldi algorithm 1 and let $U_h \in C^{n\times bh}$ and $V_k \in \mathbb{C}^{m\times bk}$ be the matrices obtained removing from U_{h+1} and V_{k+1} the last b columns. Letting $A_h = U_h^H A U_h$ and $B_k = V_k^H B V_k$, the solution X can be approximated by $X_{h,k} = U_h Y_{h,k} V_k^H$, where $Y_{h,k}$ solves the projected equation

$$A_h Y_{h,k} - Y_{h,k} B_k = U_h^H C_1 (V_k^H C_2)^H. \tag{3.3}$$

For simplicity of notation in the rest of the section, we assume that $\xi_0 = \infty$, that is,

$$U_h^H C_1 = E_1 \Theta_1 \quad \text{and} \quad V_k^H C_2 = E_1 \Theta_2,$$

where $\Theta_1, \Theta_2 \in \mathbb{C}^{b\times b}$ and E_1 is the block vector of appropriate dimensions, containing a $b \times b$ identity matrix in its first block and zeros elsewhere.

If U_{h+1} and V_{k+1} are determined as described in Section 2.2.1, the projected matrices A_h and B_k can be easily computed at each step. In the following, we show that this choice of poles also allows a cheap computation of the norm of the residual matrix

$$\mathtt{R}_{h,k} = A X_{h,k} - X_{h,k} B - C_1 C_2^H.$$

Since the last pole used to generate $\mathcal{Q}_{h+1}(A, C_1)$ is always equal to infinity, the columns of AU_h belongs to $\mathcal{Q}_{h+1}(A, C_1)$, that is,

$$U_{h+1} U_{h+1}^H A U_h = A U_h.$$

In the same way it holds

$$V_k^H B V_{k+1} V_{k+1}^H = V_k^H B.$$

Using the last two relations, the definition of $X_{h,k}$ and that the first block columns of U_{h+1} and V_{k+1} are

given by the orthonormalization of C_1 and C_2 respectively, we can rewrite the residual as

$$\begin{aligned}
\mathtt{R}_{h,k} &= U_{h+1}U_{h+1}^H A U_h Y_{h,k} V_k^H - U_h Y_{h,k} V_k^H B V_{k+1} V_{k+1}^H - U_{h+1} E_1 \Theta_1 \Theta_2^H E_1^T V_{k+1}^H \\
&= U_{h+1} \left(U_{h+1}^H A U_h Y_{h,k} \begin{bmatrix} I_{bh} & 0 \end{bmatrix} - \begin{bmatrix} I_{bk} \\ 0 \end{bmatrix} Y_{h,k} V_k^H B V_{k+1} - E_1 \Theta_1 \Theta_2^H E_1^T \right) V_{k+1}^H \\
&= U_{h+1} \begin{bmatrix} U_h^H A U_h Y_{h,k} - E_1 \Theta_1 \Theta_2^H E_1^T & -Y_{h,k} V_k^H B V^{(k+1)} \\ (U^{(h+1)})^H A U_h Y_{h,k} & 0 \end{bmatrix} V_{k+1}^H \\
&= U_{h+1} \begin{bmatrix} A_h Y_{h,k} - Y_{h,k} B_k - E_1 \Theta_1 \Theta_2^H E_1^T & -Y_{h,k} V_k^H B V^{(k+1)} \\ (U^{(h+1)})^H A U_h Y_{h,k} & 0 \end{bmatrix} V_{k+1}^H \\
&= U_{h+1} \begin{bmatrix} 0 & -Y_{h,k} V_k^H B V^{(k+1)} \\ (U^{(h+1)})^H A U_h Y_{h,k} & 0 \end{bmatrix} V_{k+1}^H
\end{aligned}$$

where $U^{(h+1)}$ and $V^{(k+1)}$ are the last block columns of U_{h+1} and V_{k+1} respectively, and the zero matrix in the top left corner of the block matrix in the last row is given by equation (3.3).

Since the columns of U_{h+1} and V_{k+1} are orthonormal, the norm of the residual is equal to the norm of the block matrix

$$\begin{bmatrix} 0 & -Y_{h,k} V_k^H B V^{(k+1)} \\ (U^{(h+1)})^H A U_h Y_{h,k} & 0 \end{bmatrix}. \tag{3.4}$$

Let us now consider the block rational Arnoldi decomposition

$$A U_{h+1} \underline{K_h}^{(A)} = U_{h+1} \underline{H_h}^{(A)}.$$

Multiplying both the terms of the equations on the right by $\left(K_h{}^{(A)} \right)^{-1}$, where $K_h{}^{(A)}$ is the $bh \times bh$ head principal submatrix of $\underline{K_h}^{(A)}$, noting that the last block row of $\underline{K_h}^{(A)}$ is equal to zero, we have

$$A U_h = U_{h+1} \underline{H_h}^{(A)} \left(K_h{}^{(A)} \right)^{-1}. \tag{3.5}$$

Analogously, if

$$B^H V_{k+1} \underline{K_k}^{(B)} = V_{k+1} \underline{H_k}^{(B)}$$

is a block rational Arnoldi decomposition, we have that

$$B^H V_k = V_{k+1} \underline{H_k}^{(B)} \left(K_k{}^{(B)} \right)^{-1}, \tag{3.6}$$

where $K_k^{(B)}$ is the head $bk \times bk$ principal submatrix of $\underline{K_k}^{(B)}$.

Using the equations (3.5) and (3.6), we can rewrite the matrix (3.4) as

$$\begin{bmatrix} 0 & -Y_{h,k} \left(K_k^{(B)} \right)^{-H} \left(\underline{H_k}^{(B)} \right)^H V_{k+1}^H V^{(k+1)} \\ (U^{(h+1)})^H U_{h+1} \underline{H_h}^{(A)} \left(K_h^{(A)} \right)^{-1} Y_{h,k} & 0 \end{bmatrix}, \tag{3.7}$$

exploiting the orthogonality of the columns of U_{k+1} and V_{k+1}, the matrix (3.7) is equal to

$$\begin{bmatrix} 0 & Y_{h,k} \left(K_k^{(B)} \right)^{-H} \left(\underline{H_k}^{(B)} \right)^H E_{k+1}^H \\ E_{h+1} \underline{H_h}^{(A)} \left(K_h^{(A)} \right)^{-1} Y_{h,k} & 0 \end{bmatrix},$$

where $E_{h+1} \in \mathbb{C}^{b(h+1) \times b}$ and $E_{k+1} \mathbb{C}^{b(k+1) \times b}$ consist of the $b \times b$ identity in the last block row and zero elsewhere. Therefore, the norm of 3.7 can be recovered by the norms of the block vectors

$$E_{h+1} \underline{H_h}^{(A)} \left(K_h^{(A)} \right)^{-1} Y_{h,k} \quad \text{and} \quad Y_{h,k} \left(K_k^{(B)} \right)^{-H} \left(\underline{H_k}^{(B)} \right)^H E_{k+1}^H.$$

In particular the computation of the norm of the residual does not involve the matrices A and B, hence it can be performed with a computational cost that does not depend on n and m.

3.2 Residual and pole selection

This section aims to prove the following theorem[1]:

Theorem 3.2.1. *Let $A \in \mathbb{C}^{n\times n}$, $B \in \mathbb{C}^{m\times m}$, be matrices with disjoint spectra and let $C_1 \in \mathbb{C}^{n\times b}$ and $C_2 \in \mathbb{C}^{m\times b}$. Let $U \in \mathbb{C}^{n\times bh}$ and $V \in \mathbb{C}^{m\times bk}$ be orthonormal block basis for $\mathcal{Q}_h(A, C_1, \boldsymbol{\xi}_h^{(A)})$ and $\mathcal{Q}_k(B^H, C_2, \boldsymbol{\xi}_k^{(B)})$, respectively, and let $A_h = U^H AU$, $B_k = V^H BV$. Assuming that A_h and B_k have disjoint spectra, let $X_{h,k} = UY_{h,k}V^H$ where $Y_{h,k}$ is the solution of the Sylvester equation*

$$A_h Y_{h,k} - Y_{h,k} B_k = C_1^{(h)} (C_2^{(k)})^H,$$

with $C_1^{(h)} = U^H C_1$, and $C_2^{(k)} = V^H C_2$. Let $\chi_A(z) \in \mathcal{P}_h(\mathbb{C}^{b\times b})$ and $\chi_B(z) \in \mathcal{P}_k(\mathbb{C}^{b\times b})$, be monic block characteristic polynomials of A_h with respect to $C_1^{(h)}$ and B_k with respect to $C_2^{(k)}$, respectively. Define

$$R_A^G(z) = \frac{\chi_A(z)}{q_A(z)} \quad \text{and} \quad R_B^G(z) = \frac{\chi_B(z)}{q_B(z)},$$

where

$$q_A(z) = \prod_{\xi \in \boldsymbol{\xi}_h^{(A)}, \xi \neq \infty} (z - \xi) \quad \text{and} \quad q_B(z) = \prod_{\xi \in \boldsymbol{\xi}_k^{(B)}, \xi \neq \infty} (z - \xi).$$

Then the residual matrix can be written as $\mathtt{R}_{h,k} = \mathtt{R}_{h,k}^{(1)} + \mathtt{R}_{h,k}^{(2)} + \mathtt{R}_{h,k}^{(3)}$, where

$$\begin{aligned}
\mathtt{R}_{h,k}^{(1)} &= U(R_B^{G^H}(A_h) \circ^{-1} C_1^{(h)})(R_B^G(B^H) \circ C_2)^H, \\
\mathtt{R}_{h,k}^{(2)} &= \left(R_A^G(A) \circ C_1\right) (R_A^{G^H}(B_k) \circ^{-1} C_2^{(k)})^H V^H, \\
\mathtt{R}_{h,k}^{(3)} &= \left(R_A^G(A) \circ C_1 \left(R_A^G(\infty)\right)^{-1}\right) \left(R_B^G(B^H) \circ C_2 \left(R_B^G(\infty)\right)^{-1}\right)^H,
\end{aligned}$$

with

$$R_A^G(\infty) = \lim_{|\lambda|\to\infty} R_A^G(\lambda) \quad \text{and} \quad R_B^G(\infty) = \lim_{|\lambda|\to\infty} R_B^G(\lambda).$$

Moreover

$$\|\mathtt{R}_{h,k}\|_F^2 = \left\|\mathtt{R}_{h,k}^{(1)}\right\|_F^2 + \left\|\mathtt{R}_{h,k}^{(2)}\right\|_F^2 + \left\|\mathtt{R}_{h,k}^{(3)}\right\|_F^2. \tag{3.8}$$

Remark 3.2.1. *If one of the poles of $\boldsymbol{\xi}_h^{(A)}$ or $\boldsymbol{\xi}_k^{(B)}$ is chosen equal to infinity, then $\mathtt{R}_{h,k}^{(3)} = 0$.*

The residual matrix representation described in Theorem 3.2.1 allows us to develop adaptive techniques for selecting poles in solving Sylvester equations, as extensively discussed in Section 3.2.2.

3.2.1 Proof of Theorem 3.2.1

Theorem 3.2.1 and the proof we provide in this section, are generalizations of the ones provided by Beckermann in [9] for the case of classical rational Krylov methods.

The next two lemmas are trivial consequences of Proposition 2.3.1 and Corollary 2.3.2.

Lemma 3.2.2. *Employing the notation of Theorem 3.2.1, for any $R_A(z) \in \mathcal{P}_h(\mathbb{C}^{b\times b})/q_A(z)$ and $R_B \in \mathcal{P}_k(\mathbb{C}^{b\times b})/q_B(z)$, we have*

$$UU^H R_A(A) \circ C_1 = UR_A(A_h) \circ C_1^{(h)}, \quad \text{and} \quad VV^H R_B(B^H) \circ C_2 = VR_B(B_k) \circ C_2^{(k)},$$

in particular, $R_A(z) \in \mathcal{P}_{h-1}(\mathbb{C}^{b\times b})/q_A(z)$, and $R_B \in \mathcal{P}_{k-1}(\mathbb{C}^{b\times b})/q_B(z)$, imply

$$R(A) \circ C_1 = UR_A(A_h) \circ C_1^{(h)}, \quad \text{and} \quad R_B(B^H) \circ C_2 = VR_B(B_k) \circ C_2^{(k)}$$

respectively.

[1] For simplicity of readability, we remove the subscript from the orthonormal basis notation of Krylov subspaces.

Lemma 3.2.3. *Let $\chi_A(z) \in \mathcal{P}_h(\mathbb{C}^{b\times b})$ and $\chi_B(z) \in \mathcal{P}_k(\mathbb{C}^{b\times b})$ be monic block characteristic polynomial for A_h with respect to $C_1^{(h)}$ and B_k with respect to $C_2^{(k)}$, respectively. Let $R_A^G(z) = \chi_A(z)/q_A(z)$ and $R_B^G(z) = \chi_B(z)/q_B(z)$. It holds*

$$U^H R_A^G(A) \circ C_1 = 0 \quad \text{and} \quad V^H R_B^G(B) \circ C_2 = 0.$$

Moreover $R_A^G(A) \circ C_1$ minimizes $\|R(A) \circ C_1\|_F$ over all the $R(z) \in \mathcal{P}_h(\mathbb{C}^{b\times b})/q_A(z)$ such that $R(z) = P(z)/q_A(z)$ where $P(z)$ is a monic matrix polynomial of degree h. Analogously $R_B^G(B) \circ C_2$ minimizes $\|R(B) \circ C_2\|_F$ over all the $R(z) \in \mathcal{P}_k(\mathbb{C}^{b\times b})/q_B(z)$ with monic numerator of degree k.

Before proceeding with the proof of the theorem, we introduce two additional lemmas that are crucial for its establishment.

Lemma 3.2.4. *Let $R_A(z) \in \mathcal{P}_h(\mathbb{C}^{b\times b})/q_A(z)$. For any z such that $det(R_A(z)) \neq 0$ it holds*

$$R_A(zI_n) \circ^{-1} [R_A(zI_n) \circ Z - R_A(A) \circ Z] = U R_A(zI_{bh}) \circ^{-1} \left[R_A(zI_{bh}) \circ \widetilde{Z} - R_A(A_h) \circ \widetilde{Z}\right], \quad (3.9)$$

where $Z := (zI_n - A)^{-1} C_1$ and $\widetilde{Z} := (zI_{bh} - A_h)^{-1} C_1^{(h)}$.

Similarly, for any $R_B(z) \in \mathcal{P}_k(\mathbb{C}^{b\times b})/q_B(z)$ and for each z such that $det(R_B(z)) \neq 0$

$$R_B(zI_m) \circ^{-1} \left[R_B(zI_m) \circ Y - R_B(B^H) \circ Y\right] = V R_B(zI_{bk}) \circ^{-1} \left[R_B(zI_{bk}) \circ \widetilde{Y} - R_B(B_k) \circ \widetilde{Y}\right],$$

where $Y := (zI_m - B^H)^{-1} C_2$ and $\widetilde{Y} := (zI_{bk} - B_k)^{-1} C_2^{(k)}$.

Proof. We only derive the first equality, the second follows by an analogous argument. Note that the operator $R_A(zI_n)\circ^{-1}$ is well-defined since $\det(R_A(z)) \neq 0$. By Lemma 2.1.4 equation (3.9) is equivalent to

$$[R_A(zI_n) \circ Z - R_A(A) \circ Z](R(z))^{-1} = U \left[R_A(zI_{bh}) \circ \widetilde{Z} - R_A(A_h) \circ \widetilde{Z}\right] (R(z))^{-1},$$

hence, multiplying both sides on the right by $R(z)$, it is sufficient to prove

$$R_A(zI_n) \circ Z - R_A(A) \circ Z = U \left[R_A(zI_{bh}) \circ \widetilde{Z} - R_A(A_h) \circ \widetilde{Z}\right].$$

Since A and $(zI_n - A)^{-1}$ commute and analogously for $(zI_{bh} - A_h)^{-1}$ and A_h, by Lemma 2.1.3 the claim can be equivalently restated as follows:

$$\begin{gathered}(zI_n - A)^{-1} [R_A(zI_n) \circ C_1 - R_A(A) \circ C_1] \\ = \\ U(zI_{bh} - A_h)^{-1} \left[R_A(zI_{bh}) \circ C_1^{(h)} - R_A(A_h) \circ C_1^{(h)}\right].\end{gathered} \quad (3.10)$$

To prove it, we introduce the auxiliary function $G_z(x) := R_A(z) - R_A(x)$. We consider $G_z(x)$ as a function in the variable x, and assume that z is fixed; in particular $G_z(x) = P_z(x)/q_A(x)$, where $P_z(x)$ is a matrix polynomial of degree h in the variable x. Note that $G_z(A) \circ C_1 = R_A(zI_n) \circ C_1 - R_A(A) \circ C_1$; indeed, letting $P_z(x) = \sum_{i=0}^h \Delta_i x^i \in \mathcal{P}_h(\mathbb{C}^{b\times b})$ and $q_A(x) = \sum_{i=0}^h q_i x^i$, from the definition of $G_z(x)$ we have that

$$R_A(x) = R_A(z) - G_z(x) = (q_A(x)R_A(z) - P_z(x))/q_A(x) = \left[\sum_{i=0}^h (q_i R_A(z) - \Delta_i)x^i\right] / q_A(x).$$

Hence,

$$\begin{aligned}R_A(A) \circ C_1 &= q_A(A)^{-1} \left[\sum_{i=0}^h A^i C_1 (q_i R_A(z) - \Delta_i)\right] \\ &= q_A(A)^{-1} \left[R_A(z) \sum_{i=0}^h q_i A^i C_1\right] - q_A(A)^{-1} \left[\sum_{i=0}^h A^i C_1 \Delta_i\right] \\ &= R_A(z) q_A(A)^{-1} q_A(A) C_1 - G_z(A) \circ C_1 = R_A(zI_n) \circ C_1 - G_z(A) \circ C_1.\end{aligned}$$

Analogously, it can be proven that $G_z(A_h) \circ C_1^{(h)} = R_A(zI_{bh}) \circ C_1^{(h)} - R_A(A_h) \circ C_1^{(h)}$. Using the equivalences introduced before, we may rewrite (3.10) as

$$(zI_n - A)^{-1} G_z(A) \circ C_1 = U(zI_{bh} - A_h)^{-1} G_z(A_h) \circ C_1^{(h)}. \tag{3.11}$$

By definition, evaluating $G_z(x)$ at $x = zI_b$ yields $G_z(zI_b) = R_A(z) - R_A(zI_b) = 0$. This implies that the linear matrix polynomial $(xI_b - zI_b)$ is a left solvent for $P_z(x)$, and we may write

$$\widetilde{G}_z(x) := (z - x)^{-1} G_z(x) = -(xI_b - zI_b)^{-1} G_z(x) \in \mathcal{P}_{h-1}(\mathbb{C}^{b\times b})/q_A(x).$$

Thanks to the exactness from Lemma 3.2.2 we have $\widetilde{G}_z(A) \circ C_1 = U\widetilde{G}_z(A_h) \circ C_1^{(h)}$, that by Lemma 2.1.2 is equal to (3.11), concluding the proof. □

Lemma 3.2.5. *Let $\chi_A(z) \in \mathcal{P}_h(\mathbb{C}^{b\times b})$ and $\chi_B(z) \in \mathcal{P}_k(\mathbb{C}^{b\times b})$ be monic block characteristic polynomials for A_h with respect to $C_1^{(h)}$ and B_k with respect to $C_2^{(k)}$, respectively. Let $R_A^G(z) = \chi_A(z)/q_A(z)$ and $R_B^G(z) = \chi_B(z)/q_B(z)$. We have that*

$$(zI_n - A)^{-1} C_1 - U(zI_{bh} - A_h)^{-1} C_1^{(h)} = R_A^G(zI_n) \circ^{-1} R_A^G(A) \circ (zI_n - A)^{-1} C_1,$$

and

$$(zI_m - B^H)^{-1} C_2 - U(zI_{bk} - B_k)^{-1} C_2^{(k)} = R_B^G(zI_m) \circ^{-1} R_B^G(B^H) \circ (zI_m - B^H)^{-1} C_2,$$

Proof. It follows from Lemma 3.2.4 observing that $R_A^G(A_h)C_1^{(h)} = 0$ and $R_B^G(B_k)C_2^{(k)} = 0$. □

We are now ready to give the proof of Theorem 3.2.1:

Proof of Theorem 3.2.1. To simplify the notation we define $Z = (zI_n - A)^{-1}C_1$, $\widetilde{Z} = (zI_{bh} - A_h)^{-1}C_1^{(h)}$, $Y = (\bar{z}I_m - B^H)^{-1}C_2$ and $\widetilde{Y} = (\bar{z}I_{bk} - B_k)^{-1}C_2^{(k)}$. According to Equation (3.2), letting X the solution of the Sylvester equation, we have

$$X - X_{h,k} = \frac{1}{2\pi i} \int_{\gamma_A} ZY^H - U\widetilde{Z}\widetilde{Y}^H V^H dz,$$

where γ_A is a compact contour with positive orientation that encloses the eigenvalues of A and A_h, but not the eigenvalues of B and B_k. Using Lemma 3.2.5 we have

$$X - X_{h,k} = \frac{1}{2\pi i} \int_{\gamma_A} \left((Z - U\widetilde{Z})Y^H + Z(Y - V\widetilde{Y})^H - (Z - U\widetilde{Z})(Y - V\widetilde{Y})^H \right) dz \tag{3.12}$$

$$= \frac{1}{2\pi i} \int_{\gamma_A} \left(R_A^G(zI_n) \circ^{-1} R_A^G(A) \circ Z \right) Y^H dz \tag{3.13}$$

$$+ \frac{1}{2\pi i} \int_{\gamma_A} Z \left(R_B^G(\bar{z}I_m) \circ^{-1} R_B^G(B^H) \circ Y \right)^H dz \tag{3.14}$$

$$- \frac{1}{2\pi i} \int_{\gamma_A} \left(R_A^G(zI_n) \circ^{-1} R_A^G(A) \circ Z \right) \left(R_B^G(\bar{z}I_m) \circ^{-1} R_B^G(B^H) \circ Y \right)^H dz. \tag{3.15}$$

The residual matrix can be written as $\mathtt{R}_{h,k} = A(X - X_{h,k}) - (X - X_{h,k})B$ that is the sum of the three differences of integrals $A\mathcal{S} - \mathcal{S}B$, where $\mathcal{S}$ is substituted by (3.13), (3.14) and (3.15). In the following, we study each difference of integrals separately. Concerning (3.13), by Lemma 2.1.3 we have

$$\frac{1}{2\pi i} A \int_{\gamma_A} \left(R_A^G(zI_n) \circ^{-1} R_A^G(A) \circ Z \right) Y^H dz - \frac{1}{2\pi i} \int_{\gamma_A} \left(R_A^G(zI_n) \circ^{-1} R_A^G(A) \circ Z \right) Y^H B dz \tag{3.16}$$

$$= \frac{1}{2\pi i} \int_{\gamma_A} \left(R_A^G(zI_n) \circ^{-1} R_A^G(A) \circ AZ \right) Y^H dz - \frac{1}{2\pi i} \int_{\gamma_A} \left(R_A^G(zI_n) \circ^{-1} R_A^G(A) \circ Z \right) (B^H Y)^H dz. \tag{3.17}$$

Let now γ_B be a positively oriented compact contour that encloses the eigenvalues of B and B_k, but not the eigenvalues of A and A_h. Since the integrand is $\mathcal{O}(z^{-2})_{z\to\infty}$, we can replace γ_A with γ_B just by changing the sign of the integral. Noting that

$$AZ = (A - zI_n)Z + zI_nZ = -C_1 + zZ \quad \text{and analogously,} \quad B^HY = -C_2 + \bar{z}Y, \tag{3.18}$$

the sum of integrals in (3.17) can be rewritten as

$$-\frac{1}{2\pi i}\int_{\gamma_A}\left(R_A^G(zI_n)\circ^{-1} R_A^G(A)\circ C_1\right)Y^H dz + \frac{1}{2\pi i}\int_{\gamma_A}\left(R_A^G(zI_n)\circ^{-1} R_A^G(A)\circ Z\right)C_2^H dz.$$

Then, changing γ_A with γ_B we obtain

$$\frac{1}{2\pi i}\int_{\gamma_B}\left(R_A^G(zI_n)\circ^{-1} R_A^G(A)\circ C_1\right)Y^H dz, \tag{3.19}$$

since the integral

$$\frac{1}{2\pi i}\int_{\gamma_B}\left(R_A^G(zI_n)\circ^{-1} R_A^G(A)\circ Z\right)C_2^H dz,$$

vanishes for the residual theorem.

The same technique can be used to write the second difference of integrals as

$$\frac{1}{2\pi i}\int_{\gamma_A} Z\left(R_B^G(\bar{z}I_m)\circ^{-1} R_B^G(B^H)\circ C_2\right)^H dz. \tag{3.20}$$

Using again the relations in (3.18), the third difference of integrals can be written as $I_{3,1} + I_{3,2}$, where

$$I_{3,1} = \frac{1}{2\pi i}\int_{\gamma_A}\left(R_A^G(zI_n)\circ^{-1} R_A^G(A)\circ C_1\right)\left(R_B^G(\bar{z}I_m)\circ^{-1} R_B^G(B^H)\circ Y\right)^H dz,$$

and

$$I_{3,2} = -\frac{1}{2\pi i}\int_{\gamma_A}\left(R_A^G(zI_n)\circ^{-1} R_A^G(A)\circ Z\right)\left(R_B^G(\bar{z}I_m)\circ^{-1} R_B^G(B^H)\circ C_2\right)^H dz. \tag{3.21}$$

For a generic choice of poles, it is only guaranteed that the integrand of $I_{3,1}$ is $\mathcal{O}(z^{-1})_{z\to\infty}$ hence, changing γ_A with γ_B, we can rewrite $I_{3,1}$ as

$$\left(R_A^G(\infty\cdot I_n)\circ^{-1} R_A^G(A)\circ C_1\right)\left(R_B^G(\infty\cdot I_m)\circ^{-1} R_B^G(B^H)\circ C_2\right)^H \tag{3.22}$$

$$-\frac{1}{2\pi i}\int_{\gamma_B}\left(R_A^G(zI_n)\circ^{-1} R_A^G(A)\circ C_1\right)\left(R_B^G(\bar{z}I_m)\circ^{-1} R_B^G(B^H)\circ Y\right)^H dz. \tag{3.23}$$

Summing (3.19), (3.20), (3.21), (3.22) and (3.23), we obtain

$$\begin{aligned}\mathtt{R}_{h,k} &= \left(R_A^G(\infty\cdot I_n)\circ^{-1} R_A^G(A)\circ C_1\right)\left(R_B^G(\infty\cdot I_m)\circ^{-1} R_B^G(B^H)\circ C_2\right)^H\\ &+\frac{1}{2\pi i}\int_{\gamma_B}\left(R_A^G(zI_n)\circ^{-1} R_A^G(A)\circ C_1\right)\left(\left(I_m - R_B^G(\bar{z}I_m)\circ^{-1} R_B^G(B^H)\right)\circ Y\right)^H dz\\ &+\frac{1}{2\pi i}\int_{\gamma_A}\left(\left(I_n - R_A^G(zI_n)\circ^{-1} R_A^G(A)\right)\circ Z\right)\left(R_B^G(\bar{z}I_m)\circ^{-1} R_B^G(B^H)\circ C_2\right)^H dz.\end{aligned}$$

Applying Lemma 3.2.5, we have

$$\begin{aligned}\mathtt{R}_{h,k} &= \left(R_A^G(\infty\cdot I_n)\circ^{-1} R_A^G(A)\circ C_1\right)\left(R_B^G(\infty\cdot I_m)\circ^{-1} R_B^G(B^H)\circ C_2\right)^H\\ &+\frac{1}{2\pi i}\int_{\gamma_B}\left(R_A^G(zI_n)\circ^{-1} R_A^G(A)\circ C_1\right)\widetilde{Y}^H V^H dz\\ &+\frac{1}{2\pi i}\int_{\gamma_A} U\widetilde{Z}\left(R_B^G(\bar{z}I_m)\circ^{-1} R_B^G(B^H)\circ C_2\right)^H dz,\end{aligned}$$

and thanks to Lemma 2.1.4 the above term can be rewritten as

$$\begin{aligned}\mathtt{R}_{h,k} = &\left(R_A^G(A)\circ C_1\left(R_A^G(\infty)\right)^{-1}\right)\left(R_B^G(B^H)\circ C_2\left(R_B^G(\infty)\right)^{-1}\right)^H\\ &+\frac{1}{2\pi i}\int_{\gamma_B}\left(R_A^G(A)\circ C_1\right)\left(R_A^{G^H}(\bar{z}I_{bk})\circ^{-1}\widetilde{Y}\right)^H V^H dz\\ &+\frac{1}{2\pi i}\int_{\gamma_A} U\left(R_B^{G^H}(zI_{bh})\circ^{-1}\widetilde{Z}\right)\left(R_B^G(B^H)\circ C_2\right)^H dz.\end{aligned}$$

Finally, by Theorem 2.1.5 we have

$$\begin{aligned}\mathtt{R}_{h,k} = &\left(R_A^G(A)\circ C_1\left(R_A^G(\infty)\right)^{-1}\right)\left(R_B^G(B^H)\circ C_2\left(R_B^G(\infty)\right)^{-1}\right)^H\\ &+\left(R_A^G(A)\circ C_1\right)\left(R_A^{G^H}(B_k)\circ^{-1} C_2^{(k)}\right)^H V^H\\ &+U\left(R_B^{G^H}(A_h)\circ^{-1} C_1^{(h)}\right)\left(R_B^G(B^H)\circ C_2\right)^H.\end{aligned}$$

To prove (3.8) consider the orthogonal projectors $\Pi_A = UU^H$ and $\Pi_B = VV^H$. Applying Lemma 3.2.3 we obtain the sought identities

$$\Pi_A\mathtt{R}_{h,k}(I_{bk}-\Pi_B) = \mathtt{R}_{h,k}^{(1)},\quad (I_{bh}-\Pi_A)\mathtt{R}_{h,k}\Pi_B = \mathtt{R}_{h,k}^{(2)}$$
$$\text{and}\quad (I_{bh}-\Pi_A)\mathtt{R}_{h,k}(I_{bk}-\Pi_B) = \mathtt{R}_{h,k}^{(3)}.$$

□

3.2.2 Pole selection

The results of Theorem 3.2.1 can be used to adaptively find good poles for the block rational Arnoldi algorithm (Algorithm 1) for the solution of Sylvester equations. The concept of employing adaptive pole selection strategies has already been developed in the literature, see for instance cite [41, 42].

During this discussion, we assume that one of the poles in $\boldsymbol{\xi}_k^{(A)}$ or $\boldsymbol{\xi}h^{(B)}$ is chosen to be equal to infinity. Therefore, by Remark 3.2.1, the term $\mathtt{R}_{h,k}^{(3)}$ in the formulation of the residual is equal to zero. With this assumption, the norm of the residual is monitored by the norms of

$$\mathtt{R}_{h,k}^{(1)} = U(R_B^{G^H}(A_h)\circ^{-1} C_1^{(h)})(R_B^G(B^H)\circ C_2)^H,$$

and

$$\mathtt{R}_{h,k}^{(2)} = \left(R_A^G(A)\circ C_1\right)(R_A^{G^H}(B_k)\circ^{-1} C_2^{(k)})^H V^H.$$

Let us start by considering the norm of $\mathtt{R}_{h,k}^{(1)}$. We have that

$$\left\|\mathtt{R}_{h,k}^{(1)}\right\|_F \le \left\|R_B^{G^H}(A_h)\circ^{-1} C_1^{(h)}\right\|_F \cdot \left\|R_B^G(B^H)\circ C_2\right\|_F.$$

By Lemma 3.2.3, the vector $R_B^G(B^H)\circ C_2$ minimizes $\left\|R(B^H)\circ C_2\right\|_F$ over all $R(z)\in\mathcal{P}_h(\mathbb{C}^{b\times b})/q_B(z)$ with monic numerator, for this reason, we choose the new pole by minimizing the norm of $R_B^{G^H}(A_h)\circ^{-1} C_1^{(h)}$.

Let $\chi_B(z) = \sum_{i=0}^k \Gamma_i z^i$ be monic block characteristic polynomial of B_k. By the definition of the operator $\circ^{-1}$, we have

$$\left\|R_B^{G^H}(A_h)\circ^{-1} C_1^{(h)}\right\|_F = \left\|(I_b\otimes\bar{q}_B(A_h))\left(\sum_{i=0}^k \bar{\Gamma}_i\otimes A_h^i\right)^{-1}\mathrm{vec}(C_1^{(h)})\right\|_2,$$

where $\bar{q}_B(z)$ is the conjugate of $q_B(z)$ and $\bar{\Gamma}_i$ denotes the conjugate of the matrix Γ_i.

Assuming for simplicity that A_h is diagonalizable, i.e., $A_h = S_h D_h {S_h}^{-1}$ with D_h diagonal matrix, we have the following bound:

$$\begin{aligned}&\left\| (I_b \otimes \bar{q}_B(A_h)) \left(\sum_{i=0}^{k} \bar{\Gamma}_i \otimes A_h^i \right)^{-1} \operatorname{vec}(C_1^{(h)}) \right\|_2 \\ \leq &\kappa(S_h) \left\| C_1 \right\|_F \left\| (I_b \otimes \bar{q}_B(D_h)) \left(\sum_{i=0}^{k} \bar{\Gamma}_i \otimes D_h^i \right)^{-1} \right\|_2,\end{aligned}$$

where $\kappa(S_h)$ denotes the condition number of S_h. The two norm of $(I_b \otimes \bar{q}_B(D_h))(\sum_{i=0}^{k} \bar{\Gamma}_i \otimes D_h^i)^{-1}$ is equal to the two norm of the matrix

$$\left(\bar{q}_B(D_h) \otimes I_b\right) \left(\sum_{i=0}^{k} D_h^i \otimes \bar{\Gamma}_i \right)^{-1} = \begin{bmatrix} \bar{R}_B^{-1}(\lambda_1) & & \\ & \ddots & \\ & & \bar{R}_B^{-1}(\lambda_h) \end{bmatrix},$$

where $\bar{R}_B(z) = \bar{\chi}_B(z)/\bar{q}_B(z) = (\sum_{i=0}^{h} z^i \bar{\Gamma}_i)/\bar{q}_B(z)$ and $\lambda_1, \ldots, \lambda_h$ are the eigenvalues of A_h. In particular

$$\left\| (I_b \otimes \bar{q}_B(D_h)) \left(\sum_{i=0}^{k} \bar{\Gamma}_i \otimes D_h^i \right)^{-1} \right\|_2 = \max_{i=1,\ldots,h} \left\| \bar{R}_B^{-1}(\lambda_i) \right\|_2.$$

This shows that keeping the function $\left\| \bar{R}_B^{-1}(z) \right\|_2$ small over the eigenvalues of A_h guarantees a small norm for $\mathtt{R}_{h,k}^{(1)}$. In order to obtain a condition independent of h, we can ask for $\left\| \bar{R}_B^{-1}(z) \right\|_2$ to be small on the field of values of A, which encloses the spectra of all A_h.

In the following, we describe practical methods to adaptively choose poles for $\boldsymbol{\xi}_k^{(B)}$. The same techniques can be used to provide poles for $\boldsymbol{\xi}_h^{(A)}$.

Let us assume to know the matrix B_{k-1} obtained after $k-1$ steps of the block rational Arnoldi algorithm (Algorithm 1) with poles $\boldsymbol{\xi}_{k-1}^{(B)}$ and that we want to choose a new pole to perform the next step of the algorithm. As we saw before the norm of $\mathtt{R}_{h,k}^{(1)}$ after the kth step can be monitored by

$$\left\| \bar{R}_B^{-1}(\lambda) \right\|_2 = |\lambda - \bar{\xi}_k| \cdot \left\| \bar{\chi}_k(\lambda)^{-1} \bar{q}_{k-1}(\lambda) \right\|_2, \tag{3.24}$$

for $\lambda \in \mathbb{W}(A)$, where $q_{k-1}(z) = \prod_{\xi \in \boldsymbol{\xi}_{k-1}^{(B)}, \xi \neq \infty} (z - \xi)$ and $\chi_k(z)$ is a monic block characteristic polynomial of B_k. In practice, we assume that the monic block characteristic polynomial of B_{k-1}, say $\chi_{k-1}(z)$, well approximates $\chi_k(z)$ over $\mathbb{W}(A)$, hence we approximate (3.24), by

$$|\lambda - \bar{\xi}_k| \cdot \left\| \bar{\chi}_{k-1}(\lambda)^{-1} \bar{q}_{k-1}(\lambda) \right\|_2. \tag{3.25}$$

To keep (3.25) small over $\mathbb{W}(A)$ we can choose ξ_k as the conjugate of

$$\arg \max_{\lambda \in \mathbb{W}(A)} \left\| \bar{\chi}_k(\lambda)^{-1} \bar{q}_{k-1}(\lambda) \right\|_2.$$

Remark 3.2.2. *By constructing χ_k as outlined in Theorem 2.1.7 and under the assumption that the eigenvalues of B_k are not within $\mathbb{W}(A)$, the function we are maximizing is analytic on $\mathbb{W}(A)$. Therefore, if $\mathbb{W}(A)$ has a nonempty interior, for the maximum modulus principle it is sufficient to maximize the function over its boundary.*

Remark 3.2.3. *In the case of rational Krylov, i.e., $b = 1$ for the solution of Lyapunov equations, that is $B = -A^H$, this result reduces to the choice of poles developed in [41] for the case of A Hermitian and in [42] for general A.*

The numerical computation of ξ_k using the definition of the block characteristic polynomial given by Theorem 2.1.7 is often inaccurate due to the large condition number of the matrices Δ_i. Overcoming this problem requires developing an alternative method to compute the norm of the evaluation of block characteristic polynomials. However, we leave this task for future research as it extends beyond the scope of this thesis.

We now present two methods to monitor the Euclidean norm of $\bar{\chi}_{k-1}(\lambda)^{-1}\bar{q}_{k-1}(\lambda)$ without performing an explicit computation. It is worth noting that this norm is equivalent to $1/\sigma_{\min}(\lambda)$, where $\sigma_{\min}(\lambda)$ represents the minimum singular value of $\bar{\chi}_{k-1}(\lambda)/\bar{q}_{k-1}(\lambda)$.

The first method is to approximate the maximizer of $1/\sigma_{min}(\lambda)$, for $\lambda \in \mathbb{W}(A)$, with the maximizer of the inverse of $|\det(\bar{\chi}_{k-1}(\lambda)/\bar{q}_{k-1}(\lambda))|$ since the absolute value of the determinant is the product of all the singular values. From Theorem 2.1.7 it can be noticed that

$$\det(\bar{\chi}_{k-1}(\lambda)) = \prod_{\mu\in\Lambda(B_{k-1})} (\lambda - \bar{\mu}),$$

hence the choice of the new pole reduces to the conjugate of

$$\arg\max_{\lambda\in\mathbb{W}(A)} \frac{\prod_{\xi\in\boldsymbol{\xi}_{k-1}^{(B)},\xi\neq\infty} |\lambda-\bar{\xi}|^b}{\prod_{\mu\in\Lambda(B_{k-1})}|\lambda-\bar{\mu}|}. \tag{3.26}$$

We refer to this pole selection strategy as *Adaptive Determinant Minimizaztion* (`ADM`).

Remark 3.2.4. *In the context of solving Lyapunov equations, this selection of poles has previously been proposed in [42] as a potential extension of techniques developed for non-block rational Krylov methods. This finding not only provides theoretical justification for such a generalization but also extends its applicability to the solution of Sylvester equations.*

To introduce the second method assume B_{k-1} diagonalizable. In such case, if we let

$$\chi_{k-1}(z) = \prod_{i=1}^{k-1}(zI_b - S_i),$$

as described in Theorem 2.1.7, also the matrices S_i are diagonalizable, hence

$$\begin{aligned}\left\|\bar{\chi}_{k-1}(\lambda)^{-1}\bar{q}_{k-1}(\lambda)\right\|_2 &\leq |\bar{q}_{k-1}(\lambda)| \prod_{i=1}^{k-1}\left\|(\lambda I_b - \bar{S}_i)^{-1}\right\|_2 \\ &\leq |\bar{q}_{k-1}(\lambda)| \prod_{i=1}^{k-1}\frac{\kappa(X_i)}{|\Lambda_{\min}(\lambda-\bar{S}_i)|},\end{aligned} \tag{3.27}$$

where X_i is the matrix of eigenvectors of $\bar{S}_i$ and $\Lambda_{\min}(\lambda - \bar{S}_i)$ denotes the smallest modulus eigenvalue of $\lambda - \bar{S}_i$ for each i. From Theorem 2.1.7 we see that the matrices S_i can be recovered by an arbitrary eigendecomposition of the matrix B_{k-1}, in particular for a fixed λ we can construct S_i using an ordered eigendecomposition of B_{k-1}, where the eigenvalues $\{\mu_i\}$ of B_{k-1} are ordered such that $|\bar{\lambda} - \mu_1| \leq |\bar{\lambda}-\mu_2| \leq \cdots \leq |\bar{\lambda}-\mu_{k-1}|$. With this construction the eigenvalues of S_i are $\mu_{(i-1)b+1}, \mu_{(i-1)b+2}, \ldots, \mu_{ib}$ and (3.27) can be rewritten as

$$\left\|\bar{\chi}_B(\lambda)^{-1}\bar{q}_{k-1}(\lambda)\right\|_2 \leq \left(\prod_{i=1}^{k-1}\kappa(X_i)\right)|\bar{q}_B(\lambda)|\left(\prod_{i=1}^{k-1}(|\lambda-\bar{\mu}_{(i-1)b+1}|)^{-1}\right).$$

This suggests a new method to choose the next shift: ξ_k can be taken as the conjugate of

$$\arg\max_{\lambda\in\mathbb{W}(A)}\left(\prod_{\xi\in\boldsymbol{\xi}_B,\xi\neq\infty}|\lambda-\bar{\xi}|\prod_{i=1}^{k-1}(|\lambda-\bar{\mu}_{(i-1)b+1}|)^{-1}\right), \tag{3.28}$$

where μ_i are the eigenvalues of B_{k-1} ordered as described before.

We refer to this pole selection strategy as *subsampled Adaptive Determinant Minimizaztion* (`sADM`).

Remark 3.2.5. *The main advantage of this choice of poles with respect to the previous one is that we have to maximize a rational function with a much smaller degree.*

3.3 Numerical experiments

In this section we provide some numerical experiment to show the convergence of the block rational Arnoldi algorithm (Algorithm 1) using poles determined in Section 3.2.2: throughout the section, the algorithms that chooses poles accordingly to (3.26) and (3.28) are denoted by `ADM` and `sADM`, respectively. The pole ξ_0 is always chosen equal to infinity, and the techniques developed in Section 2.2.1 are employed to guarantee the last pole equal to infinity at each step. This allows computing the residual as described in Section 3.1 avoiding extra computational costs. The implementation of block rational Arnoldi algorithms is based on the `rktoolbox` for MATLAB, developed in [19].

The numerical simulations have been run on a Intel(R) Core(TM) i5-8250U CPU processor running Ubuntu and MATLAB R2022b.

The experiments only involve real matrices hence, if a nonreal pole is employed, the subsequent is chosen as its conjugate, this allows us to avoid complex matrices. We refer the reader to [109] for a more complete discussion.

In the first experiment, we compute the approximate solution of the Poisson equation

$$\begin{cases} -\Delta u = f & \text{in } \Omega \\ u \equiv 0 & \text{on } \partial\Omega \end{cases}, \qquad \Omega = [0,1]^2.$$

We discretize the domain with a uniformly spaced grid with $n = 4096$ points in each direction, and the operator Δ by finite differences, which yields the Lyapunov equation

$$AX + XA = F, \quad \text{with} \quad A = \frac{1}{h^2}\begin{bmatrix} -2 & 1 & & \\ 1 & -2 & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & -2 \end{bmatrix}$$

where $h = \frac{1}{n+1}$ is the distance between the grid points and F is the matrix obtained evaluating f on the grid points. If the function f is a smooth bivariate function, the matrix F is numerically low-rank, that is it can be approximated by a low-rank matrix $C_1 C_2^H$ where $C_1, C_2 \in \mathbb{C}^{n\times b}$ for an appropriate $b \ll n$, see e.g. [65, Section 2.7].

Figure 3.1 shows the behavior of the normalized residual $\mathtt{R}_{k,k} / \left\|C_1 C_2^H\right\|_F$, for the solution of the Poisson equation with $f(x,y) = 1/(1+x+y)$ with the two proposed choices of poles. In this case, the matrix F has numerical rank 8. We also compared the results with the extended Krylov proposed in [120], which is a block rational Krylov method that alternates a pole equal to zero and a pole equal to infinity. We remark that the iterations of the extended Krylov method are usually faster than a generic block rational Krylov method since in the iterations associated with poles equal to infinity the linear systems are replaced by matrix products and the iterations associated with poles equal to zero are improved using a factorization of the matrix. Table 3.1 contains times and number of iterations required to reach a relative norm of the residual less than 10^{-8} for the solution of discretized Poisson equation with block rational Krylov methods with different choices of poles.

The second experiment is the computation of an approximate solution for the convection-diffusion partial differential equation

$$\begin{cases} -\epsilon\Delta u + \mathbf{w}\cdot\nabla u = f & \text{in } \Omega \\ u \equiv 0 & \text{on } \partial\Omega \end{cases}, \qquad \Omega = [0,1]^2,$$

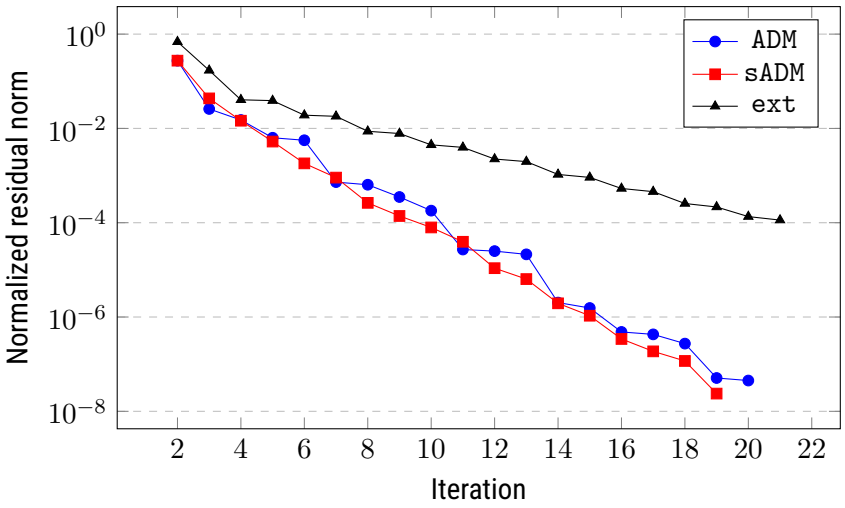


Figure 3.1: Behavior of the residual produced by solving the Poisson equation with block rational Krylov methods, with different choices of poles.

poles	iter	residual	time (s)
ADM	21	$8.82e-09$	0.92
sADM	20	$9.19e-09$	1.10
ext	53	$9.30e-09$	5.91

Table 3.1: Iterations and time needed to reach a relative norm of the residual less than 10^{-8} for the solution of discretized Poisson equation with block rational Krylov methods with different choices of poles.

where $\epsilon \in \mathbb{R}_+$ is the viscosity parameter and $\mathbf{w}$ is the convection vector. Assuming $\mathbf{w} = (\Phi(x), \Psi(y))$, and discretizing the domain with a uniformly spaced grid as before, we obtain the Sylvester equation

$$(\epsilon A + \mathbf{\Phi} B)X + X(\epsilon A + B^H \mathbf{\Psi}) = F$$

where A and F are defined as in the first experiment,

$$\mathbf{\Phi} = \begin{bmatrix} \Phi(h) & & & \\ & \Phi(2h) & & \\ & & \ddots & \\ & & & \Phi(nh) \end{bmatrix}, \qquad \mathbf{\Psi} = \begin{bmatrix} \Psi(h) & & & \\ & \Psi(2h) & & \\ & & \ddots & \\ & & & \Psi(nh) \end{bmatrix}$$

and

$$B = \frac{1}{2h} \begin{bmatrix} 0 & 1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & 1 \\ & & -1 & 0 \end{bmatrix}$$

is the discretization by centered finite differences of the first-order derivative in each direction.

Figure 3.2 shows the behavior of the normalized residual for the solution of the convection-diffusion equation with $\epsilon = 0.0083$, $f(x, y) = 1/(1+x+y)$, $\mathbf{w} = (1 + \frac{(x+1)^2}{4}, \frac{1}{2}y)$ with the two proposed choices of poles and the extended Krylov method. Table 3.2 contains times and number of iterations required to reach a relative norm of the residual less than 10^{-8} for the solution of discretized confection-diffusion equation with block rational Krylov methods with different choices of poles.

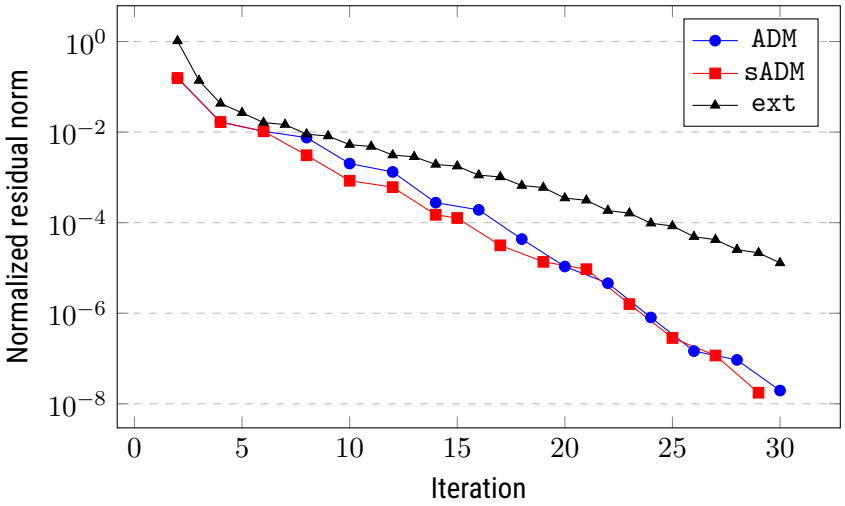


Figure 3.2: Behavior of the residual produced by solving the convection-diffusion equation with block rational Krylov methods, with different choices of poles.

poles	iter	residual	time (s)
ADM	32	$2.18e-09$	2.12
sADM	31	$9.38e-09$	2.05
ext	54	$7.55e-09$	7.42

Table 3.2: Iterations and time needed to reach a relative norm of the residual less than 10^{-8} for the solution of discretized convection-diffusion equation with block rational Krylov methods with different choices of poles.

Chapter 4

Tensor Sylvester equations

The material presented in this chapter is the result of the work described in [24].

We generalize the techniques developed in Chapter 3 to approximate the solution of the tensor Sylvester equation

$$\mathcal{X} \times_1 A_1 + \mathcal{X} \times_2 A_2 + \cdots + \mathcal{X} \times_d A_d = \mathcal{C}, \tag{4.1}$$

where $\times_i$ denotes the ith mode product for tensors [81] and $A_i \in \mathbb{C}^{n_i \times n_i}$ are square matrices for each $i = 1, \ldots, d$, the unknown $\mathcal{X}$ and the right-hand side $\mathcal{C}$ are d-dimensional tensors of size $n_1 \times \cdots \times n_d$ and $\mathcal{C}$ is low rank in Tucker or tensor train format. Tensor Sylvester equations arise, for instance, in the numerical solution of discretized d-dimensional PDEs on tensorized domains [52, 63, 123]. The problem is equivalent to solving the linear system

$$\mathbf{A}\mathbf{x} = \mathbf{c} \tag{4.2}$$

where

$$\mathbf{x} = \begin{bmatrix} \mathcal{X}(1,1,\ldots,1) \\ \mathcal{X}(2,1,\ldots,1) \\ \vdots \\ \mathcal{X}(n_1,\ldots,n_{d-1},n_d) \end{bmatrix} \in \mathbb{C}^{n_1 \cdots n_d} \quad \text{and} \quad \mathbf{c} = \begin{bmatrix} \mathcal{C}(1,1,\ldots,1) \\ \mathcal{C}(2,1,\ldots,1) \\ \vdots \\ \mathcal{C}(n_1,\ldots,n_{d-1},n_d) \end{bmatrix} \in \mathbb{C}^{n_1 \cdots n_d}$$

are vectorizations of $\mathcal{X}$ and $\mathcal{C}$ respectively, and

$$\mathbf{A} = \sum_{i=1}^{d} I_{n_d} \otimes \cdots \otimes I_{n_{i+1}} \otimes A_i \otimes I_{n_{i-1}} \otimes \cdots \otimes I_{n_1}. \tag{4.3}$$

However, for large d, employing standard computational methods to solve the linear system becomes unfeasible due to the exponential growth in the size of the involved matrix. To address this issue, several algorithms have been developed for solving linear systems involving tensors [2, 4, 6, 36, 37, 80]. In particular, Kressner and Tobler [83] studied the solution of tensor Sylvester equations with a right-hand side of rank one by projecting onto tensorized polynomial Krylov subspaces. This work extends those techniques to equations with a more general low-rank right-hand side, employing rational Krylov subspaces, which provide more flexibility in the choice of projection subspaces through pole selection.

4.1 Low rank tensors

In this section we briefly recall basic concepts about tensors, focusing on the representation of low-rank tensors in Tucker and tensor train formats. A broader treatment of the topic can be found in [65, 81, 101].

The simplest way to define the rank of a d-dimensional tensor $\mathcal{X} \in \mathbb{C}^{n_1 \times n_2 \times \cdots \times n_d}$, is the minimum number of "rank one" tensors whose sum equals to $\mathcal{X}$. Denoting by $\mathbf{x}$ a vectorization of $\mathcal{X}$, the CP rank is given by the smallest integer k such that

$$\mathbf{x} = \sum_{i=1}^{k} \mathbf{u}_{i,1} \otimes \mathbf{u}_{i,2} \otimes \cdots \otimes \mathbf{u}_{i,d}, \quad \text{with } \mathbf{u}_{i,j} \in \mathbb{C}^{n_j} \text{ for each } i, j,$$

where $\otimes$ denotes the Kronecker product. The above representation of a tensor is called Canonical Polyadic decomposition (usually denoted by CP). The main issue of this decomposition is that determining the CP rank of a given tensor is a NP-hard problem [72]. To overcome this issue, several alternative definitions of rank have been introduced. Before introducing them, we need to discuss a couple of basic concepts regarding tensors.

Let $\mathcal{X} \in \mathbb{C}^{n_1 \times \cdots \times n_d}$. For each $i = 1, \ldots, d$, the mode-i fibers of $\mathcal{X}$ are the column vectors of n_i components that contain all the entries of $\mathcal{X}$ obtained by fixing all the indices except for the ith. Moreover, the mode-i unfolding of $\mathcal{X}$ is the matrix of size $n_i \times (n_1 \cdots n_{i-1} \cdot n_{i+1} \cdots n_d)$ containing all the mode-i fibers of $\mathcal{X}$ in its columns. For a broader discussion, see [81, Section 2.4].

Definition 4.1.1. *The multilinear rank of a tensor $\mathcal{X}$ is defined as the vector $\mathbf{k} = (k_1, \ldots, k_d)$, where for each $i = 1, \ldots, d$, k_i is the rank of the mode-i unfolding $\mathcal{X}_{(i)}$.*

Similarly to how the CP rank is related to the CP decomposition, the multilinear rank can also be associated with a tensor decomposition introduced by Tucker in [126], which decomposes a tensor $\mathcal{X} \in \mathbb{C}^{n_1 \times n_2 \times \cdots \times n_d}$ into a core tensor $\mathcal{G} \in \mathbb{C}^{k_1 \times k_2 \times \cdots \times k_d}$ multiplied by tall and skinny matrices $U_i \in \mathbb{C}^{n_i \times k_i}$ with orthonormal columns along each mode i, that is,

$$\mathcal{X} = \mathcal{G} \times_1 U_1 \times_2 U_2 \times_3 \cdots \times_d U_d. \tag{4.4}$$

The generators of a Tucker decomposition are usually denoted by $[\![\mathcal{G}; U_1, \ldots, U_d]\!]$.

Remark 4.1.1. *As described in [81, Section 4], the mode-i unfolding of the tensor* (4.4) *can be written as*

$$\mathcal{X}_{(i)} = U_i \mathcal{G}_{(i)} (U_d \otimes \cdots \otimes U_{i+1} \otimes U_{i-1} \otimes \cdots \otimes U_1). \tag{4.5}$$

Note that if $k_i \ll n_i$, for each i, the Tucker decomposition allows us to significantly compress the data. For a given tensor, an approximant in Tucker format can be computed by repeatedly truncating the mode-i unfoldings. This procedure is usually known as multilinear SVD, or high-order SVD (HOSVD), see [35].

We remark that the memory needed to store a tensor in Tucker format is $\mathcal{O}(k_1 \cdots k_d + k_1 n_1 + \cdots + k_d n_d)$, which is a great benefit with respect to storing the full tensor. However, the needed storage is exponential in the dimension of the tensor, so this representation becomes unfeasible if d is too large. To overcome this problem other low-rank representations have been developed, such as tensor trains introduced by Oseledets in [101].

Given a tensor $\mathcal{X} \in \mathbb{C}^{n_1 \times \cdots \times n_d}$, a tensor train decomposition (also called TT decomposition) consists in a sequence of tensors $\mathcal{G}_1 \in \mathbb{C}^{n_1 \times r_1}, \mathcal{G}_2 \in \mathbb{C}^{r_1 \times n_2 \times r_2}, \ldots, \mathcal{G}_{d-1} \in \mathbb{C}^{r_{d-2} \times n_{d-1} \times r_{d-1}}, \mathcal{G}_d \in \mathbb{C}^{r_{d-1} \times n_d}$, called carriages, such that

$$\mathcal{X}_{(i_1, \ldots, i_d)} = \sum_{s_1, \ldots, s_{d-1}} \mathcal{G}_{1(i_1, s_1)} \mathcal{G}_{2(s_1, i_2, s_2)} \cdots \mathcal{G}_{d-1(s_{d-2}, i_{d-1}, s_{d-1})} \mathcal{G}_{d(} s_{d-1}, i_d).$$

for each $(i_1, \ldots, i_d) \leq (n_1, \ldots, n_d)$. The numbers $r_1, \ldots, r_{d-1}$ are called ranks of the decomposition.

For each $i = 1, \ldots, d-1$, let $X^{\{i\}} \in \mathbb{C}^{n_1 \cdots n_i \times n_{i+1} \cdots n_d}$ be the matrix obtained by grouping the first i indices of a tensor $\mathcal{X}$ as row indices, and the remaining ones as column indices. The TT rank of a tensor is defined as follows.

Definition 4.1.2. *Given a tensor $\mathcal{X} \in \mathbb{C}^{n_1 \times \cdots \times n_d}$, the vector $(r_1, \ldots, r_{d-1})$, where r_i is the rank of $X^{\{i\}}$, is called tensor train rank (sometimes denoted by TT rank) of $\mathcal{X}$.*

The definition of TT rank and TT decomposition are closely related, in particular for each tensor there exists a tensor train decomposition with ranks component-wise smaller than or equal to its tensor train rank (see [101, Theorem 2.1]). In practice, for any $\epsilon > 0$, every tensor $\mathcal{Y}$ can be approximated by a tensor $\mathcal{X}$ in TT format with relative accuracy ϵ, i.e.,

$$\|\mathcal{X} - \mathcal{Y}\|_F \leq \epsilon \|\mathcal{Y}\|_F,$$

employing the TT-SVD algorithm. We refer to [101] for further details.

We conclude this section deriving a low rank representation for $\mathcal{X}_{(i)}$, where $\mathcal{X}$ is a tensor in TT format with carriages $\{\mathcal{G}_1, \dots, \mathcal{G}_d\}$. First of all, we notice that any entry of $\mathcal{X}$ can be written as

$$\mathcal{X}_{(i_1,\dots,i_d)} = \mathbf{a}_{(i_1,\dots,i_{j-1})} G_j(i_j) \mathbf{b}_{(i_{j+1},\dots,i_d)},$$

or equivalently

$$\mathcal{X}_{(i_1,\dots,i_d)} = \left(\mathbf{b}^T_{(i_{j+1},\dots,i_d)} \otimes \mathbf{a}_{(i_1,\dots,i_{j-1})}\right) \mathrm{vec}(G_j(i_j)), \tag{4.6}$$

where $\mathbf{a}_{(i_1,\dots,i_{j-1})}$ and $\mathbf{b}_{(i_{j+1},\dots,i_d)}$, are a row and a column vector, respectively, defined as

$$(\mathbf{a}_{(i_1,\dots,i_{j-1})})_h = \begin{cases} \sum_{s_1,\dots,s_{j-2}} \mathcal{G}_1{}_{(i_1,s_1)} \mathcal{G}_2{}_{(s_1,i_2,s_2)} \cdots \mathcal{G}_{j-1}{}_{(s_{j-2},i_{j-1},h)} & \text{for } j > 1, \\ 1 & \text{otherwise,} \end{cases}$$

$$(\mathbf{b}_{(i_{j+1},\dots,i_d)})_k = \begin{cases} \sum_{s_{j+1},\dots,s_{d-1}} \mathcal{G}_{j+1}{}_{(k,i_{j+1},s_{j+1})} \cdots \mathcal{G}_{d-1}{}_{(s_{d-2},i_{d-1},s_{d-1})} \mathcal{G}_d(s_{d-1}, i_d) & \text{for } j < d, \\ 1 & \text{otherwise} \end{cases}$$

and $G_j(i_j)$ is a matrix defined as

$$G_j(i_j)_{h,k} = \begin{cases} \mathcal{G}_1(i_1, k) & \text{if } j = 1, \\ \mathcal{G}_d(h, i_d) & \text{if } j = d, \\ \mathcal{G}_j(h, i_j, k) & \text{otherwise.} \end{cases}$$

Noting that

$$\begin{bmatrix} \mathrm{vec}(G_j(1))^T \\ \mathrm{vec}(G_j(2))^T \\ \vdots \\ \mathrm{vec}(G_j(n_j))^T \end{bmatrix} = \begin{cases} \mathcal{G}_1 & \text{if } j = 1, \\ (\mathcal{G}_j)_{(2)} & \text{otherwise,} \end{cases}$$

from (4.6) we have

$$\mathcal{X}_{(j)} = \begin{cases} \mathcal{G}_1 \mathbf{c} & \text{if } j = 1, \\ (\mathcal{G}_j)_{(2)} \mathbf{c} & \text{otherwise,} \end{cases} \tag{4.7}$$

where $\mathbf{c}$ is the block row that has as cloumns the vectors $\mathbf{b}_{(i_{j+1},\dots,i_d)} \otimes \mathbf{a}^T_{(i_1,\dots,i_{j-1})}$, ordered lexicographically with respect to $(i_1, \cdots, i_d)$.

4.2 Tensorized Krylov methods

Employed by Kressner and Tobler in [83] for solving the tensor Sylvester equation (4.1) with $\mathcal{C}$ of CP rank one, i.e.,

$$\mathrm{vec}(\mathcal{C}) = \mathbf{c}_1 \otimes \mathbf{c}_2 \otimes \cdots \otimes \mathbf{c}_d, \quad \text{with } \mathbf{c}_i \in \mathbb{C}^{n_i},$$

tensorized Krylov subspaces are defined as

$$\mathcal{K}^{\otimes}_{\mathbf{k}}(\{A_i\}_{i=1}^d, \{\mathbf{c}_i\}_{i=1}^d) = \mathrm{span}(\mathcal{K}_{k_1}(A_1, \mathbf{c}_1) \otimes \mathcal{K}_{k_2}(A_2, \mathbf{c}_2) \otimes \cdots \otimes \mathcal{K}_{k_d}(A_d, \mathbf{c}_d)), \tag{4.8}$$

where $\mathbf{k} = (k_1, \dots, k_d)$.

Similarly to classical Sylvester equations, an approximate solution of a tensor Sylvester equation with rank one right-hand side is given by

$$\mathcal{X}_{\mathbf{k}} = \mathcal{Y}_{\mathbf{k}} \times_1 V_1 \times_2 V_2 \times_3 \cdots \times_d V_d, \tag{4.9}$$

where $\mathcal{Y}_{\mathbf{k}}$ is the solution of the projected problem onto the tensorized Krylov subspace, that is,

$$\mathcal{Y}_{\mathbf{k}} \times_1 A_1^{(k_1)} + \cdots + \mathcal{Y}_{\mathbf{k}} \times_d A_d^{(k_d)} = \mathcal{C}_{\mathbf{k}}, \tag{4.10}$$

with $A_i^{(k_i)} = V_i^H A_i V_i$ for each $i = 1, \ldots, d$ and $\mathcal{C}_{\mathbf{k}} = \mathcal{C} \times_1 V_1^H \times_2 \cdots \times_d V_d^H$, where, for each i, V_i is an orthonormal basis of $\mathcal{K}_{k_i}(A_i, C_i)$.

Tensorized Krylov subspaces can be also described using multivariate polynomials, as it is stated in the next lemma ([83, Lemma 3.2]).

Lemma 4.2.1. *Let $\mathcal{P}_{\mathbf{k}}(\mathbb{C})$ be the space of multivariate polynomials with degree bounded by $\mathbf{k}$. We have*

$$\mathcal{K}_{\mathbf{k}}^{\otimes}(\{A_i\}_{i=1}^d, \{\mathbf{c}_i\}_{i=1}^d) = \{p(A_1, ..., A_d)(\mathbf{c}_1 \otimes \cdots \otimes \mathbf{c}_d), \text{ for } p \in \mathcal{P}_{\mathbf{k}}(\mathbb{C})\},$$

where for each $p(x_1, \ldots, x_d) = \sum_{I=(i_1,\ldots,i_d)\leq \mathbf{k}} \gamma_I x^{i_1} \cdots x^{i_d} \in \mathcal{P}_{\mathbf{k}}(\mathbb{C})$,

$$p(A_1, \ldots, A_d) = \sum_{I=(i_1,\ldots,i_d)\leq \mathbf{k}} \gamma_I A_1^{i_1} \otimes \cdots \otimes A_d^{i_d}.$$

The authors have also proven that the solution of a tensor Sylvester equation can be well approximated by a low-rank tensor, relating the norm of the error with the approximation of the function $\frac{1}{x_1 s + x_2 \cdots + x_d}$ with a sum of separable multivariate functions, see [83, Theorem 2.5]. Moreover, they also analyzed the effects of using extended Krylov subspaces (i.e., $\mathcal{Q}_{k_i}(A_i, \mathbf{c}_i, \boldsymbol{\xi}_{k_i})$, where $\boldsymbol{\xi}_{k_i}$ is given by alternating 0 and ∞), in the construction of tensorized Krylov subspaces.

In the next sections, we generalize the procedure described above to the solution of the tensor Sylvester equation (4.1) for $\mathcal{C}$ with low multilinear or TT rank employing as V_i, an orthonormal basis for the block rational Krylov subspaces $\mathcal{Q}_{k_i}(A_i, C_i)$ with appropriate block vectors C_i, for $i = 1, \ldots, d$.

4.2.1 Tensorized block rational Krylov methods

One of the novelties of this thesis is to analyze the use of block rational Krylov subspaces in tensorized Krylov methods for solving the tensor Sylvester equation (4.1). On one hand, the use of block Krylov subspaces allows us to easily treat the case of C_i with more than one column, on the other hand, the use of rational Krylov subspaces gives us more freedom in the choice of the projection subspace, through the pole selection.

Definition 4.2.1. *For each $i = 1, \ldots d$, let $A_i \in \mathbb{C}^{n_i \times n_i}$ and $C_i \in \mathbb{C}^{n_i \times b_i}$. Let $\mathbf{k} = (k_1, \ldots, k_d)$, with $k_i \in \mathbb{N}$ and for each $i = 1, \ldots, d$, let $\boldsymbol{\xi}_i \subseteq \bar{\mathbb{C}}$ be a sequence of k_i poles that are not eigenvalues of A_i. We define the tensorized block rational Krylov subspace associated with $\{A_i\}_{i=1}^d, \{C_i\}_{i=1}^d$ and $\{\boldsymbol{\xi}_i\}_{i=1}^d$ as*

$$\mathcal{Q}_{\mathbf{k}}^{\otimes}(\{A_i\}, \{C_i\}, \{\boldsymbol{\xi}_i\}) = \left\{ \sum_{i=1}^{s} Z_i \Gamma_i, \text{ for } s \in \mathbb{N}, Z_i \in \mathcal{W}, \Gamma_i \in \mathbb{C}^{b_1 \cdots b_d \times b_1 \cdots b_d} \text{ for each } i \right\},$$

with $\mathcal{W} = \mathcal{Q}_{k_1}(A_1, C_1, \boldsymbol{\xi}_1) \otimes \mathcal{Q}_{k_2}(A_2, C_2, \boldsymbol{\xi}_2) \otimes \cdots \otimes \mathcal{Q}_{k_d}(A_d, C_d, \boldsymbol{\xi}_d)$.

For simplicity of notation, we sometimes omit poles, denoting a tensorized block rational Krylov subspace just by $\mathcal{Q}_{\mathbf{k}}^{\otimes}(\{A_i\}_{i=1}^d, \{C_i\}_{i=1}^d)$. The relation between rational Krylov subspaces and rational functions can be extended also in the case of tensorized block rational Krylov spaces. First of all, we define an extension of the operator $\circ$ to multivariate polynomials.

Definition 4.2.2. *Let*

$$P = \sum_{I=(i_1,\ldots,i_d)\leq \mathbf{k}} \Gamma_I x_1^{i_1} \cdot x_2^{i_2} \cdots x_d^{i_d} \in \mathcal{P}_{\mathbf{k}}(\mathbb{C}^{b_1 \cdots b_d \times b_1 \cdots b_d}),$$

and let $A_i \in \mathbb{C}^{n_i \times n_i}$, $C_i \in \mathbb{C}^{n_i \times b_i}$ *for* $i = 1, \dots, d$. *We define*

$$P(A_1, \dots, A_d) \circ (C_1, \dots, C_d) = \sum_{I=(i_1,\dots,i_d)\leq \mathbf{k}} (A_1^{i_1} C_1 \otimes A_2^{i_2} C_2 \otimes \cdots \otimes A_d^{i_d} C_d)\Gamma_I.$$

Moreover, if $R(x_1, \dots, x_d) = P(x_1, \dots, x_d)/q_1(x_1) \cdots q_d(x_d)$ *with* $q_i \in \mathcal{P}(\mathbb{C})$ *for each* i, *we define*

$$R(A_1, \dots, A_d) \circ (C_1, \dots, C_d) = q_1(A_1)^{-1} \otimes \cdots \otimes q_d(A_d)^{-1} \cdot (P(A_1, \dots, A_d) \circ (C_1, \dots, C_d)).$$

Now we can state the following lemma.

Lemma 4.2.2. *It holds*

$$\mathcal{Q}_{\mathbf{k}}^{\otimes}(\{A_i\}, \{C_i\}, \{\boldsymbol{\xi}_i\}) = \{R(A_1, \dots, A_d) \circ (C_1, \dots, C_d) : R \in \mathcal{P}_{\mathbf{k}}(\mathbb{C}^{b_1 \cdots b_d \times b_1 \cdots b_d})/q\},$$

where $q(x_1, \dots, x_d) := q_1(x_1) \cdots q_d(x_d)$, *with* $q_i(x) = \prod_{\xi \in \boldsymbol{\xi}_i, \xi \neq \infty}(x - \xi)$ *for each* i.

Proof. Let $Z \in \{\mathcal{Q}_{k_1}(A_1, C_1, \boldsymbol{\xi}_1) \otimes \mathcal{Q}_{k_2}(A_2, C_2, \boldsymbol{\xi}_2) \otimes \cdots \otimes \mathcal{Q}_{k_d}(A_d, C_d, \boldsymbol{\xi}_d)\}$. For each i there exists a univariate rational function $R_i(x) = P_i(x)/q_i(x)$ with $P_i = \sum_{j=0}^{k_i} \Gamma_j^{(i)} x^j \in \mathcal{P}_{k_i}(\mathbb{C}^{b_i \times b_i})$, such that

$$Z = R_1(A_1) \circ C_1 \otimes R_2(A_2) \circ C_2 \otimes \cdots \otimes R_d(A_d) \circ C_d.$$

Denoting by $R_Z(x_1, \dots, x_d) = \sum_{I \leq \mathbf{k}} \Gamma_I x_1^{i_1} \cdots x_d^{i_d}/q_1(x_1) \cdots q_d(x_d)$, where $\Gamma_I = \Gamma_{i_1}^{(1)} \otimes \Gamma_{i_2}^{(2)} \otimes \cdots \otimes \Gamma_{i_d}^{(d)}$, with $I = (i_1, \dots, i_d)$, we have

$$\begin{aligned}
&R_Z(A_1, A_2, \dots, A_d) \circ (C_1, C_2, \dots, C_d) \\
&= q_1(A_1)^{-1} \otimes \cdots \otimes q_d(A_d)^{-1} \sum_{I=(i_1,\dots,i_d)\leq \mathbf{k}} (A_1^{i_1} C_1 \otimes A_2^{i_2} C_2 \otimes \cdots \otimes A_d^{i_d} C_d)\Gamma_I. \\
&= R_1(A_1) \circ C_1 \otimes R_2(A_2) \circ C_2 \otimes \cdots \otimes R_d(A_d) \circ C_d \\
&= Z.
\end{aligned}$$

Hence, if $Z = \sum_{i=1}^{s} W_i \Delta_i$, with $W_i \in \{\mathcal{Q}_{k_1}(A_1, C_1, \boldsymbol{\xi}_1) \otimes \mathcal{Q}_{k_2}(A_2, C_2, \boldsymbol{\xi}_2) \otimes \cdots \otimes \mathcal{Q}_{k_d}(A_d, C_d, \boldsymbol{\xi}_d)\}$ and $\Delta_i \in \mathbb{C}^{b_1 \cdots b_d \times b_1 \cdots b_d}$ for each i, then letting

$$R_Z = \sum_{i=1}^{s} \Delta_i R_{W_i} \in \mathcal{P}_{\mathbf{k}}(\mathbb{C}^{b_1 \cdots b_d \times b_1 \cdots b_d})/q(x_1) \cdots q(x_d),$$

we have

$$Z = R_Z(A_1, A_2, \dots, A_d) \circ (C_1, C_2, \dots, C_d).$$

To prove the other inclusion let $R(x_1, \dots, x_d) = P(x_1, \dots, x_d)/q_1(x_1) \cdots q_d(x_d)$, where the numerator is defined as $P = \sum_{I \leq \mathbf{k}} \Gamma_I x_1^{i_1} \cdots x_d^{i_d} \in \mathcal{P}_{\mathbf{k}}(C^{b_1 \cdots b_d \times b_1 \cdots b_d})$. Since every matrix in $\mathbb{C}^{b_1 \cdots b_d \times b_1 \cdots b_d}$ can be written as a linear combination of matrices with only one nonzero entry, in particular, it holds

$$\operatorname{span}\{\mathbb{C}^{b_1 \times b_1} \otimes \mathbb{C}^{b_2 \times b_2} \otimes \cdots \otimes \mathbb{C}^{b_d \times b_d}\} = \mathbb{C}^{b_1 \cdots b_d \times b_1 \cdots b_d},$$

therefore, each Γ_I can be written as

$$\Gamma_I = \sum_{j=1}^{t_I} \alpha_{j,I} \Gamma_{j,I}^{(1)} \otimes \Gamma_{j,I}^{(2)} \otimes \cdots \otimes \Gamma_{j,I}^{(d)},$$

for $\alpha_{j,I} \in \mathbb{C}$ and $\Gamma_{j,I}^{(s)} \in \mathbb{C}^{b_s \times b_s}$ for each s, hence $R(A_1, \dots, A_d) \circ (C_1, \dots, C_d)$ equals to

$$\sum_{I=(i_1,\dots,i_d)\leq \mathbf{k}} \sum_{j=1}^{t_I} \alpha_{j,I} q_1(A_1)^{-1} A_1^{i_1} C_1 \Gamma_{j,I}^{(1)} \otimes q_2(A_2)^{-1} A_2^{i_2} C_2 \Gamma_{j,I}^{(2)} \otimes \cdots \otimes q_d(A_d)^{-1} A_d^{i_d} C_d \Gamma_{j,I}^{(d)},$$

that is a linear combination of elements in $\{\mathcal{Q}_{k_1}(A_1, C_1, \boldsymbol{\xi}_1) \otimes \mathcal{Q}_{k_2}(A_2, C_2, \boldsymbol{\xi}_2) \otimes \cdots \otimes \mathcal{Q}_{k_d}(A_d, C_d, \boldsymbol{\xi}_d)\}$. □

Similarly to the polynomial Krylov case, tensorized block rational Krylov subspaces can be employed for approximating the solution of (4.1) by the tensor $\mathcal{X}_{\mathbf{k}}$ defined in (4.9), where, for each i, the matrix V_i is an orthonormal basis of $\mathcal{Q}_{k_i}(A_i, C_i, \boldsymbol{\xi}_i)$ for an appropriate choice of the block vectors C_i that depends on the low-rank representation of $\mathcal{C}$. This aspect is discussed in Section 4.2.3 and Section 4.2.4 for $\mathcal{C}$ in Tucker and tensor train format, respectively.

Remark 4.2.1. *The solvability of* (4.1) *does not guarantee the solvability of the projected equations* (4.10). *A sufficient condition to avoid this issue is to require*

$$0 \notin \mathbb{W}(\oplus_{i=1}^{d} A_i) = \mathbb{W}(A_1) + \mathbb{W}(A_2) + \cdots + \mathbb{W}(A_d).$$

However, this condition can be hard to verify. In practice, if a projected equation is not solvable, we can just change the projection space, for instance, using different poles.

4.2.2 Convergence analysis

In the following, we combine the results provided by Beckermann, Kressner and Tobler in [11] with the outcomes of Chapter 3 to analyze the convergence of tensorized block rational Krylov methods. The theoretical results of this section are fundamental in developing efficient ways to adaptively determine poles for the method and to compute the residual, topics that are extensively discussed in Sections 4.3 and 4.4.

To easily apply the results of [11], we consider the tensor Sylvester equation in vectorized form, (4.3); we define $\mathbf{V} = V_d \otimes V_{d-1} \otimes \cdots \otimes V_1$, and for each $i = 1, \ldots, d$,

$$\begin{aligned} \mathbf{V}_i &= I_{n_d} \otimes \cdots \otimes I_{n_{i+1}} \otimes V_i \otimes I_{n_{i-1}} \otimes \cdots \otimes I_{n_1}, \\ \overline{\mathbf{V}}_i &= V_d \otimes \cdots \otimes V_{i+1} \otimes I_{n_i} \otimes V_{i-1} \otimes \cdots \otimes V_1. \end{aligned}$$

We denote by $r(\mathbf{V}, A_1, \ldots, A_d, \mathbf{c})$, sometimes abbreviated by $\mathbf{r}$, the residual $\mathbf{c} - \mathbf{A}\mathbf{V}\mathbf{y}$, where $\mathbf{y}$ is the vectorization of the tensor $\mathcal{Y}_{\mathbf{k}}$ that solves the projected equation (4.10). Analogously,

$$r(\mathbf{V}_i, A_1^{(k_1)}, \ldots, A_{i-1}^{(k_{i-1})}, A_i, A_{i+1}^{(k_{i+1})}, \ldots, A_d^{(k_d)}, \overline{\mathbf{V}}_i^H \mathbf{c}),$$

is defined as

$$\overline{\mathbf{V}}_i^H \mathbf{c} - \left(A_d^{(k_d)} \otimes \cdots \otimes A_{i+1}^{(k_{i+1})} \otimes A_i \otimes A_{i-1}^{(k_{i-1})} \otimes \cdots \otimes A_1^{(k_1)} \right) \mathbf{V}_i \mathbf{y}.$$

To describe a representation of the residual that depends on the poles of the rational Krylov subspaces, we start by considering Proposition 2.2 of [11].

Proposition 4.2.3 ([11]). *With the notation introduced above, the following statements hold:*

1. *The residual $\mathbf{r} = \mathbf{c} - \mathbf{A}\mathbf{y}$ can be represented as*

$$\mathbf{r} = \sum_{i=1}^{d} \overline{\mathbf{V}}_i r(\mathbf{V}_i, A_1^{(k_1)}, \ldots A_{i-1}^{(k_{i-1})}, A_i, A_{i+1}^{(k_{i+1})}, \ldots, A_d^{(k_d)}, \overline{\mathbf{V}}_i^H \mathbf{c}) + \widehat{\mathbf{c}},$$

   where the remainder term $\widehat{\mathbf{c}} = (\prod_{i=1}^{d} (I - \overline{\mathbf{V}}_i \overline{\mathbf{V}}_i^H))\mathbf{c}$ vanishes for $\mathbf{c} \in \operatorname{span}(\mathbf{V})$;

2. *The vectors $\widehat{\mathbf{c}}$ and $\overline{\mathbf{V}}_i \overline{\mathbf{V}}_i^H \mathbf{r} = \overline{\mathbf{V}}_i r(\mathbf{V}_i, A_1^{(k_1)}, \ldots A_{i-1}^{(k_{i-1})}, A_i, A_{i+1}^{(k_{i+1})}, \ldots, A_d^{(k_d)}, \overline{\mathbf{V}}_i^H \mathbf{c})$, for $i = 1, \ldots, d$ are mutually orthogonal. In particular, this implies*

$$\|\mathbf{r}\|_2^2 = \sum_{i=1}^{d} \left\| r(\mathbf{V}_i, A_1^{(k_1)}, \ldots, A_{i-1}^{(k_{i-1})}, A_i, A_{i+1}^{(k_{i+1})}, \ldots, A_d^{(k_d)}, \overline{\mathbf{V}}_i^H \mathbf{c}) \right\|_2^2 + \|\widehat{\mathbf{c}}\|_2^2 .$$

Thanks to the previous proposition, to monitor the norm of the residual it is sufficient to control the norms of $\widehat{\mathbf{c}}$ and $\overline{\mathbf{V}}_i^H \mathbf{r}$. For each i, the partial residual $\overline{\mathbf{V}}_i^H \mathbf{r}$ is the vectorization of the tensor

$$\mathcal{R}_i = \overline{\mathcal{C}}_i - \overline{\mathcal{Y}}^i_{\mathbf{k}} \times_1 A_1^{(k_1)} - \cdots - \overline{\mathcal{Y}}^i_{\mathbf{k}} \times_{i-1} A_{i-1}^{(k_{i-1})} - \overline{\mathcal{Y}}^i_{\mathbf{k}} \times_i A_i - \overline{\mathcal{Y}}^i_{\mathbf{k}} \times_{i+1} A_{i+1}^{(k_{i+1})} - \cdots - \overline{\mathcal{Y}}^i_{\mathbf{k}} \times_d A_d^{(k_d)},$$

where $\overline{\mathcal{C}}_i = \mathcal{C} \times_1 V_1^H \times_2 \cdots \times_{i-1} V_{i-1}^H \times_{i+1} V_{i+1}^H \times_{i+2} \cdots \times_d V_d^H$ and $\overline{\mathcal{Y}}^i_{\mathbf{k}} = \mathcal{Y}_{\mathbf{k}} \times_i V_i$. In particular, the Euclidean norm of $\overline{\mathbf{V}}_i^H \mathbf{r}$ equals to the Frobenius norm of the mode-i unfolding

$$(\mathcal{R}_i)_{(i)} = (\overline{\mathcal{C}}_i)_{(i)} - A_i (\overline{\mathcal{Y}}^i_{\mathbf{k}})_{(i)} - (\overline{\mathcal{Y}}^i_{\mathbf{k}})_{(i)} B_i, \tag{4.11}$$

where

$$\begin{aligned} B_i = &\sum_{j=1}^{i-1} I_{k_d} \otimes \cdots \otimes I_{k_{i+1}} \otimes I_{k_{i-1}} \otimes \cdots \otimes I_{k_{j+1}} \otimes {A_j}^{(k_j)} \otimes I_{k_{j-1}} \otimes \cdots \otimes I_{k_{i_1}} \\ &+ \sum_{j=i+1}^{d} I_{k_d} \otimes \cdots \otimes I_{k_{j+1}} \otimes {A_j}^{(k_j)} \otimes I_{k_{j-1}} \otimes \cdots \otimes I_{k_{i+1}} \otimes I_{k_{i-1}} \otimes \cdots \otimes I_{k_{i_1}}. \end{aligned} \tag{4.12}$$

Remark 4.2.2. *The matrix $(\mathcal{R}_i)_{(i)}$ is the residual of the Sylvester equation $A_i X - X B_i = (\overline{\mathcal{C}}_i)_{(i)}$, solved projecting A_i into the block rational Krylov subspace $\mathcal{Q}_{k_i}(A, C_i)$.*

Summarizing, we have the following corollary of Proposition 4.2.3.

Corollary 4.2.4. *The squared Euclidean norm of the residual $r(\mathbf{V}, A_1, \ldots, A_d, C)$ can be written as*

$$\|r(\mathbf{V}, A_1, \ldots, A_d, \mathbf{c})\|_2^2 = \sum_{i=1}^{d} \left\| (\mathcal{R}_i)_{(i)} \right\|_F^2 + \|\widehat{\mathbf{c}}\|_2^2,$$

where the remainder term $\widehat{\mathbf{c}}$ vanishes for $\mathbf{c} \in \operatorname{span}(\mathbf{V})$.

4.2.3 RHS in Tucker format

In this section we assume that the right-hand side $\mathcal{C}$ of (4.1) is given in Tucker format, generated by $[\![\mathcal{G}, U_1, \ldots, U_d]\!]$, with $\mathcal{G} \in \mathbb{C}^{b_i \times \cdots \times b_d}$ and $U_i \in \mathbb{C}^{n_i \times b_i}$ for each $i = 1, \ldots, d$. For each i, a Tucker representation of the tensor $\overline{\mathcal{C}}_i$ is generated by

$$[\![\mathcal{G}, V_1^H U_1, \ldots, V_{i-1}^H U_{i-1}, U_i, V_{i+1}^H U_{i+1} \ldots, V_d^H U_d]\!],$$

hence, from (4.5), we have that the matrix $(\overline{\mathcal{C}}_i)_{(i)}$ admits the low rank representation $(\overline{\mathcal{C}}_i)_{(i)} = U_i Z^H$, for an appropriate block vector Z. From Corollary 4.2.4, the convergence of projection methods based on tensorized block rational Krylov subspaces is related to the norms of the matrices $(\mathcal{R}_i)_{(i)}$. By Remark 4.2.2 we are implicitly solving the Sylvester equation $A_i X - X B_i = U_i Z^H$, by projecting A_i into the block rational Krylov subspace $\mathcal{Q}_{k_i}(A, C_i)$. Hence, the natural choice of the block vector for the construction of the ith block rational Krylov subspace is $C_i = U_i$.

Assume now we know $V_1, \ldots, V_d$ orthonormal basis for $\mathcal{Q}_{k_1}(A_1, U_1), \ldots, \mathcal{Q}_{k_d}(A_d, U_d)$, respectively, and the projected matrices $A_i^{(k_i)} = V_i^H A_i V_i$ and we want to solve the projected tensor equation (4.10). It is reasonable that the right-hand side $\mathcal{C}_{\mathbf{k}} \in \mathbb{C}^{b_1 k_1 \times \cdots \times b_d k_d}$, can be fully stored and the solution $\mathcal{Y}_{\mathbf{k}}$ of the projected equation can be computed by a direct method such as the one presented by Chan and Kressner in [30]. A Tucker decomposition of the approximate solution $\mathcal{X}_{\mathbf{k}}$ related with the tensorized block rational Krylov subspace is generated by $[\![\mathcal{Y}_{\mathbf{k}}, V_1, \ldots, V_d]\!]$.

4.2.4 RHS in tensor train format

The main advantage of having $\mathcal{C} \in \mathbb{C}^{n_1 \times \cdots \times n_d}$ in TT format is the possibility of handling more summands in the tensor Sylvester equation since the memory storage in this format increases only linearly with d. Clearly, in such a case it is necessary to produce an approximate solution tensor $\mathcal{X}_{\mathbf{k}}$ in TT format as well. Assume that the tensor $\mathcal{C}$ is represented in TT format with carriages $\{\mathcal{G}_1, \dots, \mathcal{G}_d\}$. For each i, a TT representation of the tensor $\overline{\mathcal{C}}_i$ is given by the carriages

$$\{\mathcal{G}_1 \times_1 V_1^H, \mathcal{G}_2 \times_2 V_2^H, \dots, \mathcal{G}_{i-1} \times_2 V_{i-1}^H, \mathcal{G}_i, \mathcal{G}_{i+1} \times_2 V_{i+1}^H, \dots, \mathcal{G}_d \times_2 V_d^H\}$$

and from (4.7) we have that the matrix $(\overline{\mathcal{C}}_i)_{(i)}$ admits the low rank representation

$$(\overline{\mathcal{C}}_i)_{(i)} = \begin{cases} \mathcal{G}_1 Z^H & \text{if } i = 1, \\ (\mathcal{G}_i)_{(2)} Z^H & \text{otherwise,} \end{cases}$$

for an appropriate block vector Z. With the same argument of the Tucker case, we have that the natural choice of the block vector for the construction of the ith block rational Krylov subspace is

$$C_i = \begin{cases} \mathcal{G}_1 & \text{if } i = 1, \\ (\mathcal{G}_i)_{(2)} & \text{otherwise.} \end{cases}$$

Assuming to know $V_1, \dots, V_d$ orthonormal bases for $\mathcal{Q}_{k_1}(A_1, C_1), \dots, \mathcal{Q}_{k_d}(A_d, C_d)$, respectively, and the projected matrices $A_i^{(k_i)} = V_i^H A_i V_i$, we want to solve the projected equation (4.10). We remark that in this case it is not guaranteed that the tensor $\mathcal{C}_{\mathbf{k}}$ can be fully stored since its size grows exponentially with d. A way to overcome this issue is to use an algorithm for the solution of the projected tensor Sylvester equation that keeps the solution in TT format, such as the AMEn algorithm described in [37].

4.3 Pole selection

In this section we derive techniques for pole selection, employing a representation of the residual that involves the poles of the block rational Krylov subspaces. Thanks to Corollary 4.2.4 and (4.11), the analysis can be reduced to the problem of minimizing the norms of the residuals $(\mathcal{R}_i)_{(i)}$ of the Sylvester equations $A_i X - X B_i = (\overline{\mathcal{C}}_i)_{(i)}$, solved projecting A_i into the block rational Krylov subspace $\mathcal{Q}_{k_i}(A, C_i)$. Since this work is devoted to studying the case of $\mathcal{C}$ in Tucker or tensor train format, we also assume that the matrices $(\overline{\mathcal{C}}_i)_{(i)}$ admit a low-rank representation $(\overline{\mathcal{C}}_i)_{(i)} = C_i Z_i^H$, as it has been shown in Sections 4.2.3 and 4.2.4. To represent the norm of the matrices $(\mathcal{R}_i)_{(i)}$ with a formulation that involves the chosen poles we can use a simplified version of Theorem 3.2.1.

Theorem 4.3.1. *Let $A \in \mathbb{C}^{n \times n}$, $B \in \mathbb{C}^{m \times m}$ be matrices with disjoint spectra and let $C \in \mathbb{C}^{n \times b}$ and $Z \in \mathbb{C}^{m \times b}$. Let $V \in \mathbb{C}^{n \times bk}$ be a matrix with orthonormal columns that spans $\mathcal{Q}_k(A, C, \boldsymbol{\xi}_k^{(A)})$ and let $A_k = V^H A V$. Assuming that A_k and B have disjoint spectra, let $X_k = V^H Y_k$ where Y_k is the solution of the Sylvester equation*

$$A_k Y_k - Y_k B = C^{(k)} Z^H,$$

where $C^{(k)} = V^H C$. Let $\chi \in \mathcal{P}_k(\mathbb{C}^{b \times b})$ be a monic block characteristic polynomial of A_k with respect to $C^{(k)}$. Define $R^G(x) = \frac{\chi(x)}{q(x)}$, where

$$q(x) = \prod_{\xi \in \boldsymbol{\xi}_k^{(A)}, \xi \neq \infty} (x - \xi).$$

Then the residual matrix is equal to

$$\left(R^G(A) \circ C\right) \left(R^{G^H}(B) \circ^{-1} Z\right)^H.$$

Proof. It is sufficient to consider Theorem 3.2.1 with $\boldsymbol{\xi}_k^{(B)}$ consisting of m infinity poles. In such a case, an orthonormal basis of $\mathcal{Q}_k(B^H, C_2, \boldsymbol{\xi}_k^{(B)})$ can be chosen as the identity matrix of size $m \times m$. □

From now on we assume that for each i, the first pole in $\boldsymbol{\xi}_i$ is equal to infinity, that is, the first block column of V_i is an orthonormal basis of the space spanned by the columns of C_i. The first consequence of this hypothesis is that the term $\widehat{\mathbf{c}}$ vanishes in the formulation of the residual given by Corollary 4.2.4, hence the convergence can be monitored just by the Frobenius norms of the matrices $(\mathcal{R}_i)_{(i)}$. Notice that, thanks to Theorem 4.3.1, we have

$$(\mathcal{R}_i)_{(i)} = \left(R_i^G(A_i) \circ C_i\right) \left(R_i^{G^H}(-B_i) \circ^{-1} Z_i\right)^H, \quad \text{with} \quad R_i^G(z) = \chi_i(z)/q_i(z), \tag{4.13}$$

where $\chi_i(z)$ is a monic block characteristic polynomial of $A_i^{(k_i)}$ with respect to $V_i^H C_i$ and

$$q_i(z) = \prod_{\xi \in \boldsymbol{\xi}_i, \xi \neq \infty} (z - \xi).$$

As it has been established in Section 3.2.2, to keep the Frobenius norm of (4.13) small it is sufficient to choose poles that minimize the norm of $R_i^G(z)^{-1}$ for every x in the field of values of $-B_i$. A way to approximatively minimize such norm is to adaptively choose the next pole for $\boldsymbol{\xi}_i$ accordingly with one of the techniques described in Section 3.2.2.

Remark 4.3.1. *The field of values of the matrix $-B_i$ is the sum of the field of values of the matrices $-A_j^{(k_j)}$, for $j \neq i$. In general, the determination $\mathbb{W}(-A_j^{(k_j)})$ is not easy. What we do in practice is to use the convex hull of the set obtained by taking the union of the eigenvalues of the matrices $-A_j^{(s)}$ for $s \leq k_j$ as an approximation of $\mathbb{W}(-A_j^{(k_j)})$.*

4.4 Computation of the residual

The explicit computation of the residual to monitor the convergence of the algorithm is usually expensive; to overcome this issue we can compute the norms of the matrices $(\mathcal{R}_i)_{(i)}$ and then recover the norm of the residual using the result of Corollary 4.2.4. If the last pole is equal to infinity, the norms of the partial residuals can be cheaply computed thanks to the following lemma.

Lemma 4.4.1. *Let $A \in \mathbb{C}^{n \times n}$, $B \in \mathbb{C}^{m \times m}$, $C \in \mathbb{C}^{n \times b}$, $Z \in \mathbb{C}^{m \times b}$ and let $\boldsymbol{\xi} = \{\xi_0, \ldots, \xi_{k+1}\} \subseteq \overline{\mathbb{C}}$ with $\xi_0 = \xi_{k+1} = \infty$ be a set of poles that are not eigenvalues of A. Let V_{k+1} be a block orthonormal basis of $\mathcal{Q}_{k+1}(A, C, \boldsymbol{\xi})$, and let $\underline{H}_k, \underline{K}_k$ be the matrices that satisfy the block rational Arnoldi relation given by Definition 2.2.1. Denoting by V_k the matrix obtained by removing from V_{k+1} the last b columns and by $\mathtt{R}_k$ the residual*

$$\mathtt{R}_k = AV_kY_k + V_kY_kB - CZ^H, \tag{4.14}$$

where Y_k solves

$$(V_k^H AV_k)Y_k + Y_kB - V_k^H CZ^H = 0, \tag{4.15}$$

we have

$$\mathtt{R}_k = V_{k+1}E_{k+1}E_{k+1}^H \underline{H}_k K_k^{-1} Y_k \quad \text{and} \quad \|\mathtt{R}_k\|_F = \left\|E_{k+1}^H \underline{H}_k K_k^{-1} Y_k\right\|_F.$$

where $E_{k+1} = [0, I_b]^H \in \mathbb{C}^{(k+1)b \times b}$, and K_k is the leading principal $bk \times bk$ submatrix of $\underline{K}_k$.

Proof. From (4.15) follows that

$$Y_kB = -(V_k^H AV_k)Y_k + V_k^H CZ^H,$$

hence (4.14) can be rewritten as

$$\mathtt{R}_k = AV_kY_k - V_k(V_k^H AV_k)Y_k + V_kV_k^H CZ^H - CZ^H.$$

Since $\xi_0 = \infty$, we have $C \in \mathcal{Q}_k(A, C, \boldsymbol{\xi})$, then $V_k V_k^H C = C$. In particular, we have

$$\mathtt{R}_k = AV_kY_k - V_k(V_k^H AV_k)Y_k.$$

Note now that since $\xi_k = \infty$, $V_{k+1}V_{k+1}^H AV_k = AV_k$, hence

$$\mathtt{R}_k = V_{k+1}\left(V_{k+1}^H A - \begin{bmatrix} V_k^H \\ 0 \end{bmatrix} A\right) V_kY_k = V_{k+1}E_{k+1}E_{k+1}^H V_{k+1}^H AV_kY_k. \tag{4.16}$$

Moreover, as shown in Section 2.2, we have that $AV_k = V_{k+1}\underline{H}_k K_k^{-1}$ and since the columns of V_{k+1} are orthonormal we have

$$\mathtt{R}_k = V_{k+1}E_{k+1}E_{k+1}^H \underline{H}_k K_k^{-1} Y_k \quad \text{and} \quad \|\mathtt{R}_k\|_F = \left\|E_{k+1}^H \underline{H}_k K_k^{-1} Y_k\right\|_F.$$

□

To avoid the multiplication by the (possibly large) matrices A_i, we can use block rational Krylov methods that start with a pole equal to infinity and after each step swap the last two poles guaranteeing that the last pole is always equal to infinity, as illustrated in Section 2.2.1.

Summarizing, if we perform a tensorized block rational Krylov method where, for each block rational Krylov subspace, we have $\xi_0 = \infty$ and we guarantee that the last pole is equal to infinity, we can write the norm of the residual as

$$\|r(\mathbf{V}, A_1, \ldots, A_d, C)\|_2 = \sqrt{\sum_{i=1}^{d} \left\|\mathcal{Y}_{\mathbf{k}} \times_i E_{k_i+1}^H \underline{H}_{k_i}^{(i)} (K_{k_i}^{(i)})^{-1}\right\|_F^2},$$

where $\mathcal{Y}_{\mathbf{k}}$ is the solution of the projected equation (4.10), for each i we have $E_{k_i+1} = [0, I_b]^H \in \mathbb{C}^{b(k_i+1)\times b}$, and $\underline{H}_{k_i}^{(i)}, K_{k_i}^{(i)}$ are generated by the block rational Arnoldi algorithm associated with $\mathcal{Q}_{k_i}(A_i, C_i)$.

4.5 Numerical experiments

In this section, we provide numerical results on the convergence of the presented algorithms, for the solution of tensor Sylvester equations with right-hand side represented in Tucker or tensor train format. The MATLAB code of the algorithms used for solving tensor Sylvester equations has been made freely available at `https://github.com/numpi/TBRK-Sylvester`.

Generalizing what was done in Section 3.3, as a test problem, we compute the approximate solution of the convection-diffusion partial differential equation

$$\begin{cases} -\epsilon\Delta u + \mathbf{w}\cdot\nabla u = f & \text{in } \Omega \\ u \equiv 0 & \text{on } \partial\Omega \end{cases}$$

on the d-dimensional hypercube $\Omega = [0,1]^d$, where $\epsilon > 0$ is the viscosity parameter and $\mathbf{w}$ is the convection vector. Assuming $\mathbf{w} = (\Phi_1(x_1), \Phi_2(x_2), \ldots, \Phi_d(x_d))$, discretizing the domain with a uniformly spaced grid with n points in each direction yields the tensor Sylvester equation

$$\mathcal{X} \times_1 (\epsilon A + \boldsymbol{\Phi}_1 B) + \cdots + \mathcal{X} \times_d (\epsilon A + \boldsymbol{\Phi}_d B) = \mathcal{F},$$

where

$$A = \frac{1}{h^2}\begin{bmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix} \quad \text{and} \quad B = \frac{1}{2h}\begin{bmatrix} 0 & 1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & 1 \\ & & -1 & 0 \end{bmatrix},$$

with $h = \frac{1}{n+1}$, are the discretizations of the operators Δ and ∇, respectively, by centered finite differences,

$$\mathbf{\Phi}_i = \begin{bmatrix} \Phi_i(h) & & & \\ & \Phi_i(2h) & & \\ & & \ddots & \\ & & & \Phi_i(nh) \end{bmatrix}$$

for $i = 1, \ldots, d$ and $\mathcal{F}$ is the tensor given by sampling f on the grid points. If the function f is a smooth multivariate function, the tensor $\mathcal{F}$ is numerically low-rank, that is, it can be approximated by a low multilinear or TT rank tensor, see [118]. Different choices of $\mathbf{w}$ are used in the experiments, in particular choosing $\mathbf{w}$ as the zero vector yields a Hermitian test problem, indicated as *Poisson equation* (in such a case we also set $\epsilon = 1$).

The numerical simulations have been run on a server with two Intel(R) Xeon(R) E5-2650v4 CPU running at 2.20 GHz and 256 GB of RAM, using MATLAB R2021a with the Intel(R) Math Kernel Library Version 2019.0.3. All the experiments are made in double precision, real arithmetic. In particular, if a nonreal pole is employed during a Krylov method, the subsequent is chosen as its conjugate allowing to keep the matrices and the tensors real; we refer the reader to [109] for a more complete discussion. This artificial addition of poles can produce Krylov subspaces of different dimensions. For the experiments several pole selection strategies are employed: we denote by `poly` the use of all poles equal to infinity and by `ext` the case in which the poles are chosen alternating 0 and infinity, moreover, we indicate by `ADM` and `sADM` the poles described in (3.26) and (3.28), respectively, that are computed using the same implementation employed in Section 3.3, which solves the involved maximization problems by maximizing the functions on a sampling of the boundary of the set described in Remark 4.3.1. For the computation of rational Krylov subspaces we employ the `rktoolbox` described in [19].

4.5.1 Tucker format

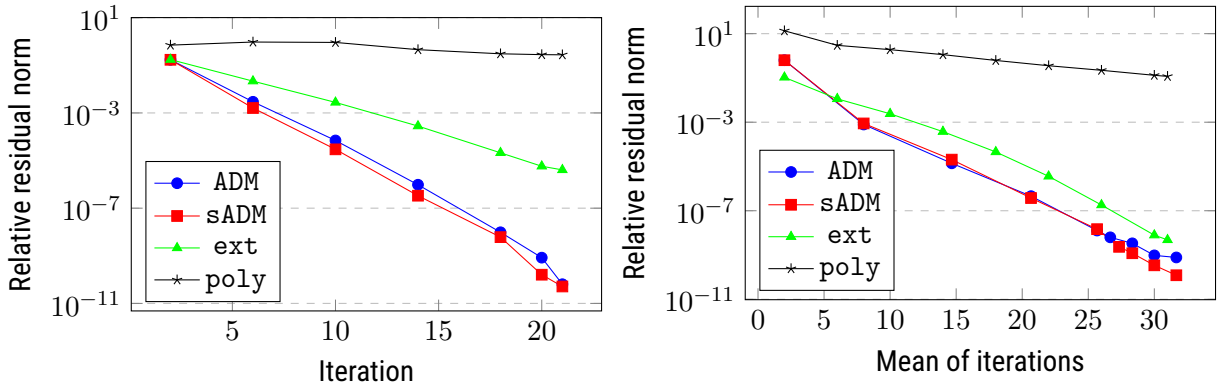
In this section, we provide numerical results for the case of right-hand side in Tucker format employing the algorithm described in Section 4.2.3, denoted by `Tuck-TBRK`. The projected tensor equations are solved by the algorithm developed in [30].

In Figure 4.1 we show the behavior of the relative norm of the residual by varying the number of Arnoldi iterations, for the solution of discretized convection-diffusion equations using different choices of poles; in particular in Figure 4.1a and Figure 4.1b we consider the problem with $\epsilon = 1$ and $\mathbf{w} = (0, 0, 0)$ and the non-Hermitian problem with $\epsilon = 0.1$ and $\mathbf{w} = (1 + \frac{(x_1+1)^2}{4}, 0, 0)$, respectively. For the latter test problem, the number of Arnoldi iterations needed to reach a relative norm of the residual less than 10^{-6} are shown in Table 4.1; the table also contains execution time of the algorithms and the total time needed to solve the projected equations. We observe that most of the computational timing is employed for solving the small-size equations, stressing the importance of a good pole selection strategy. We also notice that the procedure that employs polynomial Krylov subspaces has been stopped after 41 iteration without reaching the desired accuracy.

In Figure 4.2 the behavior of the relative norm of the residual is compared for different sizes of the discretization grid for solving a 3-dimensional Poisson equation. Note that, by the results developed in [118, Section 4.3], the minimum integer k such that there exists a tensor with multilinear rank $(b_1 k, b_2 k, b_3 k)$ that solves the discretized Poisson equation with a relative accuracy ϵ, satisfies

$$k \leq \frac{(\log(16\gamma)) \log(4\sqrt{3}/\epsilon)}{\pi^2},$$

where γ depends quadratically on n. Hence, we expect that a good pole selection strategy produces a logarithmical growth of the number of iterations in terms of n.



(a) Discretized Poisson equation with $d = 4$, $f = 1/((1 + x_1 + x_2)(1 + x_3 + x_4))$.

(b) Discretized convection-diffusion equation with $d = 3, f = 1/((1 + x_1 + x_2 + x_3))$, $\epsilon = 0.1$, $\mathbf{w} = (1 + \frac{(x_1+1)^2}{4}, 0, 0)$.

Figure 4.1: Behavior of the relative norm of the residual produced by solving discretized convection-diffusion equation employing `Tuck-TBRK` methods, comparing different choices of poles. The parameter n is set to 1024.

poles	iterations	time (s)	residual	time proj (s)
ADM	14 24 24	7.88	$4.55e-07$	6.10
sADM	14 24 24	7.84	$3.75e-07$	5.87
ext	26 26 26	24.59	$1.82e-07$	22.43
poly	41 41 41	196.61	$5.21e-02$	187.27

Table 4.1: Iterations (i.e., number of Arnoldi iterations performed in the three different Krylov subspaces) and time needed to reach a relative norm of the residual less than 10^{-6} for the solution of discretized convection-diffusion equation of dimension $d = 3$ and $n = 1024$, with $f = 1/((1 + x_1 + x_2 + x_3))$, $\epsilon = 0.1$ and $\mathbf{w} = (1 + \frac{(x_1+1)^2}{4}, 0, 0)$, employing `Tuck-TBRK` methods, with different choices of poles. The last column describes the total time needed for solving projected problems.

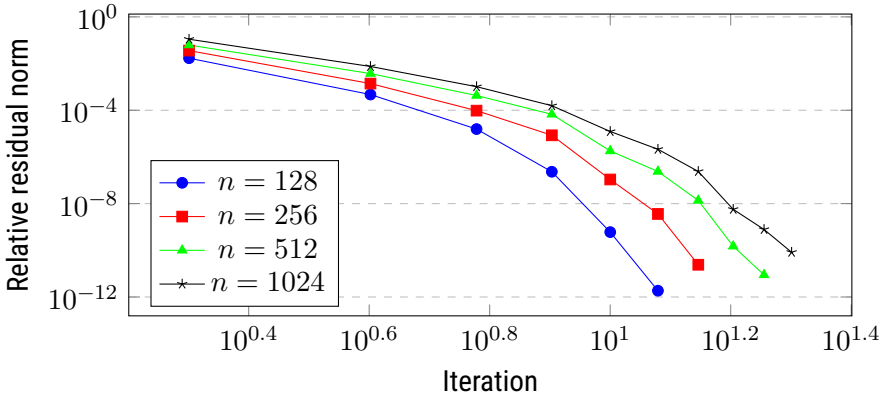
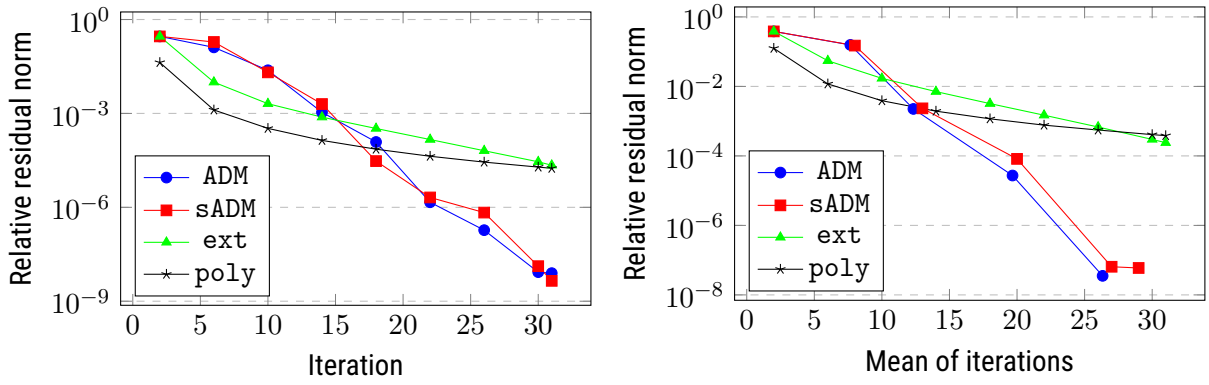


Figure 4.2: Behavior of the relative norm of the residual produced by solving the discretized Poisson equation of dimension $d = 3$ with $f = 1/(1 + x_1 + x_2 + x_3)$ employing `Tuck-TBRK` methods, with poles chosen accordingly to `ADM` for different sizes of the discretization grid.



(a) Discretized Poisson equation with $d = 6$.

(b) Discretized convection-diffusion equation with $d = 5$, $\epsilon = 0.1$, $\mathbf{w} = (1 + \frac{(x_1+1)^2}{4}, \frac{(1+x_2)}{2}, 0, 0, 0)$.

Figure 4.3: Behavior of the relative norm of the residual produced by solving discretized convection-diffusion equations with random right-hand side of TT rank $(2, 2, \ldots, 2)$, employing TT−TBRK methods, with different choices of poles. The parameter n is set to 4096.

4.5.2 Tensor train format

In this section, we provide numerical results for the case of right-hand side in tensor train format, employing the algorithm described in Section 4.2.4, denoted by TT−TBRK. We have implemented the TT−TBRK methods in MATLAB, using the TT-Toolbox version 2.2 [101] to manage tensors in TT format, in particular the projected problems are solved using the routine `amen_solve2`.

In Figure 4.3 we show the behavior of the relative norm of the residual by varying the number of Arnoldi iterations, for the solution of discretized convection-diffusion equations using different choices of poles. Despite the Tucker case, it seems that the method based on polynomial Krylov subspaces converges quite fast for the first iterations, however, if a high accuracy is required methods based on rational Krylov subspaces seem to be faster. In Figure 4.4 the behavior of the relative norm of the residual is compared for different sizes of the discretization grid for solving a 6-dimensional Poisson equation; as described in [118, Section 4.5.2], we expect that the number of iterations grows logarithmically with n. In Table 4.2 we compare the execution time of TT−TBRK and AMEn, to reach a relative norm of the residual less than 10^{-8} for the solution of a d-dimensional Poisson equation for different values of d. We remark that in the first two cases AMEn does not reach the required accuracy. In Table 4.3 we report the time and the number of Arnoldi iterations employed by TT−TBRK for the computation of the solution of high dimensional Poisson equations with a relative norm of the residual less than 10^{-6}. From the results, it appears that the number of iterations does not grow when the space dimension d increases. On the other hand for large values of d, the more than linear increase of the computational time is due to the pole selection strategy, which is based on the maximization of a rational function on the field of values of the matrices $B_i s$ defined in (4.12). To manage this trade-off between good poles and low computational time, other maximization strategies could be employed by exploiting, for instance, theoretical information about the field of values of B_i.

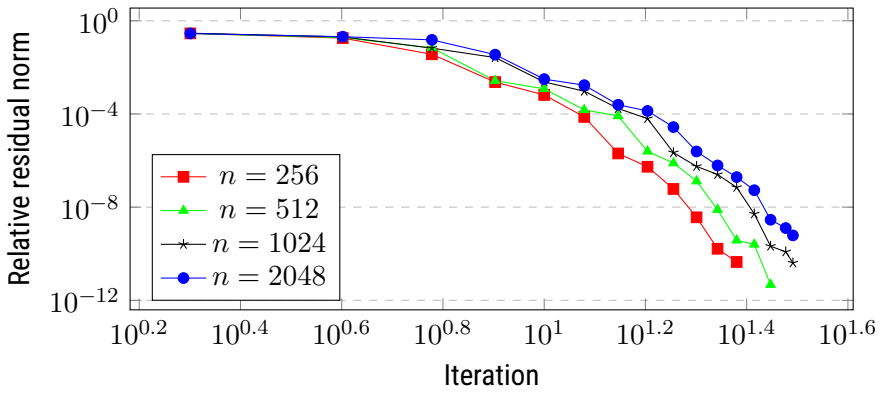


Figure 4.4: Behavior of the relative norm of the residual produced by solving the discretized Poisson equation of dimension $d = 6$ with random right-hand side of TT rank $(2, 2, \ldots, 2)$, employing TT−TBRK methods, with poles chosen accordingly to ADM for different sizes of the discretization grid.

	d	residual	time (s)
TT−TBRK	3	$5.42e-09$	20.27
AMEn	3	$3.06e-07$	2098.17
TT−TBRK	4	$5.73e-09$	52.34
AMEn	4	$4.39e-08$	5418.22
TT−TBRK	5	$6.31e-09$	74.59
AMEn	5	$5.24e-09$	4558.16

Table 4.2: Comparison of execution time and accuracy between TT−TBRK with poles chosen accordingly with ADM and AMEn to reach relative norm of the residual less than 10^{-8} for the solution of a d-dimensional Poisson equation for different values of d. The parameter n is set to 1024.

d	residual	Arnoldi iterations	time (s)	time AMEn (s)	time poles (s)
5	$1.79e-08$	26	13.21	11.23	0.84
10	$9.29e-07$	26	19.07	14.10	2.72
15	$4.59e-07$	26	158.75	33.30	121.88
20	$1.75e-07$	22	4163.33	40.38	4119.03

Table 4.3: Time, accuracy and number of Arnoldi iterations of TT−TBRK with poles chosen accordingly with ADM required to reach relative norm of the residual less than 10^{-6} for the solution of a d-dimensional Poisson equation for large values of d. The last two columns contain the time employed for solving all the projected problems by the AMEn method and the time needed for the computation of poles. The parameter n is set to 1024.

Chapter 5

Computing functions of Hermitian HSS Matrices

In earlier chapters, we employed block rational Krylov methods to address matrix equations. Moving forward, our focus shifts toward matrix functions. The material presented in this chapter is the outcome of collaboration with Daniel Kressner and Leonardo Robol, which has led to [25].

Consider a Hermitian matrix $A \in \mathbb{C}^{n\times n}$. When A is of moderate size, $f(A)$ can simply be computed according to its definition, via computing the spectral decomposition of A, or using a more specialized algorithm such as the scaling-and-squaring method for the matrix exponential [75]. These methods typically require $\mathcal{O}(n^2)$ memory and $\mathcal{O}(n^3)$ operations, and thus become infeasible for larger n. As we will extensively discuss in Chapter 6, if only the computation of $f(A)C$ for a block vector C is needed, block (rational) Krylov subspace methods are well suited when A is large and (data) sparse; see [66, 93] and the references therein.

The task of approximating the whole matrix function $f(A)$ for a large-scale matrix A is rather challenging and certainly requires additional assumptions on the data sparsity structure of A. For example, if A is a banded matrix *and* f can be well approximated by a low-degree polynomial on the spectrum of A, then $f(A)$ can also be well approximated by a banded matrix [17], leading to fast algorithms, such as the ones described in [33, 56, 105]. If, on the other hand, f does not admit good polynomial approximations then $f(A)$ usually does not admit a good approximation by a banded or, more generally, by a sparse matrix even when A is banded. Examples include the (inverse) square root or the sign function when A has eigenvalues that are close to zero relative to the width of the spectrum. For these examples, f still admits good *rational* approximations and the approximation of $f(A)$ can potentially be addressed using hierarchical low-rank techniques [70].

A matrix A is said to be *hierarchically off-diagonal low-rank* (HODLR) if it can be recursively block partitioned in a matrix with low-rank off-diagonal blocks, more specifically, there exists a block partitioning

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

such that A_{12} and A_{21} are of low rank and A_{11} and A_{22} are square matrices that can be recursively partitioned in the same way (until a minimal size of the diagonal blocks is reached). *Hierarchically semiseparable* (HSS) matrices additionally impose that the low-rank factors representing the off-diagonal blocks on the different levels of the recursion are nested; see Section 5.1 for the precise definition. Hierarchical matrices, such as HODLR and HSS, admit data-sparse representations and cover a wide variety of matrix structures, including banded matrices and rational functions thereof. In particular, they can be used to approximate $f(A)$ when A is banded or, more generally, HSS whenever f admits a good rational approximation [33]. This property has been exploited to develop iterate-and-truncate methods using hierarchical matrices [64, 85] as well as a divide-and-conquer procedure based on low-rank updates and rational Krylov subspaces [33]. While numerical experiments in [33] show that the latter method is often preferable in terms of efficiency,

it only exploits HODLR structure even when the matrix is HSS, and therefore does not fully benefit from the nestedness of low-rank factors in the HSS format.

To fully benefit from HSS structure, we will represent an HSS matrix in an unconventional way, via a telescopic decomposition. Such a decomposition was used by Levitt and Martinsson [90] to compute an HSS approximation of a matrix A from a few random matrix-vector products.[1] For the purpose of this work, we consider a more general *telescopic decomposition* (see Section 5.2), based on representing an HSS matrix A as

$$A = \mathbf{U}\widetilde{A}\mathbf{V}^H + \mathbf{D}, \tag{5.1}$$

where $\mathbf{U}$ and $\mathbf{V}$ are block diagonal, orthonormal matrices with tall and skinny diagonal blocks, $\mathbf{D}$ is a block diagonal matrix and $\widetilde{A}$ is a (smaller) square matrix recursively decomposed in the same fashion. This representation is not unique and includes the one from [90]. We also show how to convert such telescopic decompositions into the standard representation of HSS matrices.

The main contribution of this work is to design a new algorithm that returns a telescopic decomposition for an HSS approximation of $f(A)$, where A is a Hermitian HSS matrix and f admits a good rational approximation. Unlike the divide-and-conquer method described in [33], our algorithm fully exploits the nestedness relations of the HSS format and, in turn, only needs to build rational Krylov subspaces for small-sized matrices. This translates into reduced complexity and, in many cases, significantly lower execution times. Using telescopic decompositions, our new algorithm combines well with the method by Levitt and Martinsson. This combination allows one to extract an approximation to $f(A)$ from the product of A with a few random vectors.

Note that for most of Sections 5.1 to 5.2, we will not assume that A is Hermitian. Imposing Hermiticity would not simplify the exposition and our considerations on telescopic decompositions in Section 5.2 might be of independent interest. In Section 5.2.4, we will discuss the consequences of A being Hermitian.

5.1 Hierarchically semiseparable matrices

To define hierarchically semiseparable (HSS) matrices we need to introduce a way to recursively split row and column indices of a matrix. Given a vector of indices $I = [1, 2, \ldots, n]$, we use a perfect binary tree $\mathcal{T}$, called *cluster tree*, to define subsets of indices obtained by subdividing I. The root γ of the tree is associated with the full vector I; the rest of the tree is recursively defined in the following way: given a non-leaf node τ associated with an index vector I_τ, its children α, β, are associated with two vectors of consecutive indices I_α, I_β, such that I_τ is the concatenation of I_α and I_β. The depth of a node is defined as the distance from the root of the tree. The depth of the tree is denoted by L. We observe that $\mathcal{T}$ is uniquely defined by the index vectors associated with the nodes and, hence, cluster trees can be defined by simply specifying the indices associated with the leaf nodes. For each leaf node α, we use $|\alpha|$ to denote the length of the vector of indices associated with α. We assume that the leaf nodes are "small", that is, for some prescribed *threshold size* t it holds that $|\alpha| \leq t$ for every leaf α.

Given a matrix $A \in \mathbb{C}^{n\times n}$, we let $A_{\tau,\tau'}$ denote the submatrix of A obtained by selecting the row and column indices associated with nodes τ and τ', respectively. In particular, we will consider diagonal blocks ($\tau = \tau'$) and sub/supdiagonal blocks (τ and τ' are siblings, i.e., children of the same node). We are now ready to state the definition of an HSS matrix following [94, Section 3.3].

Definition 5.1.1. *Given a cluster tree $\mathcal{T}$ for the indices $[1, \ldots, n]$, a matrix $A \in \mathbb{C}^{n\times n}$ is called an HSS matrix of HSS rank r if:*

1. *for each pair of sibling nodes $\tau, \tau' \in \mathcal{T}$, there exist matrices $U_\tau^{(\mathrm{big})} \in \mathbb{C}^{|\tau|\times r}$, $V_{\tau'}^{(\mathrm{big})} \in \mathbb{C}^{|\tau'|\times r}$ with orthonormal columns and $\widetilde{A}_{\tau,\tau'} \in \mathbb{C}^{r\times r}$, such that*

$$A_{\tau,\tau'} = U_\tau^{(\mathrm{big})} \widetilde{A}_{\tau,\tau'} (V_{\tau'}^{(\mathrm{big})})^H.$$

[1] It is worth mentioning that another method for approximating HSS matrices from random matrix-vector products was recently presented by Halikias and Townsend [71], but this algorithm is not based on the telescopic decompositions considered in this work.

2. *for each non-leaf node $\tau \in \mathcal{T}$ with children α and β there exist $U_\tau, V_\tau \in \mathbb{C}^{2r\times r}$ with orthonormal columns, such that*

$$U_\tau^{(\mathrm{big})} = \begin{bmatrix} U_\alpha^{(\mathrm{big})} & \\ & U_\beta^{(\mathrm{big})} \end{bmatrix} U_\tau \qquad \text{and} \qquad V_\tau^{(\mathrm{big})} = \begin{bmatrix} V_\alpha^{(\mathrm{big})} & \\ & V_\beta^{(\mathrm{big})} \end{bmatrix} V_\tau. \tag{5.2}$$

Remark 5.1.1. *Point 1 of Definition 5.1.1 implies that every sub/supdiagonal block $A_{\tau,\tau'}$ has rank at most r. To optimize storage, one could allow for different values of r for different τ, τ'. To simplify the description, we will work with a constant rank bound r in this paper. On the other hand, our software implementation allows for non-constant ranks.*

Definition 5.1.1 corresponds to the usual way of storing an HSS matrix A [97,129]. For each leaf node τ, the matrices $U_\tau := U_\tau^{(\mathrm{big})}$, $V_\tau := V_\tau^{(\mathrm{big})}$ and for each non-leaf node τ the matrices U_τ, V_τ from (5.2) are stored. The latter pair of matrices are usually called *translation operators*. The nestedness relation (5.2) allows us to recursively recover $U_\alpha^{(\mathrm{big})}$ and $V_\alpha^{(\mathrm{big})}$ for any $\alpha \in \mathcal{T}$ from the (small) matrices U_τ, V_τ. We only need to additionally store the $r \times r$ matrices $\widetilde{A}_{\tau,\tau'}$ for every pair of sibling nodes τ, τ' and the small diagonal blocks $A_{\tau,\tau}$ (of size at most t) for every leaf τ in order to recover the whole matrix A. To see this, let α and β denote the children of the root γ. Then

$$A = A_{\gamma,\gamma} = \begin{bmatrix} A_{\alpha,\alpha} & U_\alpha^{(\mathrm{big})} \widetilde{A}_{\alpha,\beta} (V_\beta^{(\mathrm{big})})^H, \\ U_\beta^{(\mathrm{big})} \widetilde{A}_{\beta,\alpha} (V_\alpha^{(\mathrm{big})})^H, & A_{\beta,\beta} \end{bmatrix}. \tag{5.3}$$

If α and β are non-leaf nodes, the matrices $A_{\alpha,\alpha}$ and $A_{\beta,\beta}$ can be recursively recovered in the same fashion. Otherwise, if α and β are leaves, these matrices are stored explicitly. In summary, the matrices

$$\{U_\tau, V_\tau : \tau \in \mathcal{T}\}, \quad \{\widetilde{A}_{\tau,\tau'} : \tau, \tau' \,\text{sibling nodes}\}, \quad \{A_{\alpha,\alpha} : \alpha \,\text{leaf node}\} \tag{5.4}$$

define a data-sparse representation of an HSS matrix A.

Remark 5.1.2. *There is no need to impose orthonormality on U_τ, V_τ in the representation* (5.4)*. The orthogonality properties required in Definition 5.1.1 can always be ensured by the orthogonalization procedure described in [94, Section 4.2], without changing the matrix A represented by* (5.4) *through the recursion* (5.3)*.*

5.1.1 Block diagonal matrices and depth reduction

To simplify notation, we make use of the following form of block diagonal matrices. Let $\{C_\tau\}_{\tau\in\mathcal{T}}$ be a set of matrices for a cluster tree $\mathcal{T}$. Then

$$\mathbf{C}^{(\ell)} := \mathrm{blkdiag}(C_\tau : \tau \in \mathcal{T}, \mathrm{depth}(\tau) = \ell) \tag{5.5}$$

denotes the block diagonal matrix with the diagonal blocks consisting of all matrices C_τ for which τ has depth equal to ℓ.

For example, given the data-sparse representation (5.4) of an HSS matrix A, the matrices $\mathbf{U}^{(L)}$ and $\mathbf{V}^{(L)}$ are block diagonal matrices containing the orthonormal matrices U_α and V_α, respectively, for every leaf $\alpha \in \mathcal{T}$ as diagonal blocks. When considering the product

$$\widehat{A} := \left(\mathbf{U}^{(L)}\right)^H A \mathbf{V}^{(L)},$$

the representation (5.4) is affected as follows:

$$\begin{aligned} &\{U_\tau, V_\tau : \mathrm{depth}(\tau) \le L-1\}, \quad &&\{U_\alpha^H U_\alpha, V_\alpha^H V_\alpha : \alpha \,\text{leaf node}\}, \\ &\{\widetilde{A}_{\tau,\tau'} : \tau, \tau' \,\text{sibling nodes}\}, &&\{A_{\alpha,\alpha} : \alpha \,\text{leaf node}\}. \end{aligned} \tag{5.6}$$

This representation allows us to reconstruct the matrix $\widehat{A}$ by a recursion analogous to (5.3). Since $U_\alpha^H U_\alpha = I$ and $V_\alpha^H V_\alpha = I$, the orthogonality properties of Definition 5.1.1 are satisfied as well. However, the size of $\widehat{A}$ is $2^L m$, which is generally different from n and requires the cluster tree $\mathcal{T}$ to be adjusted accordingly.

Definition 5.1.2. *Given integers m and L, consider the index vector $[1, 2, \ldots, 2^L m]$. Then $\mathcal{T}_m^{(L)}$ denotes the corresponding* balanced *cluster tree of depth L, having leaves associated with $[(i-1)m+1, \ldots, im]$ for $i = 1, \ldots, 2^L$.*

In particular, the matrix $\widehat{A}$ defined above is an HSS matrix associated with the cluster tree $\mathcal{T}_m^{(L)}$. When dropping the leaves, one obtains the balanced cluster tree $\mathcal{T}_{2m}^{(L-1)}$ of depth $L-1$. The matrix $\widehat{A}$ is still an HSS matrix associated with $\mathcal{T}_{2m}^{(L-1)}$ but its parametrization (5.6) reduces to

$$\{U_\tau, V_\tau : \tau \in \mathcal{T}_{2m}^{(L-1)}\}, \quad \{\widetilde{A}_{\tau,\tau'} : \tau, \tau' \text{ siblings in } \mathcal{T}_{2m}^{(L-1)}\}, \quad \{A_{\tau,\tau} : \alpha \text{ leaf in } \mathcal{T}_{2m}^{(L-1)}\},$$

which matches the form of (5.4).

5.2 Telescopic decompositions

To develop a randomized algorithm for recovering an HSS matrix from matrix-vector products, Levitt and Martinsson [90] replaced the classical data-sparse representation (5.4) with a certain type of telescopic decomposition. As we will see in Section 5.2.2, the data-sparse representation (5.4) is directly linked to a similar but different type of telescopic decomposition. In Section 5.2.1, we introduce a general class of telescopic decompositions that includes both types. We also discuss how to convert between these types of telescopic decompositions.

5.2.1 A general telescopic decomposition

We start with a generalization of the construction from [90], recursively defining a telescopic decomposition.

Definition 5.2.1. *Let $\mathcal{T}$ be a cluster tree of depth L for the indices $[1, \ldots, n]$. A matrix $A \in \mathbb{C}^{n\times n}$ is in* telescopic decomposition *of telescopic rank r if there are real matrices*

$$\{U_\tau, V_\tau : \tau \in \mathcal{T},\, 1 \le \operatorname{depth}(\tau)\} \quad \text{and} \quad \{D_\tau : \tau \in \mathcal{T}\}$$

for brevity denoted by $\{U_\tau, V_\tau, D_\tau\}_{\tau\in\mathcal{T}}$ or simply $\{U_\tau, V_\tau, D_\tau\}$, with the following properties:

1. *D_τ is of size $|\tau| \times |\tau|$ if $\operatorname{depth}(\tau) = L$ and $2r \times 2r$ otherwise;*
2. *U_τ, V_τ have orthonormal columns; they are of size $|\tau| \times r$ if $\operatorname{depth}(\tau) = L$ and $2r \times r$ otherwise;*
3. *if $L = 0$ (i.e., $\mathcal{T}$ consists only of the root γ) then $A = D_\gamma$;*
4. *if $L \ge 1$ then*

$$A = \mathbf{D}^{(L)} + \mathbf{U}^{(L)} A^{(L-1)} (\mathbf{V}^{(L)})^H, \tag{5.7}$$

where $\mathbf{U}^{(L)}, \mathbf{V}^{(L)}, \mathbf{D}^{(L)}$ are the block diagonal matrices defined by U_τ, V_τ, D_τ as in (5.5)*, and the matrix*

$$A^{(L-1)} := (\mathbf{U}^{(L)})^H (A - \mathbf{D}^{(L)}) \mathbf{V}^{(L)} \tag{5.8}$$

has the telescopic decomposition $\{U_\tau, V_\tau, D_\tau\}_{\tau\in\mathcal{T}_{2r}^{(L-1)}}$, where $\mathcal{T}_{2r}^{(L-1)}$ denotes a balanced cluster tree of depth $L-1$ (see Definition 5.1.2).

As we will see in the following, Definition 5.2.1 offers significant freedom in the choice of diagonal blocks D_τ, giving rise to different types of telescopic decompositions. A simple procedure to explicitly reconstruct the matrix A from a telescopic decomposition is described in Algorithm 2.

Algorithm 2 Recovering A from a telescopic decomposition.

Input: $\{U_\tau, V_\tau, D_\tau\}_{\tau\in\mathcal{T}}$ telescopic decomposition of A for cluster tree $\mathcal{T}$ of depth L,
Output: A

1: $A \leftarrow D_\gamma$ where γ is the root of $\mathcal{T}$
2: **for** $\ell = 1, \dots, L$ **do**
3: $\quad A \leftarrow \mathbf{D}^{(\ell)} + \mathbf{U}^{(\ell)} A (\mathbf{V}^{(\ell)})^H$ ▷ with $\mathbf{D}^{(\ell)}$, $\mathbf{U}^{(\ell)}$, $\mathbf{V}^{(\ell)}$ defined as in (5.5)
4: **end for**

5.2.2 From HSS matrices to telescopic decompositions

The following proposition provides a construction that turns any HSS matrix into a telescopic decomposition of the same rank.

Proposition 5.2.1. *Let*

$$\{U_\tau, V_\tau, \widetilde{A}_{\tau,\tau'} : \tau, \tau' \text{ sibling nodes}\} \quad \text{and} \quad \{A_{\tau,\tau} : \tau \text{ leaf node}\}$$

be the data-sparse representation defining an HSS matrix A of HSS rank r for a cluster tree $\mathcal{T}$ of depth L. For each node $\tau \in \mathcal{T}$, define

$$D_\tau := \begin{cases} A_{\tau,\tau} & \text{if } \tau \text{ is a leaf node} \\ \begin{bmatrix} 0 & \widetilde{A}_{\alpha,\beta} \\ \widetilde{A}_{\beta,\alpha} & 0 \end{bmatrix} & \text{if } \tau \text{ has children } \alpha \text{ and } \beta. \end{cases}$$

Then $\{U_\tau, V_\tau, D_\tau\}$ is a telescopic decomposition of A of telescopic rank r associated with $\mathcal{T}$.

Proof. We proceed by induction on L. If $L = 0$, the tree only consists of the root γ and the statement trivially holds because of $D_\gamma = A$.

Let us now assume that $L \geq 1$ and consider the matrices $U_\tau^{(\mathrm{big})}$ and $V_\tau^{(\mathrm{big})}$ from Definition 5.1.1. Because of the recursion (5.2), we have

$$\mathbf{U}^{(L)} (\mathbf{U}^{(L)})^H \begin{bmatrix} U_\alpha^{(\mathrm{big})} & \\ & U_\beta^{(\mathrm{big})} \end{bmatrix} = \begin{bmatrix} U_\alpha^{(\mathrm{big})} & \\ & U_\beta^{(\mathrm{big})} \end{bmatrix},$$

and

$$\mathbf{V}^{(L)} (\mathbf{V}^{(L)})^H \begin{bmatrix} V_\alpha^{(\mathrm{big})} & \\ & V_\beta^{(\mathrm{big})} \end{bmatrix} = \begin{bmatrix} V_\alpha^{(\mathrm{big})} & \\ & V_\beta^{(\mathrm{big})} \end{bmatrix},$$

where α, β are the children of the root γ and $\mathbf{U}^{(L)}$, $\mathbf{V}^{(L)}$ are the block diagonal matrices employed in Definition 5.2.1. Combined with the HSS recursion (5.3), this shows that

$$A - \begin{bmatrix} A_{\alpha,\alpha} & \\ & A_{\beta,\beta} \end{bmatrix} = \mathbf{U}^{(L)} (\mathbf{U}^{(L)})^H \left(A - \begin{bmatrix} A_{\alpha,\alpha} & \\ & A_{\beta,\beta} \end{bmatrix} \right) \mathbf{V}^{(L)} (\mathbf{V}^{(L)})^H. \tag{5.9}$$

Noting that $A_{\alpha,\alpha}$ and $A_{\beta,\beta}$ are HSS matrices associated with trees of depth $L-1$, we can apply induction to conclude that they are both in telescopic decomposition. This means that relations of the form (5.7)–(5.8) hold for both matrices or, equivalently,

$$\begin{bmatrix} A_{\alpha,\alpha} & \\ & A_{\beta,\beta} \end{bmatrix} - \mathbf{D}^{(L)} = \mathbf{U}^{(L)} (\mathbf{U}^{(L)})^H \left(\begin{bmatrix} A_{\alpha,\alpha} & \\ & A_{\beta,\beta} \end{bmatrix} - \mathbf{D}^{(L)} \right) \mathbf{V}^{(L)} (\mathbf{V}^{(L)})^H.$$

Adding this equation to (5.9) gives

$$A = \mathbf{D}^{(L)} + \mathbf{U}^{(L)} \, A^{(L-1)} \, (\mathbf{V}^{(L)})^H, \quad \text{where} \quad A^{(L-1)} = (\mathbf{U}^{(L)})^H \big(A - \mathbf{D}^{(L)} \big) \mathbf{V}^{(L)}.$$

Together with the discussion from Section 5.1.1, it follows that $A^{(L-1)}$ is an HSS matrix associated with $\mathcal{T}_{2r}^{(L-1)}$ and defined by the data-sparse representation

$$\{U_\tau, V_\tau : \tau \in \mathcal{T}_{2r}^{(L-1)}\}, \quad \{\widetilde{A}_{\tau,\tau'} : \tau, \tau' \text{ siblings in } \mathcal{T}_{2r}^{(L-1)}\}, \quad \{A_{\tau,\tau}^{(L-1)} : \tau \text{ leaf in } \mathcal{T}_{2r}^{(L-1)}\}.$$

If $A_{\tau,\tau}^{(L-1)} = D_\tau$, this completes the proof by induction: the assumptions of this theorem are satisfied for the level-$(L-1)$ HSS matrix $A^{(L-1)}$ and thus $\{U_\tau, V_\tau, D_\tau\}_{\text{depth}(\tau)\leq L-1}$ is a telescopic decomposition of $A^{(L-1)}$. In turn, all conditions of Definition 5.2.1 are satisfied and $\{U_\tau, V_\tau, D_\tau\}$ is a telescopic decomposition for A.

It remains to show that $A_{\tau,\tau}^{(L-1)} = D_\tau$ holds for any node τ of depth $L-1$. For this purpose, let α, β denote its children, which are leaves in $\mathcal{T}$. Using that $D_\alpha = A_{\alpha,\alpha}$ and $D_\beta = A_{\beta,\beta}$, we indeed obtain that

$$A_{\tau,\tau}^{(L-1)} = \begin{bmatrix} U_\alpha^H & \\ & U_\beta^H \end{bmatrix} \left(A_{\tau,\tau} - \begin{bmatrix} D_\alpha & \\ & D_\beta \end{bmatrix} \right) \begin{bmatrix} V_\alpha & \\ & V_\beta \end{bmatrix} = \begin{bmatrix} 0 & U_\alpha^H A_{\alpha,\beta} V_\beta \\ U_\beta^H A_{\beta,\alpha} V_\alpha & 0 \end{bmatrix} = D_\tau.$$

where the last equality follows from Definition 5.1.1. □

The proof of Proposition 5.2.1 shows that the matrix $A^{(L-1)}$ generated by the telescopic decomposition $\{U_\tau, V_\tau, D_\tau\}_{\tau\in\mathcal{T}_{2r}^{(L-1)}}$ satisfies $(A^{(L-1)})_{\tau,\tau} = D_\tau$ for every leaf τ of $\mathcal{T}_{2r}^{(L-1)}$. Letting $\mathcal{T}_{2r}^{(\ell)}$ denote a balanced cluster tree of depth $\ell \leq L-1$, we can apply this property recursively and obtain

$$(A^{(\ell)})_{\tau,\tau} = D_\tau \quad \text{for every leaf } \tau \text{ of } \mathcal{T}_{2r}^{(\ell)} \text{ for all } 1 \leq \ell \leq L-1, \tag{5.10}$$

where $A^{(\ell)}$ denotes the matrix generated by the telescopic decomposition $\{U_\tau, V_\tau, D_\tau\}_{\tau\in\mathcal{T}_{2r}^{(\ell)}}$. It follows from Proposition 5.2.1 and Proposition 5.2.2 below that telescopic decompositions with this property are in a simple one-to-one correspondence with HSS matrices, which justifies the following definition.

Definition 5.2.2. *A telescopic decomposition $\{U_\tau, V_\tau, D_\tau\}_{\tau\in\mathcal{T}}$ of a matrix A is called* standard *if $D_\tau = A_{\tau,\tau}$ for each leaf node τ and* (5.10) *holds.*

Proposition 5.2.2. *Let $\mathcal{T}$ be a cluster tree of depth L and let A be the matrix generated by a standard telescopic decomposition $\{U_\tau, V_\tau, D_\tau\}_{\tau\in\mathcal{T}}$ of telescopic rank r. Then, for each nonleaf node τ with children α, β, there exist matrices $\widetilde{A}_{\alpha,\beta}, \widetilde{A}_{\beta,\alpha} \in \mathbb{C}^{kb\times kb}$ such that*

$$D_\tau = \begin{bmatrix} 0 & \widetilde{A}_{\alpha,\beta} \\ \widetilde{A}_{\beta,\alpha} & 0 \end{bmatrix}. \tag{5.11}$$

Moreover, A is an HSS matrix of HSS rank r with the data-sparse representation

$$\{U_\tau, V_\tau : \tau \in \mathcal{T}\}, \quad \{\widetilde{A}_{\tau,\tau'} : \tau, \tau' \text{ sibling nodes}\}, \quad \{A_{\alpha,\alpha} : \alpha \text{ leaf node}\}.$$

Proof. We proceed by induction on L. For $L = 0$, the result trivially holds. Suppose now that $L \geq 1$. By Definition 5.2.1, the matrix generated by the standard telescopic decomposition $\{U_\tau, V_\tau, D_\tau\}_{\tau\in\mathcal{T}_{2r}^{(L-1)}}$ is the matrix $A^{(L-1)}$ defined in (5.8). Therefore, if τ is a node of depth $L-1$ with children α and β, it follows from (5.10) that

$$D_\tau = A_{\tau,\tau}^{(L-1)} = \begin{bmatrix} 0 & U_\alpha^H A_{\alpha,\beta} V_\beta \\ U_\beta^H A_{\beta,\alpha} V_\alpha & 0 \end{bmatrix} = \begin{bmatrix} 0 & \widetilde{A}_{\alpha,\beta} \\ \widetilde{A}_{\beta,\alpha} & 0 \end{bmatrix},$$

where we set $\widetilde{A}_{\alpha,\beta} = U_\alpha^H A_{\alpha,\beta} V_\beta$ and $\widetilde{A}_{\beta,\alpha} = U_\beta^H A_{\beta,\alpha} V_\alpha$. This proves (5.11).

It remains to establish the HSS property of A, that is, Point 1 of Definition 5.1.1 (note that Point 2 is satisfied by construction). Combining (5.11) with the telescopic relation (5.7), we obtain that

$$A_{\tau,\tau} = \begin{bmatrix} D_\alpha & \\ & D_\beta \end{bmatrix} + \begin{bmatrix} U_\alpha & \\ & U_\beta \end{bmatrix} D_\tau \begin{bmatrix} V_\alpha & \\ & V_\beta \end{bmatrix} = \begin{bmatrix} D_\alpha & U_\alpha \widetilde{A}_{\alpha,\beta} V_\beta^H \\ U_\beta \widetilde{A}_{\beta,\alpha} V_\alpha^H & D_\beta \end{bmatrix}.$$

In particular, $A_{\alpha,\beta} = U_\alpha \widetilde{A}_{\alpha,\beta} V_\alpha^H$, which establishes Point 1 of Definition 5.1.1 for two sibling leaves $\tau = \alpha$, $\tau' = \beta$. To show the corresponding property for two siblings τ and τ' of depth $\ell < L$, let $\alpha_1, \ldots, \alpha_{2^{L-\ell}}$ and $\alpha'_1, \ldots, \alpha'_{2^{L-\ell}}$ denote the leaf nodes in the corresponding subtrees. Using (5.7), the proof is completed by noting that

$$\begin{aligned} A_{\tau,\tau'} &= \operatorname{blkdiag}(U_{\alpha_i}) A^{(L-1)}_{\tau,\tau'} \operatorname{blkdiag}(V_{\alpha'_i})^H \\ &= \operatorname{blkdiag}(U_{\alpha_i}) \operatorname{blkdiag}(U_{\alpha_i})^H U^{(\mathrm{big})}_\tau \widetilde{A}_{\tau,\tau'} \big(V^{(\mathrm{big})}_{\tau'}\big)^H \operatorname{blkdiag}(V_{\alpha'_i}) \operatorname{blkdiag}(V_{\alpha'_i})^H \\ &= U^{(\mathrm{big})}_\tau \widetilde{A}_{\tau,\tau'} (V^{(\mathrm{big})}_{\tau'})^H, \end{aligned}$$

where the second equality uses induction: $A^{(L-1)}$ is an HSS matrix and satisfies a relation of the form (5.3) for the parent γ of τ, τ'. □

5.2.3 Converting a general telescopic decomposition into a standard one

In the following, we describe a procedure that turns an arbitrary telescopic decomposition $\{U_\tau, V_\tau, D_\tau\}$ of a matrix A into a standard telescopic decomposition $\{U_\tau, V_\tau, C_\tau\}$. By the results of Section 5.2.2, this implies the equivalence between HSS matrices of HSS rank r and matrices that admit a telescopic decomposition of telescopic rank r. For this purpose, it is crucial to understand how we can recover the principal submatrices $A_{\alpha,\alpha}$ for leaf nodes α, since these matrices correspond to the matrices C_α in the standard telescopic decomposition.

Proposition 5.2.3. *For a cluster tree $\mathcal{T}$ of depth L, let A be a matrix in telescopic decomposition $\{U_\tau, V_\tau, D_\tau\}$ of telescopic rank r. Then the following holds:*

1. *if $L = 0$ (i.e., $\mathcal{T}$ consists only of the root γ) then $A_{\gamma,\gamma} = D_\gamma$;*
2. *if $L \geq 1$, any pair of sibling leaf nodes α, β with parent τ satisfies*

$$\begin{bmatrix} A_{\alpha,\alpha} & \\ & A_{\beta,\beta} \end{bmatrix} = \begin{bmatrix} D_\alpha & \\ & D_\beta \end{bmatrix} + \begin{bmatrix} U_\alpha \big[(A^{(L-1)}_{\tau,\tau})_{1,1}\big] V_\alpha^H & \\ & U_\beta \big[(A^{(L-1)}_{\tau,\tau})_{2,2}\big] V_\beta^H \end{bmatrix}, \tag{5.12}$$

with the matrix $A^{(L-1)}$ from Definition 5.2.1, and $(A^{(L-1)}_{\tau,\tau})_{1,1}$ and $(A^{(L-1)}_{\tau,\tau})_{2,2}$ denoting the (1,1) and (2,2) diagonal blocks of $A^{(L-1)}_{\tau,\tau} \in \mathbb{C}^{2r \times 2r}$, respectively.

Proof. Point 1 follows directly from the definition of D_γ. To prove Point 2, we observe that (5.7) implies

$$\begin{bmatrix} A_{\alpha,\alpha} & * \\ * & A_{\beta,\beta} \end{bmatrix} = A_{\tau,\tau} = \begin{bmatrix} D_\alpha & \\ & D_\beta \end{bmatrix} + \begin{bmatrix} U_\alpha & \\ & U_\beta \end{bmatrix} A^{(L-1)}_{\tau,\tau} \begin{bmatrix} V_\alpha^H & \\ & V_\beta^H \end{bmatrix}.$$

Therefore, taking the diagonal blocks concludes the proof. □

The previous proposition combined with the fact that a telescopic decomposition of the matrix $A^{(L-1)}$ employed in (5.12) is given by $\{U_\tau, V_\tau, D_\tau\}_{\operatorname{depth}(\tau) \leq L-1}$ (see Definiton 5.2.1), results in a practical way to compute the matrices $A_{\alpha,\alpha}$ for all leaves α; see Algorithm 3.

To satisfy condition (5.10) of a standard telescopic decomposition on the leaf level, we need to set

$$C_\alpha := A_{\alpha,\alpha}$$

for each leaf node α. Moreover, if $L \geq 1$, for each node τ of depth $L-1$ the matrix C_τ is given by $A^{(L-1)}_{\tau,\tau}$, where $A^{(L-1)}$ is now defined as

$$A^{(L-1)} := (\mathbf{U}^{(L)})^H \left(A - \mathbf{C}^{(L)}\right) \mathbf{V}^{(L)} = (\mathbf{U}^{(L)})^H \left(A - \mathbf{D}^{(L)}\right) \mathbf{V}^{(L)} + (\mathbf{U}^{(L)})^H \left(\mathbf{D}^{(L)} - \mathbf{C}^{(L)}\right) \mathbf{V}^{(L)},$$

Algorithm 3 Computation of principal submatrices of A given in telescopic decomposition

Input: Matrix A in telescopic decomposition $\{U_\tau, V_\tau, D_\tau\}$ for cluster tree $\mathcal{T}$ of depth L
Output: $\{A_{\alpha,\alpha} : \alpha \text{ leaf node}\}$

$\widehat{A}_{\gamma,\gamma} \leftarrow D_\gamma$ for root γ of $\mathcal{T}$
for $\ell = 0, \ldots, L-1$ **do**
  for each node τ of depth ℓ **do**
    Denoting by α, β the children of τ and defining $(\widehat{A}_{\tau,\tau})_{1,1}$, $(\widehat{A}_{\tau,\tau})_{2,2}$ as in Proposition 5.2.3
    $\begin{bmatrix} \widehat{A}_{\alpha,\alpha} & \\ & \widehat{A}_{\beta,\beta} \end{bmatrix} \leftarrow \begin{bmatrix} D_\alpha & \\ & D_\beta \end{bmatrix} + \begin{bmatrix} U_\alpha \left[(\widehat{A}_{\tau,\tau})_{1,1}\right] V_\alpha^H & \\ & U_\beta \left[(\widehat{A}_{\tau,\tau})_{2,2}\right] V_\beta^H \end{bmatrix}$
  end for
end for
for each leaf node α **do**
  $A_{\alpha,\alpha} \leftarrow \widehat{A}_{\alpha,\alpha}$
end for

with the block diagonal matrices $\mathbf{U}^{(L)}, \mathbf{V}^{(L)}, \mathbf{D}^{(L)}$ and $\mathbf{C}^{(L)}$ defined from $\{U_\tau\}, \{V_\tau\}, \{D_\tau\}$ and $\{C_\tau\}$, respectively, according to (5.5). By Definition 5.2.1, the matrix $(\mathbf{U}^{(L)})^H \left(A - \mathbf{D}^{(L)}\right) \mathbf{V}^{(L)}$ is generated by the telescopic decomposition $\{U_\tau, V_\tau, D_\tau\}_{\mathrm{depth}(\tau) \le L-1}$, hence defining

$$\widehat{D}_\tau := \begin{cases} D_\tau - \begin{bmatrix} U_\alpha^H (D_\alpha - C_\alpha) V_\alpha & \\ & U_\beta^H (D_\beta - C_\beta) V_\beta \end{bmatrix} & \text{if } \tau \text{ has depth } L-1 \text{ and children } \alpha, \beta; \\ D_\tau & \text{otherwise;} \end{cases}$$

the matrix $A^{(L-1)}$ is generated by the telescopic decomposition $\{U_\tau, V_\tau, \widehat{D}_\tau\}_{\mathrm{depth}(\tau) \le L-1}$. Therefore a standard decomposition of A can be computed by iterating Algorithm 3, as summarized in Algorithm 4. In particular, the computational complexity of transforming a telescopic decomposition into a standard one is $\mathcal{O}(r^3 2^L)$ where r is the telescopic rank of A. Assuming the threshold size and the telescopic rank to be constant, this shows linear complexity in the size of A.

Algorithm 4 Computation of a standard telescopic decomposition of A given in telescopic factors.

Input: Matrix A in telescopic decomposition $\{U_\tau, V_\tau, D_\tau\}$ for cluster tree $\mathcal{T}$ of depth L
Output: Standard telescopic decomposition $\{U_\tau, V_\tau, C_\tau\}$ of A

$\{C_\alpha : \alpha \text{ leaf node}\} \leftarrow$ Algorithm 3 applied to $\{U_\tau, V_\tau, D_\tau\}$
for each node τ **do**
  $\widehat{D}_\tau \leftarrow D_\tau$
end for
for $\ell = L-1, \ldots, 0$ **do**
  for each node τ of depth ℓ **do**
    Denoting by α, β the children of τ
    $\widehat{D}_\tau \leftarrow \widehat{D}_\tau - \begin{bmatrix} U_\alpha^H (\widehat{D}_\alpha - C_\alpha) V_\alpha & \\ & U_\beta^H (\widehat{D}_\beta - C_\beta) V_\beta \end{bmatrix}$
  end for
  $\{C_\tau : \tau \in \mathcal{T} \text{ of depth } \ell\} \leftarrow$ Algorithm 3 applied to $\{U_\tau, V_\tau, \widehat{D}_\tau\}_{\mathrm{depth}(\tau) \le l}$
end for

5.2.4 Hermitian telescopic decompositions

For a Hermitian matrix, the definition of a telescopic decomposition can be adjusted to reflect symmetry.

Definition 5.2.3. *A telescopic decomposition $\{U_\tau, V_\tau, D_\tau\}_{\tau\in\mathcal{T}}$ is said to be* Hermitian *if $V_\tau = U_\tau$ and $D_\tau^H = D_\tau$ hold for every $\tau \in \mathcal{T}$. In analogy to the non-Hermitian case, we employ the term* standard *if $D_\tau = A_{\tau,\tau}$ for each leaf node τ and* (5.10) *is satisfied. For simplicity, a Hermitian telescopic decomposition is denoted by $\{U_\tau, D_\tau\}_{\tau\in\mathcal{T}}$, avoiding the repetition of U_τ.*

If A is Hermitian and has HSS rank r, there exist data-sparse representations of the form (5.4), for which $U_\tau = V_\tau$ have r columns and $\widetilde{A}^H_{\tau,\tau'} = \widetilde{A}_{\tau',\tau}$ holds for each pair of sibling nodes τ, τ' (see [94, Section 4.1]). Therefore, Proposition 5.2.1 implies that A admits a Hermitian telescopic decomposition of rank r. We also recall that the procedure described in Section 5.2.3 converts a telescopic decomposition $\{U_\tau, V_\tau, D_\tau\}$ into a standard one, without changing the matrices U_τ, V_τ. In turn, the same procedure can be employed to convert a Hermitian telescopic decomposition into a standard Hermitian telescopic decomposition.

5.3 Computing telescopic decompositions for functions of Hermitian HSS matrices

If A is a Hermitian HSS matrix with spectrum contained in $[\lambda_{\min}, \lambda_{\max}]$ and the function f is analytic on $[\lambda_{\min}, \lambda_{\max}]$, then $f(A)$ can usually be well approximated by an HSS matrix. While this has been observed before [33, Section 3], it is nontrivial to develop an algorithm that fully exploits this property. In this section, we derive such an algorithm that computes a telescopic decomposition for an HSS approximation of $f(A)$ starting from a standard Hermitian telescopic decomposition of A. If needed, this can be converted into a standard telescopic decomposition, employing the results of Section 5.2.3, and therefore into an HSS data-sparse representation (5.4). If f is a rational function of a certain degree, we show that such an approximation is exact and, otherwise, the approximation error is bounded using a rational approximation of f on $[\lambda_{\min}, \lambda_{\max}]$.

We will apply Theorem 2.4.1 recursively to telescopic decompositions. In order to do so conveniently, we slightly loosen our assumptions on a standard telescopic decomposition. Given a cluster tree $\mathcal{T}$ associated with $[1, \dots, n]$ we assume that a matrix $A \in \mathbb{C}^{n\times n}$ can be written as

$$A = \widetilde{\mathbf{D}}^{(L)} + \mathbf{Z}^{(L)} A^{(L-1)} (\mathbf{Z}^{(L)})^H, \tag{5.13}$$

where:

- $A^{(L-1)}$ admits a standard Hermitian telescopic decomposition $\{U_\tau, D_\tau\}_{\tau\in\mathcal{T}_{2r}^{(L-1)}}$;
- $\widetilde{\mathbf{D}}^{(L)} = \mathrm{blkdiag}(\widetilde{D}_\tau \ : \ \tau \text{ leafnode in } \mathcal{T})$, and $\widetilde{D}_\tau = A_{\tau\tau}$;
- $\mathbf{Z}^{(L)} = \mathrm{blkdiag}(Z_\tau \ : \ \tau \text{ leafnode in } \mathcal{T})$, with $Z_\tau \in \mathbb{R}^{|\tau|\times r}$.

The key difference to assuming that A has a standard Hermitian telescopic decomposition is that no orthogonality is enforced on $\mathbf{Z}^{(L)}$, the factors on the leaf level. The key advantage of (5.13) is that it remains unaffected when multiplying with certain block diagonal matrices. The following proposition additionally shows how to move one level up.

Proposition 5.3.1. *Let A be an $n\times n$ matrix admitting the decomposition* (5.13)*, and $\mathbf{W}^{(L)} = \mathrm{blkdiag}(W_\tau : \tau \text{ leaf node})$ with $W_\tau \in \mathbb{C}^{|\tau|\times m}$. Then,*

$$(\mathbf{W}^{(L)})^H A \mathbf{W}^{(L)} = \widetilde{\mathbf{D}}^{(L-1)} + \mathbf{Z}^{(L-1)} A^{(L-2)} (\mathbf{Z}^{(L-1)})^H,$$

with the block diagonal matrices

$$\widetilde{\mathbf{D}}^{(L-1)} = \mathrm{blkdiag}(\widetilde{D}_\tau : \mathrm{depth}(\tau) = L-1), \quad \mathbf{Z}^{(L-1)} = \mathrm{blkdiag}(Z_\tau : \mathrm{depth}(\tau) = L-1)$$

containing the diagonal blocks

$$Z_\tau = \begin{bmatrix} W_\alpha^H Z_\beta & \\ & W_\alpha^H Z_\beta \end{bmatrix} U_\tau \in \mathbb{C}^{2m\times r},$$
$$\widetilde{D}_\tau = \begin{bmatrix} W_\alpha^H \widetilde{D}_\alpha W_\alpha & \\ & W_\beta^H \widetilde{D}_\beta W_\beta \end{bmatrix} + \begin{bmatrix} W_\alpha^H Z_\alpha & \\ & W_\beta^H Z_\beta \end{bmatrix} D_\tau \begin{bmatrix} W_\alpha^H Z_\alpha & \\ & W_\beta^H Z_\beta \end{bmatrix}^H \in \mathbb{C}^{2m\times 2m},$$

and the matrix $\mathbf{A}^{(L-2)}$ generated by the standard telescopic decomposition $\{U_\tau, D_\tau\}_{\tau\in\mathcal{T}_{2r}^{(L-2)}}$, where the matrices U_τ and D_τ stem from the telescopic decomposition of $A^{(L-1)}$. Moreover, $\widetilde{D}_\tau = A_{\tau,\tau}$ holds for each leaf node $\tau \in \mathcal{T}_{2m}^{(L-1)}$.

Proof. Considering (5.13) and applying the telescopic decomposition (5.7) of the matrix $A^{(L-1)}$ we get

$$\begin{aligned}(\mathbf{W}^{(L)})^H A\mathbf{W}^{(L)} &= (\mathbf{W}^{(L)})^H \left(\widetilde{\mathbf{D}}^{(L)} + \mathbf{Z}^{(L)}(\mathbf{D}^{(L-1)} + \mathbf{U}^{(L-1)}A^{(L-2)}(\mathbf{U}^{(L-1)}))^H(\mathbf{Z}^{(L)})^H\right)\mathbf{W}^{(L)}\\ &= \widetilde{\mathbf{D}}^{(L-1)} + \mathbf{Z}^{(L-1)}A^{(L-2)}(\mathbf{Z}^{(L-1)})^H\end{aligned}$$

with $\mathbf{D}^{(L-1)}, \mathbf{U}^{(L-1)}$ defined from $\{D_\tau\}, \{U_\tau\}$, in accordance with (5.5). Moreover, because $\{U_\tau, D_\tau\}_{\tau\in\mathcal{T}_{2r}^{(L-1)}}$ is a standard decomposition of $A^{(L-1)}$, the relation $\widetilde{D}_\tau = A_{\tau,\tau}$ holds for each leaf node $\tau \in \mathcal{T}_{2m}^{(L-1)}$. □

To approximate $f(A)$ for a matrix A admitting the decomposition (5.13), we use the construction of Theorem 2.4.1 with $B = \widetilde{\mathbf{D}}^{(L)}$ and $C = \mathbf{Z}^{(L)}$, which yields the approximation

$$f(A) \approx f(\widetilde{\mathbf{D}}^{(L)}) + \mathbf{W}^{(L)}\left[f\big((\mathbf{W}^{(L)})^H A\mathbf{W}^{(L)}\big) - f\big((\mathbf{W}^{(L)})^H\widetilde{\mathbf{D}}^{(L)}\mathbf{W}^{(L)}\big)\right](\mathbf{W}^{(L)})^H, \qquad (5.14)$$

where $\mathbf{W}^{(L)}$ is an orthogonal basis for $\mathcal{Q}_k(\widetilde{\mathbf{D}}^{(L)}, \mathbf{Z}^{(L)}, \boldsymbol{\xi}_k)$. Note that the three evaluations of f are all well defined because $\widetilde{\mathbf{D}}^{(L)}$ contains diagonal blocks of A and $(\mathbf{W}^{(L)})^H\widetilde{\mathbf{D}}^{(L)}\mathbf{W}^{(L)}$, $(\mathbf{W}^{(L)})^H A\mathbf{W}^{(L)}$ are orthogonal compressions. By eigenvalue interlacing, the spectra of these three matrices are contained in $[\lambda_{\min}, \lambda_{\max}]$. We now make two observations:

(i) Proposition 2.3.3 implies that the $n \times 2^L rk$ matrix $\mathbf{W}^{(L)}$ takes the form

$$\mathbf{W}^{(L)} = \mathrm{blkdiag}(W_\tau \ : \ \tau \text{ leaf of } \mathcal{T}),$$

with $W_\tau \in \mathbb{C}^{|\tau|\times rk}$ orthonormal basis of $\mathcal{Q}_k(\widetilde{D}_\tau, Z_\tau, \boldsymbol{\xi}_k)$.

(ii) Proposition 5.3.1 implies that $B^{(L-1)} := (\mathbf{W}^{(L)})^H A\mathbf{W}^{(L)}$ admits the decomposition

$$B^{(L-1)} = \widetilde{\mathbf{D}}^{(L-1)} + \mathbf{Z}^{(L-1)}A^{(L-2)}(\mathbf{Z}^{(L-1)})^H. \qquad (5.15)$$

In (5.14), the function f needs to be evaluated for three matrices. This requires low computational effort for $\widetilde{\mathbf{D}}^{(L)}$ and $(\mathbf{W}^{(L)})^H\widetilde{\mathbf{D}}^{(L)}\mathbf{W}^{(L)}$ because these matrices are block diagonal with small diagonal blocks, for which the evaluation of f is computed explicitly. The expensive part is the evaluation of f for $B^{(L-1)} = (\mathbf{W}^{(L)})^H A\mathbf{W}^{(L)}$. For this purpose, we use the decomposition (5.15) and apply the approximation (5.14) again:

$$f(B^{(L-1)}) \approx f(\widetilde{\mathbf{D}}^{(L-1)}) + \mathbf{W}^{(L-1)}\left[f(B^{(L-2)}) - f((\mathbf{W}^{(L-1)})^H\widetilde{\mathbf{D}}^{(L-1)}\mathbf{W}^{(L-1)})\right](\mathbf{W}^{(L-1)})^H,$$

where $B^{(L-2)} := (\mathbf{W}^{(L-1)})^H B^{(L-1)}\mathbf{W}^{(L-1)}$ and

$$\mathbf{W}^{(L-1)} = \mathrm{blkdiag}(W_\tau \ : \ \tau \text{ leaf of } \mathcal{T}_{2rk}^{(L-1)}), \text{ with } W_\tau \text{ orthonormal basis of } \mathcal{Q}_k(\widetilde{D}_\tau, Z_\tau, \boldsymbol{\xi}_k).$$

Since $\widetilde{\mathbf{D}}^{(L-1)}$ contains diagonal blocks of $B^{(L-1)}$, the three evaluations of f are again well defined.

The procedure described above is repeated recursively until reaching a tree of depth 0, and at that point one simply computes the matrix function of the corresponding dense matrix of size $2rk \times 2rk$ explicitly. The ith step of the recursive procedure proceeds as follows: one assumes a decomposition of the form

$$B^{(L-i)} = \widetilde{\mathbf{D}}^{(L-i)} + \mathbf{Z}^{(L-i)} A^{(L-i-1)} (\mathbf{Z}^{(L-i)})^H, \tag{5.16}$$

where $\widetilde{\mathbf{D}}^{(L-i)}$, $\mathbf{Z}^{(L-i)}$ are block diagonal matrices defined from $\{\widetilde{D}_\tau\}$, $\{Z_\tau\}$, in accordance with (5.5), and $A^{(L-i-1)}$ admits a standard Hermitian telescopic decomposition $\{U_\tau, D_\tau\}_{\tau \in \mathcal{T}_{2r}^{(L-i-1)}}$. Then $f(B^{(L-i)})$ is approximated by

$$f(\widetilde{\mathbf{D}}^{(L-i)}) + \mathbf{W}^{(L-i)} \left[f(B^{(L-i-1)}) - f((\mathbf{W}^{(L-i)})^H \widetilde{\mathbf{D}}^{(L-i)} \mathbf{W}^{(L-i)}) \right] (\mathbf{W}^{(L-i)})^H, \tag{5.17}$$

where $B^{(L-i-1)} := (\mathbf{W}^{(L-i)})^H B^{(L-i)} \mathbf{W}^{(L-i)}$ and

$$\mathbf{W}^{(L-i)} = \mathrm{blkdiag}(W_\tau \,:\, \tau \text{ leaf of } \mathcal{T}_{2rk}^{(L-i)}), \text{ with } W_\tau \text{ orthonormal basis of } \mathcal{Q}_k(\widetilde{D}_\tau, Z_\tau, \boldsymbol{\xi}_k),$$

explicitly computing $f(\widetilde{\mathbf{D}}^{(L-i)})$, $f((\mathbf{W}^{(L-i)})^H \widetilde{\mathbf{D}}^{(L-i)} \mathbf{W}^{(L-i)})$ and recursively approximating $f(B^{(L-i-1)})$. In particular, the procedure can be iterated considering the decomposition

$$B^{(L-i-1)} = \widetilde{\mathbf{D}}^{(L-i-1)} + \mathbf{Z}^{(L-i-1)} A^{(L-i-2)} (\mathbf{Z}^{(L-i-1)})^H,$$

given by Proposition 5.3.1.

Assuming that A admits a standard (Hermitian) telescopic decomposition $\{U_\tau, D_\tau\}$, the described procedure starts by taking $Z_\tau = U_\tau$ and $\widetilde{D}_\tau = D_\tau$ for each leaf node τ. It results in a telescopic decomposition for an approximation of $f(A)$, with generators $\{W_\tau, C_\tau\}$; the generators W_τ are defined by the rational Krylov subspaces constructed throughout the process, whereas C_τ takes the form

$$C_\tau := \begin{cases} f(\widetilde{D}_\tau) & \text{if } \tau \text{ is the root node} \\ f(\widetilde{D}_\tau) - W_\tau f(W_\tau^H \widetilde{D}_\tau W_\tau) W_\tau^H & \text{otherwise.} \end{cases}$$

The procedure is summarized in Algorithm 5.

Let us emphasize that even if the decomposition(5.13) is used in the course of the algorithm, the final result has a Hermitian telescopic decomposition in the sense of Definition 5.2.3. The non-orthogonal factors $\mathbf{Z}^{(\ell)}$ are only needed to represent intermediate stages.

To discuss the complexity of Algorithm 5, let r denote the HSS/telescopic rank of A, assume that $n = 2^L t$, with the threshold size $t \sim kr$, and that the cluster tree is balanced: $\mathcal{T} = \mathcal{T}_t^{(L)}$. On the leaf level L, the computation of all W_τ and C_τ requires $\mathcal{O}(2^L(t^3 + krt^2)) = \mathcal{O}(nk^2r^2)$ operations. On level $\ell < L$, the complexity is $\mathcal{O}(2^\ell k^3 r^3) = \mathcal{O}(2^{-(L-\ell)} nk^2r^2)$. This gives a total complexity of

$$\mathcal{O}(nk^2r^2), \tag{5.18}$$

which is linear in n *if* both r and k are considered constant.

We conclude this section with a result that bounds the approximation error of Algorithm 5 by the rational approximation error of f on $[\lambda_{\min}, \lambda_{\max}]$.

Theorem 5.3.2. *Let A be a Hermitian HSS matrix associated with a cluster tree $\mathcal{T}$ of depth L and let $\boldsymbol{\xi}_k = \{\xi_0, \ldots, \xi_{k-1}\} \subseteq \overline{\mathbb{C}}$, be a sequence of poles closed under complex conjugation. Let f be a function analytic on an interval $[\lambda_{\min}, \lambda_{\max}]$ containing the eigenvalues of A. Letting $E(f)$ denote the difference between $f(A)$ and the ouput of Algorithm 5 applied to a standard decomposition of A, it holds that*

$$\|E(f)\|_2 \leq 4L \min_{r \in \mathcal{P}_k/q_k} \|f - r\|_\infty .$$

Algorithm 5 Computation of $f(A)$ for Hermitian HSS matrix A in telescopic decomposition.

Input: $\{U_\tau, D_\tau\}$ standard Hermitian telescopic decomposition of matrix A, function f, sequence of poles $\boldsymbol{\xi}_k = \{\xi_0, \ldots, \xi_{k-1}\} \subseteq \overline{\mathbb{C}}$ closed under complex conjugation

Output: $\{W_\tau, C_\tau\}$ telescopic factorization of an approximation to $f(A)$

for $\ell = L, L-1, \ldots, 0$ **do**
  for each node τ of depth ℓ **do**
    if $\ell = L$ **then**
      $Z_\tau \leftarrow U_\tau$
      $\widetilde{D}_\tau \leftarrow D_\tau$
    else
      Let α and β be the children of τ
      $Z_\tau \leftarrow \begin{bmatrix} W_\alpha^H Z_\alpha & \\ & W_\beta^H Z_\beta \end{bmatrix} U_\tau$
      $\widetilde{D}_\tau \leftarrow \begin{bmatrix} W_\alpha^H \widetilde{D}_\alpha W_\alpha & \\ & W_\beta^H \widetilde{D}_\beta W_\beta \end{bmatrix} + \begin{bmatrix} W_\alpha^H Z_\alpha & \\ & W_\beta^H Z_\beta \end{bmatrix} D_\tau \begin{bmatrix} Z_\alpha^H W_\alpha & \\ & Z_\beta^H W_\beta \end{bmatrix}$
    end if
    if $\ell = 0$ **then**
      $C_\tau \leftarrow f(\widetilde{D}_\tau)$
    else
      $W_\tau \leftarrow$ orthonormal basis of $\mathcal{Q}_k(\widetilde{D}_\tau, Z_\tau, \boldsymbol{\xi}_k)$
      $C_\tau \leftarrow f(\widetilde{D}_\tau) - W_\tau f(W_\tau^H \widetilde{D}_\tau W_\tau) W_\tau^H$
    end if
  end for
end for

Proof. Let $\{W_\tau, C_\tau\}_{\tau \in \mathcal{T}}$ be the telescopic decomposition returned by Algorithm 5 applied to a standard Hermitian telescopic decomposition $\{U_\tau, D_\tau\}_{\tau \in \mathcal{T}}$. For each $1 \le i \le L-1$ let $B^{(L-i)}$ be the matrices defined in (5.16), and for each leaf node $\tau \in \mathcal{T}_{2rk}^{(L-i)}$, let $\widetilde{D}_\tau = B_{\tau,\tau}^{(L-i)}$. Moreover, to streamline the notation, we let $B^{(L)} := A$ and $\widetilde{D}_\tau := D_\tau$ for each leaf node $\tau \in \mathcal{T}$. For each i, let $E^{(L-i)}(f)$ be the difference between $f(B^{(L-i)})$ and its approximation (5.17). Since $[\lambda_{\min}, \lambda_{\max}]$ contains the eigenvalues of $B^{(L-i)}$ and $\widetilde{D}_\tau$ for each τ, Theorem 2.4.1 implies

$$\left\| E^{(L-i)}(f) \right\|_2 \le 4 \min_{r \in \mathcal{P}_k/q_k} \|f - r\|_\infty . \tag{5.19}$$

Denoting by $F^{(L-i)}$ the matrix generated by the Hermitian telescopic decomposition $\{W_\tau, C_\tau\}_{\tau \in \mathcal{T}_{2rk}^{L-i}}$ for $0 \le i \le L$, we have

$$f(B^{(L-i)}) - F^{(L-i)} = \begin{cases} 0 & \text{if } i = L; \\ f(B^{(L-i)}) - \mathbf{C}^{(L-i)} - \mathbf{W}^{(L-i)} F^{(L-i-1)} (\mathbf{W}^{(L-i-1)})^H & \text{otherwise,} \end{cases}$$

where $\mathbf{C}^{(L-i)}$, $\mathbf{W}^{(L-i)}$ are the block diagonal matrices defined by $\{C_\tau\}$, $\{W_\tau\}$, in accordance with (5.5). For $i < L$ we observe that

$$f(B^{(L-i)}) - \mathbf{C}^{(L-i)} = \mathbf{W}^{(L-i)} f(B^{(L-i-1)}) (\mathbf{W}^{(L-i)})^H + E^{(L-i)}(f),$$

and, hence,

$$f(B^{(L-i)}) - F^{(L-i)} = \mathbf{W}^{(L-i)} \big(f(B^{(L-i-1)}) - F^{(L-i-1)} \big) (\mathbf{W}^{(L-i)})^H + E^{(L-i)}(f).$$

Using that that the matrices $\mathbf{W}^{(L-i)}$ have orthonormal columns, this implies

$$\|E(f)\|_2 = \left\| f(B^{(L)}) - F^{(L)} \right\|_2 \le \sum_{i=0}^{L-1} \left\| E^{(L-i)}(f) \right\|_2 ,$$

which concludes the proof after applying the inequality (5.19). □

5.4 Pole selection

Theorem 5.3.2 shows that the choice of poles in the rational Krylov subspaces $\mathcal{Q}_k(\widetilde{D}_\tau, Z_\tau, \boldsymbol{\xi}_k)$ is critical to the convergence of Algorithm 5. Normally, repeated poles are preferred to reduce the cost of solving the shifted linear systems needed for constructing a basis of the subspace [67]. However, such considerations do not apply to Algorithm 5; solving linear systems with the small matrix $\widetilde{D}_\tau$ (shifted by a pole) is cheap.

By Theorem 5.3.2, if f is an analytic function on an interval $[a, b]$ that contains the eigenvalues of A and the sequence of poles $\boldsymbol{\xi}_k$ satisfies

$$\min_{r \in \mathcal{P}_k/q_k} \|f - r\|_\infty \leq \frac{\epsilon}{4L} \tag{5.20}$$

then Algorithm 5 returns an approximation of $f(A)$ within an error bounded by a user-specified tolerance $\epsilon > 0$. In the following, we describe explicit pole selection strategies that ensure (5.20) for two important classes of functions. For general f, general rational approximation methods, like the AAA algorithm [100], can be used to select the poles.

5.4.1 Exponential function

In the context of the matrix exponential, it is not uncommon to use polynomial approximations, that is, all poles are infinite. However, the corresponding (polynomial) Krylov subspace methods often converge poorly when the spectrum is wide, that is, $a \ll b$; see [12, 76] for theoretical results. As their computational overhead is small in our setting, it is preferable to use rational approximations/Krylov subspaces. Assuming $b \leq 0$ (which can always be attained by shifting the matrix), it is well known [62] that for every k there exists a sequence of poles $\boldsymbol{\xi}_k$ and a (universal) constant C such that

$$\min_{r \in \mathcal{P}_k/q_k} \|f - r\|_\infty \leq C K_e^{-k}, \quad K_e \approx 9.289.$$

In turn, this means that it suffices to choose

$$k \geq \log\left(4LC\epsilon^{-1}\right) / \log(K_e)$$

such that Algorithm 5 applied to a Hermitian negative semi-definite HSS matrix A, returns an approximation with an error ϵ. In particular, note that these estimates are independent of the width of the spectrum. Following (5.18), this gives a complexity of

$$\mathcal{O}(n(\log\log n + \log \epsilon^{-1})^2 r^2),$$

where r is the telescopic/HSS rank of A. For example, this implies that the fixed-accuracy approximation to the exponential of *any* tridiagonal Hermitian negative semi-definite matrix A has nearly linear complexity $\mathcal{O}(n(\log\log n)^2)$. We are not aware of any other algorithm that can achieve this.

5.4.2 Markov functions

Following the exposition in [10], we discuss pole selection for Markov functions, i.e., functions that can be represented as

$$f(z) = \int_\alpha^\beta \frac{\mathrm{d}\mu(x)}{z - x} \tag{5.21}$$

for some positive measure $\mu(x)$ and $-\infty \leq \alpha < \beta < \infty$. Important examples of functions in this class are

$$\frac{\log(1+z)}{z} = \int_{-\infty}^{-1} \frac{-1/x}{z - x}\,\mathrm{d}x \quad \text{and} \quad z^\gamma = \frac{\sin(\pi\gamma)}{\pi} \int_{-\infty}^{0} \frac{|x|^\gamma}{z - x}\,\mathrm{d}x,$$

with $-1 < \gamma < 0$.

Now, let f be a Markov function (5.21) and let A be a Hermitian HSS matrix whose eigenvalues are enclosed in an interval $[a, b]$ with $a > \beta$. The quasi-optimal rational approximation of f has been discussed in, e.g., [12, Section 6.2], which for every k provides a sequence of poles $\boldsymbol{\xi}_k$ such that

$$\min_{r \in \mathcal{P}_k/q_k} \|f - r\|_\infty \leq 4 \|f\|_\infty \exp\left(-k \frac{\pi^2}{\log(16(b-\beta)/(a-\beta))}\right).$$

Hence, to achieve a relative accuracy ϵ in Algorithm 5, one can choose

$$k \geq \log\left(\frac{16L}{\epsilon}\right) \frac{\log(16(b-\beta)/(a-\beta))}{\pi^2}, \tag{5.22}$$

Assuming a polynomial growth of $(b-\beta)/(a-\beta)$ (that depends on the condition number of the matrix $A - \beta I$) with respect to n, the complexity of the algorithm for Markov functions is, according to (5.18), given by

$$\mathcal{O}(n \log^2 n (\log\log n + \log \epsilon^{-1} + \log \|f\|_\infty)^2 r^2).$$

5.5 Numerical experiments

We have implemented Algorithm 5 in MATLAB and have made the code freely accessible at `https://github.com/numpi/HSS-matfun`; this implementation will be denoted by `TelFun` in the following. In our implementation, we allow for variable HSS/telescopic ranks (see Remark 5.1.1) and employ deflation criteria in the computation of orthonormal bases for rational Krylov subspaces, removing vectors that after the orthogonalization step have a norm smaller than a prescribed tolerance, proportional to the required accuracy. The threshold size t of the employed HSS matrices is fixed at 256. In our experiments, all standard operations with HSS matrices, such as matrix-vector products, have been performed using the hm-toolbox [97]. In the tables presented in this section, columns with the caption "err" denote the relative error in the Frobenius norm, compared with the result computed by a standard dense solver. Columns with the caption "time" report the observed execution time in seconds. All experiments have been executed on a server with two Intel(R) Xeon(R) E5-2650v4 CPU running at 2.20 GHz and 256 GB of RAM, using MATLAB R2021a with the Intel(R) Math Kernel Library Version 2019.0.3.

The main competitor, denoted by `CKM`, is the algorithm developed in [33], in which the authors use the HSS structure of A to perform a divide and conquer method for the computation of $f(A)$. The algorithm computes rational Krylov subspaces associated with (possibly large) HSS matrices, exploiting the structure in solving linear systems. The algorithm can also monitor the variation of the norm of the solution when a new pole is employed; this quantity can be used to stop the procedure if the desired accuracy is reached. We utilized an implementation of this algorithm available at `https://github.com/Alice94/MatrixFunctions-Banded-HSS`.

5.5.1 Computation of the inverse

Algorithm 5 is an attractive method for computing the inverse of a Hermitian positive definite HSS matrix. By Theorem 5.3.2, this algorithm returns the exact inverse (at least in exact arithmetic) when employing only one zero pole. We have tested `TelFun` in this situation for two different matrices. In Table 5.1, we report the results for the inversion of the discretized Laplacian, that is,

$$A = -\frac{1}{h^2} \begin{bmatrix} -2 & 1 & & \\ 1 & \ddots & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & -2 \end{bmatrix} \in \mathbb{C}^{n \times n}, \tag{5.23}$$

where $h = \frac{1}{n+1}$. In Table 5.2, we show the results obtained when inverting a more general HSS matrix (which is not banded) given by the Grünwald-Letnikov finite difference discretization of the symmetric fractional derivative operator $L_\alpha := \frac{\partial^\alpha}{\partial x^\alpha}$ [98] for $\alpha = 1.5$. In contrast to (5.23), this finite difference approximation does not yield a sparse matrix, coherently with the non-local properties of fractional differential operators. It can be proven that the matrix can be approximated in the HSS format [95] with an HSS rank $\mathcal{O}(\log n)$.

Additionally to `CKM`, we also compare to the randomized algorithm introduced by Levitt and Martinsson in [90] (denoted by `LM`) based on the solution of a small number of linear systems involving A and the `inv` procedure for HSS matrices implemented in the hm-toolbox which is based on the ULV factorization described in [29] and explained in [97, Section 4.3]. The HSS ranks are calculated using the `hssrank` command from [97], employing the default tolerance of 10^{-12}.

size	time `TelFun`	time `CKM`	time `LM`	time ULV	time Dense
1024	0.09	0.89	0.30	0.32	0.02
2048	0.11	1.21	0.33	0.35	0.08
4096	0.14	2.72	0.71	0.63	0.30
8192	0.27	6.94	2.03	0.91	1.12
16384	0.66	16.59	4.84	1.84	
32768	1.18	37.91	11.98	4.20	

size	err `TelFun`	err `CKM`	err `LM`	err ULV
1024	7.56e-13	7.74e-12	7.98e-12	8.03e-12
2048	6.15e-13	2.75e-12	7.29e-12	7.26e-12
4096	9.47e-12	5.86e-12	3.43e-12	3.99e-12
8192	8.15e-12	9.59e-11	1.58e-10	1.58e-10

Table 5.1: Comparison of the newly proposed algorithm `TelFun` with `CKM`, `LM`, and the `inv` command of the hm-toolbox based on the ULV decomposition, for computing A^{-1}, where A is the discretized Laplacian (5.23).

Although not specifically designed for matrix inversion, `TelFun` is always the fastest among the methods that exploit HSS structure, while attaining a comparable level of accuracy. Even the closest competitor ULV is significantly slower, by up to a factor 3–4.

5.5.2 Computation of the exponential function

To show the effectiveness of rational approximation of the exponential function, in Table 5.3 we compute the matrix exponential of a tridiagonal matrix A, whose eigenvalues are uniformly distributed in $[-10^a, 0]$, for different values of a. For the computation, we compare the presented method with optimal poles and `CKM` with both optimal and infinity poles, in the latter case the built-in stopping criteria are employed. In Table 5.4 we also report the comparison between the presented method and the `expm` function implemented in the hm-toolbox for the computation of $\exp(A)$ based on the Padè approximant, where A is the discretized Laplacian defined in (5.23).

Again, our newly proposed method `TelFun` is significantly faster than the competitors, while resulting in comparable accuracy. Note that `CKM` Poly appears to not use the correct stopping criterion for larger a, resulting in an unacceptably large error.

5.5.3 Computation of the inverse square root

To test the presented algorithm for the computation of the inverse square root of an HSS matrix, we consider the problem of sampling from a Gaussian Markov random field (see [33, Section 4.2]) which reduces

size	HSS rank A	time `TelFun`	time `CKM`	time `LM`	time ULV	time Dense
1024	29	0.11	0.65	0.31	0.26	0.08
2048	32	0.21	0.84	0.44	0.38	0.37
4096	35	0.36	2.17	1.06	0.63	1.72
8192	37	0.62	5.48	2.14	1.41	11.20

size	HSS rank A	err `TelFun`	err `CKM`	err `LM`	err ULV
1024	29	5.50e-13	1.11e-12	4.16e-13	7.24e-12
2048	32	4.78e-13	3.26e-12	1.18e-12	2.11e-11
4096	35	2.08e-12	2.75e-11	2.53e-12	6.81e-11
8192	37	4.99e-12	4.36e-11	9.68e-12	1.75e-10

Table 5.2: Comparison of the newly proposed algorithm `TelFun` with `CKM`, `LM`, and the `inv` command of the hm-toolbox based on the ULV decomposition, for computing A^{-1}, where A is the Grünwald-Letnikov finite difference discretization of the fractional derivative of order $\alpha = 1.5$ [95, 98].

a	time `TelFun`	time `CKM` Poly	time `CKM` Rat	err `TelFun`	err `CKM` Poly	err `CKM` Rat
0	0.82	1.38	13.06	1.04e-11	2.16e-10	1.52e-10
2	0.67	1.31	11.15	2.89e-10	1.75e-07	6.11e-09
4	0.76	0.48	10.32	2.50e-12	1.05e-03	1.20e-08
6	0.52	0.52	10.13	2.84e-10	1.03e-02	4.69e-11
8	0.53	0.48	10.14	3.36e-08	1.21e+01	5.28e-08

Table 5.3: Computation of the matrix exponential of a matrix of size 4096, whose eigenvalues are uniformly distributed in $[-10^a, 0]$, for different values of a. The accuracy is set to 10^{-8}.

to the computation of the inverse of the square root of a banded matrix. In Table 5.5 we compare our algorithm with optimal poles, with `CKM` with extended poles (i.e., alternating 0 and ∞); the latter choice of poles is the one made by the authors of `CKM` for solving the presented problem: since the algorithm needs to solve possibly large linear systems, the choice of using mutually different poles can often not be the most advantageous strategy. The number of poles to employ in our method is given by (5.22) (which is in practice very pessimistic) and the accuracy is only used in the determination of the deflation tolerance. The termination of `CKM` is due to the built-in stopping criteria. For completeness, we also approximate $f(A)$ by explicitly evaluating a rational approximation of f: the poles and the residuals of the rational approximation have been derived using the AAA algorithm [100], and for the evaluation, the HSS structure has been exploited using the hm-toolbox [97]. In all the cases reported, the degree of the rational approximant constructed by AAA is 12. While `TelFun` is still faster than `CKM` for sufficiently large n, its advantage in terms of speed is less evident for this example. Note, however, that its error is significantly lower.

We also show a comparison between `TelFun` and `CKM`, both with (quasi-)optimal poles, for the approximation of the fractional Laplacian i.e., the computation of $(-A)^{-1/2}$, where A is defined in (5.23). In Table 5.6 we compare the timing and the relative error between the presented algorithm and `CKM` using in both cases 50 optimal poles and varying the size of A.

5.5.4 Computation of the sign function

In this section, we compute $f(A)$ where f is the sign function, i.e.,

$$f(z) = \begin{cases} 1 & z > 0, \\ -1 & z \leq 0. \end{cases}$$

Assuming that A has both positive and negative eigenvalues (otherwise the computation of $f(A)$ is trivial) the discontinuity of the function does not allow for a reasonable rational approximation on an interval

size	time `TelFun`	time `expm`	err `TelFun`	err `expm`
1024	0.33	2.12	4.58e-10	5.35e-04
2048	0.59	3.45	2.01e-09	2.14e-03
4096	1.02	7.01	6.16e-09	8.52e-03
8192	2.03	14.87	2.75e-08	3.37e-02

Table 5.4: Computation of $\exp(A)$ where A is the discretization of the Laplacian defined in (5.23) of n using the presented method and the routine `expm` of the hm-toolbox. The accuracy is set to 10^{-8}.

size	HSS rank A	time `TelFun`	time `CKM`	time Rat	time Dense
512	22	0.13	0.22	1.28	0.03
1024	20	0.29	0.30	2.38	0.23
2048	21	0.57	0.73	5.09	0.87
4096	21	1.24	1.55	12.82	7.04
8192	23	3.36	4.07	27.32	63.07
16384	25	6.90	9.17		
32768	28	13.83	20.05		
65536	24	27.21	44.85		
131072	27	54.50	104.01		

size	HSS rank A	err `TelFun`	err `CKM`	err Rat
512	22	6.67e-14	2.02e-09	1.87e-09
1024	20	1.32e-13	2.70e-09	6.52e-09
2048	21	6.00e-11	3.64e-09	3.73e-09
4096	21	1.99e-13	3.39e-09	4.34e-09
8192	23	1.11e-13	3.72e-09	5.52e-09

Table 5.5: Comparison of the newly proposed algorithm `TelFun` (using optimal poles), `CKM` with extended Krylov subspaces, and the evaluation of a rational approximation, for the computation of $f(A)$ with accuracy of 10^{-8}, where $f(z) = 1/\sqrt{z}$, and A is the sampling from a Gaussian Markov random field.

containing the eigenvalues of A. In particular, our convergence result from Theorem 5.3.2 does not apply. On the other hand, if the eigenvalues of A are contained in $\mathbb{E} = [-b, -a] \cup [a, b]$, with $a, b, > 0$, then the best rational approximation of the sign function on $\mathbb{E}$ is explicitly known in terms of elliptic functions, see [106, Section 4.3]. In Table 5.7, we test the time and the accuracy of the proposed method on tridiagonal matrices whose positive eigenvalues are logarithmically distributed in the interval $[10^a, 1]$ and the negative ones are given by the symmetrization with respect to the imaginary axis. We compare the results with the ones obtained by running `CKM` with optimal poles and with the evaluation of the rational approximation given by the AAA algorithm [100], using the routines contained in the hm-toolbox [97].

While not covered by the theory, `TelFun` is clearly the best method and attains good accuracy until $a = -7$. For $a = -9$, the accuracy of all methods suffers from the fact that the eigenvalues get too close to zero.

size	time `TelFun`	time `CKM`	time Dense	err `TelFun`	err `CKM`
1024	1.44	7.15	0.19	1.32e-11	2.04e-11
2048	1.71	15.95	1.05	1.13e-11	4.68e-11
4096	4.68	41.94	7.96	1.31e-10	1.77e-10
8192	7.87	91.89			
16384	16.30	223.16			

Table 5.6: Comparison of the newly proposed algorithm `TelFun` with `CKM`, for the computation of $f(A)$ where $f(z) = 1/\sqrt{z}$, and A is the discretization of the Laplacian. In both algorithms 50 quasi-optimal poles have been employed.

a	time `TelFun`	time `CKM`	time Rat	err `TelFun`	err `CKM`	err Rat
-1	1.73	20.55	50.28	3.75e-10	3.72e-10	1.73e-09
-3	4.13	38.46	128.93	2.60e-10	4.10e-08	1.51e-08
-5	9.72	57.99	121.76	2.70e-10	2.12e-06	7.49e-08
-7	18.24	78.37	137.72	2.99e-08	1.79e-08	1.51e-05
-9	14.08	43.04	139.65	7.45e-02	7.83e-02	3.98e-02

Table 5.7: Computation of $\mathrm{sign}(A)$, where A is a tridiagonal matrix of size 4096 with logarithmically spaced eigenvalues, symmetric with respect to the imaginary axis contained in $[-1, -10^a] \cup [10^a, 1]$.

Chapter 6

Block Lanczos method with rational Krylov compression

The material presented in this chapter is a slight adjustment of the joint work with Igor Simunec described in [28].

A fundamental problem in numerical linear algebra is the approximation of the action of a matrix function $f(A)$ on a block vector C, where $A \in \mathbb{C}^{n\times n}$ is a matrix that is typically large and sparse, $C \in \mathbb{C}^{n\times b}$ is a block vector and f is a function defined on the spectrum of A. In this work, we focus on the case of A Hermitian.

Popular methods for the approximation of $f(A)C$ are (block) polynomial [54, 57, 69, 77, 92, 111] and rational Krylov methods [1, 18, 40, 67, 99]. The former only accesses A via matrix-vector products, while the latter requires the solution of shifted linear systems with A. When the linear systems can be solved efficiently, rational Krylov methods can be more effective than polynomial Krylov methods since they usually require much fewer iterations to converge. However, there are several situations in which rational Krylov methods are not applicable, either because the matrix A is only available implicitly via a function that computes matrix-vector products, or when A is very large and the solution of linear systems is prohibitively expensive.

When A is Hermitian, the core component of a block polynomial Krylov method is the block Lanczos algorithm [112], which constructs an orthonormal basis $\mathbf{Q}_m = [Q^{(1)} \dots Q^{(m)}]$ of the block polynomial Krylov subspace $\mathcal{K}_m(A, C)$ by exploiting a short term recurrence. The product $f(A)C$ can then be approximated by the Lanczos approximation

$$F_m := \mathbf{Q}_m f(\mathbf{T}_m) E_1 \Theta, \qquad \mathbf{T}_m := \mathbf{Q}_m^H A \mathbf{Q}_m, \tag{6.1}$$

where $E_1 = [I_b,\ 0]^H \in \mathbb{C}^{mb\times b}$ and $C = Q^{(1)}\Theta$, with $\Theta \in \mathbb{C}^{b\times b}$ is a thin QR factorization of C.

The block Lanczos algorithm uses a short-term recurrence in the orthogonalization step, so each new block vector is orthogonalized only against the last two block columns of the orthonormal basis, and only three block vectors need to be kept in memory to compute the basis $\mathbf{Q}_m$. Although the basis $\mathbf{Q}_m$ and the projected matrix $\mathbf{T}_m$ can be computed by using the short-term recurrence, which only requires the storage of the last three block columns of the basis, forming the approximate solution F_m still requires the full basis $\mathbf{Q}_m$. When the matrix A is very large, there may be a limit on the maximum number of block vectors that can be stored, so with a straightforward implementation of the block Lanczos method there is a limit on the number of iterations that can be performed and hence on the attainable accuracy. In the literature, several strategies have been proposed to deal with low memory issues. See the recent surveys [68, 69] for a comparison of several low-memory methods.

A simple but effective approach is the two-pass Lanczos method [21, 57]. With this approach, the block Lanczos method is first run once to determine the projected matrix $\mathbf{T}_m$ and compute the short vector $Z_m = f(\mathbf{T}_m)E_1\Theta$. After Z_m has been computed, the block Lanczos method is run for a second time to form the product $F_m = \mathbf{Q}_m Z_m$ as the columns of $\mathbf{Q}_m$ are computed. This method requires doubling the

number of matrix-vector products with A with respect to standard block Lanczos, but it requires storage of only three block vectors simultaneously.

Another possibility is the multi-shift conjugate gradient method [55, 127]. This method is based on an explicit approximation of f with a rational function r expressed in partial fraction form. Then, $f(A)C \approx r(A)C$ is approximated by using the conjugate gradient method to solve each of the linear systems that appear in the partial fraction representation of $r(A)C$. This can be done efficiently by exploiting the shift invariance of Krylov subspaces, i.e., the fact that $\mathcal{K}_m(A, C) = \mathcal{K}_m(A + \theta I, C)$ for any $\theta \in \mathbb{R}$, in order to use a single Krylov subspace to approximate solutions to all the linear systems simultaneously. Compared to Lanczos, this method requires performing additional vector operations and storing vectors proportionally to the number of poles of the rational approximant r.

When f is a Stieltjes function, it is possible to use a restarting strategy that is similar to the restarted Krylov subspace methods for linear systems [43, 53, 54]. By exploiting the Stieltjes integral representation of the function f, we can write the error after a certain number of Lanczos iterations as $f(A)C - F_m = f_m(A)Q_{m+1}$, where f_m is still a Stieltjes function that depends on f and $\mathbf{T}_m$. This property makes it possible to restart the Lanczos method after a certain number of iterations, and then iteratively approximate the error using the same method.

Recently, a low-memory method has been proposed to compute an approximation from a Krylov subspace to $f(A)C$ when f is a rational function [31]. This approximation is optimal in a norm that depends on the denominator of the rational function. If d is the degree of the denominator of f, the approximation from $\mathcal{K}_k(A, C)$ can be computed with $(k + d)b$ matrix-vector products with A, while storing approximately $2d$ block vectors. This method can be extended to non-rational functions f by means of rational approximations, and it often produces approximations that are comparable or better than the Lanczos approximation.

In this chapter, we propose a new low-memory algorithm for the approximation of $f(A)C$. Our method combines outer block Lanczos iterations with inner block rational Krylov subspaces, which are used to compress the outer Krylov basis whenever it reaches a certain size. Similarly to the procedure described in Section 5.3, the inner block rational Krylov subspace does not involve the matrix A, but only small matrices. This is a key observation since constructing a basis of the inner subspace does not require the solution of linear systems with A, and hence it is cheap compared to the cost of the outer block Lanczos iteration. The approximate solutions computed by our algorithm coincide with the ones constructed by the block Lanczos method when f is a rational function, and for a general function they differ by a quantity that depends on the best rational approximant of f with the poles used in the inner block rational Krylov subspace. In order to obtain a meaningful advantage when compressing the basis, the algorithm that we propose should be used when the block Lanczos method requires several iterations to converge.

If the outer Krylov basis is compressed every m iterations and the inner block rational Krylov subspace has k poles, our approach requires storing approximately $m + k$ block vectors. Additionally, due to the basis compression, our approximation requires computing functions of matrices of size at most $(m + k)b \times (m + k)b$, so the cost does not grow with the number of iterations. This represents an important advantage with respect to the block Lanczos method, since when the number of iterations is very large the evaluation of f on the projected matrix can become quite expensive.

6.1 Block Lanczos algorithm

When the Arnoldi algorithm is employed for the computation of an orthonormal basis $\mathbf{Q}_k$ of $\mathcal{K}_k(A, C)$, the orthogonalization steps can significantly slow down the procedure for large k. If A is Hermitian, orthogonalizing only with respect two block vectors at each orthogonalization step (theoretically) guarantees the orthogonality of the Krylov basis. This variant of the Arnoldi algorithm is commonly known as the block Lanczos algorithm see [112] or [61, Section 10.3.6]. After the kth step, the block Lanczos algorithm com-

Algorithm 6 Block Lanczos Algorithm

Input: $A \in \mathbb{C}^{n\times n}, C \in \mathbb{C}^{n\times b}, k \in \mathbb{N}$
Output: $\mathbf{Q}_k \in \mathbb{C}^{n\times kb}, \Delta_1, \ldots, \Delta_{k-1}, \Gamma_1, \ldots \Gamma_{k-1}, \in \mathbb{C}^{b\times b}$
1: $[Q^{(1)}, R] \leftarrow \mathrm{qr}(C)$ ▷ where qr is a thin QR factorization
2: **for** $j = 1, \ldots, k-1$ **do**
3: $[Q^{(j+1)}, \Delta_j, \Gamma_j] \leftarrow$ Algorithm 7$(A, Q^{(j-1)}, Q^{(j)}, \Gamma_{j-1})$
4: **end for**
5: $\mathbf{Q}_k \leftarrow [Q^{(1)}, \ldots, Q^{(k)}]$

Algorithm 7 Single iteration of block Lanczos

Input: $A \in \mathbb{C}^{n\times n}, Q^{(1)}, Q^{(2)} \in \mathbb{C}^{n\times b}, \Gamma_1 \in \mathbb{C}^{b\times b}$
Output: $Q^{(3)} \in \mathbb{C}^{n\times b}, \Delta_2, \Gamma_2 \in \mathbb{C}^{b\times b}$
1: $Z \leftarrow AQ^{(2)} - Q^{(1)}\Gamma_1^H$
2: $\Delta_2 \leftarrow (Q^{(2)})^H Z$
3: $Z \leftarrow Z - Q^{(2)}\Delta_2$
4: $[Q^{(3)}, \Gamma_2] \leftarrow \mathrm{qr}(Z)$ ▷ where qr is a thin QR factorization

putes the block columns of $\mathbf{Q}_k$, the block tridiagonal matrix

$$\mathbf{T}_{k-1} = \mathbf{Q}_{k-1}^H A \mathbf{Q}_{k-1} = \begin{bmatrix} \Delta_1 & \Gamma_1^H & & \\ \Gamma_1 & \Delta_2 & \ddots & \\ & \ddots & \ddots & \Gamma_{k-2}^H \\ & & \Gamma_{k-2} & \Delta_{k-1} \end{bmatrix}$$

with $\Gamma_i, \Delta_i \in \mathbb{C}^{b\times b}$ and the matrix $\Gamma_{k-1} := (Q^{(k)})^H A Q^{(k-1)} \in \mathbb{C}^{b\times b}$, where $Q^{(k)}$ and $Q^{(k-1)}$ are the last and second-to-last block columns of $\mathbf{Q}_k$, respectively, as summarized in Algorithm 6. As in the Arnoldi algorithm, if $\mathbf{Q}_k$ is computed by the block Lanczos algorithm, its first i block columns span $\mathcal{K}_i(A, C)$ for each $i \leq k$. It has been observed that the block Lanczos algorithm can produce a numerical loss of orthogonality in the columns of $\mathbf{Q}_k$ in finite precision arithmetic, see [61, Section 10.3] for more details.

Given a function f well defined on the spectrum of A, approximating $f(A)C$ by projection onto the Krylov subspace $\mathcal{K}_{k-1}(A, C)$, employing (6.1) requires the computation of the first block column of the matrix $f(\mathbf{T}_{k-1})$. The following lemma describes how to exploit the block tridiagonal structure of $\mathbf{T}_{k-1}$ when f is a rational function.

Lemma 6.1.1. *Let $r = p/q$ be a rational function with complex conjugate roots and poles, and denote by $\boldsymbol{\xi}_k$ the sequence of cardinality $k := \max\{\deg(p)+1, \deg(q)\}$ that contains the poles of r and infinity in the remaining elements. Let $A \in \mathbb{C}^{n\times n}$ be a Hermitian matrix, partitioned as*

$$A = \begin{bmatrix} A_{11} & \\ & A_{22} \end{bmatrix} + \begin{bmatrix} & A_{12} \\ A_{12}^H & \end{bmatrix},$$

with $A_{11} \in \mathbb{C}^{m\times m}$. Assume that $A_{12} = BC^H$, where $B \in \mathbb{C}^{m\times b}$ and $C \in \mathbb{C}^{(n-m)\times b}$. Then

$$r(A) = \begin{bmatrix} r(A_{11}) & \\ & r(A_{22}) \end{bmatrix} + \begin{bmatrix} U_k & \\ & I \end{bmatrix} X_k(r) \begin{bmatrix} U_k^H & \\ & I \end{bmatrix}, \tag{6.2}$$

where U_k is an orthonormal basis of $\mathcal{Q}_k(A_{11}, B, \boldsymbol{\xi}_k)$, I is the $(n-m)\times(n-m)$ identity matrix and

$$X_k(r) = r\left(\begin{bmatrix} U_k^H A_{11} U_k & U_k^H B C^H \\ C B^H U_k & A_{22} \end{bmatrix}\right) - \begin{bmatrix} r(U_k^H A_{11} U_k) & \\ & r(A_{22}) \end{bmatrix}. \tag{6.3}$$

In particular, for any block vector of the form $\begin{bmatrix} V \\ 0 \end{bmatrix}$ *where* $V \in \mathbb{C}^{m\times b}$*, we have*

$$r(A)\begin{bmatrix} V \\ 0 \end{bmatrix} = \begin{bmatrix} Y \\ 0 \end{bmatrix} + \begin{bmatrix} Z \\ 0 \end{bmatrix} + R, \tag{6.4}$$

where

$$Y = r(A_{11})V, \quad Z = -U_k r(U_k^H A_{11} U_k) U_k^H V$$

and

$$R = \begin{bmatrix} U_k & \\ & I \end{bmatrix} r\left(\begin{bmatrix} U_k^H A_{11} U_k & U_k^H B C^H \\ C B^H U_k & A_{22} \end{bmatrix}\right) \begin{bmatrix} U_k^H V \\ 0 \end{bmatrix}.$$

Proof. We can write

$$\begin{bmatrix} & A_{12} \\ A_{12}^H & \end{bmatrix} = \begin{bmatrix} B & \\ & C \end{bmatrix} \begin{bmatrix} 0 & I_b \\ I_b & 0 \end{bmatrix} \begin{bmatrix} B^H & \\ & C^H \end{bmatrix}.$$

Due to Proposition 2.3.3, we have

$$\mathcal{Q}_k\left(\begin{bmatrix} A_{11} & \\ & A_{22} \end{bmatrix}, \begin{bmatrix} B & \\ & C \end{bmatrix}, \boldsymbol{\xi}_k\right) = \operatorname{span}\left(\begin{bmatrix} U_k & \\ & V_k \end{bmatrix}\right),$$

where V_k is an orthonormal basis of $\mathcal{Q}_k(A_{22}, C, \boldsymbol{\xi}_k)$. Since

$$\operatorname{span}\left(\begin{bmatrix} U_k & \\ & V_k \end{bmatrix}\right) \subseteq \operatorname{span}\left(\begin{bmatrix} U_k & \\ & I_{n-m} \end{bmatrix}\right),$$

we can use Proposition 2.4.2 to obtain (6.2), and (6.4) as an immediate consequence. □

In particular, Lemma 6.1.1 shows that if f is a rational function, the block vector $f(\mathbf{T}_{k-1})E_1$ can be computed as the sum of the three terms given by (6.4), where the first two terms only involve a sumbatrix of $\mathbf{T}_{k-1}$ that can be taken as the projection of A onto $\mathcal{K}_j(A, C)$, for $j < k-1$ and the second term is recoursively defined as the product between the evaluation of f on a smaller matrix and a block vector. This will be the key idea for the development of the algorithm described in Section 6.2

6.2 Algorithm description

In this section, we present a low-memory implementation of a Krylov subspace method for the computation of $f(A)C$ for a Hermitian matrix A and a block vector C. On a high level, we use an outer block polynomial Krylov subspace, combined with an inner rational Krylov subspace that is employed to compress the outer Krylov subspace basis and reduce memory usage. The inner Krylov subspace is constructed using the projection of A on the outer Krylov subspace, so it does not involve any expensive operations with the matrix A. This approach is designed for scenarios where the (outer) Lanczos method requires a large number of iterations to converge, in order to take full advantage of the basis compressions. The Lanczos iterations should be cheaper than the solution of shifted linear systems involving A and the poles employed in the inner Krylov method, otherwise it would be more efficient to simply use a rational Krylov method associated with the inner poles directly on the matrix A.

The algorithm is composed of s cycles, with each cycle consisting in m iterations of the block Lanczos method (except for the initial cycle where we perform $m+k$ iterations), and a subsequent compression of the basis to k block vectors. At any given moment, the algorithm keeps in memory at most $m+k$ basis block vectors and some additional quantities (whose storage is not dependent on n), such as projected matrices of size $(m+k)b$. In total, the algorithm performs $M = k + ms$ iterations of the outer Lanczos method, and the approximation computed at the end of each cycle coincides with the approximation computed by a standard implementation of the block Lanczos method, up to an error due to the rational approximation done in the inner Krylov subspace, which is usually negligible. This error is zero in the case where f is a

rational function of type $(k-1,k)$, so for simplicity we start by describing the algorithm in this special case.

Let r be a rational function of type $(k-1,k)$, i.e., $r(z) = p(z)/q(z)$ with p and q polynomials of degrees at most $k-1$ and k, respectively. We assume that r has complex conjugate roots and poles. We let $\boldsymbol{\xi}_k$ be the sequence of poles $\{\xi_0, \ldots, \xi_{k-1}\}$ containing the roots of q and infinity in the remaining entries in the case $\deg(q) < k$. These poles will be used for the inner rational Krylov iteration. Our goal is to construct an approximation of $r(A)C$ from the outer Krylov subspace $\mathcal{K}_M(A,C)$, where $M = k+ms$. Throughout this section, we are going to denote by $\mathbf{Q}_j$, and $\mathbf{T}_j$ the matrices associated with the outer Krylov subspace $\mathcal{K}_j(A,C)$, according to the notation used in Section 6.1.

We also introduce some additional notation for the matrices associated with the outer Krylov subspace $\mathcal{K}_M(A,C)$, in order to simplify the exposition in this section. We denote by $\widehat{Q} := \mathbf{Q}_M$ the orthonormal basis of $\mathcal{K}_M(A,C)$ and by $\widehat{T} := \widehat{Q}^H A \widehat{Q} = \mathbf{T}_M$.

We denote the approximation (6.1) to $r(A)C$ from $\mathcal{K}_M(A,C)$ by

$$\widehat{Y} := \widehat{Q}\, r(\widehat{T}) E_1 \Theta, \tag{6.5}$$

where $E_1 = \mathbf{e}_1 \otimes I_b \in \mathbb{C}^{Mb\times b}$ and $\Theta \in \mathbb{C}^{b\times b}$ sathisfies $\widehat{Q}^H C = E_1\Theta$. We split $\widehat{Q} = [Q_1,\, Q_2,\, \ldots,\, Q_s]$, with $Q_1 \in \mathbb{C}^{n\times(k+m)b}$ and $Q_i \in \mathbb{C}^{n\times mb}$ for $i = 2,\ldots,s$. We also use the notation $\widehat{Q}_i = [Q_i,\, \ldots,\, Q_s]$, so that we have $\widehat{Q}_i = [Q_i,\, \widehat{Q}_{i+1}]$ for $i = 1,\ldots,s-1$.

We introduce a similar notation for the matrix $\widehat{T}$. We denote by T_i the ith diagonal block of $\widehat{T}$, with $T_1 \in \mathbb{C}^{(k+m)b\times(k+m)b}$ and $T_i \in \mathbb{C}^{mb\times mb}$ for $i = 2,\ldots,s$, and by $\widehat{T}_{i+1} \in \mathbb{C}^{(s-i)mb\times(s-i)mb}$ the trailing block on the diagonal after T_i. So we have

$$\widehat{T} = \begin{bmatrix} T_1 & B_1E_1^H \\ E_1B_1^H & \widehat{T}_2 \end{bmatrix}, \qquad \widehat{T}_i = \begin{bmatrix} T_i & B_iE_1^H \\ E_1B_i^H & \widehat{T}_{i+1} \end{bmatrix}, \quad 2 \le i \le s-1,$$

where $B_1 \in \mathbb{C}^{(m+k)b\times b}$, $B_i \in \mathbb{C}^{mb\times b}$ for $i = 1,\ldots,s-1$, E_1 is the block vector $[I_b, 0]^H$ of the appropriate dimension, and $\widehat{T}_s = T_s$. The block structure of $\widehat{T}$ results from its block tridiagonal form.

6.2.1 First cycle

We have

$$\widehat{T} = \begin{bmatrix} T_1 & \\ & \widehat{T}_2 \end{bmatrix} + \begin{bmatrix} & B_1E_1^H \\ E_1B_1^H & \end{bmatrix}.$$

Let $U_1 \in \mathbb{C}^{(m+k)b\times kb}$ be an orthonormal basis of the block rational Krylov subspace $\mathcal{Q}_k(T_1, B_1, \boldsymbol{\xi}_k)$. Using Lemma 6.1.1, we can write

$$r\left(\begin{bmatrix} T_1 & B_1E_1^H \\ E_1B_1^H & \widehat{T}_2 \end{bmatrix}\right)\begin{bmatrix} E_1\Theta \\ 0 \end{bmatrix} = \begin{bmatrix} \widetilde{Y}_1 \\ 0 \end{bmatrix} + \begin{bmatrix} \widetilde{W}_1 \\ 0 \end{bmatrix} + \widetilde{R}_1$$

where

$$\widetilde{Y}_1 = r(T_1)E_1\Theta, \qquad \widetilde{W}_1 = -U_1 r(U_1^H T_1 U_1) U_1^H E_1 \Theta,$$

and

$$\widetilde{R}_1 = \begin{bmatrix} U_1 & \\ & I \end{bmatrix} r\left(\begin{bmatrix} U_1^HT_1U_1 & U_1^HB_1E_1^H \\ E_1B_1^HU_1 & \widehat{T}_2 \end{bmatrix}\right)\begin{bmatrix} U_1^HE_1\Theta \\ 0 \end{bmatrix}.$$

Recalling the notation introduced above, we have

$$\widehat{Q}r(\widehat{T})E_1\Theta = \begin{bmatrix} Q_1 & \widehat{Q}_2 \end{bmatrix}\left(\begin{bmatrix} \widetilde{Y}_1 \\ 0 \end{bmatrix} + \begin{bmatrix} \widetilde{W}_1 \\ 0 \end{bmatrix} + \widetilde{R}_1\right) = Q_1\widetilde{Y}_1 + Q_1\widetilde{W}_1 + \begin{bmatrix} Q_1 & \widehat{Q}_2 \end{bmatrix}\widetilde{R}_1.$$

To summarize, we have obtained

$$\widehat{Y} = \widehat{Q}\, r(\widehat{T})E_1\Theta = Y_1 + W_1 + R_1 \tag{6.6}$$

where

$$Y_1 = Q_1 r(T_1) E_1 \Theta$$

is the approximation (6.1) to $r(A)C$ from $\mathcal{K}_{k+m}(A, C)$,

$$W_1 = -Q_1 U_1 r(U_1^H T_1 U_1) U_1^H E_1 \Theta$$

and

$$R_1 = \begin{bmatrix} Q_1 U_1 & \widehat{Q}_2 \end{bmatrix} r\left(\begin{bmatrix} U_1^H T_1 U_1 & U_1^H B_1 E_1^H \\ E_1 B_1^H U_1 & \widehat{T}_2 \end{bmatrix} \right) \begin{bmatrix} U_1^H E_1 \Theta \\ 0 \end{bmatrix}. \tag{6.7}$$

After the first $k+m$ iterations of the outer Krylov subspace method, we can compute Y_1 and W_1, but the remainder R_1 still involves $\widehat{Q}_2$ and $\widehat{T}_2$, which have not been computed yet. A crucial observation that allows us to save memory is that in (6.7) the basis Q_1 only appears in the product $Q_1 U_1$, so from now it is not necessary to keep in memory the whole matrix Q_1.

Introducing the notation $V_1 := Q_1$, $S_1 := T_1$, $Z_1 := E_1\Theta$ and $\widetilde{B}_1 := B_1$, we can write

$$R_1 = \begin{bmatrix} V_1 U_1 & \widehat{Q}_2 \end{bmatrix} r\left(\begin{bmatrix} U_1^H S_1 U_1 & U_1^H \widetilde{B}_1 E_1^H \\ E_1 \widetilde{B}_1^H U_1 & \widehat{T}_2 \end{bmatrix} \right) \begin{bmatrix} U_1^H Z_1 \\ 0 \end{bmatrix}. \tag{6.8}$$

The notation introduced for (6.8) sets us up for describing the $i-$th cycle of the algorithm.

6.2.2 $i-$th cycle

At the beginning of the ith cycle, with $2 \leq i \leq s-1$, our task is to compute the remainder R_{i-1} from the previous cycle, which is given by

$$R_{i-1} = \begin{bmatrix} V_{i-1} U_{i-1} & \widehat{Q}_i \end{bmatrix} r\left(\begin{bmatrix} U_{i-1}^H S_{i-1} U_{i-1} & U_{i-1}^H \widetilde{B}_{i-1} E_1^H \\ E_1 \widetilde{B}_{i-1}^H U_{i-1} & \widehat{T}_i \end{bmatrix} \right) \begin{bmatrix} U_{i-1}^H Z_{i-1} \\ 0 \end{bmatrix}. \tag{6.9}$$

Note that (6.8) coincides with (6.9) with $i = 2$.

Expanding $\widehat{T}_i$ in terms of T_i, B_i and $\widehat{T}_{i+1}$, we have

$$\begin{bmatrix} U_{i-1}^H S_{i-1} U_{i-1} & U_{i-1}^H \widetilde{B}_{i-1} E_1^H \\ E_1 \widetilde{B}_{i-1}^H U_{i-1} & \widehat{T}_i \end{bmatrix} = \begin{bmatrix} U_{i-1}^H S_{i-1} U_{i-1} & U_{i-1}^H \widetilde{B}_{i-1} E_1^H & \\ E_1 \widetilde{B}_{i-1}^H U_{i-1} & T_i & B_i E_1^H \\ & E_1 B_i^H & \widehat{T}_{i+1} \end{bmatrix}.$$

Introducing the notation $V_i := \begin{bmatrix} V_{i-1} U_{i-1} & Q_i \end{bmatrix} \in \mathbb{C}^{n \times (k+m)b}$, $Z_i := \begin{bmatrix} U_{i-1}^H Z_{i-1} \\ 0 \end{bmatrix} \in \mathbb{C}^{(k+m)b}$, $\widetilde{B}_i := \begin{bmatrix} 0 \\ B_i \end{bmatrix} \in \mathbb{C}^{(k+m)b}$ and

$$S_i = \begin{bmatrix} U_{i-1}^H S_{i-1} U_{i-1} & U_{i-1}^H \widetilde{B}_{i-1} E_1^H \\ E_1 \widetilde{B}_{i-1}^H U_{i-1} & T_i \end{bmatrix},$$

we can rewrite (6.9) as

$$R_{i-1} = \begin{bmatrix} V_i & \widehat{Q}_{i+1} \end{bmatrix} r\left(\begin{bmatrix} S_i & \widetilde{B}_i E_1^H \\ E_1 \widetilde{B}_i^H & \widehat{T}_{i+1} \end{bmatrix} \right) \begin{bmatrix} Z_i \\ 0 \end{bmatrix}. \tag{6.10}$$

The right-hand side of (6.10) can be computed with the same strategy that was used in the first cycle to compute $\widehat{Q} r(\widehat{T}) E_1 \Theta$. Letting $U_i \in \mathbb{C}^{(k+m)b \times kb}$ be an orthonormal basis of $\mathcal{Q}_k(S_i, \widetilde{B}_i, \boldsymbol{\xi}_k)$, we can use Lemma 6.1.1 once again to obtain

$$r\left(\begin{bmatrix} S_i & \widetilde{B}_i E_1^H \\ E_1 \widetilde{B}_i^H & \widehat{T}_{i+1} \end{bmatrix} \right) \begin{bmatrix} Z_i \\ 0 \end{bmatrix} = \begin{bmatrix} \widetilde{Y}_i \\ 0 \end{bmatrix} + \begin{bmatrix} \widetilde{W}_i \\ 0 \end{bmatrix} + \widehat{R}_i$$

where

$$\widetilde{Y}_i = r(S_i)Z_i, \qquad \widetilde{W}_i = -U_i r(U_i^H S_i U_i) U_i^H Z_i$$

and

$$\widehat{R}_i = \begin{bmatrix} U_i & \\ & I \end{bmatrix} r\left(\begin{bmatrix} U_i^H S_i U_i & U_i^H \widetilde{B}_i E_1^H \\ E_1 \widetilde{B}_i^H U_i & \widehat{T}_{i+1} \end{bmatrix} \right) \begin{bmatrix} U_i^H Z_i \\ 0 \end{bmatrix}.$$

From this, we easily obtain

$$R_{i-1} = V_i \widetilde{Y}_i + V_i \widetilde{W}_i + [V_i \ \widehat{Q}_{i+1}] \widehat{R}_i = V_i r(S_i) Z_i + W_i + R_i, \tag{6.11}$$

where

$$W_i = -V_i U_i r(U_i^H S_i U_i) U_i^H Z_i$$

and

$$R_i = \begin{bmatrix} V_i & \widehat{Q}_{i+1} \end{bmatrix} \widehat{R}_i = \begin{bmatrix} V_i U_i & \widehat{Q}_{i+1} \end{bmatrix} r\left(\begin{bmatrix} U_i^H S_i U_i & U_i^H \widetilde{B}_i E_1^H \\ E_1 \widetilde{B}_i^H U_i & \widehat{T}_{i+1} \end{bmatrix} \right) \begin{bmatrix} U_i^H Z_i \\ 0 \end{bmatrix}. \tag{6.12}$$

Note that if we replace i with $i-1$ in (6.12) we obtain (6.9), i.e., we have written the remainder R_i in a form that is ready for the $(i+1)$th cycle.

The approximate solution to $r(A)C$ is updated with the identity

$$Y_i = Y_{i-1} + W_{i-1} + V_i r(S_i) Z_i. \tag{6.13}$$

Recalling (6.6), in the first cycle we have $\widehat{Y} = Y_1 + W_1 + R_1$. It is easy to prove by induction that a similar identity holds in all subsequent cycles. Indeed, assuming that $\widehat{Y} = Y_{i-1} + W_{i-1} + R_{i-1}$, we have

$$\widehat{Y} = Y_{i-1} + W_{i-1} + V_i r(S_i) Z_i + W_i + R_i = Y_i + W_i + R_i,$$

where we have used (6.11) and (6.13). Note that, similarly to the first cycle, in (6.12) the matrix V_i only appears multiplied by U_i, so after computing Y_i and W_i it is no longer necessary to store the whole basis V_i.

6.2.3 Final cycle

The final cycle is slightly simpler since we can compute the remainder directly instead of using the low-rank update formula. Indeed, at the beginning of the sth cycle the remainder R_{s-1} can be computed directly from (6.12), that reads

$$R_{s-1} = \begin{bmatrix} V_{s-1} U_{s-1} & Q_s \end{bmatrix} r\left(\begin{bmatrix} U_{s-1}^H S_{s-1} U_{s-1} & U_{s-1}^H \widetilde{B}_{s-1} E_1^H \\ E_1 \widetilde{B}_{s-1}^H U_{s-1} & T_s \end{bmatrix} \right) \begin{bmatrix} U_{s-1}^H Z_{s-1} \\ 0 \end{bmatrix}.$$

Using the same notation introduced in previous cycles, we define $V_s := \begin{bmatrix} V_{s-1} U_{s-1} & Q_s \end{bmatrix}$,

$$Z_s := \begin{bmatrix} U_{s-1}^H Z_{s-1} \\ 0 \end{bmatrix} \qquad \text{and} \qquad S_s := \begin{bmatrix} U_{s-1}^H S_{s-1} U_{s-1} & U_{s-1}^H \widetilde{B}_{s-1} E_1^H \\ E_1 \widetilde{B}_{s-1}^H U_{s-1} & T_s \end{bmatrix},$$

so we have

$$R_{s-1} = V_s r(S_s) Z_s,$$

and the final approximation to $r(A)C$ is obtained as

$$Y_s = Y_{s-1} + W_{s-1} + V_s r(S_s) Z_s. \tag{6.14}$$

Note that (6.14) coincides with (6.13) where i has been replaced by s, so in the final cycle of the algorithm we compute the same update as in the other cycles, even though the derivation is different.

Since in the $(s-1)$th cycle we have $\widehat{Y} = Y_{s-1} + W_{s-1} + R_{s-1}$, it follows that $Y_s = \widehat{Y}$, i.e., in the last cycle the algorithm that we described computes the same approximation to $r(A)C$ as $M = k + sm$ iterations of the outer Lanczos method. We are going to show in Proposition 6.3.1 that the same approximations as Lanczos are computed also in the intermediate cycles. The resulting algorithm is summarized in Algorithm 8.

Algorithm 8 RK-Compressed Lanczos for $f(A)C$

Input: $A \in \mathbb{C}^{n\times n}, C \in \mathbb{C}^{n\times b}$, $k, s, m \in \mathbb{N}$, $\boldsymbol{\xi}_k = \{\xi_0, \dots, \xi_{k-1}\}$.
Output: $Y_s \approx f(A)C$, such that $Y_s \in \mathcal{K}_{k+sm}(A, C)$

▷ First outer cycle:

1: Run $m + k + 1$ iterations of the block Lanczos algorithm taking as input A and C, to compute Q_1, T_1 and B_1
2: Compute the first approximation $Y_1 = Q_1 f(T_1) E_1 \Theta$, where $E_1\Theta = Q_1^H C$
3: Define $S_1 := T_1$, $V_1 := Q_1$, $Z_1 := E_1\Theta$ and $\widetilde{B}_1 := B_1$

▷ Other outer cycles:

4: **for** $i = 2, \dots, s$ **do**
5: Construct the basis U_{i-1} of the rational Krylov subspace $\mathcal{Q}_k(S_{i-1}, \widetilde{B}_{i-1}, \boldsymbol{\xi}_k)$ employing the Arnoldi algorithm
6: Compute the correction $W_{i-1} = -V_{i-1}U_{i-1} f(U_{i-1}^H S_{i-1} U_{i-1}) U_{i-1}^H Z_{i-1}$
7: Run m additional iterations of the block Lanczos algorithm to compute Q_i, T_i and $\widetilde{B}_i$
8: Define $V_i := \begin{bmatrix} V_{i-1}U_{i-1} & Q_i \end{bmatrix}$, $Z_i = \begin{bmatrix} U_{i-1}^H Z_{i-1} \\ 0 \end{bmatrix}$, $S_i = \begin{bmatrix} U_{i-1}^H S_{i-1} U_{i-1} & U_{i-1}^H \widetilde{B}_{i-1} E_1^H \\ E_1 \widetilde{B}_{i-1}^H U_{i-1} & T_i \end{bmatrix}$.
9: Update the approximation $Y_i = Y_{i-1} + W_{i-1} + V_i f(S_i) Z_i$
10: **end for**

6.3 Analysis and comparison with existing algorithms

In this section, we analyze Algorithm 8 from both a theoretical and computational point of view, and we compare it with other low-memory methods from the literature.

6.3.1 Theoretical results

We start by showing that the iterates computed by Algorithm 8 coincide with iterates of the Lanczos method when f is a rational function.

Proposition 6.3.1. *When f is a rational function of type $(k-1, k)$ with poles given by $\boldsymbol{\xi}_k$, the approximations $\{Y_i\}_{i=1}^s$ computed by Algorithm 8 coincide with the approximations given by* (6.1) *with a block Krylov subspace of the appropriate dimension. Precisely, for any $i = 1, \dots, s$ we have*

$$Y_i = \mathbf{Q}_{k+im} f(\mathbf{T}_{k+im}) E_1 \Theta, \qquad \mathbf{T}_{k+im} = \mathbf{Q}_{k+im}^H A \mathbf{Q}_{k+im},$$

where $\mathbf{Q}_{k+im}$ is the orthonormal basis of $\mathcal{K}_k(A, C)$ generated by the block Lanczos algorithm.

Proof. We have already observed that Y_1 coincides with the approximation (6.1) from the Krylov subspace $\mathcal{K}_{k+m}(A, C)$, and we have shown at the end of Section 6.2.3 that Y_s coincides with the approximation $\widehat{Y}$ computed by the outer Krylov method after $M = k + sm$ iterations. To show that this also holds for all other cycles, let us fix $1 < i < s$ and suppose that we run a variant of Algorithm 8 with s replaced by $s' = i$ and M replaced by $M' = k + im$. It is easy to see that this variant of the algorithm performs exactly the same operations as the original one up to the $(i-1)$th cycle, and hence computes the same approximate solutions $Y_1, \dots, Y_{i-1}$. The only difference from the original algorithm is that the ith iteration is carried out by directly computing the residual R_{i-1} as described in Section 6.2.3, instead of performing the low-rank update with Lemma 6.1.1. However, as shown in Section 6.2.3 for the final iteration of the original algorithm, the update formula (6.14) for the final approximation to $r(A)C$ coincides with (6.13), so this algorithm variant also computes the same approximation Y_i as the original one. Since the final approximation computed by the variant of the algorithm coincides with the approximation generated by the outer Krylov subspace $\mathcal{K}_{k+im}(A, C)$ after $M' = k + im$ iterations, we conclude that this also holds for the approximation Y_i computed in the ith cycle of the original algorithm. □

When Algorithm 8 is applied to a general function f, we have to use Proposition 2.4.2 instead of Lemma 6.1.1, so in each cycle we introduce an error that depends on the approximation error of f with rational functions. The following proposition provides a bound on the error of Algorithm 8 with respect to the Lanczos algorithm in the general case.

Proposition 6.3.2. *Denoting by $\lambda_{\min}$ and $\lambda_{\max}$ the extrema eigenvalues of A, assume that Algorithm 8 with s cycles is applied to a function f that is analytic on $[\lambda_{\min}, \lambda_{\max}]$, using inner poles $\boldsymbol{\xi}_k$. The error of Algorithm 8 with respect to the block Lanczos algorithm is bounded by*

$$\left\| \widehat{Y} - Y_s \right\|_F \leq 4(s-1) \left\| C \right\|_F \min_{r \in \Pi_{k-1}/q} \left\| f - r \right\|_\infty , \tag{6.15}$$

where q is a polynomial with roots given by the finite elements of $\boldsymbol{\xi}_k$ and $\left\| \cdot \right\|_\infty$ denotes the supremum norm on $[\lambda_{\min}, \lambda_{\max}]$.

Proof. For convenience, we introduce the notation $\widetilde{S}_i = \begin{bmatrix} S_i & \widetilde{B}_i E_1^H \\ E_1 \widetilde{B}_i^H & \widehat{T}_{i+1} \end{bmatrix}$. Whenever we use Lemma 6.1.1 in the ith cycle of Algorithm 8, in the expression for $f(\widetilde{S}_i)$ we have an additional error term $E_i(f)$, that by Proposition 2.4.2 satisfies

$$\left\| E_i(f) \right\|_2 \leq 4 \min_{r \in \Pi_{k-1}/q} \left\| f - r \right\|_\infty .$$

Note that for all i, the spectrum of $\widetilde{S}_i$ is contained in the interval $[\lambda_{\min}, \lambda_{\max}]$. Including the error term, the expression for R_{i-1} becomes

$$R_{i-1} = V_i f(S_i) Z_i + W_i + R_i + \epsilon_i,$$

where

$$\epsilon_i = \begin{bmatrix} V_i & \widehat{Q}_{i+1} \end{bmatrix} E_i(f) \begin{bmatrix} Z_i \\ 0 \end{bmatrix}.$$

Since V_i and $\widetilde{Q}_{i+1}$ have orthonormal columns, and $\left\| Z_i \right\|_F \leq \left\| C \right\|_F$, we have[1]

$$\left\| \epsilon_i \right\|_F \leq \left\| E_i(f) \right\|_2 \left\| C \right\|_F \leq 4 \left\| C \right\|_F \min_{r \in \Pi_{k-1}/q} \left\| f - r \right\|_\infty .$$

It is easy to see that, using the update formula (6.13) for Y_i as in the rational function case, the approximation in the ith cycle satisfies the identity

$$\widehat{Y} = Y_i + W_i + R_i + \sum_{\ell=1}^{i} \epsilon_\ell .$$

Since in the final cycle the remainder R_{s-1} is computed directly, the total error of Algorithm 8 with respect to the Lanczos algorithm is thus

$$\left\| \widehat{Y} - Y_s \right\|_F \leq \sum_{\ell=1}^{s-1} \epsilon_i \leq 4(s-1) \left\| C \right\|_F \min_{r \in \Pi_{k-1}/q} \left\| f - r \right\|_\infty .$$

□

Proposition 6.3.2 shows that if we take $\boldsymbol{\xi}_k$ as the poles of a high accuracy rational approximant of f, the iterates Y_i generated by Algorithm 8 essentially coincide with the corresponding approximation computed by the Lanczos algorithm, since the error in (6.15) is going to be negligible compared to the error $\left\| f(A)C - \widehat{Y} \right\|_F$ of the outer block Lanczos method.

[1] We recall that for any couple of matrices A, B of compatible size, it holds $\left\| AB \right\|_F \leq \left\| A \right\|_2 \left\| B \right\|_F$.

6.3.2 Computational discussion

In its ith cycle, Algorithm 8 has to keep in memory the last two block columns of Q_{i-1} required to run the short recurrence for the computation of Q_i, the compressed basis $V_i \in \mathbb{C}^{n\times(k+m)b}$ and the projected matrix $S_i \in \mathbb{C}^{(k+m)b\times(k+m)b}$, as well as the vectors Y_{i-1}, W_{i-1}, V_i and $\widetilde{B}_i$.

It is important to note that Algorithm 8 only computes functions of matrices of size $kb \times kb$ and $(k+m)b \times (k+m)b$, that is independent of the number of cycles s, so the cost of computing functions of projected matrices does not increase with the iteration number, in contrast with the Lanczos method. This can lead to significant computational savings when the number of iterations is very large.

In each cycle, Algorithm 8 also has to construct a k-dimensional block rational Krylov subspace associated with a $(k+m)b \times (k+m)b$ matrix. Since k, b and m are typically much smaller than n, the cost of operations with matrices and vectors of size $(m+k)b$ is usually negligible with respect to the cost of operations with vectors of size n, so we expect that the construction of the inner rational Krylov subspace will not have a significant impact on the overall performance of the method.

In addition to the parameter k, which determines the accuracy of the inner rational approximation, and hence the highest accuracy attainable by the method, we also have to choose the parameter m, which determines the number of block vectors that are added to the basis between two consecutive compressions. This parameter should be chosen according to the available memory. Taking m small decreases memory requirements but increases the frequency of computations with $(k+m)b \times (k+m)b$ matrices.

Note that in the pseudocode of Algorithm 8, our method only computes the approximate solutions Y_i once every m iterations. However, the algorithm can be easily adapted to compute an approximate solution in every outer iteration, using the same update formula as in line 9 of Algorithm 8 but with a smaller matrix S_i. Note that the only difference with respect to the definition of S_i in line 8 is in the T_i block and in the size of the off-diagonal blocks, so the only additional operation required to compute an approximate solution is the computation of a matrix function of size at most $(k+m)b \times (k+m)b$. It is easy to see that an approximate solution computed in this way also coincides with the one constructed in the corresponding iteration of the outer Lanczos method, via the same argument used in the proof of Proposition 6.3.1.

Instead of running the method for a fixed number of cycles, in practice it is desirable to run it until we obtain a solution with a certain accuracy. A commonly used stopping criterion that can be employed for this purpose is the norm of the difference of two consecutive approximations. This simple criterion offers no guarantee on the final error and it may underestimate it in practice, especially when convergence is slow, but we found it to be accurate enough for our purposes. Observe that in order to check the convergence condition it is not necessary to form the approximate solutions Y_i of length n, but it is enough to compute differences of short vectors. Indeed, we have

$$\begin{aligned} Y_i - Y_{i-1} &= W_{i-1} + V_i f(S_i) Z_i \\ &= \begin{bmatrix} V_{i-1}U_{i-1} & Q_i \end{bmatrix} \left(\begin{bmatrix} f(U_{i-1}^H S_{i-1} U_{i-1}) U_{i-1}^H Z_{i-1} \\ 0 \end{bmatrix} + f(S_i) Z_i \right), \end{aligned}$$

and since V_i has orthonormal columns we conclude that

$$\|Y_i - Y_{i-1}\|_F = \left\| \begin{bmatrix} f(U_{i-1}^H S_{i-1} U_{i-1}) U_{i-1}^H Z_{i-1} \\ 0 \end{bmatrix} + f(S_i) Z_i \right\|_F .$$

This formulation allows us to check if the stopping criterion is satisfied without performing operations with vectors of length n, which results in some computational savings. Our implementation of Algorithm 8 employs this expression in its stopping criterion.

6.3.3 Comparison with other low-memory Krylov methods

In this section we briefly compare our method with other low-memory Krylov subspace methods from the literature, highlighting advantages and disadvantages of each method. As all comparison methods are tailored for computing the action of a matrix function on a vector, within this section, we adopt the assumption that b is equal to 1.

Similarly to multishift CG [55], Algorithm 8 is based on a rational approximant, so the attainable accuracy is ultimately limited by the available memory since the number of vectors that must be stored is proportional to the number of poles used in the rational approximation. However, while multishift CG requires an explicit rational approximant in partial fraction form, our method only needs the poles of the rational function. This makes Algorithm 8 easier to use, and less susceptible to numerical errors caused by the representation of the rational function.

Given the similarities between Algorithm 8 and multishift CG, it is useful to compare the cost of the two methods in more detail. For this purpose, we employ the simple computational model used in [69, Experiment 5.3], in which operations on vectors of length n such as scaling or addition are counted as one unit of work, denoted by $1\mathcal{V}$, and operations with vectors or matrices of size independent of n are not counted. As discussed in [69], when the underlying rational approximant has k poles the multishift CG algorithm must store $2k$ vectors of length n and perform approximately $5k\mathcal{V}$ operations in addition to the standard Lanczos algorithm. Algorithm 8 with k inner poles and compression every m outer iterations must store $k + m$ vectors of length n, and the only operation that involves vectors of length n outside of the standard Lanczos algorithm is in the compression step (line 8 in Algorithm 8), where the product $V_{i-1}U_{i-1}$ has to be computed for a cost of $2k(k + m)\mathcal{V}$ every m iterations, that on average amounts to $2km^{-1}(k + m)\mathcal{V}$ at each iteration. If we take $m = k$, so that Algorithm 8 and multishift CG have the same memory requirements and the same attainable accuracy, the cost of Algorithm 7 is $4k\mathcal{V}$ per iteration, which is smaller than the $5k\mathcal{V}$ cost of multishift CG. However, note that the efficiency of multishift CG can be improved by removing converged linear systems, i.e., by no longer updating the approximate solutions of linear systems when the residual becomes smaller than the requested tolerance [127, Section 5.3]. This can significantly reduce the cost of the method since linear systems associated with different poles often have substantially different convergence rates.

In contrast with Algorithm 8 and multishift CG, the memory requirements of the two-pass Lanczos method [21] are independent of the target accuracy, with the exception of the projected matrix, which grows in size at each iteration. When many iterations are needed to reach convergence, the computation of functions of projected matrices of increasing size can have a significant impact on the performance of the method, although this can be mitigated by only computing the approximate solution once every $d > 1$ iterations. For the same number of Lanczos iterations, the two-pass version requires twice the number of matrix-vector products with A compared to the other methods. However, in practice the cost is usually less than doubled, because in the second pass it is only necessary to recompute the Krylov basis vectors, and the orthogonalization coefficients have already been computed in the first pass.

The restarted Krylov method for Stieltjes matrix functions [53, 54] requires storing a number of vectors proportional to the restart length. Although the amount of memory available does not influence the accuracy attainable by this method, a shorter restart length can cause delays in the convergence, similarly to what happens when restarting Krylov subspace methods for the solution of linear systems. The convergence delay can be mitigated by employing deflation techniques [44]. We note that this restarted method explicitly requires the expression of the integrand function in the Stieltjes representation of f. The restarted method can also be applied to a function that is not Stieltjes by using a different integral representation, such as one based on the Cauchy integral formula. This was done in [54, Section 4.3] for the exponential function.

6.4 Numerical experiments

In this section, we compare Algorithm 8, which we denote by `RKcompress`, with other Krylov subspace methods for the computation of the action of a matrix function on a vector. For all methods, we monitor the relative norm of the difference between two consecutive computed solutions and stop when it becomes smaller than a requested tolerance.

The MATLAB code to reproduce the experiments in this section is available on GitHub at `https://github.com/casulli/ratkrylov-compress-matfun`. We use our own implementation of Algo-

Table 6.1: Number of iterations and final relative error for all the methods in Figure 6.1, using relative tolerance 10^{-10}.

t	10^{-5}	10^{-4}	10^{-3}	10^{-2}	10^{-1}
iter	26	70	212	631	1131
err	3.91e-12	6.92e-11	2.45e-10	1.03e-09	2.07e-09

rithm 8, two-pass Lanczos and multishift CG, while for the restarted Arnoldi for Stieltjes matrix functions we use the `funm_quad` implementation [54,117]. Although our implementation of Algorithm 8 has no external dependencies, to compute the rational approximant in partial fraction form employed in the multishift CG algorithm we use the implementation of the AAA algorithm [100] from the chebfun package [38]. All the experiments were performed with MATLAB R2021b on a laptop running Ubuntu 20.04, with 32 GB of RAM and an Intel Core i5-10300H CPU with clock rate 2.5 GHz, using a single thread.

6.4.1 Exponential function

As a first test problem, we consider the computation of $e^{-tA}C$, where the matrix $A \in \mathbb{C}^{n^2\times n^2}$ is the discretization of the 2D Laplace operator with zero Dirichlet boundary conditions using centered finite differences with $n+2$ points in each direction, that is

$$A = B \otimes I + I \otimes B, \quad \text{where} \quad B = \frac{1}{h^2}\begin{bmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix} \in \mathbb{C}^{n\times n}, \tag{6.16}$$

where $h = \frac{1}{n+1}$ and $C = C_1 \otimes C_2$ is the Kronecker product of two random block vectors of size $n \times b$. We compare the accuracy and timing of different methods based on Krylov subspaces using as a reference solution $e^{-tB}C_1 \otimes e^{-tB}C_2$, where the involved exponentials are computed using the MATLAB command `expm`. We set $n = 500$ (therefore the size of A is on the order of $10^5 \times 10^5$) and choose different values of t. As t increases, the eigenvalues of the matrix $-tA$ of which we compute the exponential become more spread out, so we expect an increasing number of iterations for the convergence of polynomial Krylov subspace methods (see, e.g., [12, Section 4]).

We use Algorithm 8 with $k = 25$ inner poles of the form described in [23], which guarantee a rational approximation of e^x for $x \in (-\infty, 0]$ with an absolute error of the order of machine precision. In particular, assuming that the involved matrix has no positive eigenvalues, the inner poles are independent of the spectrum of the matrix. We compare our `RKcompress` method with the standard Lanczos algorithm (`lanczos`), the two-pass version of Lanczos (`lanczos-2p`) and the Arnoldi algorithm with full orthogonalization (`arnoldi`). Note that both `lanczos` and `arnoldi` require storing the whole Krylov basis.

In Figure 6.1, we report the time needed to compute $e^{-tA}C$ with the different methods for different values of t, using a relative tolerance of 10^{-10} in the stopping criterion. For any fixed value of t, all the employed methods converge in the same number of iterations, which is reported in table 6.1 along with the final relative error attained. All the methods stop at the same iteration and produce the same approximate solution, confirming that Algorithm 8 is reproducing the convergence of the Lanczos algorithm (up to the error in the approximation of e^x with a rational function, which is negligible in this case). As the required number of iterations increases, our `RKcompress` method appears to be the fastest. This is mainly due to the fact that in Algorithm 8 the size of the matrix functions computed during the execution of the algorithm does not increase with the size of the Krylov subspace, in contrast with the other methods.

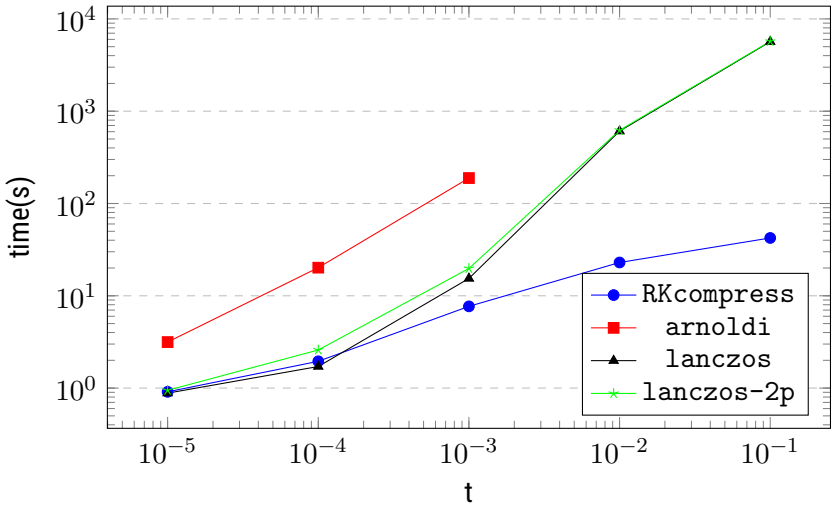


Figure 6.1: Time needed for the computation of $e^{-tA}C$ with accuracy of 10^{-10}, for different values of t and employing different methods.

6.4.2 Markov functions

As already shown in Section 5.4.2, an important class of functions for which rational approximations have been already described in the literature [12] is given by Markov functions, which can be represented as

$$f(z) = \int_{\alpha}^{\beta} \frac{d\mu(x)}{z - x}, \tag{6.17}$$

where $\mu(x)$ is a positive measure and $-\infty \leq \alpha < \beta < \infty$. If f is defined as in (6.17) and A is a Hermitian matrix with eigenvalues greater than β, we can use the approach in [12] to choose quasi-optimal inner poles for `RKcompress`. In particular, to obtain a rational approximation of f on an interval $[a, b]$ with $a > \beta$ with relative error norm bounded by ϵ it is sufficient to use k poles where

$$k \geq \log\left(\frac{4}{\epsilon}\right) \frac{\log(16(b-\beta)/(a-\beta))}{\pi^2}, \tag{6.18}$$

therefore the number of poles needed in `RKcompress` depends logarithmically on the condition number of $A - \beta I$. We refer to [12, Section 6.2] for more details regarding the choice of poles.

In the experiment that follows, we compute $A^{-1/2}\mathbf{1}$, where $A \in \mathbb{C}^{n^2 \times n^2}$ is a discretization of the 2D Laplace operator (6.16) and $\mathbf{1} \in \mathbb{C}^{n^2}$ is the vector of all ones, with increasing $n = 200, 400, \ldots, 1000$, comparing several low-memory Krylov subspace methods, using a relative tolerance of 10^{-8}. The reference solution is computed by diagonalizing A, exploiting the Kronecker sum structure. We compare Algorithm 8 (`RKcompress`) with the two-pass Lanczos method (`lanczos-2p`), the standard multishift CG method (`msCG`), a more efficient implementation of multishift CG with removal of converged linear systems (`msCG-rem`), and the restarted Krylov method for Stieltjes functions, both with and without deflation, denoted by `restart-defl` and `restart`, respectively.

For Algorithm 8, we use inner poles from [12] with k given by (6.18) and $m = k$. For a tolerance of 10^{-8}, the values of k range from $k = 23$ to $k = 29$ for the different matrix sizes. The rational approximant for multishift CG is obtained by running the AAA algorithm [100] with tolerance 10^{-12} on a discretization of the spectral interval of A; this produces a rational approximant with 18 poles for all matrix sizes, that are fewer than the ones obtained using (6.18), but it does not provide any theoretical guarantee on the approximation error. For the restarted Krylov method for Stieltjes functions, we set a restart length equal to $2k$ (so that the memory requirement is the same as `RKcompress`) and we use the default options in `funm_quad`; we retain 5 Ritz vectors in the variant with deflation.

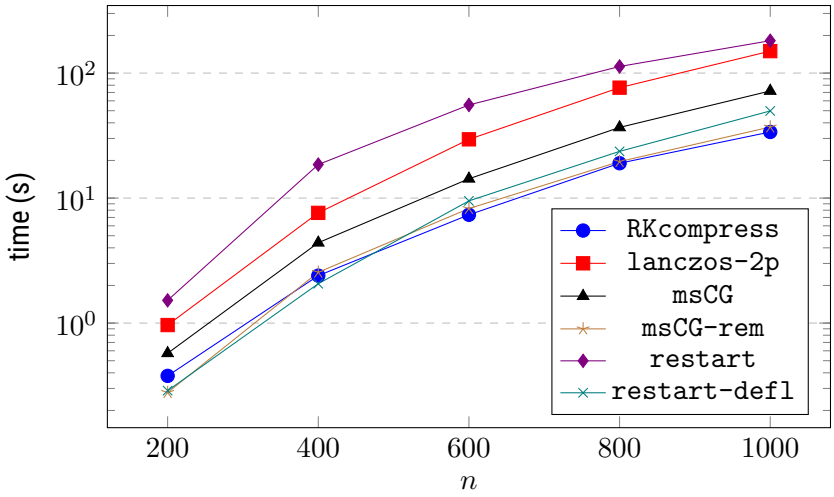


Figure 6.2: Time needed for the computation of $A^{-1/2}\mathbf{1}$ with relative tolerance 10^{-8}, where $A \in \mathbb{C}^{n^2 \times n^2}$ is a discretization of the 2D Laplace operator for increasing n, employing different low-memory methods.

Table 6.2: Final relative errors for the experiment in Figure 6.2.

size(A)	RKcompress	lanczos-2p	msCG	msCG-rem	restart	restart-defl
40000	9.01e-08	9.01e-08	7.81e-09	7.82e-09	9.75e-09	2.43e-11
160000	1.29e-07	1.29e-07	6.49e-09	6.50e-09	4.22e-08	1.20e-09
360000	1.70e-07	1.70e-07	2.69e-05	2.69e-05	4.53e-05	1.69e-09
640000	2.47e-07	2.47e-07	3.86e-06	3.86e-06	1.39e-03	4.50e-09
1000000	3.86e-07	3.86e-07	1.29e-06	1.29e-06	7.67e-03	2.34e-08

Table 6.3: Number of iterations for the experiment in Figure 6.2.

size(A)	RKcompress	lanczos-2p	msCG	msCG-rem	restart	restart-defl
40000	282	282	379	379	2760	460
160000	554	554	747	747	9050	900
360000	823	823	1122	1122	10 800	1512
640000	1085	1085	1497	1497	11 200	2184
1000000	1336	1336	1872	1872	11 600	3016

We report the times in Figure 6.2, the final errors in table 6.2 and the number of iterations in table 6.3. The methods with the shortest runtime are `RKcompress`, `msCG-rem` and `restart-defl`. Observe that `restarted-defl` requires significantly more iterations compared to the methods without restarting, so it would perform worse in a situation in which matrix-vector products with A are more expensive. Observe that for $n \geq 600$ the final error of multishift CG is significantly larger than the requested tolerance, suggesting that the rational approximant given by AAA is not accurate enough. However, it turns out that running `RKcompress` with inner poles given by the AAA approximant has approximately the same error as with the poles from [12], implying that the error in multishift CG should be mainly attributed to the explicit partial fraction representation of the rational function.

We note that the final error of `restart-defl` is smaller compared to the errors of `RKcompress` and `lanczos-2p`, since in the `funm_quad` implementation the approximate solutions are computed and compared only at the end of each restart cycle, hence the stopping criterion is more reliable and less likely to underestimate the error. The restarted Krylov method without deflation was unable to converge to the requested accuracy within 200 restart cycles for $n \geq 600$.

6.4.3 Numerical loss of orthogonality

In order to investigate the behavior of Algorithm 8 in finite precision arithmetic, we compute $e^A C$, where $A \in \mathbb{C}^{2000\times 2000}$ is a tridiagonal matrix with logspaced eigenvalues in the interval $[-10^4, -10^{-4}]$, and C is a random vector with unit normal entries. The convergence of `RKcompress`, `lanczos` and `arnoldi` are compared in Figure 6.3. Both `lanczos` and `RKcompress` exhibit a delay in convergence due to the loss of orthogonality in the Krylov basis and they are essentially indistinguishable, so it appears that Algorithm 8 also reproduces the finite precision behavior of the Lanczos algorithm.

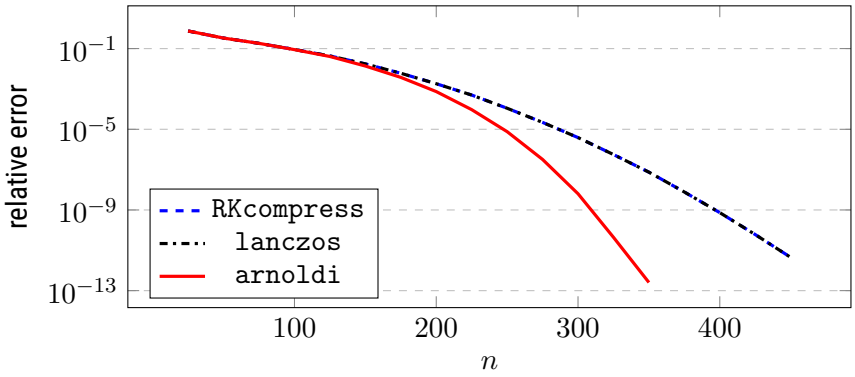


Figure 6.3: Effect of numerical loss of orthogonality in the computation of $e^A C$, where $A \in \mathbb{C}^{2000\times 2000}$ has logspaced eigenvalues in $[-10^4, -10^{-4}]$ and C is a random vector.

Chapter 7

Conclusion

In this dissertation, we have extended the basic attributes of rational Krylov methods to encompass their block counterpart. We have employed block rational Krylov methods to address Sylvester and tensor Sylvester equations, investigating the convergence of projection methods. Additionally, we have explored matrix function computations, aiming to develop a more innovative methodology.

We have proposed a method for solving low-rank Sylvester equations by means of projection onto block rational Krylov subspaces. The key advantage of the method with respect to state-of-the-art techniques is the possibility of exploiting the reordering of poles to maintain the "last" pole of the space equal to infinity. This choice makes the residual of the large-scale equation easily computable in the projected one, without the need to artificially increase the size of the subspace by introducing unnecessary poles at infinity. We have also reconsidered the convergence analysis for Krylov solvers for Sylvester equations of [9], extending it to block rational Krylov subspaces employing the theoretical tools used in [93] for the polynomial case. The analysis allows to design new strategies for adaptive pole selection, obtained by minimizing the norm of a small $b \times b$ rational matrix, where b is the block size. The minimization problem can be made simpler by replacing the norm with a surrogate function that is easier to evaluate. In [42] the authors propose a heuristic for the pole selection in block rational Krylov methods, based on their analysis of the non-block case. Choosing the determinant as surrogate function yields exactly this heuristic, and it gives a solid theoretical justification for this approach. Other choices, instead, yield completely novel strategies. One of these, called `sADM` in the thesis, has comparable or better performances than the state-of-the-art on the considered examples.

We expect that these results will facilitate the development of additional pole selection strategies and convergence analyses in block rational Krylov methods.

In the setting of tensor Sylvester equations, we have provided a characterization of tensorized block rational Krylov subspaces using multivariate rational functions and we have developed a method for solving tensor Sylvester equations with low multilinear or tensor train rank, based on projection onto a tensorized block rational Krylov subspace, providing a convergence analysis. Extending the findings established in the Sylvester case, we have devised approaches for pole selection and streamlined methods for computing the residual through pole reordering. Moreover, numerical tests have demonstrated the efficiency of this approach when compared with more traditional strategies based on polynomial and extended Krylov subspaces. Specifically, the presented method enables the handling of tensor Sylvester equations with a greater number of summands.

We expect that tensorized block rational Krylov subspaces can be used for solving more general high-dimensional tensor problems, such as the computation of functions of matrices with multiterm Kronecker structures.

In the context of matrix functions, we have developed an algorithm based on block rational Krylov subspaces for the computation of $f(A)$ for a Hermitian HSS matrix A. By generalizing the definition of telescopic decompositions, we have linked various representations of HSS matrices used in the literature and provided methods for converting between them. This new representation has allowed us to exploit the nested low-rank structure of telescopic decompositions for the computation of an approximation of $f(A)$.

Our convergence results imply nearly linear complexity for matrix exponentials and linear-polylogarithmic complexity for inverse square roots in situations of practical relevance. This favorable complexity is attained by using block rational Krylov subspaces that involve small-sized matrices only, avoiding the solution of potentially large linear systems usually associated with rational Krylov subspace techniques. Several numerical experiments show that our newly proposed algorithm is faster than existing algorithms for a variety of examples previously reported in the literature. Somewhat surprisingly, it even appears to be the method of choice for computing inverses of Hermitian HSS matrices. Several questions remain open. This includes the extension to non Hermitian matrices as well as a theoretical explanation of the good results obtained for the sign function.

Finally, we have presented a memory-efficient method for the computation of $f(A)C$ for a Hermitian matrix A. The method combines an outer block Lanczos method with inner block rational Krylov iterations, that are used to compress the Lanczos basis and reduce memory requirements. The construction of the inner block rational Krylov basis only involves operations with small projected matrices and it does not require any operation with the matrix A. We have proved that our algorithm coincides with the outer Krylov method when f is a rational function. In the general case, the error depends on a rational approximation of f. Our numerical experiments show that the proposed algorithm is competitive with other low-memory methods based on Krylov subspaces. The possibility of extending the proposed approach to non-Hermitian matrices remains an open problem.

Bibliography

[1] L. Aceto, D. Bertaccini, F. Durastante, and P. Novati. Rational Krylov methods for functions of matrices with applications to fractional partial differential equations. *J. Comput. Phys.*, 396:470–482, 2019.

[2] Roman Andreev and Christine Tobler. Multilevel preconditioning and low-rank tensor iteration for space–time simultaneous discretizations of parabolic PDEs. *Numerical Linear Algebra with Applications*, 22(2):317–337, 2015.

[3] Athanasios C. Antoulas. *Approximation of Large-Scale Dynamical Systems*. Society for Industrial and Applied Mathematics, 2005.

[4] Markus Bachmayr and Wolfgang Dahmen. Adaptive near-optimal rank tensor approximation for high-dimensional operator equations. *Foundations of Computational Mathematics*, 15:839–898, 2015.

[5] Jonas Ballani. Fast evaluation of singular bem integrals based on tensor approximations. *Numerische Mathematik*, 121(3):433–460, 2012.

[6] Jonas Ballani and Lars Grasedyck. A projection method to solve linear systems in tensor format. *Numerical linear algebra with applications*, 20(1):27–43, 2013.

[7] Richard H. Bartels and George W Stewart. Solution of the matrix equation AX+ XB= C. *Communications of the ACM*, 15(9):820–826, 1972.

[8] Ulrike Baur and Peter Benner. Factorized solution of Lyapunov equations based on hierarchical matrix arithmetic. *Computing*, 78:211–234, 2006.

[9] Bernhard Beckermann. An error analysis for rational Galerkin projection applied to the Sylvester equation. *SIAM Journal on Numerical Analysis*, 49(6):2430–2450, 2011.

[10] Bernhard Beckermann, Alice Cortinovis, Daniel Kressner, and Marcel Schweitzer. Low-rank updates of matrix functions II: rational Krylov methods. *SIAM J. Numer. Anal.*, 59(3):1325–1347, 2021.

[11] Bernhard Beckermann, Daniel Kressner, and Christine Tobler. An error analysis of Galerkin projection methods for linear systems with tensor product structure. *SIAM Journal on Numerical Analysis*, 51(6):3307–3326, 2013.

[12] Bernhard Beckermann and Lothar Reichel. Error estimates and evaluation of matrix functions via the Faber transform. *SIAM J. Numer. Anal.*, 47(5):3849–3883, 2009.

[13] Bernhard Beckermann and Alex Townsend. On the singular values of matrices with displacement structure. *SIAM Journal on Matrix Analysis and Applications*, 38(4):1227–1248, 2017.

[14] Peter Benner, Serkan Gugercin, and Karen Willcox. A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM Review*, 57(4):483–531, 2015.

[15] Peter Benner, Ren-Cang Li, and Ninoslav Truhar. On the ADI method for Sylvester equations. *Journal of Computational and Applied Mathematics*, 233(4):1035–1045, 2009.

[16] Michele Benzi, Paola Boito, and Nader Razouk. Decay properties of spectral projectors with applications to electronic structure. *SIAM Review*, 55(1):3–64, 2013.

[17] Michele Benzi and Valeria Simoncini. Decay bounds for functions of Hermitian matrices with banded or Kronecker structure. *SIAM J. Matrix Anal. Appl.*, 36(3):1263–1282, 2015.

[18] Michele Benzi and Igor Simunec. Rational Krylov methods for fractional diffusion problems on graphs. *BIT*, 62(2):357–385, 2022.

[19] Mario Berljafa, Steven Elsworth, and Stefan Güttel. A rational Krylov toolbox for MATLAB. *MIMS EPrint 2014.56, Manchester Institute for Mathematical Sciences, University of Manchester, UK, 2014*, 2014. Available for download at http://rktoolbox.org.

[20] Dario A Bini, Guy Latouche, and Beatrice Meini. *Numerical methods for structured Markov chains*. OUP Oxford, 2005.

[21] Artan Boriçi. Fast methods for computing the neuberger operator. In Andreas Frommer, Thomas Lippert, Björn Medeke, and Klaus Schilling, editors, *Numerical Challenges in Lattice Quantum Chromodynamics*, pages 40–47, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[22] R. Bouyouli, K. Jbilou, R. Sadaka, and H. Sadok. Convergence properties of some block Krylov subspace methods for multiple linear systems. *J. Comput. Appl. Math.*, 196(2):498–511, 2006.

[23] A. J. Carpenter, A. Ruttan, and R. S. Varga. Extended numerical computations on the "$1/9$" conjecture in rational approximation theory. In *Rational approximation and interpolation (Tampa, Fla., 1983)*, volume 1105 of *Lecture Notes in Math.*, pages 383–411. Springer, Berlin, 1984.

[24] Angelo A. Casulli. Tensorized block rational Krylov methods for tensor Sylvester equations. *arXiv preprint arXiv:2306.00705*, 2023.

[25] Angelo A. Casulli, Daniel Kressner, and Leonardo Robol. Computing functions of symmetric hierarchically semiseparable matrices. *arXiv preprint arXiv:2402.17369*, 2024.

[26] Angelo A. Casulli and Leonardo Robol. An efficient block rational Krylov solver for Sylvester equations with adaptive pole selection. *SIAM Journal on Scientific Computing*, 46(2):A798–A824, 2024.

[27] Angelo A. Casulli and Igor Simunec. Computation of generalized matrix functions with rational Krylov methods. *Math. Comp.*, 92(340):749–777, 2023.

[28] Angelo A. Casulli and Igor Simunec. A low-memory Lanczos method with rational Krylov compression for matrix functions. *arXiv preprint arXiv:2403.04390*, 2024.

[29] Shiv Chandrasekaran, Ming Gu, and Timothy Pals. A fast ULV decomposition solver for hierarchically semiseparable representations. *SIAM J. Matrix Anal. Appl.*, 28(3):603–622, 2006.

[30] Minhong Chen and Daniel Kressner. Recursive blocked algorithms for linear systems with Kronecker product structure. *Numerical Algorithms*, 84:1199–1216, 2020.

[31] Tyler Chen, Anne Greenbaum, Cameron Musco, and Christopher Musco. Low-memory Krylov subspace methods for optimal rational matrix function approximation. *SIAM J. Matrix Anal. Appl.*, 44(2):670–692, 2023.

[32] Sambasiva Rao Chinnamsetty, Wolfgang Hackbusch, and Heinz-Jürgen Flad. Efficient multi-scale computation of products of orbitals in electronic structure calculations. *Computing and visualization in science*, 13(8):397–408, 2010.

[33] Alice Cortinovis, Daniel Kressner, and Stefano Massei. Divide-and-conquer methods for functions of matrices with banded or hierarchical low-rank structure. *SIAM J. Matrix Anal. Appl.*, 43(1):151–177, 2022.

[34] Jane Cullum and William E Donath. A block Lanczos algorithm for computing the q algebraically largest eigenvalues and a corresponding eigenspace of large, sparse, real symmetric matrices. In *1974 IEEE Conference on Decision and Control including the 13th Symposium on Adaptive Processes*, pages 505–509. IEEE, 1974.

[35] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. A multilinear singular value decomposition. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000.

[36] Sergey V Dolgov. TT-GMRES: solution to a linear system in the structured tensor format. *Russian Journal of Numerical Analysis and Mathematical Modelling*, 28(2):149–172, 2013.

[37] Sergey V Dolgov and Dmitry V Savostyanov. Alternating minimal energy methods for linear systems in higher dimensions. *SIAM Journal on Scientific Computing*, 36(5):A2248–A2271, 2014.

[38] T. A Driscoll, N. Hale, and L. N. Trefethen. *Chebfun Guide*. Pafnuty Publications, 2014.

[39] Vladimir Druskin, Stefan Güttel, and Leonid Knizhnerman. Near-optimal perfectly matched layers for indefinite helmholtz problems. *SIAM Review*, 58(1):90–116, 2016.

[40] Vladimir Druskin and Leonid Knizhnerman. Extended Krylov subspaces: approximation of the matrix square root and related functions. *SIAM J. Matrix Anal. Appl.*, 19(3):755–771, 1998.

[41] Vladimir Druskin, Chad Lieberman, and Mikhail Zaslavsky. On adaptive choice of shifts in rational Krylov subspace reduction of evolutionary problems. *SIAM Journal on Scientific Computing*, 32(5):2485–2496, 2010.

[42] Vladimir Druskin and Valeria Simoncini. Adaptive rational Krylov subspaces for large-scale dynamical systems. *Systems & Control Letters*, 60(8):546–560, 2011.

[43] Michael Eiermann and Oliver G. Ernst. A restarted Krylov subspace method for the evaluation of matrix functions. *SIAM J. Numer. Anal.*, 44(6):2481–2504, 2006.

[44] Michael. Eiermann, Oliver. G. Ernst, and Stefan. Güttel. Deflated restarting for matrix functions. *SIAM J. Matrix Anal. Appl.*, 32(2):621–641, 2011.

[45] L. Elbouyahyaoui, A. Messaoudi, and H. Sadok. Algebraic properties of the block GMRES and block Arnoldi methods. *Electron. Trans. Numer. Anal.*, 33:207–220, 2008/09.

[46] Nancy S Ellner and Eugene L Wachspress. New ADI model problem applications. In *Proceedings of 1986 ACM Fall joint computer conference*, pages 528–534, 1986.

[47] Howard C Elman, David J Silvester, and Andrew J Wathen. *Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics*. Oxford university press, 2014.

[48] Steven Elsworth and Stefan Güttel. The block rational Arnoldi method. *SIAM J. Matrix Anal. Appl.*, 41(2):365–388, 2020.

[49] Ernesto Estrada and Desmond J Higham. Network properties revealed through matrix functions. *SIAM Review*, 52(4):696–714, 2010.

[50] Lukas Exl, Claas Abert, Norbert J Mauser, Thomas Schrefl, Hans Peter Stimming, and Dieter Suess. Fft-based kronecker product approximation to micromagnetic long-range interactions. *Mathematical Models and Methods in Applied Sciences*, 24(09):1877–1901, 2014.

[51] Lukas Exl, Johann Fischbacher, Alexander Kovacs, Harald Oezelt, Markus Gusenbauer, and Thomas Schrefl. Preconditioned nonlinear conjugate gradient method for micromagnetic energy minimization. *Computer Physics Communications*, 235:179–186, 2019.

[52] Daniel Fortunato and Alex Townsend. Fast Poisson solvers for spectral methods. *IMA Journal of Numerical Analysis*, 40(3):1994–2018, 2020.

[53] Andreas Frommer, Stefan Güttel, and Marcel Schweitzer. Convergence of restarted Krylov subspace methods for Stieltjes functions of matrices. *SIAM J. Matrix Anal. Appl.*, 35(4):1602–1624, 2014.

[54] Andreas Frommer, Stefan Güttel, and Marcel Schweitzer. Efficient and stable Arnoldi restarts for matrix functions based on quadrature. *SIAM J. Matrix Anal. Appl.*, 35(2):661–683, 2014.

[55] Andreas Frommer and Peter Maass. Fast CG-based methods for Tikhonov-Phillips regularization. *SIAM J. Sci. Comput.*, 20(5):1831–1850, 1999.

[56] Andreas Frommer, Claudia Schimmel, and Marcel Schweitzer. Analysis of probing techniques for sparse approximation and trace estimation of decaying matrix functions. *SIAM J. Matrix Anal. Appl.*, 42(3):1290–1318, 2021.

[57] Andreas Frommer and Valeria Simoncini. Matrix functions. In *Model order reduction: theory, research aspects and applications*, volume 13 of *Math. Ind.*, pages 275–303. Springer, Berlin, 2008.

[58] Israel Gohberg, Peter Lancaster, and Leiba Rodman. *Matrix polynomials*. Springer, 2005.

[59] Gene Golub, Stephen Nash, and Charles Van Loan. A Hessenberg-Schur method for the problem AX+ XB= C. *IEEE Transactions on Automatic Control*, 24(6):909–913, 1979.

[60] Gene H Golub and Richard Underwood. The block Lanczos method for computing eigenvalues. In *Mathematical software*, pages 361–377. Elsevier, 1977.

[61] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD, fourth edition, 2013.

[62] A. A. Gonchar and E. A. Rakhmanov. Equilibrium distributions and the rate of rational approximation of analytic functions. *Mat. Sb. (N.S.)*, 134(176)(3):306–352, 447, 1987.

[63] Lars Grasedyck. Existence and computation of low Kronecker-rank approximations for large linear systems of tensor product structure. *Computing. Archives for Informatics and Numerical Computation*, 72(3-4):247, 2004.

[64] Lars Grasedyck, Wolfgang Hackbusch, and Boris N. Khoromskij. Solution of large scale algebraic matrix Riccati equations by use of hierarchical matrices. *Computing*, 70(2):121–165, 2003.

[65] Lars Grasedyck, Daniel Kressner, and Christine Tobler. A literature survey of low-rank tensor approximation techniques. *GAMM-Mitteilungen*, 36(1):53–78, 2013.

[66] Stefan Güttel. *Rational Krylov methods for operator functions*. PhD thesis, Technische Universitat Bergakademie Freiberg, 2010.

[67] Stefan Güttel. Rational Krylov approximation of matrix functions: numerical methods and optimal pole selection. *GAMM-Mitt.*, 36(1):8–31, 2013.

[68] Stefan Güttel, Daniel Kressner, and Kathryn Lund. Limited-memory polynomial methods for large-scale matrix functions. *GAMM-Mitt.*, 43(3):e202000019, 19, 2020.

[69] Stefan Güttel and Marcel Schweitzer. A comparison of limited-memory Krylov methods for Stieltjes functions of Hermitian matrices. *SIAM J. Matrix Anal. Appl.*, 42(1):83–107, 2021.

[70] Wolfgang Hackbusch. *Hierarchical matrices: algorithms and analysis*, volume 49 of *Springer Series in Computational Mathematics*. Springer, Heidelberg, 2015.

[71] Diana Halikias and Alex Townsend. Structured matrix recovery from matrix-vector products. *Numer. Linear Algebra Appl.*, 31(1):Paper No. e2531, 27, 2024.

[72] Johan Håstad. Tensor rank is NP-complete. *Journal of Algorithms*, 11(4):644–654, 1990.

[73] M. Heyouni and A. Essai. Matrix Krylov subspace methods for linear systems with multiple right-hand sides. *Numer. Algorithms*, 40(2):137–156, 2005.

[74] Nicholas J. Higham. *Functions of Matrices: Theory and Computation*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2008.

[75] Nicholas J. Higham. The scaling and squaring method for the matrix exponential revisited. *SIAM Review*, 51(4):747–764, 2009.

[76] Marlis Hochbruck and Christian Lubich. On Krylov subspace approximations to the matrix exponential operator. *SIAM J. Numer. Anal.*, 34(5):1911–1925, 1997.

[77] Marlis Hochbruck and Christian Lubich. On Krylov subspace approximations to the matrix exponential operator. *SIAM J. Numer. Anal.*, 34(5):1911–1925, 1997.

[78] K. Jbilou, A. Messaoudi, and H. Sadok. Global FOM and GMRES algorithms for matrix equations. *Appl. Numer. Math.*, 31(1):49–63, 1999.

[79] Mark David Kent. *Chebyshev, Krylov, Lanczos: matrix relationships and computations*. Stanford University, 1989.

[80] Boris N Khoromskij and Christoph Schwab. Tensor-structured Galerkin approximation of parametric and stochastic elliptic PDEs. *SIAM Journal on Scientific Computing*, 33(1):364–385, 2011.

[81] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009.

[82] Daniel Kressner, Stefano Massei, and Leonardo Robol. Low-rank updates and a divide-and-conquer method for linear matrix equations. *SIAM Journal on Scientific Computing*, 41(2):A848–A876, 2019.

[83] Daniel Kressner and Christine Tobler. Krylov subspace methods for linear systems with tensor product structure. *SIAM Journal on Matrix Analysis and Applications*, 31(4):1688–1714, 2010.

[84] Daniel Kressner and Christine Tobler. Low-rank tensor Krylov subspace methods for parametrized linear systems. *SIAM Journal on Matrix Analysis and Applications*, 32(4):1288–1316, 2011.

[85] Daniel Kressner and Ana Šušnjara. Fast computation of spectral projectors of banded matrices. *SIAM J. Matrix Anal. Appl.*, 38(3):984–1009, 2017.

[86] Peter Lancaster. Explicit solutions of linear matrix equations. *SIAM Review*, 12(4):544–566, 1970.

[87] Peter Lancaster and Françoise Tisseur. Hermitian quadratic matrix polynomials: Solvents and inverse problems. *Linear algebra and its applications*, 436(10):4017–4026, 2012.

[88] Cornelius Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Research Nat. Bur. Standards*, 45:255–282, 1950.

[89] Spike T Lee, Hong-Kui Pang, and Hai-Wei Sun. Shift-invert arnoldi approximation to the toeplitz matrix exponential. *SIAM Journal on Scientific Computing*, 32(2):774–792, 2010.

[90] James Levitt and Per-Gunnar Martinsson. Linear-complexity black-box randomized compression of hierarchically block separable matrices. *arXiv preprint arXiv:2205.02990*, 2022.

[91] Jing-Rebecca Li and Jacob White. Low rank solution of Lyapunov equations. *SIAM Journal on Matrix Analysis and Applications*, 24(1):260–280, 2002.

[92] Luciano Lopez and Valeria Simoncini. Preserving geometric properties of the exponential matrix by block Krylov subspace methods. *BIT*, 46(4):813–830, 2006.

[93] Kathryn Lund. *A new block Krylov subspace framework with applications to functions of matrices acting on multiple vectors*. Temple University, 2018.

[94] Per-Gunnar Martinsson. A fast randomized algorithm for computing a hierarchically semiseparable representation of a matrix. *SIAM J. Matrix Anal. Appl.*, 32(4):1251–1274, 2011.

[95] Stefano Massei, Mariarosa Mazza, and Leonardo Robol. Fast solvers for two-dimensional fractional diffusion equations using rank structured matrices. *SIAM J. Sci. Comput.*, 41(4):A2627–A2656, 2019.

[96] Stefano Massei and Leonardo Robol. A nested divide-and-conquer method for tensor Sylvester equations with positive definite hierarchically semiseparable coefficients. *IMA Journal of Numerical Analysis*, page drad089, 12 2023.

[97] Stefano Massei, Leonardo Robol, and Daniel Kressner. hm-toolbox: MATLAB software for HODLR and HSS matrices. *SIAM J. Sci. Comput.*, 42(2):C43–C68, 2020.

[98] Mark M. Meerschaert and Charles Tadjeran. Finite difference approximations for fractional advection-dispersion flow equations. *J. Comput. Appl. Math.*, 172(1):65–77, 2004.

[99] I. Moret and P. Novati. RD-rational approximations of the matrix exponential. *BIT*, 44(3):595–615, 2004.

[100] Yuji Nakatsukasa, Olivier Sète, and Lloyd N. Trefethen. The AAA algorithm for rational approximation. *SIAM J. Sci. Comput.*, 40(3):A1494–A1522, 2018.

[101] Ivan V Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.

[102] Davide Palitta, Stefano Pozza, and Valeria Simoncini. The short-term rational Lanczos method and applications. *SIAM Journal on Scientific Computing*, 44(4):A2843–A2870, 2022.

[103] Davide Palitta and Valeria Simoncini. Matrix-equation-based strategies for convection-diffusion equations. *BIT*, 56(2):751–776, 2016.

[104] Davide Palitta and Valeria Simoncini. Numerical methods for large-scale Lyapunov equations with symmetric banded data. *SIAM Journal on Scientific Computing*, 40(5):A3581–A3608, 2018.

[105] Taejun Park and Yuji Nakatsukasa. Approximating sparse matrices and their functions using matrix-vector products. *arXiv preprint arXiv:2310.05625*, 2023.

[106] Penco Petrov Petrushev and Vasil Atanasov Popov. *Rational approximation of real functions*, volume 28 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, 1987.

[107] Somaiyeh Rashedi, Ghodrat Ebadi, Sebastian Birk, and Andreas Frommer. On short recurrence Krylov type methods for linear systems with many right-hand sides. *J. Comput. Appl. Math.*, 300:18–29, 2016.

[108] Axel Ruhe. Rational Krylov sequence methods for eigenvalue computation. *Linear Algebra and its Applications*, 58:391–405, 1984.

[109] Axel Ruhe. The rational Krylov algorithm for nonsymmetric eigenvalue problems. iii: Complex shifts for real matrices. *BIT Numerical Mathematics*, 34(1):165–176, 1994.

[110] Axel Ruhe. Rational Krylov algorithms for nonsymmetric eigenvalue problems. In *Recent advances in iterative methods*, pages 149–164. Springer, 1994.

[111] Y. Saad. Analysis of some Krylov subspace approximations to the matrix exponential operator. *SIAM J. Numer. Anal.*, 29(1):209–228, 1992.

[112] Youcef Saad. On the Lánczos method for solving symmetric linear systems with several right-hand sides. *Math. Comp.*, 48(178):651–662, 1987.

[113] Youcef Saad. Numerical solution of large Lyapunov equations. Technical report, 1989.

[114] Youcef Saad and Martin H Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing*, 7(3):856–869, 1986.

[115] Yousef Saad, James Chelikowsky, and Suzanne Shontz. Numerical methods for electronic structure calculations of materials. *SIAM Review*, 52:3–54, 03 2010.

[116] John Sabino. Solution of large-scale Lyapunov equations via the block modified smith method. *Rice University, Houston*, 2006.

[117] Marcel Schweitzer, Stefan Güttel, and Andreas Frommer. FUNM_QUAD: An implementation of a stable, quadrature based restarted Arnoldi method for matrix functions. *Technical Report*, IMACM Preprint 14/04, 2014, 2014.

[118] Tianyi Shi and Alex Townsend. On the compressibility of tensors. *SIAM Journal on Matrix Analysis and Applications*, 42(1):275–298, 2021.

[119] Valeria Simoncini. Ritz and pseudo-Ritz values using matrix polynomials. *Linear algebra and its applications*, 241:787–801, 1996.

[120] Valeria Simoncini. A new iterative method for solving large-scale Lyapunov matrix equations. *SIAM Journal on Scientific Computing*, 29(3):1268–1288, 2007.

[121] Valeria Simoncini. Computational methods for linear matrix equations. *SIAM Review*, 58(3):377–441, 2016.

[122] Valeria Simoncini and Efstratios Gallopoulos. Convergence properties of block GMRES and matrix polynomials. *Linear Algebra and its Applications*, 247:97–119, 1996.

[123] Christoph Strössner and Daniel Kressner. Fast global spectral methods for three-dimensional partial differential equations. *IMA Journal of Numerical Analysis*, 43(3):1519–1542, 2023.

[124] Alex Townsend and Sheehan Olver. The automatic solution of partial differential equations using a global spectral method. *Journal of Computational Physics*, 299:106–123, 2015.

[125] Alex Townsend and Heather Wilber. On the singular values of matrices with high displacement rank. *Linear Algebra Appl.*, 548:19–41, 2018.

[126] Ledyard R Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.

[127] J. van den Eshof, A. Frommer, Th. Lippert, K. Schilling, and H.A. van der Vorst. Numerical methods for the QCDd overlap operator. i. sign-function and error bounds. *Computer Physics Communications*, 146(2):203–224, 2002.

[128] David S Watkins. *Fundamentals of matrix computations*. John Wiley & Sons, 2004.

[129] Jianlin Xia, Shivkumar Chandrasekaran, Ming Gu, and Xiaoye S. Li. Fast algorithms for hierarchically semiseparable matrices. *Numer. Linear Algebra Appl.*, 17(6):953–976, 2010.