A nested divide-and-conquer method for tensor Sylvester equations with positive definite hierarchically semiseparable coefficients

Stefano Massei* Leonardo Robol*

Abstract

Linear systems with a tensor product structure arise naturally when considering the discretization of Laplace type differential equations or, more generally, multidimensional operators with separable coefficients. In this work, we focus on the numerical solution of linear systems of the form

$$(I \otimes \cdots \otimes I \otimes A_1 + \cdots + A_d \otimes I \otimes \cdots \otimes I)\, x = b,$$

where the matrices $A_t \in \mathbb{R}^{n\times n}$ are symmetric positive definite and belong to the class of hierarchically semiseparable matrices.

We propose and analyze a nested divide-and-conquer scheme, based on the technology of low-rank updates, that attains the quasi-optimal computational cost $\mathcal{O}(n^d \log(n))$. Our theoretical analysis highlights the role of inexactness in the nested calls of our algorithm and provides worst case estimates for the amplification of the residual norm. The performances are validated on 2D and 3D case studies.

Keywords Tensor equation, Sylvester equation, Divide and conquer, Rational approximation.

Mathematics Subject Classification 15A06, 65F10, 65Y20

1 Introduction

In this work we are concerned with the numerical solution of linear systems with a Kronecker sum-structured coefficient matrix of the form:

$$(I \otimes \cdots \otimes I \otimes A_1 + \cdots + A_d \otimes I \otimes \cdots \otimes I)\, x = b, \tag{1}$$

where the matrices $A_t \in \mathbb{R}^{n_t\times n_t}$ are symmetric and positive definite (SPD) with spectrum contained in $[\alpha_t, \beta_t] \subset \mathbb{R}^+ := \{z \in \mathbb{R} \text{ s.t. } z > 0\}$, and have low-rank off-diagonal blocks, for $t = 1, \dots, d$. By reshaping $x, b \in \mathbb{R}^{n_1 \cdots n_d}$ into d-dimensional tensors $\mathcal{X}, \mathcal{B} \in \mathbb{R}^{n_1\times\cdots n_d}$, we rewrite (1) as the tensor Sylvester equation

$$\mathcal{X} \times_1 A_1 + \cdots + \mathcal{X} \times_d A_d = \mathcal{B}, \tag{2}$$

where $\times_j$ denotes the j-mode product for tensors [17].

*Department of Mathematics, University of Pisa. E-mails: stefano.massei@unipi.it, leonardo.robol@unipi.it. Both authors are members of the INdAM/GNCS research group.

Tensor Sylvester equations, not necessarily with SPD coefficients, arise naturally when discretizing d-dimensional Laplace-like operators by means of tensorized grids that respect the separability of the operator [13, 22, 26, 30, 31]. In the case $d = 2$, we recover the well-known case of matrix Sylvester equations, that also plays a dominant role in the model reduction of dynamical systems [1].

Several methods for solving matrix and tensor Sylvester equations assume that the right-hand side has some structure, such as being low-rank. This is a necessary assumption for dealing with large scale problems. In this paper, we only consider unstructured right-hand sides $\mathcal{B}$, for which the cases of interest are those where the memory cost $\mathcal{O}(\prod_{t=1}^{d} n_t)$ still allows to store $\mathcal{B}$ and the solution $\mathcal{X}$. Note that, this limits the scenarios where our algorithm is effective to small values of d, i.e. $d = 2, 3$. The structure in the coefficients A_t (which are SPD and have off-diagonal blocks of low-rank), will be crucial to improve the complexity of the solver with respect to the completely unstructured case. We also remark that when the A_ts arise from the discretization of elliptic differential operators, the structure assumed in this work is often present [8, 15].

1.1 Related work

In the matrix case (i.e., $d = 2$ in (2)) there are two main procedures that make no assumptions on A_1, A_2, and $\mathcal{B}$: the Bartels-Stewart algorithm [3, 16] and the Hessenberg-Schur method [14]. These are based on taking the coefficients A_1, A_2 to either Hessenberg or triangular form, and then solving the linear system by (block) back-substitution. The idea has been generalized to d-dimensional tensor Sylvester equations in [9]. In the case where $n = n_1 = \ldots = n_d$, the computational complexity of these approaches is $\mathcal{O}(dn^3 + n^{d+1})$ flops.

When the right-hand side $\mathcal{B}$ is a low-rank matrix or is representable in a low-rank tensor format (Tucker, Tensor-Train, Hierarchical Tucker, ...) the tensor equation can be solved much more efficiently, and the returned approximate solution is low-rank, which allow us to store it in a low-memory format. Indeed, in this case it is possible to exploit tensorized Krylov (and rational Krylov) methods [12, 19, 29], or the factored Alternating Direction Implicit Method (fADI) [6]. The latter methods build a rank s approximant to the solution $\mathcal{X}$ by solving a $\mathcal{O}(s)$ shifted linear systems with the matrices A_t. This is very effective when also the coefficients A_t are structured. For instance, when the A_t are sparse or hierarchically low-rank, this often brings the cost of approximating X to $\mathcal{O}(sn \log^a n)$ for $a \in \{0, 1, 2\}$ [15]. In the tensor case, another option is to rely on methods adapted to the low-rank structure under consideration: AMEn [11] or TT-GMRES [10] for Tensor-Trains, projection methods in the Hierarchical Tucker format [2], and other approaches.

In this work, we consider an intermediate setting, where the coefficients A_t are structured, while the right-hand side $\mathcal{B}$ is not. More specifically, we assume that the A_t are SPD and efficiently representable in the Hierarchical SemiSeparable format (HSS) [33]. This implies that each coefficient A_t can be partitioned in a 2×2 block matrix with low-rank off-diagonal blocks, and diagonal blocks with the same recursive structure.

A particular case of this setting has been considered in [13], where the A_t are banded SPD (and therefore have low-rank off-diagonal blocks), and a nested Alternating Direction Implicit (ADI) solver is applied to a 3D tensor equation with no structure in $\mathcal{B}$. The complexity of the algorithm is quasi-optimal $\mathcal{O}(n^3 \log^3 n)$, but the hidden constant is very large, and the approach is not practical already for moderate n; see [30] for a comparison with methods with a higher asymptotic complexity.

We remark that the tensor equation (2) can be solved by diagonalization of the A_ts in a stable way, as described in the pseudocode of Algorithm 1. Without further assumptions, this costs $\mathcal{O}(dn^3 + n^{d+1})$ when all dimensions are equal.

Algorithm 1

procedure LYAPND_DIAG($A_1, A_2, \ldots, A_d, \mathcal{B}$)
  for $i = 1, \ldots, d$ **do**
    $[S_i, D_i] = \texttt{eig}(A_i)$
    $\mathcal{B} \leftarrow \mathcal{B} \times_i S_i^*$
  end for
  for $i_1 = 1, \ldots, n_1, \ldots, i_d = 1, \ldots, n_d$ **do**
    $\mathcal{X}(i_1, \ldots, i_d) \leftarrow \mathcal{B}(i_1, \ldots, i_d) \cdot ([D_1]_{i_1 i_1} + \cdots + [D_d]_{i_d i_d})^{-1}$
  end for
  for $i = 1, \ldots, d$ **do**
    $\mathcal{X} \leftarrow \mathcal{X} \times_i S_i$
  end for
  return $\mathcal{X}$
end procedure

If the A_t can be efficiently diagonalized, then Algorithm 1 attains a quasi-optimal complexity. For instance, in the case of finite difference discretizations of the d-dimensional Laplace operator, diagonalizing the matrices A_t via the fast sine or cosine transforms (depending on the boundary conditions) yields the complexity $\mathcal{O}(n^d \log n)$. Recently, it has been shown that positive definite HSS enjoy a structured eigendecomposition [25], that can be retrieved in $\mathcal{O}(n \log^2 n)$ time. In addition, multiplying a vector by the eigenvector matrix costs only $\mathcal{O}(n \log(n))$ because the latter can be represented as a product of permutations and Cauchy-like matrices, of logarithmic length. These features can be exploited into Algorithm 1 to obtain an efficient solver. The approach proposed in this work has the same $\mathcal{O}(n^d \log n)$ asymptotic complexity but, as we will demonstrate in our numerical experiments, will result in significantly lower computational costs.

1.2 Main contributions

The main contribution is the design and analysis of an algorithm with $\mathcal{O}(\prod_{j=1}^{d} n_j \log(\max_j n_j))$ complexity for solving the tensor equation (2) with HSS and SPD coefficients A_t. The algorithm is based on a divide-and-conquer scheme, where (2) is decomposed into several tensor equations that have either a low-rank right-hand side, or a small dimension. In the tensor case, the low-rank equations are solved exploiting nested calls to the $(d-1)$-dimensional solver. Concerning the theoretical analysis, we provide the following contributions:

- An error analysis that, assuming a bounded residual error on the low-rank subproblems, guarantees the accuracy on the final result of the divide-and-conquer scheme (Theorem 3.6 and Lemma 4.1).
- A novel a priori error analysis for the use of fADI with inexact solves; more precisely, in Theorem 4.3 we provide an explicit bound for the difference between the residual norm after s steps of fADI in exact arithmetic, and the one obtained by performing the fADI steps with inexact solves. This enables us to control the residual norm of the error based only on the number of shifts used in all calls to fADI in our solver (Theorem 4.4). These results are very much related to those in [20, Theorem 3.4 and Corollary 3.1], where also the convergence of fADI with inexact solves is analyzed. Nevertheless, the assumptions and the techniques used in the proofs of such results are quite different. The goal of [20] is to progressively increase the level of inexactness, along the iterations of fADI, ensuring that the final residual norm remain below a target threshold. The authors proposed an adaptive relaxation strategy that requires the computation of intermediate quantities generated during the execution of the algorithm. In our work, the level of inexactness is fixed and,

by exploiting the decay of the residual norm when using optimal shifts, we provide upper bounds for the number of iterations needed to attain the target accuracy.

- We prove that for a d-dimensional problem, the condition number κ of the tensor Sylvester equation can (in principle) amplify the residual norm by a factor κ^{d-1}, when a nested solver is used. When the A_t are M-matrices, we show that the impact is reduced to $(\sqrt{\kappa})^{d-1}$ (Lemma 3.7).
- A thorough complexity analysis (Theorem 3.9 and 4.6), where the role of the HSS ranks, the target accuracy, and the condition number of the A_ts are fully revealed. In particular, we show that the condition numbers have a mild impact on the computational cost.

The paper is organized as follows. In Section 2, we provide a high-level description of the proposed scheme, for a d-dimensional tensor Sylvester equation. Section 3 and Section 4 are dedicated to the theoretical analysis of the algorithm for the matrix and tensor case, respectively. Finally, in the numerical experiments of Section 5 we compare the proposed algorithm with Algorithm 1 where the diagonalization is performed with a dense method or with the algorithm proposed in [25] for HSS matrices.

1.3 Notation

Throughout the paper, we denote matrices with capital letters (X, Y, ...), and tensors with calligraphic letters ($\mathcal{X}, \mathcal{Y}$, ...). We use the same letter with different fonts to denote matricizations of tensors (e.g., X is a matricization of $\mathcal{X}$). The Greek letters α_t, β_t indicate the extrema of the interval $[\alpha_t, \beta_t]$ enclosing the spectrum of A_t, and κ is used to denote the upper bound on the condition number of the Sylvester operator $\kappa = (\beta_1 + \ldots + \beta_d)/(\alpha_1 + \ldots + \alpha_d)$.

2 High-level description of the divide-and-conquer scheme

We consider HSS matrices A_t, so that each A_t can be decomposed as $A_t = A_t^{\text{diag}} + A_t^{\text{off}}$ where A_t^{diag} is block diagonal with square diagonal blocks, A_t^{off} is low-rank and the decomposition applies recursively to the blocks of A_t^{diag}. A particular case, where this assumption is satisfied, is when the coefficients A_t are all banded.

In the spirit of divide and conquer solvers for matrix equations [18,24], we remark that, given the additive decomposition $A_1 = A_1^{\text{diag}} + A_1^{\text{off}}$, the solution $\mathcal{X}$ of (2) can be written as $\mathcal{X}^{(1)} + \delta\mathcal{X}$ where

$$\mathcal{X}^{(1)} \times_1 A_1^{\text{diag}} + \mathcal{X}^{(1)} \times_2 A_2 + \cdots + \mathcal{X}^{(1)} \times_d A_d = \mathcal{B}, \tag{3}$$

$$\delta\mathcal{X} \times_1 A_1 + \delta\mathcal{X} \times_2 A_2 + \cdots + \delta\mathcal{X} \times_d A_d = -\mathcal{X}^{(1)} \times_1 A_1^{\text{off}}. \tag{4}$$

If $A_1^{\text{diag}} = \begin{bmatrix} A_{1,11}^{(1)} & \\ & A_{1,22}^{(1)} \end{bmatrix}$, then (3) decouples into two tensor equations of the form

$$\mathcal{X}_j^{(1)} \times_1 A_{1,jj}^{(1)} + \mathcal{X}_j^{(1)} \times_2 A_2 + \cdots + \mathcal{X}_j^{(1)} \times_d A_d = \mathcal{B}_j, \qquad j = 1, 2, \tag{5}$$

with $\mathcal{X}_1^{(1)}$ containing the entries of $\mathcal{X}^{(1)}$ with the first index restricted to the column indices of $A_{1,11}^{(1)}$, and $\mathcal{X}_2^{(1)}$ to those of $A_{1,22}^{(1)}$[1]. Equation (4) has the notable property that its right-hand

[1] In Matlab notation, if $A_{1,11}^{(1)}$ is of size $m \times m$ we have $\mathcal{X}_1^{(1)} = \mathcal{X}^{(1)}(1:m, :, \ldots, :)$ and $\mathcal{X}_2^{(1)} = \mathcal{X}^{(1)}(m+1:\text{end}, :, \ldots, :)$

side is a d-dimensional tensor multiplied in the first mode by a low-rank matrix. Merging the modes from 2 to d in equation (4) yields the matrix Sylvester equation

$$A_1 \delta X + \delta X \left(I \otimes \ldots I \otimes A_2 + \cdots + A_d \otimes I \otimes \cdots \otimes I \right) = -A_1^{\text{off}} X^{(1)}. \tag{6}$$

In particular, the right-hand side of (6) has rank bounded by $\text{rank}(A_1^{\text{off}})$ and the matrix coefficients of the equation are positive definite. This implies that δX has numerically low-rank and can be efficiently approximated with a low-rank Sylvester solver such as a rational Krylov subspace method [12, 29] or the *alternating direction implicit method* (ADI) [6].

We note that, applying the splitting simultaneously to all d modes yields an update equation for $\delta\mathcal{X}$ of the form

$$\delta\mathcal{X} \times_1 A_1 + \delta\mathcal{X} \times_2 A_2 + \cdots + \delta\mathcal{X} \times_d A_d = -\sum_{t=1}^{d} \mathcal{X}^{(1)} \times_t A_t^{\text{off}}, \tag{7}$$

and 2^d recursive calls:

$$\mathcal{X}^{(1)}_{j_1,\ldots,j_d} \times_1 A^{(1)}_{1,j_1 j_1} + \cdots + \mathcal{X}^{(1)}_{j_1,\ldots,j_d} \times_d A_{d,j_d j_d} = \mathcal{B}_{j_1,\ldots,j_d}, \qquad j_t \in \{1,2\}. \tag{8}$$

However, when $d > 2$, the right-hand side of Equation (7) is not necessarily low-rank for any matricization. On the other hand, by additive splitting of the right-hand side we can write $\delta\mathcal{X} := \delta\mathcal{X}_1 + \ldots + \delta\mathcal{X}_d$, where $\delta\mathcal{X}_t$ is the solution to an equation of the form (4).

In view of the above discussion, we propose the following recursive strategy for solving (1):

1. if all the n_is are sufficiently small then solve (1) by diagonalization,
2. otherwise split the equation along all modes as in (7) and (8),
3. compute $\mathcal{X}^{(1)}$ by solving the 2^d equations in (8) recursively,
4. approximate $\delta\mathcal{X}_t$ by applying a low-rank matrix Sylvester solver for $t = 1, \ldots, d$.
5. return $\mathcal{X} = \mathcal{X}^{(1)} + \delta\mathcal{X}_1 + \ldots + \delta\mathcal{X}_d$.

The procedure will be summarized in Algorithm 4 of Section 4, where we will consider the case of tensors in detail. To address point 4. we can use any of the available low-rank solvers for Sylvester equations [29]; in this work we consider the fADI and the rational Krylov subspace methods that are discussed in detail in the next sections. In Algorithm 4 we refer to the chosen method with LOW_RANK_SYLV. We remark that both these choices require to have a low-rank factorization of the mode j unfolding of $\mathcal{X}_0 \times_j A_j^{\text{off}}$ and to solve shifted linear systems with a Kronecker sum of $d-1$ matrices A_t. The latter task is again of the form (1) with $d-1$ modes and is performed recursively with Algorithm 4, when $d > 2$; this makes our algorithm a nested solver. At the base of the recursion, when (1) has only one mode, this is just a shifted linear system. We discuss this in detail in section 3.3.

2.1 Notation for Hierarchical matrices

The HSS matrices A_t $(t = 1, \ldots, d)$ can be partitioned as follows:

$$A_t = \begin{bmatrix} A^{(1)}_{t,11} & A^{(1)}_{t,12} \\ A^{(1)}_{t,21} & A^{(1)}_{t,22} \end{bmatrix} \in \mathbb{R}^{n_t \times n_t}, \tag{9}$$

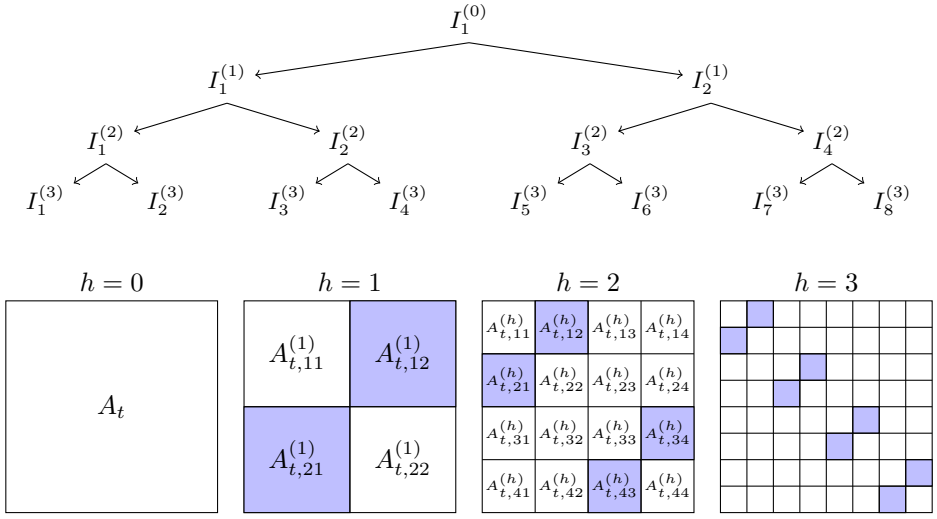

Figure 1: Example of the hierarchical low-rank structure obtained with the recursive partitioning in (9). The light blue blocks are the low-rank submatrices identified at each level.

where $A_{t,12}^{(1)}$ and $A_{t,21}^{(1)}$ have low rank, and $A_{t,ii}^{(1)}$ are HSS matrices. In particular, the diagonal blocks are square and can be recursively partitioned in the same way $\ell_t - 1$ times. The depth ℓ_t is chosen to ensure that the blocks at the lowest level of the recursion are smaller than a prescribed minimal size $n_{\min} \times n_{\min}$.

More formally, after one step of recursion we partition $I_1^{(0)} = \{1, \dots, n_t\} = I_1^{(1)} \sqcup I_2^{(1)}$ where $I_1^{(1)}$ and $I_2^{(1)}$ are two sets of contiguous indices; the matrices $A_{t,ij}$ in (9) have $I_i^{(1)}$ and $I_j^{(1)}$ as row and column indices, respectively, for $i, j = 1, 2$.

Similarly, after $h \leq \ell_t$ steps of recursion, one has the partitioning $\{1, \dots, n_t\} = I_1^{(h)} \sqcup \ldots \sqcup I_{2^h}^{(h)}$, and we denote by $A_{t,ij}^{(h)}$ with $1 \leq i, j \leq 2^h$ the submatrices of A_t with row indices $I_i^{(h)}$ and column indices $I_j^{(h)}$. Note that $A_{t,ii+1}^{(h)}$ and $A_{t,i+1i}^{(h)}$ indicate the low-rank off-diagonal blocks uncovered at level h (see Figure 1).

The quad-tree structure of submatrices of A_t, corresponding to the above described splitting of row and column indices, is called the *cluster tree* of A_t; see [24, Definition 1] for the rigorous definition. The integer ℓ_t is called the *depth of the cluster tree*.

Often, we will need to group together all the diagonal blocks at level h; we denote by $A_t^{(h)}$ such matrix, that is:

$$A_t = \underbrace{\begin{bmatrix} A_{t,11}^{(h)} & & & \\ & \ddots & & \\ & & \ddots & \\ & & & A_{t,2^h 2^h}^{(h)} \end{bmatrix}}_{A_t^{(h)}} + \begin{bmatrix} 0 & \star & \dots & \star \\ \star & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \star \\ \star & \dots & \star & 0 \end{bmatrix}. \tag{10}$$

Finally, the maximum rank of the off-diagonal blocks of an HSS matrix is called the *HSS rank*.

2.2 Representation and operations with HSS matrices

An $n \times n$ matrix in the form described in the previous section, with HSS rank k, can be effectively stored in the HSS format [33], using only $\mathcal{O}(nk)$ memory. Using this structured representation, matrix-vector multiplications and solution of linear systems can be performed with $\mathcal{O}(nk)$ and $\mathcal{O}(nk^2)$ flops, respectively.

Our numerical results leverage the implementation of this format and the related matrix operations available in `hm-toolbox` [23].

3 The divide-and-conquer approach for matrix Sylvester equations

We begin by discussing the case $d = 2$, that is the matrix Sylvester equation

$$A_1 X + X A_2 = B, \qquad B \in \mathbb{C}^{n_1 \times n_2}, \tag{11}$$

since there is a major difference with respect to $d > 2$ that makes the theoretical analysis much simpler. Indeed, the call to LOW_RANK_SYLV for solving (6) does not need to recursively call the divide-and-conquer scheme, which is needed when $d > 2$ for generating the right Krylov subspaces. Moreover, we assume that $\ell_1 = \ell_2 =: \ell$, that automatically implies that n_1 and n_2 are of the same order of magnitude; the algorithm can be easily adjusted for unbalanced dimensions, as we discuss in detail in Section 3.2.

We will denote by $B^{(h)} = [B_{ij}^{(h)}]$ the matrix B seen as a block matrix partitioned according to the cluster trees of A_1 and A_2 at level h for the rows and columns, respectively.

Splitting both modes at once yields the update equation

$$A_1 \delta X + \delta X A_2 = -A_1^{\text{off}} X^{(1)} - X^{(1)} A_2^{\text{off}}, \qquad X^{(1)} = \begin{bmatrix} X_{11}^{(1)} & X_{12}^{(1)} \\ X_{21}^{(1)} & X_{22}^{(1)} \end{bmatrix} \tag{12}$$

where $X_{ij}^{(1)}$ is the solution of $A_{1,ii}^{(1)} X_{ij}^{(1)} + X_{ij}^{(1)} A_{2,jj}^{(1)} = B_{ij}^{(1)}$ and $B_{ij}^{(1)}$ is the block in position (i, j) in $B^{(1)}$. The right-hand side of (12) has rank bounded by $\operatorname{rank}(A_1^{\text{off}}) + \operatorname{rank}(A_2^{\text{off}})$, therefore we can use LOW_RANK_SYLV to solve (12). The procedure is summarized in Algorithm 2.

Algorithm 2

1: **procedure** LYAP2D_D&C_BALANCED(A_1, A_2, B, ϵ)
2: **if** $\max_i n_i \leq n_{\min}$ **then**
3: **return** LYAPND_DIAG(A_1, A_2, B)
4: **else**
5: $X_{ij}^{(1)} \leftarrow$ LYAP2D_D&C($A_{1,ii}^{(1)}, A_{2,jj}^{(1)}, B_{ij}^{(1)}$), for $i, j = 1, 2$
6: $X^{(1)} \leftarrow \begin{bmatrix} X_{11}^{(1)} & X_{12}^{(1)} \\ X_{21}^{(1)} & X_{22}^{(1)} \end{bmatrix}$
7: Retrieve a low-rank factorization of $A_1^{\text{off}} X^{(1)} + X^{(1)} A_2^{\text{off}} = UV^T$
8: $\delta X \leftarrow$ LOW_RANK_SYLV(A_1, A_2, U, V, ϵ)
9: **return** $X^{(1)} + \delta X$
10: **end if**
11: **end procedure**

Remark 1. *Efficient solvers of Sylvester equations with low-rank right-hand sides need a factorization of the latter; see line 7 in Algorithm 2. Once the matrix $X^{(1)}$ is computed, the factors*

U and V are retrieved with explicit formula involving the factorizations of the matrices A_1^{off} and A_2^{off} [18, Section 3.1]. The low-rank representation can be cheaply compressed via a QR-SVD based procedure; our implementation always apply this compression step.

3.1 Analysis of the equations generated in the recursion

In this section we introduce the notation for all the equations solved in the recursion, as this will be useful in the error and complexity analysis.

We denote with capital letters (e.g., $X, \delta X$) the exact solutions of such equations, and with an additional tilde (i.e., $\widetilde{X}$ and $\delta\widetilde{X}$) their inexact counterparts obtained in finite precision computations.

3.1.1 Exact arithmetic

We begin by considering the computation, with Algorithm 2, of the solution of a matrix Sylvester equation, assuming that all the equations generated during the recursion are solved exactly. In this scenario, the solution X admits the additive splitting

$$X = X^{(\ell)} + \delta X^{(\ell-1)} + \ldots + \delta X^{(0)},$$

where $X^{(\ell)}$ or $\delta X^{(h)}$ contains all the solutions determined at depth ℓ and h in the recursion, respectively. More precisely, $X^{(\ell)}$ takes the form

$$X^{(\ell)} := \begin{bmatrix} X_{1,1}^{(\ell)} & \cdots & X_{1,2^\ell}^{(\ell)} \\ \vdots & & \vdots \\ \mathcal{X}_{2^\ell,1}^{(\ell)} & \cdots & \mathcal{X}_{2^\ell,2^\ell}^{(\ell)}. \end{bmatrix}, \tag{13}$$

where $X_{i,j}^{(\ell)}$ solves the Sylvester equation $A_{1,ii}^{(\ell)} X_{i,j}^{(\ell)} + X_{i,j}^{(\ell)} A_{2,jj}^{(\ell)} = B_{i,j}$; for $h < \ell$, we denote $X^{(h)} := X^{(\ell)} + \delta X^{(\ell-1)} + \ldots + \delta X^{(h)}$, that solves $A_1^{(h)} X^{(h)} + X^{(h)} A_2^{(h)} = B^{(h)}$.

The matrix $\delta X^{(h)} = [\delta X_{i,j}^{(h)}]$ containing the solutions of the update equations at level $h < \ell$ is block-partitioned analogously to $B^{(h)}$ and $X^{(h)}$. The diagonal blocks $A_{1,ii}^{(h)}$ can be in turn split into their diagonal and off-diagonal parts as follows:

$$A_{1,ii}^{(h)} = \begin{bmatrix} A_{1,2i-1,2i-1}^{(h+1)} & \\ & A_{1,2i,2i}^{(h+1)} \end{bmatrix} + \begin{bmatrix} 0 & A_{1,2i-1,2i}^{(h+1)} \\ A_{1,2i,2i-1}^{(h+1)} & 0 \end{bmatrix},$$

and the same holds for $A_{2,jj}^{(h)}$. Then $\delta X_{i,j}^{(h)}$ solves $A_{1,ii}^{(h)} \delta X_{i,j}^{(h)} + \delta X_{i,j}^{(h)} A_{2,jj}^{(h)} = \Xi_{ij}^{(h)}$ where

$$\begin{aligned} \Xi_{ij}^{(h)} := &- \begin{bmatrix} 0 & A_{1,2i-1,2i}^{(h+1)} \\ A_{1,2i,2i-1}^{(h+1)} & 0 \end{bmatrix} \begin{bmatrix} X_{2i-1,2i-1}^{(h+1)} & X_{2i-1,2i}^{(h+1)} \\ X_{2i,2i-1}^{(h+1)} & X_{2i,2i}^{(h+1)} \end{bmatrix} \\ &- \begin{bmatrix} X_{2i-1,2i-1}^{(h+1)} & X_{2i-1,2i}^{(h+1)} \\ X_{2i,2i-1}^{(h+1)} & X_{2i,2i}^{(h+1)} \end{bmatrix} \begin{bmatrix} 0 & A_{2,2j-1,2j}^{(h+1)} \\ A_{2,2j,2j-1}^{(h+1)} & 0 \end{bmatrix}. \end{aligned} \tag{14}$$

Since $X_{ij}^{(h)}$ solves the Sylvester equation

$$A_{1,ii}^{(h)} X_{ij}^{(h)} + X_{ij}^{(h)} A_{2,jj}^{(h)} = B_{ij}^{(h)},$$

then, by rewriting the above equation as a linear system, we can bound

$$\|X_{ij}^{(h)}\|_F \leq \frac{\|B_{ij}^{(h)}\|_F}{\alpha_1+\alpha_2}.$$

Applying this relation in Equation (14) we get the following bound for the norm of the right-hand side $\Xi_{ij}^{(h)}$:

$$\begin{aligned}\|\Xi_{ij}^{(h)}\|_F &\leq (\beta_1+\beta_2)\left\|\begin{bmatrix} X_{2i-1,2i-1}^{(h+1)} & X_{2i-1,2i}^{(h+1)} \\ X_{2i,2i-1}^{(h+1)} & X_{2i,2i}^{(h+1)}\end{bmatrix}\right\|_F \\ &\leq \frac{\beta_1+\beta_2}{\alpha_1+\alpha_2}\sqrt{\|B_{2i-1,2i-1}^{(h+1)}\|_F^2+\|B_{2i-1,2i}^{(h+1)}\|_F^2+\|B_{2i,2i-1}^{(h+1)}\|_F^2+\|B_{2i,2i}^{(h+1)}\|_F^2} \\ &= \frac{\beta_1+\beta_2}{\alpha_1+\alpha_2}\|B_{ij}^{(h)}\|_F.\end{aligned}$$

We define the block matrix $\Xi^{(h)}=[\Xi_{ij}^{(h)}]$; collecting all the previous relation as (i,j) varies, we obtain $A_1^{(h)}\delta X^{(h)}+\delta X^{(h)}A_2^{(h)}=\Xi^{(h)}$ and $\|\Xi^{(h)}\|_F \leq \frac{\beta_1+\beta_2}{\alpha_1+\alpha_2}\|B\|_F$.

3.1.2 Inexact arithmetic

In a realistic scenario, the Sylvester equations for determining $X^{(\ell)}$ and $\delta X^{(h)}$ are solved inexactly. We make the assumption that all Sylvester equations of the form $A_1X+XA_2=B$ are solved with a residual satisfying

$$A_1\widetilde{X}+\widetilde{X}A_2=B+R, \qquad \|R\|_F\leq\epsilon\|B\|_F. \tag{15}$$

Then, the approximate solutions computed throughout the recursion verify:

$$\begin{aligned}A_{1,ii}^{(\ell)}\widetilde{X}_{i,j}^{(\ell)}+\widetilde{X}_{i,j}^{(\ell)}A_{2,jj}^{(\ell)} &= B_{i,j}^{(\ell)}+R_{ij}^{(\ell)} \\ A_{1,ii}^{(h)}\delta\widetilde{X}_{i,j}^{(h)}+\delta\widetilde{X}_{i,j}^{(h)}A_{2,jj}^{(h)} &= \widetilde{\Xi}_{ij}^{(h)}+R_{ij}^{(h)},\end{aligned}$$

where $\widetilde{\Xi}_{ij}^{(h)}$ is defined by replacing $X_{ij}^{(h+1)}$ with $\widetilde{X}_{ij}^{(h+1)}$ in (14). Thanks to our assumption on the inexact solver, we have that $\|R_{ij}^{(\ell)}\|_F\leq\epsilon\|B_{ij}^{(\ell)}\|_F$; bounding $\|R_{ij}^{(h)}\|_F$ for $h<\ell$, is slightly more challenging, since it depends on the accumulated inexactness. Let us consider the matrices $R^{(h)}=[R_{ij}^{(h)}]$ that correspond to the residuals of the Sylvester equations

$$A_1^{(\ell)}\widetilde{X}^{(\ell)}+\widetilde{X}^{(\ell)}A_2^{(\ell)}=B^{(\ell)}+R^{(\ell)}, \tag{16}$$

$$A_1^{(h)}\delta\widetilde{X}^{(h)}+\delta\widetilde{X}^{(h)}A_2^{(h)}=\widetilde{\Xi}^{(h)}+R^{(h)}. \tag{17}$$

A bound on $\|R^{(h)}\|_F$ can be derived by controlling the ones of $\widetilde{\Xi}^{(h)}$.

Lemma 3.1. *If the Sylvester equations generated in Algorithm 4 are solved with the accuracy prescribed in* (15) *then* $\widetilde{X}^{(h)}:=\widetilde{X}^{(\ell)}+\delta\widetilde{X}^{(\ell-1)}+\ldots+\delta\widetilde{X}^{(h)}$ *satisfies*

$$A_1^{(h)}\widetilde{X}^{(h)}+\widetilde{X}^{(h)}A_2^{(h)}=B+R^{(\ell)}+\ldots+R^{(h)}, \tag{18}$$

where $R^{(h)}$ *are the residuals of* (16) *and* (17)*. In addition, if* $\kappa\epsilon<1$ *where* $\kappa:=\frac{\beta_1+\beta_2}{\alpha_1+\alpha_2}$*, then*

$$\|R^{(h)}\|_F\leq\kappa\epsilon(1+\epsilon)(1+\kappa\epsilon)^{\ell-h-1}\|B\|_F.$$

Proof. We start with the proof of (18) by induction over h. For $h = \ell$, the claim follows by (16). If $h < \ell$, we decompose $\widetilde{X}^{(h)} = \widetilde{X}^{(h+1)} + \delta\widetilde{X}^{(h)}$ to obtain

$$\begin{aligned} A_1^{(h)}\widetilde{X}^{(h)} + \widetilde{X}^{(h)}A_2^{(h)} &= A_1^{(h)}\widetilde{X}^{(h+1)} + \widetilde{X}^{(h+1)}A_2^{(h)} + A_1^{(h)}\delta\widetilde{X}^{(h)} + \delta\widetilde{X}^{(h)}A_2^{(h)} \\ &= \underbrace{(A_1^{(h)} - A_1^{(h+1)})\widetilde{X}^{(h+1)} + \widetilde{X}^{(h+1)}(A_2^{(h)} - A_2^{(h+1)})}_{-\widetilde{\Xi}^{(h)}} \\ &\quad + A_1^{(h+1)}\widetilde{X}^{(h+1)} + \widetilde{X}^{(h+1)}A_2^{(h+1)} + \widetilde{\Xi}^{(h)} + R^{(h)}, \end{aligned}$$

and the claim follows by the induction step.

We now show the second claim, once again, by induction. For $h = \ell$, we obtain the result by collecting all the residuals together in a block matrix:

$$\|R_{ij}^{(\ell)}\|_F \le \epsilon\|B_{ij}^{(\ell)}\|_F \implies \|R^{(\ell)}\|_F \le \epsilon\|B^{(\ell)}\|_F.$$

Since $\kappa \ge 1$, we have $\epsilon \le \kappa\epsilon(1+\epsilon)(1+\kappa\epsilon)^{-1}$, so that the bound is satisfied. For $h < \ell$ we have

$$\begin{aligned} \|R^{(h)}\|_F &\le \epsilon\|\widetilde{\Xi}^{(h)}\|_F \le \epsilon(\beta_1+\beta_2)\|\widetilde{X}^{(h+1)}\|_F \\ &\le \epsilon(\beta_1+\beta_2)\|X^{(h+1)}\|_F + \epsilon(\beta_1+\beta_2)\|\widetilde{X}^{(h+1)} - X^{(h+1)}\|_F. \end{aligned}$$

By subtracting $A_1^{(h+1)}X^{(h+1)} + X^{(h+1)}A_2^{(h+1)} = B$ from (18) we obtain

$$A_1^{(h+1)}(\widetilde{X}^{(h+1)} - X^{(h+1)}) + (\widetilde{X}^{(h+1)} - X^{(h+1)})A_2^{(h+1)} = R^{(\ell)} + \ldots + R^{(h+1)}.$$

Bounding the norm of the solution of this Sylvester equations by $\frac{1}{\alpha_1+\alpha_2}$ times the norm of the right-hand side yields

$$\|R^{(h)}\|_F \le \kappa\epsilon\left(\|B\|_F + \sum_{j=h+1}^{\ell}\|R^{(j)}\|_F\right).$$

For $h < \ell$, by the induction step, we have

$$\begin{aligned} \|R^{(h)}\|_F &\le \kappa\epsilon\left(\|B\|_F + \|R^{(\ell)}\|_F + \sum_{j=h+1}^{\ell-1}\|R^{(j)}\|_F\right) \\ &\le \kappa\epsilon\left(1 + \epsilon + \kappa\epsilon(1+\epsilon)\sum_{j=h+1}^{\ell-1}(1+\kappa\epsilon)^{\ell-j-1}\right)\|B\|_F \\ &= \kappa\epsilon(1+\epsilon)\left(1 - \kappa\epsilon\frac{1-(1+\kappa\epsilon)^{\ell-h-1}}{\kappa\epsilon}\right)\|B\|_F \\ &= \kappa\epsilon(1+\epsilon)(1+\kappa\epsilon)^{\ell-h-1}\|B\|_F. \end{aligned}$$

□

We can leverage the previous result to bound the residual of the approximate solution $\widetilde{X}$ returned by Algorithm 2.

Lemma 3.2. *Under the assumptions of Lemma 3.1, with the additional constraint $\kappa\epsilon < \frac{2}{\ell}$ the approximate solution $\widetilde{X} := \widetilde{X}^{(0)}$ returned by Algorithm 4 satisfies*

$$\|A_1\widetilde{X} + \widetilde{X}A_2 - B\|_F \le (\ell+1)^2\kappa\epsilon\|B\|_F.$$

Proof. In view of Lemma 3.1 the residual associated with $\widetilde{X} = \widetilde{X}^{(0)}$ satisfies

$$\|A_1\widetilde{X}^{(0)} + \widetilde{X}^{(0)}A_2 - B\|_F \leq \|R^{(0)}\|_F + \|R^{(1)}\|_F + \cdots + \|R^{(\ell)}\|_F.$$

Hence, summing the upper bounds for $\|R^{(h)}\|_F$ given in Lemma 3.1 we obtain

$$\begin{aligned}\|A_1\widetilde{X}^{(0)} + \widetilde{X}^{(0)}A_2 - B\|_F &\leq \left(\frac{\kappa\epsilon(1+\epsilon)}{1+\kappa\epsilon}\sum_{h=0}^{\ell}(1+\kappa\epsilon)^{\ell-h}\right)\|B\|_F \\ &\leq \left[(1+\kappa\epsilon)^{\ell+1} - 1\right]\|B\|_F = \left[\sum_{h=1}^{\ell+1}\binom{\ell+1}{h}(\kappa\epsilon)^h\right]\|B\|_F.\end{aligned}$$

The assumption $\kappa\epsilon < \frac{2}{\ell}$ guarantees that the dominant term in the sum occurs for $h = 1$, and therefore we have

$$\|A_1\widetilde{X}^{(\ell)} + \widetilde{X}^{(\ell)}A_2 - B\|_F \leq (\ell+1)^2\kappa\epsilon\|B\|_F.$$

□

3.2 Unbalanced dimensions

In the general case, when $n_1 \gg n_2$ and we employ the same $n_{\min}$ for both cluster trees of A_1 and A_2, we end up with $\ell_1 > \ell_2$. We can artificially obtain $\ell_1 = \ell_2$ by adding $\ell_1 - \ell_2$ auxiliary levels on top of the cluster tree of A_2. In all these new levels, we consider the trivial partitioning $\{1, \ldots, n_2\} = \{1, \ldots, n_2\} \cup \emptyset$.

This choice implies that for the first $\ell_1 - \ell_2$ levels of the recursion only the first dimension is split, so that only 2 matrix equations are generated by these recursive calls. This allows us to extend all the results in Section 3.4 by setting $\ell = \max\{\ell_1, \ell_2\}$.

In the practical implementation, for $n_1 \geq n_2$, this approach is encoded in the following steps:

- As far as n_1 is significantly larger than n_2, e.g. $n_1 \geq 2n_2$, apply the splitting on the first mode only.
- When n_1 and n_2 are almost equal, apply Algorithm 2.

The pseudocode describing this approach is given in Algorithm 3.

3.3 Solving the update equation

The update equation (12) is of the form

$$A_1\delta X + \delta X A_2 = UV^* \tag{19}$$

where A_1, A_2 are positive definite matrices with spectra contained in $[\alpha_1, \beta_1]$ and $[\alpha_2, \beta_2]$, respectively, $U \in \mathbb{R}^{m\times k}$, $V \in \mathbb{R}^{n\times k}$ with $k \ll \min\{m, n\}$. Under these assumptions, the singular values $\sigma_j(\delta X)$ of the solution δX of (19) decay rapidly to zero [5, 27]. More specifically, it holds

$$\sigma_{1+jk}(\delta X) \leq \sigma_1(\delta X)Z_j([\alpha_1, \beta_2], [-\beta_2, -\alpha_2]), \qquad Z_j(E, F) := \min_{r(z)\in\mathcal{R}_{j,j}} \frac{\max_E |r(z)|}{\min_F |r(z)|}$$

where $\mathcal{R}_{j,j}$ is the set of rational functions of the form $r(z) = p(z)/q(z)$ having both numerator and denominator of degree at most j. The optimization problem associated with $Z_j(E, F)$ is known in the literature as *third Zolotarev problem* and explicit estimates for the decay rate of $Z_j(E, F)$, as j increases, are available when E and F are disjoint real intervals [5]. In particular, we will make use of the following result.

Algorithm 3

procedure LYAP2D_D&C(A_1, A_2, B, ϵ)
  if $\max_i n_i \leq n_{\min}$ **then**
    return LYAPND_DIAG(A_1, A_2, B)
  else if $n_1 \leq 2n_2$ and $n_2 \leq 2n_1$ **then**
    return LYAP2D_D&C_BALANCED(A_1, A_2, B, ϵ) ▷ Algorithm 2
  else
    if $n_1 > 2n_2$ **then**
      Partition B as $\left[\begin{smallmatrix} B_1 \\ B_2 \end{smallmatrix}\right]$, according to the partitioning in $A_1^{(1)}$
      $X_1 \leftarrow$ LYAP2D_D&C($A_{1,11}^{(1)}, A_2, B_1$), $X_2 \leftarrow$ LYAP2D_D&C($A_{1,22}^{(1)}, A_2, B_2$)
      $X^{(1)} \leftarrow \left[\begin{smallmatrix} X_1 \\ X_2 \end{smallmatrix}\right]$
      Retrieve a low-rank factorization of $A_1^{\text{off}} X^{(1)} = UV^T$
    else if $n_2 > 2n_1$ **then**
      Partition B as $[\, B_1 \; B_2 \,]$, according to the partitioning in $A_2^{(1)}$
      $X_1 \leftarrow$ LYAP2D_D&C($A_1, A_{2,11}^{(1)}, B_1$), $X_2 \leftarrow$ LYAP2D_D&C($A_1, A_{2,22}^{(1)}, B_2$)
      $X^{(1)} \leftarrow [\, X_1 \; X_2 \,]$
      Retrieve a low-rank factorization of $X^{(1)} A_2^{\text{off}} = UV^T$
    end if
    $\delta X \leftarrow$ LOW_RANK_SYLV(A_1, A_2, U, V, ϵ)
    return $X^{(1)} + \delta X$
  end if
end procedure

Lemma 3.3 ([5, Corollary 4.2]). *Let $E = [\alpha_1, \beta_1] \subset \mathbb{R}^+$, $F = [-\beta_2, -\alpha_2] \subset \mathbb{R}^-$ be non-empty real intervals, then*

$$Z_j(E, F) \leq 4 \exp\left(\frac{\pi^2}{2\log(16\gamma)}\right)^{-2j}, \qquad \gamma := \frac{(\alpha_1 + \beta_2)(\alpha_2 + \beta_1)}{(\alpha_1 + \alpha_2)(\beta_1 + \beta_2)}. \tag{20}$$

Lemma 3.3 guarantees the existence of accurate low-rank approximations of δX. In this setting, the extremal rational function for $Z_j(E, F)$ is explicitly known and close form expressions for its zeros and poles are available[2]. We will see in the next sections that this enables us to design approximation methods whose convergence rate matches the one in (20).

3.3.1 A further property of Zolotarev functions

We now make an observation that will be relevant for the error analysis in Section 4.

Consider the Zolotarev problem associated with the symmetric configuration $[\alpha, \beta] \cup [-\beta, -\alpha]$, and let us indicate with $p_1, \ldots, p_s$ and $q_1, \ldots, q_s$ the zeros and poles of the optimal Zolotarev rational function. The symmetry of the configuration yields $p_i = -q_i$, and in turn the bound $|z - p_i|/|z - q_i| \leq 1$ for all $z \in [\alpha, \beta]$.

The last inequality also holds for nonsymmetric spectra configurations $[\alpha_1, \beta_1] \cup [-\beta_2, -\alpha_2]$. Indeed, the minimax problem is invariant under Möbius transformations, and the property holds for the evaluations of rational functions on any transformed domains. Since the optimal rational Zolotarev function on $[\alpha_1, \beta_1] \cup [-\beta_2, -\alpha_2]$ can be obtained by remapping the configuration into a symmetric one, the inequality holds on $[\alpha_1, \beta_1]$.

[2]Given $E = [\alpha_1, \beta_1] \subset \mathbb{R}^+$, $F = [-\beta_2, -\alpha_2] \subset \mathbb{R}^-$ an expression in terms of elliptic functions for the zeros and poles of the extremal rational function is given in [5, Eq. (12)]. Our implementation is based on the latter.

3.3.2 Alternating direction implicit method

The ADI method iteratively approximates the solution of (19) with the following two-steps scheme, for given scalar parameters $p_j, q_j \in \mathbb{C}$:

$$(A_1 - q_{s+1}I)\delta X_{s+\frac{1}{2}} = UV^* - \delta X_s(A_2 + q_{s+1}I),$$
$$\delta X_{s+1}(A_2 + p_{s+1}I) = UV^* - (A_1 - p_{j+1}I)\delta X_{s+\frac{1}{2}}.$$

The initial guess is $\delta X_0 = 0$, and it is easy to see that $\operatorname{rank}(\delta X_s) \leq sk$. This property is exploited in a specialized version of ADI, which is called *factored ADI* (fADI) [6]; the latter computes a factorized form of the update $\Delta_s := \delta X_s - \delta X_{s-1} = (q_s - p_s)W_s Y_s^*$ with the following recursion:

$$\begin{cases} W_1 = (A_1 - q_1 I)^{-1}U \\ W_{j+1} = (A_1 - q_{j+1})^{-1}(A_1 - p_j)W_j \end{cases} \qquad \begin{cases} Y_1 = -(A_2 + p_1 I)^{-1}V \\ Y_{j+1} = (A_2 + p_{j+1})^{-1}(A_2 + q_j)Y_j \end{cases}. \tag{21}$$

The approximate solution δX_s after s steps of fADI takes the form

$$\delta X_s = \Delta_1 + \ldots + \Delta_s = \sum_{j=1}^{s}(q_j - p_j)W_j Y_j^*.$$

Observe that, the most expensive operations when executing s steps of fADI are the solution of s shifted linear systems with the matrix A_1 and the same amount with the matrix A_2. Moreover, the two sequences can be generated independently.

The choice of the parameters p_j, q_j is crucial to control the convergence of the method, as highlighted by the explicit expressions of the residual and the approximation error after s steps [6]:

$$\delta X - \delta X_s = r_s(A_1)\delta X r_s(-A_2)^{-1} \tag{22}$$
$$A_1\delta X_s + \delta X_s A_2 - UV^* = -r_s(A_1)UV^* r_s(-A_2)^{-1}, \tag{23}$$

where $r_s(z) = \prod_{j=1}^{s}\frac{z-p_j}{z-q_j}$ and the second identity is obtained by applying the operator $X \mapsto A_1X + XA_2$ to $\delta X_s - \delta X$. Taking norms yields the following upper bound for the approximation error:

$$\|\delta X_s - \delta X\|_F \leq \frac{\max_{z\in[\alpha_1,\beta_1]}|r(z)|}{\min_{z\in[-\beta_2,-\alpha_2]}|r(z)|}\|\delta X\|_F, \tag{24}$$

$$\|A_1\delta X_s - \delta X A_2 - UV^*\|_F \leq \frac{\max_{z\in[\alpha_1,\beta_1]}|r(z)|}{\min_{z\in[-\beta_2,-\alpha_2]}|r(z)|}\|UV^*\|_F, \tag{25}$$

Inequalities (24) and (25) guarantee that if the shift parameters p_j, q_j are chosen as the zeros and poles of the extremal rational function for $Z_s([\alpha_1,\beta_1],[-\beta_2,-\alpha_2])$, then the approximation error and the residual norm decay (at least) as prescribed by Lemma 3.3. This allows us to give an a priori bound to the number of steps required to achieve a target accuracy ϵ.

Lemma 3.4. *Let $\epsilon > 0$, δX be the solution of* (19) *and δX_s the solution returned by applying the fADI method to* (19) *with parameters chosen as the zeros and poles of the extremal rational function for $Z_s([\alpha_1,\beta_1],[-\beta_2,-\alpha_2])$. If*

$$s \geq \frac{1}{\pi^2}\log\left(\frac{4}{\epsilon}\right)\log\left(16\frac{(\alpha_1+\beta_2)(\alpha_2+\beta_1)}{(\alpha_1+\alpha_2)(\beta_1+\beta_2)}\right), \tag{26}$$

then

$$\|A_1\delta X_s + \delta X_s A_2 - UV^*\|_F \leq \epsilon\|UV^*\|_F.$$

Proof. Combining Lemma 3.3 and inequality (25) yields the claim. □

3.3.3 Rational Krylov (RK) method

Another approach for solving (19) is to look for a solution in a well-chosen tensorization of low-dimensional subspaces. Common choices are rational Krylov subspaces of the form $\mathcal{K}_{s,A_1} := \text{span}\{U, (A_1-p_1I)^{-1}U, \dots, (A_1-p_sI)^{-1}U\}$, $\mathcal{K}_{s,A_2} := \text{span}\{V, (A_2^T+q_1I)^{-1}V, \dots, (A_2^T+q_sI)^{-1}V\}$; more specifically, one consider an approximate solution $\delta X_s = Q_{s,A_1}\delta Y Q_{s,A_2}^*$ where Q_{s,A_1}, Q_{s,A_2} are orthonormal bases of $\mathcal{K}_{s,A_1}, \mathcal{K}_{s,A_2}$, respectively, and δY solves the projected equation

$$(Q_{s,A_1}^* A_1 Q_{s,A_1})\delta Y + \delta Y (Q_{s,A_2}^* A_2 Q_{s,A_2}) = Q_{s,A_1}^* U V^* Q_{s,A_2}. \tag{27}$$

Similarly to fADI, when the parameters q_j, p_j are chosen as the zeros and poles of the extremal rational function for $Z_s([\alpha_1,\beta_1],[-\beta_2,-\alpha_2])$ then the residual of the approximation can be related to (20) [4, Theorem 2.1]:

$$\|A_1\delta X_s + \delta X_s A_2 - UV^*\|_F \le 2\left(1+\frac{\beta_1+\beta_2}{\alpha_1+\alpha_2}\right) Z_s([\alpha_1,\beta_1],[-\beta_2,-\alpha_2])\|UV^*\|_F. \tag{28}$$

Based on (28) we can state the analogous of Lemma 3.4 for rational Krylov.

Lemma 3.5. *Let $\epsilon > 0$, δX be the solution of* (19) *and δX_s the solution returned by applying the rational Krylov method to* (19) *with shifts chosen as the zeros and poles of the extremal rational function for $Z_s([\alpha_1,\beta_1],[-\beta_2,-\alpha_2])$. If*

$$s \ge \frac{1}{\pi^2}\log\left(\frac{8(\alpha_1+\alpha_2+\beta_1+\beta_2)}{\epsilon(\alpha_1+\alpha_2)}\right)\log\left(16\frac{(\alpha_1+\beta_2)(\alpha_2+\beta_1)}{(\alpha_1+\alpha_2)(\beta_1+\beta_2)}\right), \tag{29}$$

then

$$\|A_1\delta X_s + \delta X_s A_2 - UV^*\|_F \le \epsilon\|UV^*\|_F.$$

3.4 Error analysis for Algorithm 3

In Section 3.3 we have discussed two possible implementations of LOW_RANK_SYLV in Algorithm 3 that return an approximate solution of the update equation, with the residual norm controlled by the choice of the parameter s. This allows us to use Lemma 3.2 to bound the error in Algorithm 3.

Theorem 3.6. *Let A_1, A_2 be symmetric positive definite HSS matrices with cluster trees of depth at most ℓ, and spectra contained in $[\alpha_1,\beta_1]$ and $[\alpha_2,\beta_2]$, respectively. Let $\widetilde{X}$ be the approximate solution of* (11) *returned by Algorithm 3 where* LOW_RANK_SYLV *is either fADI or RK and uses the s zeros and poles of the extremal rational function for $Z_s([\alpha_1,\beta_1],[-\beta_2,-\alpha_2])$ as input parameters. If $\epsilon > 0$ and s is such that $s \ge s_{\text{ADI}}$ (when* LOW_RANK_SYLV *is fADI) or $s \ge s_{\text{RK}}$ (when* LOW_RANK_SYLV *is RK) where*

$$s_{\text{ADI}} = \frac{1}{\pi^2}\log\left(4\frac{(\beta_1+\beta_2)}{\epsilon(\alpha_1+\alpha_2)}\right)\log\left(16\frac{(\alpha_1+\beta_2)(\alpha_2+\beta_1)}{(\alpha_1+\alpha_2)(\beta_1+\beta_2)}\right),$$
$$s_{\text{RK}} = \frac{1}{\pi^2}\log\left(8\frac{(\beta_1+\beta_2)(\alpha_1+\alpha_2+\beta_1+\beta_2)}{\epsilon(\alpha_1+\alpha_2)^2}\right)\log\left(16\frac{(\alpha_1+\beta_2)(\alpha_2+\beta_1)}{(\alpha_1+\alpha_2)(\beta_1+\beta_2)}\right),$$

and Algorithm 1 solves the Sylvester equations at the base of the recursion with a residual norm bounded by ϵ times the norm of the right-hand side, then the computed solution satisfies:

$$\|A_1\widetilde{X} + \widetilde{X}A_2 - B\|_F \le (\ell+1)^2\frac{\beta_1+\beta_2}{\alpha_1+\alpha_2}\epsilon\|B\|_F. \tag{30}$$

Proof. In view of Lemma 3.4 (for fADI) and Lemma 3.5 (for RK), the assumption on s guarantees that the norm of the residual of each update equation is bounded by ϵ times the norm of its right-hand side. In addition, the equations at the base of the recursion are assumed to be solved at least as accurately as the update equations, and the claim follows by applying Lemma 3.2. □

We remark that, usually, the cluster trees for A_1 and A_2 are chosen by splitting the indices in a balanced way; this implies that their depths verify $\ell_i \sim O(\log(n_i/n_{\min}))$.

Remark 2. *We note that s_{RK} in Theorem 3.6 is larger than s_{ADI}; under the assumption $\alpha_1 + \alpha_2 + \beta_1 + \beta_2 \approx \beta_1 + \beta_2$, we have that*

$$s_{\mathrm{RK}} - s_{\mathrm{ADI}} \approx \frac{2}{\pi^2} \log\left(2\frac{\beta_1+\beta_2}{\alpha_1+\alpha_2}\right) \log\left(16\frac{(\alpha_1+\beta_2)(\alpha_2+\beta_1)}{(\alpha_1+\alpha_2)(\beta_1+\beta_2)}\right).$$

In practice, it is often observed that the rational Krylov method requires fewer shifts than fADI to attain a certain accuracy, because its convergence is linked to a rational function which is automatically optimized over a larger space [4].

3.4.1 The case of M-matrices

The term κ in the bound of Theorem 3.6 arises from controlling the norms of $\Xi^{(h)}$ and $\widetilde{\Xi}^{(h)} - \Xi^{(h)}$. In the special case where A_j are positive definite M-matrices this can be reduced to $\sqrt{\kappa}$ as shown in the following result.

Lemma 3.7. *Let A_1, A_2 be symmetric positive definite M-matrices. Then, the right-hand sides $\Xi^{(h)}$ of the intermediate Sylvester equations solved in exact arithmetic in Algorithm 3 satisfy*

$$\|\Xi^{(h)}\|_F \le \sqrt{\kappa}\|B\|_F, \qquad \|\widetilde{\Xi}^{(h)} - \Xi^{(h)}\|_F \le \sqrt{\kappa}\|R^{(\ell)} + \ldots + R^{(h+1)}\|_F,$$

where $\kappa = (\beta_1+\beta_2)/(\alpha_1+\alpha_2)$.

Proof. We note that $\Xi^{(h)}_{ij}$ can be written as $\Xi^{(h)}_{ij} = \mathcal{N}\mathcal{M}^{-1}B^{(h)}_{ij}$, where $\mathcal{N}$ and $\mathcal{M}$ are the operators

$$\mathcal{M}Y := \begin{bmatrix} A^{(h+1)}_{1,2i-1,2i-1} & \\ & A^{(h+1)}_{1,2i,2i} \end{bmatrix} Y + Y \begin{bmatrix} A^{(h+1)}_{2,2i-1,2i-1} & \\ & A^{(h+1)}_{2,2i,2i} \end{bmatrix}$$

$$\mathcal{N}Y := -\begin{bmatrix} & A^{(h+1)}_{1,2i-1,2i} \\ A^{(h+1)}_{1,2i,2i-1} & \end{bmatrix} Y - Y \begin{bmatrix} & A^{(h+1)}_{2,2i-1,2i} \\ A^{(h+1)}_{2,2i,2i-1} & \end{bmatrix}$$

In addition, $\mathcal{M} - \mathcal{N} = I \otimes A^{(h)}_{1,ii} + A^{(h)}_{2,ii} \otimes I$ is a positive definite M-matrix, and so is $\mathcal{M}$. In particular, $\mathcal{M} - \mathcal{N}$ is a regular splitting, and therefore $\rho(\mathcal{M}^{-1}\mathcal{N}) < 1$. Hence,

$$\begin{aligned}\|\Xi^{(h)}_{ij}\|_F &\le \|\mathcal{N}\mathcal{M}^{-1}\|_2 \|\mathcal{B}_{ij}\|_F \\ &\le \|\mathcal{M}^{\frac{1}{2}}\|_2 \|\mathcal{M}^{-\frac{1}{2}}\mathcal{N}\mathcal{M}^{-\frac{1}{2}}\|_2 \|\mathcal{M}^{-\frac{1}{2}}\|_2 \|\mathcal{B}_{ij}\|_F \le \sqrt{\kappa}\|\mathcal{B}_{ij}\|_F,\end{aligned}$$

where we have used that the matrix $\mathcal{M}^{-\frac{1}{2}}\mathcal{N}\mathcal{M}^{-\frac{1}{2}}$ is symmetric and similar to $\mathcal{N}M^{-1}$, and therefore has both spectral radius and spectral norm bounded by 1.

For the second relation we have

$$\widetilde{\Xi}^{(h)} - \Xi^{(h)} = -(A^{(h)}_1 - A^{(h+1)}_1)(\widetilde{X}^{(h+1)} - X^{(h+1)}) - (\widetilde{X}^{(h+1)} - X^{(h+1)})(A^{(h)}_2 - A^{(h+1)}_2),$$

and $\widetilde{X}^{(h+1)} - X^{(h+1)}$ solves the Sylvester equation (see Lemma 3.1)

$$A^{(h+1)}_1(\widetilde{X}^{(h+1)} - X^{(h+1)}) + (\widetilde{X}^{(h+1)} - X^{(h+1)})A^{(h+1)}_2 = R^{(\ell)} + \ldots + R^{(h+1)}.$$

Following the same argument used for the first point we obtain the claim. □

Corollary 3.8. *Under the same hypotheses and settings of Theorem 3.6, with the additional assumption of A_1, A_2 being M-matrices, Algorithm 1 computes an approximate solution $\widetilde{X}$ that satisfies*

$$\|A_1\widetilde{X} + \widetilde{X}A_2 - B\|_F \leq (\ell+1)^2\sqrt{\frac{\beta_1+\beta_2}{\alpha_1+\alpha_2}}\epsilon\|B\|_F. \tag{31}$$

3.4.2 Validation of the bounds

We now verify the dependency of the final residual norm on the condition number of the problem to inspect the tightness of the behavior predicted by Theorem 3.6 and Corollary 3.8. We consider two numerical tests concerning the solution of Lyapunov equations of the form $A_{(i)}X + XA_{(i)} = C$, where the $n \times n$ matrices $A_{(i)}$ have size $n = 256$ and increasing condition numbers with respect to i; the minimal block size is set to $n_{\min} = 32$.

In the first example we generate $A_{(i)} = QD_{(i)}Q^*$ where

- $D_{(i)} = D^{p_i}$ is the p_ith power of the diagonal matrix D containing the eigenvalues of the 1D discrete Laplacian, i.e. $2 + 2\cos(\pi j/(n+1))$, $j = 1, \ldots, n$. The p_i are chosen with a uniform sampling of $[1, 2.15]$, so that the condition numbers range approximately between 10^4 and 10^9.
- Q is the Q factor of the QR factorization of a randomly generated matrix with lower bandwidth equal to 8, obtained with the MATLAB command `[Q, ~] = qr(triu(randn(n), -8))`. This choice ensures that $A_{(i)}$ is SPD and HSS with HSS rank bounded by 8 [32].
- $C = QSQ^*$, where Q is the matrix defined in the previous point, and S is diagonal with entries $S_{ii} = \left(\frac{i-1}{n-1}\right)^{10}$. The latter choice aims at giving more importance to the component of the right-hand side aligned with the smallest eigenvectors of the Sylvester operator. We note that this is helpful to trigger the worst case behavior of the residual norm.

For each value of i we have performed 100 runs of Algorithm 3, using fADI as LOW_RANK_SYLV with threshold $\epsilon = 10^{-6}$. The measured residual norms $\|A_{(i)}\widetilde{X} + \widetilde{X}A_{(i)} - C\|_F/\|C\|_F$ are reported in the left part of Figure 2. We remark that the growth of the residual norm appears to be proportional to $\sqrt{\kappa}$ suggesting that the bound from Theorem 3.6 is pessimistic.

In the second test, the matrices $A_{(i)}$ are chosen as shifted 1D discrete Laplacian, where the shift is selected to match the same range of condition number of the first experiment; note that, the matrices $A_{(i)}$ are SPD M-matrices, with HSS rank 1 (they are tridiagonal). The right-hand side C is chosen as before, by replacing Q with the eigenvector matrix of the 1D discrete Laplacian. Corollary 3.8 would predict an $\mathcal{O}(\sqrt{\kappa})$ growth for the residual norms; however, as shown in the right part of Figure 2, the residual norms seems not influenced by the condition number of the problem.

Remark 3. *The examples reported in this section have been chosen to display the worst case behavior of the residual norms; in the typical case, for instance by choosing `C = randn(n)`, the influence of the condition number on the residual norms is hardly visible also in the case of non M-matrix coefficients.*

3.5 Complexity of Algorithm 3

In order to simplify the complexity analysis of Algorithm 3 we assume that $n_1 = 2^{\ell_1}n_{\min}$ and $n_2 = 2^{\ell_2}n_{\min}$ and that A_1, A_2 are HSS matrices of HSS rank k, with a partitioning of depth ℓ_1, ℓ_2 obtained by halving the dimension at every level.

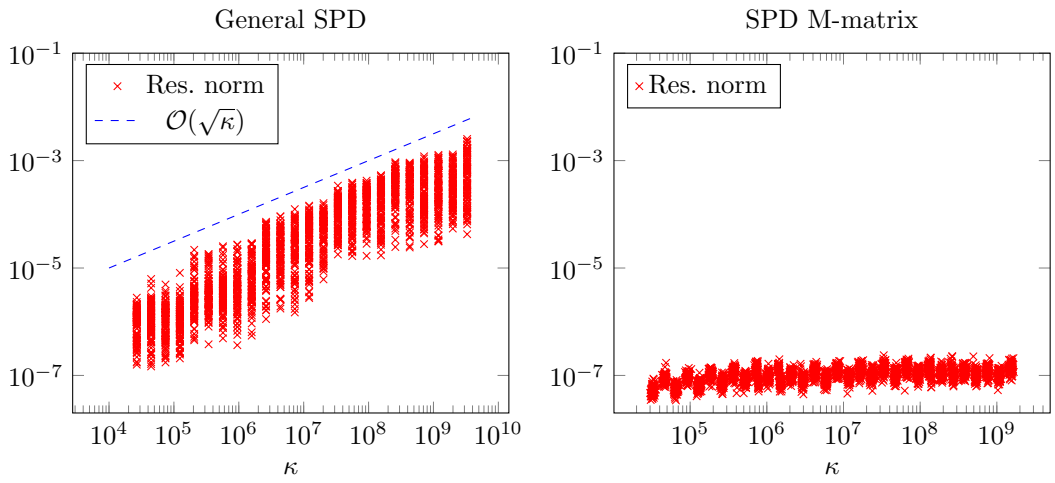

Figure 2: Residual norm behavior for Algorithm 3, with respect to the conditioning of problem. On the left, the coefficients of the matrix equation are generic HSS SPD matrices; On the right, the coefficients have the additional property of being M-matrices.

We start by considering the simpler case $n := n_1 = n_2$ and therefore $\ell = \ell_1 = \ell_2$. We remark that given $s \in \mathbb{N}$, executing s steps of fADI or RK for solving (6) with size $n \times n$, requires

$$\mathcal{C}_{ADI}(n, s, k) = \mathcal{O}(k^2 sn), \qquad \mathcal{C}_{RK}(n, s, k) = \mathcal{O}(k^2 s^2 n), \tag{32}$$

flops, respectively. Indeed, each iteration of fADI involves the solution of two block linear systems with $2k$ columns each and an HSS matrix coefficient; this is performed by computing two ULV factorizations ($\mathcal{O}(k^2 n)$, see [33]) and solving linear systems with triangular and unitary HSS matrices ($\mathcal{O}(kn)$ for each of the $2k$ columns). The cost analysis of rational Krylov is analogous, but it is dominated by reorthogonalizing the basis at each iteration; this requires $\mathcal{O}(k^2 jn)$ flops at iteration j, for $j = 1, \dots, s$.

Combining these ingredients yields the following.

Theorem 3.9. *Let $A_1, A_2 \in \mathbb{R}^{n \times n}$, $n = 2^\ell n_{\min}$, and assume that A_1, A_2 are HSS matrices of HSS rank k, with a partitioning of depth ℓ obtained by halving the dimension at every level. Then, Algorithm 2 requires:*

(i) $\mathcal{O}((k \log(n) + n_{\min} + sk^2)n^2)$ flops if LOW_RANK_SYLV *implements s steps of the fADI method,*

(ii) $\mathcal{O}((k \log(n) + n_{\min} + s^2 k^2)n^2)$ flops if LOW_RANK_SYLV *implements s steps of the rational Krylov method.*

Proof. We only prove (i) since (ii) is obtained with an analogous argument. Let us analyze the cost of Algorithm 4 at level j of the recursion. For $j = \ell$ it solves $2^\ell \cdot 2^\ell$ Sylvester equations of size $n_{\min} \times n_{\min}$ by means of Algorithm 1; this requires $\mathcal{O}(4^\ell \cdot n_{\min}^3) = \mathcal{O}(n_{\min} n^2)$ flops. At level $j < \ell$, 4^j Sylvester equations of size $\frac{n}{2^j} \times \frac{n}{2^j}$ and right-hand side of rank (at most) $2k$ are solved via fADI. According to (32), the overall cost of these is $\mathcal{O}(2^j k^2 sn)$. Finally, we need to evaluate 4^j multiplications at line 11 which yields a cost $\mathcal{O}(4^j k(n/2^j)^2) = \mathcal{O}(kn^2)$. Summing over all levels $j = 0, \dots, \ell - 1$ we get $\mathcal{O}(\sum_{j=0}^{\ell-1}(kn^2 + 2^j k^2 sn)) = \mathcal{O}(\ell kn^2 + 2^\ell k^2 sn)$. Noting that $\ell = \mathcal{O}(\log(n))$ provides the claim. □

We now consider the cost of Algorithm 3 in the more general case $n_1 \neq n_2$. Without loss of generality, we assume $n_1 = 2^{\ell_1} n_{\min} > 2^{\ell_2} n_{\min} = n_2$; Algorithm 3 begins with $\ell_1 - \ell_2$ splittings on the first mode. This generates $2^{\ell_1-\ell_2}$ subproblems of size $n_2 \times n_2$, $2^{\ell_1-\ell_2} - 1$ calls to LOW_RANK_SYLV and $2^{\ell_1-\ell_2} - 1$ multiplications with a block vector at line 12. In particular, we have 2^j calls to LOW_RANK_SYLV for problems of size $n_1/2^j \times n_2$, $j = 0, \dots, \ell_1 - \ell_2 - 1$; this yields an overall cost of $\mathcal{O}((\ell_1 - \ell_2)sk^2n_1)$ and $\mathcal{O}((\ell_1 - \ell_2)s^2k^2n_1)$ when fADI and rational Krylov are employed, respectively. Analogously, the multiplications at line 11 of this initial phase require $\mathcal{O}((\ell_1 - \ell_2)kn_1n_2)$ flops. Summing these costs to the estimate provided by Theorem 3.9, multiplied by $2^{\ell_1-\ell_2}$, yields the following corollary.

Corollary 3.10. *Under the assumptions of Theorem 3.9, apart from $n_1 = 2^{\ell_1} n_{\min} \geq n_2 = 2^{\ell_2} n_{\min}$ with $n_2 \geq \log(n_1/n_2)$, we have that Algorithm 3 costs:*

(i) $\mathcal{O}((k\log(n_1)+n_{\min}+sk^2)n_1n_2)$ *if* LOW_RANK_SYLV *implements s steps of the fADI method,*

(ii) $\mathcal{O}((k\log(n_1) + n_{\min} + s^2k^2)n_1n_2)$ *if* LOW_RANK_SYLV *implements s steps of the rational Krylov method.*

4 Tensor Sylvester equations

We now proceed to describe the procedure sketched in the introduction for $d > 2$. Initially, we assume that all dimensions are equal, so that the splitting is applied to all modes at every step of the recursion.

We identify two major differences with respect to the case $d = 2$, both concerning the update equation (7):

$$\delta\mathcal{X} \times_1 A_1 + \delta\mathcal{X} \times_2 A_2 + \cdots + \delta\mathcal{X} \times_d A_d = -\sum_{t=1}^{d} \mathcal{X}^{(1)} \times_t A_t^{\text{off}}.$$

First, the latter equation cannot be solved directly with a single call to LOW_RANK_SYLV, since the right-hand side does not have a low-rank matricization. However, the t-mode matricization of the term $\mathcal{X}^{(1)} \times_t A_t^{\text{off}}$ has rank bounded by k for $t = 1, \dots, d$. Therefore, the update equation can be addressed by d separate calls to the low-rank solver, by splitting the right-hand side, and matricizing the tth term in a way that separates the mode t from the rest, as follows:

$$A_t\delta X_t + \delta X_t \sum_{j\neq t} I \otimes \cdots \otimes I \otimes A_j \otimes I \otimes \cdots \otimes I = -A_t^{\text{off}} X^{(1)}, \tag{33}$$

where, in this case, $X^{(1)}$ denotes the t-mode matricization of $\mathcal{X}^{(1)}$.

Second, the solution of (33) by means of LOW_RANK_SYLV requires solving shifted linear systems with a Kronecker sum of $d - 1$ matrices, that is performed by recursively calling the divide-and-conquer scheme. This generates an additional layer of inexactness, that we have to take into account in the error analysis. This makes the analysis non-trivial, and requires results on the error propagation in LOW_RANK_SYLV; in the next section we address this point in detail for fADI, that will be the method of choice in our implementation.

In the case where the dimensions n_i are unbalanced, we follow a similar approach to the case $d = 2$. More precisely, at each level we only split on the r dominant modes which satisfy $2n_i \geq \max_j n_j$, and which are larger than $n_{\min}$. This generates 2^r recursive calls and r update equations.

We summarize the procedure in Algorithm 4.

Algorithm 4

1: **procedure** LYAPND_D&C($A_1, A_2, \ldots, A_d, \mathcal{B}, \epsilon$)
2: **if** $\max_i n_i \leq n_{\min}$ **then**
3: **return** LYAPND_DIAG($A_1, A_2, \ldots, A_d, \mathcal{B}$)
4: **else**
5: Permute the modes to have $2n_i \geq \max_j\{n_j\}$ and $n_i \geq n_{\min}$, for $i \leq r$.
6: **for** $j_1, \ldots, j_r = 1, 2$ **do**
7: $\mathcal{Y}^{(j_1,\ldots,j_r)} \leftarrow$LYAPND_D&C($A_1^{(j_1 j_1)}, \ldots, A_s^{(j_r j_r)}, A_{r+1}, \ldots, A_d, \mathcal{B}^{(j_1,\ldots,j_r)}, \epsilon$)
8: Copy $\mathcal{Y}^{(j_1,\ldots,j_r)}$ in the corresponding entries of $\mathcal{X}^{(1)}$.
9: **end for**
10: **for** $j = 1, \ldots, r$ **do**
11: Retrieve a rank k factorization $A_j^{\text{off}} = U\widetilde{V}^T$
12: $V \leftarrow$ RESHAPE$(-\mathcal{X}^{(1)} \times_j \widetilde{V}, \prod_{i\neq j} n_i, k)$
13: $\delta X_j \leftarrow$ LOW_RANK_SYLV$(A_j, \sum_{i\neq j} I \otimes \cdots \otimes I \otimes A_i \otimes I \otimes \cdots \otimes I, U, V, \epsilon)$
14: $\delta\mathcal{X}_j \leftarrow$ tensorize δX_j, inverting the j-mode matricization.
15: **end for**
16: **return** $\mathcal{X}^{(1)} + \delta\mathcal{X}_1 + \ldots + \delta\mathcal{X}_r$.
17: **end if**
18: **end procedure**

4.1 Error analysis

The use of an inexact solver for tensor Sylvester equations with $d-1$ modes in LOW_RANK_SYLV has an impact on the achievable accuracy of the method. Under the assumption that all the update equations at line 13 are solved with a relative accuracy ϵ we can easily generalize the analysis performed in Section 3.1.1.

More specifically, we consider the additive decomposition of $\widetilde{\mathcal{X}} = \widetilde{\mathcal{X}}^{(0)} = \widetilde{\mathcal{X}}^{(\ell)} + \delta\widetilde{\mathcal{X}}^{(\ell-1)} + \ldots + \delta\widetilde{\mathcal{X}}^{(0)}$, where the equations solved by the algorithm are the following:

$$\sum_{t=1}^{d} \widetilde{\mathcal{X}}^{(\ell)} \times_t A_t^{(\ell)} = \mathcal{B} + \mathcal{R}^{(\ell)} \tag{34}$$

$$\sum_{t=1}^{d} \delta\widetilde{\mathcal{X}}^{(h,j)} \times_t A_t^{(h)} = \widetilde{\Xi}^{(h,j)} + \mathcal{R}^{(h,j)}, \tag{35}$$

with $A_t^{(h)}$ are the block-diagonal matrices defined in (10), and $\delta\widetilde{\mathcal{X}}^{(h)} = \delta\widetilde{\mathcal{X}}^{(h,1)} + \ldots + \delta\widetilde{\mathcal{X}}^{(h,d)}$, and $\widetilde{\Xi}^{(h,j)}$ is defined as follows:

$$\widetilde{\Xi}^{(h,j)} := -\widetilde{\mathcal{X}}^{(h-1)} \times_j (A_j^{(h)} - A_j^{(h+1)}).$$

We remark that the matrix $A_j^{(h)} - A_j^{(h+1)}$ contains the off-diagonal blocks that are present in $A_j^{(h)}$ but not in $A_j^{(h+1)}$. When the dimensions n_i are unbalanced we artificially introduce some levels in the HSS partitioning (see Section 3.2), some of these may be the zero matrix. Then, we state the higher-dimensional version of Lemma 3.2.

Lemma 4.1. *If the tensor Sylvester equations* (34) *and* (35) *are solved with the relative accuracies* $\|\mathcal{R}^{(\ell)}\|_F \leq \epsilon\|\mathcal{B}\|_F$ *and* $\|\mathcal{R}^{(h,j)}\|_F \leq \epsilon\|\widetilde{\Xi}^{(h,j)}\|_F$ *and* $\kappa\epsilon < 1$, *with* $\kappa := \frac{\beta_1+\ldots+\beta_d}{\alpha_1+\ldots+\alpha_d}$, *then the approximate solution* $\widetilde{\mathcal{X}}$ *returned by Algorithm 4 satisfies*

$$\Big\| \sum_{i=1}^{d} \widetilde{\mathcal{X}} \times_i A_i - \mathcal{B} \Big\|_F \leq (\ell+1)^2 \kappa\epsilon \|\mathcal{B}\|_F.$$

Proof. Let us consider $\widetilde{\mathcal{X}}^{(h)} = \widetilde{\mathcal{X}}^{(h)} + \delta\widetilde{\mathcal{X}}^{(\ell-1)} + \ldots + \delta\widetilde{\mathcal{X}}^{(h)}$. Following the same argument in the proof of Lemma 3.1, we obtain

$$\sum_{i=1}^{d} \widetilde{\mathcal{X}}^{(h)} \times_i A_i = \mathcal{B} + \mathcal{R}^{(\ell)} + \ldots + \mathcal{R}^{(h)},$$

where $\mathcal{R}^{(h)} := \sum_{j=1}^{d} \mathcal{R}^{(h,j)}$. Hence, we bound

$$\begin{aligned}
\|\mathcal{R}^{(h)}\|_F &\leq \sum_{j=1}^{d} \|\mathcal{R}^{(h,j)}\|_F \leq \epsilon \sum_{j=1}^{d} \|\widetilde{\Xi}^{(h,j)}\|_F \\
&\leq \epsilon \sum_{j=1}^{d} \beta_j (\|\mathcal{X}^{(h+1)}\|_F + \|\mathcal{X}^{(h+1)} - \widetilde{\mathcal{X}}^{(h+1)}\|_F) \\
&\leq \epsilon\kappa (\|\mathcal{B}\|_F + \|\mathcal{R}^{(\ell)}\|_F + \ldots + \|\mathcal{R}^{(h+1)}\|_F).
\end{aligned}$$

By induction one can show that

$$\|\mathcal{R}^{(h)}\|_F \leq \kappa\epsilon(1+\epsilon)(1+\kappa\epsilon)^{\ell-h-1}\|\mathcal{B}\|_F,$$

and the claim follows applying the same reasoning as in the proof of Lemma 3.2. □

Lemma 4.1 ensures that if the update equations are solved with uniform accuracy the growth in the residual is controlled by a moderate factor, as in the matrix case.

We now investigate what can be ensured if we perform a constant number of fADI steps throughout all levels of recursions (including the use of the nested Sylvester solvers). This requires the development of technical tools for the analysis of factored ADI with inexact solves.

4.1.1 Factored ADI with inexact solves

Algorithm 4 solves update equations that can be matricized as $A_1\delta X + \delta X A_2 = UV^*$, where the factors U and V are retrieved analogously to the matrix case, see Remark 1, and A_1 is the Kronecker sum of $d-1$ matrices. In particular, linear systems with A_1 are solved inexactly by a nested call to Algorithm 4. In this section we investigate how the inexactness affects the residual of the solution computed with s steps of fADI.

We begin by recalling some properties of fADI. Assume to have carried out $j+1$ steps of fADI applied to the equation $A_1\delta X + \delta X A_2 = UV^*$. Then, the factors W_{j+1}, Y_{j+1} can be obtained by running a single fADI iteration for the equation

$$A_1\delta X + \delta X A_2 = -r_j(A_1)UV^* r_j(-A_2)^{-1}, \quad r_j(z) := \prod_{h=1}^{j} \frac{z-p_h}{z-q_h},$$

using parameters p_{j+1}, q_{j+1}.

We now consider the sequence $\widetilde{W}_j$ obtained by solving the linear systems with A_1 inexactly:

$$(A_1 - q_{j+1}I)\widetilde{W}_{j+1} = (A_1 - p_j I)\widetilde{W}_j + \eta_{j+1}, \quad (A_1 - q_1 I)\widetilde{W}_1 = U + \eta_1, \tag{36}$$

where $\|\eta_{j+1}\|_2 \leq \epsilon\|U\|$.

The following lemma quantifies the distance between $\widetilde{W}_j$ and W_j. Note that we make the assumption $|z-p_j| \leq |z-q_j|$ over $[\alpha_1, \beta_1]$, that is satisfied by the parameters of the Zolotarev functions, as discussed in Section 3.3.1.

Lemma 4.2. *Let A_1 be positive definite, and p_j, q_j satisfying $|z-p_j| \leq |z-q_j|$ for any $z \in [\alpha_1, \beta_1]$. Let $\widetilde{W}_j$ be the sequence defined as in Equation* (36) *Then, it holds*

$$(A - p_j I)\widetilde{W}_j = r_j(A_1)U + M_j, \qquad \|M_j\|_F \leq j\epsilon\|U\|_F, \qquad r_j(z) := \prod_{i=1}^{j} \frac{z-p_i}{z-q_i}.$$

Proof. For $j = 1$, the claim follows directly from the assumptions. For $j > 1$, we have

$$\begin{aligned}(A_1 - p_{j+1}I)\widetilde{W}_{j+1} &= (A_1 - p_{j+1}I)(A_1 - q_{j+1}I)^{-1}\left[(A_1 - p_jI)\widetilde{W}_j + \eta_{j+1}\right] \\ &= \frac{r_{j+1}}{r_j}(A_1)\left[r_j(A_1)U + M_j + \eta_{j+1}\right] \\ &= r_{j+1}(A_1)U + \frac{r_{j+1}}{r_j}(A_1)\left(M_j + \eta_{j+1}\right).\end{aligned}$$

The claim follows by setting $M_{j+1} := \frac{r_{j+1}}{r_j}(A_1)\left(M_j + \eta_{j+1}\right)$, and using the property $r_{j+1}(z)/r_j(z) = |z - p_{j+1}|/|z - q_{j+1}| \leq 1$ over the spectrum of A_1. □

Remark 4. *We remark that the level of accuracy required in* (36) *is guaranteed up to second order terms in ϵ, whenever the residual norm for the linear systems $(A_1 - q_{j+1}I)\widetilde{W}_{j+1} = (A_1 - p_jI)\widetilde{W}_j + \eta_{j+1}$ satisfies $\|\eta_{j+1}\| \leq \epsilon\|(A_1 - p_jI)\widetilde{W}_j\|_F$. Indeed, the argument used in the proof of Lemma 4.2 can be easily modified to get*

$$\|M_j\|_F \leq [(1+\epsilon)^j - 1]\|U\|_F = j\epsilon\|U\|_F + \mathcal{O}(\epsilon^2).$$

The slightly more restrictive choice made in (36)*, allows us to obtain more readable results.*

We can then quantify the impact of carrying out the fADI iteration with inexactness in one of the two sequences on the residual norm.

Theorem 4.3. *Let us consider the solution of equation* (19) *by means of the fADI method using shifts satisfying the property $|z - p_j| \leq |z - q_j|$ for any $z \in [\alpha_1, \beta_1]$, and let $\epsilon > 0$ such that the linear systems defining $\widetilde{W}_j$ are solved inexactly as in Equation* (36)*. Then, the computed solution $\delta\widetilde{X}_s$ verifies:*

$$\|A_1\delta\widetilde{X}_s + \delta\widetilde{X}_sA_2 - UV^*\|_F \leq \epsilon_{s,\mathrm{ADI}} + 2s\epsilon\|U\|_F\|V\|_2,$$

where $\epsilon_{s,\mathrm{ADI}} := \|A_1\delta X_s + \delta X_sA_2 - UV^*\|_F$ is the norm of the residual after s steps of the fADI method in exact arithmetic.

Proof. We indicate with $\delta\widetilde{X}_j$ the inexact solution computed at step j of fADI. We note that $\delta\widetilde{X}_1$ corresponds to the outcome of one exact fADI iteration for the slightly modified right-hand side $(U + \eta_1)V^*$; hence, by Equation (23) $\delta\widetilde{X}_1$ satisfies the residual equation:

$$A_1\delta\widetilde{X}_1 + \delta\widetilde{X}_1A_2 - (U + \eta_1)V^* = -r_1(A_1)(U + \eta_1)V^*r_1(-A_2)^{-1}.$$

which allows to express the residual on the original equation as follows:

$$A_1\delta\widetilde{X}_1 + \delta\widetilde{X}_1A_2 - UV^* = -r_1(A_1)(U + \eta_1)V^*r_1(-A_2)^{-1} + \eta_1V^*.$$

We now derive an analogous result for the update $\Delta_{j+1} := \delta\widetilde{X}_{j+1} - \delta\widetilde{X}_j = (q_{j+1} - p_{j+1})\widetilde{W}_{j+1}Y_{j+1}^*$ where $\Delta_1 = \delta\widetilde{X}_1$ by setting $\delta\widetilde{X}_0 = 0$. We prove that for any j the correction Δ_{j+1} satisfies

$$\begin{aligned}A_1\Delta_{j+1} + \Delta_{j+1}A_2 = &\left[(A_1 - p_jI)\widetilde{W}_j + \eta_{j+1}\right]V^*r_j(-A_2)^{-1} \\ &- \frac{r_{j+1}}{r_j}(A_1)\left[(A_1 - p_jI)\widetilde{W}_j + \eta_{j+1}\right]V^*r_{j+1}(-A_2)^{-1}, \qquad j \geq 1.\end{aligned}$$

We verify the claim for $j = 1$; $\widetilde{W}_2$ is defined by the relation

$$(A_1 - q_2 I)\widetilde{W}_2 = (A_1 - p_1 I)\widetilde{W}_1 + \eta_2 = r_1(A_1)(U + \eta_1) + \eta_2.$$

Hence, as discussed in the beginning of this proof, $\widetilde{W}_2$ is the outcome of one exact iteration of fADI applied to the equation $A_1\delta X + \delta X A_2 - (r_1(A_1)(U + \eta_1) + \eta_2)V^* r_1(-A_2)^{-1} = 0$. Hence, thanks to Equation (23) the residual of the computed update Δ_2 verifies

$$\begin{aligned} A_1\Delta_2 + \Delta_2 A_2 - (r_1(A_1)(U + \eta_1) + \eta_2)V^* r_1(-A_2)^{-1} \\ = -\frac{r_2}{r_1}(A_1)\left[r_1(A_1)(U + \eta_1) + \eta_2)V^*\right] r_2(-A_2)^{-1} \end{aligned}$$

Then, the claim follows for $j = 1$ noting that $(A - p_1 I)\widetilde{W}_1 = r_1(A_1)(U + \eta_1)$ using the first relation in Equation (36). For $j > 1$ we have $(A_1 - q_{j+1}I)\widetilde{W}_{j+1} = (A - p_j I)\widetilde{W}_j + \eta_{j+1}$ and with a similar argument Δ_{j+1} is obtained as a single fADI iteration in exact arithmetic for the equation $A_1\delta X + \delta X A_2 - ((A - p_j I)\widetilde{W}_j + \eta_{j+1})V^* r_j(-A_2)^{-1} = 0$, and in view of the residual equation (23)

$$\begin{aligned} A_1\Delta_{j+1} + \Delta_{j+1}A_2 - \left[(A_1 - p_j I)\widetilde{W}_j + \eta_{j+1}\right] V^* r_j(-A_2)^{-1} \\ = -\frac{r_{j+1}}{r_j}(A_1)\left[(A_1 - p_j I)\widetilde{W}_j + \eta_{j+1}\right] V^* r_{j+1}(-A_2)^{-1}. \end{aligned}$$

We now write $\delta\widetilde{X}_s = \sum_{j=1}^{s}\Delta_j$, so that by linearity the residual associated with $\delta\widetilde{X}_s$ satisfies

$$\begin{aligned} A_1\delta\widetilde{X}_s + \delta\widetilde{X}_s A_2 - UV^* &= A_1\Delta_1 + \Delta_1 A_2 - UV^* + \sum_{j=1}^{s-1}(A_1\Delta_{j+1} + \Delta_{j+1}A_2) \\ &= -r_1(A_1)(U + \eta_1)V^* r_1(-A_2)^{-1} + \eta_1 V^* \\ &+ \sum_{j=1}^{s-1}\Bigg\{ -\frac{r_{j+1}}{r_j}(A_1)\left[(A_1 - p_j I)\widetilde{W}_j + \eta_{j+1}\right] V^* r_{j+1}(-A_2)^{-1} \\ &+ \left[(A_1 - p_j I)\widetilde{W}_j + \eta_{j+1}\right] V^* r_j(-A_2)^{-1}\Bigg\}. \end{aligned}$$

We now observe that, thanks to Equation (36)

$$\frac{r_{j+1}}{r_j}(A_1)\left[(A_1 - p_j I)\widetilde{W}_j - \eta_{j+1}\right] = (A_1 - p_{j+1}I)\widetilde{W}_{j+1}.$$

Plugging this identity in the summation yields

$$\begin{aligned} &\sum_{j=1}^{s-1}\left\{\left[(A_1 - p_j I)\widetilde{W}_j + \eta_{j+1}\right] V^* r_j(-A_2)^{-1} - \tfrac{r_{j+1}}{r_j}(A_1)\left[(A_1 - p_j I)\widetilde{W}_j + \eta_{j+1}\right] V^* r_{j+1}(-A_2)^{-1}\right\} \\ &= \sum_{j=1}^{s-1}\left\{\left[(A_1 - p_j I)\widetilde{W}_j + \eta_{j+1}\right] V^* r_j(-A_2)^{-1} - (A - p_{j+1}I)\widetilde{W}_{j+1}V^* r_{j+1}(-A_2)^{-1}\right\} \\ &= (A - p_1 I)\widetilde{W}_1 V^* r_1(-A_2)^{-1} - (A_1 - p_s I)\widetilde{W}_s V^* r_s(-A_2)^{-1} + \sum_{j=1}^{s-1}\eta_{j+1}V^* r_j(-A_2)^{-1} \end{aligned}$$

Summing this with the residual yields

$$A_1\delta\widetilde{X}_s + \delta\widetilde{X}_s A_2 - UV^* = -(A - p_s I)\widetilde{W}_s V^* r_s(-A_2)^{-1} + \sum_{j=0}^{s-1} \eta_{j+1} V^* r_j(-A_2)^{-1},$$

where we have used the relation $r_1(A_1)(U + \eta_1) = (A - p_1 I)\widetilde{W}_1$ from Equation (36). In view Lemma 4.2 we can write $(A - p_s I)\widetilde{W}_s = r_s(A_1)U + M_s$, where $\|M_s\|_F \le s\epsilon$. Therefore,

$$\begin{aligned}\|A_1\delta\widetilde{X}_s + \delta\widetilde{X}_s A_2 + UV^*\|_F &\le \|r_s(A_1)UV^* r_s(-A_2)^{-1}\|_F + s\epsilon\|U\|_F\|V\|_2 \\ &\quad + \sum_{j=0}^{s-1} \|\eta_{j+1} V^* r_j(-A_2)^{-1}\|_F \\ &\le \epsilon_{ADI,s} + 2s\epsilon\|U\|_F\|V\|_2.\end{aligned}$$

□

Remark 5. *We note that, the error in Theorem 4.3 depends on $\|U\|_F\|V\|_2$; this may be larger than $\|UV^*\|_F$, which is what we need to ensure the relative accuracy of the algorithm. However, under the additional assumption that V has orthogonal columns, we have $\|UV^*\|_F = \|U\|_F\|V\|_2$. We can always ensure that this condition is satisfied by computing a thin QR factorization of V, and right-multiplying U by R^*. This does not increase the complexity of the algorithm.*

4.1.2 Residual bounds with inexactness

We can now exploit the results on inexact fADI to control the residual norm of the approximate solution returned by Algorithm 4, assuming that all the update equations are solved with a fixed number s of fADI steps with optimal shifts.

Theorem 4.4. *Let $A_1, \dots, A_d \in \mathbb{R}^{n\times n}$, symmetric positive definite with spectrum contained in $[\alpha, \beta]$, and $\kappa := \frac{\beta}{\alpha}$. Moreover, assume that the A_js are HSS matrices of HSS rank k, with a partitioning of depth ℓ. Let $\epsilon > 0$ and suppose that* LOW_RANK_SYLV *uses fADI with the s zeros and poles of the extremal rational function for $Z_s([\alpha, \beta], [-(d-1)\beta, -\alpha])$ as input parameters, with right-hand side reorthogonalized as described in Remark 5. If*

$$s \ge \frac{1}{\pi^2} \log\left(2\frac{d\kappa}{\epsilon}\right) \log\left(8\frac{(\alpha + (d-1)\beta)(\alpha+\beta)}{d\alpha\beta}\right)$$

and Algorithm 1 solves the Sylvester equations at the base of the recursion with residual bounded by ϵ times the norm of the right-hand side, then the solutions $\widetilde{\mathcal{X}}$ computed by Algorithm 4 satisfies:

$$\|\widetilde{\mathcal{X}} \times_1 A_1 + \cdots + \widetilde{\mathcal{X}} \times_d A_d - \mathcal{B}\|_F \le \left(\left[\kappa(\ell+1)^2\right]^{d-1} (1+2s)^{d-2}\epsilon\right) \|\mathcal{B}\|_F.$$

Proof. Let $\epsilon_{\mathrm{lr},d}$ be the relative residual at which the low-rank update equations with d modes are solved in the recursion and let $\epsilon_d := (\ell+1)^2\kappa\epsilon_{\mathrm{lr},d}$. Note that, thanks to Lemma 4.1, we have

$$\|\widetilde{\mathcal{X}} \times_1 A_1 + \cdots + \widetilde{\mathcal{X}} \times_d A_d - \mathcal{B}\|_F \le \epsilon_d \|\mathcal{B}\|_F,$$

so that ϵ_d is an upper bound for the relative residual of the target equation. Moreover, using the error bound for inexact fADI of Theorem 4.3, we can write $\epsilon_{\mathrm{lr},d} \le (1+2s)\epsilon_{d-1}$, which implies

$$\epsilon_d \le \begin{cases} (\ell+1)^2\kappa\epsilon & d = 2 \quad \text{(Theorem 3.6)} \\ (\ell+1)^2\kappa(1+2s)\epsilon_{d-1}, & d \ge 3. \end{cases},$$

where ϵ is $Z_s([\alpha, \beta], [-(d-1)\beta, -\alpha])$. Expanding the recursion yields the sought bound. □

Theorem 4.4 bounds the residual error with a constant depending on κ^{d-1}, which can often be pessimistic. This term arises when bounding $\|\Xi^{(h,j)}\|_F$ with $\|A_j^{(h)}\|_2$ multiplied by $\|\mathcal{X}^{(h+1)}\|_F$. When the A_js are M-matrices, this can be improved, by replacing κ with $\sqrt{\kappa}$.

Corollary 4.5. *Under the same hypotheses of Theorem 4.4 and the additional assumption that the A_ts are symmetric positive definite M-matrices, we have*

$$\|\widetilde{\mathcal{X}} \times_1 A_1 + \cdots + \widetilde{\mathcal{X}} \times_d A_d - \mathcal{B}\|_F \leq \left(\left[d\sqrt{\kappa}(\ell+1)^2 \right]^{d-1} (1+2s)^{d-2} \epsilon \right) \|\mathcal{B}\|_F.$$

Proof. By means of the same argument used in Lemma 3.7 we have that $\|\Xi^{(h,j)}\| \leq \sqrt{\kappa}\|\mathcal{B}\|_F$ and $\|\Xi^{(h,j)} - \widetilde{\Xi}^{(h,j)}\|_F \leq \sqrt{\kappa}\|\mathcal{R}^{(\ell)} + \ldots + \mathcal{R}^{(h+1)}\|_F$. Plugging these bounds in the proof of Lemma 4.1 yields the inequality

$$\begin{aligned}\|\mathcal{R}^{(h)}\|_F &\leq \epsilon \sum_{j=1}^{d} \left[\|\Xi^{(h,j)}\|_F + \|\widetilde{\Xi}^{(h,j)} - \Xi^{(h,j)}\|_F \right] \\ &\leq d\epsilon\sqrt{\kappa} \left[\|\mathcal{B}\|_F + \|\mathcal{R}^{(\ell)}\|_F + \ldots + \|\mathcal{R}^{(h+1)}\|_F \right].\end{aligned}$$

Then, following the same steps of Lemma 4.1 yields

$$\|\mathcal{R}^{(h)}\|_F \leq d\sqrt{\kappa}\epsilon(1+\epsilon)(1+d\sqrt{\kappa}\epsilon)^{\ell-h-1}\|\mathcal{B}\|_F.$$

Using this bound in Theorem 4.4 yields the claim. □

4.2 Complexity analysis

Theorem 3.9 can be generalized to the d-dimensional case, providing a complexity analysis when nested solves are used.

Theorem 4.6. *Let $A_i \in \mathbb{R}^{n_i \times n_i}$ $n_i = 2^{\ell_i} n_{\min}$, with $n_1 \geq n_2 \geq \cdots \geq n_d$, $n_d \geq skd$, $n_d \geq \log(n_1/n_d)$, and assume that A_i are HSS matrices of HSS rank k, with a partitioning of depth ℓ_i obtained by halving the dimension at every level, for $i = 1, \ldots, d$. Then, Algorithm 4 costs:*

- *(i) $\mathcal{O}((k(d + \log(n_1)) + n_{\min} + sk^2)n_1 \ldots n_d)$ if* LOW_RANK_SYLV *implements s steps of the fADI method,*
- *(ii) $\mathcal{O}((k(d + \log(n_1)) + n_{\min} + s^2k^2)n_1 \ldots n_d)$ if* LOW_RANK_SYLV *implements s steps of the rational Krylov method.*

Proof. We only prove (i) because (ii) is completely analogous.

Let us assume that we have r different mode sizes and each of those occurs d_h times, i.e.:

$$\underbrace{n_1 = n_2 = \cdots = n_{i_1}}_{d_1} > \underbrace{n_{i_1+1} = \cdots = n_{i_2}}_{d_2} > \cdots > \underbrace{n_{i_{r-1}+1} = \cdots = n_{i_r}}_{d_r}.$$

We proceed by (bivariate) induction over $d \geq 2$ and $\ell_1 \geq 0$; the cases $d = 2$ and $\ell_1 \geq 0$ are given by Theorem 3.9 and Corollary 3.10. For $d > 2$ and $\ell_1 = 0$, we have that $n := n_1 = \cdots = n_d = n_{\min}$ and Algorithm 4 is equivalent to Algorithm 1 whose cost is $\mathcal{O}(n^{d+1}) = \mathcal{O}(n_{\min} n^d)$; so also in this case the claim is true. For $d > 2$ and $\ell_1 > 0$, the algorithm begins by halving the first d_1

modes generating 2^{d_1} subproblems having dominant size $n_1/2 = 2^{\ell_1 - 1} n_{\min}$. By induction these subproblems cost

$$\mathcal{O}\left(2^{d_1}(k(d+\log(n_1/2)) + n_{\min} + sk^2)(n_{i_1}/2)^{d_1} n_{i_2}^{d_2} n_{i_3}^{d_3} \dots n_{i_r}^{d_r}\right)$$
$$= O\left((k(d+\log(n_1)) + n_{\min} + sk^2)\prod_{i=1}^{d} n_i\right).$$

Then, we focus on the cost of the update equations and of the tensor times (block) vector multiplications, in this first phase of Algorithm 4. The procedure generates d_1 update equations of size $n_1 \times \cdots \times n_d$. The cost of each call to LOW_RANK_SYLV is dominated by the complexity of solving sk (shifted) linear systems with a Kronecker sum structured matrix with $d-1$ modes. By induction, the cost of all the calls to LOW_RANK_SYLV is bounded by

$$O\left(skd_1(k(d+\log(n_1)) + n_{\min} + sk^2)\prod_{j=1}^{d-1} n_j\right)$$
$$= \mathcal{O}\left((k(d+\log(n_1)) + n_{\min} + sk^2)\prod_{j=1}^{d} n_j\right).$$

Finally, since the tensor times (block) vector multiplications are in one-to-one correspondence with the calls to LOW_RANK_SYLV, we have that the algorithm generates d_1 products of complexity $\mathcal{O}(kn_1 \dots n_d)$. Adding the contribution $\mathcal{O}(dkn_1 \dots n_d)$ to the cost of the subproblems provides the claim. □

5 Numerical experiments

We now test the proposed algorithm against some implementations of Algorithm 1 where the explicit diagonalization of the matrix coefficients A_t is either done in dense arithmetic or via the algorithm proposed in [25]. Note that, the dense solver delivers accuracy close to machine precision while the other approaches aim at a trading some accuracy for a speedup in the computational time. We assess this behavior on 2D and 3D examples.

5.1 Details of the implementation

An efficient implementation of Algorithm 3 and Algorithm 4 takes some care. In particular:

- In contrast to the numerical tests of Section 3.4, the number of Zolotarev shifts for fADI and RK is adaptively chosen on each level of the recursion to ensure the accuracy described in (15). More precisely, this requires estimates of the spectra of the matrix coefficients $A_t^{(h)}$ at all levels of recursion h; this is done via the MATLAB built-in function `eigs`. However, since $A_i^{(h)}$ appears in $2^{(d-1)h}$ equations, estimating the spectra in each recursive call would incur in redundant computations. Therefore, we precompute estimates of the spectra for each block before starting Algorithm 3 and 4, by walking the cluster tree.
- Since the A_ts are SPD, we remark that the correction equation can be slightly modified to obtain a right-hand side with half of the rank. This is obtained by replacing A_j^{off} with a low-rank matrix having suitable non-zero diagonal blocks. See [18, Section 4.4.2] for the details on this idea. This is crucial for problems with higher off-diagonal ranks, and is used in the 2D Fractional cases described below.

- A few operations in Algorithm 4 are well suited for parallelism: the solution of the Sylvester equations at the base of the recursions are all independent, and the same holds for the body of the for loop at lines 11-14. We exploit this fact by computing all the solutions in parallel using multiple cores in a shared memory environment.
- When the matrices A_t are both HSS and banded, they are represented within the sparse format and the sparse direct solver of MATLAB (the backslash operator) is used for the corresponding system solving operations. Note that, in this case the peculiar location of the nonzero entries makes easy to construct the low-rank factorizations of the off-diagonal blocks.

In addition to fADI and rational Krylov, we consider another popular low-rank solver for Sylvester equations: the *extended Krylov* method [28] (EK). The latter corresponds to the rational Krylov method where the shift parameters alternate between the values 0 and ∞. In particular, EK's iteration leverages the precomputation of the Cholesky factorization of the coefficient matrices, as the shift parameters does not vary. A slight downside is that we do not have a priori bounds on the error, and we have to monitor the residual norm throughout the iterations to detect convergence.

An implementation of the proposed algorithms is freely available at `https://github.com/numpi/teq_solver`, and requires `rktoolbox`[3] [7] and `hm-toolbox`[4] [23] as external dependencies. The repository contains the numerical experiments included in this document, and includes a copy of the SuperDC solver[5] [25].

The experiments have been run on a server with two Intel(R) Xeon(R) E5-2650v4 CPU with 12 cores and 24 threads each, running at 2.20 GHz, using MATLAB R2021a with the Intel(R) Math Kernel Library Version 11.3.1. The examples have been run using the SLURM scheduler, allocating 8 cores and 240 GB of RAM.

5.2 Laplace and fractional Laplace equations

In this first experiment we validate the asymptotic complexity of Algorithm 3 and compare the performances of various low-rank solvers for the update equation. As case studies we select two instances of the matrix equation $AX + XA = B$. In one case $A \in \mathbb{R}^{n\times n}$ is chosen as the usual central finite difference discretization of the 1D Laplacian. In the other case A is the Grünwald-Letnikov finite difference discretization of the 1D fractional Laplacian with order 1.5, see [21, Section 2.2.1]. In both cases the reference solution X is randomly generated by means of the MATLAB command `randn(n)`, and B is computed as $B := AX + XA$. We remark that for both equations the matrix A is SPD and HSS; in particular for the Laplace equation A is tridiagonal and stored in the sparse format, while for the fractional Laplace equation it has the rank structure depicted in Figure 3. For the Laplace equation we set $n_{\min} = 512$. In view of the higher off-diagonal ranks of the fractional case, we consider the larger minimal block size $n_{\min} = 2048$. In the next section we will investigate how varying this parameter affects the performances.

We consider increasing sizes $n = 2^j$, $j = 10, \dots, 15$ and the following solvers:

diag Algorithm 1 with explicit diagonalization of the matrix A performed in dense arithmetic.

dst Algorithm 1 incorporating the fast diagonalization by means of the Discrete Sine Transform (DST). This approach is only considered for the 2D Laplace equation.

[3] `https://rktoolbox.org`
[4] `https://github.com/numpi/hm-toolbox`
[5] `https://github.com/fastsolvers/SuperDC`

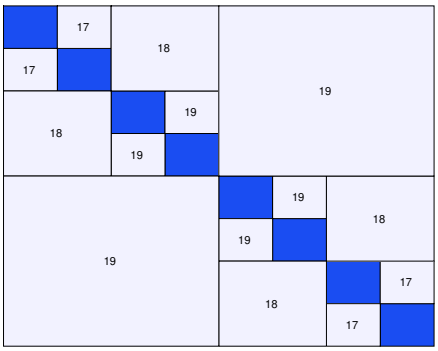

Figure 3: Hierarchical low-rank structure of the 1D fractional Laplace operator discretized through Grünwald-Letnikov finite differences. The blue blocks are dense, while the gray blocks are stored as low-rank matrices whose rank is indicated by the number in the center.

superdc solver implementing Algorithm 1 using the fast diagonalization for SPD HSS matrices described in [25].

dc_adi Algorithm 3 where the fADI iteration is used as LOW_RANK_SYLV.

dc_rk Algorithm 3 where the rational Krylov method is used as LOW_RANK_SYLV.

dc_ek Algorithm 3 where the extended Krylov method is used as LOW_RANK_SYLV.

The shifts used in **dc_adi** and **dc_rk** are the optimal Zolotarev zeros and poles. The number of shifts is chosen to obtain similar residual norms $\text{Res} := \|A\widetilde{X} + \widetilde{X}A - B\|_F / \|B\|_F$ of about 10^{-10}.

We start by comparing the different implementation of Algorithm 3 with **diag**. A detailed comparison with **dst** and **superdc** is postponed to Section 5.4.

The running times and residuals are reported in Table 1 for the Laplace equation. The fractional case is reported in Table 2, for which we do not report the timings for $n = 1024, 2048$ since our choice of $n_{\min}$ makes Algorithm 3 equivalent to Algorithm 1.

For both experiments, using fADI as low-rank solver yields the cheapest method. We remark that, in the fractional case, **dc_ek** outperforms **dc_rk** since the precomputation of the Cholesky factorization of A (and of its sub blocks) makes the iteration of extended Krylov significantly cheaper than the one of rational Krylov.

In Figure 4 and 5 we display how the time is distributed among the various subtasks of **dc_adi**, i.e., the time spent on solving dense matrix equations, computing the low-rank updates, forming the RHS of the update equation and updating the solution, and estimating the spectra.

5.3 Varying the block size

We perform numerical tests similar to the ones of the previous section, but we only consider the low-rank solver fADI for the update equations and, instead, we vary the minimal block size, aiming at determining the best block-size for each test problem. Table 1 and Table 2 report the results concerning $n_{\min} \in \{256, 512, 1024\}$ for the 2D Laplace equation, and $n_{\min} \in \{2048, 4096, 8192\}$ for the fractional Grünwald-Letnikov finite differences case.

Table 1: Timings and residuals for the solution of the 2D Laplace equation of size $n \times n$ by means of Algorithm 3 using different low rank solvers, and $n_{\min} = 512$.

	diag		**dc_adi** $n_{\min} = 512$		**dc_rk** $n_{\min} = 512$		**dc_ek** $n_{\min} = 512$	
n	Time	Res	Time	Res	Time	Res	Time	Res
1,024	0.2	$2.9 \cdot 10^{-13}$	0.6	$2.5 \cdot 10^{-10}$	0.6	$1.4 \cdot 10^{-10}$	0.5	$3.1 \cdot 10^{-10}$
2,048	0.9	$3.9 \cdot 10^{-13}$	1.0	$3.1 \cdot 10^{-10}$	2.0	$1.5 \cdot 10^{-10}$	2.4	$4.0 \cdot 10^{-10}$
4,096	5.9	$1.2 \cdot 10^{-12}$	3.5	$3.4 \cdot 10^{-10}$	8.4	$1.5 \cdot 10^{-10}$	10.5	$4.4 \cdot 10^{-10}$
8,192	41.9	$3.4 \cdot 10^{-12}$	14.9	$3.7 \cdot 10^{-10}$	35.5	$1.5 \cdot 10^{-10}$	43.2	$4.7 \cdot 10^{-10}$
16,384	320.8	$7.8 \cdot 10^{-12}$	61.4	$3.7 \cdot 10^{-10}$	149.6	$1.5 \cdot 10^{-10}$	181.5	$4.8 \cdot 10^{-10}$
32,768	2,511.3	$8.9 \cdot 10^{-12}$	258.1	$3.7 \cdot 10^{-10}$	625.8	$1.5 \cdot 10^{-10}$	760.0	$4.9 \cdot 10^{-10}$

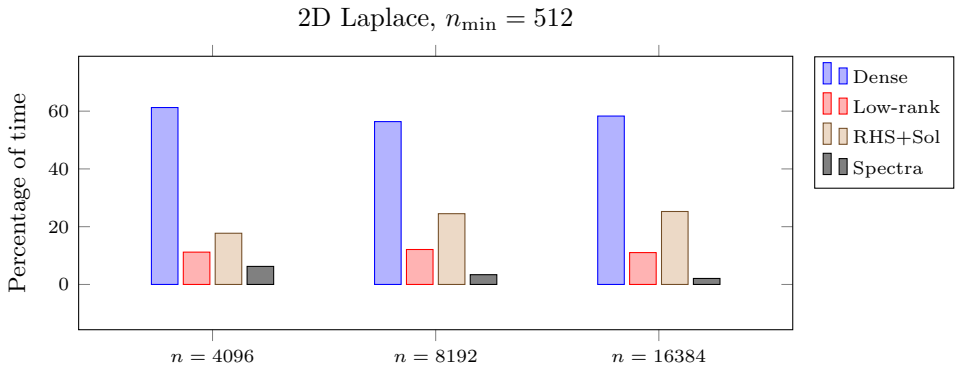

Figure 4: Distribution of the time spent in the different tasks in Algorithm 3. The results are for some instances of the 2D Laplace equation considered in Section 5.2 with fADI as low-rank solver and $n_{\min} = 512$.

Table 2: Timings and residuals for the solution of the 2D fractional Laplace equation of size $n \times n$ by means of Algorithm 3 using different low rank solvers, and $n_{\min} = 2048$.

	diag		**dc_adi** $n_{\min} = 2048$		**dc_rk** $n_{\min} = 2048$		**dc_ek** $n_{\min} = 2048$	
n	Time	Res	Time	Res	Time	Res	Time	Res
4,096	16.0	$8.3 \cdot 10^{-15}$	20.9	$4.5 \cdot 10^{-11}$	24.7	$1.6 \cdot 10^{-11}$	19.1	$1.2 \cdot 10^{-10}$
8,192	137.2	$1.8 \cdot 10^{-14}$	99.4	$5.9 \cdot 10^{-11}$	120.7	$2.5 \cdot 10^{-10}$	97.8	$1.2 \cdot 10^{-10}$
16,384	1,061.1	$2.3 \cdot 10^{-14}$	439.0	$6.8 \cdot 10^{-11}$	593.2	$2.2 \cdot 10^{-10}$	471.3	$1.6 \cdot 10^{-10}$
32,768	8,297.0	$3.2 \cdot 10^{-14}$	1,998.0	$3.1 \cdot 10^{-10}$	2,948.1	$3.6 \cdot 10^{-10}$	2,467.1	$3.0 \cdot 10^{-10}$

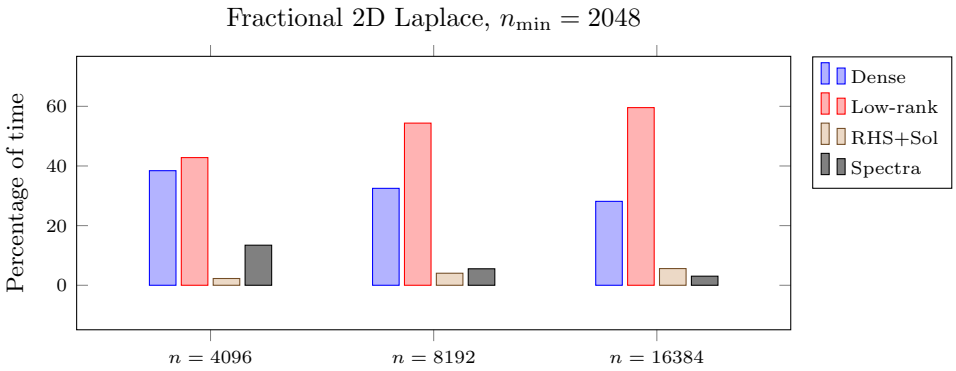

Figure 5: Distribution of the time spentin the different tasks in Algorithm 3. The results are for some instances of the 2D Fractional Laplace equation considered in Section 5.2 with fADI as low-rank solver and $n_{\min} = 2048$.

Table 3: Timings and residuals for the solution of the 2D Laplace equation of size $n \times n$ by means of Algorithm 3 using fADI as low rank solver, with different choices of $n_{\min}$.

	diag		**dc_adi** $n_{\min} = 256$		**dc_adi** $n_{\min} = 512$		**dc_adi** $n_{\min} = 1024$	
n	Time	Res	Time	Res	Time	Res	Time	Res
2,048	1.1	$5.2 \cdot 10^{-13}$	2.0	$3.7 \cdot 10^{-10}$	1.0	$3.1 \cdot 10^{-10}$	1.0	$1.8 \cdot 10^{-10}$
4,096	5.9	$1.3 \cdot 10^{-12}$	5.0	$3.9 \cdot 10^{-10}$	3.5	$3.4 \cdot 10^{-10}$	3.7	$2.0 \cdot 10^{-10}$
8,192	42.0	$3.3 \cdot 10^{-12}$	19.6	$4.2 \cdot 10^{-10}$	14.7	$3.7 \cdot 10^{-10}$	16.5	$2.4 \cdot 10^{-10}$
16,384	319.9	$7.8 \cdot 10^{-12}$	80.4	$4.2 \cdot 10^{-10}$	59.8	$3.7 \cdot 10^{-10}$	67.1	$2.5 \cdot 10^{-10}$
32,768	2,512.5	$8.9 \cdot 10^{-12}$	340.5	$4.2 \cdot 10^{-10}$	259.1	$3.7 \cdot 10^{-10}$	283.3	$2.6 \cdot 10^{-10}$

The results indicate that the choice $n_{\min} = 512$ is ideal for the 2D Laplace case, and $n_{\min} = 4096$ for the fractional case. We expect that problems involving larger off-diagonal ranks will need a larger choice of $n_{\min}$.

5.4 Comparison with 2D state-of-the art solvers

In this section we compare Algorithm 3 with solvers based on fast diagonalization strategies (**dst** and **superdc**). The fast diagonalization procedure of **superdc** requires to set a minimal block size. We have chosen $n_{\min} = 2048$, which yields the best results on the cases of study.

The results are shown in Table 5 and 6. Both **dst** and **superdc** have the quasi-optimal complexity $\mathcal{O}(n^2 \log n)$, as Algorithm 3. Algorithm 3 significantly outperforms **superdc** in all examples, with an acceleration of more than 10x on the largest examples. The performances in the 2D Laplace case are comparable with those of **dst**. On the other hand, **dst** only applies to the specific case of the 2D Laplace equation with constant coefficients.

5.5 3D Laplace equation

We now test the 3D version of the Laplace solver described in Section 5.2. More precisely, we solve the tensor equation

$$\mathcal{X} \times_1 A_1 + \mathcal{X} \times_2 A_2 + \mathcal{X} \times_3 A_3 = \mathcal{B}, \tag{37}$$

Table 4: Timings and residuals for the solution of the 2D fractional Laplace equation of size $n \times n$ by means of Algorithm 3 using fADI as low rank solver, with different choices of $n_{\min}$.

	diag		**dc_adi** $n_{\min} = 2048$		**dc_adi** $n_{\min} = 4096$		**dc_adi** $n_{\min} = 8192$	
n	Time	Res	Time	Res	Time	Res	Time	Res
2,048	2.2	$6.4 \cdot 10^{-15}$	–	–	–	–	–	–
4,096	16.8	$8.2 \cdot 10^{-15}$	21.8	$5.4 \cdot 10^{-11}$	–	–	–	–
8,192	139.3	$1.8 \cdot 10^{-14}$	108.9	$1.2 \cdot 10^{-10}$	100.6	$5.5 \cdot 10^{-11}$	–	–
16,384	1,036.1	$2.3 \cdot 10^{-14}$	524.6	$1.5 \cdot 10^{-10}$	478.9	$8.6 \cdot 10^{-11}$	652.6	$5.5 \cdot 10^{-11}$
32,768	7,934.2	$3.2 \cdot 10^{-14}$	2,337.5	$1.7 \cdot 10^{-10}$	2,034.1	$1.2 \cdot 10^{-10}$	2,741.5	$8.2 \cdot 10^{-11}$

Table 5: Timings and residuals for the solution of the 2D Laplace equation of size $n \times n$ by means of Algorithm 3 using fADI and $n_{\min} = 512$, **superdc** and **dst**.

	dc_adi $n_{\min} = 512$		**dst**		**superdc** $n_{\min} = 2048$	
n	Time	Res	Time	Res	Time	Res
1,024	0.6	$2.5 \cdot 10^{-10}$	0.2	$6.3 \cdot 10^{-14}$		
2,048	1.0	$3.1 \cdot 10^{-10}$	0.9	$1.1 \cdot 10^{-13}$		
4,096	3.5	$3.4 \cdot 10^{-10}$	3.3	$3.3 \cdot 10^{-13}$	15.5	$2.9 \cdot 10^{-12}$
8,192	14.9	$3.7 \cdot 10^{-10}$	13.5	$1.1 \cdot 10^{-12}$	95.3	$4.5 \cdot 10^{-11}$
16,384	61.4	$3.7 \cdot 10^{-10}$	55.1	$2.7 \cdot 10^{-12}$	593.4	$9.6 \cdot 10^{-11}$
32,768	258.1	$3.7 \cdot 10^{-10}$	240.6	$2.4 \cdot 10^{-12}$	3,342.6	$1.5 \cdot 10^{-10}$

Table 6: Timings and residuals for the solution of the 2D Fractional Laplace equation of size $n \times n$ by means of Algorithm 3 using fADI and $n_{\min} = 2048, 4096$, and **superdc**.

	dc_adi $n_{\min} = 2048$		**dc_adi** $n_{\min} = 4096$		**superdc** $n_{\min} = 2048$	
n	Time	Res	Time	Res	Time	Res
4,096	21.8	$5.4 \cdot 10^{-11}$			91.4	$4.0 \cdot 10^{-9}$
8,192	108.9	$1.2 \cdot 10^{-10}$	100.6	$5.5 \cdot 10^{-11}$	685.7	$5.3 \cdot 10^{-9}$
16,384	524.6	$1.5 \cdot 10^{-10}$	478.9	$8.6 \cdot 10^{-11}$	4,715.0	$5.7 \cdot 10^{-9}$
32,768	2,337.5	$1.7 \cdot 10^{-10}$	2,034.1	$1.2 \cdot 10^{-10}$	27,546.0	$5.9 \cdot 10^{-9}$

Table 7: Timings and residuals for the solution of the 3D Laplace equation of dimension $n \times n \times n$ by means of Algorithm 3 using fADI as low rank solver, with different choices of $n_{\min}$ for the 3D splitting. The $n_{\min}$ used in the recursive 2D solver is fixed to $n_{\min} = 1024$.

	diag		**dc_adi** $n_{\min} = 128$		**dc_adi** $n_{\min} = 256$		**dc_adi** $n_{\min} = 512$	
n	Time	Res	Time	Res	Time	Res	Time	Res
256	1.0	$9.9 \cdot 10^{-15}$	4.8	$1.6 \cdot 10^{-8}$	–	–	–	–
512	11.5	$1.1 \cdot 10^{-14}$	25.4	$2.1 \cdot 10^{-8}$	18.1	$1.3 \cdot 10^{-8}$	–	–
1,024	144.1	$1.8 \cdot 10^{-14}$	258.0	$2.4 \cdot 10^{-8}$	242.5	$1.7 \cdot 10^{-8}$	193.4	$1.1 \cdot 10^{-8}$

where A_t are finite difference discretization of the 1D Laplacian with zero Dirichlet boundary condition of sizes $n_i \times n_i$. The reference solution $\mathcal{X}$ is randomly generated by means of `randn(n1,n2,n3)`, and $\mathcal{B}$ is set evaluating (37). We remark that in the 3D case we can choose two different block sizes: one for the recursion in the tensor equation, and one for the recursive calls to the 2D solver. In this section we indicate the former with $n_{\min}$ and the latter is set to 1024 in all the examples, with the only exception of the scaling test in Section 5.6, where all block sizes (2D and 3D) are set to 32.

The low-rank solver for the update equations is fADI, and the tolerance ϵ in Algorithm 4 is set to $\epsilon = 10^{-6}$. We consider two test cases.

Test 1 We choose $n = n_1 = n_2 = n_3$ ranging in $\{256, 512, 1024\}$ and the considered block sizes are $n_{\min} \in \{128, 256, 512\}$. Note that, in this test case all the 2D problems in the recursion are solved with the dense method, in view of our choice of the minimal block size. The results are reported in Table 7.

The results show that the dense method is faster for all choices of n and $n_{\min}$, although the scaling suggests that a breakeven point should be reached around $n = 2048$ and $n_{\min}$ 1024. However, this is not achievable with the computational resources at our disposal, since in the case $n = 2048$ the solution cannot be stored in the system memory.

Test 2 We choose the unbalanced dimensions $n_1 \times 512 \times 512$ with $n_1 = 2^j$ for $j = 10, \ldots, 14$. We choose $n_{\min} = 256$ for the 3D splitting. Since the recursion is structured to split larger dimensions first, also in this case the recursive 2D problems are solved with the dense solver. The results in Table 8 confirm the expected linear scaling with respect to n_1, and the approach is faster than the dense solver from dimension $n_1 = 4096$.

5.6 Asymptotic complexity in the 3D case

The previous experiment provides too few data points to assess the expected cubic complexity. In addition, the use of the dense solver does not allow to validate the error analysis that we have performed, and that guarantees that the inexact solving of the subproblems does not destroy the final accuracy.

To validate the scaling and accuracy of Algorithm 4 as n grows, we set the minimal block size to the small value $n_{\min} = 32$, and we measure the timings for problems of size between $n = 64$ and $n = 1024$. The results are reported in Figure 6 and confirm the predicted accuracy and cubic scaling.

Table 8: Timings and residuals for the solution of the 3D Laplace equation of dimension $n_1 \times 512 \times 512$ by means of Algorithm 3 using fADI as low rank solver, with $n_{\min} = 256$ for the 3D splitting, and $n_{\min} = 1024$ for the recursive 2D solver.

	diag		**dc_adi** $n_{\min} = 256$	
n_1	Time	Res	Time	Res
1,024	25.7	$1.3 \cdot 10^{-14}$	41.0	$1.4 \cdot 10^{-8}$
2,048	68.5	$1.5 \cdot 10^{-14}$	87.9	$1.4 \cdot 10^{-8}$
4,096	202.0	$1.4 \cdot 10^{-14}$	187.5	$1.4 \cdot 10^{-8}$
8,192	680.8	$1.5 \cdot 10^{-14}$	404.0	$1.4 \cdot 10^{-8}$
16,384	2,417.9	$1.7 \cdot 10^{-14}$	975.1	$1.4 \cdot 10^{-8}$

	dc_adi $n_{\min} = 32$	
n	Time	Res
64	0.6	$2.3 \cdot 10^{-8}$
128	1.7	$2.9 \cdot 10^{-8}$
256	11.5	$3.4 \cdot 10^{-8}$
512	89.1	$3.6 \cdot 10^{-8}$
1,024	839.8	$7.8 \cdot 10^{-8}$

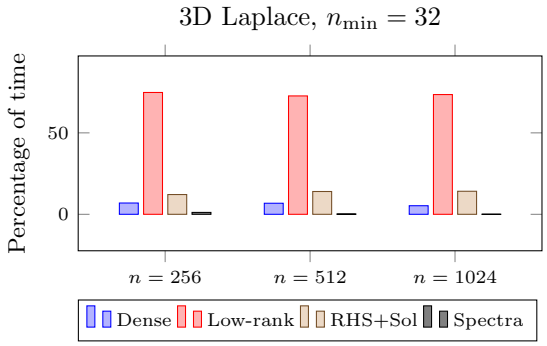

Figure 6: On the left, timings and residuals for the 3D Laplace example in Section 5.6, with $n_{\min} = 32$ and fADI as a low-rank solver for the update equations. On the right, the distribution of the time spent in the different subtasks of Algorithm 4.

In addition, in the right part of Figure 6 we display the time spent in the various parts of Algorithm 4. This highlights that the solution of the update equations dominates the other costs, which is expected in view of the small $n_{\min}$. We remark that the latter include the calls to the 2D solver described in Algorithm 3.

6 Conclusions

We have proposed a new solver for positive definite tensor Sylvester equation with hierarchically low-rank coefficients that attains the quasi-optimal complexity $\mathcal{O}(n^d \log(n))$. Our procedure is based on a nested divide-and-conquer paradigm. We have developed an error analysis that reveals the relation between the level on inexactness in the solution of the nested subproblems and the final accuracy.

The numerical results demonstrate that the proposed solver can significantly speed up the solution of matrix Sylvester equations of medium size. In the 3D-tensor case with equal size, the method is slower than the dense solver based on diagonalization, when addressing sizes up to $1024 \times 1024 \times 1024$. On the other hand, the performances are quite close and we expect that running the simulations in a distributed memory environment or on a machine with a high level of performance would uncover a breakeven point around $2048 \times 2048 \times 2048$.

A further speed up might be reached employing a relaxation strategy for the inexactness of

the linear system solving in fADI or RK [20]. This may provide significant advantages for $d > 2$, where the time spent on solving the equations with low-rank right-hand side is above the 70% of the total.

Another promising direction is to adapt the method to exploit block low-rank structures in the right-hand side; this will subject of future investigations.

References

[1] A. C. Antoulas, *Approximation of large-scale dynamical systems*, vol. 6 of Advances in Design and Control, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2005. With a foreword by Jan C. Willems.

[2] J. Ballani and L. Grasedyck, *A projection method to solve linear systems in tensor format*, Numer. Linear Algebra Appl., 20 (2013), pp. 27–43.

[3] R. H. Bartels and G. W. Stewart, *Solution of the Matrix equation AX + XB = C [F4]*, Commun. ACM, 15 (1972), pp. 820–826.

[4] B. Beckermann, *An error analysis for rational Galerkin projection applied to the Sylvester equation*, SIAM J. Numer. Anal., 49 (2011), pp. 2430–2450.

[5] B. Beckermann and A. Townsend, *On the singular values of matrices with displacement structure*, SIAM J. Matrix Anal. Appl., 38 (2017), pp. 1227–1248.

[6] P. Benner, R.-C. Li, and N. Truhar, *On the ADI method for Sylvester equations*, J. Comput. Appl. Math., 233 (2009), pp. 1035–1045.

[7] M. Berljafa and S. Güttel, *Generalized rational Krylov decompositions with an application to rational approximation*, SIAM J. Matrix Anal. Appl., 36 (2015), pp. 894–916.

[8] S. Börm, *Data-sparse approximation of non-local operator by $\mathscr{H}^2$-matrices*, Linear Algebra Appl., 422 (2007), pp. 380–403.

[9] M. Chen and D. Kressner, *Recursive blocked algorithms for linear systems with Kronecker product structure*, Numer. Algorithms, 84 (2020), pp. 1199–1216.

[10] S. V. Dolgov, *TT-GMRES: solution to a linear system in the structured tensor format*, Russian J. Numer. Anal. Math. Modelling, 28 (2013), pp. 149–172.

[11] S. V. Dolgov and D. V. Savostyanov, *Alternating minimal energy methods for linear systems in higher dimensions*, SIAM J. Sci. Comput., 36 (2014), pp. A2248–A2271.

[12] V. Druskin, L. Knizhnerman, and V. Simoncini, *Analysis of the rational Krylov subspace and ADI methods for solving the Lyapunov equation*, SIAM J. Numer. Anal., 49 (2011), pp. 1875–1898.

[13] D. Fortunato and A. Townsend, *Fast Poisson solvers for spectral methods*, IMA J. Numer. Anal., 40 (2020), pp. 1994–2018.

[14] G. H. Golub, S. Nash, and C. Van Loan, *A Hessenberg-Schur method for the problem $AX + XB = C$*, IEEE Trans. Automat. Control, 24 (1979), pp. 909–913.

[15] W. Hackbusch, *Hierarchical matrices: algorithms and analysis*, vol. 49 of Springer Series in Computational Mathematics, Springer, Heidelberg, 2015.

[16] I. Jonsson and B. Kågström, *Recursive blocked algorithm for solving triangular systems. I. One-sided and coupled Sylvester-type matrix equations*, ACM Trans. Math. Software, 28 (2002), pp. 392–415.

[17] T. G. Kolda and B. W. Bader, *Tensor decompositions and applications*, SIAM Rev., 51 (2009), pp. 455–500.

[18] D. Kressner, S. Massei, and L. Robol, *Low-rank updates and a divide-and-conquer method for linear matrix equations*, SIAM J. Sci. Comput., 41 (2019), pp. A848–A876.

[19] D. Kressner and C. Tobler, *Krylov subspace methods for linear systems with tensor product structure*, SIAM J. Matrix Anal. Appl., 31 (2009/10), pp. 1688–1714.

[20] P. Kürschner and M. A. Freitag, *Inexact methods for the low rank solution to large scale Lyapunov equations*, BIT, 60 (2020), pp. 1221–1259.

[21] S. Massei, M. Mazza, and L. Robol, *Fast solvers for two-dimensional fractional diffusion equations using rank structured matrices*, SIAM J. Sci. Comput., 41 (2019), pp. A2627–A2656.

[22] S. Massei and L. Robol, *Rational Krylov for Stieltjes matrix functions: convergence and pole selection*, BIT, 61 (2021), pp. 237–273.

[23] S. Massei, L. Robol, and D. Kressner, *hm-toolbox: Matlab software for HODLR and HSS matrices*, SIAM J. Sci. Comput., 42 (2020), pp. C43–C68.

[24] ——, *Hierarchical adaptive low-rank format with applications to discretized partial differential equations*, Numer. Linear Algebra Appl., (2022), p. e2448.

[25] X. Ou and J. Xia, *Superdc: Superfast divide-and-conquer eigenvalue decomposition with improved stability for rank-structured matrices*, SIAM J. Sci. Comput., 44 (2022), pp. A3041–A3066.

[26] D. Palitta and V. Simoncini, *Matrix-equation-based strategies for convection–diffusion equations*, BIT, 56 (2016), pp. 751–776.

[27] T. Penzl, *Eigenvalue decay bounds for solutions of Lyapunov equations: the symmetric case*, Systems Control Lett., 40 (2000), pp. 139–144.

[28] V. Simoncini, *A new iterative method for solving large-scale Lyapunov matrix equations*, SIAM J. Sci. Comput., 29 (2007), pp. 1268–1288.

[29] V. Simoncini, *Computational methods for linear matrix equations*, SIAM Rev., 58 (2016), pp. 377–441.

[30] C. Strössner and D. Kressner, *Fast global spectral methods for three-dimensional partial differential equations*, IMA J. Numer. Anal., (2022).

[31] A. Townsend and S. Olver, *The automatic solution of partial differential equations using a global spectral method*, J. Comput. Phys., 299 (2015), pp. 106–123.

[32] R. Vandebril, M. Van Barel, and N. Mastronardi, *Matrix computations and semiseparable matrices: linear systems*, vol. 1, JHU Press, 2007.

[33] J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li, *Fast algorithms for hierarchically semiseparable matrices*, Numer. Linear Algebra Appl., 17 (2010), pp. 953–976.