ENTER JOURNAL CODE (submitted)

Generic factorizations and alternating direction method for fixed-rank approximation of matrix polynomials

Sweta Das, Andrii Dmytryshyn, Froilán M. Dopico, and Mårten Gulliksson

Abstract. We present an algorithm for approximating a given matrix polynomial of degree d by a matrix polynomial of prescribed rank and degree at most d. The method builds on recent advances in the theory of generic eigenstructures and factorizations of matrix polynomials with bounded rank and degree. We also give a specialized treatment of the case of matrix pencils (degree one matrix polynomials), motivated by its particular theoretical and practical importance. Lastly, numerical experiments are reported to assess the performance of the proposed approach and to compare it with existing methods.

1. Introduction

Low-rank approximation is one of the fundamental mechanisms by which large-scale linear-algebraic models are made tractable. For constant matrices, the theory is classical: the singular value decomposition gives both a structural description of low-rank matrices and, in unitarily invariant norms, optimal approximants. This theory underlies a wide range of methods in numerical linear algebra, systems theory, model reduction, system identification, and signal processing.

In many applications, however, the objects of interest are not constant but polynomial matrices, also known as matrix polynomials, [2, 22, 24]

$$P(\lambda) = \sum_{i=0}^{d} P_i \lambda^i, \qquad P_i \in \mathbb{C}^{m\times n}, \qquad i = 0, \dots, d,$$

where $\mathbb{C}^{m\times n}$ is the set of $m \times n$ matrices with entries from the field of complex numbers $\mathbb{C}$.

Matrix polynomials arise naturally in polynomial eigenvalue problems, linear systems and control, signal processing, and various forms of structured modeling [2, 24]. Their rank is the *normal rank*, that is the rank of $P(\lambda)$ over the field of rational functions $\mathbb{C}(\lambda)$, and

Mathematics Subject Classification 2020: 65F55 (primary); 65K10, 15A22, 15A54 (secondary).
Keywords: matrix polynomials, generic factorization, fixed-rank approximation, low-rank matrix approximation, alternating direction method.

is denoted by $\operatorname{nrank}(P(\lambda))$. Equivalently, $\operatorname{nrank}(P(\lambda)) = \max_{\mu\in\mathbb{C}} \operatorname{rank}(P(\mu))$. The normal rank is a global property of a matrix polynomial and depends on all its coefficients. Consequently, the geometry of bounded-rank matrix polynomials is far more intricate than that of bounded-rank constant matrices, and this fundamentally changes the nature of the low-rank approximation problem. Despite the importance of matrix polynomials in applications, a reliable framework for their low-rank approximation has therefore remained elusive.

The purpose of this paper is to develop such a framework. Our approach is based on recent advances in the theory of generic eigenstructures and factorizations of matrix polynomials of bounded normal rank [8, 10]. These results identify generic families of matrix polynomials with prescribed rank and provide explicit parametrizations of such families. We use these parametrizations to reformulate the constrained low-rank approximation problem

$$\min_{S(\lambda)} \|P(\lambda) - S(\lambda)\|_F, \quad \text{such that} \quad \deg(S(\lambda)) \le d, \quad \operatorname{nrank}(S(\lambda)) \le r, \tag{1.1}$$

as an unconstrained problem (presented in (4.13)), that allows efficient and reliable solution.

Our framework applies to matrix polynomials of arbitrary degree and to any prescribed target rank $r < \min(m, n)$. Its main feature is that the rank and degree constraints are built into the parametrization. Thus every polynomial produced by the algorithm has normal rank at most r, avoiding a common difficulty of losing the rank condition in methods that impose low rank only indirectly or through penalty terms. The resulting optimization problem is then approached using the alternating least-squares method. The parametrized formulation also reflects the intrinsic dimension of the rank-constrained family: as the target rank decreases, so does the number of parameters. This makes the approach particularly attractive for computing very-low-rank approximations of matrix polynomials.

We deliberately refer to the construction as a framework rather than as a single algorithm. It consists of several largely independent components, e.g., the choice of generic rank-r families, the choice of a parametrization of these families, and the numerical method used to solve the resulting optimization problem. Each of them can be modified, providing us with the flexibility to create algorithms tailored to particular applications, including structure-preserving approximations and more scalable optimization schemes.

The paper is organized as follows. Section 2 reviews the current state of the art on low-rank approximation of matrix polynomials and related special cases. In Sections 3 and 4, we develop the proposed framework first for matrix pencils and then for matrix polynomials of arbitrary degree, including the necessary theoretical background. Section 5 discusses convergence properties of the resulting iterative schemes. Numerical experiments illustrating the behavior and computational performance of the proposed approach are presented in Section 6. We end with conclusions and some future research directions in Section 7.

2. State-of-the-art for approximations with matrix polynomials

We naturally divide this section by the degree of the considered polynomials. Note also that the majority of the results mentioned here are dealing with the partial case of the problem (1.1), defined by $r = \min(m, n) - 1$. For $m = n$, this case is known as the "distance-to-singularity problem" and is motivated by the need for robust computations; see, for example, [15], where the proximity to singularity causes instability in a delay differential equation.

The case of matrices, $d = 0$**.** The problem of approximating a given matrix A by a matrix of prescribed rank $r < \min(m, n)$ is among the best understood problems in numerical linear algebra. The Eckart–Young–Mirsky theorem characterizes the optimal rank-r approximation via the truncated singular value decomposition (SVD) [17], providing both the existence and the conditions of the uniqueness of the best approximant as well as a constructive algorithm.

The case of matrix pencils, $d = 1$**.** The problem already becomes complicated in this setting. It was first introduced by Byers, He, and Mehrmann [4] in the special case of $r = \min(m, n) - 1$ and $m = n$. After the characterizations and upper and lower bounds for matrix pencils developed in [4], essentially no further progress was made for almost 20 years due to the intrinsic complexity of the problem [19]. More recently, interest in this type of problem has been growing. In [18], Guglielmi, Lubich, and Mehrmann propose an approach for computing nearest singular pencils based on solving ordinary differential equations. Dopico, Noferini, and Nyman in [11] suggest a method based on Riemannian optimization. It is important to emphasize that both [11, 18] address only the case $r = \min(m, n) - 1$ and $m = n$ too. Several other works, which are also able to treat matrix polynomials, are discussed below.

The case of matrix polynomials, $d \geq 1$**.** Naturally, low-rank approximation for matrix polynomials is significantly more challenging. In particular, extending the methods developed for the pencil case may be extremely difficult or even impossible. The work of Giesbrecht, Haraldson, and Labahn [12] was the first to address the problem for matrix polynomials. More recently, several new approaches have been proposed in [5, 14–16], leading to substantial progress in this area. The first named, introduced by Das and Bora [5], suggests methods based on perturbation theory. In [15], Gnazzo and Guglielmi formulate the distance to singularity problem ($r = \min(m, n) - 1$) for general nonlinear matrix-valued functions as a nonconvex optimization problem with singularity imposed as a constraint; the polynomial case is described in detail in [14]. The third approach, known as the Riemann-Oracle [16], developed by Gnazzo, Noferini, Nyman, and Poloni, provides a general-purpose Riemannian optimization framework for a broad class of matrix nearness problems, including the distance to singularity problem and, in Section 9, a strategy as to how to approach the case of any rank r, though an implementation is missing.

3. Approximation with low-rank matrix pencils

Matrix pencils are matrix polynomials of degree at most one. So, the approach presented in Section 4 for matrix polynomials of arbitrary degree covers the pencil case. Nevertheless, we provide a complete and self-contained treatment of this case in this section. The main motivation is the relative simplicity of matrix pencils, as well as their intrinsic importance. In fact, in the literature on distance problems and low-rank approximations, many contributions focus exclusively on the pencil case, see, e.g., [4, 11, 18, 19]. We believe that presenting this case separately significantly simplifies the understanding of the general polynomial case, since the main steps of the method remain the same, but the technicalities are substantially reduced.

It is also worth noting that the set of bounded-rank matrix pencils admits a description in terms of factorizations (or in terms of a sum of pencils of rank at most one) that does not require taking closures of the involved subsets, in contrast to the general polynomial case (compare Theorems 1 and 2). Although this distinction is conceptually important, it does not play an essential role in the developments of our method.

In Sections 3.2 and 3.3, we formulate the minimization problems for matrix pencils using Theorem 1 and solve them using the alternating least squares method, see, e.g., [21, 23]. The convergence properties of this approach are analyzed in Section 5 in the general polynomial case. Before we proceed, some preliminary information about matrix pencils is summarized in the next subsection.

3.1. Preliminary information about matrix pencils

All the matrices in this paper have complex entries, unless otherwise specified. To avoid repetitions, we define some basic concepts immediately for matrix polynomials. $\mathbb{C}[\lambda]$ denotes the ring of polynomials in one variable λ with coefficients in $\mathbb{C}$, $\mathbb{C}[\lambda]^{m\times n}$ denotes the set of $m \times n$ matrix polynomials, i.e., the set of $m \times n$ matrices with entries from $\mathbb{C}[\lambda]$, and $\mathbb{C}[\lambda]^m$ stands for $\mathbb{C}[\lambda]^{m\times 1}$, i.e., for the set of column vector polynomials with m entries. The degree of a matrix polynomial $P(\lambda) \in \mathbb{C}[\lambda]^{m\times n}$ is the largest degree of its entries and is denoted by $\deg(P)$. The Frobenius norm of $P(\lambda) = \sum_{i=0}^{d} \lambda^i P_i$ is defined in terms of its matrix coefficients as follows $\|P(\lambda)\|_F = \left(\sum_{i=0}^{d} \|P_i\|_F^2\right)^{\frac{1}{2}}$.

We denote by $\mathcal{P}_r^{m\times n}$ the set of $m \times n$ matrix pencils with normal rank at most r, i.e., $\mathcal{P}_r^{m\times n} := \{P_0 + \lambda P_1 : P_0, P_1 \in \mathbb{C}^{m\times n},\ \operatorname{nrank}(P_0 + \lambda P_1) \le r\}$.

Definition 1. [7, Section 2] *Let m, n and r be integers such that $m, n \ge 2$ and $0 < r < \min\{m, n\}$, and a be an integer such that $0 \le a \le r$. Let us define the following subsets of*

$\mathcal{P}_r^{m\times n}$

$$C_{r,a}^{m\times n} = \left\{ u_1(\lambda)v_1(\lambda)^T + \cdots + u_r(\lambda)v_r(\lambda)^T \ : \begin{array}{l} u_i(\lambda) \in \mathbb{C}[\lambda]^m,\ v_j(\lambda) \in \mathbb{C}[\lambda]^n, \\ \deg(u_i) \le 1, \quad \textit{for } i = 1,\ldots,r, \\ \deg(v_j) \le 1, \quad \textit{for } j = 1,\ldots,r, \\ \deg(u_1) = \cdots = \deg(u_a) = 0, \\ \deg(v_{a+1}) = \cdots = \deg(v_r) = 0 \end{array} \right\}.$$

We recall in Theorem 1 a description of $\mathcal{P}_r^{m\times n}$ as the union of the $r+1$ sets of matrix pencils from Definition 1. This description is crucial for our method to compute the closest matrix pencil in $\mathcal{P}_r^{m\times n}$ to a given $m \times n$ matrix pencil $P_0 + \lambda P_1$.

Theorem 1. ([6, Lemma 3.1] and [7, Lemma 4]) *Let $\mathcal{P}_r^{m\times n}$ be the set of $m \times n$ matrix pencils with normal rank at most r, $0 < r < \min\{m,n\}$, and $C_{r,a}^{m\times n}$ be the subset in Definition 1, for $0 \le a \le r$. Then*

$$\mathcal{P}_r^{m\times n} = C_{r,0}^{m\times n} \cup C_{r,1}^{m\times n} \cup \cdots \cup C_{r,r}^{m\times n}. \tag{3.1}$$

For the sake of brevity, we do not explain the nature and properties of the description of $\mathcal{P}_r^{m\times n}$ in Theorem 1, nor its relation with the generic eigenstructures of pencils with bounded rank, but we refer the curious reader to [7, 8, 10] for more details.

3.2. Formulation of the minimization problem for matrix pencils

Let $P_0 + \lambda P_1$ be a given $m \times n$ complex matrix pencil. Our goal is to find a matrix pencil in $\mathcal{P}_r^{m\times n}$ that is nearest to $P_0 + \lambda P_1$ and the distance to it, i.e., to find

$$\begin{aligned} \operatorname{dist}_{\mathcal{P}_r^{m\times n}}(P_0 + \lambda P_1) &= \min_{S_0+\lambda S_1 \in \mathcal{P}_r^{m\times n}} ||(P_0 - S_0) + \lambda(P_1 - S_1)||_F, \\ L_0 + \lambda L_1 \in &\operatorname*{arg\,min}_{S_0+\lambda S_1 \in \mathcal{P}_r^{m\times n}} ||(P_0 - S_0) + \lambda(P_1 - S_1)||_F. \end{aligned} \tag{3.2}$$

The method we develop gives us a minimizer $L_0 + \lambda L_1$ and thus also the distance $\operatorname{dist}_{\mathcal{P}_r^{m\times n}}(P_0 + \lambda P_1)$. For brevity and convenience, in the rest of the section we will not repeat the transformations in the expression for the minimizer in (3.2).

Using Theorem 1, we can substitute the minimization problem (3.2) with $r+1$ minimization problems as follows

$$\begin{aligned} \operatorname{dist}_{\mathcal{P}_r^{m\times n}}(P_0 + \lambda P_1) &= \min_{0\le a\le r} \left(\min_{S_0+\lambda S_1 \in C_{r,a}^{m\times n}} \|(P_0 - S_0) + \lambda(P_1 - S_1)\|_F \right) \\ &= \min_{0\le a\le r} \operatorname{dist}_{C_{r,a}^{m\times n}}(P_0 + \lambda P_1). \end{aligned}$$

Now the explicit description of a pencil $S_0 + \lambda S_1 \in C_{r,a}^{m\times n}$, see Definition 1, i.e.,

$$\begin{aligned} S_0 + \lambda S_1 = {} & u_{0,1}^0 (v_{0,1}^1 + \lambda v_{1,1}^1)^T + \cdots + u_{0,a}^0 (v_{0,a}^1 + \lambda v_{1,a}^1)^T \\ & + (u_{0,a+1}^1 + \lambda u_{1,a+1}^1) v_{0,a+1}^{0,T} + \cdots + (u_{0,r}^1 + \lambda u_{1,r}^1) v_{0,r}^{0,T}, \end{aligned}$$

in terms of arbitrary vectors $u^0_{0,j}$, $1 \le j \le a$, $u^1_{0,j}, u^1_{1,j}$, $(a+1) \le j \le r$, in $\mathbb{C}^m$ and $v^0_{0,j}$, $(a+1) \le j \le r$, $v^1_{0,j}, v^1_{1,j}$, $1 \le j \le a$, in $\mathbb{C}^n$, allows us to formulate the minimization problem for (the square of) $\operatorname{dist}_{C^{m\times n}_{r,a}}(P_0 + \lambda P_1)$, for each a, as the following *unconstrained minimization problem*

$$\min_{\substack{u^0_{0,j},u^1_{0,j},u^1_{1,j}\in\mathbb{C}^m \\ v^0_{0,j},v^1_{0,j},v^1_{1,j}\in\mathbb{C}^n}} \left\| P_0 - \left(\sum_{j=1}^{a} u^0_{0,j} v^{1,T}_{0,j} + \sum_{j=a+1}^{r} u^1_{0,j} v^{0,T}_{0,j} \right) \right\|_F^2 + \left\| P_1 - \left(\sum_{j=1}^{a} u^0_{0,j} v^{1,T}_{1,j} + \sum_{j=a+1}^{r} u^1_{1,j} v^{0,T}_{0,j} \right) \right\|_F^2 . \tag{3.3}$$

Equivalently, using matrix notation, we have for each a

$$\begin{aligned} &(\operatorname{dist}_{C^{m\times n}_{r,a}}(P_0 + \lambda P_1))^2 \\ &\quad = \min_{\substack{U^0_0,U^1_0,U^1_1, \\ V^0_0,V^1_0,V^1_1}} \left\| P_0 - U^1_0 V^{0\,T}_0 - U^0_0 V^{1\,T}_0 \right\|_F^2 + \left\| P_1 - U^1_1 V^{0\,T}_0 - U^0_0 V^{1\,T}_1 \right\|_F^2, \end{aligned} \tag{3.4}$$

where

$$\begin{aligned} U^0_0 &= \begin{bmatrix} u^0_{0,1} & \cdots & u^0_{0,a} \end{bmatrix} \in \mathbb{C}^{m\times a}, & V^0_0 &= \begin{bmatrix} v^0_{0,a+1} & \cdots & v^0_{0,r} \end{bmatrix} \in \mathbb{C}^{n\times(r-a)}, \\ U^1_0 &= \begin{bmatrix} u^1_{0,a+1} & \cdots & u^1_{0,r} \end{bmatrix} \in \mathbb{C}^{m\times(r-a)}, & V^1_0 &= \begin{bmatrix} v^1_{0,1} & \cdots & v^1_{0,a} \end{bmatrix} \in \mathbb{C}^{n\times a}, \\ U^1_1 &= \begin{bmatrix} u^1_{1,a+1} & \cdots & u^1_{1,r} \end{bmatrix} \in \mathbb{C}^{m\times(r-a)}, & V^1_1 &= \begin{bmatrix} v^1_{1,1} & \cdots & v^1_{1,a} \end{bmatrix} \in \mathbb{C}^{n\times a}. \end{aligned}$$

Note that the sizes of the matrices in (3.4) depend on a. In particular, if $a = 0$, then the summands $U^0_0 V^{1\,T}_0$ and $U^0_0 V^{1\,T}_1$ are absent, and if $a = r$, then the summands $U^1_0 V^{0\,T}_0$ and $U^1_1 V^{0\,T}_0$ are absent.

3.3. Solution of the minimization problem over each $C^{m\times n}_{r,a}$

An optimization problem consisting of several independent sets of unknowns may in some cases be efficiently solved in a sequence set by set, i.e., we alternate between solving a problem in which the unknowns of one set are fixed and solving a problem in which the unknowns of another set are fixed. The problem in (3.4) admits several variants of independent sets from which an alternating optimization algorithm can be formulated.

In this work we will use the alternating least squares method, see, e.g., [21,23]. We first summarize the idea of the method in the case of a very simple problem and then describe how it applies for the problem in (3.4). Let $R \in \mathbb{C}^{m\times n}$ be a given matrix, $S \in \mathbb{C}^{m\times p}$, $T \in \mathbb{C}^{p\times n}$ and consider the problem

$$\min_{S,T} \|R - ST\|_F^2 . \tag{3.5}$$

First we fix the matrix S and compute the Matlab's backslash solution $\widehat{T}$ to $\min_T \|R - ST\|_F^2$. Now we take S as the unknown in (3.5) and fix $\widehat{T}$ to solve the transposed problem $\min_S \left\|R^T - \widehat{T}^T S^T\right\|_F^2$, obtaining the Matlab's backslash solution $\widehat{S}$. Then $\widehat{S}$ is fixed to compute with Matlab's backslash the next iteration of T, and so on until convergence.

For our problem (3.4), we choose to alternate between the U- and V-variables, as this allows a nice reformulation of the problem. We (randomly) choose the matrices U_0^0, U_0^1, and U_1^1, resulting in the minimization over V-variables

$$\min_{V_0^0, V_0^1, V_1^1} \left\|P_0 - U_0^1 V_0^{0T} - U_0^0 V_0^{1T}\right\|_F^2 + \left\|P_1 - U_1^1 V_0^{0T} - U_0^0 V_1^{1T}\right\|_F^2,$$

which can be equivalently written as

$$\min_{V_0^0, V_0^1, V_1^1} \left\| \begin{bmatrix} P_0 \\ P_1 \end{bmatrix} - \begin{bmatrix} U_0^1 & U_0^0 & 0 \\ U_1^1 & 0 & U_0^0 \end{bmatrix} \begin{bmatrix} V_0^{0T} \\ V_0^{1T} \\ V_1^{1T} \end{bmatrix} \right\|_F^2. \tag{3.6}$$

Analogously, the minimization problem over the U-variables can be written as

$$\min_{U_0^0, U_0^1, U_1^1} \left\| \begin{bmatrix} P_0^T \\ P_1^T \end{bmatrix} - \begin{bmatrix} V_0^1 & V_0^0 & 0 \\ V_1^1 & 0 & V_0^0 \end{bmatrix} \begin{bmatrix} U_0^{0T} \\ U_0^{1T} \\ U_1^{1T} \end{bmatrix} \right\|_F^2. \tag{3.7}$$

So using a random starting guess for the U-variables and Matlab's backslash command, we solve the problem for the V-variables, i.e., (3.6), then use this solution and Matlab's backslash command for solving the problem for the U-variables, i.e., (3.7), and we alternate between the solution of the least squares problems (3.6) and (3.7) until convergence. See Section 5 for more details on convergence.

Remark 1. *As mentioned at the end of Section 3.2, if $a = 0$, then the summands $U_0^0 V_0^{1T}$ and $U_0^0 V_1^{1T}$ disappear from (3.4), which in turn results in least squares problems (3.6) and (3.7) with the simpler shapes*

$$\min_{V_0^0} \left\| \begin{bmatrix} P_0 \\ P_1 \end{bmatrix} - \begin{bmatrix} U_0^1 \\ U_1^1 \end{bmatrix} V_0^{0T} \right\|_F^2 \quad \text{and} \quad \min_{U_0^1, U_1^1} \left\| \begin{bmatrix} P_0^T \\ P_1^T \end{bmatrix} - \begin{bmatrix} V_0^0 & 0 \\ 0 & V_0^0 \end{bmatrix} \begin{bmatrix} U_0^{1T} \\ U_1^{1T} \end{bmatrix} \right\|_F^2.$$

Similarly, if $a = r$, then the summands $U_0^1 V_0^{0T}$ and $U_1^1 V_0^{0T}$ disappear from (3.4), resulting in the following simplified versions of the problems (3.6) and (3.7)

$$\min_{V_0^1, V_1^1} \left\| \begin{bmatrix} P_0 \\ P_1 \end{bmatrix} - \begin{bmatrix} U_0^0 & 0 \\ 0 & U_0^0 \end{bmatrix} \begin{bmatrix} V_0^{1T} \\ V_1^{1T} \end{bmatrix} \right\|_F^2 \quad \text{and} \quad \min_{U_0^0} \left\| \begin{bmatrix} P_0^T \\ P_1^T \end{bmatrix} - \begin{bmatrix} V_0^1 \\ V_1^1 \end{bmatrix} U_0^{0T} \right\|_F^2.$$

Remark 2. *Observe that*

$$\begin{bmatrix} U_0^1 & U_0^0 & 0 \\ U_1^1 & 0 & U_0^0 \end{bmatrix} \in \mathbb{C}^{2m\times(r+a)} \text{ and } \begin{bmatrix} V_0^1 & V_0^0 & 0 \\ V_1^1 & 0 & V_0^0 \end{bmatrix} \in \mathbb{C}^{2n\times(2r-a)}. \tag{3.8}$$

So, both matrices have more rows than columns because $0 \le a \le r < \min\{m,n\}$. *Therefore, generically, the least squares problems* (3.6) *and* (3.7) *have a unique solution. In the non-generic case that the columns of some of the matrices in* (3.8) *are linearly dependent, the corresponding least squares problem has infinitely many solutions. Among these infinitely many solutions we take the one given by Matlab's backslash command, which is not the one provided by the Moore-Penrose inverse, in general. We emphasize that the convergence results in Section 5 are valid for any of the infinitely many possible solutions under the natural assumption that the sequence of solutions of the alternating least squares problems remains bounded.*

4. Approximation with low-rank matrix polynomials

Similarly to Section 3 for matrix pencils, the main goal of this section is to formulate the minimization problem for matrix polynomials of arbitrary grade in terms of a finite number of minimization sub-problems over certain sets of matrix polynomials that can be explicitly described in factorized form and to solve each of these subproblems using the alternating least squares method. The main theoretical ingredients are now Definition 2 and Theorem 2, which are the counterparts of Definition 1 and Theorem 1 used in the case of pencils.

4.1. Preliminary information about matrix polynomials

Recall that a matrix polynomial $P(\lambda) = \sum_{i=0}^{d} \lambda^i P_i$ has a grade d, i.e., a degree less than or equal to d. If $P_d \neq 0$, then $P(\lambda)$ is a matrix polynomial with degree d. In addition to the notation introduced at the beginning of Subsection 3.1, $\mathbb{C}[\lambda]_d^{m\times n}$ denotes the set of $m \times n$ matrix polynomials with grade d and $\mathbb{C}[\lambda]_{d,r}^{m\times n}$ denotes the set of $m \times n$ matrix polynomials with grade d and normal rank at most r. Notably, $\mathbb{C}[\lambda]_d^{m\times n} \supseteq \mathbb{C}[\lambda]_{d,r}^{m\times n}$, where equality holds if $r = \min\{m,n\}$. A distance on $\mathbb{C}[\lambda]_d^{m\times n}$ is defined as $\rho(P,Q) = \|P(\lambda) - Q(\lambda)\|_F$, making $\mathbb{C}[\lambda]_d^{m\times n}$ a metric space and allowing us to define the closure of any subset $\mathcal{S} \subseteq \mathbb{C}[\lambda]_d^{m\times n}$, which is denoted as $\overline{\mathcal{S}}$. Given a matrix polynomial $P(\lambda)$, its ith row is represented by $P_{i*}(\lambda)$, or simply by P_{i*}, and its jth column by $P_{*j}(\lambda)$, or simply by P_{*j}. The following subsets of $\mathbb{C}[\lambda]_d^{m\times n}$ are fundamental in this work.

Definition 2. [10, Definition 4.12] *Let* m, n, r *and* d *be integers such that* $m, n \ge 2$, $d \ge 1$ *and* $0 < r < \min\{m,n\}$, *and* a *be an integer such that* $0 \le a \le rd$. *Let us define* $d_R := \lfloor a/r \rfloor$,

$t_R := a \bmod r$ *and the following subset of polynomial matrices*

$$C^{m\times n}_{d,r,a} := \left\{ L(\lambda)R(\lambda) \; : \; \begin{array}{l} L(\lambda)\in\mathbb{C}[\lambda]^{m\times r},\ R(\lambda)\in\mathbb{C}[\lambda]^{r\times n}, \\ \deg(R_{i*}) \le d_R+1, \quad \textit{for } i=1,\dots,t_R, \\ \deg(R_{i*}) \le d_R, \quad \textit{for } i=t_R+1,\dots,r, \\ \deg(L_{*i}) \le d-d_R-1, \quad \textit{for } i=1,\dots,t_R, \\ \deg(L_{*i}) \le d-d_R, \quad \textit{for } i=t_R+1,\dots,r \end{array} \right\} \subset \mathbb{C}[\lambda]^{m\times n}_{d,r}.$$

Note that for $d=1$, we have that $C^{m\times n}_{1,r,a} = C^{m\times n}_{r,a}$, with $C^{m\times n}_{r,a}$ the set of pencils in Definition 1.

The following theorem portrays $\mathbb{C}[\lambda]^{m\times n}_{d,r}$, for $0<r<\min\{m,n\}$ and $d\ge 1$, as the union of the closures of the $rd+1$ sets of matrix polynomials from Definition 2.

Theorem 2. [10, Theorem 4.13-(iii)] *Let $C^{m\times n}_{d,r,a}$ be the set of matrix polynomials introduced in Definition 2. Then*

$$\mathbb{C}[\lambda]^{m\times n}_{d,r} = \bigcup_{0\le a\le rd} \overline{C^{m\times n}_{d,r,a}}.$$

Observe that Theorem 1 is stronger than the result obtained from making $d=1$ in Theorem 2, because Theorem 1 does not involve closures in the sets $C^{m\times n}_{r,a}$. A detailed discussion about this issue can be found in [10, Remark 4.14-Example 4.15]. Once again we refer the reader to [8,10] for more insight on the description of $\mathbb{C}[\lambda]^{m\times n}_{d,r}$ in Theorem 2.

4.2. Formulation of the minimization problem for matrix polynomials

Let $P(\lambda)=\sum_{i=0}^{d}\lambda^i P_i \in \mathbb{C}[\lambda]^{m\times n}_{d}$. Our goal is to find a matrix polynomial in $\mathbb{C}[\lambda]^{m\times n}_{d,r}$ that is nearest to $P(\lambda)$ and the distance to it, i.e.,

$$\begin{aligned} \operatorname{dist}_{\mathbb{C}[\lambda]^{m\times n}_{d,r}}\left(\sum_{i=0}^{d}\lambda^i P_i\right) &= \min_{\sum_{i=0}^{d}\lambda^i S_i\in\mathbb{C}[\lambda]^{m\times n}_{d,r}} \left\|\sum_{i=0}^{d}\lambda^i (P_i-S_i)\right\|_F, \\ \sum_{i=0}^{d}\lambda^i L_i &\in \operatorname*{arg\,min}_{\sum_{i=0}^{d}\lambda^i S_i\in\mathbb{C}[\lambda]^{m\times n}_{d,r}} \left\|\sum_{i=0}^{d}\lambda^i (P_i-S_i)\right\|_F. \end{aligned} \tag{4.1}$$

As in the case of pencils, we will not repeat the transformations in the expression for the minimizer in (4.1) but we emphasize that the method we develop gives us a minimizer $\sum_{i=0}^{d}\lambda^i L_i$ and thus also the distance $\operatorname{dist}_{\mathbb{C}[\lambda]^{m\times n}_{d,r}}\left(\sum_{i=0}^{d}\lambda^i P_i\right)$.

Similarly as we did for matrix pencils in Section 3.2, but now using Theorem 2, we substitute the minimization problem (4.1) with $rd+1$ minimization problems as follows

$$\begin{aligned}\operatorname{dist}_{\mathbb{C}[\lambda]^{m\times n}_{d,r}}\left(\sum_{i=0}^{d}\lambda^i P_i\right) &= \min_{\sum_{i=0}^{d}\lambda^i S_i\in\left(\bigcup_{0\le a\le rd}\overline{C^{m\times n}_{d,r,a}}\right)}\left\|\sum_{i=0}^{d}\lambda^i(P_i-S_i)\right\|_F\\ &= \min_{0\le a\le rd}\left(\min_{\sum_{i=0}^{d}\lambda^i S_i\in C^{m\times n}_{d,r,a}}\left\|\sum_{i=0}^{d}\lambda^i(P_i-S_i)\right\|_F\right)\\ &= \min_{0\le a\le rd}\operatorname{dist}_{C^{m\times n}_{d,r,a}}\left(\sum_{i=0}^{d}\lambda^i P_i\right).\end{aligned} \tag{4.2}$$

Now we consider each of the $rd+1$ minimization problems derived above:

$$\operatorname{dist}_{C^{m\times n}_{d,r,a}}\left(\sum_{i=0}^{d}\lambda^i P_i\right) = \min_{\sum_{i=0}^{d}\lambda^i S_i\in C^{m\times n}_{d,r,a}}\left\|\begin{bmatrix}P_0\\P_1\\\vdots\\P_d\end{bmatrix}-\begin{bmatrix}S_0\\S_1\\\vdots\\S_d\end{bmatrix}\right\|_F. \tag{4.3}$$

The remainder of this subsection is devoted to reformulating (4.3) in a form that explicitly exploits the structure of the polynomials in $C^{m\times n}_{d,r,a}$; see Definition 2. For this, we introduce some additional definitions and notation. Since we consider $rd+1$ cases, one for each value of a, our goal is to develop a unified approach that covers all of them. Example 1 is included to illustrate these definitions in a simple setting.

For given integers d, m, n, r, a, with $d\ge 1, m, n\ge 2, 0<r<\min\{m,n\}, 0\le a\le rd$, we define two partitioned matrices U_a and V_a as follows

$$U_a^T = [U_0^0 \mid U_0^1\ U_1^1 \mid U_0^2\ U_1^2\ U_2^2 \mid \cdots \mid U_0^{d-1}\cdots U_{d-1}^{d-1} \mid U_0^d\cdots U_d^d], \quad \text{and} \tag{4.4}$$

$$V_a^T = [V_0^0 \mid V_0^1\ V_1^1 \mid V_0^2\ V_1^2\ V_2^2 \mid \cdots \mid V_0^{d-1}\cdots V_{d-1}^{d-1} \mid V_0^d\cdots V_d^d], \tag{4.5}$$

where $U_i^{d-k}\in\mathbb{C}^{m\times n_{d-k}}$, $V_j^k\in\mathbb{C}^{n\times n_{d-k}}$, $n_{d-k}:=\max\{0, r-|a-kr|\}$, $k=0,1,\ldots,d$, $i=0,1,\ldots,d-k$, and $j=0,1,\ldots,k$. Note that n_{d-k} is the number of columns in each of the matrices U_i^{d-k} and V_j^k and that if $n_{d-k}=0$, then the corresponding matrices are absent from U_a and V_a. Next, we define the matrix

$$\mathbf{M}(U_a) := \left[M_0(U_a)|M_1(U_a)|M_2(U_a)|\cdots|M_{d-1}(U_a)|M_d(U_a)\right]$$

$$:= \left[\begin{array}{c|cc|ccc|c|c|ccccc}
U_0^d & U_0^{d-1} & 0 & U_0^{d-2} & 0 & 0 & & U_0^1 & 0 & 0 & U_0^0 & & & & \\
U_1^d & U_1^{d-1} & U_0^{d-1} & \vdots & U_0^{d-2} & 0 & & U_1^1 & \vdots & 0 & & U_0^0 & & & \\
\vdots & \vdots & U_1^{d-1} & U_{d-3}^{d-2} & \vdots & U_0^{d-2} & \cdots & 0 & \ddots\ 0 & \vdots & & & U_0^0 & & \\
\vdots & U_{d-2}^{d-1} & \vdots & U_{d-2}^{d-2} & U_{d-3}^{d-2} & \vdots & & 0 & U_0^1 & 0 & & & & \ddots & \\
U_{d-1}^d & U_{d-1}^{d-1} & U_{d-2}^{d-1} & 0 & U_{d-2}^{d-2} & U_{d-3}^{d-2} & & \vdots & U_1^1 & U_0^1 & & & & & U_0^0 \\
U_d^d & 0 & U_{d-1}^{d-1} & 0 & 0 & U_{d-2}^{d-2} & & 0 & 0 & U_1^1 & & & & & U_0^0
\end{array}\right]. \tag{4.6}$$

Observe that $M_k(U_a) \in \mathbb{C}^{(d+1)m\times(k+1)n_{d-k}}$ has the same number of columns as $[V_0^k \cdots V_k^k] \in \mathbb{C}^{n\times(k+1)n_{d-k}}$ for $k = 0, 1, \ldots, d$. Analogously, we define

$$\mathbf{M}(V_a) := \left[M_0(V_a)|M_1(V_a)|M_2(V_a)|\cdots|M_{d-1}(V_a)|M_d(V_a)\right], \tag{4.7}$$

by replacing in (4.6) each matrix U_i^j by V_i^j. In this case, the sizes of the submatrices are as follows: $M_k(V_a) \in \mathbb{C}^{(d+1)n\times(k+1)n_k}$ and $[U_0^k \cdots U_k^k] \in \mathbb{C}^{m\times(k+1)n_k}$ for $k = 0, 1, \ldots, d$.

Lemma 1 shows that $n_{d-k} = \max\{0, r - |a - kr|\}$ is strictly greater than zero for at most two values of k. Thus, among $U_0^0, U_0^1, U_0^2, \ldots, U_0^d$ at most two U_0^{d-k} are present, i.e., they have a positive number of columns. Clearly, the absence of U_0^{d-k} implies the absence of U_i^{d-k}, for $i = 0, \ldots, d-k$. Similarly, among $V_0^0, V_0^1, V_0^2, \ldots, V_0^d$ in (4.5) at most two V_0^k are present.

Lemma 1. *Let d, r, and a be given integers, with d and r positive and such that $0 \le a \le rd$. Consider $n_{d-k} = \max\{0, r - |a - kr|\}$ for $k = 0, 1, \ldots, d$. Then, the following two statements hold:*

(a) $n_{d-k} > 0$ *for at most two values of k, namely for $k \in (a/r - 1, a/r + 1) \cap \mathbb{Z}$;*

(b) $\sum_{k=0}^{d} n_{d-k} = r$.

Proof. The solution of $r - |a - kr| > 0$ shows the first item. For the second item, note that $k \in (a/r - 1, a/r + 1) \cap \mathbb{Z}$ means $k = a/r$ if $a/r = \lfloor a/r \rfloor$, and k is equal to $\lfloor a/r \rfloor$ and to $\lfloor a/r \rfloor + 1$ if $a/r \neq \lfloor a/r \rfloor$. Adding n_{d-k} for these values of k proves the second item. ∎

As a corollary of Lemma 1, its proof and the number of columns of the submatrices of (4.4)-(4.7), we obtain the following result about the matrices $\mathbf{M}(U_a)$, V_a, $\mathbf{M}(V_a)$, and U_a. Its simple proof is omitted.

Lemma 2. *Let m, n, r, d and a be integers such that $m, n \ge 2$, $d \ge 1$, $0 < r < \min\{m, n\}$ and $0 \le a \le rd$. Let $d_R = \lfloor a/r \rfloor$, $t_R = a \bmod r$, $n_{d-k} = \max\{0, r - |a - kr|\}$, $\mathbf{M}(U_a)$ and V_a be the matrices in (4.6) and (4.5), and $\mathbf{M}(V_a)$ and U_a be the matrices in (4.7) and (4.4).*

(a) *If $d_R = a/r$, then $n_{d-d_R} = r$,*

$$\begin{aligned}
&\mathbf{M}(U_a) = M_{d_R}(U_a) \in \mathbb{C}^{(d+1)m\times(r+a)},\\
&V_a^T = [V_0^{d_R}\ V_1^{d_R} \cdots V_{d_R}^{d_R}] \in \mathbb{C}^{n\times(r+a)},\\
&\mathbf{M}(V_a) = M_{d-d_R}(V_a) \in \mathbb{C}^{(d+1)n\times((d+1)r-a)}, \quad \textit{and}\\
&U_a^T = [U_0^{d-d_R}\ U_1^{d-d_R} \cdots U_{d-d_R}^{d-d_R}] \in \mathbb{C}^{m\times((d+1)r-a)}.
\end{aligned}$$

(b) *If* $d_R \neq a/r$, *then* $n_{d-d_R} = r - t_R$, $n_{d-d_R-1} = t_R$,

$$\mathbf{M}(U_a) = [M_{d_R}(U_a)|M_{d_R+1}(U_a)] \in \mathbb{C}^{(d+1)m\times(r+a)},$$
$$V_a^T = [V_0^{d_R}\ V_1^{d_R}\cdots V_{d_R}^{d_R}|\ V_0^{d_R+1}\ V_1^{d_R+1}\cdots V_{d_R+1}^{d_R+1}] \in \mathbb{C}^{n\times(r+a)},$$
$$\mathbf{M}(V_a) = [M_{d-d_R-1}(V_a)|M_{d-d_R}(V_a)] \in \mathbb{C}^{(d+1)n\times((d+1)r-a)}, \quad \textit{and}$$
$$U_a^T = [U_0^{d-d_R-1}\ U_1^{d-d_R-1}\cdots U_{d-d_R-1}^{d-d_R-1}|\ U_0^{d-d_R}\ U_1^{d-d_R}\cdots U_{d-d_R}^{d-d_R}] \in \mathbb{C}^{m\times((d+1)r-a)}.$$

Example 1. *We illustrate the definitions of* $\mathbf{M}(U_a)$ *and* V_a *in the case* $d = 2$ *and* $r = 2$ *for all the possible values of* a. *In this situation*

$$U_a^T = [U_0^0 \mid U_0^1\ U_1^1 \mid U_0^2\ U_1^2\ U_2^2], \ \ 0 \le a \le 4, \tag{4.8}$$

before removing the submatrices with zero columns. The corresponding full block matrix reads

$$\mathbf{M}(U_a) = \left[M_0(U_a)|M_1(U_a)|M_2(U_a)\right] = \left[\begin{array}{c|cc|ccc} U_0^2 & U_0^1 & 0 & U_0^0 & 0 & 0 \\ U_1^2 & U_1^1 & U_0^1 & 0 & U_0^0 & 0 \\ U_2^2 & 0 & U_1^1 & 0 & 0 & U_0^0 \end{array}\right]$$

and

$$V_a^T = [V_0^0 \mid V_0^1\ V_1^1 \mid V_0^2\ V_1^2\ V_2^2].$$

For each a, *the matrices* $\mathbf{M}(U_a)$ *and* V_a^T *and their sizes are obtained from Lemma 2. The results are summarized in Table 1. The matrices* $\mathbf{M}(V_a)$ *and* U_a^T *can be obtained analogously for* $a = 0, 1, 2, 3, 4$. *For brevity, we just show the case* $a = 3$*:*

$$\mathbf{M}(V_3) = \left[\begin{array}{c|cc} V_0^2 & V_0^1 & 0 \\ V_1^2 & V_1^1 & V_0^1 \\ V_2^2 & 0 & V_1^1 \end{array}\right] \in \mathbb{C}^{3n\times 3}, \quad U_3^T = [\ U_0^0 \mid U_0^1\ U_1^1\ ] \in \mathbb{C}^{m\times 3}.$$

■

Remark 3. *Lemma 2 shows that, for each* a, $U_a \mapsto \mathbf{M}(U_a)$ *is a linear function of* U_a *with domain the vector space* $\mathbb{C}^{((d+1)r-a)\times m}$ *and codomain* $\mathbb{C}^{(d+1)m\times(r+a)}$. *Analogously,* $V_a \mapsto \mathbf{M}(V_a)$ *is a linear function of* V_a *with domain the vector space* $\mathbb{C}^{(r+a)\times n}$ *and codomain* $\mathbb{C}^{(d+1)n\times((d+1)r-a)}$. *This fact will be used in the sequel and, in particular, in Section 5.*

To connect $\mathbf{M}(U_a)$ and V_a with the set $\mathcal{C}_{d,r,a}^{m\times n}$ in the minimization problem (4.3), we define the set

$$\widetilde{\mathcal{C}}_{d,r,a}^{m\times n} := \left\{ \left[\begin{array}{c} S_0 \\ S_1 \\ \vdots \\ S_d \end{array}\right] : \sum_{i=0}^{d} \lambda^i S_i \in \mathcal{C}_{d,r,a}^{m\times n} \right\} \subset \mathbb{C}^{(d+1)m\times n}. \tag{4.9}$$

a	$\mathbf{M}(U_a)$	size($\mathbf{M}(U_a)$)	V_a^T	size(V_a^T)
0	$\begin{bmatrix} U_0^2 \\ U_1^2 \\ U_2^2 \end{bmatrix}$	$3m \times 2$	$[\, V_0^0 \,]$	$n \times 2$
1	$\left[\begin{array}{c\|cc} U_0^2 & U_0^1 & 0 \\ U_1^2 & U_1^1 & U_0^1 \\ U_2^2 & 0 & U_1^1 \end{array}\right]$	$3m \times 3$	$[\, V_0^0 \mid V_0^1 \; V_1^1 \,]$	$n \times 3$
2	$\begin{bmatrix} U_0^1 & 0 \\ U_1^1 & U_0^1 \\ 0 & U_1^1 \end{bmatrix}$	$3m \times 4$	$[\, V_0^1 \; V_1^1 \,]$	$n \times 4$
3	$\left[\begin{array}{cc\|ccc} U_0^1 & 0 & U_0^0 & 0 & 0 \\ U_1^1 & U_0^1 & 0 & U_0^0 & 0 \\ 0 & U_1^1 & 0 & 0 & U_0^0 \end{array}\right]$	$3m \times 5$	$[\, V_0^1 V_1^1 \mid V_0^2 V_1^2 V_2^2 \,]$	$n \times 5$
4	$\begin{bmatrix} U_0^0 & 0 & 0 \\ 0 & U_0^0 & 0 \\ 0 & 0 & U_0^0 \end{bmatrix}$	$3m \times 6$	$[\, V_0^2 \; V_1^2 \; V_2^2 \,]$	$n \times 6$

Table 1. Explicit shapes and sizes of $\mathbf{M}(U_a)$ and V_a^T for all possible a, in the case of $d = 2$ and $r = 2$.

Observe that $\widetilde{C}_{d,r,a}^{m\times n}$ is isometric to $C_{d,r,a}^{m\times n}$ via the Frobenius norm. Lemma 3 establishes the connection and relates the matrices $\mathbf{M}(U_a) \cdot V_a \in \mathbb{C}^{(d+1)m\times n}$ and $\mathbf{M}(V_a) \cdot U_a \in \mathbb{C}^{(d+1)n\times m}$.

Lemma 3. *Let m, n, r and d be integers such that $m, n \geq 2$, $d \geq 1$ and $0 < r < \min\{m, n\}$, and a be an integer such that $0 \leq a \leq rd$. Let $\mathbf{M}(U_a)$, V_a, $\mathbf{M}(V_a)$, U_a be matrices as in Lemma 2, and $\widetilde{C}_{d,r,a}^{m\times n}$ be the set in (4.9). Then*

$$\widetilde{C}_{d,r,a}^{m\times n} = \left\{ \mathbf{M}(U_a) \cdot V_a \;:\; U_a \in \mathbb{C}^{((d+1)r-a)\times m},\, V_a \in \mathbb{C}^{(r+a)\times n} \right\}. \tag{4.10}$$

Moreover,

$$\begin{bmatrix} S_0 \\ S_1 \\ \vdots \\ S_d \end{bmatrix} = \mathbf{M}(U_a) \cdot V_a \;\; \textit{if and only if} \;\; \begin{bmatrix} S_0^T \\ S_1^T \\ \vdots \\ S_d^T \end{bmatrix} = \mathbf{M}(V_a) \cdot U_a. \tag{4.11}$$

Proof. Fix a and let $[S_0^T, S_1^T, \ldots, S_d^T]^T \in \widetilde{C}_{d,r,a}^{m\times n}$, which is equivalent to $\sum_{i=0}^{d} \lambda^i S_i \in C_{d,r,a}^{m\times n}$ by the definition of $\widetilde{C}_{d,r,a}^{m\times n}$. Let $d_R = \lfloor a/r \rfloor$ and assume $d_R \neq a/r$; the case $d_R = a/r$ is

analogous but simpler, and is omitted. By Definition 2, $\sum_{i=0}^{d} \lambda^i S_i \in C_{d,r,a}^{m\times n}$ is equivalent to

$$\sum_{i=0}^{d} \lambda^i S_i = \left(\sum_{i=0}^{d-d_R} \lambda^i U_i^{d-d_R}\right)\left(\sum_{i=0}^{d_R} \lambda^i {V_i^{d_R}}^T\right) + \left(\sum_{i=0}^{d-d_R-1} \lambda^i U_i^{d-d_R-1}\right)\left(\sum_{i=0}^{d_R+1} \lambda^i {V_i^{d_R+1}}^T\right), \tag{4.12}$$

with $\left(\sum_{i=0}^{d-d_R} \lambda^i U_i^{d-d_R}\right) \in \mathbb{C}[\lambda]^{m\times(r-t_R)}$, $\left(\sum_{i=0}^{d-d_R-1} \lambda^i U_i^{d-d_R-1}\right) \in \mathbb{C}[\lambda]^{m\times t_R}$, $\left(\sum_{i=0}^{d_R} \lambda^i {V_i^{d_R}}^T\right)$ $\in \mathbb{C}[\lambda]^{(r-t_R)\times n}$ and $\left(\sum_{i=0}^{d_R+1} \lambda^i {V_i^{d_R+1}}^T\right) \in \mathbb{C}[\lambda]^{t_R\times n}$. The equality (4.12) can be rewritten using the block-Toeplitz matrices $M_{d_R}(U_a)$ and $M_{d_R+1}(U_a)$ built as in (4.6) from the matrix polynomials $\sum_{i=0}^{d-d_R} \lambda^i U_i^{d-d_R}$ and $\sum_{i=0}^{d-d_R-1} \lambda^i U_i^{d-d_R-1}$, respectively, as follows

$$\begin{bmatrix} S_0 \\ S_1 \\ \vdots \\ S_d \end{bmatrix} = M_{d_R}(U_a) \begin{bmatrix} {V_0^{d_R}}^T \\ {V_1^{d_R}}^T \\ \vdots \\ {V_{d_R}^{d_R}}^T \end{bmatrix} + M_{d_R+1}(U_a) \begin{bmatrix} {V_0^{d_R+1}}^T \\ {V_1^{d_R+1}}^T \\ \vdots \\ {V_{d_R+1}^{d_R+1}}^T \end{bmatrix} = \mathbf{M}(U_a)\cdot V_a,$$

where the last equality follows from Lemma 2-(b). This yields (4.10). Taking transposes in (4.12) and proceeding in a similar way yields (4.11). ∎

Lemma 3 together with Remark 3 allow us to formulate the minimization problem (4.3) as the following *unconstrained minimization problem*

$$\operatorname{dist}_{C_{d,r,a}^{m\times n}}\left(\sum_{i=0}^{d} \lambda^i P_i\right) = \min_{\substack{U_a\in\mathbb{C}^{((d+1)r-a)\times m},\\ V_a\in\mathbb{C}^{(r+a)\times n}}} \left\| \begin{bmatrix} P_0 \\ P_1 \\ \vdots \\ P_d \end{bmatrix} - \mathbf{M}(U_a)\cdot V_a \right\|_F. \tag{4.13}$$

It is important to note that the cost function of this minimization problem admits another alternative expression, because (4.11) implies

$$\left\| \begin{bmatrix} P_0 \\ P_1 \\ \vdots \\ P_d \end{bmatrix} - \mathbf{M}(U_a)\cdot V_a \right\|_F = \left\| \begin{bmatrix} P_0^T \\ P_1^T \\ \vdots \\ P_d^T \end{bmatrix} - \mathbf{M}(V_a)\cdot U_a \right\|_F. \tag{4.14}$$

These two expressions play a key role in the solution of the problem via the method in Section 4.3.

4.3. Solution of the minimization problem over each $C_{d,r,a}^{m\times n}$

Similarly as for matrix pencils, we solve (4.13) by using an alternating least squares method between the $U-$ and $V-$ variables. To be exact, we (randomly) choose a matrix $U_a \in$

$\mathbb{C}^{((d+1)r-a)\times m}$, construct with its (adequate) submatrices $\mathbf{M}(U_a)$, and solve the standard least squares problem

$$\min_{V_a\in\mathbb{C}^{(r+a)\times n}} \left\| \begin{bmatrix} P_0 \\ \vdots \\ P_d \end{bmatrix} - \mathbf{M}(U_a)\cdot V_a \right\|_F , \tag{4.15}$$

using, in practice, Matlab's backslash command. From the computed V_a, we construct $\mathbf{M}(V_a)$, and, taking into account (4.14), solve the standard least squares problem

$$\min_{U_a\in\mathbb{C}^{((d+1)r-a)\times m}} \left\| \begin{bmatrix} P_0^T \\ \vdots \\ P_d^T \end{bmatrix} - \mathbf{M}(V_a)\cdot U_a \right\|_F , \tag{4.16}$$

using, in practice, again Matlab's backslash command. Then we again repeat the first step with the computed U_a and so on until convergence.

Remark 4. *Observe that* $\mathbf{M}(U_a) \in \mathbb{C}^{(d+1)m\times(r+a)}$ *and* $\mathbf{M}(V_a) \in \mathbb{C}^{(d+1)n\times((d+1)r-a)}$ *are both matrices with more rows than columns because* $0 \le a \le rd$ *and* $0 < r < \min\{m,n\}$. *Therefore, generically, the least squares problems* (4.15) *and* (4.16) *have a unique solution, as in the case of pencils. In the non-generic cases in which the columns of* $\mathbf{M}(U_a)$ *or* $\mathbf{M}(V_a)$ *are linearly dependent, the corresponding least squares problem has infinitely many solutions and we take the one given by Matlab's backslash command. The convergence analysis in Section 5 is valid for any other selection of a solution under the natural assumption that the sequence of solutions of the alternating least squares problems remains bounded.*

Remark 5. (On the real minimization problem). *If the polynomial* $P(\lambda) = \sum_{i=0}^{d} \lambda^i P_i \in \mathbb{R}[\lambda]_d^{m\times n}$ *is real and the initial iterate* $U_a \in \mathbb{R}^{((d+1)r-a)\times m}$ *is chosen to be real, then all the iterates computed by the alternating least squares method described above remain real. In fact, all the results presented in Section 4 are valid if* $\mathbb{C}$ *is replaced by* $\mathbb{R}$. *Therefore, the presented method solves the real version of the problem* (4.1).

5. Convergence analysis

In this section, we analyze the convergence of the algorithm described in Section 4.3. We will work with complex matrix polynomials, but we emphasize that the analysis remains valid in the real case (recall Remark 5).

Let us start by introducing some notation. For brevity, we define $p_a = (d+1)r - a$ and $q_a = r + a$. According to (4.13), we take the matrices $U_a \in \mathbb{C}^{p_a\times m}$ and $V_a \in \mathbb{C}^{q_a\times n}$ in Lemma 2 as the variables of the minimization problem. Recall that according to Remark 3, for $\alpha, \beta \in \mathbb{C}$, $U_a, E_a \in \mathbb{C}^{p_a\times m}$, and $V_a, F_a \in \mathbb{C}^{q_a\times n}$, the equalities

$$\mathbf{M}(\alpha\, U_a + \beta\, E_a) = \alpha\, \mathbf{M}(U_a) + \beta\, \mathbf{M}(E_a) \quad \text{and} \quad \mathbf{M}(\alpha\, V_a + \beta F_a) = \alpha\, \mathbf{M}(V_a) + \beta\, \mathbf{M}(F_a) \tag{5.1}$$

hold for the matrices $\mathbf{M}(U_a)$ and $\mathbf{M}(V_a)$ in Lemma 2. Taking into account (4.13), for each value of a, $a = 0, 1, \ldots, rd$, we define the function $\Phi : \mathbb{C}^{p_a \times m} \times \mathbb{C}^{q_a \times n} \to \mathbb{R}$ as

$$\Phi(U_a, V_a) := \left\| \begin{bmatrix} P_0 \\ \vdots \\ P_d \end{bmatrix} - \mathbf{M}(U_a) \cdot V_a \right\|_F^2, \tag{5.2}$$

and consider the optimization problem

$$\min_{(U_a, V_a) \in \mathbb{C}^{p_a \times m} \times \mathbb{C}^{q_a \times n}} \Phi(U_a, V_a). \tag{5.3}$$

Observe that we define Φ as the square of the cost function in (4.13), because we wish to deal with differentiable functions. This does not change the minimizers nor the sequence of matrices U_a and V_a computed by the method in Section 4.3.

Although we deal with complex matrices, we consider Φ as a function of real variables. More precisely, as a function of the real and imaginary parts of the entries of U_a and V_a. In this setting, Φ is a polynomial of degree four over $\mathbb{R}^{2(p_a m + q_a n)}$ ($\cong \mathbb{C}^{p_a \times m} \times \mathbb{C}^{q_a \times n}$) and, so, it is infinitely differentiable on the whole space. To be consistent with this view, $\mathbb{C}^{p_a \times m} \times \mathbb{C}^{q_a \times n}$ is considered a real vector space endowed with the real inner product

$$\langle (A_1, B_1), (A_2, B_2) \rangle = \Re(\mathrm{trace}(A_1^* A_2)) + \Re(\mathrm{trace}(B_1^* B_2)), \tag{5.4}$$

where $\Re(z)$ stands for the real part of $z \in \mathbb{C}$.

With the notation and conventions above, the method introduced in Section 4.3 can be concisely described with the following pseudocode.

Algorithm 1 Alternating least squares for approximating over $\mathcal{C}_{d,r,a}^{m \times n}$

1: **Input:** polynomial P, tolerance tol, maximal number of iterations $maxiter$
2: **Initialization:** $U_a^{(0)}$
3: **Repeat** recursively generate $(U_a^{(k)}, V_a^{(k)})$ by
4: $\quad V_a^{(k)} = \underset{V_a \in \mathbb{C}^{q_a \times n}}{\arg\min}\ \Phi(U_a^{(k-1)}, V_a)$
5: $\quad U_a^{(k)} = \underset{U_a \in \mathbb{C}^{p_a \times m}}{\arg\min}\ \Phi(U_a, V_a^{(k)})$
6: **Until** $\left| \sqrt{\Phi(U_a^{(k-1)}, V_a^{(k-1)})} - \sqrt{\Phi(U_a^{(k)}, V_a^{(k)})} \right| \leq tol * \|P\|_F$ or $maxiter$ is reached

As commented in Section 4.3, the problems in lines 4 and 5 of Algorithm 1 are standard least squares problems and, therefore, have global minimizers for all k. Such minimizers are unique if and only if the matrix $\mathbf{M}(U_a^{(k-1)})$ has full column rank, for the problem in line 4, and if and only if the matrix $\mathbf{M}(V_a^{(k)})$ has full column rank, for the problem in line 5.

Otherwise, there are infinitely many minimizers. Though the generic situation is that the minimizers are unique, as commented in Remark 4, we do not know if it holds for all k, or, more precisely, for which initializations this holds for all k. In this setting, we remark that the results in this section are valid even in the case the minimizers are not unique and for any possible choice of them.

Algorithm 1 is a particular instance of the well-known block coordinate descent methods, whose convergence has been analyzed in depth in the literature. See, for instance, [20, 25]. However, one of the key assumptions in these analyses does not hold for the function Φ in (5.2), a question that we discuss in Section 5.2. Therefore, we prove directly two results on the convergence of Algorithm 1. First, we prove in Theorem 3 that the sequence $\left\{\Phi(U_a^{(k)}, V_a^{(k)})\right\}$ is convergent without the need of any assumption. Second, under the assumption that the sequence $\left\{(U_a^{(k)}, V_a^{(k)})\right\}$ is bounded, we prove in Theorem 4 that the gradient of Φ converges to zero. Observe that these results do not imply that $\left\{(U_a^{(k)}, V_a^{(k)})\right\}$ is convergent. The assumption about the boundedness of $\left\{(U_a^{(k)}, V_a^{(k)})\right\}$ has been imposed in general convergence analyses [20, 25] and has been always satisfied in our numerical tests.

Theorem 3. *Let $\left\{(U_a^{(k)}, V_a^{(k)})\right\}$ be the sequence of iterates generated by Algorithm 1 without stopping criterion and $\left\{\Phi(U_a^{(k)}, V_a^{(k)})\right\}$ be the sequence of real numbers obtained by evaluating the function Φ in (5.2) at these iterates. Then:*

(a) $\Phi(U_a^{(k+1)}, V_a^{(k+1)}) \leq \Phi(U_a^{(k)}, V_a^{(k+1)}) \leq \Phi(U_a^{(k)}, V_a^{(k)})$, *for* $k = 1, 2, \ldots$.

(b) $\left\{\Phi(U_a^{(k)}, V_a^{(k)})\right\}$ *is convergent.*

Proof. Part (a) follows directly from the definition of Algorithm 1. Part (a) proves that $\left\{\Phi(U_a^{(k)}, V_a^{(k)})\right\}$ is a monotone non-increasing sequence. Moreover, it is bounded from below by zero, because $\Phi(U^{(k)}, V^{(k)}) \geq 0$, for all k, and, therefore, is convergent. ∎

5.1. Convergence to zero of the gradient

Since the function $\Phi : \mathbb{C}^{p_a \times m} \times \mathbb{C}^{q_a \times n} \to \mathbb{R}$ is differentiable everywhere, its differential at (U_a, V_a) is a linear map, denoted by $\mathrm{D}\Phi(U_a, V_a)$, with domain $\mathbb{C}^{p_a \times m} \times \mathbb{C}^{q_a \times n}$ and codomain $\mathbb{R}$. Using the inner product in (5.4), the gradient of Φ at (U_a, V_a), denoted by $\operatorname{grad} \Phi(U_a, V_a)$, is the element of $\mathbb{C}^{p_a \times m} \times \mathbb{C}^{q_a \times n}$ that satisfies

$$\langle \operatorname{grad} \Phi(U_a, V_a)\,,\, (E_a, F_a) \rangle = \mathrm{D}\Phi(U_a, V_a)[(E_a, F_a)], \tag{5.5}$$

for all $(E_a, F_a) \in \mathbb{C}^{p_a \times m} \times \mathbb{C}^{q_a \times n}$. See [3, p. 23] for more details on these concepts. From (5.5), it is possible to obtain the explicit expression of $\operatorname{grad} \Phi(U_a, V_a)$.

Lemma 4. *Let* $\Phi : \mathbb{C}^{p_a \times m} \times \mathbb{C}^{q_a \times n} \to \mathbb{R}$ *be the function in* (5.2). *Then* $\operatorname{grad} \Phi(U_a, V_a) = (\operatorname{grad}_U \Phi(U_a, V_a), \operatorname{grad}_V \Phi(U_a, V_a))$, *with*

$$\operatorname{grad}_U \Phi(U_a, V_a) = 2\,\mathbf{M}(V_a)^* \left(\mathbf{M}(V_a) \cdot U_a - \begin{bmatrix} P_0^T \\ \vdots \\ P_d^T \end{bmatrix} \right),$$

$$\operatorname{grad}_V \Phi(U_a, V_a) = 2\,\mathbf{M}(U_a)^* \left(\mathbf{M}(U_a) \cdot V_a - \begin{bmatrix} P_0 \\ \vdots \\ P_d \end{bmatrix} \right).$$

Proof. We only sketch the proof. It follows from a direct computation of the right-hand side of (5.5) as

$$\mathrm{D}\Phi(U_a, V_a)[(E_a, F_a)] = \lim_{t \to 0} \frac{\Phi(U_a + tE_a, V_a + tF_a) - \Phi(U_a, V_a)}{t},$$

taking into account (5.2), that $\|A\|_F^2 = \operatorname{trace}(A^*A)$ for any matrix A, and the definition (5.4). In the process, it is needed to apply the equality

$$\operatorname{trace}\left(\left(\mathbf{M}(U_a) \cdot V_a - \begin{bmatrix} P_0 \\ \vdots \\ P_d \end{bmatrix} \right)^* \mathbf{M}(E_a) \cdot V_a \right) = \operatorname{trace}\left(\left(\mathbf{M}(V_a) \cdot U_a - \begin{bmatrix} P_0^T \\ \vdots \\ P_d^T \end{bmatrix} \right)^* \mathbf{M}(V_a) \cdot E_a \right),$$

which follows from (4.11) and from $\operatorname{trace}(A^*B) = \sum_{i=1}^{p} \sum_{j=1}^{q} \overline{a_{ij}}\, b_{ij}$ for any $A, B \in \mathbb{C}^{p \times q}$. ∎

Lemma 4 proves that $\operatorname{grad}_U \Phi(U_a, V_a)$ (resp. $\operatorname{grad}_V \Phi(U_a, V_a)$) is two times the difference between the left and the right-hand sides of the normal equations corresponding to the least squares problem (4.16) (resp. (4.15)). Since the normal equations are satisfied by any solution of these least squares problems, we immediately get the following lemma.

Lemma 5. *Let* $\left\{ (U_a^{(k)}, V_a^{(k)}) \right\}$ *be the sequence of iterates generated by Algorithm* 1 *without stopping criterion. Then*

$$\operatorname{grad}_U \Phi(U_a^{(k)}, V_a^{(k)}) = 0 \qquad \textit{and} \qquad \operatorname{grad}_V \Phi(U_a^{(k-1)}, V_a^{(k)}) = 0, \qquad \textit{for all } k \geq 1.$$

Observe that Lemma 5 allows us to interpret Algorithm 1 as a process that alternates between setting to zero $\operatorname{grad}_V \Phi$ and $\operatorname{grad}_U \Phi$.

The rest of this section is devoted to proving, under the assumption that $\left\{ (U_a^{(k)}, V_a^{(k)}) \right\}$ is bounded, that $\lim_{k \to \infty} \operatorname{grad}_V \Phi(U_a^{(k)}, V_a^{(k)}) = 0$, which, combined with Lemma 5, implies $\lim_{k \to \infty} \operatorname{grad} \Phi(U_a^{(k)}, V_a^{(k)}) = 0$. For that we need the following lemma.

Lemma 6. *Let $\|\cdot\|$ be the norm derived from the inner product (5.4) and β be a positive number. If the sequence of iterates generated by Algorithm 1 without stopping criterion satisfies $\|(U_a^{(k)}, V_a^{(k)})\| \leq \beta$ for all $k \geq 1$, then*

$$\Phi(U_a^{(k)}, V_a^{(k)}) - \Phi(U_a^{(k)}, V_a^{(k+1)}) \geq \frac{1}{4(d_R+2)\beta^2} \|\mathrm{grad}_V \Phi(U_a^{(k)}, V_a^{(k)})\|_F^2,$$

for all $k \geq 1$, where $d_R = \lfloor a/r \rfloor$.

Proof. Note that $\|(U_a^{(k)}, V_a^{(k)})\| \leq \beta$ implies that $\|U_a^{(k)}\|_F^2 \leq \beta^2$. Then, Lemma 2 and (4.6) imply $\|\mathbf{M}(U_a^{(k)})\|_F^2 \leq (d_R+2)\beta^2$, for $k \geq 1$. Let $\mathbf{M}(U_a^{(k)}) = X\Sigma Y^*$ be an economy-size singular value decomposition of $\mathbf{M}(U_a^{(k)})$, with $X \in \mathbb{C}^{(d+1)m \times s}$, $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_s) \in \mathbb{R}^{s\times s}$, $Y \in \mathbb{C}^{q_a \times s}$, and $\sigma_1 \geq \ldots \geq \sigma_s > 0$ the nonzero singular values of $\mathbf{M}(U_a^{(k)})$. Observe that $s = \mathrm{rank}\,(\mathbf{M}(U_a^{(k)}))$, which can be any value $s \leq (r+a)$. However, if $s = 0$, then $\mathbf{M}(U_a^{(k)}) = 0$ and $\mathrm{grad}_V \Phi(U_a^{(k)}, V_a^{(k)}) = 0$, which, combined with Theorem 3-(a), implies trivially the result. Therefore, we assume in the rest of the proof that $s \geq 1$. Let $\begin{bmatrix} X & X_\perp \end{bmatrix} \in \mathbb{C}^{(d+1)m \times (d+1)m}$ be a unitary matrix obtained by completing the columns of X. If

$$\begin{bmatrix} Z^* \\ Z_\perp^* \end{bmatrix} := \begin{bmatrix} X^* \\ X_\perp^* \end{bmatrix} \begin{bmatrix} P_0 \\ \vdots \\ P_d \end{bmatrix},$$

where $Z^* \in \mathbb{C}^{s\times n}$, then

$$\Phi(U_a^{(k)}, V_a^{(k)}) = \left\| \begin{bmatrix} X^* \\ X_\perp^* \end{bmatrix} \left(\begin{bmatrix} P_0 \\ \vdots \\ P_d \end{bmatrix} - \mathbf{M}(U_a^{(k)}) \cdot V_a^{(k)} \right) \right\|_F^2 = \|Z^* - \Sigma Y^* V_a^{(k)}\|_F^2 + \|Z_\perp^*\|_F^2.$$

Analogously,

$$\Phi(U_a^{(k)}, V_a^{(k+1)}) = \|Z^* - \Sigma Y^* V_a^{(k+1)}\|_F^2 + \|Z_\perp^*\|_F^2 .$$

Moreover, in the equality above, $\|Z^* - \Sigma Y^* V_a^{(k+1)}\|_F^2 = 0$, because $V_a^{(k+1)}$ is the output[1] of line 4 in Algorithm 1 (with k replaced by $k+1$). So,

$$\Phi(U_a^{(k)}, V_a^{(k)}) - \Phi(U_a^{(k)}, V_a^{(k+1)}) = \|Z^* - \Sigma Y^* V_a^{(k)}\|_F^2 .$$

On the other hand, from Lemma 4 and the SVD $\mathbf{M}(U_a^{(k)}) = X\Sigma Y^*$, we get

$$\begin{aligned} \|\mathrm{grad}_V \Phi(U_a^{(k)}, V_a^{(k)})\|_F^2 &= 4\,\|\Sigma(\Sigma Y^* V_a^{(k)} - Z^*)\|_F^2 \\ &\leq 4\,\|\Sigma\|_F^2\,\|\Sigma Y^* V_a^{(k)} - Z^*\|_F^2 \\ &= 4\,\|\mathbf{M}(U_a^{(k)})\|_F^2\,(\Phi(U_a^{(k)}, V_a^{(k)}) - \Phi(U_a^{(k)}, V_a^{(k+1)})), \end{aligned}$$

which combined with $\|\mathbf{M}(U_a^{(k)})\|_F^2 \leq (d_R+2)\beta^2$ completes the proof. ∎

[1] Note that $V_a^{(k+1)} = Y\Sigma^{-1}Z^* + W$, with W a matrix whose colums are in the null space of Y^*, which is equal to the null space of $\mathbf{M}(U_a^{(k)})$.

A direct corollary of Lemmas 5 and 6 and Theorem 3 is that the gradient of the cost function converges to zero. This is stated in Theorem 4.

Theorem 4. *Let $\|\cdot\|$ be the norm derived from the inner product (5.4) and β be a positive number. If the sequence of iterates generated by Algorithm 1 without stopping criterion satisfies $\|(U_a^{(k)}, V_a^{(k)})\| \leq \beta$ for all $k \geq 1$, then*

$$\lim_{k\to\infty} \operatorname{grad} \Phi(U_a^{(k)}, V_a^{(k)}) = 0.$$

Remark 6. (On coordinate-wise minima) *If $\{(U_a^{(k)}, V_a^{(k)})\}$ is bounded, then it has a convergent subsequence $\{(U_a^{(k_j)}, V_a^{(k_j)})\}$, with limit denoted by $\{(\widetilde{U}_a, \widetilde{V}_a)\}$. Since* grad Φ *is a continuous function by Lemma 4, we get from Theorem 4 that* grad $\Phi(\widetilde{U}_a, \widetilde{V}_a) = 0$. *The expressions of* $\operatorname{grad}_U \Phi(\widetilde{U}_a, \widetilde{V}_a)$ *and* $\operatorname{grad}_V \Phi(\widetilde{U}_a, \widetilde{V}_a)$ *imply that $\{(\widetilde{U}_a, \widetilde{V}_a)\}$ satisfies simultaneously the normal equations of the least squares problems (4.15) and (4.16). Thus, $\{(\widetilde{U}_a, \widetilde{V}_a)\}$ is a coordinate-wise minimum of Φ* [1, Definition 14.2]*, i.e., $\Phi(\widetilde{U}_a, \widetilde{V}_a) \leq \Phi(\widetilde{U}_a, V_a)$ for any V_a and $\Phi(\widetilde{U}_a, \widetilde{V}_a) \leq \Phi(U_a, \widetilde{V}_a)$ for any U_a. Therefore, if $\{(U_a^{(k)}, V_a^{(k)})\}$ is bounded, any accumulation point of $\{(U_a^{(k)}, V_a^{(k)})\}$ is a first order stationary point and a coordinate-wise minimum of Φ.*

5.2. Relations with general convergence analyses of block coordinate descent methods

Theorems 2.8 and 2.9 in [25] provide results on global convergence and convergence rates of general block coordinate descent methods under certain assumptions and the additional assumption that the sequence of iterates has a finite accumulation point (which is guaranteed by a boundedness assumption as that in Theorem 4). The three key assumptions on the function Φ in (5.2) that would allow us to apply the results in [25] are: (1) the functions $f_k(V_a) := \Phi(U_a^{(k-1)}, V_a)$ and $g_k(U_a) := \Phi(U_a, V_a^{(k)})$ are strongly convex for all k and their strongly convex parameters have positive lower and upper bounds independent of k [25, p. 1763]; (2) grad Φ is Lipschitz continuous on any bounded set; and (3) Φ satisfies the Kurdyka–Łojasiewicz property. The function Φ satisfies (2) because the entries of the gradient are cubic polynomials of the real and imaginary parts of the entries of U_a, V_a. It also satisfies (3) because Φ is a quartic polynomial of the real and imaginary parts of the entries of U_a, V_a [25, p. 1768]. However, Φ does not satisfy (1). Since $f_k(V_a)$ and $g_k(U_a)$ are quadratic functions, it is not difficult to prove using [1, Example 5.19] that the condition (1) is equivalent to the existence of two numbers ℓ, L, $0 < \ell < L$, such that the matrices $\mathbf{M}(U_a^{(k)})$ and $\mathbf{M}(V_a^{(k)})$ have full column rank and their minimum singular values are included in the interval $[\ell\,,\ L]$ for all k. It is easy to construct matrices $\mathbf{M}(U_a)$ and $\mathbf{M}(V_a)$ which do not have full column rank. Thus, the required strong convexity cannot be guaranteed in general. However, it might hold in practice, i.e., for most initializations of the algorithm, which would explain the good behavior that we have observed in practice.

6. Numerical experiments

In this section, we compare our algorithm with existing methods. First, to evaluate if our method can match the best existing approximations, we apply it to matrix pencils and polynomials that have been investigated before. Then we compute the distance to singularity for randomly generated matrix pencils and polynomials using our method and the other state-of-the-art methods. The restriction to the distance-to-singularity problem in the comparisons is necessary because mostly the currently available methods do not address the more general problem of approximation by matrix polynomials of arbitrary prescribed rank, such as rank at most r. Finally, we illustrate the behavior of our method as the approximating rank r increases. We refer to our method as GEAR, since it is based on GEneric factorizations and Alternating direction method for Rank-r approximation of matrix polynomials.

All experiments were carried out in MATLAB R2022b Update 8 on a MacBook Pro with an Apple M1 Max processor and 32GB of memory. The implementation is available on GitHub.[2]

6.1. Experiments with specific matrix pencils

We consider regular matrix pencils $A_0 + \lambda A_1$ taken from different studies on distance to singularity and present the estimated distances for them in Table 2. In the last column, we cite the papers where these pencils appear. Our results are typically the same or a bit better than those of the previous studies. Note also that computed distance can often be slightly improved by decreasing the tolerance, but it comes with an increase of the computing time.

Define the following matrices that will be used as building blocks in the example in row 6 of Table 2,

$$K_0 = \begin{bmatrix} -67.4894 & -69.2393 & 69.2393 \\ -69.8124 & -1.68624 & 1.68617 \\ 69.8123 & 1.68617 & 68.2707 \end{bmatrix}, \quad D_0 = \begin{bmatrix} 1.52143 & 1.55168 & -1.55168 \\ -3.22064 & -3.28467 & 3.28467 \\ 3.22064 & 3.28467 & -3.28467 \end{bmatrix},$$

$$M_0 = \begin{bmatrix} 18.7532 & -7.94493 & 7.94494 \\ -7.94493 & 31.8182 & -26.8182 \\ 7.94494 & -26.8182 & 26.8182 \end{bmatrix}, \quad F_0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

6.2. Experiments with specific matrix polynomials

In this subsection, we consider several matrix polynomial examples that have previously appeared in the literature.

Example 2. *Consider the degree–three matrix polynomial* $P(\lambda) = P_0 + \lambda P_1 + \lambda^2 P_2 + \lambda^3 P_3$, *where*

$$P_0 = \begin{bmatrix} -0.1414 & -0.149 \\ 1.1928 & 0.9702 \end{bmatrix}, \quad P_1 = \begin{bmatrix} 0.8837 & 0.9969 \\ 0.219 & 0.0259 \end{bmatrix},$$

[2] https://github.com/dmand/GEAR_framework

Table 2. Computed distances to singularity, d_{sing}, for specific matrix pencils obtained with our method, together with references to previous studies of these problems.

Ex.	A_0	A_1	d_{sing}	References
1	$\begin{bmatrix} 1 & 1/\varepsilon \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 1/\varepsilon \\ 0 & 1 \end{bmatrix}$	ε	[4,5,11]
2	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \varepsilon & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$	ε	[4,5,11]
3	$\begin{bmatrix} 0 & 0.04 & 0.89 \\ 0.15 & -0.02 & 0 \\ 0.92 & 0.11 & 0.066 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	0.1155462894	[5,11,13,18]
4	$\begin{bmatrix} -1.79 & 0.1 & -0.6 \\ 0.84 & -0.54 & 0.49 \\ -0.89 & 0.3 & 0.74 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	0.9435641675	[5,11,13,18]
5	$\begin{bmatrix} 1 & -1 & -1 & \dots & -1 \\ & 1 & -1 & \dots & -1 \\ & & \ddots & \ddots & \vdots \\ & & & 1 & -1 \\ & & & & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & 1 & 1 & \dots & 1 \\ & -1 & 1 & \dots & 1 \\ & & \ddots & \ddots & \vdots \\ & & & -1 & 1 \\ & & & & -1 \end{bmatrix}$	4.0460×10^{-6}	[4,11]
6	$\begin{bmatrix} 0 & I_3 & 0 \\ K_0 & D_0 & F_0^T \\ F_0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} I_3 & 0 & 0 \\ 0 & M_0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	0.0111718453	[4,5,11,18]

$$P_2 = \begin{bmatrix} 0.6346 & 0.9689 \\ 0.6252 & -0.0649 \end{bmatrix}, \quad P_3 = \begin{bmatrix} -1.9867 & 1.28 \\ 0.6097 & -0.1477 \end{bmatrix}.$$

Using our algorithm, we obtain $d_{\text{sing}} = 1.67654181429$ *which is consistent with the estimated distance reported in Example 9.3 of* [5].

Example 3. *We consider the quadratic matrix polynomial* $P(\lambda) = P_0 + \lambda P_1 + \lambda^2 P_2$, *with*

$$P_0 = \begin{bmatrix} 0.0278 & 0.0563 & 0.1141 \\ -0.1758 & 0.327 & -0.173 \\ -0.056 & 0.0321 & -0.075 \end{bmatrix}, \quad P_1 = \begin{bmatrix} -0.2122 & 0.363 & -0.1385 \\ 0.18027 & -0.151 & 0.469 \\ -0.106 & 0.212 & -0.1514 \end{bmatrix},$$

$$P_2 = \begin{bmatrix} -0.0376 & 0.107 & 0.293 \\ 0.003 & -0.14914 & -0.2859 \\ 0.0577 & 0.1455 & 0.231 \end{bmatrix}.$$

This example was previously studied in Example 9.8 of [5] *and in Section 5.2 of* [12], *where the estimated distance to singularity was reported as* 0.026604, *which agrees with our computations.*

Example 4. *We consider a degree–three matrix polynomial* $P(\lambda) = P_0 + \lambda P_1 + \lambda^2 P_2 + \lambda^3 P_3$, *where*

$$P_3 = \begin{bmatrix} 0 & 0.01837967 & 0 & 0 \\ 0 & 0.01843603 & 0 & 0 \\ 0 & 0.01829203 & 0 & 0 \\ 0 & 0.01842778 & 0 & 0 \end{bmatrix}, \quad P_2 = \begin{bmatrix} 0.01833149 & 0.0366177 & 0.01824331 & 0.03660918 \\ 0.01837542 & -0.05442525 & 0 & 0.01832234 \\ 0.01841784 & 0.00003900436 & 0 & 0.0183651578 \\ 0.01840752 & 0.00001508311 & 0.01839699 & 0.0365917 \end{bmatrix},$$

$$P_1 = \begin{bmatrix} 0.09116729 & 0.0000179769 & 0.2550857 & 0.05475106 \\ 0.0001156514 & 0.00001659159 & 0.09108906 & -0.05447104 \\ 0.05470823 & 0.03662426 & 0.1276959 & 0.03650378 \\ 0.05472202 & -0.1091389 & 0.1458359 & -0.09090507 \end{bmatrix},$$

$$P_0 = \begin{bmatrix} 0.09108776 & -0.05442464 & 0.3645006 & 0.01821543 \\ -0.1456436 & 0.03647524 & -0.07277662 & 0.07305016 \\ 0.05478714 & -0.05444916 & 0.437322 & 0.05478385 \\ -0.1274211 & 0.09124859 & -0.6556615 & -0.0544685 \end{bmatrix}.$$

The distances to rank-2 approximations reported in the literature for this polynomial are 7.7049×10^{-4} *in* [5, Example 9.1] *and* 7.844×10^{-4} *in* [13, Example 5.6]. *Once again, our algorithm matches the best reported result, computing an approximation at a distance of* 7.7049×10^{-4}.

6.3. Speed and quality for random matrix pencils

Example 5. *We calculate a distance to singularity using the three potentially fastest available algorithms, i.e., GEAR (our algorithm), Riemannian* [11], *and BFGS* [5], *for 40 random matrix pencils of each size (20 real and 20 complex), with the sizes from* 5×5 *to* 35×35. *The results for matrix pencils with real entries are summarized in Figure 1 and Table 3. The graph on the left in Figure 1 shows that for pencils of these sizes our algorithm is significantly (100 times) faster than competitors. Moreover, the graph on the right in Figure 1 and Table 3 show that the computed distance to singularity by our algorithm is typically better (smaller).*

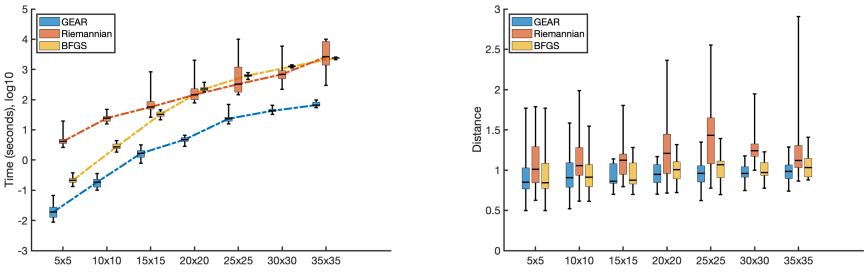


Figure 1. Comparison of the speed and computed minimal distances for random real matrix pencils of given size.

	5×5	10×10	15×15	20×20	25×25	30×30	35×35
GEAR	4	10	10	17	15	14	18
Riemannian	1	0	1	2	1	1	1
BFGS	15	10	9	1	4	5	1

Table 3. The table shows how many times each algorithm, i.e., GEAR, Riemannian, or BFGS (listed in the first column), returned the best answer out of 20 experiments for random real matrix pencils of a corresponding size (between 5 and 35, all sizes are listed in the first row).

The results for matrix pencils with complex entries are summarized in Figure 2 and Table 4. The graph on the left in Figure 2 shows that for pencils of sizes up to 30×30 *our algorithm is faster than competitors. For* 35×35 *pencils Riemannian is the fastest algorithm, but the variation of its outputs is very large comparing to the other two algorithms. The graph on the right in Figure 2 and Table 4 show that the computed distance to singularity by our algorithm is typically better (smaller).*

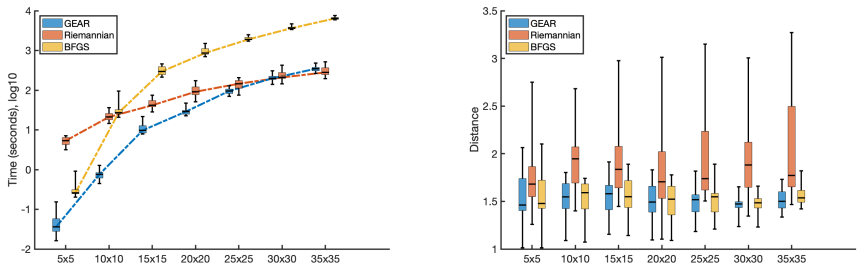


Figure 2. Comparison of the speed and computed minimal distances for random complex matrix pencils of given size.

Example 6. *We calculate a distance to singularity using the two fastest algorithms (based on Example 5), i.e., GEAR (our algorithm) and Riemannian* [11]*, for 20 random matrix pencils of each size (10 real and 10 complex), with the sizes* 50×50*,* 75×75*, and* 100×100*. The results for matrix pencils with real and complex entries are summarized in Figures 3 and 4, respectively. The plots in Figure 3 indicate that, for real pencils of these sizes, GEAR is consistently faster than Riemannian and we checked that each time it computes a smaller distance to singularity. Figure 4 shows that for complex pencils of these sizes our algorithm with the tolerance parameter set to* 10^{-7} *is slower than Riemannian but consistently produces approximations at a smaller distance to singularity, while our algorithm with the tolerance parameter set to* 10^{-5} *is slightly faster than Riemannian; however, as the pencil*

	5×5	10×10	15×15	20×20	25×25	30×30	35×35
GEAR	3	7	7	8	11	13	18
Riemannian	0	0	1	0	0	0	1
BFGS	17	13	12	12	9	7	1

Table 4. The table shows how many times each algorithm, i.e., GEAR, Riemannian, or BFGS (listed in the first column), returned the best answer out of 20 experiments for random complex matrix pencils of a corresponding size (between 5 and 35, all sizes are listed in the first row).

size increases, it more frequently produces approximations at a larger distance than those obtained by Riemannian.

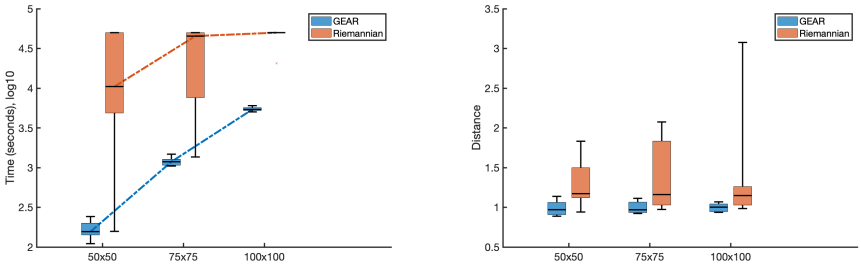


Figure 3. Comparison of the speed and computed minimal distances for random real matrix pencils of given size. For a large fraction of experiments and, in particular for all the 100×100 experiments, Riemannian reached a maximum allowed time without converging (increasing the maximum allowed time did not improve the situation). In addition, we checked that GEAR found a smaller minimum than Riemannian in each of the 30 performed experiments.

6.4. Speed and quality for random matrix polynomials

Example 7. *We calculate a distance to singularity using four available algorithms, i.e., GEAR (our algorithm), MVF* [14]*, Riemann-Oracle (RO)* [16]*, and BFGS* [5]*, for 40 random matrix polynomials of degree* 3*, and of sizes from* 3×3 *to* 15×15 *(we take 16 polynomials of each size, 8 real and 8 complex). The results for matrix polynomials with real entries are summarized in Figure* 5*. The graph on the left in Figure* 5 *shows that for polynomials of these sizes our algorithm is the fastest among the tested algorithms. To compare the quality of the minima obtained by the algorithms, we now assess the "singularity" of the obtained output, as seen in the graph on the right in Figure* 5*, as well as the distance to it, which is shown in the middle graph in Figure* 5 *(comparing just the latter was sufficient in the pencil case). Our algorithm and BFGS produce the answers that are "the most singular".*

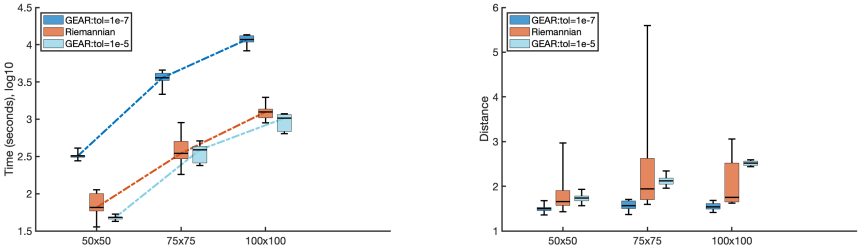


Figure 4. Comparison of the speed and computed minimal distances for random complex matrix pencils of given size. In addition, we checked that GEAR with the tolerance parameter set to 10^{-7} produced approximations at the smallest distance to singularity in each of the 30 performed experiments.

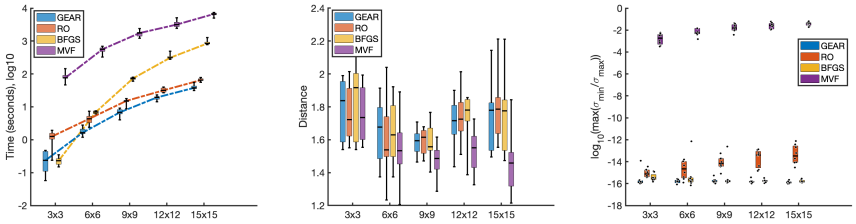


Figure 5. Comparison of the speed, computed minimal distances to singularity, and singularity of the outputs for random real matrix polynomials of degree 3 and given size. To compare the "singularity" of the output, we plot, in the right-most graph, the maximum value of the ratio of the smallest singular value $\sigma_{\min}$ to the largest singular value $\sigma_{\max}$ of the constructed matrix polynomial evaluated at the points $x_j + \sqrt{-1}\, y_j$, where $[X, Y] = \texttt{meshgrid}(-100 : 20 : 100)$. In all the experiments performed, GEAR produces outputs with this ratio at the level of machine precision.

The results for matrix polynomials with complex entries are summarized in Figure 6. The left plot in Figure 6 shows that, for polynomials of these sizes, our algorithm is the fastest. To compare the quality of the minima obtained by the different algorithms, we again assess both the "singularity" of the computed outputs, shown in the right plot of Figure 6, and their distance to the original polynomial, shown in the middle plot. Our algorithm and BFGS perform best in preserving the singularity, followed by RO. The latter three algorithms produce outputs at approximately the same distance from the original polynomial.

Example 8. *We illustrate how the time needed for computing an approximation with GEAR decreases with the decrease of the rank. As an example, we use 60 random real matrix polynomials of degree 2 and sizes* 70×70, 120×120, *and* 170×170 *(4 matrix polynomials per each size and rank). In particular, in all the cases considered, computing a rank-5*

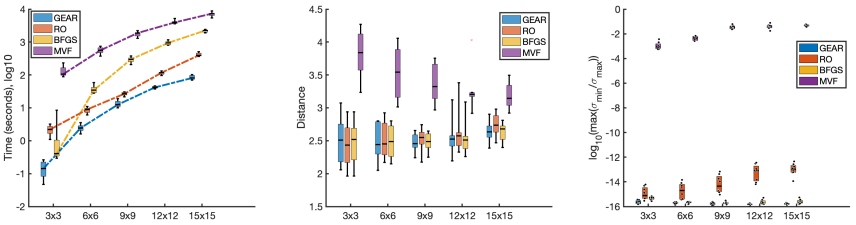


Figure 6. Comparison of the speed, computed minimal distances to singularity, and singularity of the outputs for random complex matrix polynomials of degree 3 and given size. To compare the "singularity" of the output, we plot, in the right-most graph, the maximum value of the ratio of the smallest singular value $\sigma_{\min}$ to the largest singular value $\sigma_{\max}$ of the constructed matrix polynomial evaluated at the points $x_j + \sqrt{-1}\, y_j$, where $[X, Y] = \texttt{meshgrid}(-100 : 20 : 100)$. In all the experiments performed, GEAR produces outputs with this ratio at the level of machine precision.

approximation is around 100 times faster than computing a rank-45 approximation, see more in Figure 7.

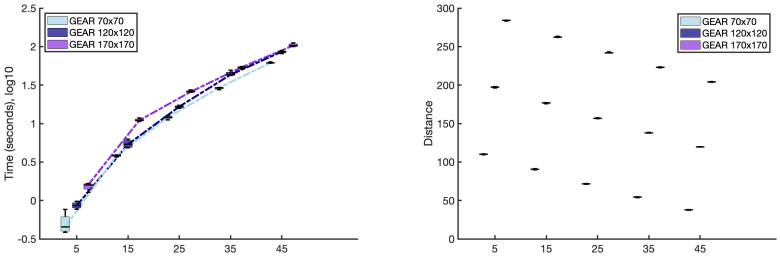


Figure 7. Speed and computed minimal distances to polynomials of fixed rank (varying from 5 to 45) for random real matrix polynomials of degree 2 and sizes 70×70, 120×120, and 170×170.

7. Conclusions and future work

We have introduced a novel framework for low-rank approximation of matrix polynomials. By using explicit parametrizations of generic orbits of matrix polynomials with prescribed normal rank, the original constrained problem is transformed into an unconstrained nonlinear least-squares problem. One of the main advantages of this formulation is that the rank constraint is enforced by construction: every polynomial generated by the method has normal rank at most the prescribed target rank. The framework applies to arbitrary degrees and to arbitrary ranks $r < \min(m, n)$, and is not restricted to the distance-to-singularity

case. The alternating least-squares method in this paper provides a simple and practical realization of this idea, and the numerical experiments show that it is very effective.

The proposed framework is modular. Different generic rank-r families, different parametrizations, and different solvers can be combined following the same approach. Therefore, our framework opens up multiple directions for future research. The most pressing one is the development of faster and more scalable optimization methods. Speed is still a bottleneck for large problems; improving it would make it possible to handle larger matrix polynomials and to use multiple starting guesses for problems of moderate size. Since the resulting nonlinear least-squares problems are nonconvex, multiple initializations may be important for obtaining better approximations. Other important directions include the design of reliable initialization strategies, the development of structure-preserving methods (feasibility and efficiency of this direction has been showed in a recent preprint [9]), and the study of alternative parametrizations. Moreover, improving our understanding of the geometry of the involved objects may also lead to sharper convergence results, improved stopping criteria, and better indicators of the quality of the computed approximations.

Funding. S. Das and A. Dmytryshyn were supported by the Swedish Research Council (VR) grant 2021-05393. F. M. Dopico was funded by MICIU/AEI/10.13039/501100011033/ and by FEDER/UE through grant PID2023-147366NB-I00.

References

[1] A. Beck. *First-order methods in optimization*, volume 25 of *MOS-SIAM Ser. Optim.* Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM), 2017.

[2] T. Betcke, N. Higham, V. Mehrmann, C. Schröder, and F. Tisseur. NLEVP: A collection of nonlinear eigenvalue problems. *ACM Trans. Math. Software*, 39(2):7:1–7:28, 2013.

[3] N. Boumal. *An introduction to optimization on smooth manifolds*. Cambridge: Cambridge University Press, 2023.

[4] R. Byers, C. He, and V. Mehrmann. Where is the nearest non-regular pencil? *Linear Algebra Appl.*, 285(1):81–105, 1998.

[5] B. Das and S. Bora. Nearest rank deficient matrix polynomials. *Linear Algebra Appl.*, 674:304–350, 2023.

[6] F. De Terán and F. M. Dopico. Generic change of the partial multiplicities of regular matrix pencils under low-rank perturbations. *SIAM J. Matrix Anal. Appl.*, 37(3):823–835, 2016.

[7] F. De Terán, F. M. Dopico, and J. M. Landsberg. An explicit description of the irreducible components of the set of matrix pencils with bounded normal rank. *Linear Algebra Appl.*, 520:80–103, 2017.

[8] A. Dmytryshyn and F. M. Dopico. Generic complete eigenstructures for sets of matrix polynomials with bounded rank and degree. *Linear Algebra Appl.*, 535:213–230, 2017.

[9] A. Dmytryshyn, F. M. Dopico, and R. Hellberg. Distance to nearest skew-symmetric matrix polynomials of bounded rank. *arXiv preprint arXiv:2601.16676*, 2026.

[10] A. Dmytryshyn, F. M. Dopico, and P. Van Dooren. Minimal rank factorizations of polynomial matrices. *Linear Algebra Appl.*, 721:736–774, 2025.

[11] F. M. Dopico, V. Noferini, and L. Nyman. A Riemannian optimization method to compute the nearest singular pencil. *SIAM J. Matrix Anal. Appl.*, 45(4):2007–2038, 2024.
[12] M. Giesbrecht, J. Haraldson, and G. Labahn. Computing the nearest rank-deficient matrix polynomial. In *Proceedings of the 2017 ACM International Symposium on Symbolic and Algebraic Computation*, ISSAC '17, pages 181–188, New York, NY, USA, 2017. ACM.
[13] M. Giesbrecht, J. Haraldson, and G. Labahn. Computing lower rank approximations of matrix polynomials. *Journal of Symbolic Computation*, 98:225–245, 2020.
[14] M. Gnazzo and N. Guglielmi. Approximating the closest structured singular matrix polynomial. *Linear and Multilinear Algebra*, 73(9):2222–2250, 2025.
[15] M. Gnazzo and N. Guglielmi. On the numerical approximation of the distance to singularity for matrix-valued functions. *SIAM J. Matrix Anal. Appl.*, 46(2):1484–1517, 2025.
[16] M. Gnazzo, V. Noferini, L. Nyman, and F. Poloni. Riemann-oracle: A general-purpose Riemannian optimizer to solve nearness problems in matrix theory. *Found. Comput. Math.*, pages 1–45, 2025.
[17] G. Golub and C. Van Loan. *Matrix Computations*. John Hopkins University Press, Baltimore, MD, 4rd edition, 2013.
[18] N. Guglielmi, C. Lubich, and V. Mehrmann. On the nearest singular matrix pencil. *SIAM J. Matrix Anal. Appl.*, 38:776–806, 2017.
[19] D. Kressner and M. Voigt. Distance problems for linear dynamical systems. In *Numerical Algebra, Matrix Theory, Differential-Algebraic Equations and Control Theory*, pages 559–583. Springer, Cham, 2015.
[20] Q. Li, Z. Zhu, and G. Tang. Alternating minimizations converge to second-order optimal solutions. In *International Conference on Machine Learning*, pages 3935–3943. PMLR, 2019.
[21] Y. Li, Y. Liang, and A. Risteski. Recovery guarantee of weighted low-rank approximation via alternating minimization. In M. F. Balcan and K. Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2358–2367, New York, New York, USA, 20–22 Jun 2016. PMLR.
[22] D. S. Mackey, N. Mackey, and F. Tisseur. Polynomial Eigenvalue Problems: Theory, Computation, and Structure. In *Numerical Algebra, Matrix Theory, Differential-Algebraic Equations and Control Theory*, pages 319–348. Springer, Cham, 2015.
[23] N. Srebro and T. Jaakkola. Weighted low-rank approximations. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 720–727, 2003.
[24] F. Tisseur and K. Meerbergen. The quadratic eigenvalue problem. *SIAM Rev.*, 43(2):235–286, 2001.
[25] Y. Xu and W. Yin. A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. *SIAM J. Imaging Sci.*, 6(3):1758–1789, 2013.

Sweta Das
School of Science and Technology, Örebro University, 701 82 Örebro, Sweden; sweta.das@oru.se

Andrii Dmytryshyn
Department of Mathematical Sciences, Chalmers University of Technology and University of Gothenburg, Chalmers Tvärgata 3, 412 96 Gothenburg, Sweden; Corresponding author; andrii@chalmers.se
Author IDs: ORCID 0000-0001-9110-6182

Froilán M. Dopico
Departamento de Matemáticas, Universidad Carlos III de Madrid, Avenida de la Universidad, 30 (edificio Sabatini), 28911 Leganés (Madrid), Spain; dopico@math.uc3m.es
Author IDs: ORCID 0000-0001-7779-8960

Mårten Gulliksson
School of Science and Technology, Örebro University, 701 82 Örebro, Sweden;
marten.gulliksson@oru.se