Calcolo (2025) 62:2
https://doi.org/10.1007/s10092-024-00623-y

Implementation improvements and extensions of an ODE-based algorithm for structured low-rank approximation

Antonio Fazzi[1] · **Ivan Markovsky**[2,3] · **Konstantin Usevich**[1]

Received: 24 February 2022 / Revised: 8 November 2024 / Accepted: 11 November 2024

Abstract
In the framework of structured matrix low-rank approximation, we propose implementation improvements and extensions of a gradient system methodology that is based on the iterative integration of a system of ODEs. The improvements are based on numerical techniques for the computation of SVDs and rank-1 matrices projection. Some extensions of the numerical method to variations of the classical structured low-rank approximation problems are then proposed.

Keywords Numerical optimization · Structured low-rank approximation · ODEs integration · Matrix nearness problems

Mathematics Subject Classification 15A24 · 41A29 · 65F45 · 65K10

1 Introduction

The problem of approximating a structured matrix with another one of lower rank (Structured Low-Rank Approximation, SLRA) is an important problem in the numerical linear algebra community. This is a difficult non-convex optimization with no

Antonio Fazzi, Ivan Markovsky, and Konstantin Usevich contributed equally to this work.

✉ Antonio Fazzi
antonio.fazzi@univ-lorraine.fr

Ivan Markovsky
imarkovsky@cimne.upc.edu

Konstantin Usevich
konstantin.usevich@univ-lorraine.fr

[1] Centre de Recherche en Automatique de Nancy (CRAN), Université de Lorraine, CNRS, Campus Sciences, 54506 Vandœuvre-lès-Nancy, France

[2] International Centre for Numerical Methods in Engineering, Gran Capità, 08034 Barcelona, Spain

[3] Catalan Institution for Research and Advanced Studies, Pg. Lluis Companys 23, 08010 Barcelona, Spain

Published online: 27 November 2024

closed-form solution [1], for which several formulations of the problem based on different cost functions to be minimized and associated constraints exist. New algorithms for solving this problem are based on modern numerical techniques aiming to improve the computational speed and/or the accuracy of the error on the computed solution. Following these ideas, in this paper, we want to propose computational improvements and extensions of a recent methodology for the numerical solution of some SLRA problems based on the integration of ordinary differential equations (ODEs). The methodology was introduced in [2, 3] for the computation of approximate common factors of polynomials (Sylvester low-rank approximation) and in [4] for Hankel low-rank approximation. The intent to improve these algorithms is motivated by their numerical performance. In comparison with other existing methods, they showed an improvement in the accuracy of the computed solutions, but sometimes they lacked from the point of view of the computational speed.

The goal of the paper is twofold: on the one hand, we propose improvements in terms of computational cost and time for the main numerical computations in ODE-based algorithms; on the other hand, we show how to adapt some of these algorithms in order to deal with other applications and with more general structures than the ones considered in the previous works. However, as discussed in the final part of the paper, this is only a small part (even if still relevant) of the possible improvements that can be tested for this methodology. We remark that some of the observed results are numerical evidence only.

Paper organization

The paper is organized as follows: in Sect. 2 we recall the main points and features of the algorithms in [2–4]; in particular, we provide a unified view of several algorithms and a formulation that naturally covers other types of structured matrices. Sections 3 and 4 deal with the problems of Sylvester and Hankel low-rank approximation, respectively, describing the improvements in the computational performance and testing them on numerical examples. Finally, Sect. 5 describes how to adapt the considered algorithm to the estimation of time series with missing coefficients (Sect. 5.1) and to block structured matrices (Sect. 5.2). A general discussion about possible directions for further improvements is given in Sect. 6.

Nomenclature The numerical algorithms considered in the paper are variations of a methodology based on the iterative integration of a system of ODEs. Implementation details and numerical techniques differ among the various versions. To ease the readability of the manuscript, we give some short names to the different versions of the gradient system methodology:

- Syl-eig [2]: SLRA of a Sylvester matrix by the minimization of the smallest (in modulus) eigenvalue;
- Syl-svd [3]: SLRA of a Sylvester matrix by the minimization of the smallest singular values;
- Hank-svds [4]: SLRA of a Hankel matrix by the minimization of the smallest singular values.

In the next sections, we define new acronyms to underline the numerical techniques that characterize the numerical computations in the corresponding variation of the algorithm.

2 The algorithm

In this section we briefly summarize the computational steps of the algorithms Syl-svd [3] and Hank-svds [4] for the problems of Sylvester and Hankel structured low-rank approximation, respectively.

2.1 Structured low-rank approximation problems

Sylvester and Hankel matrices are functions of a parameter vector $p \in \mathbb{R}^{n_p}$ and they are in the image of an affine function $\mathcal{S} : \mathbb{R}^{n_p} \to \mathbb{R}^{m \times n}$ (usually $1 \leq n_p \ll mn$).

Definition 1 (Sylvester matrices) Given $p = \{\bar{p}_1; \ldots; \bar{p}_n\}$, where each $\bar{p}_i \in \mathbb{R}^{d+1}$ represents the $d+1$ coefficients of a degree-d polynomial (possibly padding with zeros the missing leading coefficients), the Sylvester matrix $Syl(p) \in \mathbb{R}^{nd \times 2d}$ is defined as the block column matrix

$$Syl(p) = \begin{pmatrix} \mathcal{T}(\bar{p}_1) \\ \vdots \\ \mathcal{T}(\bar{p}_n) \end{pmatrix},$$

where the matrix $\mathcal{T}(\cdot)$ is defined on a degree-d polynomial as

$$\mathcal{T}(q) = \begin{pmatrix} q_d & \cdots & q_1 & q_0 & 0 & \cdots & 0 \\ 0 & q_d & \cdots & q_1 & q_0 & \cdots & 0 \\ & \ddots & \ddots & & \ddots & \ddots & \\ 0 & \cdots & 0 & q_d & \cdots & q_1 & q_0 \end{pmatrix} \in \mathbb{R}^{d \times 2d}.$$

Definition 2 (Hankel matrices) Given a vector $p \in \mathbb{R}^{n_p}$, a Hankel matrix with m rows is defined as

$$\mathcal{H}_m(p) = \begin{pmatrix} p_1 & p_2 & \cdots & p_{n_p-m+1} \\ p_2 & p_3 & \cdots & p_{n_p-m+2} \\ \vdots & \vdots & & \vdots \\ p_m & p_{m+1} & \cdots & p_{n_p} \end{pmatrix} \in \mathbb{R}^{m \times (n_p-m+1)}. \tag{1}$$

We assume that the parameter vector p is such that the starting matrix is full-rank and the problem objective is to compute an approximation $\hat{p}$ such that the structured matrix (Sylvester or Hankel) is of lower rank. More formally,

Problem 1 *Given a vector $p \in \mathbb{R}^{n_p}$, a structure specification $\mathcal{S} : \mathbb{R}^{n_p} \to \mathbb{R}^{m \times n}$ (in this paper $\mathcal{S}$ can be Syl or $\mathcal{H}_m$) and a bound on the rank $r < \min(m, n)$, we want to solve the optimization problem*

$$\min_{\hat{p} \in \mathbb{R}^{n_p}} \|\mathcal{S}(p) - \mathcal{S}(\hat{p})\|_W \quad \textit{such that} \quad \textit{rank}\, \mathcal{S}(\hat{p}) \leq r, \tag{2}$$

where $\|\cdot\|_W$ is an elementwise weighted (semi-)norm on the space of $m \times n$ matrices, defined by a positive semidefinite matrix $W \in \mathbb{R}^{mn \times mn}$:

$$\|X\|_W^2 = vec(X)^\top W vec(X), \tag{3}$$

and $vec(X)$ is the standard vectorization operator.

The value of r in (2) is usually context dependent. There exist applications where a rank reduction by one is suitable, such as the distance to uncontrollability [5, Section 3] or identification of autonomous linear systems [4, Section 2.1].

A typical choice of the seminorm is the Frobenius norm (with $W = I_{mn}$ in (3)), as in Syl-svd [3]. In other papers, SLRA is stated as minimization of the error on the parameter vector $\hat{p}$ (see [1, 6]), such as in Hank-svds [4]. As it is shown in [7, Lemma 2.2], the two formulations are equivalent.

Example 1 For an $m \times n$ Hankel matrix, $m \leq n$, the 2-norm approximation of the parameter vector is achieved by

$$W = \text{diag}(\underbrace{1, \ldots, 1}_{m}, \underbrace{0, \ldots, 0, 1}_{m}, \ldots, \underbrace{0, \ldots, 0, 1}_{m}),$$

so that $\|p - \hat{p}\|_2 = \|\mathcal{S}(p) - \mathcal{S}(\hat{p})\|_W$ for such choice of the matrix W. There are other alternative weighting matrices, for example, the matrix W_* proposed in [7, Section 3].

Remark 1 In this paper, we work with real vectors and matrices. But all the ideas can be naturally extended to the complex case, as discussed in [3, Section 5.2].

2.2 ODE-based methodology

The solution method exploits the equivalence between the rank constraint on the considered matrix and the nullity of a set of singular values. More formally, it computes the smallest structured perturbation $\epsilon E = \mathcal{S}(\hat{p}) - \mathcal{S}(p)$ to the structured matrix whose $(r+1)$-th largest singular value is minimized until an admissible solution is obtained:

$$\epsilon^* = \min\{\epsilon \in \mathbb{R} : \sigma_{r+1}(\mathcal{S}(p) + \epsilon E) = 0, \quad \|E\|_W = 1\}.$$

The methodology iteratively integrates a suitable system of ODEs that describes a descent direction for the singular value σ_{r+1} by preserving the matrix structure. The main idea is to split the perturbation into two factors that are updated iteratively and independently on two iteration levels:

- at the inner level, the norm of the perturbation is fixed and we compute a normalized perturbation E which minimizes the $(r+1)-$th singular value σ_{r+1} of the matrix $\mathcal{S}(p)+\epsilon E$. This is done by integrating the following system of ODEs (that describes a descent dynamic for σ_{r+1} over the ball of matrices whose norm is at most ϵ):
$$v\dot{E} = -\mathcal{P}_{\mathcal{S}}(uv^T) + \langle \mathcal{P}_{\mathcal{S}}(uv^T), E\rangle_W E, \tag{4}$$
where u, v are the singular vectors of $\mathcal{S}(p)+\epsilon E$ corresponding to σ_{r+1}, $\mathcal{P}(\cdot)_{\mathcal{S}}$ denotes the orthogonal projection of the argument matrix onto the set of structured matrices $\mathcal{S}$, and v is the weighted norm of the expression on the right-hand side.
- at the outer level, we update the norm ϵ of the total perturbation ϵE, which can be done by a root-finding algorithm, e.g. a Newton iteration that requires computing the norm of the projection $\mathcal{P}_{\mathcal{S}}(uv^T)$;
- in some of the algorithms, such as Hank-svds [4], additional steps of free (unconstrained) dynamics are performed by removing the constraint on the norm of E in (4). In this case, we get an analogous gradient system that increases the norm on the perturbation on the data, and where the computations are similar to (4).

Remark 2 All the matrices in (4) have the same structure S, hence $E = \mathcal{S}(q_1)$, $\dot{E} = \mathcal{S}(\dot{q}_1)$, $\mathcal{P}_S(uv^T) = \mathcal{S}(q_2)$ for some vectors $q_1, q_2 \in \mathbb{R}^{n_p}$. By replacing each matrix with the corresponding generating vector, (4) can be rewritten as a vectorial equation. This idea is developed in the algorithm Hank-svds [4].

Remark 3 The weighted semi-norm formulation unifies the algorithms in Syl-svd [3] and Hank-svds [4] in the following sense: it can be shown that the updates (4) for Hankel matrices and the choice of the weighted norm as in Example 1 becomes equivalent to the updates described in Hank-svds [4].

By looking at the scheme above, we see that the main computational cost lies in the numerical integration of systems of ODEs. These integrations require the numerical computation of the $(r+1)$-th singular triplet, and then the projection of a rank-one matrix onto the set of structured matrices. Therefore, we discuss alternative (numerical) computations for these two operations.

3 Sylvester low-rank approximation

The Sylvester low-rank approximation problem is usually linked to the approximate common factor computation of polynomials. It is a well-known problem because of its applications in the field of systems and control [1, 5]. A short review of the related literature is presented in [3, 8], and its numerical solution using Syl-svd is in [3].

A brief description of the problem is: given a set $\bar{p}_1, \ldots, \bar{p}_n$ of coprime polynomials, find some polynomials $\hat{p}_1, \ldots, \hat{p}_n$ (as close as possible to the starting ones) having a common factor of degree d. It can be recast in the framework of Problem 1 because of the relation between polynomials common factors and the rank of the associated Sylvester matrix [9]. Therefore, given a full rank Sylvester matrix, we aim at finding a rank-deficient Sylvester matrix that is as close as possible to the original one. The degree of the sought common factor determines the value of r in Problem 1.

We are interested in possible numerical improvements with respect to the algorithm Syl-svd in [3]. Looking back at Sect. 2, we discuss:

1. the projection of a rank-one matrix onto the set of Sylvester matrices;
2. the computation of the $(r+1)$-th singular triplet of a Sylvester matrix.

A classical algorithm for the projection of an arbitrary matrix onto the set of Sylvester matrices (used in Syl-svd) is to partition it into blocks (whose dimensions match the blocks $\mathcal{T}(\bar{p}_1), \ldots, \mathcal{T}(\bar{p}_n)$) and to replace the entries in each block with the average of the corresponding diagonal, following the pattern of zeros of the Sylvester matrix. As a preliminary step, we have to build the rank-one matrix to be projected. The computational cost of this method is quadratic in the dimension of the involved (square) matrix. The same operations can be performed efficiently (via the Fast Fourier Transform) by exploiting the fact that the starting matrix has rank one. In particular, we can adapt the algorithm in [10, Section 5] (described for Hankel matrices) by replacing antidiagonals with diagonals. The solution is computed directly from the vectors u, v without building the matrix uv^T. The computational cost drops from $O(n^2)$ to $O(n \log n)$, where n is the dimension of the matrix.

About the computation of the $(r+1)-$th singular triplet, Syl-svd computed the whole SVD of the matrix to get the sought triplet (via the Matlab function *svd*). Since we only need the $(r+1)-$th singular value with the associated singular vectors, we can use the function *svds*. We exclude the techniques based on eigenvalues: the gain in the use of *svd* in place of *eig* has been observed in [3], and the use of *eigs* would not be very efficient. A further term of comparison (that works for the computation of the smallest singular triplet only) is [11, Algorithm 1]; this is an SVD-based inverse iteration method that is connected with the following equalities (satisfied by the matrix A and its smallest singular triplet u_n, σ_n, v_n)

$$\begin{aligned} Av_n &= \sigma_n u_n \\ v_n^T v_n &= 1. \end{aligned}$$

Numerical experiments

For simplicity, we consider only two polynomials, but the same ideas can be naturally extended to sets of more polynomials, as illustrated in [3]. We consider the following variations of the algorithm Syl-svd:

1. Syl-svd-fft: it still computes the whole SVD to get the $(r+1)$-th singular triplet, but it uses the FFT-based formula for the projection of the rank-one matrix;
2. Syl-svds: it uses the Matlab function *svds* to compute the $(m-r)$-th smallest singular triplet and the FFT-based formula for the projection of the rank-one matrix.

Remark 4 To avoid confusion, we remark that the $(r+1)-$th (largest) singular triplet is the same as the $(m-r)-$th smallest singular triplet. The term *smallest* means that the singular values are increasing (in the function *svds* above, we add the option *smallest*). The same terminology is used in the rest of the paper.

The algorithm Syl-svd-fft only tests the numerical performance of the different method proposed for the projection, while Syl-svds also uses a different technique to compute the singular triplet. While the previous two algorithms work for every value of m, r, if we restrict to the rank reduction by one ($r = m - 1$ in Problem 1), we can add a further algorithm for comparison (that works for the computation of the smallest singular triplet only):

3. Syl-apsvd: it uses [11, Algorithm 1] for the computation of the smallest singular triplet and the FFT-based formula for the projection of the rank-one matrix.

A different numerical method for comparison comes from the literature on the topic, and it is the function *gcd-nls* from the *slra* toolbox [8].

The setup of the experiments is as follows:

- we fix a common factor c of degree 1;
- we multiply c by two random polynomials (of degree $d - 1$) in order to have two polynomials $\bar{p}_1^0$, $\bar{p}_2^0$ of degree d having one common root;
- we generate the coprime polynomials as $\bar{p}_i = \bar{p}_i^0 + \sigma s \|\bar{p}_i^0\|_2$, $i = 1, 2$, where s is a norm 1 vector whose entries come from i.i.d. standard normal distributions and σ is a constant denoted as noise level.

By adding Syl-apsvd in the comparison, we restrict the experiment to the computation of a common factor of degree 1, that is a rank reduction by 1 of the starting Sylvester matrix. In the numerical experiment (whose code is available at [12]) we fix the noise level $\sigma = 0.1$ and we gradually increase the degrees of the polynomials to analyze the numerical performances for matrices of increasing dimension. We mainly focus on the computational times needed by the different algorithms and the relative errors on the computed solutions. We do not consider large degree polynomials in the paper, but we expect the general behavior to be clear enough. All the results are the average over ten different perturbations on the same data polynomials to lower the effect of possible misleading results. We discarded from the experiments the runs where at least one of the methods needed more than 200 iterations to converge (the *convergence test* is to check that the smallest singular value of $\mathcal{S}(\hat{p})$ is lower than 10^{-6}). All the experiments in the paper have been run with MATLAB R2023a on a laptop with a 2.7 GHz Intel Core i5 processor. The outcome of the experiment is in Table 1.

In Table 1, by comparing the first two columns, we can appreciate the gain in the use of the Fast Fourier method for the projection, since the times in the second column are lower without changing the relative error on the solution (the two different algorithms that project the rank-one matrix return the same result, up to machine precision). We observe that Syl-apsvd is even faster than Syl-svd-fft, but the cost to be paid is the slightly higher error on the computed solutions, due to the different algorithms that compute the smallest singular triplet. Syl-svds is faster than Syl-svd-fft only when the dimension of the problems increases, but its performance is, in general, worse than Syl-apsvd.

However, the observed computational times are still not competitive with the function *gcd-nls* from the *SLRA toolbox* [8]. About the errors on the computed solutions in Table 1, most of the time *gcd-nls* achieves better results than Syl-svds and Syl-apsvd, and sometimes the improvement is significant. On the other hand, looking

Table 1 Approximate common factor of degree 1 for two polynomials of increasing degree: computational times and relative errors for different algorithms and problems of increasing dimension

$\deg(\tilde{p}_i)$ / size($S(p)$)		Syl-svd	Syl-svd-fft	Syl-svds	Syl-apsvd	gcd-nls
30 /	time (sec.)	0.8315	0.8024	1.1086	0.5390	0.0278
——	——	——	——	——	——	——
60 × 60	rel. err	0.0105	0.0105	0.0183	0.0159	0.0068
60 /	time (sec.)	1.6730	1.5493	1.4198	0.7970	0.0195
——	——	——	——	——	——	——
120 × 120	rel. err	0.0018	0.0018	0.0065	0.0063	0.0017
90 /	time (sec.)	3.3895	3.0296	3.5985	1.9083	0.0032
——	——	——	——	——	——	——
4 180 × 180	rel. err	0.0011	0.0011	0.0014	0.0030	0.0022
120 /	time (sec.)	14.8563	11.8187	4.1562	2.6040	0.0034
——	——	——	——	——	——	——
240 × 240	rel. err	0.0014	0.0014	0.0039	0.0049	0.0017
150	time (sec.)	41.8031	30.4069	6.8686	3.1824	0.0039
——	——	——	——	——	——	——
300 × 300	rel. err	0.0008	0.0008	0.0027	0.0023	0.0017
180	time (sec.)	31.1686	16.9135	11.5349	8.9463	0.0046
——	——	——	——	——	——	——
360 × 360	rel. err	0.0002	0.0003	0.0006	0.0009	0.0015

at the errors of Syl-svd and Syl-svd-fft, they are the smallest overall for the higher dimension problems (from the third row on).

Remark 5 The same ideas to speed up the algorithm Syl-svd can be adapted to the computation of approximate common factors of matrix polynomials (polynomials whose coefficients are matrices), presented in [13].

4 Hankel low-rank approximation

We switch now to the problem of Hankel low-rank approximation. These matrices play a leading role in system theory and identification since a well-known result [14, Theorem 7.2] links the order of a linear time-invariant autonomous system with the rank of the Hankel matrix built from one of its trajectories. We remark that the Hankel matrices considered in these problems (and in the paper) are, in general, rectangular Hankel matrices with more columns than rows.

The problem we aim to solve can be briefly described as follows: given a full rank Hankel matrix $\mathcal{H}_m(p)$, compute a vector $\hat{p}$ (as close as possible to p) such that $\mathcal{H}_m(\hat{p})$ is rank deficient. We observe that, differently from Sect. 3, the problem Hank-svds in [4] is stated directly on the parameter vector p instead of the Hankel matrix $\mathcal{H}_m(p)$.

As we did in Sect. 3, we take into account the following two main computations performed during the integration of the associated gradient systems:

1. the projection of a rank-one matrix onto the set of Hankel matrices;
2. the computation of the $(r+1)$-th singular triplet of a Hankel matrix.

For the first point, we can apply directly the algorithm in [10, Section 5].

For the second point, [11, Algorithm 1] cannot be used on $\mathcal{H}_m(p)$ since the matrix is not square, and we avoid building $\mathcal{H}_m(p)^T\mathcal{H}_m(p)$ since it would square the condition number.

The size of the involved Hankel matrices (that have a few rows and many columns) can be exploited to use a randomized SVD algorithm [15] which computes a truncated SVD; this turns out to be efficient since the involved matrices always have a few rows.

Numerical experiments

We are now ready to run some simulations to test alternative numerical computations in the scheme of the algorithm Hank-svds. We recall that Hank-svds [4] used the Matlab function *svds* to compute the needed singular triplet and the antidiagonal averaging to project the matrix uv^T onto the set of Hankel matrices. We consider the following variations

1. Hank-svds-fft: it still uses the function *svds* for the computation of the smallest $(m-r)$-th singular triplet, but the FFT-based formula for the projection of the rank-one matrix;
2. Hank-rsvd: it uses a randomized SVD to get the $(r+1)-$th singular triplet (we run this computation using the function in [16]) and the FFT-based formula for the projection of the rank-one matrix.

The algorithm Hank-svds-fft tests the numerical performances of the projection algorithm only, while Hank-rsvd also uses a different technique to compute the $(r+1)$-th singular triplet. Another term of comparison is the function *slra* from the homonymous toolbox [6].[1]

The setup of the numerical experiments follows:

- we build a (random) state-space model sys of fixed order $k = 5$;
- we consider a time series p_0 as the response to a (random) initial condition of the given state-space model sys (observe that the Hankel matrix $\mathcal{H}_{k+\ell}(p_0)$ has rank $k \; \forall \; \ell > 0$ [14]);
- we generate the problem data by perturbing the time series p_0 as $p = p_0 + \sigma s \| p_0 \|_2$, where s is a normalized vector whose entries come from i.i.d. standard normal distributions and σ is a constant denoted by noise level;
- we aim at approximating p with a vector $\hat{p}$ such that the rectangular Hankel matrix $\mathcal{H}_{k+\ell}(\hat{p})$ has rank k.

We run two different numerical experiments (whose codes are available at [12]): in both of them we fix the noise level $\sigma = 0.1$ and we consider time series of increasing length, but the involved matrices have different dimensions (hence the rank reductions are different since the system order is 5 in both experiments): we consider $\ell = 2$ in the first experiment and $\ell = 5$ in the second.

All the results are the average over ten different perturbations on the same time series. We discarded from the simulations all the runs where at least one of the methods needed more than two minutes to get a solution. The outcomes of the numerical experiments are in Tables 2 and 3, respectively.

First of all, we compare the first two columns to observe the performance of the FFT-based method for the projection of the rank-one matrix (implemented in Hank-svds-fft) with respect to the antidiagonal average [4, Lemma 2] used in Hank-svds. Differently from Table 1, all the computational times are now very close, and Hank-svds-fft is slightly faster than Hank-svds in Table 2 but not in Table 3. This is probably due to the fact that the size of the matrices increases in one dimension only (the number of columns).

Looking at all four columns, Hank-rsvd has the smallest computational times in both Table 2 and Table 3. The computational times of Hank-svds and Hank-svds-fft are close to (and most of the times slightly smaller than) the ones of slra. Regarding the relative errors in the computed solutions, there are no significant differences among all the computational methods and the two tables.

5 Possible extensions

We saw how the considered algorithm works on the problems of Sylvester and Hankel low-rank approximation, but a larger class of (structured) low-rank approximation problems can be approached by the same method. We describe now two possible extensions that are of interest in an application setting: the identification of time series

[1] The function slra can only deal with problems where the rank reduction is one. To deal with rank reductions greater than one, we have to appropriately weight the entries of p.

Table 2 Low-rank approximation of Hankel matrices: rank reduction by 2. Comparison of different implementation techniques for time series of increasing length

length(p) size(S(p))		Hank-svds	Hank-svds-fft	Hank-rsvd	slra
51	time (sec)	0.5048	0.4948	0.2040	0.5566
	——	——	——	——	——
7×45	rel. err	0.0968	0.0968	0.0970	0.0916
101	time (sec)	2.2634	2.2568	0.9745	2.3330
	——	——	——	——	——
7×95	rel. err	0.0953	0.0953	0.0956	0.0931
151	time (sec)	15.1904	13.7776	6.3698	15.3719
	——	——	——	——	——
7×145	rel. err	0.1010	0.1010	0.1023	0.0958
201	time (sec)	5.9258	5.8425	2.4479	6.0902
	——	——	——	——	——
7×195	rel. err	0.0981	0.0981	0.0982	0.0967
251	time (sec)	4.9840	4.2523	2.4723	5.0937
	——	——	——	——	——
7×245	rel. err	0.0993	0.0993	0.1009	0.0975
301	time (sec)	16.6570	15.6608	8.8651	16.9055
	——	——	——	——	——
7×295	rel. err	0.0979	0.0979	0.1094	0.0975

with missing coefficients and the low-rank approximation of block-structured matrices. On the other hand, it has already been described in [3] how to deal with complex coefficients and how to fix some entries of the starting parameter vector.

5.1 Time series with missing coefficients

The considered problem is the identification of a linear time-invariant system (that is, low-rank approximation of a rectangular Hankel matrix) with the additional assumption that some entries of the parameter vector p are unknown. This is achieved by choosing the weights as in Example 1, but by setting any weight for missing observations to 0. The missing values are initialized by averaging their neighbors.

The following simulation shows the numerical performance of the proposed approach. The results are compared with the algorithm in [7] on the following numerical example. The true (noiseless) signal is the sum of two exponentially modulated cosines:

$$\begin{aligned} y_0(t) &= y_{0,1}(t) + y_{0,2}(t) \\ y_{0,1}(t) &= 0.9^t \cos\left(\frac{\pi}{5}t\right) \\ y_{0,2}(t) &= \frac{1}{5}1.05^t \cos\left(\frac{\pi}{12}t + \frac{\pi}{4}\right) \end{aligned} \tag{5}$$

Table 3 Low-rank approximation of Hankel matrices: rank reduction by 5. Comparison of different implementation techniques for time series of increasing length

length(p) size(S(p))		Hank-svds	Hank-svds-fft	Hank-rsvd	slra
51	time (sec)	0.5238	0.4955	0.2693	0.5775
	——	——	——	——	——
10×42	rel. err	0.0867	0.0867	0.0883	0.0893
101	time (sec)	1.8284	1.8334	0.9495	1.9196
	——	——	——	——	——
10×92	rel. err	0.0932	0.0932	0.0942	0.0920
151	time (sec)	1.8569	1.8585	1.0527	1.9693
	——	——	——	——	——
10×142	rel. err	0.0966	0.0966	0.0972	0.0960
201	time (sec)	7.3412	6.2034	3.9425	7.5193
	——	——	——	——	——
10×192	rel. err	0.1008	0.1008	0.1017	0.0984
251	time (sec)	10.6441	10.1989	7.3877	10.8229
	——	——	——	——	——
10×242	rel. err	0.0989	0.0989	0.1055	0.0970
301	time (sec)	17.1525	17.4631	11.7912	17.3553
	——	——	——	——	——
10×292	rel. err	0.0994	0.0994	0.0996	0.0987

for $t = 1, \ldots, 50$. The rank of any rectangular Hankel matrix (having more than 4 rows) is $k = 4$, and in this example, we choose $\mathcal{H}_9(y_0(t))$. The data for the numerical experiment are built by adding a random perturbation as follows

$$y(t) = y_0(t) + 0.2\frac{e(t)}{\|e(t)\|_2}\|y_0(t)\|_2 \tag{6}$$

where the entries of the vector $e(t)$ come from independent standard normal distributions. Starting from the (full rank) noisy Hankel matrix $\mathcal{H}_9(y(t))$, we want to compute a rank 4 approximation under the assumption that some of the coefficients are unknown (missing). In particular, as stated in [7], since the system generating the signal (5) has order 4, standard system identification algorithms need at least 5 consecutive data points. To make the problem harder, we remove every fifth data point in the time series. In Fig. 1 we plot the true (unknown) signal and two different reconstructions from the available data: the first comes from Hank-rsvd, while the second from the algorithm in [7] (labeled as slra-reg[2]) starting from the matrix $\mathcal{H}_9(y(t))$.

As we can see from Fig. 1, the observed results are similar to the ones in Tables 2 and 3. The distances between the computed signals and the true one are both very

[2] slra-reg can deal with rank reductions greater than one, differently from the function *slra* used in Sect. 4.

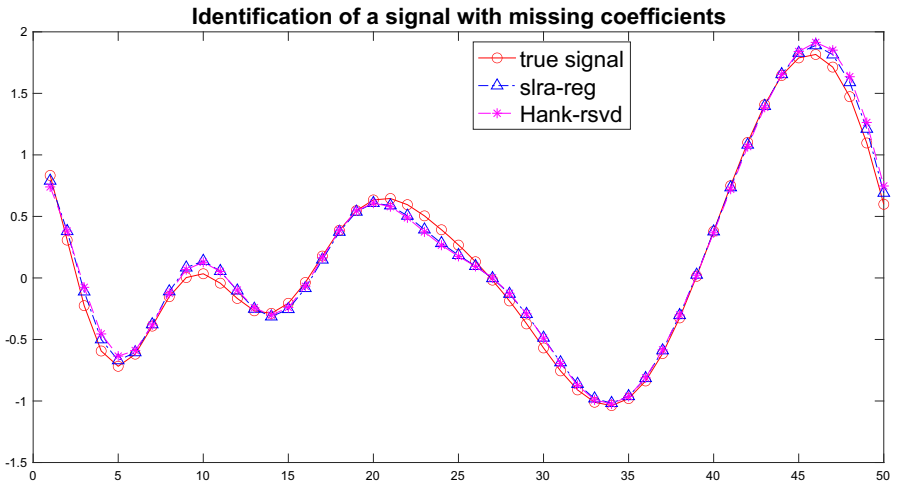

Fig. 1 Identification of a time series with missing coefficients: comparison between two different approaches

small, that is the errors have the same order of magnitude, and the computational time of slra-reg is slightly higher than Hank-rsvd (despite the solution method of slra-reg being different than slra, see [7]).

5.2 Extension to block structured matrices

The Sylvester and Hankel low-rank approximation problems can be extended to more general block matrices whose blocks have Toeplitz/Hankel structure.

The first application of this problem is the identification of linear time-invariant systems which are not autonomous, but depend on both inputs and outputs. In this case, each entry of the Hankel matrix (1) is a vector built from inputs and outputs. The formulation of the problem is similar, that is we need to compute a low-rank approximation of a block-Hankel matrix [1] with a possible rank reduction greater than one. The value of the rank is associated with the dimension of the corresponding system (see, for example, [14]).

Hankel matrices can be extended to mosaic Hankel, which are block matrices whose blocks are rectangular Hankel matrices stacked in a row [6]

$$K = [\mathcal{H}_m(p_1), \cdots, \mathcal{H}_m(p_N)].$$

Such structure arises, e.g., in the problem of common dynamics estimation, which can be restated as a generalized structured low-rank approximation problem with multiple rank constraints [17]. The proposed algorithm can also be adapted to reduce the rank of the matrix K: this can be done by perturbing all the time series $p_1, \ldots, p_N$ to reduce the rank of each block $\mathcal{H}_m(p_i)$ and by imposing the rank reduction on the matrix K at the same time. In detail, we first compute the needed singular triplet σ, u, v of the matrix K, then we split the vector v in N blocks $v_1, \ldots, v_N$, according to the

column dimensions of the different blocks of K. The (sub-optimal) perturbations of the different time series $p_1, \ldots, p_N$ which decrease the singular value σ of K come from the N different projections of the corresponding rank-one matrices:

$$\left[-\mathcal{P}(uv_1^T),\ \ldots,\ -\mathcal{P}(uv_N^T)\right].$$

This algorithm has been already proposed for the common dynamics estimation problem [18].

The proposed perturbation strategy on different time series can allow us to deal with structured low-rank approximation problems involving different block matrices whose blocks are not necessarily stacked in one row/column.

6 Conclusion and possible future work

We analyzed some possible computational improvements and extensions of a recent algorithm for solving structured low-rank approximation problems involving Sylvester and Hankel matrices. Concerning this, we proposed some numerical techniques to speed up two main computations in the recursive integration of some systems of ODEs: the numerical computation of a given singular triplet and the orthogonal projection of a rank-one matrix onto the set of Toeplitz/Hankel matrices. The identification of a time series with missing coefficients and the structured low-rank approximation of block matrices are discussed as possible extensions of the same methodology.

We remark that we intentionally did not change the ideas behind the algorithms proposed in the previous works. A possible improvement could be to change the initial estimates in the gradient systems in order to reduce the number of iterations. Moreover, one can try to modify the size of the Hankel matrices to possibly improve the estimate of the computed solution (see [19]).

Alternative possible improvements are related to the choice of the integration scheme. We list, as examples, higher order Runge–Kutta, implicit methods or low-rank integrators.

Acknowledgements The authors thank an anonymous reviewer for his comments and suggestions that improved the quality of the paper.

Funding The research leading to these results has received funding from the Catalan Institution for Research and Advanced Studies (ICREA), the Fond for Scientific Research Vlaanderen (FWO) projects G090117N and G033822N and the FNRS-FWO under Excellence of Science (EOS) Project no 30468160.The first author is member of the Gruppo Nazionale Calcolo Scientifico-Istituto Nazionale di Alta Matematica (GNCS-INdAM).

Data availability All data generated or analysed during this study are included in this published article.

Declarations

Conflict of interest The authors declare no Conflict of interest.

References

1. Markovsky, I.: Low-Rank Approximation, 2nd edn. Springer, Cham, Switzerland (2019)
2. Guglielmi, N., Markovsky, I.: An ODE based method for computing the distance of coprime polynomials to common divisibility. SIAM J. Numer. Anal. **55**, 1456–1482 (2017)
3. Fazzi, A., Guglielmi, N., Markovsky, I.: An ODE based method for computing the Approximate Greatest Common Divisor of polynomials. Numer. Algorithms **81**, 719–740 (2019). https://doi.org/10.1007/s11075-018-0569-0
4. Fazzi, A., Guglielmi, N., Markovsky, I.: A gradient system approach for Hankel structured low-rank approximation. Linear Algebra. appl. **623**, 236–257 (2021). https://doi.org/10.1016/j.laa.2020.11.016
5. Markovsky, I., Fazzi, A., Guglielmi, N.: Applications of polynomial common factor computation in signal processing. In: et al., Y.D. (ed) Latent Variable Analysis and Signal Separation. Lecture Notes in Computer Science, Springer, Guildford, UK (2018).
6. Usevich, K., Markovsky, I.: Variable projection for affinely structured low-rank approximation in weighted 2-norms. J. Comput. Appl. Math. **272**, 430–448 (2014)
7. Ishteva, M., Usevich, K., Markovsky, I.: Factorization Approach to Structured Low-Rank Approximation with Applications. SIAM J. Matrix Anal. Appl. **35**, 1180–1204 (2014)
8. Usevich, K., Markovsky, I.: Variable projection methods for approximate (greatest) common divisor computations. Theoret. Comput. Sci. **681**, 176–198 (2017)
9. Sylvester, J.J.: On a theory of the syzygetic relations of two rational integral functions, comprising an application to the theory of Sturm's functions, and that of the greatest algebraical common measure. Philosophical Trans. **143**, 407–548 (1853)
10. Korobeynikov, A.: Computation- and space-efficient implementation of ssa. Stat. Interface **3**, 357–368 (2009)
11. Schwetlick, H., Schnabel, U.: Iterative computation of the smallest singular value and the corresponding singular vectors of a matrix. Linear Algebra Its Appl. **371**, 1–30 (2003)
12. Fazzi, A., Markovsky, I., Usevich, K.: Matlab codes for the experiments in "Implementation Improvements and extensions of an ODE-based algorithm for structured low-rank approximation.
13. Fazzi, A., Guglielmi, N., Markovsky, I.: Generalized algorithms for the approximate matrix polynomial GCD of reducing data uncertainties with application to MIMO system and control. J. Comput. Appl. Math. (2021). https://doi.org/10.1016/j.cam.2021.113499
14. Markovsky, I., Willems, J.C., Van Huffel, S., De Moor, B.: Exact and Approximate Modeling of Linear Systems: A Behavioral Approach. SIAM, Philadelphia (2006)
15. Halko, N., Martinsson, P.G., Tropp, J.A.: Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions. SIAM Rev. **53**, 217–288 (2011)
16. Liutkus, A.: Randomized Singular Value Decomposition https://nl.mathworks.com/matlabcentral/fileexchange/47835-randomized-singular-value-decomposition, MATLAB Central File Exchange. Retrieved in 2023-03-01
17. Markovsky, I., Liu, T., Takeda, A.: Subspace methods for multi-channel sum-of-exponentials common dynamics estimation. In: Proc. of the IEEE Conf. on Decision and Control, Nice, France, 2672–2675 (2019).
18. Fazzi, A., Guglielmi, N., Markovsky, I., Usevich, K.: Common dynamic estimation via structured low-rank approximation with multiple rank constraints. IFAC - Papers Online **54**, 103–107 (2021)
19. Usevich, K.: Improved initial approximation for errors-in-variables system identification. In: Proc. of 20th Mediterranean Conference on Control and Automation, pp. 198–203 (2012)