Structured low-rank approximation with missing data

Ivan Markovsky and Konstantin Usevich

Department ELEC
Vrije Universiteit Brussel
{imarkovs,kusevich}@vub.ac.be

Abstract

We consider low-rank approximation of affinely structured matrices with missing elements. The method proposed is based on reformulation of the problem as inner and outer optimization. The inner minimization is a singular linear least-norm problem and admits an analytic solution. The outer problem is a nonlinear least squares problem and is solved by local optimization methods: minimization subject to quadratic equality constraints and unconstrained minimization with regularized cost function. The method is generalized to weighted low-rank approximation with missing values and is illustrated on approximate low-rank matrix completion, system identification, and data-driven simulation problems. An extended version of the paper is a literate program, implementing the method and reproducing the presented results.

Keywords: low-rank approximation, missing data, variable projections, system identification, approximate matrix completion.

AMS subject classifications: 93A30, 93B30, 93B40, 37M10, 41A29, 62J12, 49N30, 15A83.

1 Introduction

The paper describes a solution method for matrix structured low-rank approximation, *i.e.*, approximation of a given matrix by another matrix whose elements satisfy certain predefined relations (matrix structure) and whose rank is less than or equal to a predefined value. The combination of matrix and low-rank structure makes structured low-rank approximation a tool for data modeling. Low-rank property of a matrix is equivalent to existence of an exact low-complexity linear model for the data. Moreover, the rank of the matrix is related to the complexity of the model. The structure, imposed on the approximation, is related to properties of the model. For example, Hankel structure corresponds to time-invariance of a linear dynamical model for the data. A tutorial exposition of the subject with an emphasis on applications in systems theory, control, and signal processing, is given in [12, 14].

A novel feature of the low-rank approximation problem, considered in this paper, is that elements of the data matrix can be missing (not specified). Missing data may occur in practical applications due to malfunctioning of measurement device, communication channel, or storing device. In such cases, the most common strategy is to collect a complete data record by repeating the data collection experiment. In other applications, however, the missing data problem is intrinsic and can not be avoided by repeated experiments. Two such applications reviewed in Subsection 1.2 and further illustrated by numerical examples in Section 5 are recommender systems and data-driven simulation.

Although structured low-rank approximation and approximate low-rank matrix completion (missing data estimation in low-rank matrices) are independently active research topics, the combined problem of missing data estimation in affine structured low-rank matrices has not been considered before. Both the structured low-rank approximation and approximate matrix completion problems are nonconvex optimization problems that in general admit no analytic solution. Therefore, in both domains local optimization and convex relaxation heuristics are used as solution techniques. In this paper, we use the local optimization approach.

Structured low-rank approximation has been studied in the literature from different viewpoints: numerical algorithm for computing locally optimal or suboptimal solutions, statistical properties of the resulting estimators, and applications. From a numerical point of view, the main challenge is to achieve fast and robust computational methods that can deal effectively with large data sets. From a statistical point of view, the main challenge is to establish conditions for consistency and efficiency of the methods. Our objective in this paper is different: we aim to unify as many data modeling applications as possible and derive a single algorithm that solves them. Of course, this goal can be achieved by brute force optimization. The challenge is to discover and use effectively the structure of the problem. In the general problem considered, this structure is a separation of variables with analytic solution over one set of variables. This approach is related to the variable projections method [8] used in [18].

The general solution method proposed in the paper has computational complexity $O(n^3)$ per iteration, where n is the number of columns of the data matrix. This makes it unsuitable for large scale applications. In special cases, such as unstructured matrix with missing data and Hankel structured matrix, there are efficient $O(n)$ methods. Modification of the methods in the paper for efficient computation in the case of mosaic-Hankel [9] matrices, is possible and will be presented elsewhere.

1.1 Problem formulation

We denote missing data values by the symbol `NaN` ("not a number"). The considered low-rank approximation problem is: Given a data vector $p \in (\mathbb{R} \cup \{\texttt{NaN}\})^{n_p}$,

$$\begin{aligned} &\text{minimize} \quad \text{over } \widehat{p} \in \mathbb{R}^{n_p} \sum_{\{i \mid p_i \neq \texttt{NaN}\}} (p_i - \widehat{p}_i)^2 \\ &\text{subject to} \quad \operatorname{rank}\big(\mathscr{S}(\widehat{p})\big) \leq r, \end{aligned} \tag{SLRA}$$

where

$$\mathscr{S} : \mathbb{R}^{n_p} \to \mathbb{R}^{m \times n}, \qquad \text{defined by} \quad \mathscr{S}(\widehat{p}) = S_0 + \sum_{i=1}^{n_p} S_i \widehat{p}_i, \tag{$\mathscr{S}$}$$

2 ⟨*default* s0 2⟩≡ (6d 10d)
```
  if ~exist('s0') || isempty(s0), s0 = zeros(mp, n); end
```

is the matrix structure—an affine function from the structure parameter space $\mathbb{R}^{n_p}$ to the set of matrices $\mathbb{R}^{m \times n}$. With $\mathscr{G}$ denoting the vector of indices of the given values $\{i \mid p_i \neq \texttt{NaN}\}$ and $p_{\mathscr{G}}$ denoting the subvector of p with indices in $\mathscr{G}$, the approximation criterion can be written as

$$\sum_{i \in \mathscr{G}} (p_i - \widehat{p}_i)^2 = \|p_{\mathscr{G}} - \widehat{p}_{\mathscr{G}}\|_2^2.$$

Without loss of generality, we assume throughout the paper that $r < m \leq n$. Using the kernel representation of the rank constraint

$$\operatorname{rank}\big(\mathscr{S}(\widehat{p})\big) \leq r \quad \iff \quad \text{there is } R \in \mathbb{R}^{(m-r)\times m}, \text{ such that } R\mathscr{S}(\widehat{p}) = 0 \text{ and } R \text{ has full row rank}, \tag{KER}$$

the following equivalent problem to (SLRA) is obtained

$$\begin{aligned} &\text{minimize} \quad \text{over } \widehat{p} \in \mathbb{R}^{n_p} \text{ and } R \in \mathbb{R}^{(m-r)\times m} \quad \|p_{\mathscr{G}} - \widehat{p}_{\mathscr{G}}\|_2^2 \\ &\text{subject to} \quad R\mathscr{S}(\widehat{p}) = 0 \quad \text{and} \quad R \text{ has full row rank.} \end{aligned} \tag{SLRA$_R$}$$

Problem (SLRA$_R$) is a double minimization over the parameters R and $\widehat{p}$

$$\text{minimize} \quad \text{over } R \in \mathbb{R}^{(m-r)\times m} \quad M(R) \quad \text{subject to} \quad R \text{ has full row rank}, \tag{SLRA$'_R$}$$

where

$$M(R) := \min_{\widehat{p}} \|p_{\mathscr{G}} - \widehat{p}_{\mathscr{G}}\|_2^2 \quad \text{subject to} \quad R\mathscr{S}(\widehat{p}) = 0. \tag{INNER}$$

The evaluation of the cost function M, *i.e.*, solving (INNER) for a given value of R, is refered to as the *inner minimization problem*. This problem is solved analytically in Section 2. The remaining problem of minimizing M over R is refered to as the *outer minimization problem*. It is a nonlinear least-squares problem, which, in general, admits no analytic solution. General purpose local optimization methods are used in Section 3 for its numerical solution. In Section 4, the approach is generalized to weighted 2-norm approximation criteria with missing values. Numerical examples of solving approximation problems with missing data by the proposed methods are shown in Section 5.

1.2 Applications

1.2.1 Linear static modeling with missing data: An approximate low-rank matrix completion problem

Consider a set of vectors $\mathscr{D} = \{ d_1, \dots, d_N \}$ in $\mathbb{R}^q$. A linear model $\mathscr{B}$ for the data $\mathscr{D}$ is a subspace of the data space $\mathbb{R}^q$, and the dimension of $\mathscr{B}$ is a measure of the model's complexity. Linear static modeling is the problem of finding a low-complexity model (low-dimensional subspaces) that fits the data as close as possible. Existence of an exact linear model $\mathscr{B}$ for the data $\mathscr{D}$, *i.e.*, $\mathscr{D} \subset \mathscr{B}$, with complexity at most r is equivalent to the data matrix $D = \begin{bmatrix} d_1 & \cdots & d_N \end{bmatrix}$ having rank at most r. Therefore, measuring the fit between the data point d_i and the model $\mathscr{B}$ by the orthogonal distance

$$\operatorname{dist}(d_i, \mathscr{B}) = \min_{\widehat{d}_i \in \mathscr{B}} \|d_i - \widehat{d}_i\|_2,$$

the approximate fitting problem becomes a rank r matrix approximation problem (SLRA) with unstructured data matrix $\mathscr{S}(p) = D$.

Suppose now that some elements d_{ij}, $(i, j) \in \mathscr{I}_{\text{missing}}$ of the data matrix are missing. Equivalently, only the elements d_{ij}, $(i, j) \in \mathscr{I}_{\text{given}}$ of D are specified. The exact linear static modeling problem becomes a low-rank matrix completion problem [7]:

$$\text{find} \quad \widehat{D} \quad \text{such that} \quad \operatorname{rank}(\widehat{D}) \leq r \text{ and } \widehat{D}_{\mathscr{I}_{\text{given}}} = D_{\mathscr{I}_{\text{given}}}.$$

Here $D_{\mathscr{I}}$ denotes the vector of elements of D with indeces in $\mathscr{I}$. In the context of approximate data fitting by linear static model and missing data values, the relevant problem is approximate low-rank matrix completion [6]:

$$\text{minimize} \quad \text{over } \widehat{D} \quad \|\widehat{D}_{\mathscr{I}_{\text{given}}} - D_{\mathscr{I}_{\text{given}}}\|_2^2 \quad \text{subject to} \quad \operatorname{rank}(\widehat{D}) \leq r. \tag{AMC}$$

The approximate low-rank matrix completion problem (AMC) is used for building recommender systems. In recommender system applications, there is a set of users and a set of products. Some users rate some products. The goal is to predict the user ratings on products that they have not rated. The underlying assumption that makes the solution of this problem possible is that the full user-ratings matrix is low-rank. The low-rank property is observed empirically and can be explained intuitively as existence of a small number of groups of users with the same "taste" (*i.e.*, users that like or dislike the same products). In practice, the low-rank assumption is satisfied only approximately, which makes the approximation aspect of the problem essential.

The main issue in building real-life recommender systems is the high dimensionality and sparsity of the data matrix. Additional important issues in building practical recommender systems is the fact that the given and missing ratings are discrete and that apart from the users' ratings, there is demographic information about the users. Taking into account this prior information may improve significantly the accuracy of the missing values estimation. These issues, however, are outside the scope of the present paper.

1.2.2 System identification with missing data: An approximate block-Hankel structured low-rank matrix completion problem

A discrete-time linear time-invariant dynamical model is a set of time series

$$\mathscr{B}(R) := \big\{ w \mid R_0 w(t) + R_1 w(t+1) + \cdots + R_\ell w(t+\ell) = 0, \text{ for all } t \big\} \tag{$\mathscr{B}$}$$

that satisfy a constant coefficients difference equation. The matrices $R_0, R_1, \dots, R_\ell \in \mathbb{R}^{p \times q}$ are parameters specifying the model. Note that the linear static model is a special case of a linear time-invariant dynamical model when the lag ℓ of the difference equation representing the system is equal to zero.

A finite time series

$$w = \big(w(1), \dots, w(T)\big), \qquad \text{where} \quad w(t) \in \mathbb{R}^q,$$

is an exact trajectory of the system defined in ($\mathscr{B}$) if the following matrix equation is satisfied

$$\underbrace{\begin{bmatrix} R_0 & R_1 & \cdots & R_\ell \end{bmatrix}}_{R} \underbrace{\begin{bmatrix} w(1) & w(2) & w(3) & \cdots & w(T-\ell) \\ w(2) & w(3) & \iddots & & w(T-\ell+1) \\ w(3) & \iddots & & & \vdots \\ \vdots & & & & \\ w(\ell+1) & w(\ell+2) & \cdots & & w(T) \end{bmatrix}}_{\mathscr{H}_{\ell+1}(w)} = 0.$$

Without loss of generality, we assume that the parameter matrix R has full row rank $\mathtt{p}$, which implies that

$$\operatorname{rank}\big(\mathscr{H}_{\ell+1}(w)\big) \leq q(\ell+1) - \mathtt{p}.$$

We showed above that the data w is an exact trajectory of a system $\mathscr{B}(R)$, if the block-Hankel matrix $\mathscr{H}_{\ell+1}(w)$ is rank deficient. Therefore, as in the static case, approximate modeling by a linear time-invariant system is a low-rank approximation problem

$$\text{minimize} \quad \text{over } \widehat{w} \quad \|w - \widehat{w}\|_2^2 \quad \text{subject to} \quad \operatorname{rank}\big(\mathscr{H}_{\ell+1}(\widehat{w})\big) \leq q(\ell+1) - \mathtt{p}. \qquad \text{(SYSID)}$$

Note, however, that the linear time-invariant model class imposes a block-Hankel structure constraint on the approximation matrix.

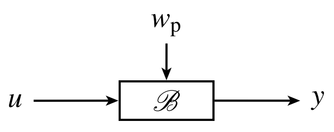
4a ⟨*structure specification for identification* 4a⟩≡ (20a)
```
  s.m = (ell + 1) * ones(q, 1); s.n = T - ell; r = ell * q + m;
```

4b ⟨*weights for identification with missing values* 4b⟩≡ (20a)
```
  s.w = double(~isnan(w(:)));
```

Identification from a trajectory with missing elements is therefore a block-Hankel structured low-rank matrix completion problem. A special system identification problems for the class of auto-regressive exogenous systems with missing data is considered in [10] and a method based on frequency domain techniques is proposed in [24]. These papers do not link the system identification problem to the block-Hankel low-rank approximation problem (SYSID), so that their approaches are different from ours. The method developed in this paper, when specialized to block-Hankel structure can be used for general multivariable system identification in the time domain.

1.2.3 Data-driven simulation and control

The trajectory w of a dynamical system can be partitioned into inputs u, *i.e.*, free variables, and outputs y, *i.e.*, variables that are determined by the inputs, initial conditions, and the model. Let $w = (u, y)$ be such a partition. The output $y = \big(y(1), \dots, y(T)\big)$ of $\mathscr{B}$ is uniquely determined by the input $u = \big(u(1), \dots, u(T)\big)$ and the initial conditions



$$w_\mathrm{p} = \big(w(-\ell+1), w(-\ell+1), \dots, w(0)\big).$$

This gives a "signal processor" interpretation of a dynamical system.

The simulation problem aims to find the output y_f of a system $\mathscr{B}$, corresponding to a given input u_f and initial conditions w_p, *i.e.*,

$$\text{find} \quad y_\mathrm{f} \quad \text{such that} \quad w_\mathrm{p} \wedge (u_\mathrm{f}, y_\mathrm{f}) \in \mathscr{B}.$$

($w_\mathrm{p} \wedge w_\mathrm{f}$ denotes the concatenation of w_p and w_f.) This is a classic problem in system theory and numerical linear algebra, for which many solutions exist, *e.g.*, for systems with no inputs, the problem is related to the computation of the matrix exponential [22]. The classical simulation methods require a representation (state space, transfer function, convolution kernel, *etc.*) of the model. Such a representation is often obtained from data by system identification. The question occurs of solving the simulation problem directly from the data without identifying a model representation as a byproduct and using it in a model based solution. We call the direct problem data-driven simulation [15].

The data-driven simulation problem is a mosaic-Hankel structured low-rank approximation problem with fixed (exact) and missing data. To see this, let w' denotes the data that implicitly specifies the model and $w'' = w''_{\mathrm{p}} \wedge (u''_{\mathrm{f}}, y''_{\mathrm{f}})$ be the to-be-simulated trajectory. We express the condition that w' and w'' are trajectories of a linear time-invariant system with lag ℓ in matrix language as

$$\operatorname{rank}\left(\begin{bmatrix} \mathscr{H}_{\ell+1}(w') & \mathscr{H}_{\ell+1}(w'') \end{bmatrix} \right) \leq q(\ell+1) - \mathtt{p}.$$

This is a model-free description of the simulation problem—the existence of a model is implicit in the rank constraint. The fixed data are the initial conditions w''_{p} and the input u''_{f} and the missing data are the to-be computed response y''_{p}. When the data w' is not an exact trajectory of the model, the matrix $\mathscr{H}_{\ell+1}(w')$ is generically full rank, so that an approximation is needed. The resulting data-driven simulation problem is a mosaic-Hankel structured low-rank approximation with missing data:

$$\begin{array}{ll} \text{minimize} & \text{over } \widehat{w}' \text{ and } \widehat{w}'' \quad \|w' - \widehat{w}'\|_2^2 \\ \text{subject to} & \operatorname{rank}\left(\begin{bmatrix} \mathscr{H}_{\ell+1}(\widehat{w}') & \mathscr{H}_{\ell+1}(\widehat{w}'') \end{bmatrix} \right) \leq q(\ell-1) - \mathtt{p}, \\ & \widehat{w}''_{\mathrm{p}} = w''_{\mathrm{p}}, \quad \text{and} \quad \widehat{u}''_{\mathrm{f}} = u''_{\mathrm{f}}. \end{array} \tag{DDSIM}$$

5a ⟨*structure specification for data-driven simulation* `slra-m` 5a⟩≡ (20d 21a)
```
tts = [blkhank(reshape(1:(T1 * q), q, T1), ell + 1) ...
       blkhank(reshape(T1 * q + (1:(T2 * q)), q , T2), ell + 1)];
w = [ones(q * T1, 1); inf * vec(~isnan(w2'))]; r = ell * q + m;
```

5b ⟨*structure specification for data-driven simulation* `slra-c` 5b⟩≡ (21b)
```
s.m = [ell + 1 ell + 1]; s.n = [T1 T2] - ell; r = ell * q + m;
s.w = [ones(q * T1, 1); inf * vec(~isnan(w2))];
```

5c ⟨*extract* $\widehat{h}$ *from* $\widehat{w}''$ `slra-m` 5c⟩≡ (20d 21a)
```
wh2 = reshape(wh(T1 * q + 1:end), q, T2); hh = wh2(m + 1:end, (ell + 1):end)';
```

5d ⟨*extract* $\widehat{h}$ *from* $\widehat{w}''$ `slra-c` 5d⟩≡ (21b)
```
wh2 = reshape(wh(T1 * q + 1:end), T2, q)'; hh = wh2(m + 1:end, (ell + 1):end)';
```

Notation

In the rest of the paper, we use the following notation.

- $A_{\mathscr{I},\mathscr{J}}$ is the submatrix of A with rows in $\mathscr{I}$ and columns in $\mathscr{J}$. The row/column index can be replaced by the symbol ":", in which case all rows/columns are selected.
- $\mathscr{M}$ / $\mathscr{G}$ is the vector of indices of p that are missing / given, and n_{m} / n_{g} is the number of missing / given elements.

5e ⟨*define* $\mathscr{M}$ *and* $\mathscr{G}$ 5e⟩≡ (6d 10c 13a)
```
Im = find(isnan(p)); Ig = setdiff(1:np, Im);
```

- A^{+} is the pseudo inverse of A and $A^{\perp}$ is a matrix which rows form a basis for the left null space of A.

5f ⟨*define* `perp` 5f⟩≡ (6b)
```
perp = @(a) null(a')';
```

- $\operatorname{vec}(\cdot)$ is the column-wise vectorization operator.

2 Analytical solution of the inner minimization problem

In this section, we consider the inner minimization problem (INNER).

Problem 1. Given affine structure $\mathscr{S}$, structure parameter vector $p \in \{\mathbb{R} \cup \{\mathtt{NaN}\}\}^{n_p}$, and a kernel parameter $R \in \mathbb{R}^{(m-r)\times m}$, evaluate the cost function $M(R)$, defined in (INNER), and find a point $\widehat{p}$ that attains the minimum.

For a given structure $\mathscr{S}$ and $R \in \mathbb{R}^{(m-r)\times m}$, we define the matrix

$$G := \begin{bmatrix} \operatorname{vec}(RS_1) & \cdots & \operatorname{vec}(RS_{n_p}) \end{bmatrix} \in \mathbb{R}^{(m-r)n \times n_p}. \tag{G}$$

6a $\langle$*define G* 6a$\rangle\equiv$ (6d)
```
  g = reshape(R * phi * reshape(bfs, mp, n * np), size(R, 1) * n, np);
```
(`reshape(bfs, mp, n * np)`: $\mathbf{S} \mapsto \begin{bmatrix} S_1 & \cdots & S_{n_p} \end{bmatrix}$)

Theorem 2. *Under the following assumptions:*

1. $G_{:,\mathscr{M}}$ *is full column rank,*
2. $1 \leq (m-r)n - n_{\mathrm{m}} \leq n_g$*, and*
3. $\bar{G} := G^{\perp}_{:,\mathscr{M}} G_{:,\mathscr{G}}$ *is full row rank,*

6b $\langle$*define* $\bar{G}$ 6b$\rangle\equiv$ (6d)
```
  ⟨define perp 5f⟩, perp_gm = perp(g(:, Im)); bg = perp_gm * g(:, Ig);
```
Problem 1 has a unique global minimum

$$\widehat{p}_{\mathscr{G}} = p_{\mathscr{G}} - \bar{G}^{\top}\left(\bar{G}\bar{G}^{\top}\right)^{-1} s \qquad \text{and} \qquad \widehat{p}_{\mathscr{M}} = -G^{+}_{:,\mathscr{M}}\left(\operatorname{vec}(RS_0) + G_{:,\mathscr{G}}\widehat{p}_{\mathscr{G}}\right), \tag{$\widehat{p}$}$$

with objective function value

$$M(R) = s^{\top}\left(\bar{G}\bar{G}^{\top}\right)^{-1} s, \qquad \text{where} \quad s := \bar{G}p_{\mathscr{G}} + G^{\perp}_{:,\mathscr{M}} \operatorname{vec}(RS_0). \tag{M}$$

6c $\langle$*compute M and* $\widehat{p}$ 6c$\rangle\equiv$ (6d)
```
  dpg = bg' * pinv(bg * bg') * (bg * p(Ig) + perp_gm * vec(R * phi * s0));
  M = dpg' * dpg; ph(Ig) = p(Ig) - dpg; ph = ph(:);
  ph(Im) = - pinv(g(:, Im)) * (vec(R * phi * s0) + g(:, Ig) * ph(Ig));
```

Proof. Defining

$$\Delta p_{\mathscr{G}} := p_{\mathscr{G}} - \widehat{p}_{\mathscr{G}}$$

and using the identity

$$R\mathscr{S}(\widehat{p}) = 0 \qquad \Longleftrightarrow \qquad G\widehat{p} = -\operatorname{vec}(RS_0),$$

we have

$$R\mathscr{S}(\widehat{p}) = 0 \qquad \Longleftrightarrow \qquad \begin{bmatrix} G_{:,\mathscr{G}} & G_{:,\mathscr{M}} \end{bmatrix} \begin{bmatrix} p_{\mathscr{G}} - \Delta p_{\mathscr{G}} \\ \widehat{p}_{\mathscr{M}} \end{bmatrix} = -\operatorname{vec}(RS_0).$$

Therefore, (INNER) is equivalent to

$$M(R) := \min_{\Delta p_{\mathscr{G}} \in \mathbb{R}^{n_g},\, \widehat{p}_{\mathscr{M}} \in \mathbb{R}^{n_{\mathrm{m}}}} \|\Delta p_{\mathscr{G}}\|_2^2 \quad \text{subject to} \quad \begin{bmatrix} G_{:,\mathscr{G}} & G_{:,\mathscr{M}} \end{bmatrix} \begin{bmatrix} \Delta p_{\mathscr{G}} \\ -\widehat{p}_{\mathscr{M}} \end{bmatrix} = G_{:,\mathscr{G}} p_{\mathscr{G}} + \operatorname{vec}(RS_0),$$

which is a generalized linear least norm problem. The solution follows from Lemma 3. □

6d $\langle$`misfit_ext` 6d$\rangle\equiv$
```
  function [M, ph] = misfit_ext(R, tts, p, w, bfs, phi, s0)
  ⟨S ↦ (m,n,n_p) 24a⟩
  ⟨default phi 24d⟩
  ⟨default s0 2⟩
  if ~exist('bfs') | isempty(bfs), ⟨S ↦ S 24b⟩, end
  ⟨define M and G 5e⟩
  ⟨preprocessing for weighted problem 12d⟩
  ⟨define G 6a⟩
  ⟨define Ḡ 6b⟩
  ⟨compute M and p̂ 6c⟩
  ⟨postprocessing for weighted problem 12e⟩
```

Generalized least norm problem

Lemma 3. *Consider the generalized linear least norm problem*

$$f = \min_{x,y} \quad \|x\|_2^2 \quad \textit{subject to} \quad Ax + By = c, \tag{GLN}$$

with $A \in \mathbb{R}^{m \times n_x}$, $B \in \mathbb{R}^{m \times n_y}$, and $c \in \mathbb{R}^m$. Under the following assumptions:

1. *B is full column rank,*
2. *$1 \leq m - n_y \leq n_x$, and*
3. *$\bar{A} := B^{\perp} A$ is full row rank,*

problem (GLN) has a unique solution

$$\begin{aligned} &f = c^{\top} (B^{\perp})^{\top} \left(\bar{A}\bar{A}^{\top}\right)^{-1} B^{\perp} c, \\ &x = \bar{A}^{\top} \left(\bar{A}\bar{A}^{\top}\right)^{-1} B^{\perp} c \qquad \textit{and} \qquad y = B^{+}(c - Ax). \end{aligned} \tag{SOL}$$

Proof. Under assumption 1, B has a nontrivial left kernel of dimension $m - n_y$. Therefore for the nonsingular matrix $T = \begin{bmatrix} B^+ \\ B^{\perp} \end{bmatrix} \in \mathbb{R}^{m \times m}$

$$TB = \begin{bmatrix} B^+ \\ B^{\perp} \end{bmatrix} B = \begin{bmatrix} B^+ B \\ B^{\perp} B \end{bmatrix} = \begin{bmatrix} I_{n_y} \\ 0 \end{bmatrix}.$$

Pre-multiplying both sides of the constraint of (GLN) by T, we have the following equivalent constraint

$$\begin{bmatrix} B^+ Ax \\ B^{\perp} Ax \end{bmatrix} + \begin{bmatrix} y \\ 0 \end{bmatrix} = \begin{bmatrix} B^+ c \\ B^{\perp} c \end{bmatrix}.$$

The first equation

$$y = B^+(c - Ax)$$

uniquely determines y, given x. The second equation

$$B^{\perp} Ax = B^{\perp} c \tag{$*$}$$

defines a linear constraint for x only. By assumption 2, it is an underdetermined system of linear equations. Therefore, (GLN) is equivalent to the following standard least norm problem

$$f = \min_{x} \ \|x\|_2^2 \quad \text{subject to} \quad B^{\perp} Ax = B^{\perp} c. \tag{GLN'}$$

By assumption 3 the solution is unique and is given by (SOL). ☐

Note 4 (About assumptions 1–3). Assumption 1 is a necessary condition for uniqueness of the solution. Relaxing assumption 1 implies that any vector in the affine space

$$\mathscr{Y} = B^+(c - Ax) + \operatorname{null}(B)$$

is a solution to (GLN). Assumption 2 ensures that the problem is a least norm problem and has a nontrivial solution. In the case $m = n_y$, the problem has a trivial solution $f = 0$. In the case $m - n_y > n_x$, the problem generically has no solution because the constraint $(*)$ is an overdetermined system of equations. Assumption 3 is also required for uniqueness of the solution. It can also be relaxed, making y nonunique.

Note 5 (Link to weighted least norm problems with singular weight matrix). Consider the weighted least norm problem

$$\min_z z^\top W z \quad \text{subject to} \quad Dz = c,$$

with singular positive semidefinite weight matrix W. Using a change of variables $\bar{z} = T^{-1}z$, where T is a nonsingular matrix, we obtain the equivalent problem

$$\min_z \bar{z}^\top T^\top W T \bar{z} \quad \text{subject to} \quad DT\bar{z} = c.$$

There exists a nonsingular matrix T, such that

$$T^\top W T = \begin{bmatrix} I_{n_x} & \\ & 0 \end{bmatrix}.$$

Partitioning $\bar{z}$ and $\bar{D} := DT^{-1}$ conformably as

$$\bar{z} = \begin{bmatrix} x \\ y \end{bmatrix} \qquad \text{and} \qquad \bar{D} = \begin{bmatrix} A & B \end{bmatrix}$$

we obtain problem (GLN).

3 Outer minimization problem

The outer minimization problem (SLRA$_R$) is a nonlinear least-squares problem, which we solve by general purpose local optimization methods. In order to apply standard optimization methods, however, we need first to replace the rank constraint with equivalent equality or inequality constraints.

The kernel parameter R is constrained to have a specified structure

$$\mathscr{R} : \mathbb{R}^{n_\theta} \to \mathbb{R}^{(m-r)\times m},$$

i.e., $R = \mathscr{R}(\theta)$, for some $\theta \in \mathbb{R}^{n_\theta}$. An example of a kernel structure $\mathscr{R}$ is a linear function

$$R = \mathscr{R}(\theta) := \operatorname{vec}^{-1}_{m-r}(\theta\Psi), \qquad (\theta \mapsto R)$$

defined by a matrix $\Psi \in \mathbb{R}^{n_\theta \times (m-r)m}$.

8a ⟨*default* `th2R` 8a⟩≡ (10d)
```
  if ~exist('th2R') || isempty(th2R), th2R = @(th) reshape(th * psi, m - r, m); end
```

8b ⟨*default* `psi` 8b⟩≡ (10d)
```
  if ~exist('psi', 'var') | isempty(psi), psi = eye(m * (m - r)); end
```

The full row rank constraint on R is equivalent to and can be enforced in the parameter optimization method by the equality constraint

$$RR^\top = I_{m-r}. \qquad \text{(f.r.r. } R\text{)}$$

$$\mathscr{C}(\theta) := \mathscr{R}(\theta)\mathscr{R}^\top(\theta) - I_{m-r} = 0. \qquad \text{(f.r.r. } R\text{)}$$

8c ⟨*default* $\mathscr{C}$ 8c⟩≡ (10d)
```
  if ~exist('C') || isempty(C), C = @(th) th2R(th) * th2R(th)' - eye(m - r); end
```

Then, the outer minimization problem becomes a constrained nonlinear least squares problems

$$\text{minimize} \quad \text{over } R \in \mathbb{R}^{(m-r)\times m} \quad M(R) \quad \text{subject to} \quad RR^\top - I_{m-r} = 0. \tag{SLRA$'_R$}$$

(SLRA$'_R$) is an optimization problem on a Stiefel manifold [2] and can be solved by specialized methods, *e.g.*, the GenRTR package [3], or by general purpose penalty methods for constrained optimization [23]. Next, we consider a penalty method, *i.e.*, reformulation of (SLRA$'_R$) as a regularized unconstrained nonlinear least squares problem by adding the regularization term $\gamma\|RR^\top - I_{m-r}\|_\text{F}^2$, where $\|\cdot\|_\text{F}$ is the Frobenius norm, to the cost function

$$\text{minimize} \quad \text{over } R \in \mathbb{R}^{(m-r)\times m} \quad M(R) + \gamma\|RR^\top - I_{m-r}\|_\text{F}^2. \tag{SLRA$''_R$}$$

The parameter γ should be chosen "large enough" in order to enforce the constraint (f.r.r. R). A corollary of the following theorem shows that $\gamma = \|p_\mathscr{G}\|_2^2$ is sufficiently large for linear structures.

9 $\langle$*define* γ 9$\rangle\equiv$ (10c)
```
  if ~exist('opt') || ~isfield(opt, 'g') || isempty(opt.g), opt.g = norm(p(Ig)) ^ 2; end
```

Theorem 6. *Let $M : \mathbb{R}^{(m-r)\times m} \to \mathbb{R}_+$ be a homogeneous function,* i.e.*, $M(R) = M(TR)$, for any R and a nonsingular $m \times m$ matrix T. Assume that γ satisfies*

$$\gamma > \min_{\{R \mid \operatorname{rank}(R) = m-r\}} M(R). \tag{γ}$$

Then, the optimal solutions of problem (SLRA$''_R$) coincide with the optimal solutions of (SLRA$'_R$).

Proof. We call a set $\mathscr{R} \subset \mathbb{R}^{d\times m}$ a "homogeneous set" if for all $R \in \mathscr{R}$ and for all nonsingular matrices $T \in \mathbb{R}^{d\times d}$, $TR \in \mathscr{R}$. Let R be a solution to (SLRA$''_R$) with the constraint $R \in \mathscr{R}$, where $\mathscr{R}$ is a homogeneous set. We will show that

$$\|RR^\top - I_{m-r}\|_\text{F}^2 = m - r - \operatorname{rank}(R). \tag{$*$}$$

There exists an orthogonal matrix U diagonalizing $RR^\top$. We have,

$$\begin{aligned}\|RR^\top - I_{m-r}\|_\text{F}^2 &= \|URR^\top U^\top - I_{m-r}\|_\text{F}^2 \\ &= \|\operatorname{diag}(a_1,\dots,a_{\operatorname{rank}(R)},0,\dots,0) - I_{m-r}\|_\text{F}^2, \qquad \text{where } a_i > 0 \\ &= \sum_{i=1}^{\operatorname{rank}(R)} (a_i - 1)^2 + m - r - \operatorname{rank}(R).\end{aligned}$$

Suppose that $a_i \neq 1$ for some i. The matrix

$$R' = \operatorname{diag}(1,\dots,1,1/\sqrt{a_i},1,\dots,1)R$$

has the same row span and rank as R, so that by the homogeneity property of M, $M(R) = M(R')$. However, we have

$$\|RR^\top - I_{m-r}\|_\text{F}^2 > \|R'R'^\top - I_{m-r}\|_\text{F}^2,$$

so that $R' \in \mathscr{R}$ achieves smaller value of the cost function of (SLRA$''_R$) than R. This is a contradiction. Therefore, $a_i = 1$ for all i. This concludes the proof of $(*)$.

So far we showed that minimization of the cost function in (SLRA$''_R$) on homogeneous sets is equivalent to minimization of

$$M(R) + \gamma\big(m - r - \operatorname{rank}(R)\big). \tag{M''}$$

The set of full row rank matrices

$$\mathscr{R}_f := \{R \in \mathbb{R}^{(m-r)\times m} \mid \operatorname{rank}(R) = m - r\}$$

and the set of rank-deficient matrices

$$\mathscr{R}_d := \{R \in \mathbb{R}^{(m-r)\times m} \mid \operatorname{rank}(R) < m - r\}$$

are homogeneous. Denote the solutions of (SLRA$''_R$) on these sets as

$$M_f^* := \inf_{R\in\mathscr{R}_f} M(R) + \gamma\|RR^\top - I_{m-r}\|_{\mathrm{F}}^2 \overset{(*)}{=} \inf_{R\in\mathscr{R}_f} M(R) < \gamma,$$

$$M_d^* := \inf_{R\in\mathscr{R}_d} M(R) + \gamma\|RR^\top - I_{m-r}\|_{\mathrm{F}}^2 \overset{(*)}{=} \inf_{R\in\mathscr{R}_d} \underbrace{M(R)}_{\geq 0} + \underbrace{\gamma\big(m - r - \operatorname{rank}(R)\big)}_{\geq\gamma}.$$

Then, $M_f^* < \gamma \leq M_d^*$ and

$$M^* := \inf_{R\in\mathbb{R}^{(m-r)\times m}} M(R) + \gamma\|RR^\top - I_{m-r}\|_{\mathrm{F}}^2 = M_f^*.$$

In addition, the minimum of (SLRA$'_R$) coincides with M_f^* by the homogeneity of M. Therefore, the solutions of (SLRA$''_R$) and (SLRA$'_R$) coincide if one of them exists. □

Note 7 (Choice of γ). $\gamma = \max_{R\in\mathscr{R}_f} M(R)$ always satisfies condition (γ). In particular, for a linear structure $\mathscr{S}$, it is sufficient to take $\gamma = \|p_{\mathscr{G}}\|_2^2$, because $\mathscr{S}(0)$ has zero rank, and $R\mathscr{S}(0) = 0$ holds for any R.

10a ⟨*set optimization solver and options* 10a⟩≡ (10c)
```
  prob = optimset();
  reg = exist('opt') && isfield(opt, 'method') && strcmp(opt.method, 'reg');
  if reg
    prob.solver = 'fminunc';
  else
    prob.solver = 'fmincon';
  end
  prob.options = optimset('disp', 'off');
  pext = [0; p];
  prob.x0 = R2th(Rini, phi * (s0 + pext(tts + 1)), psi);
```

10b ⟨*call optimization solver* 10b⟩≡ (10c)
```
  if reg
    [x, fval, flag, info] = fminunc(prob);
  else
    [x, fval, flag, info] = fmincon(prob);
  end
  info.fmin = fval;
```

10c ⟨*nonlinear optimization over R* 10c⟩≡ (10d)
```
  ⟨set optimization solver and options 10a⟩
  ⟨define ℳ and 𝒢 5e⟩
  ⟨preprocessing for weighted problem 12d⟩
  if reg
    ⟨define γ 9⟩
    prob.objective = @(th) misfit_ext(th2R(th), tts, p, [], bfs, phi, s0) ...
                          + opt.g * norm(C(th), 'fro') ^ 2;
  else
    prob.objective = @(th) misfit_ext(th2R(th), tts, p, [], bfs, phi, s0);
    prob.nonlcon = @(th) deal([], C(th));
  end
  ⟨call optimization solver 10b⟩, info.Rh = th2R(x);
  [M, ph] = misfit_ext(info.Rh, tts, p, [], bfs, phi, s0);
  ⟨postprocessing for weighted problem 12e⟩
```

The resulting function is:

10d ⟨*Structured low-rank approximation* 10d⟩≡
```
  function [ph, info] = slra_ext(tts, p, r, w, Rini, phi, psi, opt, th2R, C, s0)
  ⟨S ↦ (m,n,n_p) 24a⟩
  ⟨S ↦ 𝐒 24b⟩
  ⟨default phi 24d⟩
```

⟨*default* `s0` 2⟩
⟨*default* `psi` 8b⟩
⟨*default* `th2R` 8a⟩
⟨*default* $\mathscr{C}$ 8c⟩
⟨*default initial approximation* 11a⟩
⟨*nonlinear optimization over R* 10c⟩
⟨*define* `lra` 11b⟩
⟨*define* `R2th` 11d⟩

Note 8 (Initial approximation). Solving the outer minimization problem by local minimization requires an initial approximation for the parameter R, *i.e.*, a suboptimal solution of the structured low-rank approximation problem. Such a solution can be computed from a heuristic that ignores the data matrix structure $\mathscr{S}$ and fills in the missing values with initial estimates. Rigorous analysis of the missing values imputation question is done in [11]. Theorem 1.1 of [11] gives theoretical justification for the zero imputation in the case of unstructured $\mathscr{S}$. The resulting unstructured low-rank approximation problem can then be solved analytically in terms of the singular value decomposition.

11a ⟨*default initial approximation* 11a⟩≡ (10d)
```
if ~exist('Rini') | isempty(Rini)
  pext = [0; p];
  Rini = lra(phi * (s0 + pext(tts + 1)), r);
end
```

11b ⟨*define* `lra` 11b⟩≡ (10d)
```
function [R, P, dh] = lra(d, r)
d(find(isnan(d))) = 0;
[u, s, v] = svd(d); R = u(:, (r + 1):end)'; P = u(:, 1:r);
if nargout > 2, dh = u(:, 1:r) * s(1:r, 1:r) * v(:, 1:r)'; end
```

The computed or user supplied initial approximation R_{ini} may not satisfy the constraint ($\theta \mapsto R$). In the case when Ψ is square and nonsingular, for any R_{ini}, there is corresponding θ_{ini} parameter:

$$\theta_{\text{ini}} := \text{vec}^\top(R_{\text{ini}})\Psi^{-1}.$$

In the general case of rectangular Ψ matrix, an approximation is needed in order to obtain a θ_{ini} parameter, such that $\mathscr{R}(\theta_{\text{ini}})$ is in some sense close to R_{ini}. Let $\widehat{D}_{\text{ini}}$ be the best unstructured approximation with image is equal to $\ker(R_{\text{ini}})$ of the data matrix.

11c ⟨$(R,D) \mapsto \widehat{D}$ 11c⟩≡ (11d)
```
P = null(R); dh = P * (P \ d);
```

Since, $R_{\text{ini}}\widehat{D}_{\text{ini}} = 0$, the closeness between R_{ini} and $\mathscr{R}(\theta_{\text{ini}})$ can be measured by the Frobenius norm of the residual $\mathscr{R}(\theta_{\text{ini}})\widehat{D}_{\text{ini}}$. Imposing the normalization constraint $\|\theta_{\text{ini}}\| = 1$, the resulting approximation problem is

$$\text{minimize} \quad \text{over } \theta \quad \|\mathscr{R}(\theta)\widehat{D}_{\text{ini}}\|_{\text{F}} \quad \text{subject to} \quad \|\theta\| = 1, \tag{INI}$$

which is equivalent to unstructured approximation of the matrix $\Psi(D \otimes I_{m-r})$ by a rank $n_\theta - 1$ matrix.

11d ⟨*define* `R2th` 11d⟩≡ (10d)
```
function th = R2th(R, d, psi)
if size(psi, 1) == size(psi, 2)
  th = R(:)' / psi;
else
  ⟨(R,D) ↦ D̂ 11c⟩
  th = lra(psi * kron(dh, eye(size(R, 1))), size(psi, 1) - 1);
end
```

Note 9 (Efficient computation and software implementation). Efficient evaluation of the cost function and its derivatives in the special case of mosaic-Hankel matrix structure is presented in a companion paper [26]. The method, presented in this paper (general affine structure) and the efficient methods of [26] are implemented in Matlab (using Optimization Toolbox) and in C++ (using by the Levenberg-Marquardt algorithm [20] from the GNU Scientific Library [1]), respectively. Description of the software and overview of its applications is given in [16].

4 Weighted approximation

Problem (SLRA) is generalized in this section to the weighted structured low-rank approximation problem

$$\begin{array}{lll}\text{minimize} & \text{over } \widehat{p} \in \mathbb{R}^{n_p} & (p_{\mathscr{G}} - \widehat{p}_{\mathscr{G}})^\top W_{\text{g}} (p_{\mathscr{G}} - \widehat{p}_{\mathscr{G}}) \\ \text{subject to} & \operatorname{rank}\big(\mathscr{S}(\widehat{p})\big) \leq r, & \end{array} \tag{WSLRA}$$

where W_{g} is a positive definite matrix.

In case of a diagonal weight matrix

$$W_{\text{g}} = \operatorname{diag}(w_{\text{g}}) = \operatorname{diag}(w_1, \dots, w_{n_{\text{g}}}), \tag{w_{g}}$$

the weights can be specified by a positive vector w_{g}.

12a ⟨$w \mapsto W$ 12a⟩≡ (12d)
```
  if any(size(w) == 1), w = diag(w); end
```

In case of missing data, w_{g} is n_{g} dimensional and its indices do not correspond to the indeces of the structure parameter vector p. Then, it is convenient to specify the weights as n_p vector (or $n_p \times n_p$ matrix) with weights corresponding to the missing elements set as zeros.

12b ⟨$n_p \times n_p$ *dimensional* $W \mapsto n_g \times n_g$ *dimensional* W 12b⟩≡ (12d)
```
  if size(w, 1) == np, w = w(Ig, Ig); end
```

The change of variables

$$p'_{\mathscr{G}} = W_g^{1/2} p_{\mathscr{G}} \qquad \text{and} \qquad \widehat{p}'_{\mathscr{G}} = W_{\text{g}}^{1/2} \widehat{p}_{\mathscr{G}} \tag{$p \mapsto p'$}$$

reduces Problem (WSLRA) to an equivalent unweighted problem (SLRA). We have

$$\mathscr{S}(\widehat{p}) = S_0 + \operatorname{vec}^{-1}(\mathbf{S}\widehat{p}), \qquad \text{where} \quad \mathbf{S} := \begin{bmatrix}\operatorname{vec}(S_1) & \cdots & \operatorname{vec}(S_{n_p})\end{bmatrix} \in \mathbb{R}^{mn \times n_p}. \tag{$\mathbf{S}$}$$

12c ⟨$(\mathbf{S}, \widehat{p}) \mapsto \widehat{D} = \mathscr{S}(\widehat{p})$ 12c⟩≡
```
  dh = phi * reshape(bfs * ph, mp, n);
```

The structure $\mathscr{S}'$ of the equivalent problem is defined by the matrices S_0 and

$$\mathbf{S}' = \begin{bmatrix}\operatorname{vec}(S'_1) & \cdots & \operatorname{vec}(S'_{n_p})\end{bmatrix}, \qquad \text{where} \quad \mathbf{S}'_{:,\mathscr{G}} = \mathbf{S}_{:,\mathscr{G}} W_{\text{g}}^{-1/2} \text{ and } \mathbf{S}'_{:,\mathscr{M}} = \mathbf{S}_{:,\mathscr{M}}. \tag{$\mathscr{S} \mapsto \mathscr{S}'$}$$

We showed that problem (WSLRA) is solved by:

1. preprocessing the data p and the structure $\mathscr{S}$, as in ($p \mapsto p'$) and ($\mathscr{S} \mapsto \mathscr{S}'$),

12d ⟨*preprocessing for weighted problem* 12d⟩≡ (6d 10c)
```
  if exist('w') & ~isempty(w)
    ⟨w ↦ W 12a⟩
    ⟨n_p × n_p dimensional W ↦ n_g × n_g dimensional W 12b⟩
    ⟨preprocessing for fixed values 13a⟩
    sqrt_w = sqrtm(w); inv_sqrt_w = pinv(sqrt_w);
    bfs = double(bfs); p(Ig) = sqrt_w * p(Ig);
    bfs(:, Ig) = bfs(:, Ig) * inv_sqrt_w;
  end
```

2. solving the equivalent unweighted problem with structure parameter vector p', structure specification $\mathscr{S}'$, and rank specification r, and

3. postprocessing the solution $\widehat{p}'$, obtained in step 2, in order to obtain the solution $\widehat{p}_{\mathscr{G}} = W_{\text{g}}^{-1/2} \widehat{p}'_{\mathscr{G}}$ of the original problem.

12e ⟨*postprocessing for weighted problem* 12e⟩≡ (6d 10c)
```
  if exist('w') & ~isempty(w)
    ph(Ig) = inv_sqrt_w * ph(Ig);
    ⟨postprocessing for fixed values 13b⟩
  end
```

Using the transformation ($p \mapsto p'$), ($\mathscr{S} \mapsto \mathscr{S}'$) and the solution (M) of (SLRA), we obtain the following explicit expression for the cost function of (WSLRA)

$$M(R) = \big(\bar{G}p_{\mathscr{G}} - G^{\perp}_{:,\mathscr{M}}\operatorname{vec}(RS_0)\big)^{\top} W_{\mathrm{g}}^{-1}\bar{G}^{\top}\big(\bar{G}W_{\mathrm{g}}^{-1}\bar{G}^{\top}\big)^{-1}\bar{G}W_{\mathrm{g}}^{-1}\big(\bar{G}p_{\mathscr{G}} - G^{\perp}_{:,\mathscr{M}}\operatorname{vec}(RS_0)\big), \tag{M$_W$}$$

where $\bar{G} = G^{\perp}_{:,\mathscr{M}}G_{:,\mathscr{G}}$ and G is defined in (G).

Note 10 (Specification of fixed parameter values by infinite weights). In the case of a diagonal weight matrix (w_{g}), an infinite weight $w_j = \infty$ specifies a fixed parameter value $\widehat{p}_j = p_j$. A problem with infinite weights is equivalent to a regular structured low-rank approximation problem with fixed parameters assigned to the constant term S_0 of the structure specification. Let $\mathscr{I}_{\mathrm{f}}$ be the set of indeces of the fixed structure parameters and $\overline{\mathscr{I}}_{\mathrm{f}}$ its complement

$$\mathscr{I}_{\mathrm{f}} = \{\, j \in \{1,\dots,n_p\} \mid \widehat{p}_j = p_j \,\} \qquad \text{and} \qquad \overline{\mathscr{I}}_{\mathrm{f}} = \{\, j \in \{1,\dots,n_p\} \mid j \notin \mathscr{I}_{\mathrm{f}} \,\}.$$

The equivalent problem has structure, defined by

$$\mathscr{S}'(\widehat{p}') = \underbrace{S_0 + \sum_{i\in\mathscr{I}_{\mathrm{f}}} S_i p_i}_{S_0'} + \sum_{i\in\overline{\mathscr{I}}_{\mathrm{f}}} S_i\widehat{p}_i, \qquad \text{where} \quad \widehat{p}' := \widehat{p}|_{\overline{\mathscr{I}}_{\mathrm{f}}}.$$

13a ⟨*preprocessing for fixed values* 13a⟩≡ (12d)
```
  If = find(isinf(diag(w)));
  if ~isempty(If)
    pf = p(Ig(If));
    s0 = s0 + reshape(bfs(:, Ig(If)) * pf, mp, n);
    w(If, :) = []; w(:, If) = []; p(Ig(If)) = [];
    bfs(:, Ig(If)) = [];
    Ig_ = Ig; np_ = np; np = length(p);
    tts = reshape(bfs * vec(1:np), mp, n);
    ⟨define 𝓜 and 𝓖 5e⟩
  end
```

The estimated vector $\widehat{p}$ is recovered from the parameter vector $\widehat{p}'$ of the equivalent problem by

$$\widehat{p}|_{\overline{\mathscr{I}}_{\mathrm{f}}} = \widehat{p}' \qquad \text{and} \qquad \widehat{p}|_{\mathscr{I}_{\mathrm{f}}} = p|_{\mathscr{I}_{\mathrm{f}}}.$$

13b ⟨*postprocessing for fixed values* 13b⟩≡ (12e)
```
  if ~isempty(If)
    ph_ = Inf * ones(np_, 1);
    ph_(Ig_(If)) = pf; ph_(isinf(ph_)) = ph;
    ph = ph_;
  end
```

Note 11 (Solving (SLRA) as weighted unstructured problem). Consider an instance of problem (SLRA), refered to as problem P1, with structure $\mathscr{S} = \mathscr{S}_1$ and an instance of problem (WSLRA), refer to as problem P2, with unstructured correction ($\mathscr{S}_2 = \operatorname{vec}^{-1}$, $n_{p_2} = mn$) and weight matrix

$$W_2^{-1} = \mathbf{S_1}\mathbf{S_1}^{\top}. \tag{$\mathscr{S}_1 \mapsto W_2$}$$

13c ⟨s2w 13c⟩≡
```
  function w = s2w(s)
  tts = s2s(s);
  ⟨S ↦ (m,n,n_p) 24a⟩
  ⟨S ↦ S 24b⟩
  w = double(bfs) * double(bfs)';
```

It can be verified by inspection that the cost functions (M) and (M_W) of problems P1 and P2, respectively, coincide. The weight matrix $W_2 \in \mathbb{R}^{mn \times mn}$, defined in ($\mathscr{S}_1 \mapsto W_2$), however is singular ($\operatorname{rank}(W_2)$ is equal to the number of structure parameters of problem P1, which is less than mn). In the derivation of the cost function (M_W) it is assumed that W_g is positive definite, so that minimization of (M_W) is not equivalent to problem P2.

14a ⟨*test equivalence of structure and weights* 14a⟩≡
```
% Hankel
s1.m = 2; s1.n = 5; r = 1;
np1 = s2np(s1); p1 = 0.8 .^ (1:np1)' + 0.01 * randn(np1, 1);
opt.solver = 'm';
[ph1, info1] = slra(p1, s1, r, opt);
Dh1 = blkhank(ph1, s1.m);
D = blkhank(p1, s1.m); norm(D - Dh1, 'fro')

% Weighed
s2.m = ones(s1.m, 1); s2.n = ones(s1.n, 1); s2.w = pinv(s2w(s1));
p2 = vec(D);
[ph2, info2] = slra(p2, s2, r, opt);
Dh2 = reshape(ph2, s1.m, s1.n); norm(D - Dh2, 'fro')
%norm(p1 - [Dh2(1:s1.m, 1); Dh2(end, 2:s1.n)'])

vec(D - Dh2)' * s2.w * vec(D - Dh2)

addpath ~/mfiles/wtls
[R, P] = lra(Dh2, r);
opt.p0 = P; opt.MaxIter = 1000; opt.TolFun = 1e-10; opt.Display = 'off';
[Ph, M, Dh3, info3] = wtlsap(D, r, s2.w, opt)

vec(D - Dh3)' * s2.w * vec(D - Dh3)
```

5 Numerical examples

In this section, we present numerical examples with the three problems covered in the introduction:

- unstructured noisy matrix completion,
- system identification with missing data, and
- data-driven simulation.

The correctness of the results and the effectiveness of the methods in the paper is validated by comparison with alternative methods, specifically developed for these applications. All simulations are done in Matlab and are reproducible in the sense of [5]. An extended version [17] of this paper is a literate program (in noweb format [25]), implementing the methods in the paper and generating the presented numerical results. The necessary m-files can be downloaded from

`http://homepages.vub.ac.be/~imarkovs/publications.html`

5.1 Approximate matrix completion

In the approximate matrix completion problem (AMC) of Section 1.2.1, the methods in the paper are compared with the following alternative methods:

`wlra` — the alternating projections method of [13],

14b ⟨`wlra` 14b⟩≡ (17g)
```
method = 'wlra';
addpath ~/mfiles/missing-data;
[Ph, Lh, info_wlra] = wlra(d, r, E); dh = Ph * Lh;
```

`optspace` — a method based on spectral techniques and manifold optimization [11],

15a ⟨`optspace` 15a⟩≡ (17g)
```
method = 'optspace';
addpath ~/slra-ext-code/OptSpace_matlab_Reg/;
[X S Y info_opt] = OptSpace(sparse(d), r, 100, [], 0, E); dh = X * S * Y';
```

`lmafit` — the successive over-relaxation algorithm of [27], and

15b ⟨`lmafit` 15b⟩≡ (17g)
```
method = 'lmafit';
addpath ~/slra-ext-code/LMaFit-adp/;
addpath ~/slra-ext-code/LMaFit-adp/Utilities;
opts = []; opts.maxit = 1000; opts.est_rank = 0;
opts.rank_min = r; opts.rank_max = r;
[Ik, Jk] = ind2sub([m n], Ig);
data = partXY(eye(m), d, Ik, Jk, length(Ig));
[X, Y, info_adp] = lmafit_mc_adp(m, n, r, Ig', data, opts); dh = X * Y;
```

`rtrmc` — the Riemannian trust-region method of [4].

15c ⟨`rtrmc` 15c⟩≡
```
method = 'rtrmc';
addpath ~/slra-ext-code/;
addpath ~/slra-ext-code/GenRTR/solvers;
Jrtr = floor((Ig - 1) ./ m) + 1; Irtr = rem((Ig - 1), m) + 1;
problem = buildproblem(Irtr, Jrtr, d(Ig), ones(size(Ig)), m, n, r, 1e-6);
U0 = initialguess(problem);
[U W stats] = rtrmc(problem, opts, U0); dh = reshape(U * W, m, n);
```

These methods use an image representation

$$\widehat{D} = \mathscr{S}(\widehat{p}) = PL, \qquad \text{where} \quad P \in \mathbb{R}^{m\times r} \text{ and } L \in \mathbb{R}^{r\times n},$$

of the unstructured $m \times n$ rank r matrix $\widehat{D}$. The number of optimization variables in the image representation is $r(m+n)$ (rm in the case of `rtrmc`), so that the methods are suitable for problems with small rank r. The above cited methods are developed for recommender system applications, where the data matrix is sparse in the given elements and this sparsity is effectively used in the computations.

In contrast, the kernel representation (KER) has $m(m-r)$ variables, so that it is suitable for problems with small co-rank $m-r$. The implementation [17] of the methods in the paper is applicable only for small size problems (say, $m < 10$, $n < 100$, and $m-r < 3$). For larger problems, the efficient C implementation [16] (denoted by `slra-c` below), of the variable projection approach can be used by setting small values for the weights corresponding to the missing values. In the reported results, `slra-m` corresponds to (SLRA$'_R$) and `slra-r` corresponds to (SLRA$''_R$) with $\gamma = \|p_{\mathscr{G}}\|_2^2$ (see Note 7). `slra-c` is applied by setting the weights of the missing values to 10^{-6}.

15d ⟨`slra-m` 15d⟩≡ (17g)
```
method = 'slra-m'; p = d(:); p(Im) = NaN;
if n < 150,
  [ph, info_slra_m] = slra_ext(tts, p, r, vec(E)); dh = reshape(ph, m, n);
else, dh = NaN * ones(m, n); end,
```

16a ⟨slra-r 16a⟩≡ (17g)

```
  method = 'slra-r'; opt.method = 'reg';
  if n < 150,
    [ph, info_slra_m_reg] = slra_ext(tts, p, r, vec(E), [], [], [], opt); dh = reshape(ph, m,
  else, dh = NaN * ones(m, n); end,
```

16b ⟨slra-c 16b⟩≡ (17g)

```
  method = 'slra-c';
  s.m = ones(m, 1); s.n = ones(n, 1); s.w = vec(E); opt.maxiter = 200;
  ⟨approximate zero weights by small positive weights (never defined)⟩
  [ph, info_slra_c] = slra(d(:), s, r, opt); dh = reshape(ph, m, n);
```

16c ⟨*approximate zero weights by small positive weights* 16c⟩≡ (18h)

```
  s.w(Im) = 1e-6;
```

The data matrices, used in the simulation examples, are generated as random rank r matrices $\mathscr{S}(\bar{p})$ plus noise, where the noise matrix is zero mean Gaussian with independent identically distributed elements. A fraction of randomly selected elements of the data matrix are missing. The simulation parameters for the experiments are:

- matrix dimensions and rank, defined by variables `m`, `n`, and `r`, respectively;
- noise level, defined by a variable `nl`, and
- fraction of given elements, defined by a variable `eps`.

The relative approximation errors

$$e_{\rm g} = \frac{\|p_{\mathscr{G}} - \widehat{p}_{\mathscr{G}}\|_2}{\|p_{\mathscr{G}}\|_2} \qquad \text{and} \qquad e_{\rm m} = \frac{\|\bar{p}_{\mathscr{M}} - \widehat{p}_{\mathscr{M}}\|_2}{\|\bar{p}_{\mathscr{M}}\|_2}$$

(rows `eg` and `em`) and execution time (row `t`) are shown below for the methods compared in three simulation problems. In a problem with exact 10×100 matrix of rank 8 with 8 missing values (exact matrix completion problem):

16d ⟨*AMC, example 1* 16d⟩≡

```
  ex = 'amc_ex1'; m = 10; n = 100; r = 8; nl = 0; eps = 0.995; test_amc
```

the methods in the paper, `wlra`, and `rtrmc` recover exactly the missing values:

```
'  '    'slra-m'    'slra-r'    'slra-c'       'wlra'      'optspace'    'lmafit'        'rtrmc'
'eg'    [ 2e-16]    [ 2e-16]    [1.2410e-04]   [ 5e-11]    [   0.0213]   [2.5156e-04]    [ 2e-12]
'em'    [ 1e-14]    [ 1e-14]    [    0.0013]   [ 3e-09]    [   0.9204]   [    0.1923]    [ 2e-10]
't '    [1.3632]    [3.5105]    [    0.2710]   [0.1614]    [   0.2992]   [    0.0440]    [0.2823]
```

For the same setup with 10% noise (approximate matrix completion):

16e ⟨*AMC, example 2* 16e⟩≡

```
  ex = 'amc_ex2'; m = 10; n = 100; r = 8; nl = 0.1; eps = .995; test_amc
```

the methods in the paper, `wlra`, and `rtrmc` obtain the same approximation error (around 8% for the missing values).

```
'  '    'slra-m'     'slra-r'     'slra-c'    'wlra'      'optspace'    'lmafit'    'rtrmc'
'eg'    [ 0.0117]    [ 0.0117]    [0.0154]    [0.0117]    [  0.0231]    [0.0118]    [0.0117]
'em'    [ 0.0796]    [ 0.0797]    [0.1439]    [0.0803]    [  0.9376]    [0.1233]    [0.0796]
't '    [10.0445]    [36.9907]    [0.2400]    [0.0968]    [  0.3364]    [0.0465]    [0.2638]
```

For a problem with 10×1000 matrix of rank 9 with 101 missing values and 5% noise

16f ⟨*AMC, example 3* 16f⟩≡

```
  ex = 'amc_ex3'; m = 100; n = 1000; r = 9; nl = 0.05; eps = .99; test_amc
```

`slra-m` and `slra-r` are not applicable (they require too much time and memory). However, the C implementation `slra-c` is competitive with the alternative methods in terms of execution time and obtains the same approximation error as `rtrmc`. In this example, `lmafit` achieves the smallest error and is also the fastest of all compared methods.

```
'  '    'slra-m'    'slra-r'    'slra-c'    'wlra'        'optspace'    'lmafit'    'rtrmc'
'eg'    [   NaN]    [   NaN]    [0.0041]    [ 0.0062]    [  0.0209]    [0.0041]    [0.0041]
'em'    [   NaN]    [   NaN]    [0.2633]    [86.1488]    [  0.9021]    [0.2116]    [0.2634]
't '    [   NaN]    [   NaN]    [0.0732]    [ 1.5501]    [  1.5773]    [0.1047]    [0.2978]
```

17a ⟨*evaluate AMC* 17a⟩≡ (17g)
```
i = i + 1; methods{i} = method;
if isnan(dh), t(i) = NaN; else t(i) = toc; end
eg(i) = norm( d(Ig) - dh(Ig)) / norm( d(Ig));
em(i) = norm(d0(Im) - dh(Im)) / norm(d0(Im));
```

The data is simulated using the random number generators `rand` and `randn`. In order to make the results reproducible, we first initialize the random number generators' seeds.

17b ⟨`test_amc` 17b⟩≡ 17c▷
```
randn('seed', 0); rand('seed', 0); methods = {}; i = 0;
```

An $m \times n$ random matrix $\bar{D}$ (`d0`) with rank r is generated as the product of $m \times r$ and $r \times n$ factors that have independent uniformly distributed elements in the interval $[0,1]$.

17c ⟨`test_amc` 17b⟩+≡ ◁17b 17d▷
```
d0 = rand(m, r) * rand(r, n);
```

From the viewpoint of (SLRA), we are dealing with unstructured problem.

17d ⟨`test_amc` 17b⟩+≡ ◁17c 17e▷
```
np = m * n; tts = reshape(1:np, m, n);
```

The noisy data matrix is the sum of the true matrix $\bar{D}$ and random matrix $\widetilde{D}$, whose elements are independent normally distributed with zero mean and variance proportional to `nl`

$$D = \bar{D} + \widetilde{D}, \qquad \text{where} \quad \operatorname{vec}(\widetilde{D}) \sim \mathrm{N}\left(0, \mathtt{nl}^2 \operatorname{var}(\bar{p}) I\right).$$

17e ⟨`test_amc` 17b⟩+≡ ◁17d 17f▷
```
dt = randn(m, n); d = d0 + nl * std(d0(:)) * dt;
```

The location of the missing elements is specified by the zeros of the matrix `E` and are randomly chosen.

17f ⟨`test_amc` 17b⟩+≡ ◁17e 17g▷
```
E = 1 - ceil(rand(m, n) - eps);
Ig = find(vec(E)); Im = vec(setdiff(1:np, Ig));
d(Im) = 0; % NaN;
```

The approximation methods are applied and the relative errors e_{m} and e_{g} are computed.

17g ⟨`test_amc` 17b⟩+≡ ◁17f
```
tic, ⟨slra-m 15d⟩     ⟨evaluate AMC 17a⟩
tic, ⟨slra-r 16a⟩     ⟨evaluate AMC 17a⟩
tic, ⟨slra-c 16b⟩     ⟨evaluate AMC 17a⟩
tic, ⟨wlra 14b⟩       ⟨evaluate AMC 17a⟩
tic, ⟨optspace 15a⟩   ⟨evaluate AMC 17a⟩
tic, ⟨lmafit 15b⟩     ⟨evaluate AMC 17a⟩
tic, ⟨rtrmc>> «evaluate AMC (never defined)⟩

res = ['  ' methods;
 'eg' num2cell(eg);
 'em' num2cell(em);
 't ' num2cell(t) ];
% diary(ex), disp(res), diary off
```

5.2 System identification with missing data

Consider the system identification problem (SYSID), described in Section 1.2.2. The data w is a noisy $T = 100$ samples long random trajectory of a single-input single-output linear time-invariant system $\mathscr{B}(\bar{R})$ with lag $\ell = 2$. Samples $w(t)$, for $t \in \mathscr{T}_\mathrm{m}$, are missing. The noise is zero mean white Gaussian process with covariance matrix $\sigma^2 I_q$, *i.e.*, both the inputs $u = w_1$ and the outputs $y = w_2$ are perturbed and the input and the output noise variances are equal.

The true model parameters

$$\bar{R}_0 = \begin{bmatrix} -1 & 0.81 \end{bmatrix}, \quad \bar{R}_1 = \begin{bmatrix} 1 & -1.456 \end{bmatrix}, \quad \bar{R}_2 = \begin{bmatrix} \bar{Q}_2 & \bar{P}_2 \end{bmatrix} = \begin{bmatrix} -1 & 1 \end{bmatrix} \qquad (*)$$

are normalized with $\bar{P}_2 = 1$ and the same normalization is used for the identified model parameter $\widehat{R}$. This ensures that the parameters are unique and the systems $\mathscr{B}(\bar{R})$ and $\mathscr{B}(\widehat{R})$ can be compared by the relative parameter error

$$e_R = \frac{\|\bar{R} - \widehat{R}\|_2}{\|\bar{R}\|_2}.$$

18a ⟨*evaluate SYSID* 18a⟩≡ (20a)
```
R0_vec = vec(shiftdim(R0, 3))'; e = norm(R0_vec - Rh) / norm(R0_vec);
```

Assuming SISO system:

18b ⟨*true system* 18b⟩≡ (20a 21e)
```
R0(1, :, :) = [1 0.81]; R0(2, :, :) = [-1 -1.456]; R0(3, :, :) = [1 1];
[ell1, p, q] = size(R0); ell = ell1 - 1; m = q - p;
Q0 = - R0(:, :, 1:m); P0 = R0(:, :, m + 1:q);
sys0 = tf(fliplr(Q0(:)'), fliplr(P0(:)'), 1);
```

The Control Toolbox [21] of Matlab is used to simulate a response of the model.

18c ⟨*simulate* $\bar{w}$ 18c⟩≡ (20a 21e)
```
u0 = rand(T, m); y0 = lsim(sys0, u0); w0 = [u0 y0]; w0 = [u0 y0];
```

Note that in the Control Toolbox, the coefficients of the numerator and denominator polynomials are ordered in decreasing degrees, while the parameters in ($\mathscr{B}$) and the kernel coefficients in (SYSID) are ordered in increasing degrees. This requires interchanging the order when passing parameters to and receiving parameters from the Control Toolbox functions.

18d ⟨*add noise* 18d⟩≡ (20a 21e)
```
wt = rand(T, q); w = w0 + nl * wt / norm(wt, 'fro') * norm([u0 y0], 'fro');
```

18e ⟨*missing data* 18e⟩≡ (20a)
```
w(Tm, :) = NaN;
```

The identification problem is solved by the methods `slra-m` and `slra-r`, developed in the paper; the efficient software `slra-c` of [16] (with zero weights substituted by small constants); and the method `sysid` of [24].

18f ⟨`slra-m` *SYSID* 18f⟩≡ (20a)
```
[wh, info] = slra_ext(s2s(s), w(:), r, s.w); wh = reshape(wh, T, q);
Rh = info.Rh / info.Rh(end);
```

18g ⟨`slra-r` *SYSID* 18g⟩≡ (20a)
```
opt.method = 'reg';
[wh, info] = slra_ext(s2s(s), w(:), r, s.w, [], [], [], opt);
wh = reshape(wh, T, q); Rh = info.Rh / info.Rh(end);
```

18h ⟨`slra-c` *SYSID* 18h⟩≡ (20a)
```
Im = find(s.w == 0); ⟨approximate zero weights by small positive weights 16c⟩, opt.Rini = Rh;
[wh, info] = slra(w(:), s, r, opt); wh = reshape(wh, T, q);
Rh = info.Rh / info.Rh(end);
```

19a ⟨misdata 19a⟩≡

```
try
  idw = iddata(w(:, 2), w(:, 1)); idwh = misdata(idw, idss(ell));
  wh = [idwh.u idwh.y]; [Qh Ph] = tfdata(sysh, 'v');
  Rh = [- fliplr(Qh) fliplr(Ph)];
catch
  wh = NaN; Rh = NaN;
end
```

19b ⟨nucnrm 19b⟩≡

```
addpath ~/mfiles/nucnrm/
ww = s.w; ww(isnan(ww)) = 0;
wh = slra_nn(ww(:), s2s(s), r, [], [], s.w);
Rh = ??; wh = reshape(wh, T, q);
```

19c ⟨sysid 19c⟩≡ (20a)

```
addpath ~/mfiles/missing-data-dynamic/rik/
try
  [sysh, wh_m] = sysid_missing_data_rik(w(:, 1), w(:, 2), ell, std(wt));
  [Qh Ph] = tfdata(sysh, 'v');
  Rh = [- fliplr(Qh) fliplr(Ph)];
catch,
  wh_m = NaN; Rh = NaN;
end
wh = zeros(size(w)); wh(Tm, :) = wh_m;
```

The simulation parameters in the experiments are the

- number of samples T,
- set of missing values $\mathscr{T}_{\mathrm{m}}$, specified by a variable Tm, and
- noise variance interval, specified by a vector NL.

The reported results show the estimation error e_R for the compared methods and for the different noise levels specified in NL. In the case of uniformly distributed missing data samples:

19d ⟨*SYSID example 1* 19d⟩≡

```
ex = 'sysid_ex1'; T = 100; N = 8; NL = linspace(0, 0.1, N); Tm = 30:3:70; test_sysid
```

the developed methods perform slightly better than the alternative method for small noise level and slightly worse for high noise level:

```
'nl'       [          0]    [0.0143]    [0.0286]    [0.0429]    [0.0571]    [0.0714]    [0.0857]    [0.1000]
'slra-m'   [1.5624e-15]    [0.0085]    [0.0134]    [0.0390]    [0.0325]    [0.0693]    [0.0347]    [0.0778]
'slra-r'   [1.5624e-15]    [0.0085]    [0.0134]    [0.0390]    [0.0325]    [0.0693]    [0.0347]    [0.0778]
'slra-c'   [2.9202e-06]    [0.0085]    [0.0134]    [0.0390]    [0.0325]    [0.0693]    [0.0347]    [0.0778]
'sysid'    [5.6315e-15]    [0.0096]    [0.0148]    [0.0434]    [0.0356]    [0.0697]    [0.0322]    [0.0758]
```

Similar results are obtained in the case of consecutive missing samples:

19e ⟨*SYSID example 2* 19e⟩≡

```
ex = 'sysid_ex2'; T = 100; N = 8; NL = linspace(0, 0.1, N); Tm = 40:60; test_sysid
```

```
'nl'       [          0]    [0.0143]    [0.0286]    [0.0429]    [0.0571]    [0.0714]    [0.0857]    [0.1000]
'slra-m'   [1.6123e-15]    [0.0063]    [0.0142]    [0.0366]    [0.0529]    [0.0738]    [0.0410]    [0.0850]
'slra-r'   [1.6123e-15]    [0.0063]    [0.0142]    [0.0366]    [0.0529]    [0.0738]    [0.0411]    [0.0851]
'slra-c'   [6.5227e-07]    [0.0063]    [0.0142]    [0.0366]    [0.0529]    [0.0738]    [0.0411]    [0.0851]
'sysid'    [3.3311e-16]    [0.0080]    [0.0191]    [0.0431]    [0.0578]    [0.0766]    [0.0417]    [0.0829]
```

This latter problem can be solved also by treating the data as two independent trajectories without missing data.

20a ⟨test_sysid 20a⟩≡
```
  randn('seed', 0); rand('seed', 0);
  ⟨true system 18b⟩, ⟨simulate w̄ 18c⟩
  ⟨structure specification for identification 4a⟩
  for i = 1:N
    nl = NL(i); ⟨add noise 18d⟩, ⟨missing data 18e⟩
    ⟨weights for identification with missing values 4b⟩
    ⟨slra-m SYSID 18f⟩, ⟨evaluate SYSID 18a⟩, e_slra_m(i) = e;
    ⟨slra-r SYSID 18g⟩, ⟨evaluate SYSID 18a⟩, e_slra_r(i) = e;
    ⟨slra-c SYSID 18h⟩, ⟨evaluate SYSID 18a⟩, e_slra_c(i) = e;
    ⟨sysid 19c⟩, ⟨evaluate SYSID 18a⟩, e_sysid(i) = e;
  end

  labels = {'nl'; 'slra-m'; 'slra-r'; 'slra-c'; 'sysid'};
  res = [labels, num2cell([NL; e_slra_m; e_slra_r; e_slra_c; e_sysid])]
  % diary(ex), disp(res), diary off
```

5.3 Data-driven simulation

Consider the data-driven simulation problem (DDSIM), described in Section 1.2.3. The to-be-simulated system is $\bar{\mathscr{B}} = \mathscr{B}(\bar{R})$, with parameter $\bar{R}$ given in ($*$) and with an input/output partition $w = (u, y)$ of the variables. The given trajectory $w' = (u', y') \in (\mathbb{R}^2)^{T'}$ is a noise corrupted random trajectory of $\bar{\mathscr{B}}$ (the same simulation setup as in Section 5.2) and the to-be-simulated trajectory $w'' = (u'', y'') \in (\mathbb{R}^2)^{T''}$ is the impulse response $\bar{h}$ of $\bar{\mathscr{B}}$, *i.e.*, the response of $\bar{\mathscr{B}}$ to pulse input under zero initial conditions:

$$u'' = (\underbrace{0,\dots,0}_{\ell},\underbrace{1,0,\dots,0}_{\text{pulse input}}) \qquad \text{and} \qquad y'' = (\underbrace{0,\dots,0}_{\ell},\underbrace{\widehat{h}(0),\widehat{h}(1),\dots,\widehat{h}(T_2-\ell-1)}_{\text{impulse response}}).$$

20b ⟨w'' impulse response 20b⟩≡ (21e)
```
  u2 = [zeros(ell, 1); 1; zeros(T2 - ell - 1, 1)];
  y2 = [zeros(ell, 1); NaN * ones(T2 - ell, 1)]; w2 = [u2 y2];
```

The methods in the paper are compared with an alternative subspace-type method `ddsim` [19, 15] in terms of the relative approximation error

$$e_h = \frac{\|\bar{h} - \widehat{h}\|_2}{\|\bar{h}\|_2}.$$

Subspace methods are multi stage methods, *i.e.*, they split the nonconvex optimization problem in several steps that are individually convex, but do not guarantee global or local optimally with respect to the overall problem. In general, subspace-type methods are more efficient but less accurate than local optimization-based methods. The results for $T' = 30$, $T'' = 52$, and noise level in the range 0–40%:

20c ⟨DDSIM example 20c⟩≡
```
  T1 = 30; T2 = 52; N = 6; NL = linspace(0, 0.1, N); test_ddsim
```
confirm this rule of thumb:

```
 'nl'        [          0]    [0.0200]    [0.0400]    [0.0600]    [0.0800]    [0.1000]
 'slra-m'    [1.8920e-15]    [0.0451]    [0.1149]    [0.1652]    [0.2678]    [0.2301]
 'slra-r'    [1.8920e-15]    [0.0451]    [0.1150]    [0.1652]    [0.2679]    [0.2298]
 'slra-c'    [1.0123e-05]    [0.0451]    [0.1149]    [0.1652]    [0.2679]    [0.2298]
 'ddsim'     [3.2215e-15]    [0.0572]    [0.1727]    [0.2772]    [0.3012]    [0.5311]
```

The true impulse response and the approximations for the 40% noise run are plotted in Figure 1.

20d ⟨slra-m DDSIM 20d⟩≡ (21e)
```
  ⟨structure specification for data-driven simulation slra-m 5a⟩
  [wh, info] = slra_ext(tts, vec([w1' w2']), r, diag(w));
  ⟨extract ĥ from ŵ'' slra-m 5c⟩
```

21a ⟨slra-r *DDSIM* 21a⟩≡ (21e)

```
⟨structure specification for data-driven simulation slra-m 5a⟩,
opt.method = 'reg'; R0_vec = vec(shiftdim(R0, 3))'; opt.Rini = R0_vec;
[wh, info] = slra_ext(tts, vec([w1' w2']), r, diag(w), [], [], [], opt);
⟨extract ĥ from ŵ'' slra-m 5c⟩
```

21b ⟨slra-c *DDSIM* 21b⟩≡ (21e)

```
⟨structure specification for data-driven simulation slra-c 5b⟩
[wh, info] = slra([vec(w1); vec(w2)], s, r);
⟨extract ĥ from ŵ'' slra-c 5d⟩
```

21c ⟨*alternative method for DDSIM* 21c⟩≡ (21e)

```
addpath ~/mfiles/detss/
hh = uy2h(w1(:, 1:m), w1(:, m + 1:end), ell, 10, T2 - ell);
```

21d ⟨*evaluation DDSIM* 21d⟩≡ (21e)

```
e = norm(h0 - hh) / norm(hh);
```

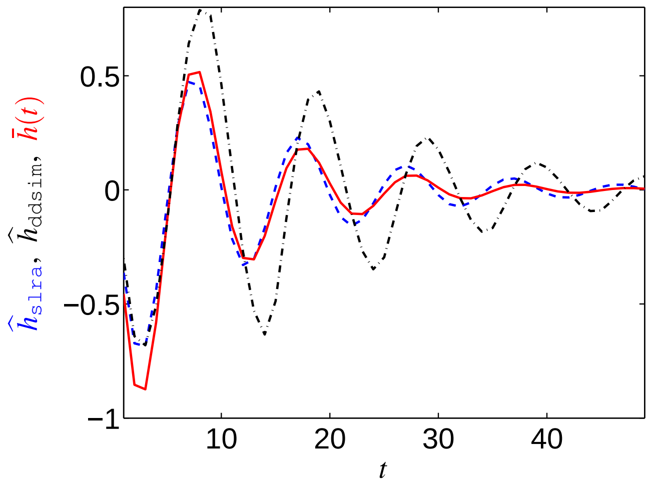


Figure 1: Data-driven simulation of impulse response: true impulse response — solid line, approximation by the methods in the paper — dashed line, approximation by the methods of [15] — dashed-dotted line.

21e ⟨test_ddsim 21e⟩≡

```
randn('seed', 0); rand('seed', 0);
⟨true system 18b⟩ h0 = impulse(sys0, T2 - ell - 1);
for i = 1:N
  nl = NL(i);
  T = T1; ⟨simulate w̄ 18c⟩, ⟨add noise 18d⟩, w1 = w;
  ⟨w'' impulse response 20b⟩
  ⟨slra-m DDSIM 20d⟩ ⟨evaluation DDSIM 21d⟩, e_slra_m(i) = e; hh1 = hh;
  ⟨slra-r DDSIM 21a⟩ ⟨evaluation DDSIM 21d⟩, e_slra_r(i) = e;
  ⟨slra-c DDSIM 21b⟩ ⟨evaluation DDSIM 21d⟩, e_slra_c(i) = e;
  ⟨alternative method for DDSIM 21c⟩, ⟨evaluation DDSIM 21d⟩, e_ddsim(i) = e; hh2 = hh;
end

labels = {'nl'; 'slra-m'; 'slra-r'; 'slra-c'; 'ddsim'};
res = [labels, num2cell([NL; e_slra_m; e_slra_r; e_slra_c; e_ddsim])];
% diary('ddsim_ex'), disp(res), diary off

figure
plot(hh1(2:end), '-b'), hold on, plot(h0(2:end), '-r'), plot(hh2(2:end), '-.k')
ax = axis; axis([1, T2 - ell - 1, ax(3:4)]), print_fig('slra-ext-f2')
```

Compare the result with the one obtained by identification of a model and model based simulation:

22 ⟨*alternative method: SYSID + simulation* 22⟩≡
```
tts = blkhank(reshape(1:(T1 * q), q, T1), ell + 1);
[wh1_, info_] = slra_ext(tts, vec(w1'), q * ell + 1);
sysh_ = tf(fliplr(info_.Rh(1:2:end)), - fliplr(info_.Rh(2:2:end)), -1);
hh_ = impulse(sysh_, T2 - ell); hh_ = hh_(1:T2 - ell);
norm(hh - hh_)
```

6 Conclusions

A variable-projection-like method for structured low-rank approximation with missing data was developed. The approach was furthermore generalized to weighted structured low-rank approximation with missing values. After elimination of the approximation $\widehat{p}$, the remaining nonlinear least-squares problem subject to quadratic equality constraints was solved as an equivalent regularized unconstrained optimization problem.

The problem and solution methods developed have applications in matrix completion (unstructured problems), system identification with missing data, and data-driven simulation and control (mosaic-Hankel structured problems). The performance of the methods in the paper was illustrated on small-size simulation examples and was compared with the performance of problem specific methods. Efficient computation for large scale problems appearing in applications such as recommender systems and system identification is a topic of future research.

Acknowledgements

The implementation of the method `sysid` for system identification with missing data, used in the numerical examples of Section 5.2, was kindly provided to us by Rik Pintelon. The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC Grant agreement number 258581 "Structured low-rank approximation: Theory, algorithms, and applications".

References

[1] GSL — GNU scientific library. `www.gnu.org/software/gsl/`.

[2] P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, Princeton, NJ, 2008.

[3] C. Baker, P.-A. Absil, and K. Gallivan. GenRTR Riemannian optimization package. `http://www.math.fsu.edu/~cbaker/GenRTR`.

[4] N. Boumal and P.-A. Absil. RTRMC: A Riemannian trust-region method for low-rank matrix completion. In J. Shawe-Taylor, R.S. Zemel, P. Bartlett, F.C.N. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24 (NIPS)*, pages 406–414. 2011.

[5] J. Buckheit and D. Donoho. *Wavelets and statistics*, chapter "Wavelab and reproducible research". Springer-Verlag, Berlin, New York, 1995.

[6] E. Candes and Y. Plan. Matrix completion with noise. *arXiv:0903.3131*, March 2009.

[7] E. Candés and B. Recht. Exact matrix completion via convex optimization. *Found. of Comput. Math.*, 9:717–772, 2009.

[8] G. Golub and V. Pereyra. Separable nonlinear least squares: the variable projection method and its applications. *Institute of Physics, Inverse Problems*, 19:1–26, 2003.

[9] G. Heinig. Generalized inverses of Hankel and Toeplitz mosaic matrices. *Linear Algebra Appl.*, 216(0):43–59, February 1995.

[10] A. Isaksson. Identification of ARX-models subject to missing data. *IEEE Trans. Automat. Control*, 38(5):813–819, May 1993.

[11] R. Keshavan, A. Montanari, and S. Oh. Matrix completion from noisy entries. *J. Mach. Learn. Res.*, 11:2057–2078, August 2010.

[12] I. Markovsky. Structured low-rank approximation and its applications. *Automatica*, 44(4):891–909, 2008.

[13] I. Markovsky. *Algorithms and literate programs for weighted low-rank approximation with missing data*, volume 3 of *Springer Proc. Mathematics*, pages 255–273. Springer, 2011.

[14] I. Markovsky. *Low Rank Approximation: Algorithms, Implementation, Applications*. Springer, 2012.

[15] I. Markovsky and P. Rapisarda. Data-driven simulation and control. *Int. J. Control*, 81(12):1946–1959, 2008.

[16] I. Markovsky and K. Usevich. Software for weighted structured low-rank approximation. Technical Report 339974, Univ. of Southampton, `http://eprints.soton.ac.uk/339974`, 2012.

[17] I. Markovsky and K. Usevich. Structured low-rank approximation with missing values. Technical Report 340718, Univ. of Southampton, `homepages.vub.ac.be/~imarkovs/publications.html`, 2012.

[18] I. Markovsky, S. Van Huffel, and R. Pintelon. Block-Toeplitz/Hankel structured total least squares. *SIAM J. Matrix Anal. Appl.*, 26(4):1083–1099, 2005.

[19] I. Markovsky, J. C. Willems, P. Rapisarda, and B. De Moor. Algorithms for deterministic balanced subspace identification. *Automatica*, 41(5):755–766, 2005.

[20] D. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM J. Appl. Math.*, 11:431–441, 1963.

[21] The MathWorks. *Control System Toolbox: User's guide*.

[22] C. Moler and Ch. Van Loan. Nineteen dubious ways to compute the exponential of a matrix. *SIAM Review*, 20(4):801–836, 1978.

[23] J. Nocedal and S. Wright. *Numerical optimization*. Springer-Verlag, 1999.

[24] R. Pintelon and J. Schoukens. Frequency domain system identification with missing data. *IEEE Trans. Automat. Control*, 45(2):364–369, February 2000.

[25] N. Ramsey. Literate programming simplified. *IEEE Software*, 11:97–105, 1994.

[26] K. Usevich and I. Markovsky. Variable projection for affinely structured low-rank approximation in weighted 2-norm, 2012. Available from `http://arxiv.org/abs/1211.3938`.

[27] Z. Wen, W. Yin, and Y. Zhang. Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm. Technical report, Rice University, 2010. CAAM Technical Report TR10–07.

A Special case of affine structure

A commonly encountered special case of the affine structure is

$$\big[\mathscr{S}(\widehat{p})\big]_{ij} = S_{0,ij} + \widehat{p}_{\mathtt{S}_{ij}} \qquad \text{for some} \qquad \mathtt{S}_{ij} \in \{1,\dots,n_p\}^{m\times n}. \tag{S}$$

In (S), each element of the structured matrix $\mathscr{S}(p)$ is equal to the sum of the $\mathtt{S}_{ij}$th element of the parameter vector p and the constant $S_{0,ij}$. The structure is then specified by the matrices S_0 and $\mathtt{S}$. Although (S) is a special case of the general affine structure ($\mathscr{S}$), it covers many linear modeling problems and will therefore be used in the implementation of the solution method.

In the implementation of the algorithm, the matrix S corresponds to a variable `tts`. Given the matrix S, specifying the structure, and a structure parameter vector $\widehat{p}$, the structured matrix $\mathscr{S}(\widehat{p})$ is constructed by `dh = s0 + ph(tts)`. The matrix dimensions `m`, `n`, and the number of parameters `np` are obtained from S as follows:

24a ⟨S $\mapsto (m,n,n_p)$ 24a⟩≡ (6d 10d 13c)
```
  [mp, n] = size(tts); np = max(max(tts));
  if exist('phi', 'var') && ~isempty(phi), m = size(phi, 1); else m = mp; end
```
The transition from the specification of (S) to the specification in the general affine case ($\mathscr{S}$) is done by

24b ⟨S $\mapsto$ **S** 24b⟩≡ (6d 10d 13c)
```
  vec_tts = tts(:); NP = 1:np;
  bfs = vec_tts(:, ones(1, np)) == NP(ones(mp * n, 1), :);
```
$\mathscr{S}$ is represented by the $mn \times n_p$ matrix (`bfs` in the code)

$$\mathbf{S} := \begin{bmatrix} \operatorname{vec} S_1 & \cdots & \operatorname{vec} S_{n_p} \end{bmatrix}.$$

Conversely, for an affine structure of the type (S), defined by **S** (and m, n), the matrix S is constructed by

24c ⟨**S** $\mapsto$ S 24c⟩≡
```
  tts = reshape(bfs * (1:np)', mp, n);
```
For compatibility with the software package for mosaic-Hankel matrices [26], we consider structures of the form $\Phi\mathscr{S}$, where Φ is a full row rank matrix and $\mathscr{S}$ is an affine structure. The default value for Φ is the identity matrix I_m.

24d ⟨*default* phi 24d⟩≡ (6d 10d)
```
  if ~exist('phi', 'var') | isempty(phi), phi = eye(size(tts, 1)); end
```