Identification of Shifted-Affine Linear Parameter-Varying Models in the Errors-In-Variables Setting*

Leander Hemelhof[1], Chris Verhoek[2,3], Roland Tóth[2,4], Ivan Markovsky[5,6], and Panagiotis Patrinos[1]

[1]Department of Electrical Engineering (ESAT-STADIUS), KU Leuven, Leuven, Belgium
[2]Control Systems group, Dept. of Electrical Engineering, Eindhoven University of Technology, Eindhoven, The Netherlands
[3]Dept. of Electrical and Systems Engineering, University of Pennsylvania, Pennsylvania, Philadelphia,
[4]Systems and Control Lab, HUN-REN Institute for Computer Science and Control, Budapest, Hungary
[5]International Centre for Numerical Methods in Engineering (CIMNE), Barcelona, Spain
[6]Catalan Institution for Research and Advanced Studies (ICREA), Barcelona, Spain

Tuesday 19th May, 2026

Abstract

This paper presents an identification method for linear parameter-varying (LPV) models with affine dependence on the scheduling variables, referred to as shifted-affine linear parameter-varying (SALPV) models. Utilizing recent work extending the Fundamental Lemma by Willems et al. to LPV models, the identification problem is shown to be equivalent to a structured low-rank approximation problem and is solved using a variable projection method. The proposed method exploits the Kronecker-Hankel structure, a block Hankel matrix where each block element is the Kronecker product of a known vector with an unknown vector, which is a structure not previously considered in related literature. We consider noise on both the input and output signal, and noise-free scheduling variables, which corresponds to the errors-in-variables (EIV) setting in the linear time-invariant (LTI) literature, a setting which is not typically considered in existing LPV identification methods. Under reasonable assumptions, the proposed method is guaranteed to converge to a stationary point of the cost function, which is proportional to the log-likelihood under the considered noise setting, and has a runtime per iteration that scales linearly with the number of samples. Extensions of the proposed method are introduced to allow for additional freedom in the problem definition, including missing and fixed elements and a weighted cost function. The proposed method is compared to state-of-the-art methods on a simulated mass-spring-damper system.

Keywords: *System identification, Linear parameter-varying systems, Behavioral approach, Structured low-rank approximation.*

1 Introduction

Accurate identification of LPV models is critical for applications where system behaviors change with operating conditions, such as automotive systems [1, 2], aerospace [3, 4], and industrial processes [5, 6]. While many identification methods exist for LPV models, they typically assume noise only on the outputs, which is not always realistic. The setting where both inputs and outputs are corrupted by measurement noise is known in the literature as the EIV noise setting [7]. Existing identification methods are not able to consistently estimate under this noise setting without significantly increasing the complexity of the estimated noise model, motivating this work.

Compared to LTI models, LPV models depend on time-varying signals called *scheduling variables*. This dependence allows for LPV models to better capture the nonlinear behavior of a plant or external effects influencing the dynamics [8]. The advantage over more general nonlinear models is that LPV models result in an LTI model for fixed values of the scheduling variables, allowing for

*Corresponding author: Leander Hemelhof. leander.hemelhof@esat.kuleuven.be

the use of linear tools, e.g., semidefinite programming (SDP) methods, to achieve nonlinear control design. In particular, the model class of SALPV models is a popular choice. SALPV models admit an input-output representation where the coefficients of the inputs and outputs at time t depend affinely on the scheduling variables at time t (see (1)). This fact can be leveraged for more efficient controller design.

Depending on the requirements of the model and the available measurements, the most suitable identification method differs. Local models describe the behavior of the plant around a set of distinct operating points, i.e. fixed values of the scheduling variables, and are suitable for specific control applications and plants that only operate on or near a few distinct operating points. Local LPV identification methods identify a set of LTI models, each valid for a specific operating point. Then, these separate LTI models can be interpolated to describe the plant behavior between the fixed operating points, forming the local LPV model, which aims to be accurate near the considered operating regime. The exact mechanism of interpolation depends on the particular method used [9, 10]. Local LPV models have commonly been used in control applications using *gain scheduling* [11]. Gain scheduling is a nonlinear controller design procedure where a linear controller is designed for each of the LTI models, which are interpolated over the operating space to determine the exact gains of the controller based on the current value of the scheduling variable.

Global LPV models describe the behavior of the plant for all scheduling trajectories by embedding the scheduling variables into the model representation. These models are identified using input-scheduling-output trajectories, and aim to be valid for all possible scheduling variations. Popular global LPV identification methods are methods for LPV state-space (LPV-SS) [12, 13] and LPV input-output (LPV-IO) [14, 15, 16] representations. For mechatronic applications, however, it is often more convenient to design controllers in the frequency domain, using estimated frequency response functions (FRFs) of the model. For LPV models, these FRFs depend on the scheduling variables, and different types of FRFs are available depending on the use case, representing local or global relationships of the signals [17]. There exist various identification methods for FRFs of LPV models, e.g., [6], which have also been successfully used in controller design [3, 5].

Existing LPV identification methods for the SALPV model class can handle the identification of models with multiple inputs and multiple outputs under different noise models, the most general of which is the Box-Jenkins noise model. Noise on both the inputs and the outputs, however, is not directly handled by these methods. By transforming the input noise to output noise, methods that can estimate models with an additive scheduling-dependent Box-Jenkins noise model under input-correlated output noise are theoretically capable of consistent estimation under the considered EIV setting. However, the identification problem becomes significantly more expensive to solve even for simple models, and due to the additional parameters needed, the variance of the estimated model parameters will increase. As an example, prediction-error methods (PEM) (see e.g., [14]) can generally make use of the above strategy. For LPV models, we can also consider noise on the scheduling variables. So far, there is no available literature that can handle noise on inputs, outputs, and scheduling variables simultaneously. There exist works that assume noise-free inputs and can handle noise on both output and scheduling variables with some assumptions, such as [15], later improved in [18]. When considering noise on the inputs and outputs, but not on the scheduling variables, only limited work is available for LPV models, e.g., [19]. The cited work is designed for a more general LPV model class than SALPV, which results in a complicated identification method with limited guarantees. Based on the available structure in the SALPV model class, there is significant potential for a simpler and efficient identification method, offering another motivation for this work.

The main difficulty for identification in the EIV setting, as mentioned above, is that applying noisy inputs to a model results in additional noise at the output level, which can result in a biased model estimate if not handled correctly. One way to avoid many complications is to consider the behavioral setting [20]. The kernel representation that follows from this setting is defined in function of input-output trajectories without specific input-output partitioning. This allows all parts of the sample, be it input or output, to be treated equally during the identification process. In fact, for kernel representations, it is more convenient to consider noise on both inputs and outputs, rather than only on the outputs.

Recent works have investigated SALPV models in the context of the behavioral setting [21, 22], extending the fundamental lemma by Willems et al. to SALPV models. The work [22] has extended the notion of *most powerful unfalsified model* (MPUM) of a signal from LTI models to SALPV models, which is pivotal in the identification of model representations for this model class. Both works are limited to noise-free data, leaving the identification of SALPV models from noisy data in the behavioral setting an open question. However, [21] and [22] have opened the path to extending

existing identification methods for LTI models in the behavioral setting to SALPV models. An example of such a method is [23], where the LTI model identification problem is solved in the context of structured low-rank approximation (SLRA) with affinely structured matrices.

The problem considered in this paper is SALPV model identification in the EIV setting, i.e., a setting where all variables of the system (both inputs and outputs) are observed with additive measurement noise. The scheduling variables are assumed to be known exactly, and are not measured. The EIV identification problem is extensively treated in the LTI setting (see, e.g., the overview paper [7] and the book [24]), however, not yet in the LPV setting. To the best of our knowledge this is the first work addressing EIV identification of SALPV models directly.

In order to formally pose and solve the problem, we use the behavioral framework, which defines a dynamical model as a set of trajectories. The lack of a priori distinction of the variables into inputs and outputs in the behavioral framework is consistent with the assumption that all variables are contaminated by noise in the EIV setting. As in the LTI case, the considered SALPV identification problem can be seen as a complexity-vs-accuracy trade-off. Here accuracy is the 2-norm of the difference between the given data and its best approximation by forward simulation of the model, while the complexity of the model characterizes the dimension of the space of model trajectories. The precise definition of complexity is given in Section 2.2, where the behavioral approach to the class of SALPV models is introduced.

In this work, we propose an identification method for SALPV models based on solving an SLRA problem, resulting in an LPV kernel representation of the model. Due to a Kronecker product-like structure in the definition of the LPV kernel representation, the low-rank matrix will have a Kronecker-Hankel structure, i.e., a block Hankel matrix where each block element is the Kronecker product of a known vector with an unknown vector. The SLRA problem solved by the proposed method is formulated in a way such that each value of the optimization variables corresponds to a unique causal model representation. The method will initially be discussed considering multiple-input multiple-output (MIMO) SALPV models in the EIV setting where all outputs have the same lag. Later in the paper, the method will be generalized to different output lags per channel. Further extensions are also provided for handling fixed and missing data, combining multiple independent data sets, and handling different noise settings. In particular, the complexity of identifying kernel models with arbitrary model order are discussed together with a way to deal with the accompanying complexity. The proposed method with all extensions mentioned in this work is implemented in a publicly available Matlab software package (`https://github.com/LeanderHemelhof/LPV-SLRA`).

In summary, the main contributions of this paper are the following:

C1 We formulate the identification problem of SALPV models as an SLRA problem of a Kronecker-Hankel structured matrix. The resulting optimization problem guarantees that all values of the optimization variables correspond to unique causal model representations. The solution to this problem is the maximum likelihood estimate of the model parameters if the inputs and outputs are both corrupted by zero-mean white Gaussian noise.

C2 We analyze the matrix structures occurring in the SLRA problem with Kronecker-Hankel structure, which enables efficient solution of SLRA problems with this structure. Furthermore, these structures can be used in other applications, such as other Fundamental Lemma-based identification and control methods with SALPV models.

C3 We propose an identification method for SALPV models that solves the resulting SLRA problem with a computational complexity that scales linearly with the number of measured samples. This identification method is optimal under the EIV noise setting, which is not typically considered in other LPV identification methods. Furthermore, under reasonable assumptions, the proposed method is guaranteed to converge to a first-order stationary point of the likelihood function under the considered EIV noise setting.

C4 We discuss the difficulty of identifying kernel representations for MIMO SALPV models with arbitrary model order and propose a way to deal with it. Note that this difficulty is also present, but not often discussed, when identifying kernel representations for MIMO LTI models with arbitrary order, making this contribution relevant for LTI identification of kernel representations as well.

Section 2 presents the notation used throughout this paper, introduces the model class of interest, and formulates the problem considered in this work, constituting to C1. Section 3 reformulates the identification problem and derives the method proposed in C3, giving C2 in the process. Section 4 lists a number of extensions of the problem that can also be handled by the proposed method,

providing C4. Section 5 compares the proposed method with state-of-the-art LPV identification methods using simulated data. Finally, Section 6 concludes the paper.

2 Preliminaries

We start by introducing the notation used throughout this work. After that follows a brief introduction to the behavioral framework and the useful properties and notation for both LTI and SALPV models. Finally, we end by introducing the kernel representation, a model representation typically considered in the behavioral framework, for both LTI and SALPV models.

2.1 Notation

To keep the notation orderly, everything is split up into different sections. The first section describes notation related to signals and their corresponding operators. The second section describes special matrices and operators that operate on vectors and matrices. The final sections describes notation related to LPV models.

2.1.1 Signals

The terms *sequence*, *(discrete-time) signal*, and *trajectory* are used interchangeably in this work. Only discrete-time signals will be considered. Denote the set of natural numbers as $\mathbb{N} = \{1, 2, \ldots\}$, and the set of $a \times b$ real polynomial matrices in the parameter z by $\mathbb{R}^{a\times b}[z]$. The set of finite-length real-valued q-variate signals of length T is denoted by $(\mathbb{R}^q)^T$. Similarly, the set of infinite-length q-variate signals is denoted by $(\mathbb{R}^q)^{\mathbb{N}}$. For a signal $w \in (\mathbb{R}^q)^{\mathbb{N}}$ we define the following operators:

- The forward shift operator $\sigma(\cdot)$ such that $\sigma w(t) = w(t+1)$ for all $t \in \mathbf{N}$.
- The cut operator $w|_T$ such that

  $$w|_T := \big(w(1), w(2), \ldots, w(T)\big),$$

  for all $T \in \mathbb{N}$. More generally, we denote the truncation of a signal $w \in (\mathbb{R}^q)^T$ as $w_{[T_1:T_2]}$, with $T_1 \leq T_2 \leq T$, which represents the signal

  $$\big(w(T_1), w(T_1+1), \ldots, w(T_2)\big) \in (\mathbb{R}^q)^{T_2-T_1+1}.$$

  T_1 and T_2 can be omitted from the notation, with $w_{[:T_2]}$ defined as $w_{[1:T_2]}$ and $w_{[T_1:]}$ defined as $w_{[T_1:T]}$. The same subscript notation will be used to denote slices of matrix rows and columns.
- For finite-length signals $w \in (\mathbb{R}^q)^T$, the vectorization operator is defined as

  $$\operatorname{col}(w) := \begin{bmatrix} w(1)^\top & w(2)^\top & \ldots & w(T)^\top \end{bmatrix}^\top \in \mathbb{R}^{qT}.$$

Applying a signal operator $\mathcal{A}(\cdot)$ to a set of signals $\mathcal{B}$ applies this operator to each of its elements, i.e.,

$$\mathcal{A}(\mathcal{B}) := \{\mathcal{A}(w) \mid w \in \mathcal{B}\}.$$

Both the forward shift operator $\sigma(\cdot)$ and the cut operator $\cdot|_T$ are examples of such an operator.

2.1.2 Matrices and Vectors

For some cases it is more convenient to work with signals in the context of matrices and vectors, for which specific operators are defined. The 2-norm of a finite-length signal $w \in (\mathbb{R}^q)^T$ is defined to be the norm of its vectorization, i.e.,

$$\|w\|_2 := \|\operatorname{col}(w)\|_2,$$

and other vector norms are defined analogously for signals, such as the weighted squared 2-norm of a vector x with weighting matrix W, defined as

$$\|x\|_W^2 := x^\top W x.$$

The block Hankel operator with L block rows of a signal $w \in (\mathbb{R}^q)^T$ is denoted by $H_L(w)$, with

$$H_L(w) := \begin{bmatrix} w(1) & w(2) & \cdots & w(T-L+1) \\ w(2) & w(3) & \cdots & w(T-L) \\ \vdots & \vdots & & \vdots \\ w(L) & w(L+1) & \cdots & w(T) \end{bmatrix},$$

which is a $qL \times (T-L+1)$ matrix. Denote the block diagonal operator by

$$\text{blkdiag}(M_1, M_2, \ldots, M_K) := \begin{bmatrix} M_1 & & & \\ & M_2 & & \\ & & \ddots & \\ & & & M_K \end{bmatrix}.$$

We also make use of some additional common notation and operators from linear algebra. The Kronecker product of two matrices A and B is denoted by $A \otimes B$. Denote the $a \times b$ matrix of all zeros (respectively ones) by $0_{a\times b}$ (respectively $1_{a\times b}$), and the identity matrix of size a by I_a. The i-th unit vector is denoted by $e^{(i)}$, with the dimension determined by the context it is used in.

2.1.3 LPV Notation

Moving on to the LPV related notation, define a MIMO SALPV system with scheduling variables $\rho(t) \in \mathbb{P}$, where $\mathbb{P}$ is a subset of $\mathbb{R}^{n_\rho}$, such that it admits the following minimal input-output representation

$$\sum_{i=0}^{\ell} A_i\big(\rho(t-i)\big)y(t-i) = \sum_{j=0}^{\ell} B_j\big(\rho(t-j)\big)u(t-j) \quad \forall t \in \mathbb{N},\ t > \ell, \tag{1a}$$

with

$$A_i\big(\rho(t)\big) := a_{i,0} + \sum_{k=0}^{n_\rho} a_{i,k}\rho_k(t), \tag{1b}$$

$$B_j\big(\rho(t)\big) := b_{j,0} + \sum_{k=0}^{n_\rho} b_{j,k}\rho_k(t), \tag{1c}$$

with model parameters $a_{i,k} \in \mathbb{R}^{p\times p}$ and $b_{j,k} \in \mathbb{R}^{p\times m}$. Here m represents the number of inputs, p represents the number of outputs, and ℓ represents the lag of the model.

Given a trajectory $w \in (\mathbb{R}^q)^T$ and a scheduling sequence $\rho \in \mathbb{P}^T$, we define the extended sequence of scheduling variables as

$$\rho'(t) := \begin{bmatrix} 1 \\ \rho(t) \end{bmatrix} \quad \forall t \in (1, 2, \ldots, T), \tag{2}$$

and we define the extended trajectory w_{ext} such that

$$w_{\text{ext}}(t) := \rho'(t) \otimes w(t) \quad \forall t \in (1, 2, \ldots, T). \tag{3}$$

For more compact notation we will also write (3) as

$$w_{\text{ext}} := \rho' \otimes w.$$

The extended trajectory is a q_{ext}-variate trajectory with $q_{\text{ext}} := (n_\rho + 1)q$. Define the corresponding matrix

$$P_t := \rho'(t) \otimes I_q, \tag{4}$$

such that $w_{\text{ext}}(t) = P_t w(t)$.

2.2 The Behavioral Framework

In the behavioral framework, models are described by the set of trajectories that are admissible for this model. We refer to this set as the behavior of the model, and denote it by the symbol $\mathcal{B}$. Compared to the classical setting, the behavior of a model is an abstraction that allows us to

reason about models without the need to consider a particular representation (e.g., state-space or input-output representation). Model properties are instead related to properties of the behavior $\mathcal{B}$. For example, a model is linear if its behavior is a linear subspace of $(\mathbb{R}^q)^{\mathbb{N}}$, and is time-shift invariant if $\sigma\mathcal{B} \subseteq \mathcal{B}$ [25]. Although we still keep the behavior abstract at this point, it will be made more specific later in this work. For LTI models, the dimension of the truncated behavior $\mathcal{B}|_L$, i.e., the linear subspace of model trajectories of length L, depends on the *complexity* of the LTI model, which is characterized by three integer invariants of the model

$$\mathbf{c}(\mathcal{B}) := (\mathbf{m}(\mathcal{B}), \mathbf{n}(\mathcal{B}), \boldsymbol{\ell}(\mathcal{B})),$$

which are respectively the number of inputs, the order, and the lag of the model [20]. More specifically, $\dim \mathcal{B}|_L = \mathbf{m}(\mathcal{B})L + \mathbf{n}(\mathcal{B})$ for any $L \geq \boldsymbol{\ell}(\mathcal{B})$.

For SALPV models we consider the behavior $\mathcal{B}$ to be a subset of $(\mathbb{R}^q \times \mathbb{P})^{\mathbb{N}}$. Given the input-output representation (1), each element (w, ρ) of $(\mathbb{R}^q \times \mathbb{P})^{\mathbb{N}}$ is an element of the behavior $\mathcal{B}$ if it fulfills the given representation, i.e.,

$$\mathcal{B} := \left\{ (w, \rho) \in \left(\mathbb{R}^q \times \mathbb{P}\right)^{\mathbb{N}} \;\middle|\; (w, \rho) \text{ fulfills IO relationship (1)} \right\}.$$

We refer to the trajectory ρ as the scheduling trajectory. Equivalent to the LTI case, the complexity of a SALPV model is given by the same triple $(\mathbf{m}(\mathcal{B}), \mathbf{n}(\mathcal{B}), \boldsymbol{\ell}(\mathcal{B}))$ [21].

The projection of the behavior $\mathcal{B}$ on a given scheduling trajectory $\rho \in \mathbb{P}^{\mathbb{N}}$ is denoted by

$$\mathcal{B}_\rho := \left\{ w \in (\mathbb{R}^q)^{\mathbb{N}} \;\middle|\; (w, \rho) \in \mathcal{B} \right\}. \tag{5}$$

Since the above projection is only defined for infinite-length scheduling trajectories, for finite-length scheduling trajectories, we consider the cut operator to have precedence over the projection, i.e., for $\rho \in \mathbb{P}^T$ we have

$$\mathcal{B}_\rho|_T := \left\{ w \in (\mathbb{R}^q)^T \;\middle|\; (w, \rho) \in \mathcal{B}|_T \right\}. \tag{6}$$

Note that $\mathcal{B}_\rho$ and $\mathcal{B}_\rho|_T$ are both linear subspaces. Finally, for a given complexity $c = (m, n, \ell)$ we denote the set of SALPV behaviors with complexity at most c, with q-variate trajectories and with n_ρ scheduling variables by $\mathcal{P}_c^{q,n_\rho}$. Since most of the time both q and n_ρ are clear from context, we will often omit them from the notation, keeping only $\mathcal{P}_c$.

2.3 Kernel Representation

This section introduces the kernel representation and the specialized notation and terminology related to it. Although this work is concerned with the SALPV case, most of the established notation and conventions from the LTI case still hold. For this reason, we start by introducing the kernel representation for the LTI case. After that we illustrate variants of the notation for SALPV models, although the corresponding derivations follow later in this work. For further reading regarding system properties and identifiability based on the behavioral framework and the kernel representation in the LTI case, refer to [26].

2.3.1 LTI Case

For finite dimensional LTI models, it is possible to represent the subspace of possible trajectories $\mathcal{B} \subset (\mathbb{R}^q)^{\mathbb{N}}$ as the kernel of an operator $\mathcal{R}(\sigma)$:

$$\mathcal{B} = \ker \mathcal{R}(\sigma) := \{ w \mid \mathcal{R}(\sigma) w = 0 \}, \tag{7}$$

with $\mathcal{R}(\sigma)$ defined according to the polynomial matrix

$$\begin{aligned} \mathcal{R}(z) &= \begin{bmatrix} \mathcal{R}^1(z) \\ \vdots \\ \mathcal{R}^g(z) \end{bmatrix} \\ &= \begin{bmatrix} R_0^1 + R_1^1 z + \cdots + R_{\ell_1}^1 z^{\ell_1} \\ \vdots \\ R_0^g + R_1^g z + \cdots + R_{\ell_g}^g z^{\ell_g} \end{bmatrix} \in \mathbb{R}^{g \times q}[z], \end{aligned} \tag{8}$$

with each $R_j^i \in \mathbb{R}^{1\times q}$. For a minimal kernel representation, we have $g = \mathbf{p}(\mathcal{B}) := q - \mathbf{m}(\mathcal{B})$ linearly independent polynomials and have $\ell := \max(\ell_1, \ldots, \ell_g)$ minimal over all kernel representations, i.e., $\ell = \boldsymbol{\ell}(\mathcal{B})$. The operators corresponding to the different rows $\mathcal{R}^i(\sigma)$ are known as *annihilators* of $\mathcal{B}$. For a minimal kernel representation, the $q - \mathbf{m}(\mathcal{B})$ annihilators that form the operator $\mathcal{R}(\sigma)$ also form a basis for all annihilators of $\mathcal{B}$. In the sequel, we will assume that all discussed kernel representations are minimal.

For multiple-output models, additional considerations have to be made regarding the values of the lags $\ell_1, \ldots, \ell_p$. To avoid needless complication that detracts from the introduction of the required concepts, the remainder of this section will focus on the case where all lags are the same, i.e., $\ell_i = \boldsymbol{\ell}(\mathcal{B})$ for all $i = 1, \ldots, p$. More general cases will be discussed in Section 4. In the same-lag case, all annihilators in a minimal kernel representation have the same order, such that

$$\mathcal{R}(z) = R_0 + R_1 z + \cdots + R_\ell z^\ell,$$

with each $R_i \in \mathbb{R}^{p\times q}$. We refer to the $p(L-\ell)\times qL$ matrix

$$\mathcal{M}_L\big(\mathcal{R}(z)\big) := \begin{bmatrix} R_0 & R_1 & \cdots & R_\ell & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 \\ 0 & R_0 & \cdots & R_{\ell-1} & R_\ell & \cdots & 0 & 0 & \cdots & 0 & 0 \\ & & \vdots & & & & & & \vdots & & \\ 0 & 0 & \cdots & 0 & 0 & \cdots & R_0 & R_1 & \cdots & R_\ell & 0 \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & R_0 & \cdots & R_{\ell-1} & R_\ell \end{bmatrix}$$

as the *multiplication matrix* of polynomial matrix $\mathcal{R}(z)$. By convention, for $L \leq \ell$, we define the multiplication matrix as a $0 \times qL$ matrix. For a finite-length trajectory $w \in (\mathbb{R}^q)^T$, we have

$$w \in \mathcal{B}|_T \Leftrightarrow \mathcal{M}_T\big(\mathcal{R}(z)\big)\,\mathrm{col}\,(w) = 0.$$

We can introduce the coefficient matrix

$$R := \begin{bmatrix} R_0 & \cdots & R_\ell \end{bmatrix}.$$

In later parts of this work, we will use a parameterized version of this matrix R, so to differentiate this coefficient matrix from the original polynomial matrix, the polynomial matrix is always denoted by a calligraphic $\mathcal{R}$. Using the coefficient matrix R, the condition $\mathcal{M}_T\big(\mathcal{R}(z)\big)\,\mathrm{col}\,(w) = 0$ can also be written as

$$RH_{\ell+1}(w) = 0.$$

When making use of a row vector R that contains the polynomial coefficients of a kernel representation $\mathcal{R}(\sigma)$, we will also sometimes write $\mathcal{M}_T\big(\mathcal{R}(z)\big)$ as $\mathcal{M}_T(R)$. Note that the matrix R contains the model parameters of the kernel representation, and, even in the simplest single-output case, the kernel representation is not unique, since each multiple of R still results in an equivalent model.

2.3.2 SALPV Case

Instead of LTI models, this work is concerned with SALPV models. As described in [21], it is possible to rewrite the input-output representation of a SALPV model, as given by (1), as the kernel representation of a lifted LTI model. Define the lifted trajectory

$$w_{\mathrm{ext}}(t) := \rho'(t) \otimes w(t),$$

recalling that $\rho'(t) := \begin{bmatrix} 1 \\ \rho(t) \end{bmatrix}$. The original IO equation can be rewritten into the form

$$R_0 w_{\mathrm{ext}}(t) + R_1 w_{\mathrm{ext}}(t+1) + \cdots + R_\ell w_{\mathrm{ext}}(t+\ell) = 0,$$

valid for all $t \in (1, 2, \ldots, T-\ell)$, which is a kernel representation of the lifted model based on the polynomial matrix

$$\mathcal{R}(z) = R_0 + R_1 z + \cdots + R_\ell z^\ell, \tag{9}$$

with each $R_i \in \mathbb{R}^{g\times q_{\mathrm{ext}}}$, g the number of rows in (1a), and $q_{\mathrm{ext}} := (n_\rho + 1)q$. The derivation that results in this kernel representation can be found in Appendix A. Similar to the LTI case, this kernel representation is minimal if the rows are linearly independent and g is equal to the number of outputs. We will again assume that all discussed kernel representations are minimal. For the sake

$$\mathcal{M}_T(R,\rho) := \begin{bmatrix} R_0P_1 & R_1P_2 & \cdots & R_\ell P_{\ell+1} & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 \\ 0 & R_0P_2 & \cdots & R_{\ell-1}P_{\ell+1} & R_\ell P_{\ell+2} & \cdots & 0 & 0 & \cdots & 0 & 0 \\ & & \vdots & & & & & & \vdots & & \\ 0 & 0 & \cdots & 0 & 0 & \cdots & R_0P_{T-\ell} & R_1P_{T-\ell+1} & \cdots & R_\ell P_{T-1} & 0 \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & R_0P_{T-\ell+1} & \cdots & R_{\ell-1}P_{T-1} & R_\ell P_T \end{bmatrix}. \quad (12)$$

of simplicity, first we will focus on the case where all outputs have the same lag. The more general cases will be discussed in Section 4.

We can again define the matrix of coefficients

$$R := \begin{bmatrix} R_0 & \cdots & R_\ell \end{bmatrix},$$

with each $R_i \in \mathbb{R}^{p \times q_{\text{ext}}}$. New in this work is the introduction of a matrix $\mathcal{M}(\mathcal{R}(z), \rho)$ which, making use of a given scheduling sequence $\rho \in \mathbb{P}^T$, is the SALPV equivalent of the multiplication matrix $\mathcal{M}(\mathcal{R}(z))$ in the LTI case. We will often write $\mathcal{M}(R, \rho)$, making use of the matrix of coefficients R of this polynomial, similar to the LTI case. Also similar to the LTI case is the fact that the condition

$$\mathcal{M}_T(R,\rho)\,\mathrm{col}\,(w) = 0$$

guarantees that $w \in \mathcal{B}_\rho|_T$. As will be shown later in this work, we find that the condition

$$R_0 w_{\text{ext}}(t) + R_1 w_{\text{ext}}(t+1) + \cdots + R_\ell w_{\text{ext}}(t+\ell) = 0 \quad (10)$$

for all $t \in (1, 2, \ldots, T-\ell)$ is equivalent to the condition

$$\mathcal{M}_T(R,\rho)\,\mathrm{col}\,(w) = 0, \quad (11)$$

with $\mathcal{M}_T(R,\rho)$ as defined in (12) and P_i as defined in (4). Remark that $\mathcal{M}_T(R,\rho)$ has a similar structure to $\mathcal{M}_T(R)$ in the LTI case, but each block of the i-th block column is multiplied by a matrix based on $\rho(i)$. We will refer to the matrix $\mathcal{M}_L(R,\rho)$ as the *parameter-varying multiplication matrix* of polynomial $\mathcal{R}(z)$ based on the scheduling trajectory ρ. This matrix is not defined for $L > T$, with T the length of the scheduling trajectory ρ, and is taken to be a $0 \times qL$ matrix for $L \leq \ell$. Each block row of the $\mathcal{M}_T(R,\rho)$ matrix can be interpreted as the coefficients of a time-shifted version of the time-varying matrix polynomial

$$\mathcal{R}(z,t) = R_0P_t + R_1P_{t+1}z + \ldots + R_\ell P_{t+\ell}z^\ell. \quad (13)$$

Remark that despite the similarities between the LTI and SALPV cases, where the SALPV kernel representation is just the kernel representation of a lifted LTI model, there is need for a distinction between the two. Compared to the LTI kernel representation, not all trajectories $w_{\text{ext}} \in (\mathbb{R}^{q_{\text{ext}}})^T$ that fulfill the lifted kernel representation (10) are valid trajectories for the SALPV model. The reason for this is that the lifting enforces specific structure in the additional inputs, i.e., the Kronecker products. This additional structure needs to be handled to correctly make use of the kernel representation, such that existing LTI model identification methods are not sufficient. The properties of the parameter-varying multiplication matrix in (12) will play an important role in this work and will be discussed in more detail in Section 3, where the equivalence in (11) will follow from Lemma 1.

To conclude this section, for single-output SALPV models, the condition $w \in \mathcal{B}_\rho|_T$, with $\mathcal{B}_\rho|_T \subseteq (\mathbb{R}^q)^T$, can be reformulated in three equivalent ways using the kernel representation:

- $\mathcal{R}(\sigma) w_{\text{ext}} = 0$,
- $R H_{\ell+1}(w_{\text{ext}}) = 0$,
- $\mathcal{M}_T(R,\rho)\,\mathrm{col}\,(w) = 0$.

Given that the matrix R, which contains the polynomial coefficients, forms our model parameters, it is useful to consider the conditions that a general matrix R should fulfill to correspond to a valid and minimal kernel representation. In general, the only requirement for the matrix R to be a valid kernel representation is that R has full row rank. For a minimal kernel representation, it is also required for R to have the correct number of rows for the given model complexity (as discussed above), and for the same-lag assumption we require every row of R_ℓ to be nonzero. Remark that just being a valid kernel representation does not enforce system properties such as causality or stability, which require further limitations on the R matrix. We will discuss these requirements and propose a parameterization of the R matrix that guarantees causality in Section 3.1.

2.4 Problem Formulation

In real applications, there are situations where we do not have access to the exact input signal applied to the to-be-modeled plant. One example is that the to-be-modeled system is part of a larger system such that the input is applied indirectly, and only noisy measurements of the input signals are available. This is a common occurrence when measuring physical phenomena, where the input is generated by an actuator that translates the digital commands into the model input (e.g., a current) and we do not want to include this actuator in the model. When additionally dealing with output noise, this setting is referred to as the EIV noise setting. Refer to [7] for a more detailed discussion.

For the remainder of this work, consider an unknown SALPV model $\mathcal{B}$ of known complexity c and a given pair of signals $(w, \rho) \in (\mathbb{R}^q \times \mathbb{P})^T$. We assume that the signal w can be expressed as $\bar{w} + v$, where $\bar{w} \in \mathcal{B}_\rho|_T$ is a model trajectory and $v \in (\mathbb{R}^q)^T$ is a zero mean Gaussian signal with covariance $\eta^2 I_{qT}$. this implies that the samples $v(t)$ are independent across time, and the components of each $v(t)$ are independent and identically distributed. The nonnegative scalar η^2 denotes the noise variance. Although the noise covariance matrix is assumed to be a multiple of the identity matrix here, this assumption will be relaxed to allow for general positive definite covariance matrices in Section 4.1. In the behavioral setting, we can consider the problem of model identification from noisy data as a joint optimization over the model set and the noise-free trajectory estimate. For single-output models, this results in the following problem:

Problem 1 (Same-Lag Identification Problem)**.** Given a complexity $c = (m, p\ell, \ell)$, a trajectory $w \in (\mathbb{R}^q)^T$, and a scheduling sequence $\rho \in \mathbb{P}^T$, solve the following optimization problem:

$$\underset{\hat{w}, \mathcal{B}}{\text{minimize}} \ \|\hat{w} - w\|_2^2 \quad \text{s.t. } \hat{w} \in \mathcal{B}_\rho|_T, \ \mathcal{B} \in \mathcal{P}_c^{q, n_\rho}. \tag{14}$$

The goal is to find a trajectory $\hat{w} \in \mathcal{B}_\rho|_T$, consistent with a model $\mathcal{B} \in \mathcal{P}_c$, which minimizes the squared distance $\|\hat{w} - w\|_2^2$. In this work we consider cases where w is a trajectory of an underlying SALPV model $\bar{\mathcal{B}} \in \mathcal{P}_c$, but corrupted by noise. The scheduling variables ρ are assumed to be known exactly, not corrupted by noise. Note that in the considered noise setting, we consider specifically i.i.d. Gaussian noise with a covariance matrix proportional to the identity matrix. The reason for this is that only under these noise assumptions, the solution to optimization problem (14) is statistically optimal. More specifically, this solution is the maximum likelihood estimate under the given noise assumptions. In Section 4 we will introduce a generalization of the problem statement that allows us to relax the noise assumptions somewhat without losing statistical optimality.

3 Main Result

Problem formulation (14) is general, but cannot be solved directly because the optimization variable $\mathcal{B}$ is an infinite set. However, $\mathcal{B} \in \mathcal{P}_c$ is finitely parameterizable. By making use of a model representation we can instead optimize over the finite number of model parameters. The first subsection will reformulate the set inclusion constraint using the kernel representation, resulting in an optimization problem over $\hat{w}$ and a finite-dimensional parameter vector $\theta \in \mathbb{R}^{n_\theta}$. The resulting problem is solveable, but scales poorly due to the length of $\hat{w}$ increasing with the number of samples. Minimizing the number of parameters is especially important for the nonconvex optimization this parameter estimation problem turns out to be. The second subsection will reformulate the problem into a form that is well suited for numerical optimization. Solving the resulting problem in a scalable manner requires exploiting the structure of the resulting matrices. These structures are analyzed in the third subsection. Finally, the fourth subsection details an algorithm that can be used to solve the resulting estimation problem efficiently.

3.1 Parameterizing The Model Set

3.1.1 Choosing a Model Representation

From Section 2.3, we know that the set $\mathcal{B} \in \mathcal{P}_c$, with $c = (m, n, \ell)$, can be parameterized using the kernel representation

$$\mathcal{B}\big(\mathcal{R}(\sigma)\big) := \left\{ (w, \rho) \in (\mathbb{R}^q \times \mathbb{P})^{\mathbb{N}} \;\middle|\; \mathcal{R}(\sigma)(\rho' \otimes w) = 0 \right\},$$

with the operator $\mathcal{R}(\sigma)$ based on the polynomial

$$\mathcal{R}(z) = R_0 + R_1 z + \cdots + R_\ell z^\ell$$

with each $R_i \in \mathbb{R}^{p\times(n_\rho+1)q}$. Define the matrix

$$R := \begin{bmatrix} R_0 & R_1 & \cdots & R_\ell \end{bmatrix},$$

which consists of the coefficients of the kernel polynomial. This matrix contains exactly the model parameters. Defining the set $\mathcal{B}$ as a function of the matrix R automatically guarantees that $\mathcal{B} \in \mathcal{P}_c$, and optimizing over $\mathcal{B}$ can be done by optimizing over R instead, as long as R contains the coefficients of a valid kernel representation. At this point, the optimization problem looks as follows:

$$\begin{aligned} &\underset{\hat{w},R}{\text{minimize}} \ \ \|\hat{w} - w\|_2^2 \\ &\text{s.t. } RH_{\ell+1}(\rho' \otimes \hat{w}) = 0, \ \mathcal{R}(z) \text{ full row rank.} \end{aligned} \tag{15}$$

Other than enforcing full row rank, Another issue with using R directly as optimization parameter is the problem of non-uniqueness. For each matrix R, the matrix MR, with M a nonsingular matrix, represents the same model, and will as such result in the same value of the cost function.

Finally, when specializing for physical systems, there are fundamental properties we want to enforce, the most important of which is causality. The next subsection will look into what structure is required in the kernel coefficients R_i to solve all three issues mentioned above.

3.1.2 Enforcing Desired Model Properties

Enforcing model properties means enforcing structure in the R matrix. To allow for this, we introduce a parameterization of the R matrix in function of a parameter vector $\theta \in \mathbb{R}^{n_\theta}$. The details of this parameterization are worked out in the remainder of this section.

When considering what model properties are desired, we will specialize for physical systems, for which we assume to be given an input-output partitioning. This is reasonable since estimating a model based on noisy measurements is the main focus of this work. Physical systems are inherently causal, which means that the resulting models should be as well, making causality our main desired property. We will start by discussing the LTI case, which will form the basis for the SALPV case.

Enforcing causality in the LTI case is simple. Given an input-output partitioning of $w(t)$, we can write

$$R_\ell w(t) = R_{\ell,y} y(t) + R_{\ell,u} u(t).$$

Causality requires that $R_{\ell,y} \in \mathbb{R}^{p\times p}$ is non-singular, since otherwise there would exist a matrix R' that represents the same model, but where matrix $R'_{\ell,y}$ has a zero row. This breaks causality, since all rows of R'_ℓ are nonzero under our same-lag requirement, meaning that the corresponding row of $R'_{\ell,u}$ is nonzero. Since the null space of R is invariant under left multiplication of R with a non-singular matrix, normalization is often introduced with respect to $R_{\ell,y}$, resulting in a unique representation, such that we enforce

$$R_\ell w(t) = y(t) + R_{\ell,u} u(t).$$

Note that all three goals, namely R full row rank, uniqueness of the representation, and causality, are guaranteed using this parameterization.

For SALPV models, a similar reasoning holds, but the existence and freedom of the scheduling variables adds additional complexity. Doing a similar input-output partitioning of $w_{\text{ext}}(t)$ gives

$$R_\ell w_{\text{ext}}(t) = R_{\ell,y}\big(\rho'(t) \otimes y(t)\big) + R_{\ell,u}\big(\rho'(t) \otimes u(t)\big).$$

The first Kronecker product is equivalent to $(\rho'(t) \otimes I_p)y(t)$. As was the case for LTI models, given that we are interested in minimal kernel representations, the necessary condition for causality is for $R_{\ell,y}(\rho'(t) \otimes I_p)$ to be non-singular, where $R_{\ell,y}$ is now a $p \times (n_\rho + 1)p$-dimensional matrix. Notice that in contrast to the LTI case, just making $R_{\ell,y}$ non-singular is no longer sufficient (or possible, given that the matrix is no longer square), since the product with $(\rho'(t) \otimes I_p)$ could still make the coefficients of $y(t)$ zero for some values of $\rho'(t)$. A sufficient, albeit not necessary condition for causality to hold for all $\rho'(t)$ is to force $R_{\ell,y}(\rho'(t) \otimes I_p) = I_p$, which is equivalent to forcing $A_0\big(\rho(t)\big) = I_p$ in (1a). Choosing this structure not only fixes a '1' element in each row of the R_ℓ matrix, but also forces n_ρ other elements of each row to be zero. Although this choice also removes

valid causal models, it is a common choice in literature (see e.g., [10, 15, 16]) and also enforces our three goals.

In conclusion, to handle identification of minimal kernel representations for models of physical systems, we require some elements of R_ℓ to be fixed zeros or ones, while all other elements are free. The parameter vector θ consists of exactly the n_θ free elements of R.

3.2 Problem Reformulation

Before reformulating the problem, we recall that, based on the previous section, to guarantee our desired model properties, we introduce a parameterization $R(\theta)$ of the kernel matrix R, where $\theta \in \mathbb{R}^{n_\theta}$ represents the free elements. Most elements of R are free, except for the final block R_ℓ, where the coefficients corresponding to $y(t)$ are fixed as I_p, and all coefficients corresponding to $\rho(t) \otimes y(t)$ are fixed as zero. Instead of optimizing over R, we will optimize over this parameter vector $\theta \in \mathbb{R}^{n_\theta}$.

Starting from optimization problem (15) and making use of the parametrization proposed in the previous section, we end up with the following problem:

$$\underset{\hat{w},\theta}{\text{minimize}} \ \|\hat{w} - w\|_2^2 \quad \text{s.t. } R(\theta) H_{\ell+1}(\rho' \otimes \hat{w}) = 0. \tag{16}$$

Since both sides of the matrix multiplication depend on optimization variables ($\hat{w}$ and θ), the equality constraints are bilinear, such that the problem is nonconvex. Worse is that the number of optimization variables scales with the number of samples, resulting in possibly thousands of optimization variables. The goal is to reformulate the above problem into an equivalent problem with only the vector θ as an optimization variable, whose length depends only on q, the model complexity c, and the number of scheduling variables n_ρ.

We start off by introducing a lemma that vectorizes equality constraints of the form

$$R H_{\ell+1}(\rho' \otimes x) = B,$$

with B some suitably sized matrix, into equality constraints that are linear in x. Using this lemma, we then prove the main theorem.

Lemma 1. *Given a scheduling sequence $\rho \in \mathbb{P}^T$, an unknown trajectory $x \in (\mathbb{R}^q)^T$, a matrix $B \in \mathbb{R}^{g \times T - \ell}$, and a matrix*

$$R = \begin{bmatrix} R_0 & R_1 & \cdots & R_\ell \end{bmatrix},$$

with each $R_i \in \mathbb{R}^{g \times (n_\rho + 1)q}$, the system of equations

$$R H_{\ell+1}(\rho' \otimes x) = B$$

is equivalent to the system of equations

$$G(R) \operatorname{col}(x) = \operatorname{col}(B),$$

with

$$G(R) := \mathcal{M}_T(R, \rho).$$

Furthermore, if the matrix R is $R(\theta)$, parameterized as in Section 3.1.2, the matrix $G\big(R(\theta)\big)$ has full row rank for all parameter values θ.

Proof. The proof is given in Appendix B. □

Theorem 1 (Same-Lag Problem Reformulation)**.** *When $R(\theta)$ is parameterized as described in Section 3.1.2, solving Problem* (14) *is equivalent to solving*

$$\underset{\theta \in \mathbb{R}^{n_\theta}}{\text{minimize}} \ \|\Delta w^*(\theta)\|_2^2, \tag{17}$$

with

$$\Delta w^*(\theta) := G\big(R(\theta)\big)^\dagger s(\theta), \tag{18a}$$

$$s(\theta) := \operatorname{col}\big(R(\theta) H_{\ell+1}(w_{\text{ext}})\big), \tag{18b}$$

with $G\big(R(\theta)\big)$ as in Lemma 1 and with $G\big(R(\theta)\big)^\dagger$ denoting the right pseudoinverse of $G\big(R(\theta)\big)$.

Proof. The proof is given in Appendix C. □

At first glance, Theorem 1 does not seem to help much with solving the problem, since the resulting problem is still non-convex. Furthermore, the matrix $G\big(R(\theta)\big)$, which plays a big part in the problem formulation, scales poorly with the number of samples. The advantage of the resulting formulation is the fact that the nonlinear least squares problem only needs to be solved for n_θ parameters, compared to the $n_\theta + qT$ parameters we started with. These additional parameters are still there, but hidden in a closed-form solution, and can be solved for by evaluating $\Delta w^*(\theta)$, which is itself the least-norm solution of a linear least-squares problem. The problematic scaling of $G\big(R(\theta)\big)$ can also be mitigated by exploiting the sparse structure of this matrix, which will be discussed in the next section.

Another remark is the need for a parameterization of R. Although all previous derivations hold even without a specific parameterization, Theorem 1 is the first place where a specific parameterization is required. The reason for this is to guarantee that every found R matrix represents a valid kernel representation. The full row rank condition on $G\big(R(\theta)\big)$, which is guaranteed with the chosen parameterization, is also crucial to have, since it guarantees a closed-form solution of the pseudoinverse without requiring expensive SVD computations for which it is harder to exploit the available structure.

3.3 Exploiting the structure

3.3.1 Structure of the SALPV multiplication matrix

Consider the matrix $G\big(R(\theta)\big) = \mathcal{M}_T\big(R(\theta), \rho\big)$ in (12). Since $R(\theta)$ is parameterized by θ, each $R_i(\theta)$ is also parameterized by θ. The matrix $G\big(R(\theta)\big)$ is a $T - \ell$ by T block matrix consisting of p-by-q-sized blocks, such that $G\big(R(\theta)\big) \in \mathbb{R}^{p(T-\ell)\times qT}$. Remark that both the number of rows and number of columns scale linearly with T, the number of samples. Each block row consists of exactly $\ell + 1$ non-zero blocks, such that the total number of non-zero blocks is $p(T - \ell)(\ell + 1) \leq p(T - \ell)T$.

When the matrix $G\big(R(\theta)\big)$ has full row rank, as is the case with our proposed parametrization according to Lemma 1, its pseudoinverse is a right inverse, and has a closed form expression:

$$\begin{aligned} G\big(R(\theta)\big)^\dagger &= G\big(R(\theta)\big)^\top \Big(G\big(R(\theta)\big)G\big(R(\theta)\big)^\top\Big)^{-1} \\ &= G\big(R(\theta)\big)^\top \Gamma\big(R(\theta)\big)^{-1}, \end{aligned}$$

with

$$\Gamma\big(R(\theta)\big) := G\big(R(\theta)\big)G\big(R(\theta)\big)^\top. \tag{19}$$

This final matrix $\Gamma\big(R(\theta)\big)$, which we refer to as the Gram matrix, also scales in both width and height with the number of samples, and has to be inverted to compute the pseudoinverse (or at least be used to solve a linear system of equations). For this reason, looking at its structure in more detail will be important for optimizing the solver.

3.3.2 Structure of the Gram matrix

Other than the structure of $G\big(R(\theta)\big)$, we are also interested in the structure of $\Gamma\big(R(\theta)\big) := G\big(R(\theta)\big)G\big(R(\theta)\big)^\top$, or more specifically, efficient ways to solve systems of equations with this matrix.

To more conveniently formulate the resulting structure, we introduce two additional parameterized matrices

$$\begin{aligned} M_{i,j} &:= \operatorname{blkdiag}(P_i P_i^\top, P_{i+1} P_{i+1}^\top, \ldots, P_j P_j^\top), \\ V_{i,i+\Delta i} &:= \begin{bmatrix} 0_{q_{\text{ext}}\Delta i \times q_{\text{ext}}(\ell+1-\Delta i)} & 0_{q_{\text{ext}}\Delta i \times q_{\text{ext}}\Delta i} \\ M_{i+\Delta i, i+\ell} & 0_{q_{\text{ext}}(\ell+1-\Delta i)\times q_{\text{ext}}\Delta i} \end{bmatrix}. \end{aligned} \tag{20}$$

The matrix $V_{i,i+\Delta i}$ is a square matrix of size $q_{\text{ext}}(\ell + 1)$, and is properly defined using the above equation for Δi in the interval $(0, 1, \ldots, \ell)$, and for values of i between 0 and $T-\ell-\Delta i$, both inclusive. For other values of i and Δi, we have $V_{i,i+\Delta i} = 0$, and for all cases we have $V_{i+\Delta i,i} = V_{i,i+\Delta i}^\top$. The matrix $M_{i,j}$ is well defined for all values of i less than or equal to j, and both between 0 (exclusive) and T (inclusive). Using these matrices, we arrive to the following structure of $p \times p$ blocks:

$$\Gamma\big(R(\theta)\big) := \begin{bmatrix} R(\theta)V_{1,1}R^\top(\theta) & \cdots & R(\theta)V_{1,b}R^\top(\theta) \\ \vdots & & \vdots \\ R(\theta)V_{b,1}R^\top(\theta) & \cdots & R(\theta)V_{b,b}R^\top(\theta) \end{bmatrix}, \tag{21}$$

with $b = T - \ell$ the number of columns of the block Hankel matrix $H_{\ell+1}(w_{\text{ext}})$. Refer to Appendix D for a full derivation of this structure. From the definition, a few properties are clear to see. Firstly, the matrix is symmetric, as expected. Secondly, while the diagonal blocks $V_{i,i}$ are block diagonal themselves, the further the blocks are from the main diagonal, the sparser they become, resulting in a block-banded matrix with ℓ additional block diagonals on each side of the main block diagonal. This block-banded structure makes it such that the total number of non-zero elements is less than $p(2\ell+1)(T-\ell)$, which scales linearly with T. With the matrix structures analyzed, the next section goes into more detail about how to solve the reformulated problem (17).

3.4 Proposed Algorithm

Problem (17) is non-convex, and more specifically a nonlinear least squares problem, making it difficult to find the global minimum. It is, however, quite easy to find a local minimum using gradient-based methods. The downside of local minima is that the end result depends on having a good initial estimate of the true parameter vector θ. Any solver that can handle nonlinear least squares problems works, but given how the matrices that occur in the problem scale with the number of samples, it is best to make use of a simple method that allows us to exploit the structure of the matrices without adding too much overhead.

In this work, we propose the use of Levenberg-Marquardt, which is a robust gradient-based algorithm with global convergence guarantees that has been used to solve nonlinear least squares problems for many years. There exist many variations, so the remainder of this section will detail the variant used here, which is based on [27]. Consider our cost function

$$f(\theta) := \|\Delta w^*(\theta)\|_2^2 .$$

The basic idea is to approximate $\Delta w^*(\theta_j + \Delta\theta)$ by the expression

$$\Delta w^*(\theta_j) + J^\top(\theta_j)\Delta\theta,$$

with $J(\theta_j)$ the Jacobian of $\Delta w^*(\theta)$ at θ_j. Levenberg-Marquardt then uses this approximation to choose the optimal next step $\Delta\theta$ by minimizing the expression

$$m_j(\Delta\theta) := \left\|\Delta w^*(\theta_j) + J^\top(\theta_j)\Delta\theta\right\|_2^2 + \gamma_j \|\Delta\theta\|_2^2 , \tag{22}$$

which is just $f(\theta_j + \Delta\theta)$ with the approximation plugged in and with an additional regularization term with real scalar weight γ_j. A common choice is to tune a regularization parameter μ_j and letting $\gamma_j = \mu_j f(\theta_j)$. After each iteration we obtain a value $\Delta\theta$ and compute the ratio

$$\kappa = \frac{f(\theta) - f(\theta + \Delta\theta)}{f(\theta) - \|\Delta w^*(\theta_j) + J^\top(\theta_j)\Delta\theta\|_2^2}, \tag{23}$$

which is a figure of merit for the quality of our approximation, and represents the ratio between the actual reduction and predicted reduction. If the ratio is above some user-specified bound $\kappa_{\min}$ we consider the approximation passable and accept the step, setting $\theta_{j+1} = \theta_j + \Delta\theta$. If the step is not accepted, we set $\theta_{j+1} = \theta_j$. When the step is accepted we also set $\mu_{j+1} = \frac{\mu_j}{\alpha}$, with α a user-specified scalar greater than one, otherwise we set $\mu_{j+1} = \alpha\mu_j$. In any case we set $\gamma_{j+1} = \mu_{j+1} f(\theta_{j+1})$. The implementation given in [27] instead chooses to update μ_{j+1} as $\frac{\bar{\mu}}{\alpha}$ when the step is accepted, with $\bar{\mu}$ the last successful μ_j beforehand. The reasoning for this is to balance global and local convergence properties, but the claims made in this paper do not require this choice. The result of updating our regularization parameter as described is that when the approximation is bad we bias our next attempt to take a smaller step, since the approximation will hold better the closer we stay to the original point. On the contrary, if the step is accepted we can slightly relax the regularization for the next iteration.

It can be seen that compared to simple gradient-based methods such as *gradient descent*, we might spend multiple iterations without taking a step, highlighting the need for efficient computations of both the function evaluation $f(\theta)$, and of the Jacobian matrix $J(\theta)$ of $\Delta w^*(\theta)$. Without considering structure in the matrices, the expressions for these terms are expensive to compute, scaling poorly with the number of samples. In the following parts, we will describe how to efficiently compute these terms while exploiting the relevant structures, and show what the complexity of these computations is using big O notation for the number of floating point operations (FLOPs).

3.4.1 Evaluation of the cost function

Consider the solution z of the system of equations

$$\Gamma\big(R(\theta)\big)z = s(\theta), \tag{24}$$

such that $z(\theta) := \Gamma\big(R(\theta)\big)^{-1}s(\theta)$. We find that

$$\Delta w^*(\theta) := G^\top(R(\theta))\Gamma\big(R(\theta)\big)^{-1}s(\theta) = G^\top(R(\theta))z(\theta),$$

such that

$$\begin{aligned} f(\theta) &:= \|\Delta w^*(\theta)\|_2^2 \\ &= z^\top(\theta)G\big(R(\theta)\big)G\big(R(\theta)\big)^\top z(\theta) \\ &= z^\top(\theta)\Gamma\big(R(\theta)\big)z(\theta) \\ &= z^\top(\theta)s(\theta). \end{aligned} \tag{25}$$

The vector $z(\theta)$ can be found efficiently from $s(\theta)$ and $G\big(R(\theta)\big)$ by exploiting the block-banded structure of $\Gamma\big(R(\theta)\big)$. Computing $s(\theta)$ can be done in $O(pq_{\text{ext}}\ell T)$ FLOPs, while $z(\theta)$ can be computed in $O((p\ell)^2T)$ FLOPs if $s(\theta)$ and $\Gamma\big(R(\theta)\big)$ are available. Refer to Appendix E for the the detailed derivation.

3.4.2 Computation of the Jacobian

Levenberg-Marquardt requires an estimate of the Jacobian of $\Delta w^*(\theta)$ with respect to θ. Although it is possible to directly derive a formula for the Jacobian with respect to θ, this section opts to derive the Jacobian with respect to the components of R instead. This is done for two reasons: Firstly, it highlights the parallel with existing results for the linear case [23]; Secondly, the expressions for $G\big(R(\theta)\big)$ and $s(\theta)$ are simpler when expressed in function of the matrix $R(\theta)$ directly. The Jacobian with respect to θ can be recovered by removing the columns corresponding to the fixed elements of $R(\theta)$. For the remainder of this section, all vectors and matrices depending on θ are expressed in function of R instead, such that

$$\Delta w^*(R) = G(R)^\top\Gamma(R)^{-1}s(R).$$

Keeping this in mind, we find the partial derivative of $\Delta w^*(R)$ to be

$$\frac{\partial \Delta w^*}{\partial R_{i,j}} = \frac{\partial G^\top}{\partial R_{i,j}}z(R) + G^\top(R)\Gamma^{-1}(R)z_{i,j}, \tag{26}$$

with $R_{i,j}$ the element of R on row i and column j, and with

$$z_{i,j} := \frac{\partial s(R)}{\partial R_{i,j}} - \frac{\partial \Gamma(R)}{\partial R_{i,j}}z(R).$$

This formula for the partial derivatives of Δw^* is equivalent to the corresponding formulation for the LTI case introduced in [23]. We start by introducing a matrix Z with the same number of rows as the matrix R, defined such that $z(R)$ is the vertical concatenation of the columns of Z from left to right. Following the same idea as in [23], we can then express $z_{i,j}$ as

$$z_{i,j} := \Big((H_L^\top(w_d)_{[:,j]} - z_j^{(1)}) \otimes e^{(i)}\Big) - z_{i,j}^{(2)},$$

with

$$z_j^{(1)} := \sum_{k=1}^{b} \begin{bmatrix} (e^{(j)})^\top V_{1,k}R^\top Z_{[:,k]} \\ (e^{(j)})^\top V_{2,k}R^\top Z_{[:,k]} \\ \vdots \\ (e^{(j)})^\top V_{b,k}R^\top Z_{[:,k]} \end{bmatrix},$$

and

$$z_{i,j}^{(2)} := \sum_{k=1}^{b} Z_{[i,k]} \begin{bmatrix} RV_{1,k}e^{(j)} \\ RV_{2,k}e^{(j)} \\ \vdots \\ RV_{b,k}e^{(j)} \end{bmatrix}.$$

Combined, all computations for the Jacobian matrix require $O(pq_{\text{ext}}^2\ell^3T)$ FLOPs, which is the most expensive part of the entire algorithm. A detailed explanation that leads to this complexity can be found in Appendix E.

3.4.3 Full Algorithm

The full algorithm is summarized in Algorithm 1.

Algorithm 1 Summary of simple LPV-SLRA

procedure LPV-SLRA(θ_0, w, c, μ_0)
  set γ_0 to $\mu_0 f(\theta_0)$
  while $j < j_{\max}$ **do**
    Compute the Jacobian $J(\theta_j)$ using (26)
    Obtain step $\Delta\theta$ from minimizing (22)
    if $\kappa_{\min} \leq \frac{f(\theta_j) - f(\theta_j + \Delta\theta)}{f(\theta_j) - \|\Delta w^*(\theta_j) + J^\top(\theta_j)\Delta\theta\|_2^2}$ **then**
      Set μ_{j+1} to $\max(\mu_{\min}, \frac{\bar{\mu}}{\alpha})$
      Set θ_{j+1} to $\theta_j + \Delta\theta$
      Update $\bar{\mu}$ to μ_j
    else
      Set μ_{j+1} to $\alpha\mu_j$
      Keep θ_{j+1} the same
    end if
    Update γ_{j+1}
  end while
  return θ_j
end procedure

Under the assumption that $\sigma_{\min}\big(G\big(R(\theta)\big)\big)$ is uniformly lower bounded, the proposed method is guaranteed to converge to a stationary point. This fact follows from the following theorem:

Theorem 2 (Convergence of Algorithm 1)**.** *Assume that the smallest singular value of $G\big(R(\theta)\big)$ is uniformly lower bounded, i.e., there exists a $\sigma_G > 0$ such that $\sigma_{\min}(G\big(R(\theta)\big)) \geq \sigma_G$ for all $\theta \in \mathbb{R}^{n_\theta}$. Then, for any initial parameter vector $\theta_0 \in \mathbb{R}^{n_\theta}$, the sequence of parameter vectors $(\theta_0, \theta_1, \ldots)$ generated by Algorithm 1 converges to a first-order stationary point θ^* of $f(\theta)$, i.e.,*

$$\nabla f(\theta^*) = 0.$$

Proof. The proof is given in Appendix F. □

4 Extensions

The following subsections contain extensions of the original algorithm which are useful for specific applications. These generalizations are kept separate from the original algorithm to make each part easier to understand. The first extension makes the cost function more general, adding a weighting matrix to it. The second extension allows the algorithm to handle more general cases of lags. The third extension allows for fixing specific elements such that they are kept constant during the optimization procedure. The final extension allows the use of multiple separate data sets at the same time.

4.1 Weighted cost function

The results of this work still hold when replacing the squared 2-norm $\|\Delta w^*\|_2^2$ with a weighted 2-norm $(\Delta w^*)^\top W \Delta w^*$, where W is positive definite. The weighting matrix W can be used to include prior knowledge regarding the noise covariance. In Section 2.4, we considered noise with a positive definite covariance matrix proportional to the identity matrix, but if we instead consider a more general case, where the covariance matrix at time t is given by Σ_t, a $q \times q$ positive definite matrix, i.e.,

$$v(t) \in \mathcal{N}(0, \Sigma_t),$$

for which the optimal choice of W becomes

$$W = \text{blkdiag}(\Sigma_1^{-1}, \ldots, \Sigma_T^{-1}),$$

which is block diagonal with $q \times q$ blocks. The choice of W is optimal in the sense that the resulting optimization problem gives the maximum likelihood estimate of the model parameters under this more general noise setting. For the remainder of this section we will continue using this noise setting.

To obtain the convenient form of problem (17) in the proof of Theorem 1 (found in Appendix C), we introduced the substitution $\Delta w := \hat{w} - w$. Now consider the Cholesky factorization of W

$$W = L_W^\top L_W.$$

Changing the substitution to $\Delta w := L_W(\hat{w} - w)$, rewritten as $w = \hat{w} - L_W^{-1}\Delta w$, the weighted cost function

$$\|\hat{w} - w\|_W^2$$

becomes

$$\Delta w^\top L_W^{-\top} W L_W^{-1} \Delta w = \|\Delta w\|_2^2 .$$

We can see that this new substitution results in exactly the same problem as the original one after applying Lemma 1 in the proof, including an unweighted cost function after using the above observation, except that the vectorized equality becomes $G\big(R(\theta)\big)L_W^{-1}\Delta w = s(R)$ with the same $s(R)$, and $G\big(R(\theta)\big)$. The matrix $G\big(R(\theta)\big)L_W^{-1}$ can be seen as a re-weighted G matrix, and is still full row rank since the matrix W is assumed to be positive definite. Furthermore, taking into account the structure of W, the structure of the re-weighted G matrix remains the same, but each matrix P_i is right multiplied with $L_{\Sigma_i}^{-1}$, the inverse Cholesky factor of Σ_i^{-1}, which results in a re-weighted form of P.

4.2 Extension to more general MIMO Models

For MIMO models, it is known that the number of rows in the polynomial matrix of a minimal kernel representation is exactly the number of outputs of the model. The complication comes from the fact that each of the rows in the polynomial matrix, i.e., each annihilator, can have a different order, i.e., the different outputs can have a different lag. Consider the case with p outputs, and denote the lag of the i-th output by ℓ_i, with $\ell := \max(\ell_1, \ell_2, \ldots, \ell_p)$. Recall the original polynomial definition of the kernel representation from (8)

$$\mathcal{R}(z) = \begin{bmatrix} \mathcal{R}^1(z) \\ \vdots \\ \mathcal{R}^p(z) \end{bmatrix} = \begin{bmatrix} R_0^1 + R_1^1 z + \cdots + R_{\ell_1}^1 z^{\ell_1} \\ \vdots \\ R_0^p + R_1^p z + \cdots + R_{\ell_p}^p z^{\ell_p} \end{bmatrix} \in \mathbb{R}^{p\times q}[z],$$

There are three cases to consider for the MIMO extension, which are increasingly difficult to handle:

1. All ℓ_i are equal to ℓ.
2. All ℓ_i are known.
3. Only ℓ, an upper bound for each ℓ_i, is known.

These cases are important since the lag directly corresponds to the model order, which is exactly the sum of all lags [20, Proposition X.4], i.e.,

$$n := \ell_1 + \cdots + \ell_p.$$

The first option is the most common in the literature, and has been assumed in this work as well up to this point. The downside is that the model order is fixed to $n = p\ell$, i.e., only specific values of the model order are possible. The second option requires changes to the definitions of both $G\big(R(\theta)\big)$ and $\Gamma\big(R(\theta)\big)$ among others. Remark that the first option is a special case of the second option. The third option will not be considered in detail, but will be briefly discussed at the end of this section.

For the second option, consider the coefficient matrix of the i-th row

$$R^i := \begin{bmatrix} R_0^i & R_1^i & \cdots & R_{\ell_i}^i \end{bmatrix} \in \mathbb{R}^{1\times(\ell_i+1)q_{\text{ext}}}.$$

We still require all considerations from Section 3.1.2 to hold, even with different lags. In practice, this means that the relationship

$$R_{\ell_i}^i(\theta_i)P_t w(t) = y_i(t) + R_{\ell_i,u}\big(\rho'(t)\otimes u(t)\big)$$

should hold for all $t = (1, 2, \ldots, T)$ and $i = (1, 2, \ldots, p)$. If we define the complete coefficient matrix as

$$R(\theta) := \begin{bmatrix} 0_{1\times(\ell-\ell_1)q_{\text{ext}}} & R^1(\theta_1) \\ \vdots & \\ 0_{1\times(\ell-\ell_p)q_{\text{ext}}} & R^p(\theta_p) \end{bmatrix},$$

such that the ends of each $R^i(\theta_i)$ are aligned, and with θ being the concatenation of all θ_i's, The parameterization of each $R^i(\theta_i)$ follows directly from the procedure in Section 3.1.2 applied to $R(\theta)$. The extension for the second point is given by Theorem 3 below.

Theorem 3 (General lag extension of Theorem 1)**.** *If the model has p outputs, the results of Theorem 1 still hold, as long as the matrix $G\big(R(\theta)\big)$ and the vector $s(\theta)$ are replaced by*

$$G\big(R(\theta)\big) := \begin{bmatrix} \mathcal{M}_T(R^1(\theta_1), \rho) \\ \mathcal{M}_T(R^2(\theta_2), \rho) \\ \vdots \\ \mathcal{M}_T(R^p(\theta_p), \rho) \end{bmatrix}, \tag{27}$$

$$s(\theta) := \begin{bmatrix} \operatorname{col}\left(R^1(\theta_1) H_{\ell_1+1}(\rho' \otimes w)\right) \\ \operatorname{col}\left(R^2(\theta_2) H_{\ell_2+1}(\rho' \otimes w)\right) \\ \vdots \\ \operatorname{col}\left(R^p(\theta_p) H_{\ell_p+1}(\rho' \otimes w)\right) \end{bmatrix}. \tag{28}$$

Furthermore, if each row vector $R^i(\theta_i)$ is parameterized as described above, the new matrix $G(R(\theta))$ has full row rank for all θ.

Proof. See Appendix G. □

Note that in the first two cases, the order of the model n is fixed based on the knowledge of the lags. In the first case, the order is limited to $n = \ell p$, while in the second case the order is the sum of the lags.

Remark 1 (Unknown lags)**.** *As can be seen from the handled cases, it is always required to specify the lag of the outputs. When the lags are not precisely known, it is possible to use an upper bound in exchange for additional model parameters and an increase in the variance of the estimated model parameters. The obtained estimate can be further refined if needed by running the method for multiple choices of lags, each time starting based on our previous estimate, until a suitable model is found. Choosing between the different models can be done by applying model selection criteria such as the Akaike Information Criterion (AIC) [28]. Since usually the goal is to obtain a simple model of sufficient quality, it is recommended to start with all lags equal and small, i.e., the first bullet point case, unless prior knowledge suggests otherwise. Increase the lag for each subsequent run until the model is acceptable according to the chosen criterion. If needed, the model obtained this way can be refined by further varying the relative lags of the outputs.*

4.3 Structural restriction, fixed and missing data

There are a few generalizations that concern the way the proposed method deals with elements of R and w, which are collected in this section due to their similarities. The first part will discuss allowing further restrictions on the structure of the R matrix to incorporate prior knowledge. The second part will talk about missing elements in the measured trajectory w. Finally, the third part will consider the situation where we want to keep certain elements of w unchanged during the estimation procedure.

4.3.1 Structural restriction

We can consider the situation where we have some prior knowledge regarding the model structure, i.e., the output at time t does not depend on the input at time t. This type of prior knowledge can be encoded as fixed elements of the R matrix. Since the proposed method already requires the ability to fix elements of R, this generalization is actually free.

4.3.2 Missing data

In real-world applications, it happens that some elements of the measured trajectory are missing, i.e., there are some times t for which not all elements of $w(t)$ are available. We refer to these elements as *missing elements*. It is not possible to exactly handle missing elements with the proposed method. However, it is possible to get close to it by making use of of a weighted cost function as described above. By setting the missing elements to zero and assigning these elements a small weight relative to the other elements (e.g., 10^{-5}) it is possible to treat these elements as "free" while only slightly biasing the cost function.

4.3.3 Fixed data

There are situations where some elements of w are known exactly. Two cases are of special interest, namely the case when all inputs are noise-free (the Output Error noise setting), and the case when the initial state (here the first ℓ samples) is known exactly. We refer to these elements as *fixed elements*, and to the others as *free elements*. Without loss of generality we are able to define a permutation matrix Π that splits the fixed and the free elements in w such that

$$\operatorname{col}(w) := \Pi \begin{bmatrix} \operatorname{col}(w_{\text{fixed}}) \\ \operatorname{col}(w_x) \end{bmatrix} = \begin{bmatrix} \Pi_{\text{fixed}} & \Pi_x \end{bmatrix} \begin{bmatrix} \operatorname{col}(w_{\text{fixed}}) \\ \operatorname{col}(w_x) \end{bmatrix}, \tag{29}$$

and similar for $\hat{w}$. For the fixed elements we have $\hat{w}_{\text{fixed}} := w_{\text{fixed}}$ by definition. Using the permutation matrix Π the cost function becomes

$$\begin{aligned} \|\operatorname{col}(\hat{w} - w)\|_2^2 &= \left\| \Pi \begin{bmatrix} \operatorname{col}(w_{\text{fixed}} - w_{\text{fixed}}) \\ \operatorname{col}(\hat{w}_x - w_x) \end{bmatrix} \right\|_2^2 \\ &= \|\operatorname{col}(\hat{w}_x - w_x)\|_2^2 . \end{aligned} \tag{30}$$

We can use the Δw substitution as used in Theorem 1 again, but only considering the free elements:

$$\operatorname{col}(\hat{w}_x) := \operatorname{col}(w_x) - \Delta w_x. \tag{31}$$

The cost function becomes $\|\Delta w_x\|_2^2$. Similarly, using the Δw_x substitution, we find that

$$\operatorname{col}(\hat{w}) = \Pi \begin{bmatrix} \operatorname{col}(w_{\text{fixed}}) \\ \operatorname{col}(w_x) - \Delta w_x \end{bmatrix}.$$

Based on the original derivation of $G\big(R(\theta)\big)$ in Theorem 1, plugging this into the system of equations

$$G\big(R(\theta)\big) \operatorname{col}(\hat{w}) = 0$$

results in

$$\begin{aligned} & G\big(R(\theta)\big)(\Pi_{\text{fixed}} \operatorname{col}(w_{\text{fixed}}) + \Pi_x \operatorname{col}(w_x) - \Pi_x \Delta w_x) = 0 \\ \Rightarrow\ & G\big(R(\theta)\big)\Pi_x \Delta w_x = s(\theta). \end{aligned}$$

The matrix $G_x\big(R(\theta)\big) = G\big(R(\theta)\big)\Pi_x$ is exactly $G\big(R(\theta)\big)$, but only contains the columns corresponding to free indices in the vector $\operatorname{col}(\hat{w})$. This new matrix still has full row rank for the given parameterization of R, assuming some limitations on which elements are fixed.

Assumption 1 (Fixed elements)**.** *Only at most the first ℓ output samples, and no other output samples, are fixed.*

We give this specific assumption to describe the most limited case where we can guarantee that the proposed method still works. Note that there are almost no situations where we would want to consider outputs not belonging to the initial ℓ samples to be fixed, so that this assumption is not limiting in practice. With this additional assumption, we prove that the matrix $G_x\big(R(\theta)\big)$ still has full row rank:

Lemma 2 (Rank of $G_x\big(R(\theta)\big)$)**.** *If Assumption 1 holds, then the matrix $G_x\big(R(\theta)\big)$ has full row rank when $R(\theta)$ is parameterized as described in Section 3.1.2.*

Proof. The proof is given in Appendix H □

Another way to look at removing columns of $G\big(R(\theta)\big)$ is to consider it as removing columns of P_t corresponding to the fixed elements in $w(t)$. Using the truncated versions of each P_i allows us to keep all other notation the same.

4.4 Multiple datasets

With a few small tweaks it is easy to allow for multiple independent datasets of varying length to be considered by the algorithm simultaneously. Consider that we have k datasets with corresponding trajectories $w^{(k)}$ and $\rho^{(k)}$ of length $T^{(k)}$. All that is required to define $G^{(k)}\big(R(\theta)\big)$ is to use $P_i^{(k)}$, which makes use of $\rho^{(k)}$, to define Δw as the concatenation of the different $\Delta w^{(i)}$'s,

and to define $s^{(k)}(\theta)$ as $\operatorname{col}\big(R(\theta)H_{\ell+1}(w^{(k)} \otimes \rho'^{(k)})\big)$. With these definitions, it suffices to take $G\big(R(\theta)\big) = \operatorname{blkdiag}\big(G^{(1)}\big(R(\theta)\big), \ldots, G^{(k)}\big(R(\theta)\big)\big)$ and

$$s(\theta) = \begin{bmatrix} s^{(1)}(\theta) \\ \vdots \\ s^{(k)}(\theta) \end{bmatrix}.$$

Remark that these definitions result in $\Gamma(R) := G\big(R(\theta)\big)G\big(R(\theta)\big)^\top = \operatorname{blkdiag}\big(\Gamma^{(1)}(R), \ldots, \Gamma^{(k)}(R)\big)$. Solving a system of equations with this matrix is equivalent to concatenating the solutions of the different sub-problems, i.e.,

$$z(\theta) := \Gamma^{-1}s(\theta) = \begin{bmatrix} (\Gamma^{(1)})^{-1}s^{(1)}(\theta) \\ \vdots \\ (\Gamma^{(k)})^{-1}s^{(k)}(\theta) \end{bmatrix}.$$

Further remark that the matrix $G\big(R(\theta)\big)$ is a block diagonal matrix with each block full row rank, which gives that it is full row rank.

5 Simulation Examples

All the following numerical examples will consider variations of mass-spring-damper (MSD) models with scheduling-varying spring constants. The reason for this is twofold. Firstly, MSD models are easy to generalize to an arbitrary number of inputs and outputs. Secondly, the discretized dynamics of MSD models exhibit significant sparsity in the model parameters, which is a commonality with many physical models, making it a representative example to showcase the results. As inputs we consider the forces applied to the different masses, and as outputs the displacement of the masses with respect to the left wall.

In the single-input single-output (SISO) case, the continuous dynamics are given by the differential equation

$$m\ddot{y}(\tau) + d\dot{y}(\tau) + k(p(\tau))y(\tau) = u(\tau), \quad \forall \tau \in \mathbb{R}$$

with m the mass, d the damping, $p(\tau) \in [-1, 1]$ the value of the scheduling variable at time τ, and $k : [-1, 1] \to \mathbb{R}$ the function describing the spring constant variation. We consider this function to be affine, i.e.,

$$k(p(\tau)) := k_a + k_b p(\tau), \quad p(\tau) \in [-1, 1], \tau \in \mathbb{R}.$$

Discretizing these dynamics using a forward Euler scheme with sample time T_s, introducing the discrete time t such that $t \cdot T_s = \tau$, results in the input-output representation

$$\begin{aligned} y(t) + \left(\frac{dT_s}{m} - 2\right) y(t-1) + \\ \left(1 + \frac{k_a T_s^2 - dT_s}{m} + \frac{k_b T_s^2}{m} p(t-2)\right) y(t-2) = \frac{T_s^2}{m} u(t-2). \end{aligned}$$

For the remainder of this section, we will use the parameter values $\begin{bmatrix} m & d & k_a & k_b & T_s \end{bmatrix} = \begin{bmatrix} 25 & 1 & 1.25 & 0.75 & 0.2 \end{bmatrix}$. As MIMO case, we consider an MSD model with 2 inputs, 2 outputs and 2 scheduling variables formed by chaining a second mass to the first mass with a spring and damper. A schematic representation can be seen in Figure 1a.

The continuous dynamics in this case are given by the differential equation

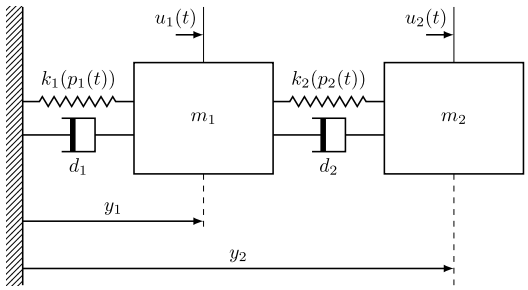
$$\begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix} \ddot{y}(\tau) + \begin{bmatrix} d_1 + d_2 & -d_2 \\ -d_2 & d_2 \end{bmatrix} \dot{y}(\tau) + \begin{bmatrix} k_1 + k_2 & -k_2 \\ -k_2 & k_2 \end{bmatrix} y = u(\tau), \quad \forall \tau \in \mathbb{R}$$

where the scheduling dependence of k_1 and k_2 are not explicitly stated to improve readability and with

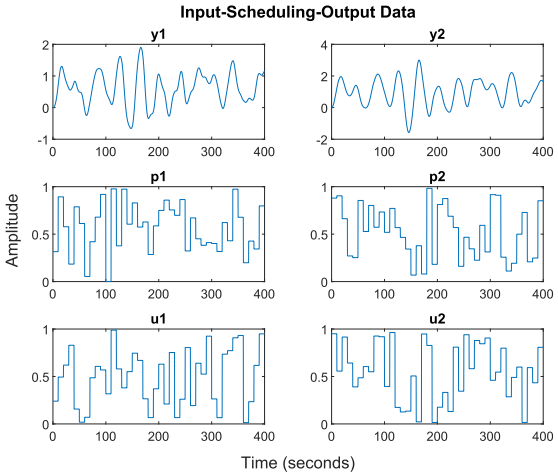
$$k_i(p_i(\tau)) := k_{i,a} + k_{i,b} p_i(\tau), \quad i = 1, 2.$$

For convenience, we will use the same parameters for both masses, i.e., $d_1 = d_2 = d$ and similar for the other parameters. For this choice, the input-output representation following the discretization of the continuous MIMO dynamics is given by

$$my(t) + \begin{bmatrix} 2dT_s - 2m & -dT_s \\ -dT_s & dT_s \end{bmatrix} y(t-1) + My(t-2) = T_s^2 u(t-2),$$



(a) The mass-spring-damper model used in Example 1 with parameter dependent spring constants.



(b) The data-generating input and scheduling trajectory with the corresponding output trajectory used in Example 1. This trajectory is sampled with noise on the inputs and outputs to obtain the estimation data for the methods.

Figure 1: The mass-spring-damper model used throughout the examples and an example data trajectory.

with

$$M := \begin{bmatrix} m - 2T_s d - 2T_s^2 k_a & T_s d - T_s^2 k_a \\ T_s d - T_s^2 k_a & m - T_s d - T_s^2 k_a \end{bmatrix} + T_s^2 k_b \begin{bmatrix} p_1(t-2) + p_2(t-2) & -p_2(t-2) \\ -p_2(t-2) & p_2(t-2) \end{bmatrix}.$$

For each example we consider K independent noise realizations v consisting of T i.i.d. samples, each following the EIV noise setting as described in Section 2.4. As a reminder, this means that each measured sample $w(t)$ is the sum of a true value $\bar{w}(t)$ and a noise value $v(t)$, where $\bar{w}$ is a trajectory of the true model and each $v(t) \sim \mathcal{N}(0, \eta^2 I_q)$ unless otherwise specified. In all examples we assume that there is no correlation between noise samples at different times, i.e., $v(t_1)$ is independent from $v(t_2)$ for $t_1 \neq t_2$. Depending on the example, the values of T or η are varied.

The Levenberg-Marquardt parameters are kept constant over the examples as $\alpha = 3$, $\mu_0 = 1$ and $\kappa_{\min} = 0.95$. The proposed method is allowed to keep running until the improvement of the cost function is less than 10^{-8}, the step has a norm less than 10^{-6}, the gradient has a norm less that 10^{-6}, or the maximum number of iterations is reached.

The following examples aim to showcase the proposed methods and generalizations. The first example will compare with state-of-the-art identification methods for SALPV models. The second example will illustrate the consistency and show that the runtime per iteration of the proposed method indeed scales linearly with the number of samples. The following examples each showcase how some specific generalizations can be used and how they impact the identification performance.

5.1 Example 1: MIMO Comparison

As a first example we will compare the performance of the proposed method with state-of-the-art identification methods for SALPV models. We will consider three methods, all iterative:

1. SLRA-Toolbox: SLRA toolbox from [23] applied to the Kronecker-Hankel matrix, only preserving the Hankel structure.
2. LPV-SLRA: The method proposed in this paper.
3. LPVcore-OE: The LPVcore toolbox (see [29]) estimating an input-output representation of the correct order assuming Output Error (OE) noise using a prediction error method.

Although the SLRA-Toolbox does not identify SALPV models, it is included here to test the hypothesis that identification methods for linear models will not work well, even though the lifted model is linear. The LPVcore toolbox has no method implemented that can work with EIV noise directly, unless a complicated Box-Jenkins parameterization is used. One of the goals of this example is to show the difference this noise assumption makes. We simulate the discretized model (using the LPVcore toolbox) using a 2000 sample long input and scheduling signal with each element pulled

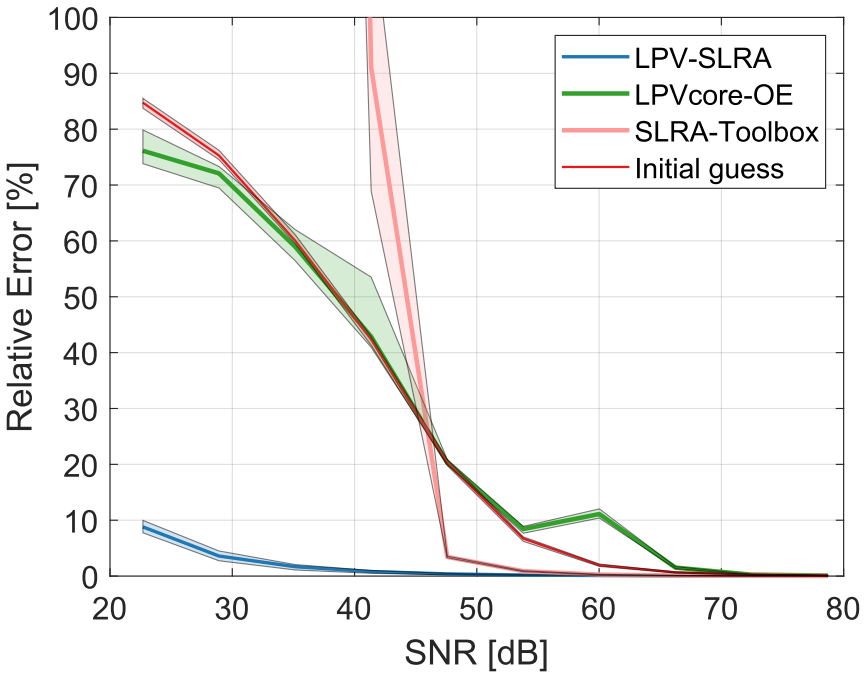


Figure 2: Comparison of the relative errors for different noise levels. 'LPV-SLRA' (—) consistently performs better than the alternatives 'LPVcore-OE' (—), 'SLRA-Toolbox' (—) and 'Initial guess' (—), only degrading in model quality for high noise levels. The other methods exhibit significant degradation in model quality for low noise levels.

uniformly from the interval $]0, 1[$, but kept fixed for 50 samples at a time. See Figure 1b for an example trajectory.

As a figure of merit, we use the relative parameter error, which is given by the ratio of Frobenius norms

$$\mathrm{RE}(\theta) := \frac{\left\|R(\theta) - \bar{R}\right\|_F}{\left\|\bar{R}\right\|_F},$$

with $\bar{R}$ the true R matrix. We compare the resulting R matrices instead of the parameter vectors θ themselves to account for solutions that do not respect the considered structure, such as the SLRA-Toolbox. The methods will be initialized using the ARX estimate, obtained by applying the `lpvarx` function of the LPVcore toolbox. This function computes a biased pre-estimate of the model parameters by solving a least-squares problem. Out of runtime considerations, we cap the maximum number of iterations for each iterative method to stay under 10 minutes of runtime for a single identification, or 1000 iterations at most. This results in a maximum of 1000 iterations for LPV-SLRA and only 200 iterations for LPV-OE. Note that both of these methods use relatively slow native Matlab code, compared to the SLRA-Toolbox, which uses compiled C++ code. The SLRA-Toolbox is also limited to 1000 iterations. We repeat the identification for 10 different noise levels, varying linearly from 78.7dB to 22.7dB, and 10 independent noise realizations. In total this means that each identification method is called 100 times. Statistics such as the median and quantiles are computed over the different noise realizations for each noise level. The results can be seen in Figure 2. Compared to the other methods, the proposed method significantly outperforms the alternatives, only degrading in model quality for high noise levels. The alternatives already degrade in model quality for low noise levels. After analysis, it seems that the MIMO identification problem is sensitive to noise. In our setting, the approximate left null space of the Hankel matrix $H_L(w_{\mathrm{ext}})$, which is the space spanned by the singular vectors corresponding to the two smallest singular values, changes significantly at noise levels as low as 60dB SNR. This means that initial guesses generated by more naive methods are more likely to be of lower quality. These changes increase the chance of encountering local minima. Furthermore, the MIMO identification problem seems to have many flat areas in the optimization landscape, which are slow to pass through.

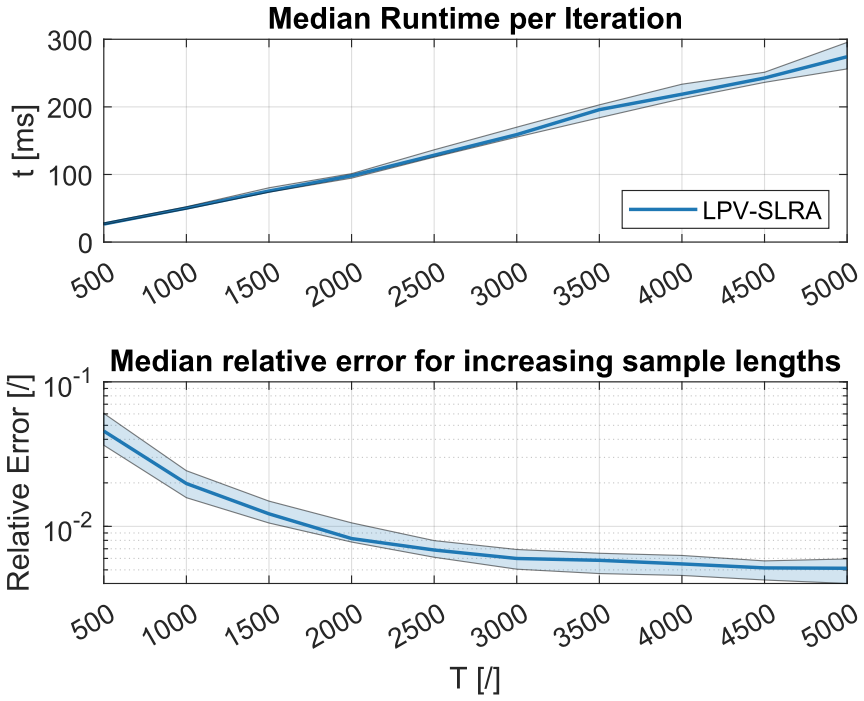


Figure 3: Distribution of the runtime per iteration (top) and the relative parameter error (bottom) over 50 independent noise realizations for the 'LPV-SLRA' method. The thick lines (—) represent the median over the noise realizations, while the shaded area represents the area between the first and third quartiles. The top figure shows a clear linear relationship. The bottom figure shows a clear decrease in relative error for larger sample sizes.

5.2 Example 2: Consistency and Linear Runtime Scaling

For this example we fix the noise variance as $\eta^2 = 4 \cdot 10^{-4}$, which corresponds to a SNR of around 38.3dB. The goal of this example is twofold: show that for increasing sample lengths T the runtime per iteration of the proposed method increases linearly, and show that for a good initial guess the relative parameter error after convergence decreases for larger T. The relative error we are interested in is the same as used in Example 1. The initial guess will be computed using the ARX estimate based on the longest tested trajectory for each noise realization.

We generated 20 independent Gaussian noise realizations $v(t)$ with variance η^2 of length 5000. For each realization $w(t) = \bar{w}(t) + v(t)$, the proposed method is run until termination making use of only the first $T = 500, 1000, \ldots, 5000$ samples. The runtime per iteration and the relative error are shown in Figure 3. For each plot, the blue line represents the median value over the 20 noise realizations, while the shaded area represents the area between the first and third quartiles. It is clear from the figure that the runtime per iteration indeed scales linearly with the number of samples. There is also a noticeable decrease in the relative error for larger sample sizes.

5.3 Example 3: Exact Data and Multiple Datasets

The goal of this example is to take a look at the generalizations allowing for fixing elements of the trajectory w and allowing for optimizing over multiple independent datasets simultaneously.

The setup is mostly the same as in Example 1, except for a few key points:

- The single 2000 sample long trajectories are split into 4 input and scheduling signals of 500 samples, for which the corresponding output is simulated by the true system from the zero state, resulting in 4 separate trajectories with known initial state.
- We only compare the proposed method (LPV-SLRA) using the long trajectory and the 4 shorter trajectories, not including the other methods.
- We assume that the initial state is known exactly and is zero for the entire past of the system.

Fixed initial states can be accounted for by adding ℓ zero samples with an infinite weight, and corresponds to the 'fixed data' extension. We compare the models obtained by using the four independent trajectories with the model obtained by using a single long trajectory. This long trajectory

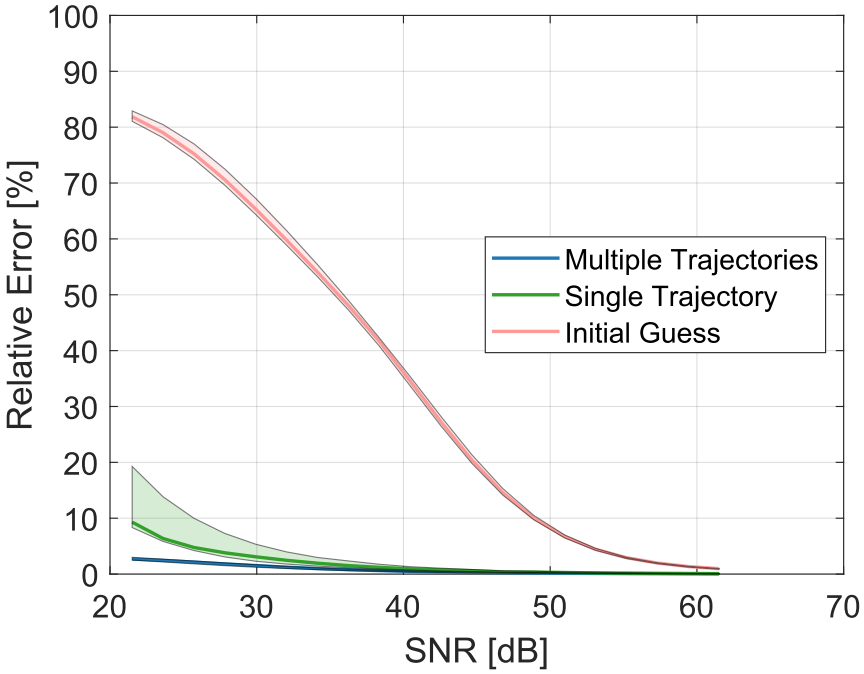


Figure 4: Distribution of the relative parameter error over increasing noise levels, with η the standard deviation of the noise signal. The estimation using 4 separate trajectories starting from the known zero state (—) has a very low spread around the median value, and is visibly better than the single trajectory with the same total number of samples (—) for higher noise levels. Both estimates significantly improve upon the initial guess (—).

is generated by simulating the concatenation of the small input and scheduling trajectories. The results are given in Figure 4. We can see that both the single- and multi trajectory cases significantly improve upon the initial guess. Using multiple trajectories with known initial states offers visual improvement in both the median and the spread of the relative error compared to the single trajectory case, especially for higher noise levels. This is to be expected, since fixing initial states restricts the possible estimated trajectories to those reachable from the given initial state, and more information results in more restrictions.

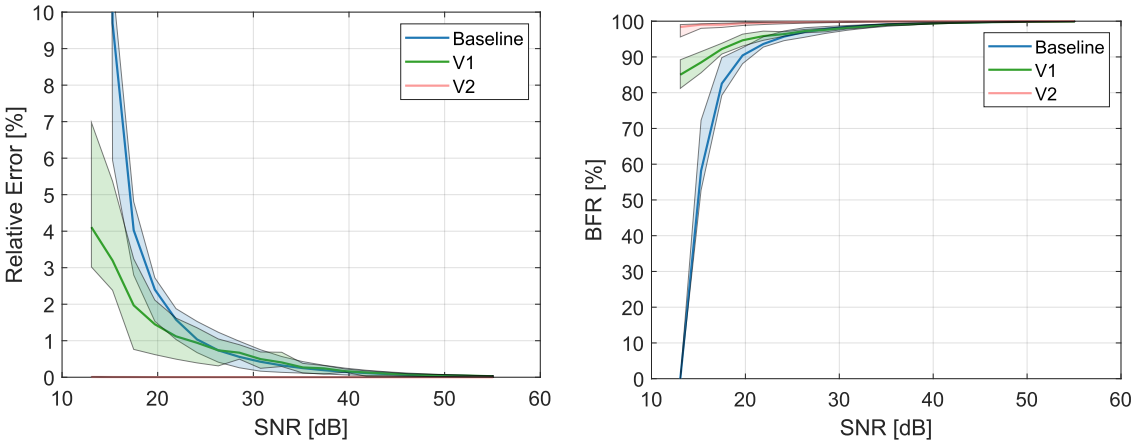
5.4 Example 4: Other Generalizations

For this example we make use of the SISO mass-spring-damper model. Compared to Example 1, we make a one additional change: The noise covariance changes from $\eta^2 I$ to $\eta^2\Sigma = \eta^2 \begin{bmatrix} 0.1 & -0.15 \\ -0.15 & 2 \end{bmatrix}$, which results in correlation between the input and output noise. The noise samples $v(t)$ are still assumed to be i.i.d. with $v(t) \sim \mathcal{N}(0, \eta^2\Sigma)$. A few variations of the proposed method will be compared:

- Baseline: Does not use any of the generalizations.
- V1: Uses a known zero initial state and knowledge about the correct noise covariance matrix.
- V2: Same as V1, but also uses prior knowledge about which elements of the R matrix are zero, which corresponds to the 'structural restriction' extension.

Contrary to previous examples, not all setups will use the same initial guess. The V2 setup will be initialized with the R matrix obtained from the V1 setup after correctly setting the fixed elements, while the others still use the ARX estimate, which uses the same information as the Baseline setup. The reasoning for this is that prematurely fixing elements of the R matrix significantly slows down the convergence speed by eliminating many paths to the optimal solution. Fixing known sparsity in the model structure is advantageous when refining an obtained model that is already in the neighborhood of the optimal solution.

To compensate for the better initial guess, the V2 setup is only allowed to run for at most 200 iterations. Baseline and V1 are each allowed 500 iterations. Note that for higher noise levels, the solution given by the V1 setup usually violates the assumed sparsity structure significantly, such



(a) The median relative error (thick line) of the estimated R matrix. The colored areas indicate the range between the first and third quartiles. V2 is barely visible at the bottom, peaking at around 0.01 percent relative error for the worst tested noise realization.

(b) The median Best Fit Rate (BFR) (thick line) of the estimated R matrix. The colored areas indicate the range between the first and third quartiles. V2 (orange) significantly outperforms the other setups. V2 (green) remains high even for high noise levels.

Figure 5: Figures-of-merit for the estimated model R. Setups 'Baseline' (—) and 'V1' (—) perform similarly until higher noise levels, after which 'Baseline' performs significantly worse than 'V1'. Setup 'V2' (—) consistently outperforms the other setups.

that the initial R matrix given to V2 (with the sparsity structure enforced by directly setting these elements) is farther away from the optimal solution than before, albeit still closer than the original initial guess. This means that we cannot reduce the number of iterations for the V2 setup too much. We repeat the identification for 10 independent noise realizations. For each noise realization, we compute our figures of merit for 20 SNR values linearly spaced between 55dB and 13dB. Other than the relative error we will also consider the Best Fit Rate (BFR) for the simulated response as a figure of merit. The BFR is a measure of how well the model is able to reproduce a validation trajectory, and is given by the following formula:

$$\mathrm{BFR}(\hat{y}) = \max\left\{1 - \frac{\sum_{t=1}^{T} \|y_{\mathrm{val}}(t) - \hat{y}(t)\|}{\sum_{t=1}^{T} \|y_{\mathrm{val}}(t) - \bar{y}\|}, 0\right\},$$

with $\bar{y}$ the sample mean of y_{val}, taken on a noise-free validation data set of the same size as the estimation data set, and $\hat{y}$ the simulated output of the model-under-test given the validation input and scheduling data. Note that both figures of merit apply to the obtained model R, and are not directly related to the approximation of our noisy data during the estimation procedure. The results are shown in Figure 5. We can see that both figures-of-merit tell the same story. Setups Baseline and V1 have similar performance until about 25dB SNR. This is to be expected, since higher noise levels enlarge the impact of a mismatch between expected and actual noise covariance. V2 always performs exceptionally well. It should be noted that V1 always stopped due to a lack of possible improvement, and not due to reaching the maximum number of iterations. This implies that just the additional number of iterations done by V2 on the end point of V1 is not enough to explain this improvement. We can conclude that, in the case where prior knowledge about the sparse structure of R is available, using this knowledge to refine a good estimate is worthwhile to do.

6 Conclusion and future work

This paper introduces a novel data-driven identification method for SALPV models in the EIV noise setting based on a LPV kernel representation. Considering the given extensions, the proposed method has many features compared to state-of-the-art identification methods. These features include handling the EIV noise setting directly, handling fixed and missing data, simplicity in estimating MIMO models, assigning weights to the data samples, and the ability to handle multiple independent data sets simultaneously. Furthermore, the method is implemented in a publicly available Matlab package, and has a theoretical runtime per iteration that scales linearly with the number of samples. The proposed method and its extensions are tested on a simulated MIMO mass-spring-damper (MSD) model. We observe better performance in the EIV noise setting compared to

the tested state-of-the-art LPV identification methods designed for the OE noise setting, with the performance gap increasing for higher noise levels. Additional examples show how the extensions to the proposed method can offer significant improvements over the basic implementation.

Future work needed to improve the proposed method is finding a way to handle noise on the scheduling variables while still keeping the global convergence guarantees. Additionally, although the proposed method can deal with more noise settings than just EIV, it is not yet able to optimally handle colored noise akin to the Box-Jenkins model.

Acknowledgments

This work is supported by: the Research Foundation Flanders (FWO) research projects G081222N, G033822N, G0A0920N and the Research Council KU Leuven C1 project with ID C14/24/103. IM is funded by the Catalan Institution for Research and Advanced Studies (ICREA).

References

[1] D. Rodriguez-Guevara, A. Favela-Contreras, F. Beltran-Carbajal, D. Sotelo, and C. Sotelo, "Active Suspension Control Using an MPC-LQR-LPV Controller with Attraction Sets and Quadratic Stability Conditions," *Mathematics*, vol. 9, p. 2533, Jan. 2021.

[2] F. Blanchini, D. Casagrande, S. Miani, and U. Viaro, "Robust linear parameter-varying control of induction motors," *International Journal of Robust and Nonlinear Control*, vol. 25, no. 12, pp. 1783–1800, 2015.

[3] T. Bloemers, T. Oomen, and R. Tóth, "Frequency Response Data-Based LPV Controller Synthesis Applied to a Control Moment Gyroscope," *IEEE Transactions on Control Systems Technology*, vol. 30, pp. 2734–2742, Nov. 2022.

[4] G. Zong, D. Yang, J. Lam, and X. Song, "Fault-Tolerant Control of Switched LPV Systems: A Bumpless Transfer Approach," *IEEE/ASME Transactions on Mechatronics*, vol. 27, pp. 1436–1446, June 2022.

[5] P. Schuchert and A. Karimi, "Frequency domain LPV controller synthesis for a positioning system with uncertain scheduling parameters," *IFAC-PapersOnLine*, vol. 56, pp. 1363–1369, Jan. 2023.

[6] R. de Rozario and T. Oomen, "Frequency Response Function Identification of LPV Systems: a Global Approach with Application to Mechanical Systems," *IFAC-PapersOnLine*, vol. 51, pp. 108–113, Jan. 2018.

[7] T. Söderström, "Errors-in-variables methods in system identification," *Automatica*, vol. 43, pp. 939–958, June 2007.

[8] R. Tóth and C. Verhoek, "Modeling and control of lpv systems," in *Encyclopedia of Systems and Control Engineering (First Edition)* (Z. Ding, ed.), vol. Volume 1, pp. 405–418, Oxford: Elsevier, first edition ed., 2026.

[9] X. Jin, B. Huang, and D. S. Shook, "Multiple model LPV approach to nonlinear process identification with EM algorithm," *Journal of Process Control*, vol. 21, pp. 182–193, Jan. 2011.

[10] Y. Zhao, B. Huang, H. Su, and J. Chu, "Prediction error method for identification of LPV models," *Journal of Process Control*, vol. 22, pp. 180–193, Jan. 2012.

[11] W. J. Rugh and J. S. Shamma, "Research on gain scheduling," *Automatica*, vol. 36, pp. 1401–1425, Oct. 2000.

[12] J.-W. van Wingerden and M. Verhaegen, "Subspace identification of Bilinear and LPV systems for open- and closed-loop data," *Automatica*, vol. 45, pp. 372–381, Feb. 2009.

[13] P. B. Cox, R. Tóth, and M. Petreczky, "Towards efficient maximum likelihood estimation of LPV-SS models," *Automatica*, vol. 97, pp. 392–403, Nov. 2018.

[14] R. Tóth, P. S. C. Heuberger, and P. M. J. Van den Hof, "Prediction-Error Identification of LPV Systems: Present and Beyond," in *Control of Linear Parameter Varying Systems with Applications* (J. Mohammadpour and C. W. Scherer, eds.), pp. 27–58, Boston, MA: Springer US, 2012.

[15] M. Butcher, A. Karimi, and R. Longchamp, "On the Consistency of Certain Identification Methods for Linear Parameter Varying Systems," *IFAC Proceedings Volumes*, vol. 41, pp. 4018–4023, Jan. 2008.

[16] V. Laurain, M. Gilson, R. Tóth, and H. Garnier, "Refined instrumental variable methods for identification of LPV Box–Jenkins models," *Automatica*, vol. 46, pp. 959–967, June 2010.

[17] M. Schoukens and R. Tóth, "Frequency Response Functions of Linear Parameter-Varying Systems," *IFAC-PapersOnLine*, vol. 52, pp. 32–37, Jan. 2019.

[18] D. Piga, P. Cox, R. Tóth, and V. Laurain, "LPV system identification under noise corrupted scheduling and output signal observations," *Automatica*, vol. 53, pp. 329–338, Mar. 2015.

[19] X. Liu, G. Han, and X. Yang, "Robust Global Identification of LPV Errors-in-Variables Systems With Incomplete Observations," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, pp. 3799–3807, June 2022.

[20] J. C. Willems, "Paradigms and puzzles in the theory of dynamical systems," *IEEE Transactions on Automatic Control*, vol. 36, no. 3, pp. 259–294, 1991.

[21] C. Verhoek, I. Markovsky, S. Haesaert, and R. Tóth, "A behavioral approach for LPV data-driven representations," *IEEE Transactions on Automatic Control*, pp. 1–14, 2025.

[22] I. Markovsky, C. Verhoek, and R. Toth, "The most powerful unfalsified linear parameter-varying model," *Automatica*, 2026.

[23] K. Usevich and I. Markovsky, "Variable projection for affinely structured low-rank approximation in weighted 2-norms," *Journal of Computational and Applied Mathematics*, vol. 272, pp. 430–448, Dec. 2014.

[24] T. Söderström, *Errors-in-Variables Methods in System Identification*. Springer, 2018.

[25] J. C. Willems and J. W. Polderman, *Introduction to Mathematical Systems Theory: A Behavioral Approach*. Springer Science & Business Media, 1997.

[26] I. Markovsky and F. Dörfler, "Identifiability in the Behavioral Setting," *IEEE Transactions on Automatic Control*, vol. 68, pp. 1667–1677, Mar. 2023.

[27] E. H. Bergou, Y. Diouane, and V. Kungurtsev, "Convergence and Complexity Analysis of a Levenberg–Marquardt Algorithm for Inverse Problems," *Journal of Optimization Theory and Applications*, vol. 185, pp. 927–944, June 2020.

[28] P. Stoica and Y. Selen, "Model-order selection: a review of information criterion rules," *IEEE Signal Processing Magazine*, vol. 21, pp. 36–47, July 2004.

[29] P. den Boef, P. B. Cox, and R. Tóth, "LPVcore: MATLAB Toolbox for LPV Modelling, Identification and Control," *IFAC-PapersOnLine*, vol. 54, pp. 385–390, Jan. 2021.

[30] G. H. Golub and V. Pereyra, "The Differentiation of Pseudo-Inverses and Nonlinear Least Squares Problems Whose Variables Separate," *SIAM Journal on Numerical Analysis*, vol. 10, pp. 413–432, Apr. 1973.

[31] G. H. Golub and C. F. V. Loan, *Matrix Computations*. JHU Press, Feb. 2013.

A Transforming SALPV IO Representation to Kernel Representation

It is possible to rewrite the IO representation (1) as a kernel representation of a lifted LTI model, i.e., an LTI model with additional inputs. In this case, the additional inputs are $\rho(t) \otimes u(t)$ and $\rho(t) \otimes y(t)$, with ρ the scheduling variable. The remainder of this appendix will work out this reformulation in detail. Before beginning, recall the following notation:

$$\begin{aligned} \rho'(t) &:= \begin{bmatrix} 1 \\ \rho(t) \end{bmatrix}, \\ w_{\text{ext}}(t) &:= \rho'(t) \otimes w(t). \end{aligned}$$

Introducing this notation greatly simplifies the notation for the coming reformulation. Starting from the original IO representation, we have

$$\begin{aligned} & \sum_{i=0}^{\ell} \sum_{k=0}^{n_\rho} a_{i,k} \rho_k'(t-i) y(t-i) = -\sum_{j=0}^{\ell} \sum_{k=0}^{n_\rho} b_{j,k} \rho_k'(t-j) u(t-j) \\ \Rightarrow & \sum_{i=1}^{\ell} a_i (\rho'(t-i) \otimes y(t-i)) = -\sum_{j=0}^{\ell} b_j (\rho'(t-j) \otimes u(t-j)) \\ \Rightarrow & \sum_{i=0}^{\ell} \begin{bmatrix} a_i & b_i \end{bmatrix} \begin{bmatrix} \rho'(t-i) \otimes y(t-i) \\ \rho'(t-i) \otimes u(t-i) \end{bmatrix} = 0 \\ \Rightarrow & \sum_{i=0}^{\ell} R_{\ell-i} (\rho'(t-i) \otimes w(t-i)) = 0, \end{aligned}$$

with

$$\begin{aligned} a_i &:= \begin{bmatrix} a_{i,0} & a_{i,1} & \cdots & a_{i,n_\rho} \end{bmatrix}, \\ b_i &:= \begin{bmatrix} b_{i,0} & b_{i,1} & \cdots & b_{i,n_\rho} \end{bmatrix}, \\ w(t) &:= \begin{bmatrix} u(t) \\ y(t) \end{bmatrix}. \end{aligned}$$

With the notation defined above, after explicitly writing out the sum and shifting the entire equation in time we obtain

$$R_0 w_{\text{ext}}(t) + \cdots + R_\ell w_{\text{ext}}(t+\ell) = 0.$$

B Proof of Lemma 1

Define the linear mapping $F_{L,T} : \mathbb{R}^{qT} \to \mathbb{R}^{qL(T-L+1)}$, such that

$$F_{L,T} \operatorname{col}(w) = \operatorname{col}(H_L(w)),$$

with $\operatorname{col}(w) \in \mathbb{R}^{qT}$. This mapping maps the vectorized signal to the vectorized block Hankel matrix of this signal. A vector version of the forward shift operator $\sigma(\cdot)$ is the block shift matrix J_T such that for any signal $w \in (\mathbb{R}^q)^T$ we have

$$J_b \begin{bmatrix} w(1) \\ w(2) \\ \vdots \\ w(T-1) \\ w(T) \end{bmatrix} = \begin{bmatrix} w(2) \\ w(3) \\ \vdots \\ w(T) \\ 0_{q \times 1} \end{bmatrix}.$$

These two matrices will be used during the following proof.

This proof consists of three parts:

1. Show that the two systems of equations are equivalent.
2. Show that $G(R)$ has the required structure.

3. Show that $G\big(R(\theta)\big)$ has full row rank for the specified parameterization of R.

Note that the bulk of this lemma holds for a general R matrix with size $d \times q_{\text{ext}}(\ell+1)$. It is only the final claim, i.e., that the matrix $G\big(R(\theta)\big)$ has full row rank, that requires a parameterized R matrix of size $p \times q_{\text{ext}}(\ell+1)$. The first step of the proof is manipulating the left hand side to extract x from the Kronecker-Hankel matrix. Vectorizing both sides results in

$$\begin{aligned} & RH_{\ell+1}(\rho' \otimes x) = B \\ \Leftrightarrow\ & (I_{T-\ell} \otimes R)\,\mathrm{col}\,(H_{\ell+1}(\rho' \otimes x)) = \mathrm{col}\,(B)\,. \end{aligned}$$

Making use of the matrix $F_{\ell+1,T}$, as defined in the notation section and recalling that $q_{\text{ext}} := (n_\rho + 1)q$, we can write

$$\mathrm{col}\,(H_{\ell+1}(\rho' \otimes x)) = F_{\ell+1,T}\,\mathrm{col}\,(\rho' \otimes x)\,.$$

Recall that each sample of the trajectory $\rho' \otimes x$ is given by

$$\rho'(t) \otimes x(t) = P_t x(t) \quad \forall t \in (1, 2, \ldots, T),$$

with $P_t := \rho'(t) \otimes I_q$. Putting everything together gives

$$\begin{aligned} & RH_{\ell+1}(\rho' \otimes x) = B \\ \Leftrightarrow\ & (I_{T-\ell} \otimes R)F_{\ell+1,T}P\,\mathrm{col}\,(x) = \mathrm{col}\,(B)\,, \end{aligned}$$

with $P := \mathrm{blkdiag}\,(P_1, P_2, \ldots, P_T)$. Defining the matrix

$$G(R) := (I_{T-\ell} \otimes R)F_{\ell+1,T}P$$

proves the first part of the lemma.

The matrix $G(R)$ consists of three building blocks that are multiplied together. The matrix $F_{\ell+1,T}$ can be written in function of the block shift matrix $J_{q_{\text{ext}},T}$ as follows:

$$F_{\ell+1,T} = \begin{bmatrix} \begin{bmatrix} I_{q_{\text{ext}}(\ell+1)} & \bar{0} \end{bmatrix} (J_T)^0 \\ \begin{bmatrix} I_{q_{\text{ext}}(\ell+1)} & \bar{0} \end{bmatrix} (J_T)^1 \\ \vdots \\ \begin{bmatrix} I_{q_{\text{ext}}(\ell+1)} & \bar{0} \end{bmatrix} (J_T)^{T-\ell-1} \end{bmatrix} = \begin{bmatrix} F_1 \\ F_2 \\ \vdots \\ F_{T-\ell} \end{bmatrix},$$

with $\bar{0} = 0_{q_{\text{ext}}(\ell+1) \times q_{\text{ext}}(T-\ell-1)}$. This structure can be interpreted in the following way: each block row consists of a shifted identity matrix. The first identity matrix is unshifted, but by moving downwards a block row, the identity matrix is shifted q_{ext} elements to the right. Since the identity matrices have size $q_{\text{ext}}(\ell+1)$ we need $\ell+1$ shifts before the two identity matrices have no columns in common.

Left multiplying P by this matrix results in a staircase structure formed of block diagonal matrices of P_is. More specifically, multiplying the i-th block row F_i with P results in

$$F_i P = \begin{bmatrix} 0_{q_{\text{ext}}(\ell+1) \times q(i-1)} & \begin{bmatrix} P_i & & & \\ & P_{i+1} & & \\ & & \ddots & \\ & & & P_{i+\ell} \end{bmatrix} & 0_{q_{\text{ext}}(\ell+1) \times q(T-\ell-i)} \end{bmatrix},$$

with empty space corresponding to zero values, and by definition each P_i is of size $q_{\text{ext}} \times q$. We name this product K_i, which is more convenient to refer to, since this specific matrix will be used later on as well, resulting in

$$K_i := F_i P. \tag{32}$$

Each block row of the Kronecker product $(I_{T-\ell} \otimes R)$ is just R shifted an additional $q_{\text{ext}}(\ell+1)$ elements to the right. Given the fact that this is also both the width of R and the height of each F_i, we find that the i-th block row of $G(R)$ is equal to RF_iP, which has the following structure:

$$RF_iP = \begin{bmatrix} 0_{i-1} & \begin{bmatrix} R_0 P_i & R_1 P_{i+1} & \cdots & R_\ell P_{i+\ell} \end{bmatrix} & 0_{T-\ell-i} \end{bmatrix},$$

where the notation 0_i is introduced as a more compact way to write $0_{q_{\text{ext}}(\ell+1) \times qi}$. The matrix $G(R)$ is formed by vertically stacking these RF_iP block rows, which is exactly the matrix $\mathcal{M}_T(R, \rho)$ as defined in (12) with $p \times q$ blocks.

The final step is to show that $G\big(R(\theta)\big)$ has full row rank irrespective of the choice of $\theta \in \mathbb{R}^{n_\theta}$. For single-output models, the matrix $G\big(R(\theta)\big)$ has $T-\ell$ rows and qT columns. A necessary condition for $G\big(R(\theta)\big)$ to be full row rank is that it has at least as many columns as rows, i.e., $T-\ell \leq qT$, which is always the case since $q = m+p \geq 1$.

The matrix $G\big(R(\theta)\big)$, with parameterized R matrix, consists of blocks with size $1 \times q$, and each block is the result of a multiplication in the form $R_i P_j$. The R_i blocks are the result of partitioning the matrix $R(\theta)$ into q_{ext}-wide blocks, i.e.,

$$R(\theta) = \begin{bmatrix} R_0(\theta) & R_1(\theta) & \cdots & R_\ell(\theta) \end{bmatrix}.$$

When considering the structure of $G\big(R(\theta)\big)$, we see that each row consists of exactly $\ell+1$ non-zero blocks, with the last non-zero block of the form $R_\ell P_i$. The parameterization of R makes it such that $R_\ell(\theta)P_i w(i) = y(i) + R_u(\theta)\rho'(i)u(i)$, for all $i \in (1,2,\ldots,T)$ and for some matrix $R_u(\theta)$. This implies that there is a fixed one element in the row vector $R_\ell P_i$, which corresponds to the $y(t)$ coefficient. Consider a column-permuted version of $G\big(R(\theta)\big)$ that moves this one element to the end of the block for each block column. This is possible, since the matrix R_ℓ appears at most once in each block column. The resulting matrix has the same rank as $G\big(R(\theta)\big)$, and is in row echelon form. Since there are no zero rows and $G\big(R(\theta)\big)$ has at least as many columns as rows, the matrix $G\big(R(\theta)\big)$ has full row rank.

C Proof of Theorem 1

Assume that we are given a valid input-output partitioning, i.e., which signals in w can be treated as inputs and outputs, which is required to define the structured matrix $R(\theta)$. This is a reasonable assumption for most modeling applications. If no valid partitioning is given, the algorithm can be repeated for multiple possible partitionings until a suitable solution is found. The existence of at least one valid input-output partitioning is guaranteed under the noise model assumption of Section 2.4.

For finite-length trajectories, the kernel represented behavior in (7) can be written as

$$\begin{aligned} &\mathcal{B}_\rho|_T(\theta) = \\ &\left\{ w \in (\mathbb{R}^q)^T \;\middle|\; \sum_{i=0}^{\ell} R_i(\theta) w_{\text{ext}}(t+i) = 0 \quad \forall t \leq T-\ell \right\}, \end{aligned} \tag{33}$$

with $q := m+p$. The constraint can alternatively be written as the function of a Hankel matrix: $R(\theta)H_{\ell+1}(w_{\text{ext}}) = 0$. Making use of (33) and considering the set $\mathcal{P}_c^{q,n_\rho} := \left\{ \mathcal{B}(\theta) \subset (\mathbb{R}^q)^{\mathbb{N}} \;\middle|\; \theta \in \mathbb{R}^{n_\theta} \right\}$, Problem (14) is equivalently written as

$$\underset{\hat{w} \in (\mathbb{R}^q)^T, \theta \in \mathbb{R}^{n_\theta}}{\text{minimize}} \quad \|\hat{w} - w\|_2^2 \quad \text{s.t. } R(\theta)H_{\ell+1}(\rho' \otimes \hat{w}) = 0. \tag{34}$$

Consider the substitution $\Delta w = w - \hat{w}$. The resulting problem becomes

$$\begin{aligned} \underset{\Delta w \in (\mathbb{R}^q)^T, \theta \in \mathbb{R}^{n_\theta}}{\text{minimize}} \quad & \|\Delta w\|_2^2 \\ \text{s.t.} \quad & R(\theta)H_{\ell+1}(\rho' \otimes \Delta w) = R(\theta)H_{\ell+1}(\rho' \otimes w), \end{aligned} \tag{35}$$

and is equivalent to problem (34). For fixed values of $\theta \in \mathbb{R}^{n_\theta}$, this problem is a linear least-squares problem with linear equality constraints. This fact can be leveraged using the Variable Projection principle, first discussed in [30], which is commonly used to solve nonlinear least-squares problems of similar structure. Define the function

$$f(\theta) = \left\{ \begin{array}{ll} \min\limits_{\Delta w \in (\mathbb{R}^q)^T} & \|\Delta w\|_2^2 \\ \text{s.t. } & R(\theta)H_{\ell+1}(\rho' \otimes \Delta w) = R(\theta)H_{\ell+1}(\rho' \otimes w) \end{array} \right\}.$$

The problem

$$\underset{\theta \in \mathbb{R}^{n_\theta}}{\text{minimize}} \;\; f(\theta) \tag{36}$$

is equivalent to problem (35), as shown in [30]. While the original problem is a nonlinear least-squares problem due the bilinear equality constraint, the problem in $f(\theta)$ is now linear, and has a

closed-form solution. Invoking Lemma 1 on the equality constraints, i.e., taking x as Δw and B as $R(\theta)H_{\ell+1}(\rho' \otimes w)$, allows us to rewrite the constraints into

$$f(\theta) = \left\{ \min_{\Delta w \in (\mathbb{R}^q)^T} \ \|\Delta w\|_2^2 \quad \text{s.t. } G\big(R(\theta)\big)\,\mathrm{col}\,(\Delta w) = s(\theta) \right\},$$

with

$$s(\theta) := \mathrm{col}\,(R(\theta)H_{\ell+1}(\rho' \otimes w))\,.$$

Furthermore, Lemma 1 also shows that $G\big(R(\theta)\big)$ has full row rank for all θ, such that the problem in $f(\theta)$ has a unique global minimizer $\Delta w^*(\theta)$ given by (18a), resulting in

$$f(\theta) = \|\Delta w^*(\theta)\|_2^2\,.$$

Problem (36), which is equivalent to Problem (14) given the chain of equivalences in this proof, becomes

$$\underset{\theta \in \mathbb{R}^{n_\theta}}{\text{minimize}} \ \|\Delta w^*(\theta)\|_2^2\,,$$

concluding the proof.

D Derivation of the structure of the Gram matrix

in Section 3.3.2, the matrix of interest is

$$\begin{aligned}\Gamma\big(R(\theta)\big) &= G\big(R(\theta)\big)G\big(R(\theta)\big)^\top \\ &= (I_b \otimes R(\theta))F_{L,T}PP^\top(F_{L,T})^\top(I_b \otimes R(\theta)^\top) \\ &= (I_b \otimes R(\theta))V(I_b \otimes R(\theta)^\top),\end{aligned}$$

with $b := T - \ell$, and with $F_{L,T}$ defined as in Appendix B. Note that the matrix denoted by V only depends on the structure, and is independent of R. Based on the Kronecker products it makes sense to partition this matrix into square blocks of size $q_{\text{ext}}L$, such that

$$V := \begin{bmatrix} V_{1,1} & \cdots & V_{1,b} \\ \vdots & & \vdots \\ V_{b,1} & \cdots & V_{b,b} \end{bmatrix}, \qquad V_{i,j} \in \mathbb{R}^{q_{\text{ext}}L \times q_{\text{ext}}L}.$$

Based on this partitioning, and making use of the row partitioning of $F_{L,T}$ as defined in Appendix B, we find that

$$V_{i,j} := F_i PP^\top F_j^\top\,.$$

Based on the structure of P, we can see that $PP^\top$ is a block diagonal matrix with blocks of size $q_{\text{ext}} \times q_{\text{ext}}$, and the i-th block is exactly $P_iP_i^\top$. Both the matrix V and the matrix $\Gamma(R)$ are symmetric, such that we only have to consider $V_{i,i+\Delta i}$, with $0 \leq \Delta i \leq b - i$.

The structure of F_iP for any valid choice of i has already been discussed in Appendix B, such that

$$F_iPP^\top F_j = K_iK_j^\top,$$

with the matrix K_i as defined in (32). From this expression we can see that the blocks are symmetric around the diagonal, i.e., $V_{i,j} = V_{j,i}^\top$, such that we only need to consider the upper triangular part, since the lower triangular part follows from that. We can also see that for any overlap to exist, the difference between i and $j \geq i$, denoted by Δi, has to be limited to be at most ℓ, i.e., $0 \leq \Delta i \leq \ell$. Each time that Δi increases by 1, the overlap between the blocks decreases by 1 block until no overlap remains. Introduce the matrix $M_{i,j}$ such that

$$M_{i,j} := \begin{bmatrix} P_iP_i^\top & & & \\ & P_{i+1}P_{i+1}^\top & & \\ & & \ddots & \\ & & & P_jP_j^\top \end{bmatrix}.$$

For $\Delta i = 0$, we find

$$F_iPP^\top F_i^\top = M_{i,i+\ell}.$$

For $\Delta i = 1$, this becomes

$$F_i P P^\top F_{i+1}^\top = \begin{bmatrix} 0_{q_{\text{ext}} \times q_{\text{ext}} \ell} & 0_{q_{\text{ext}} \times q_{\text{ext}}} \\ M_{i+1,i+\ell} & 0_{q_{\text{ext}} \ell \times q_{\text{ext}}} \end{bmatrix}.$$

The general pattern above the diagonal is then given by

$$V_{i,i+\Delta i} = \begin{bmatrix} 0_{q_{\text{ext}} \Delta i \times q_{\text{ext}}(\ell+1-\Delta i)} & 0_{q_{\text{ext}} \Delta i \times q_{\text{ext}} \Delta i} \\ M_{i+\Delta i,i+\ell} & 0_{q_{\text{ext}}(\ell+1-\Delta i) \times q_{\text{ext}} \Delta i} \end{bmatrix},$$

which is valid for $\Delta i = (0, 1, \ldots, \ell)$ and $i \leq b - \Delta i$. For higher values of Δi, the matrix $V_{i,i+\Delta i}$ is zero. The second condition limits Δi to lower values if the index i is close to its maximum value of b. Recall that elements of the bottom triangular part ($V_{i+\Delta i,i}$) are given by the transposed matrices $V_{i,i+\Delta i}^\top$.

E Computational complexities

This section will discuss the computational complexity of computing the different matrices and terms used in Algorithm 1.

V: For the first step, the computation of blocks $V_{k,l}$, we know that each of the b block rows consists of only ℓ non-zero blocks to compute after taking into account symmetry. Furthermore, each block away from the diagonal is a shifted sub-matrix of the diagonal block, such that the computational burden is limited to computing the b diagonal blocks. For each diagonal block we require $\ell + 1$ matrix multiplications of the form $P_i P_i^\top$, which take at most $O(q q_{\text{ext}}^2)$ FLOPs, but each diagonal block elements has a significant overlap in required multiplications, such that we only require all T $P_i P_i^\top$ multiplications, which can as such be done in $O(q q_{\text{ext}}^2 T)$ FLOPs, which is linear in T, the largest term.

$\mathbf{s}(\theta)$ and $\mathbf{z}(\theta)$: In the second step, the vector $s(\theta)$ is obtained by computing the matrix-vector product $R(\theta)c_i$ for each column c_i of the Hankel matrix $H_{\ell+1}(\rho' \otimes w)$. At the start of the algorithm, we can compute and store $\rho' \otimes w$ as the computation of $s(\theta)$ only requires this trajectory, avoiding the need to store the entire Hankel matrix. Computing $s(\theta)$ takes $O(p q_{\text{ext}}(\ell+1)(T-\ell))$ FLOPs, which is equal to $O(p q_{\text{ext}} \ell T)$. Computing $z(\theta)$ can be done efficiently by noticing that $\Gamma(\theta)$ is a $(p(\ell+1)-1)$-banded matrix, such that the system of equations can be solved in about $O((p(\ell+1)-1)^2 T)$ FLOPs (see, e.g., [31, Chapter 4.3.5]), which simplifies to $O((p\ell)^2 T)$. In general q_{ext}, p, and ℓ are of similar size, making the total number of computations for both $s(\theta)$ and $z(\theta)$ equal to $O((p q_{\text{ext}} \ell + (p\ell)^2) T)$ FLOPs.

$\mathbf{f}(\theta)$: In the next step, for computing $f(\theta)$, we consider two cases. We either have $s(\theta)$ and $z(\theta)$ available, or we do not. The second case happens during Levenberg-Marquardt, since the function value at each proposed next θ has to be evaluated, while the first case happens for $f(\theta)$ at the current value, for which the vectors can be reused from the Jacobian computation. In the first case, it is only required to compute $z^\top(\theta) s(\theta)$, which takes $O(pT)$ FLOPs. In the second case, we have to again compute both vectors, taking the total cost to $O((p q_{\text{ext}} \ell + (p\ell)^2) T)$ FLOPs, which is the same as above, since the contribution of the term pT is negligible compared to the cost of computing both vectors.

J: Computing the Jacobian is the most computationally expensive step of the algorithm, and has to be computed exactly once in each iteration. For each column of the Jacobian, the first term $\frac{\partial G}{\partial R_{i,j}} z(\theta)$ is dominated by the second term, such that only the second term will be considered in the complexity analysis. First consider $z_j^{(1)}$. We can pre-compute $R^\top Z$ in $O(p q_{\text{ext}} \ell T)$ FLOPs. Each term in the sum has $\ell + 1$ non-zero elements, which each take $O(q_{\text{ext}} \ell)$ FLOPs, since the left multiplication with $e^{(j)}$ just selects the correct row of $V_{\cdot,k}$. The computation of a single $z_j^{(1)}$ takes $O(q_{\text{ext}} \ell^2 T)$ FLOPs. In total, the computation of all $q_{\text{ext}}(\ell + 1)$ $z_j^{(1)}$s takes $O(q_{\text{ext}}^2 \ell^3 T)$ FLOPs, and the influence of the $R^\top Z$ computation is negligible. For the vector $z_{i,j}^{(2)}$, each term in the summation only has $\ell + 1$ non-zero elements, and the right multiplication with $e^{(j)}$ selects the correct column of $V_{[:,k]}$. Since only the scalar $Z_{[i,k]}$ depends on i, we can compute the sum for each i value at once,

adding a complexity of $O(p\ell T)$ for each j value of $z_{i,j}^{(2)}$. This term is negligible compared to the cost of computing the sums themselves, which takes $O(pq_{\text{ext}}\ell^2 T)$ FLOPs for each of the $q_{\text{ext}}(\ell+1)$ values of j. Computing all $z_{i,j}^{(2)}$s can be done in $O(pq_{\text{ext}}^2\ell^3 T)$ FLOPs. Computing $z_{i,j}$ from $z_j^{(1)}$ and $z_{i,j}^{(2)}$ takes considerably less FLOPs to compute, and is as such dominated by those two terms. Each Jacobian column requires solving a system of equations with $\Gamma(R)$ and a multiplication with $G^\top(R)$. As discussed before, the system of equations can be solved in $O((p\ell)^2 T)$ FLOPs. If $G(R)$ is constructed as a sparse matrix, it has at most $pq_{\text{ext}}(\ell+1)(T-\ell)$ non-zero elements, or $O(pq_{\text{ext}}\ell T)$ elements. Each column of the Jacobian, as such, requires an additional $O(((p\ell)^2 + pq_{\text{ext}}\ell)T)$ FLOPs, which is dominated by the computation of $z_{i,j}$, such that the final complexity of computing the Jacobian becomes $O(pq_{\text{ext}}^2\ell^3 T)$ FLOPs.

F Proof of Theorem 2

This theorem will follow directly from [27, Theorem 3.2]. Before we can invoke it, we require three assumptions to be met:

- Sufficient descent: there exists a constant C_d such that the predicted improvement $m_j(0) - m_j(\Delta\theta)$ is always greater than $C_d \frac{\|\nabla f(\theta_j)\|_2^2}{\|J_j\|_2^2+\gamma_j}$, where J_j denotes the Jacobian of $\Delta w^*(\theta)$ at iteration j.
- The function $f(\theta)$ is continuously differentiable in an open set containing the level set $\{\theta \in \mathbb{R}^{n_\theta} \mid f(\theta) \leq f(\theta_0)\}$ with Lipschitz-continuous gradient on this level set with constant $\nu > 0$.
- The Jacobian J of $\Delta w^*(\theta)$ is uniformly bounded for all $\theta \in \mathbb{R}^{n_\theta}$, i.e., the spectral norm $\|J\|_2$ is upper bounded by a value $C > 0$ for all $\theta \in \mathbb{R}^{n_\theta}$.

The first point is simple to show. Given the fact that we compute $\Delta\theta_j$ as the global minimizer of $m_j(\Delta\theta)$, we are guaranteed to maximize the predicted improvement. Since this assumption is just to check that our step is at least as good as the Cauchy step, i.e., the minimizer of $m_j(\Delta\theta)$ along the negative gradient [27], picking the optimal step certainly fulfills this criterion. We will come back to second point, and will first show that the Jacobian is upper bounded in spectral norm for all $\theta \in \mathbb{R}^{n_\theta}$. The second bullet point will follow from this.

We start by noting that $\Delta w^*(\theta)$ is given by

$$\Delta w^*(\theta) = G(\theta)^\top \Gamma^{-1} G(\theta) \operatorname{col}(w) = \Pi(\theta) \operatorname{col}(w), \tag{37}$$

where $\Pi(\theta)$ denotes the projection matrix onto the column space of $G(\theta)$. Furthermore, we also have

$$\Pi(\theta) = G(\theta)^\dagger G(\theta). \tag{38}$$

These equations will be used later. The Jacobian of $\Delta w^*(\theta)$ is given by

$$J(\theta) = \begin{bmatrix} \frac{\partial \Delta w^*}{\partial \theta_1} & \frac{\partial \Delta w^*}{\partial \theta_2} & \cdots & \frac{\partial \Delta w^*}{\partial \theta_{n_\theta}} \end{bmatrix} (\theta),$$

where θ_j indicates the j-th element of the θ vector instead of the j-th iteration. To show boundedness of the Jacobian we will start by bounding the separate columns, from which the requested follows directly. Before actually bounding the partial derivatives we require bounds of the matrix $G(\theta)$.

Bounding $G(\theta)$ The matrix $G(\theta)$ is affine in θ, i.e., $G(\theta) = G_0 + \sum_{j=1}^{n_\theta} \theta_i G_i$. From this we can already see that $\partial G(\theta)/\partial\theta_j = G_j$ is constant, and as such bounded. We denote this as

$$C_{\partial G} := \max_{j,\theta} \|G_i\|. \tag{39}$$

From our assumption that the smallest singular value of $G(\theta)$ is uniformly lower bounded, i.e.,

$$\sigma_{\min}(G(\theta)) \geq \sigma_G > 0, \tag{40}$$

we immediately find the bounds

$$\left\|G(\theta)^\dagger\right\|_2 \leq \sigma_G^{-1}. \tag{41}$$

and

$$\left\|\Gamma^{-1}(\theta)\right\|_2 \leq \sigma_G^{-2}. \tag{42}$$

These are the most common bounds that will be used throughout the remainder of this proof. The next step is bounding the partial derivatives, from which the uniform boundedness of the Jacobian follows.

Bounding the partial derivatives We focus on the j-th partial derivative, which can be expanded to

$$\frac{\partial \Delta w^*}{\partial \theta_j}(\theta) = \frac{\partial G^\dagger}{\partial \theta_j}(\theta) G(\theta) \operatorname{col}(w) + G(\theta)^\dagger \frac{\partial G}{\partial \theta_j}(\theta) \operatorname{col}(w).$$

We will now bound these two terms separately. The second term can be bounded directly from previously defined bounds:

$$\left\| G(\theta)^\dagger \frac{\partial G}{\partial \theta_j}(\theta) w \right\|_2 \leq \left\| G(\theta)^\dagger \right\|_2 \left\| \frac{\partial G}{\partial \theta_j}(\theta) \right\|_2 \left\| \operatorname{col}(w) \right\|_2 \leq \sigma_G^{-1} C_{\partial G} C_w,$$

with C_w the norm of vector $\operatorname{col}(w)$, which is constant. Bounding the first term requires additional work. We make use of the fact that

$$\frac{\partial G^\dagger}{\partial \theta_j} = \frac{\partial G^\top}{\partial \theta_j} \Gamma^{-1} - G^\top \Gamma^{-1} \frac{\partial \Gamma}{\partial \theta_j} \Gamma^{-1},$$

where the dependence on θ is omitted to improve readability, and using the formula $\partial(A^{-1}) = -A^{-1}\partial(A)A^{-1}$. The first term of this expression is clearly bounded by $C_{\partial G}\sigma_G^{-2}$. The second term can be expanded as

$$-G^\top \Gamma^{-1} G_j G^\top \Gamma^{-1} - G^\top \Gamma^{-1} G G_j^\top \Gamma^{-1} = -G^\dagger G_j G^\dagger - \Pi(\theta) G_j^\top \Gamma^{-1}.$$

Using the fact that $\left\| \Pi(\theta) \right\|_2 = 1$ and the bounds (39), (41) and (42) we find

$$\left\| G^\top \Gamma^{-1} \frac{\partial \Gamma}{\partial \theta_j} \Gamma^{-1} \right\|_2 = \left\| G^\top \frac{\partial \Gamma^{-1}}{\partial \theta_j} \right\|_2 \leq 2 C_{\partial G} \sigma_G^{-2}, \tag{43}$$

which results in the following bound for the partial derivative of $G(\theta)^\dagger$:

$$\left\| \frac{\partial G^\dagger}{\partial \theta_j}(\theta) \right\|_2 \leq C_{\partial G}\sigma_G^{-2} + 2 C_{\partial G}\sigma_G^{-2} = C_A. \tag{44}$$

From this we obtain

$$\left\| \frac{\partial \Delta w^*}{\partial \theta_j}(\theta) \right\|_2 \leq \sigma_G^{-1} C_{\partial G} C_w + \sigma_G^{-1} C_A C_w = C_B. \tag{45}$$

The requested Jacobian bound follows directly from (45):

$$\left\| J(\theta) \right\|_2 \leq \left\| J \right\|_F = \sqrt{\sum_{j=1}^{n_\theta} \left\| \frac{\partial \Delta w^*}{\partial \theta_j}(\theta) \right\|_2^2} \leq \sqrt{n_\theta} C_B = C. \tag{46}$$

All that remains is showing the second bullet point.

Showing Lipschitz continuity of the gradient Given that the Jacobian of $\Delta w^*(\theta)$ is uniformly bounded, it is easy to see that the function $f(\theta)$ is continuously differentiable. The gradient $\nabla f(\theta)$ is given by $J^\top(\theta)\Pi(\theta)w$, and from the reasoning above we have

$$\begin{aligned} \left\| \nabla f(\theta) \right\|_2 &= \left\| J^\top(\theta) \Pi(\theta) \operatorname{col}(w) \right\|_2 \\ &\leq \left\| J(\theta) \right\|_2 \left\| \operatorname{col}(w) \right\|_2 \\ &\leq C C_w. \end{aligned}$$

This shows that the gradient is uniformly bounded. The gradient is also continuous, such that the function $f(\theta)$ is continuously differentiable for all θ.

A sufficient condition for the gradient $\nabla f(\theta)$ to be Lipschitz continuous is for the Hessian of $f(\theta)$ to be uniformly bounded. The i, j-th element of this matrix is given by

$$\frac{\partial^2 f}{\partial \theta_i \partial \theta_j}(\theta) = \left(\frac{\partial^2 \Delta w^*}{\partial \theta_i \partial \theta_j}(\theta) \right)^\top \Pi(\theta) \operatorname{col}(w) + J^\top \frac{\partial \Delta w^*}{\partial \theta_j}(\theta).$$

The second term is clearly uniformly bounded by CC_B. For the first term we require uniform bounds on the second partial derivatives of $\Delta w^*(\theta)$. We have

$$\begin{aligned}\frac{\partial^2 \Delta w^*}{\partial\theta_i \partial\theta_j} = \Bigg(& G_i^\top \frac{\partial \Gamma^{-1}}{\partial \theta_j} G + G_i^\top \Gamma^{-1} G_j + G_J^\top \Gamma^{-1} G_i + G^\top \frac{\partial \Gamma^{-1}}{\partial \theta_j} G_i \\ & + G_J^\top \frac{\partial \Gamma^{-1}}{\partial \theta_i} G + G^\top \frac{\partial^2 \Gamma^{-1}}{\partial \theta_i \partial \theta_j} G + G^\top \frac{\partial \Gamma^{-1}}{\partial \theta_i} G_j \Bigg) \operatorname{col}(w).\end{aligned}$$

Except for the second-to-last term containing the second partial derivative of Γ^{-1}, it is easy to see that all other terms are uniformly bounded. This follows from (43), the transpose of that, and from

$$G^\top \frac{\partial \Gamma^{-1}}{\partial \theta_j} G = -G^\dagger G_j \Pi - \Pi^\top G_i^\top G^\dagger,$$

which is also uniformly bounded. The final term to be bounded can be expanded to

$$\begin{aligned}G^\top \frac{\partial^2 \Gamma^{-1}}{\partial \theta_i \partial \theta_j} G = & - G^\top \frac{\partial \Gamma^{-1}}{\partial \theta_i} G_i \Pi - G^\dagger G_i \frac{\partial G^\dagger}{\partial \theta_i} G \\ & - G^\top \left(\frac{\partial G^\dagger}{\partial \theta_i} \right) G_i^\top G^\dagger - \Pi^\top G_i^\top \frac{\partial \Gamma^{-1}}{\partial \theta_i} G,\end{aligned}$$

Every term can be uniformly bounded using bounds that we have previously found. The less trivial terms are products of the partial derivative of $G^\dagger$ and G, but it is easy to show that this product is uniformly bounded as well.

In total, we find that each element of the Hessian is uniformly bounded. This means that the Hessian of $f(\theta)$ itself is also uniformly bounded, from which we conclude that the gradient $\nabla f(\theta)$ is Lipschitz continuous.

G Proof of Theorem 3

Proof. The proof consists of two main steps. First, we generalize Theorem 1 to MIMO models, Second, we show that with a correct redefinition of $G(R)$, the matrix still has full row rank for the given parameterization of R.

To start with generalizing Theorem 1, recall the definition of the kernel representation given by (7), formulated for the lifted trajectories $w_{\text{ext}} = \rho' \otimes w$:

$$\mathcal{B} = \ker \mathcal{R}(\sigma) := \left\{ (w, \rho) \in (\mathbb{R}^q \times \mathbb{P})^{\mathbb{N}} \;\middle|\; \mathcal{R}(\sigma)(w_{\text{ext}}) = 0 \right\}.$$

Recall that the kernel operator $\mathcal{R}(\sigma)$ is based on the $p \times q_{\text{ext}}$ polynomial matrix

$$\mathcal{R}(z) = \begin{bmatrix} \mathcal{R}^1(z) \\ \vdots \\ \mathcal{R}^p(z) \end{bmatrix} = \begin{bmatrix} R_0^1 + R_1^1 z + \cdots + R_{\ell_1}^1 z^{\ell_1} \\ \vdots \\ R_0^p + R_1^p z + \cdots + R_{\ell_p}^p z^{\ell_p} \end{bmatrix}.$$

We consider the coefficient matrix of the i-th row

$$R^i(\theta_i) := \begin{bmatrix} R_0^i & R_1^i & \cdots & R_{\ell_i}^i \end{bmatrix}.$$

Each element of the kernel has to be zero for each row of the operator $\mathcal{R}(\sigma)$ separately as well. Considering the equality constraint obtained from the single-output case for the i-th row of the polynomial matrix, we find

$$G_i\big(R^i(\theta_i)\big) \operatorname{col}(w) = s_i(\theta),$$

with

$$s_i(\theta_i) := \operatorname{col}\big(R^i(\theta_i) H_{\ell_i+1}(\rho' \otimes w)\big)$$

and

$$G_i\big(R^i(\theta_i)\big) := \mathcal{M}_T\big(R^i(\theta_i), \rho\big).$$

For a valid MIMO extension, it suffices to take $G\big(R(\theta)\big)$ and $s(\theta)$ as the vertical concatenations of $G_i\big(R^i(\theta_i)\big)$ and $s_i(\theta_i)$, respectively.

To show that the matrix $G\big(R(\theta)\big)$ is still full row rank for each choice of θ we first recall that each $R^i(\theta_i)$ is parameterized such that the fixed '1' element corresponds to the coefficient of $y_i(t)$, the i-th output, and is zero for each of the other output coefficients. This means that each row of $G_i\big(R^i(\theta_i)\big)$ is linearly independent from each $G_j\big(R^j(\theta_j)\big)$, $i \neq j$. The matrix $G_i\big(R^i(\theta_i)\big)$ has exactly $T - \ell_i$ rows and qT columns. Given that each $G_i\big(R^i(\theta_i)\big)$ is full row rank, the matrix $G\big(R(\theta)\big)$ is also full row rank as long as it has at least as many columns as rows. The total number of rows is

$$pT - \sum_{i=1}^{p} \ell_i = pT - \mathbf{n}(\mathcal{B}),$$

which is smaller than $qT = (p+m)T$, the number of columns, such that the matrix $G\big(R(\theta)\big)$ is indeed full row rank. □

H Proof of Lemma 2

Proof. Denote the number of fixed elements by N_f. Based on Assumption 1, N_f is upper bounded as $N_f \leq mT + p\ell$. The matrix $G_x\big(R(\theta)\big)$ has $p(T-\ell)$ rows and $qT - N_f$ columns. A necessary condition for the matrix to be full row rank is that it has at least as many columns as rows, which in this case means

$$p(T-\ell) = qT - (mT + p\ell) \leq qT - N_f,$$

which is always the case.

For the matrix $G_x\big(R(\theta)\big)$ to be row echelon, only the blocks of the form $R_\ell P_i$ are important. Given Assumption 1, the only columns corresponding to output elements that can be removed are situated in the first ℓ blocks, such that no block of the form above can lose a column corresponding to an output. Since each row of the original matrix $G\big(R(\theta)\big)$ has a guaranteed element with value 1 in such a column, this element is still guaranteed in $G_x\big(R(\theta)\big)$ as well. From this we conclude that $G_x\big(R(\theta)\big)$ is row echelon. There are still no zero rows in $G_x\big(R(\theta)\big)$, such that, combined with the necessary condition at the start of the proof, the matrix $G_x\big(R(\theta)\big)$ has full row rank. □