Learning Stable and Robust Linear Parameter-Varying State-Space Models

Chris Verhoek, Ruigang Wang and Roland Tóth

Abstract— This paper presents two direct parameterizations of stable and robust *linear parameter-varying state-space* (LPV-SS) models. The model parametrizations guarantee a priori that for all parameter values during training, the allowed models are stable in the contraction sense or have their Lipschitz constant bounded by a user-defined value γ. Furthermore, since the parametrizations are *direct*, the models can be trained using unconstrained optimization. The fact that the trained models are of the LPV-SS class makes them useful for, e.g., further convex analysis or controller design. The effectiveness of the approach is demonstrated on an LPV identification problem.

I. INTRODUCTION

Systems in engineering are becoming more complex and are continuously being pushed to increase their efficiency, performance and throughput. This makes their behaviors becoming more and more dominated by nonlinearities, which makes the process of modeling these systems much more difficult, as modeling based on first-principles quickly becomes too tedious, costly, and/or inaccurate. Therefore, efficient data-driven modeling tools for these type of engineering systems are getting increasingly more important.

The class of *linear parameter-varying* (LPV) systems has been established to provide a middle ground between the complex, but general, nonlinear system models and the easy-to-use, but rather limited, *linear time-invariant* (LTI) system descriptions. In LPV systems, the signal relations are considered to be linear, just as in the LTI case. However, the parameters that define these relations are assumed to be functions of a measurable, time-varying signal – the so-called *scheduling variable* p, which captures the nonlinear/time-varying effects of the underlying system [1]. The linearity property of LPV systems makes them attractive for modeling, analysis and control and the framework is supported by extensions of many powerful approaches of the LTI framework.

LPV system identification methods [1], [2] have also matured to provide LPV surrogate models of systems based on data. However, despite the many advances, it has remained an open question whether it is possible to *a priori* enforce stability and performance properties on the identified model. Despite the promising results that have been achieved for set

C. Verhoek and R. Tóth are with the Control Systems Group, Eindhoven University of Technology, The Netherlands. R. Wang is with the Australian Centre for Robotics and the School of Aerospace, Mechanical and Mechatronic Engineering, The University of Sydney, Australia. R. Tóth is also with the Vehicle Industry Research Center, Széchenyi István University, Hungary.
C. Verhoek and R. Wang both contributed equally to this paper.
Corresponding author: C. Verhoek (`c.verhoek@tue.nl`).

membership identification based on LPV *input-output* (IO) models [3] with a computationally intensive approach, the problem has remained unsolved for other LPV model classes.

Over the years, *deep-learning*-based system identification methods have been introduced for the data-driven modeling of complex nonlinear systems [4], including methods that focus on LPV models [5]–[7]. Generally, the *recurrent neural network* (RNN) model structures, such as LPV-SS models with NN-based coefficient dependencies has been the main point of interest. This is because such models can provide efficient learning of the (often difficult to model) scheduling dependencies, significantly contributing to the accuracy and automation of the overall modeling process. However, the dynamic nature of RNNs implies that stability of the model plays a significant role in the training [8]. In modeling of general nonlinear systems with RNNs, this stability problem gained interest in recent years [9]–[11] and lead to the developments of so-called implicit ANN network structures [12], which allow for more systematic analysis. Based on this implicit structure, a major research effort has been spent on stability and performance analysis of dynamic neural network models [13], [14], mainly based on Lipschitz and contraction [15] properties of the models. Although promising, many of these techniques require constrained optimization for the training of the networks, due to the enforced stability and/or performance constraint that increases the computational complexity. Inspired by this drawback, *direct* parametrizations of robust and stable RNNs have been introduced in recent years [16], [17], which allow for learning stable and robust deep-learning-based nonlinear models using unconstrained optimization.

In this work, we join the efficient and attractive properties of the LPV framework with the recently introduced direct parametrization approaches that can give a priori stability and performance guarantees. More specifically, as our main contributions, we propose two direct parametrizations of LPV-SS models with NN-based coefficients, which automatically guarantee that the LPV-SS model is stable in terms of contraction or have a prescribed bound on its Lipschitz constant. We achieve this by making use of the Cayley transform, which has been recently applied to achieve similar parametrizations for convolutional neural networks [18]. The added value of the LPV-SS model structure is that the learned model could later be used for further analysis and controller design using the well-established tools of the LPV framework. Moreover, we want to highlight that this a priori guaranteed stability and robustness property of the LPV-SS model is attractive to use in modeling problems where experiment-design is limited in terms of excitation range or

impact on the production process (e.g., tank reactors in the process industry), while the model is expected to accurately describe the system behavior over the entire operating range.

To achieve this, first we introduce the problem setting in Section II, while the proposed solution, i.e., our main result, is given in Section III. We demonstrate the effectiveness of our results on an example in Section IV and the conclusions are drawn in Section V.

Notation: $\mathbb{N}$ denotes the set of non-negative integers and $\mathbb{D}_+^n$ is the set of n-dimensional positive diagonal matrices. $\|\cdot\|_2$ denotes the Euclidian vector norm. For a matrix $A \in \mathbb{R}^{n \times n}$, $\mathrm{tril}(A)$ corresponds to the lower triangular part of A. Given a square matrix M with $I + M$ invertible, its Cayley transform is defined as $\mathrm{Cayley}(M) := (I - M)(I + M)^{-1}$.

II. PROBLEM STATEMENT

Given a data-set $\mathcal{D}_T := \{u_t, p_t, \tilde{y}_t\}_{t=1}^T$ where $u_t \in \mathbb{R}^{n_u}$, $p_t \in \mathbb{R}^{n_p}$, $\tilde{y}_t \in \mathbb{R}^{n_y}$ are input, scheduling, and output signals of length $T \in \mathbb{N}$. We are interested in learning, i.e., identifying, a *linear parameter-varying state-space* (LPV-SS) model $\mathcal{M}_\theta$ via

$$\min_{\theta \in \Theta} \quad \mathcal{L}(\mathcal{M}_\theta(u, p), \tilde{y}) \tag{1}$$

where $\mathcal{L}$ is the ℓ_2-loss of the simulation error, i.e., $\sum_{t=1}^T \|\tilde{y}_t - y_t\|_2^2$, with $y = \mathcal{M}_\theta(u, p)$ describing the forward simulated model response of $\mathcal{M}_\theta$ along the given input and scheduling trajectory (u, p) in $\mathcal{D}_T$ and estimated initial conditions. The model $\mathcal{M}_\theta$ is described as

$$\begin{bmatrix} x_{t+1} \\ y_t \end{bmatrix} = \overbrace{\begin{bmatrix} A(p_t) & B(p_t) \\ C(p_t) & D(p_t) \end{bmatrix}}^{W(p_t)} \begin{bmatrix} x_t \\ u_t \end{bmatrix} + b(p_t), \tag{2}$$

where $x_t \in \mathbb{R}^{n_x}, u_t \in \mathbb{R}^{n_u}, y_t \in \mathbb{R}^{n_y}, p_t \in \mathbb{P} \subseteq \mathbb{R}^{n_p}$ are the state, input, output and scheduling signals at time-instant $t \in \mathbb{N}$, respectively. Here the actual functional dependency on the scheduling p_t of the matrices $A(p_t), \ldots, D(p_t)$ and of $b(p_t)$, which is a possible bias (trimming term), are collected into the function ψ_θ (see Fig. 1). The function

$$\psi_\theta : p \in \mathbb{P} \mapsto \{W, b\}, \tag{3}$$

is considered as a *deep neural network* (DNN) parametrized with $\theta \in \mathbb{R}^{n_\theta}$, which correspond to the learnable parameters. This construction of the LPV model allows for a flexible choice of the dependency structure in $A(p_t), \ldots, D(p_t), b(p_t)$, for instance, one can learn an affine scheduling relationship

$$\begin{bmatrix} \mathrm{Vec}(W(p_t)) \\ b(p_t) \end{bmatrix} = \psi_\theta(p_t) := S_1 p_t + S_0, \tag{4}$$

with $\theta = (S_0, S_1)$ as the learnable parameters. In [5], ψ_θ is considered as a linear mapping while a $\mu(u_t, u_{t-1}, \ldots, y_t, y_{t-1}, \ldots)$ is learnt with a deep-neural network to synthesize the scheduling signal from input-output signals directly as $p_t = \mu(u_t, u_{t-1}, \ldots, y_t, y_{t-1}, \ldots)$. In this paper, we consider the scheduling signal to be given and being part of the data-set $\mathcal{D}_T$.
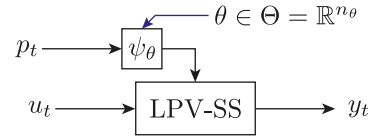

Fig. 1: The LPV state-space model and its parameterized scheduling dependency ψ_θ.

Furthermore, for the sake of simplicity, we consider (2) without a dedicated noise model, under the assumption that the data-generating system has an *output-error* (OE) type of noise structure. Note that estimation under an innovation noise model can be easily incorporated into (1), see [5], and the results of the paper can be easily generalized to the resulting model structure.

In many applications, it is highly desirable to learn LPV-SS models via (1) with stability and robustness guarantees. Especially with a DNN parametrization of the coefficient functions, models estimated along the trajectory $\mathcal{D}_T$ tend to provide deteriorated performance and even unstable behavior when the scheduling trajectory leaves the region where $\mathcal{D}_T$ was obtained, causing much concern in their utilization for industrial applications. To prevent such phenomena occurring, we aim to ensure the following strong notions, which help the model to exponentially forget the initial conditions and generalize to unseen data in a robust and stable manner:

Definition 1. The system represented by (2) is said to be *contracting* if for any two initial conditions $x_0^a, x_0^b \in \mathbb{R}^{n_x}$, any bounded sequences $p \in \mathbb{P}^{\mathbb{N}}$, $u \in (\mathbb{R}^{n_u})^{\mathbb{N}}$, the corresponding state sequences x^a, x^b satisfy

$$\|x_t^a - x_t^b\|_2 \le K\alpha^t \|x_t^a - x_t^b\|_2, \quad \forall t \in \mathbb{N}, \tag{5}$$

for some $K > 0$ and $\alpha \in (0, 1)$.

Definition 2. The system represented by (2) is said to be γ-*Lipschitz* for some $\gamma > 0$, if for any initial state $x_0 \in \mathbb{R}^{n_x}$, bounded parameter sequence $p \in \mathbb{P}^{\mathbb{N}}$, and bounded input sequence pair $(u^a, u^b) \in (\mathbb{R}^{2n_u})^{\mathbb{N}}$, the corresponding output pair (y^a, y^b) satisfies

$$\sum_{t=0}^T \|y_t^a - y_t^b\|_2^2 \le \gamma^2 \sum_{t=0}^T \|u_t^a - u_t^b\|_2^2, \quad \forall T \in \mathbb{N}. \tag{6}$$

Using these definitions, we solve the following problems in this paper:

Problem 1. Construct the model parameterizations

$$\mathcal{M}^c := \{\mathcal{M}_\theta \mid \mathcal{M}_\theta \text{ is contracting } \forall \theta \in \mathbb{R}^{n_\theta}\}, \tag{7a}$$

$$\mathcal{M}^\gamma := \{\mathcal{M}_\theta \mid \mathcal{M}_\theta \text{ is } \gamma\text{-Lipschitz } \forall \theta \in \mathbb{R}^{n_\theta}\}. \tag{7b}$$

Remark 1. The following remarks are important:

- With the parameterizations in (7), the learning problem (1) can be formulated as an unconstrained optimization problem that can be solved by off-shelf first-order methods (e.g., stochastic gradient descent). This is because $\theta \in \Theta = \mathbb{R}^{n_\theta}$.
- The γ-Lipschitz property is equivalent to having an incremental ℓ_2-gain bound of γ on (2), which in turn *implies* an ℓ_2-gain bound of γ on (2) [19].

- See also [15] for the connections between contraction and incremental stability.

III. MAIN RESULTS

In this section, we first give sufficient conditions for contracting/γ-Lipschitz LPV-SS models and then present a direct parameterization such that those conditions are automatically satisfied during training.

A. Stable and robust LPV-SS models

To study the contracting or γ-Lipschitz property of (2), we first consider the error dynamics between two arbitrary trajectories of (2) with the same scheduling signal, i.e., (u^a, x^a, y^a, p) and (u^b, x^b, y^b, p). For these trajectories, the error dynamics are:

$$\begin{bmatrix} \Delta x_{t+1} \\ \Delta y_t \end{bmatrix} = \begin{bmatrix} A(p_t) & B(p_t) \\ C(p_t) & D(p_t) \end{bmatrix} \begin{bmatrix} \Delta x_t \\ \Delta u_t \end{bmatrix}, \tag{8}$$

where $\Delta x = x^a - x^b$, $\Delta u = u^a - u^b$ and $\Delta y = y^a - y^b$. Then, (2) is contracting if (8) is exponentially stable, while (2) is γ-Lipschitz if (8) has an ℓ_2-gain bound of γ.

Proposition 1. *The LPV-SS model* (2) *describes a contracting system, if there exist a $\mathcal{X} \succ 0$ and an $\alpha \in (0, 1]$ s.t.*

$$\alpha^2 \mathcal{X} - A^\top(p) \mathcal{X} A(p) \succ 0, \quad \forall p \in \mathbb{P}. \tag{9}$$

The system is γ-Lipschitz, if there exist a $\mathcal{X} \succ 0$ s.t.

$$\begin{bmatrix} \mathcal{X} & 0 \\ 0 & \gamma^2 I \end{bmatrix} - W^\top(p) \begin{bmatrix} \mathcal{X} & 0 \\ 0 & I \end{bmatrix} W(p) \succ 0, \quad \forall p \in \mathbb{P}. \tag{10}$$

Proof. Contraction of the system represented by (2) is defined for the differential state under the same input sequence, hence (8) with $\Delta u_t = 0$ becomes

$$\Delta x_{t+1} = A(p_t) \Delta x_t. \tag{11}$$

Based on (9), we have

$$\alpha^2 V(\Delta x_t) \geq V(\Delta x_{t+1}), \tag{12}$$

where $V(\Delta x) = \Delta x^\top \mathcal{X} \Delta x$, showing exponential (Lyapunov) stability of the error dynamics. This implies that the corresponding LPV-SS model is contracting.

To prove the γ-Lipschitz property of (2), we first multiply (10) from the left and right with $\begin{bmatrix} \Delta x_t^\top & \Delta u_t^\top \end{bmatrix}$ and $\begin{bmatrix} \Delta x_t^\top & \Delta u_t^\top \end{bmatrix}^\top$, respectively. This leads to

$$\gamma^2 \|\Delta u_t\|_2^2 - \|\Delta y_t\|_2^2 \geq V(\Delta x_{t+1}) - V(\Delta x_t). \tag{13}$$

Using a telescoping sum based on the above inequality and that $\Delta x_0 = 0$, (6) is satisfied. ∎

B. Model parameterization via Cayley transform

The challenge in estimating ψ_θ and ensuring stability of (2) is that Condition (9) needs to hold for all $p \in \mathbb{P} \subset \mathbb{R}^{n_\mathrm{P}}$, representing an infinite-dimensional constraint that is required to be added to (1). While it is possible to achieve some relaxation of this constraint, e.g., by restricting ψ_θ to be linear and $\mathbb{P}$ to a convex polytope and turn (9) to a finite *semi-definite programming* (SDP) problem, such relaxations

(i) seriously restrict the representable class of systems and (ii) still involve a significant amount of computation time, which can quickly make the training intractable. We tackle those issues by deriving an analytic solution to (9).

Theorem 1. *The model* (2) *defined by coefficient function ψ_θ satisfies* (9)*, if and only if there exist $d \in \mathbb{R}^{n_\mathrm{x}}$, $\alpha \in (0, 1]$, $\mathcal{Y} \in \mathbb{R}^{n_\mathrm{x} \times n_\mathrm{x}}$ and a mapping $\phi : p \mapsto (X, Y)$ with $X(p), Y(p) \in \mathbb{R}^{n_\mathrm{x} \times n_\mathrm{x}}$ such that*

$$A(p) = \alpha Q \Lambda^{-1} M(p) \Lambda Q^\top, \tag{14}$$

with $\Lambda = \mathrm{diag}(e^d)$ and

$$Q = \mathrm{Cayley}(\mathcal{Y} - \mathcal{Y}^\top), \quad M(p) = \mathrm{Cayley}(N(p)), \tag{15}$$

where $N(p) = X^\top(p) X(p) + Y(p) - Y^\top(p) + \epsilon I$ for some small positive constant ϵ.

Proof. We first show that (9) $\Leftrightarrow$ (14) and then we prove that the invertible mapping between $\Lambda, Q, M(p)$ and $d, \mathcal{Y}, X(p), Y(p)$ can be easily established based on Lemmas 1 and 2, which are given in the Appendix. For the sake of notational simplicity, we use subscript p to denote the dependency on the scheduling variable.

We first show that (14) $\Rightarrow$ (9). By taking $\mathcal{X} = Q \Lambda^2 Q^\top$, $\mathcal{X} \succ 0$ as $Q Q^\top = I$ due to Lemma 2. Then,

$$\alpha^2 \mathcal{X} - A_p^\top \mathcal{X} A_p = \alpha^2 Q \Lambda (I - M_p^\top M_p) \Lambda Q^\top \succ 0, \tag{16}$$

where positive definiteness of $I - M_p^\top M_p$ follows by Lemma 1. Next, we show (9) $\Rightarrow$ (14). Since $\mathcal{X} \succ 0$, its *singular value decomposition* (SVD) has the form $\mathcal{X} = Q \Sigma Q^\top$ with $\Sigma \in \mathbb{D}_+^{n_\mathrm{x}}$ and $Q^\top Q = I$, and Q cannot have -1 as an eigenvalue. By letting $\Lambda = \Sigma^{1/2}$, we have that

$$(9) \Rightarrow I - M_p^\top M_p \succ 0, \tag{17}$$

where $M_p = \frac{1}{\alpha} \Lambda Q^\top A_p Q \Lambda^{-1}$, which gives (14). ∎

Thm. 1 reveals that we can represent any ψ_θ coefficient function parametrization for which the defined model (2) satisfies (9) by the parameters $d, \mathcal{Y}$ and unconstrained mapping

$$\phi_{\tilde{\theta}} : p \mapsto (X, Y, B, C, D, b),$$

which can be chosen as a DNN parametrized in $\tilde{\theta}$. This means that we can transform the learnable parameters θ to new parameters $\{d, \mathcal{Y}, \tilde{\theta}\}$ that guarantee that, for any value of them, the corresponding model (2) satisfies (9).

In fact, we can use any parameterization for $\phi_{\tilde{\theta}}$, like a simple linear mapping (4), or a polynomial parametrization, etc. This underlines the applicability of Thm. 1 beyond deep-learning-based identification of LPV models. Similar results can be derived for the γ-Lipschitz property.

Theorem 2. *The model* (2) *defined by coefficient function ψ_θ satisfies* (10)*, if and only if there exist $d \in \mathbb{R}^{n_\mathrm{x}}$, $\mathcal{Y} \in \mathbb{R}^{n_\mathrm{x} \times n_\mathrm{x}}$ and a mapping $\phi : p \mapsto (X, Y, Z)$ with $X(p), Y(p) \in \mathbb{R}^{n \times n}$ and $Z(p) \in \mathbb{R}^{n_0 \times n}$, where $n = n_\mathrm{x} + \min(n_\mathrm{u}, n_\mathrm{y})$ and $n_0 = |n_\mathrm{y} - n_\mathrm{u}|$, such that*

$$W(p) = \begin{bmatrix} Q \Lambda^{-1} & 0 \\ 0 & I \end{bmatrix} M(p) \begin{bmatrix} \Lambda Q^\top & 0 \\ 0 & \gamma I \end{bmatrix}, \tag{18}$$

with

$$\begin{bmatrix} \text{Cayley}(N(p)) \\ -2Z(p)(I+N(p))^{-1} \end{bmatrix} = \begin{cases} M(p), & \text{if } n_{\mathrm{y}} \geq n_{\mathrm{u}}, \\ M^{\top}(p), & \text{if } n_{\mathrm{y}} < n_{\mathrm{u}}, \end{cases} \quad (19)$$

where $N(p) = X(p)^{\top}X(p) + Y(p) - Y(p)^{\top} + Z(p)^{\top}Z(p) + \epsilon I$ *with* ϵ *as a small positive constant.*

Proof. We first rewrite (10) as follows

$$\mathcal{X}_{\gamma} - W_p^{\top}\mathcal{X}_I W_p \succ 0 \quad (20)$$

where $\mathcal{X}_{\gamma} = \mathrm{diag}(\mathcal{X}, \gamma^2 I)$ and $\mathcal{X}_I = \mathrm{diag}(\mathcal{X}, I)$. By taking the SVD decomposition $\mathcal{X} = Q\Sigma Q^{\top}$ and letting $\Lambda = \Sigma^{1/2}$, we have $I - M_p^{\top}M_p \succ 0$ where

$$M_p = \begin{bmatrix} \Lambda Q^{\top} & 0 \\ 0 & I \end{bmatrix} W_p \begin{bmatrix} Q\Lambda^{-1} & 0 \\ 0 & \gamma^{-1}I \end{bmatrix}. \quad (21)$$

Then, the techniques used in the proof of Thm. 1 can be directly applied to prove (10) ⇔ (18). ∎

Remark 2. The transformation in (19) can be considered as the Cayley transform for non-square matrices. When $n_{\mathrm{y}} = n_{\mathrm{u}}$, the normal Cayley transform is recovered, as in that case, $Z(p)$ is an empty matrix.

IV. EXAMPLE

With the following example[1], we aim to demonstrate the effectiveness of the proposed robust and stable LPV-SS parametrization for deep-learning-based identification by comparing the training results with these models to the training results under a general LPV model structure.

A. Data-generation

The data-generating system is considered to be in an LPV-SS form with output noise:

$$x_{t+1} = A^{\mathrm{d}}(p_t)x_t + B^{\mathrm{d}}(p_t)u_t, \quad (22\mathrm{a})$$
$$\tilde{y}_t = C^{\mathrm{d}}(p_t)x_t + D^{\mathrm{d}}(p_t)u_t + e_t, \quad (22\mathrm{b})$$

where, at time $t \in \mathbb{N}$, $u_t \in \mathbb{R}$ is the input, $p_t \in \mathbb{R}^3$ is the scheduling, $x_t \in \mathbb{R}^3$ is the state, and $\tilde{y}_t \in \mathbb{R}$ is the output that is disturbed by an i.i.d. white noise signal $e_t \sim \mathcal{N}(0, 0.08)$. The matrices $A^{\mathrm{d}}, \ldots, D^{\mathrm{d}}$ have static-affine dependence on p_t, i.e., $A^{\mathrm{d}}(p_t), \ldots, D^{\mathrm{d}}(p_t)$ are of the form $X(p_t) = X_0 + \sum_{i=1}^{n_{\mathrm{p}}} X_i p_{i,t}$ with

$$A_0^{\mathrm{d}} = \begin{bmatrix} -0.3885 & -0.1912 & 0.1631 \\ 0.3261 & -0.2583 & -0.9150 \\ -0.1664 & -0.1384 & 0.0768 \end{bmatrix}, \quad B_0^{\mathrm{d}} = \begin{bmatrix} -3.4269 \\ -0.3316 \\ -2.1006 \end{bmatrix},$$

$$A_1^{\mathrm{d}} = \begin{bmatrix} 0.2650 & -0.2214 & -0.1866 \\ 0.1747 & 0.1687 & -0.5876 \\ -0.0477 & -0.1313 & 0.2863 \end{bmatrix}, \quad B_1^{\mathrm{d}} = \begin{bmatrix} -1.1096 \\ -0.8456 \\ -0.5727 \end{bmatrix},$$

$$A_2^{\mathrm{d}} = \begin{bmatrix} 0.1476 & 0.1390 & 0.0901 \\ -0.1242 & 0.1903 & 0.4027 \\ 0.0403 & 0.0845 & 0.0971 \end{bmatrix}, \quad B_2^{\mathrm{d}} = \begin{bmatrix} -0.5587 \\ 0.1784 \\ -0.1969 \end{bmatrix},$$

$$A_3^{\mathrm{d}} = \begin{bmatrix} 0.1613 & -0.0998 & -0.1652 \\ 0.0349 & 0.0645 & -0.1630 \\ 0.0098 & -0.0529 & 0.0591 \end{bmatrix}, \quad B_3^{\mathrm{d}} = 0_{3\times 1},$$

$$C_0^{\mathrm{d}} = \begin{bmatrix} -0.2097 & 0.0607 & 0.1421 \end{bmatrix}, \quad C_1^{\mathrm{d}} = C_2^{\mathrm{d}} = C_3^{\mathrm{d}} = 0_{1\times 3},$$

$$D_0^{\mathrm{d}} = 0.3, \quad D_1^{\mathrm{d}} = 0.01, \quad D_2^{\mathrm{d}} = 0, \quad D_3^{\mathrm{d}} = 0.04.$$

---

[1]The data-sets and code used for this example can be found at https://tinyurl.com/robstablpv.

TABLE I: Specifications of the generated data-sets

Item \ Data-set	TRAINING	VALIDATION	TEST-A	TEST-B
Range u_t	$[-1, 1]$	$[-1, 1]$	$[-1, 1]$	$[-20, 20]$
Range p_t	$0.3\mathbb{P}$	$0.3\mathbb{P}$	$0.3\mathbb{P}$	$\mathbb{P}$
T	200	200	200	6000
N_{b}	3200	1280	30	1

For this system, $A^{\mathrm{d}}(p_t)$ satisfies that the spectral radius of $A^{\mathrm{d}}(p_t)$ is less than 1 for $p_t \in [-1, 1] \times [0, 4] \times [-2, 2] = \mathbb{P}$, which is considered as the scheduling range.

From (22), four data-sets are obtained: one TRAINING and VALIDATION data-set and two test-sets; TEST-A and TEST-B. We have generated these sets by applying an input to (22) that is constructed with a white noise-signal with variance 0.05 added to a multi-sine. The multi-sine signal contains 10 sinusoidal components evenly distributed over the full normalized frequency spectrum. The scheduling signal is taken as a white noise with a uniform distribution over $\mathbb{P}$. The data-sets are composed of N_{b} trajectories, each of length T. The generated data-sets and their individual length-T trajectories are uncorrelated. The specific details for the generated data-sets are listed in Table I. Hence, data-set TEST-B is excited by and scheduled with an input and scheduling that are *outside* the range represented in the TRAINING and VALIDATION data-sets. The generated data-sets are shown in Figs. 2–5. We want to highlight that with the aforementioned specification on the output-noise e_t, the *signal-to-noise ratio* (SNR) for the TRAINING, VALIDATION and TEST-A data-sets is 12 dB. This implies that the lowest possible *normalized root-mean-square error* (NRMSe) that we can achieve when simulating the trained models is approximately 25%.

B. Considered model structures

To learn, i.e., identify, (22), we consider the γ-Lipschitz LPV-SS model parametrization of Thm. 2 with the following hyperparameters: The state-dimension of the γ-Lipschitz LPV-SS model is chosen as $n_{\mathrm{x}} = 3$. The mapping $\phi_{\tilde{\theta}} : p \mapsto (X, Y, b)$ according to Thm. 2 is chosen as a feedforward neural network for each component with 2 hidden layers, each with 50 ReLU activation neurons and a linear in- and output layer (note that Z is empty). The value for γ is set to 1, such that the model is ensured to have a Lipschitz bound of 1. Note that it is always possible to perform a hyperparameter optimization for γ to improve the performance of the model.

The results of the identification with the γ-Lipschitz LPV-SS model are compared to estimation of an LPV model given by the following *linear fractional representation* (LFR):

$$\begin{bmatrix} x_{t+1} \\ z_t \\ y_t \end{bmatrix} = \begin{bmatrix} A(p_t) & B_{\mathrm{w}}(p_t) & B_{\mathrm{u}}(p_t) \\ C_{\mathrm{z}}(p_t) & 0 & D_{\mathrm{zu}}(p_t) \\ C_{\mathrm{y}}(p_t) & D_{\mathrm{yw}}(p_t) & D_{\mathrm{yu}}(p_t) \end{bmatrix} \begin{bmatrix} x_t \\ w_t \\ u_t \end{bmatrix} + \begin{bmatrix} b_x(p_t) \\ b_z(p_t) \\ b_y(p_t) \end{bmatrix},$$
$$w_t = \sigma(z_t), \quad (23)$$

where $\sigma : \mathbb{R}^{n_{\mathrm{z}}} \to \mathbb{R}^{n_{\mathrm{w}}}$ is a ReLU activation function and $A, \ldots, D_{\mathrm{yu}}$ have affine dependence on p_t. Note that the data-generating system (22) is contained in the model structure corresponding the LPV-LFR model (23). For this
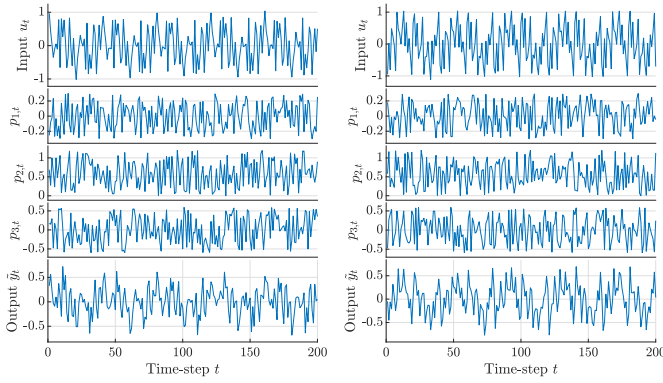
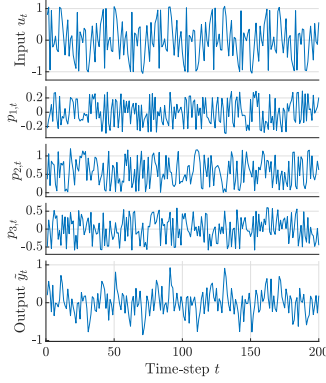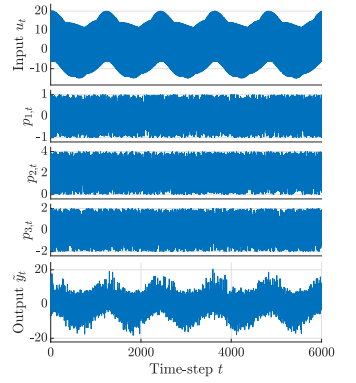Fig. 2: TRAINING data-set   Fig. 3: VALIDATION data-set   Fig. 4: TEST-A data-set   Fig. 5: TEST-B data-set

model, we choose the following hyperparameters: The state-dimension is chosen as $n_x = 3$. The dependency of the matrices in (23) on p_t is, as aforementioned, static-affine. The dimension of w_t and z_t is 100, which implies that the corresponding NN component has one hidden layer with 100 neurons. The models are initialized randomly with matrices that have entries between -0.1 and 0.1.

C. Training of the models

We choose Adam [20] as the optimizer with a learning-rate of 10^{-2} to minimize the loss function $\mathcal{L}$ in (1). During training (similarly for validation), we forward simulate the models for the trajectories in the TRAINING data-set, i.e., for 200 steps, initialized with a random initial condition taken from a normalized uniform distribution. To eliminate the effect of the wrongly chosen initial condition, we compute the simulation loss from $t = 10$. The models are trained for 20 epochs with a batch size of N_b during optimization.

D. Comparison of the results

After training the models, we forward simulated them on both the TEST-A and TEST-B data-sets, again initialized with a random initial condition taken from a normalized uniform distribution. To assess the correctness of the training result, we measure the simulation accuracy using the NRMSe, i.e.,

$$\text{NRMSe}(\tilde{y}, y) = \frac{1}{n_y} \sum_{i=1}^{n_y} \frac{\left(\frac{1}{T} \sum_{t=1}^{T} (\tilde{y}_{i,t} - y_{i,t})^2\right)^{\frac{1}{2}}}{\text{std}(\tilde{y}_i)},$$

with $\text{std}(\tilde{y})$ the sample standard deviation of the measured output sequence $\tilde{y}$. The simulation results for both models on data-set TEST-A are shown in Fig. 6, while the simulation responses for both models on data-set TEST-B are shown in Fig. 7. In Fig. 6, we only show the simulation result that had the lowest NRMSe out of the 30 responses coming from TEST-A. The average NRMSe over the 30 trajectories is 0.4507 for the γ-Lipschitz LPV-SS model and 0.3378 for the LPV-LFR model.

The results in Fig. 6 show that after 20 epochs of training, the LPV-LFR model resulted in a better prediction model for the data-generating system (22) in terms of the NRMSe. This can be caused by a too conservative choice of γ for

the γ-Lipschitz model to represent the actual dynamics and the more simple parametrization of the LPV-LFR compared to the γ-Lipschitz model, which could result in a faster convergence during optimization.

The strength of the γ-Lipschitz model comes forward when we simulate the trained models on the TEST-B data-set. As can be observed in Fig. 7, the response of the γ-Lipschitz model to a scheduling that goes beyond the range that was seen during training still respects the Lipschitz bound of 1, while the output response of the LPV-LFR model explodes in terms of magnitude. In fact, although not further discussed in this paper, when we identify an LPV-SS model using state-of-the-art LPV identification methods [2], [21], we obtain unstable behavior when simulated on TEST-B. As highlighted in Section I, this convenient property of the γ-Lipschitz LPV-SS model makes this parametrization attractive to use for modeling problems in, e.g., the process industry, where the data-generating system is often dependent on many measurable exogenous parameters. In these situations, experiment-design is often limited in terms of excitation due to cost, while, during normal operation, the true underlying system (e.g., reactor) is fed with inputs far outside the excitation range of the experiment. The guaranteed Lipschitz property ensures that the trained LPV-SS model will not behave unexpectedly when simulated with the typical inputs.

V. CONCLUSIONS

This work introduces stable and robust parametrizations of LPV state-space models based on the Cayley transform. By means of contraction theory, we can *a priori* guarantee *global* stability and performance (in terms of γ-Lipschitz) properties of the to-be-trained model. The proposed model parametrizations are highly flexible and require no further constraints or optimization based stability checks compared to alternative solutions. The strength of having these guaranteed properties is demonstrated in an example that considers an LPV system-identification problem.

APPENDIX

The proofs of Thms. 1 and 2 make use of the following lemmas. See [22] for the proofs of these lemmas.
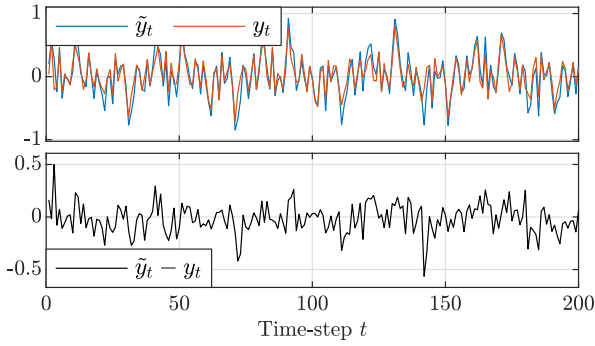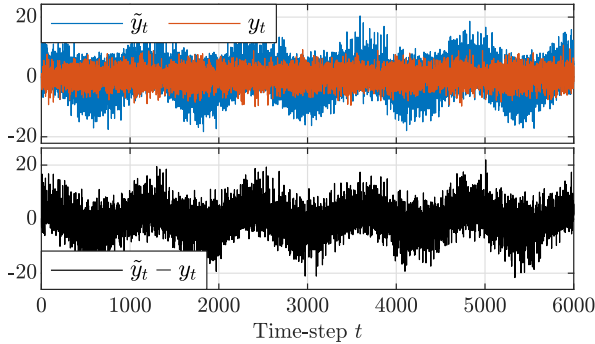
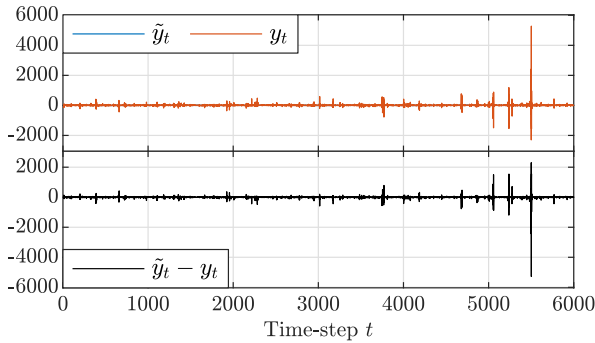(a) γ-Lipschitz LPV-SS model (NRMSe: 0.4179)
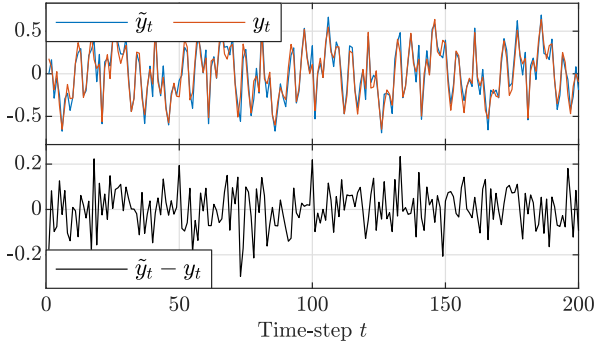
(b) LPV-LFR model (NRMSe: 0.2910)

Fig. 6: Simulation results on the TEST-A data-set, with $\tilde{y}$ the output in the data-set, and y the predicted output of the trained models.

(a) γ-Lipschitz LPV-SS model (NRMSe: 1.0227)

(b) LPV-LFR model (NRMSe: 20.599)

Fig. 7: Simulation results on the TEST-B data-set, with $\tilde{y}$ the output in the data-set, and y the predicted output of the trained models.

Lemma 1. *Let $M \in \mathbb{R}^{n \times m}$ with $n \geq m$. Then, $M^\top M \prec I$ if and only if there exist $X, Y \in \mathbb{R}^{m \times m}$ and $Z \in \mathbb{R}^{(n-m) \times m}$ such that*

$$M = \begin{bmatrix} \mathrm{Cayley}(N) \\ -2Z(I+N)^{-1} \end{bmatrix} \quad (24)$$

where $N = X^\top X + Y - Y^\top + Z^\top Z + \epsilon I$.

Lemma 2. *Let M be a square matrix that does not have an eigenvalue of -1. Then, $M^\top M = I$ if and only if there exists a square matrix Y such that $M = \mathrm{Cayley}(Y - Y^\top)$.*

REFERENCES

[1] R. Tóth, *Modeling and Identification of Linear Parameter-Varying Systems*, 1st ed. Springer-Verlag, 2010.
[2] P. B. Cox and R. Tóth, "Linear parameter-varying subspace identification: A unified framework," *Automatica*, vol. 123, 2021.
[3] V. Cerone, D. Piga, D. Regruto, and R. Tóth, "Input-output LPV model identification with guaranteed quadratic stability," in *Proc. of the 16th Symp. on System Identification*, 2012, pp. 1767–1772.
[4] J. Sjöberg *et al.*, "Nonlinear black-box modeling in system identification: a unified overview," *Automatica*, vol. 31, pp. 1691–1724, 1995.
[5] C. Verhoek, G. I. Beintema, S. Haesaert, M. Schoukens, and R. Tóth, "Deep-learning-based identification of LPV models for nonlinear systems," in *Proc. of the 61st IEEE Conf. on Decision and Control*, 2022, pp. 3274–3280.
[6] S. Z. Rizvi, J. M. Velni, F. Abbasi, R. Tóth, and N. Meskin, "State-space LPV model identification using kernelized machine learning," *Automatica*, vol. 88, pp. 38–47, 2018.
[7] N. Lachhab, H. S. Abbas, and H. Werner, "A neural-network based technique for modelling and LPV control of an arm-driven inverted pendulum," in *Proc. of the 47th IEEE Conf. on Decision and Control*, 2008, pp. 3860–3865.
[8] N. E. Barabanov and D. V. Prokhorov, "Stability analysis of discrete-time recurrent neural networks," *IEEE Trans. on Neural Networks*, vol. 13, no. 2, pp. 292–303, 2002.
[9] J. Miller and M. Hardt, "Stable recurrent models," in *Proc. of the Int. Conf. on Learning Representations*, 2019, pp. 1–23.
[10] M. Fazlyab, A. Robey, H. Hassani, M. Morari, and G. Pappas, "Efficient and accurate estimation of Lipschitz constants for deep neural networks," *Adv. Neural Inf. Process. Syst.*, vol. 32, 2019.
[11] J. Cohen, E. Rosenfeld, and Z. Kolter, "Certified adversarial robustness via randomized smoothing," in *Proc. of the Int. Conf. on Machine Learning*, 2019, pp. 1310–1320.
[12] L. El Ghaoui *et al.*, "Implicit deep learning," *SIAM J. on Mathematics of Data Science*, no. 3, pp. 930–958, 2021.
[13] M. Revay, R. Wang, and I. R. Manchester, "Lipschitz bounded equilibrium networks," *Preprint arXiv:2010.01732*, 2020.
[14] P. Pauli, A. Koch, J. Berberich, P. Kohler, and F. Allgöwer, "Training robust neural networks using Lipschitz bounds," *IEEE Control Systems Letters*, vol. 6, pp. 121–126, 2021.
[15] W. Lohmiller and J.-J. E. Slotine, "On contraction analysis for non-linear systems," *Automatica*, vol. 34, no. 6, pp. 683–696, 1998.
[16] R. Wang and I. Manchester, "Direct parameterization of lipschitz-bounded deep networks," in *Proc. of the Int. Conf. on Machine Learning*, 2023, pp. 36 093–36 110.
[17] M. Revay, R. Wang, and I. R. Manchester, "Recurrent equilibrium networks: Flexible dynamic models with guaranteed stability and robustness," *IEEE Trans. on Automatic Control*, 2023.
[18] P. Pauli, R. Wang, I. R. Manchester, and F. Allgöwer, "Lipschitz-bounded 1D convolutional neural networks using the Cayley transform and the controllability Gramian," *Preprint arXiv:2303.11835*, 2023.
[19] C. Verhoek, P. J. W. Koelewijn, S. Haesaert, and R. Tóth, "Convex incremental dissipativity analysis of nonlinear systems," *Automatica*, vol. 150, p. 110859, 2023.
[20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *Preprint arXiv:1412.6980*, 2014.
[21] P. den Boef, P. B. Cox, and R. Tóth, "LPVCORE: MATLAB toolbox for LPV modelling, identification and control of non-linear systems," in *Proc. of the 19th Symp. on System Identification*, 2021, pp. 385–390.
[22] C. Verhoek, R. Wang, and R. Tóth, "Learning stable and robust linear parameter-varying state-space models," *Preprint arXiv:2304.01828*, 2023.