

LPVcore: MATLAB Toolbox for LPV Modelling, Identification and Control

Pascal den Boef* Pepijn B. Cox**** Roland Tóth**,***

* *Drebble, Horsten 1, 5612AX, Eindhoven, The Netherlands*

** *Control Systems Group, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands*

*** *Systems and Control Laboratory, Institute for Computer Science and Control, Kende u. 13-17, H-1111 Budapest, Hungary.*

**** *Radar Technology, TNO, P.O. Box 96864, 2509 JG The Hague, The Netherlands*

Abstract: This paper describes the LPVCORE software package for MATLAB developed to model, simulate, estimate and control systems via *linear parameter-varying* (LPV) *input-output* (IO), *state-space* (SS) and *linear fractional* (LFR) representations. In the LPVCORE toolbox, basis affine parameter-varying matrix functions are implemented to enable users to represent LPV systems in a global setting, i.e., for time-varying scheduling trajectories. This is a key difference compared to other software suites that use a grid or only LFR-based representations. The paper contains an overview of functions in the toolbox to simulate and identify IO, SS and LFR representations. Based on various prediction-error minimization methods, a comprehensive example is given on the identification of a DC motor with an unbalanced disc, demonstrating the capabilities of the toolbox. The software and examples are available on www.lpvcore.net.

Copyright © 2021 The Authors. This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0>)

Keywords: Linear parameter-varying systems, software tools, system identification

1. INTRODUCTION

In recent years, the *linear parameter-varying* (LPV) modelling paradigm has received considerable attention from the identification and control community (e.g., see Tóth, 2010; Lopes dos Santos et al., 2011; Mohammadpour and Scherer, 2012; Sename et al., 2012; Briat, 2015; Hoffmann and Werner, 2015). The LPV framework has become popular as it can represent non-linear and time-varying behavior encountered in real-world systems while, at the same time, the linearity in the model can be exploited to have convex optimization for synthesizing observers and controllers including performance guarantees. An LPV representation describes a linear relation between the inputs and the outputs of the model, but this linear relation is a function of a measurable, time-varying signal, the so-called *scheduling signal* p . These variations w.r.t. the scheduling signal enable the representation of non-linear and non-stationary behaviour. Conceptually, LPV models of the underlying system can be obtained by 1) interpolating various local linear time-invariant representations around fixed operating points, i.e., with constant p , often referred to as the *local* or grid based model; or by 2) formulating one overall representation of the system via the so-called embedding principle, i.e., the *global* form. In general, global models capture the dynamics of the system w.r.t. the scheduling signal and, for local models, the scheduling signal is used to interpolate (schedule) between the local linear time-invariant models.

The LPV analysis, modelling, identification, and control literature has become mature in recent years. To accelerate

the applicability of LPV methodologies to applications, various computational tools have become available, e.g., LPVTools (Hjartarson et al., 2015), Control Systems and System Identification Toolbox in MATLAB (MATLAB, 2014), *Predictor-Based Subspace IDentification* (PBSID) Toolbox (van Wingerden and Verhaegen, 2009), and *LPV Input/Output Systems Identification Toolbox* (LPVOID) (Rabiei et al., 2015). The analysis and control methods in the LPVTools and Control Systems toolboxes are mainly focused on a local grid based formulation (Jacobian-linearizations) or a grid based linear fractional representation. The Control Systems Toolbox does include a limited set of tools based on a global form. The PBSID and LPVOID toolboxes focus on specific algorithms for identification of global LPV models in either SS or IO form, respectively, but with limited options to further utilize these models and with only support for affine dependency on the scheduling signal.

The concept of the initial release of the LPVCORE toolbox is to facilitate modelling, simulation, and identification of LPV representations in a global form in MATLAB. Modelling and simulating systems can be performed based on a *continuous-time* (CT) or *discrete-time* (DT) LPV representation. To include the parameter-varying nature, the real and complex number based matrix representations in MATLAB are extended to facilitate parameter-variations. These parameter-varying matrix functions form the core of the toolbox to facilitate algebraic operations. In addition, the LPVCORE toolbox includes identification methods to identify discrete-time LPV IO, SS and LFR models. The initial release of the toolbox does not cover the comprehensive LPV literature. However, it provides a solid basis to incorporate LPV analysis and control methods, implement

¹ This work has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement nr. 714663).

realization tools in a computationally efficient manner, and extend the set of identification methods in the future.

The intention of this paper is to provide an overview of the methods in the initial release of the LPVCORE toolbox. In Section 2, the handling of parameter-varying matrix functions and the various LPV representation forms are introduced. The LPV-IO, LPV-SS and LPV-LFR identification methods are described in Section 3. Section 4 presents an example in which LPVCORE is used to estimate a model of a DC motor with unbalanced disc. In Section 5, the conclusions are given.

2. LPV REPRESENTATIONS

In this section, the LPV representations are described to model the system at hand. The key element in formulating the representations in a global form is the parameter-varying matrix object of the LPVCORE toolbox, as discussed in Section 2.1. With the parameter-varying matrix object, LPV input-output, state-space and linear-fractional representations are constructed, as detailed in Sections 2.2, 2.4 and 2.3.

2.1 Parameter-varying matrix function

The LPV representations are defined using *parameter-varying matrix functions*. MATLAB does not have a native implementation of these functions. Therefore, LPVCORE introduces the `pmatrx` object, defining the following parameter-varying matrix function:

$$(A \diamond p) = A_0 + \sum_{i=1}^{n_\alpha} A_i (\alpha_i \diamond p), \quad (1)$$

where $p : \mathbb{T} \rightarrow \mathbb{P} \subseteq \mathbb{R}^{n_p}$ is the scheduling variable, $A_i \in \mathbb{R}^{k \times l}$ are the parameters, and $\alpha_i \in \mathcal{R}$ are scheduling-dependent coefficient functions. For the CT case, $\mathbb{T} = \mathbb{R}$ and, for the DT case, $\mathbb{T} = \mathbb{Z}$. \mathbb{P} is a compact subset of \mathbb{R}^{n_p} . The set \mathcal{R} is defined as the set of real-analytic functions of the form $f : \mathbb{R}^{n_p} \times \dots \times \mathbb{R}^{n_p} \rightarrow \mathbb{R}$. Furthermore, in the CT case, for a $p \in \mathbb{P}$, we define the following notation: if $f \in \mathcal{R}$, then $(f \diamond p) = f(p, \frac{d}{dt}p, \frac{d^2}{dt^2}p, \dots)$ where $\frac{d}{dt}p$ is the time derivative of the scheduling signal. Similarly, in the DT case, for a $p \in \mathbb{P}$, we define: if $f \in \mathcal{R}$, then $(f \diamond p) = f(p, q^1p, q^{-1}p, q^2p, \dots)$ where q is the time-shift operator, i.e., $qp_t = p_{t+1}$. Hence, the set \mathcal{R} allows to include various types of non-linearities and time-varying effects into a representation. For example, the affine basis functions $\alpha_i \in \mathcal{R}$ can represent polynomials, cosines, exponentials, logarithms, and rational functions in $\frac{d^i}{dt^i}p$ or $p_{t \pm i}$ with $i \geq 0$. For the full mathematical treatment of \mathcal{R} see Tóth (2010, Chapter 3). Moreover, for notational simplicity, we denote by $\mathcal{R}^{k \times l}$ the set of all functions of the form (1).

In LPVCORE, the `timemap` object is introduced to simplify the handling of dynamic dependency on p , by introducing the *extended scheduling signal* ρ that includes all the shifted versions (DT) or time derivatives (CT) of p that arise from the use of the \diamond operator in (1). For example, to represent dynamic dependency on both p_t and p_{t-1} , the user can generate the extended scheduling signal using `timemap([0, -1], 'dt')`. The first argument represents the time shifts or derivatives and the second argument denotes the time domain ('dt' and 'ct' for DT and CT, respectively). Then, to model a parameter-varying matrix function in LPVCORE, two more ingredients are required: (1) the coefficients $\{A_i\}_{i=1}^{n_\alpha}$,

and (2) the basis functions $\{\alpha_i\}_{i=1}^{n_\alpha}$. To support simple use of parameter-varying matrix functions for casual users, but also support the more advanced needs of researchers, a flexible `pmatrx` object is defined, which represents a parameter-varying matrix function based on the three ingredients described above. The first argument of `pmatrx` specifies the coefficients, followed by name-value pairs that describe the basis functions and the extended scheduling signal. To demonstrate the degree of flexibility in specifying parameter-varying matrix functions, the use of `pmatrx` is shown in a few examples:

Example 1. Consider the following affine parameter-varying matrix function:

$$(A \diamond p)_t = A_0 + A_1 p_t + A_2 p_{t-1}. \quad (2)$$

The detailed way to represent this function in LPVCORE is to first construct a `timemap` object to extend the scheduling signal with the time-shifted version, and then use this object to create a `pmatrx`, i.e.:

```
rho = timemap([0, -1], 'dt')
A = pmatrx(cat(3, A0, A1, A2), ...
    'BasisType', 'affine', ...
    'BasisParametrization', {0, 1, 2}, ...
    'SchedulingTimeMap', rho)
```

Note that the parameter '`BasisParametrization`' is a cell array of indices of the extended scheduling signal, with 0 representing the constant factor 1. Indices 1 and 2 refer to p_t and p_{t-1} , respectively. As an alternative to `pmatrx`, one can simply use `preal` and `assemble` (2) in one line by algebraic operations (see Example 4).

Example 2. Consider the following affine parameter-varying matrix function with two scheduling signals:

$$(A \diamond (p, q))_t = p_{t-1} + 2q_{t-1} + 3p_t + 4q_t \quad (3)$$

Additional scheduling signals can be specified in the '`Name`' option of `timemap`. Furthermore, affine basis type is the default and need not be specified when creating the `pmatrx`:

```
rho = timemap([-1, 0], 'dt', ...
    'Name', {'p', 'q'})
A = pmatrx(cat(3, 1, 2, 3, 4), ...
    'BasisParametrization', ...
    {1, 2, 3, 4}, ...
    'SchedulingTimeMap', rho)
```

Example 3. Consider the following polynomial parameter-varying matrix function:

$$(A \diamond p)_t = p_t^2 + 2p_t p_{t-2} + p_{t-2}^2. \quad (4)$$

Setting '`BasisType`' to '`poly`' specifies polynomial dependence, for which the associated parametrization is a vector with the degree of each term of the extended scheduling signal, i.e.:

```
rho = timemap([-2, 0], 'dt')
A = pmatrx(cat(3, 1, 2, 1), ...
    'BasisType', 'poly', ...
    'BasisParametrization', ...
    {[2, 0], [1, 1], [0, 2]}, ...
    'SchedulingTimeMap', rho)
```

Custom basis functions are supported, too. For detailed examples see Den Boef et al. (2021). In LPVCORE, many key operations available in MATLAB for constant matrices, such as taking the sum or product, are extended for

Table 1. Operators for `pmatrix`.

Function	Description
<code>P1 + P2</code>	Addition
<code>P1 - P2</code>	Subtraction
<code>P1 .* P2</code>	Element-wise multiplication
<code>P1 * P2</code>	Matrix multiplication
<code>[P1, P2]</code>	Horizontal concatenation
<code>[P1; P2]</code>	Vertical concatenation
<code>P.^2</code>	Matrix power
<code>P.^2</code>	Element-wise power
<code>P(:)</code>	Vectorization
<code>p = P(i, j)</code>	Subscripted reference
<code>P(i, j) = p</code>	Subscripted assignment
<code>kron(P1, P2)</code>	Kronecker product
<code>P'</code>	Conjugate transposition
<code>P.'</code>	Transposition
<code>diag([p1, p2])</code>	Create diagonal matrix
<code>diag(P)</code>	Get diagonal elements
<code>sum(P)</code>	Sum along columns
<code>pshift(P, k)</code>	Shift k time-steps (DT only)
<code>pdiff(P, k)</code>	Differentiate k times (CT only)

`pmatrix` objects, providing an intuitive manner to manipulate these objects. The `pmatrix` object automatically merges different scheduling signals when these operations are used (see Table 1 for an overview).

For convenience, the function `preal` is also available to generate a scalar coefficient. Combined with `pdiff` and `pshift` to differentiate or shift the scheduling in CT or DT respectively, it can be used as an alternative syntax for constructing parameter-varying matrix functions, e.g.:

Example 4. The parameter-varying matrix function (2) can be constructed by a combination of `preal` and algebraic operations in the following way:

```
p = preal('p', 'dt')
A = A0 + A1 * p + A2 * pshift(p, -1)
```

As highlighted by the above examples, an important strength of LPV`CORE` is the implementation of the parameter-varying matrix functions and, as shown in the next sections, the ability to model, analyze, identify, and control LPV representations in a global form. Local representational forms can be obtained via conversion functions, which will be introduced for compatibility with LPV`Tools` in the future.

2.2 LPV-IO representation

With the parameter-varying matrix function defined, various forms of LPV representations of a system at hand can be formulated. A commonly used representation form is the *input-output* (IO) representation, given by the following equation:

$$(\mathcal{A}(\xi) \diamond p)_t y_t = (\mathcal{B}(\xi) \diamond p)_t u_t, \quad (5)$$

where $y : \mathbb{T} \rightarrow \mathbb{Y} = \mathbb{R}^{n_y}$ is the measured output signal and $u : \mathbb{T} \rightarrow \mathbb{U} = \mathbb{R}^{n_u}$ denotes the input signal. For the CT case, $\xi = \frac{d}{dt}$, i.e., $\xi y_t = \frac{d}{dt} y_t$ denotes the time derivative of the output; and for the DT case, $\xi = q^{-1}$ is the backward time-shift operator. The scheduling-dependent polynomials $\mathcal{A}(\xi)$, $\mathcal{B}(\xi)$ are polynomials in the indeterminate ξ with scheduling-dependent coefficients:

$$\mathcal{A}(\xi) \diamond p = I_{n_y} + \sum_{i=1}^{n_a} (A_i \diamond p) \xi^i, \quad (6a)$$

$$\mathcal{B}(\xi) \diamond p = \sum_{j=0}^{n_b} (B_j \diamond p) \xi^j, \quad (6b)$$

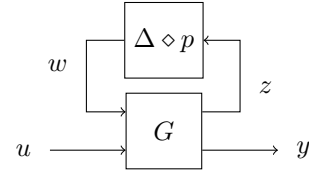


Fig. 1. Schematic representation of an LPV-LFR model.

with finite polynomial orders $n_a \geq 0$ and $n_b \geq 0$, respectively, and $A_i \in \mathcal{R}^{n_y \times n_y}$ and $B_j \in \mathcal{R}^{n_y \times n_u}$ are scheduling-dependent matrix coefficient functions that are affine in the basis functions $\{\alpha_{i,l}\}_{i=1,l=1}^{n_a, n_\alpha}$, $\{\beta_{j,l}\}_{j=0,l=1}^{n_b, n_\beta}$, respectively, similarly to (1). The LPV-IO representation (5) can be created by using the `lpvio` command.

2.3 LPV-SS representation

The system can also be represented by a first-order differential or difference equation, i.e., state-space representation:

$$\xi x_t = (A \diamond p)_t x_t + (B \diamond p)_t u_t, \quad (7a)$$

$$y_t = (C \diamond p)_t x_t + (D \diamond p)_t u_t. \quad (7b)$$

where $x : \mathbb{T} \rightarrow \mathbb{X} = \mathbb{R}^{n_x}$ is the so-called state (latent) variable and $A \in \mathcal{R}^{n_x \times n_x}$, $B \in \mathcal{R}^{n_x \times n_u}$, $C \in \mathcal{R}^{n_y \times n_x}$, and $D \in \mathcal{R}^{n_y \times n_u}$ are matrix coefficient functions. For the CT case, $\xi = \frac{d}{dt}$ and, for the DT case, $\xi = q$ is the forward time-shift operator. The LPV-SS representation (7) can be created by using the `lpvss` command.

2.4 LPV-LFR representation

An LPV-LFR representation can be used if the dynamics of the system can be decomposed as the linear fractional transform (LFT) of an LTI system G with a parameter-varying matrix function Δ , as shown in Figure 1. The input-output behavior of the system can then be described by the following equations:

$$\xi x_t = A x_t + B_w w_t + B_u u_t, \quad (8a)$$

$$z_t = C_z x_t + D_{zw} w_t + D_{zu} u_t, \quad (8b)$$

$$y_t = C_y x_t + D_{yw} w_t + D_{yu} u_t, \quad (8c)$$

$$w_t = (\Delta \diamond p)_t z_t. \quad (8d)$$

The LPV-LFR representation can be created by using the `lpvlfr` command. The LPV-LFR representation is a generalization of the LPV-SS representation, as can be seen by setting $D_{zw} = 0$. Therefore, LPV`CORE` internally treats LPV-SS models as LPV-LFR models, and every operation defined on LPV-LFR models, such as multiplication, concatenation and feedback interconnection, can directly be used on LPV-SS models. For a complete list of supported operations, see www.lpvcore.net.

3. IDENTIFICATION OF LPV MODELS

LPV`CORE` includes several system identification methods to estimate DT LPV-IO and LPV-SS models in a global setting, and one method to estimate LPV-LFR models in a local setting. For the global approach, the estimation of the model parameters is performed by minimizing the prediction-error. Hence, in Section 3.1, the *prediction-error minimization* (PEM) setting is briefly introduced. Next, the LPV-IO model set and the corresponding identification methods are discussed in Section 3.2. In Section 3.3, the model set and PEM for LPV-SS models are discussed. For the local approach, to identify LPV-LFR models, the estimation is based on minimizing the worst-case local \mathcal{H}_∞ -error between the model and a given set of LTI models at the measured operating points. This approach is discussed in Section 3.4.

3.1 The PEM identification setting

This section provides a short overview of PEM, for a more detailed description, see e.g. Tóth et al. (2012). In PEM, model estimates are obtained by minimizing their associated prediction-error with respect to a loss function where the (unknown) model parameters are the optimization variables. Hence, the concept is to obtain a parameter estimate $\hat{\theta}_N$ of the parameter θ_o describing the data-generating system by minimizing the least-squares criterion, i.e., by

$$\hat{\theta}_N = \underset{\theta \in \mathbb{R}^{n_\theta}}{\operatorname{argmin}} V(\mathcal{D}_N, \theta), \quad (9)$$

with

$$V(\mathcal{D}_N, \theta) = \frac{1}{N} \sum_{t=1}^N \|\varepsilon_{t|\theta}\|_2^2 = \frac{1}{N} \|\varepsilon_\theta\|_{\ell_2}^2,$$

where $\mathcal{D}_N = \{u_t, p_t, y_t\}_{t=1}^N$ is the observed data sequence of the system and the one-step-ahead prediction-error is $\varepsilon_{t|\theta} = y_t - \hat{y}_{t|\theta, t-1}$. In Sections 3.2 and 3.3, the one-step-ahead prediction-error $\varepsilon_{t|\theta}$ associated with the IO and SS model sets will be introduced.

In system identification, the data \mathcal{D}_N is assumed to be corrupted with noise. Hence, it is essential to extend the system representations with a noise process. These extensions are discussed in the next sections.

3.2 LPV-IO model identification

Model set: The IO representation of the data-generating system is captured by the parametrized model \mathcal{M}_θ :

$$(\mathcal{F}(q^{-1}, \theta) \diamond p)_t \check{y}_t = (\mathcal{B}(q^{-1}, \theta) \diamond p)_t q^{-\tau_a} u_t, \quad (10a)$$

$$(\mathcal{D}(q^{-1}, \theta) \diamond p)_t v_t = (\mathcal{C}(q^{-1}, \theta) \diamond p)_t \varepsilon_{t|\theta}, \quad (10b)$$

$$(\mathcal{A}(q^{-1}, \theta) \diamond p)_t y_t = \check{y}_t + v_t, \quad (10c)$$

where $\mathcal{A}(\cdot), \dots, \mathcal{D}(\cdot), \mathcal{F}(\cdot)$ are polynomials in q^{-1} with scheduling-dependent coefficients, which are parametrized in θ using a-priori given basis functions.

The model (10) with $\mathcal{A} \triangleq I_{n_y}$ is also known as the LPV *Box-Jenkins* (BJ) model. By considering $\mathcal{C} \triangleq \mathcal{D} \triangleq \mathcal{F} \triangleq I_{n_y}$ in (10), the LPV version of the so-called *auto regressive with exogenous input* (ARX) model structure is obtained. Similarly, the *auto regressive moving average with exogenous input* (ARMAX) model is found by considering $\mathcal{D} \triangleq \mathcal{F} \triangleq I_{n_y}$, and the *output-error* (OE) model by $\mathcal{A} \triangleq \mathcal{C} \triangleq \mathcal{D} \triangleq I_{n_y}$.

Each of the above model structures is available in LPV-CORE using the `lpvidpoly` object, which serves as an LPV extension to the `idpoly` object included in the System Identification Toolbox. A powerful feature of `lpvidpoly` is the ability to exploit known sparsity of θ , as demonstrated in the following example.

Example 5. (LPV-ARX model). Consider an LPV-ARX model of the form $A(p, \theta)q^{-1}y + y = B(\theta)u$:

$$A(p, \theta) = \begin{pmatrix} \theta_1 & \theta_2 \\ 0 & \theta_3 \end{pmatrix} p, \quad B(\theta) = \begin{pmatrix} \theta_4 \\ 0 \end{pmatrix}. \quad (11)$$

This model structure can be created as follows:

```
p = preal('p', 'dt');
A = [1, 1; 0, 1] * p;
B = [1; 0];
sys = lpvidpoly({eye(2), A}, B);
```

The object `sys` is then used as a *template model structure* in subsequent LPV-IO identification methods, which use it to extract the scheduling dependence and, in some methods, the initial parameter values. Zero elements are, by default, excluded from the set of identifiable parameters, providing an intuitive method for semi-grey box identification of LPV-IO representations. This behaviour can be disabled: see documentation at `www.lpvcore.net`.

Identification schemes: The toolbox contains four time-domain methods for identifying IO model structures (10): 1) linear regression, 2) pseudo-linear regression, 3) gradient-based search, and 4) instrumental variable method. These methods are based on Butcher et al. (2008); Tóth et al. (2012); Cox (2018). Linear regression is applicable for the ARX model set and the model estimate is obtained by executing `lpvarx`. For the `lpvarx` method, regularization is included by solving a Tikhonov regression problem, where various weighting matrices can automatically be tuned using generalised cross-validation (Golub and Van Loan, 2013, Section 6.1.4) or marginal likelihood optimization (Rasmussen and Williams, 2006). These options can be selected using the option set `lpvarxOptions`. The pseudo-linear regression algorithm is executed for the ARMAX, OE, and BJ model sets by calling `lpvarmax`, `lpvoe`, and `lpvbj`, respectively. The gradient-based search is applicable for all afore-mentioned model types and is executed by calling `lpvpolyest`. The instrumental variable method is applied by the command `lpviv`. At the moment, this function is available for model structures with a single input and a single output (SISO), and will be extended to the general case in a later release.

3.3 LPV-SS model identification

Model set: The model set for LPV-SS representations supports *innovation noise*, i.e.:

$$x_{t+1} = (A(\theta) \diamond p)_t x_t + (B(\theta) \diamond p)_t u_t + (K(\theta) \diamond p)_t e_t^o, \quad (12a)$$

$$y_t = (C(\theta) \diamond p)_t x_t + (D(\theta) \diamond p)_t u_t + e_t^o, \quad (12b)$$

where the matrix coefficient functions A, \dots, D, K are affine in the set of a-priori specified basis functions $\{\psi_i \diamond p\}_{i=1}^{n_\psi}$ similar to (1). The sequence e_t^o is the sample path realization of the zero-mean stationary process:

$$e_t^o \sim \mathcal{N}(0, \Xi), \quad (13)$$

with Ξ a positive-definite, symmetric real matrix.

Identification schemes: To identify the state-space model with parametrization (12), the toolbox includes an iterative, *gradient-based* (GB) optimization technique which obtains an estimate of the innovation model. In a future release, a second iterative optimization technique which is based on the *expectation-maximization* (EM) (Wills and Ninness (2011)) method will be included. The `lpvssset` function estimates an LPV-SS model with the GB algorithm. Since this method is based on a non-linear optimization, an initial estimate must be provided by the user. This initial estimate can be obtained by identifying an IO model and transforming it into the state-space form (available in a future release of LPV-CORE) or by subspace identification which is the default option (Cox and Tóth, 2021; van Wingerden and Verhaegen, 2009). The GB method includes several steps to efficiently estimate the parametric LPV-SS model with innovation noise (12). Among others, the implementation ensures that the non-linear optimization does not wander among parametrizations of SS mod-

els with equivalent input-scheduling-output behavior. The method is explained in detail in Cox (2018, Sec. 8.2).

3.4 LPV-LFR model identification

The local identification scheme for LPV-LFR representations included in LPVCORE is based on Vizer et al. (2013). Due to a lack of space, a detailed explanation can be found in Den Boef et al. (2021).

3.5 Using identified models for control synthesis

LPVCORE includes several control synthesis algorithms. LPV-SS models identified using the previously introduced schemes can directly be used for control by synthesizing a stabilizing controller based on given performance specifications in a variety of settings. For a complete overview of the available algorithms, consult the documentation which is included in LPVCORE.

4. SIMULATION STUDY

4.1 Non-linear data-generating system

To illustrate the effectiveness of LPVCORE, several of the implemented identification methods are applied to estimate a model of a DC motor with an unbalanced disc. The system consists of a vertically mounted disc with a lumped mass m at distance l from the center, actuated by an input voltage u . Due to the gravitational force exerted on the lumped mass, the system exhibits non-linear dynamics, which can be represented in the following differential equation:

$$\ddot{\theta} = -\frac{1}{\tau}\dot{\theta} + \frac{K_m}{\tau}u - \frac{mgl}{J}\sin(\theta) \quad (14)$$

where θ is the disc angular position, τ is the lumped back EMF constant, K_m is the motor constant, J is the complete disc inertia and g is the acceleration due to gravity. The goal of this section is to identify the dynamics of (14) based on simulation data. The simulated output is the angular position θ corrupted with additive white Gaussian noise ε of a specified *signal-to-noise ratio* SNR.

4.2 LPV identification

An LPV embedding of the non-linear dynamics in (14) can be obtained by introducing the scheduling signal p :

$$p := \text{sinc}(\theta) = \frac{\sin(\theta)}{\theta}. \quad (15)$$

Substitution of (15) into (14) leads to the following LPV-IO representation of the dynamics of the unbalanced disc:

$$\ddot{\theta} = -\frac{1}{\tau}\dot{\theta} - \frac{mgl}{J}p\theta + \frac{K_m}{\tau}u \quad (16)$$

Global identification approaches available in LPVCORE operate in DT. Therefore, the LPV system (16) is discretized using forward Euler, i.e.:

$$\dot{\theta}_k := \dot{\theta}(kT_s) \approx \frac{\theta_{k+1} - \theta_k}{T_s}, \quad (17)$$

with $T_s = 0.75$ ms the sampling time. Applying (17) to (16) yields the following DT approximation:

$$\theta_k + A_1\theta_{k-1} + (A_2 \diamond p)_k \theta_{k-2} = \frac{K_m T_s^2}{\tau} u_{k-2}, \quad (18)$$

with

$$A_1 = \frac{T_s}{\tau} - 2, \quad (A_2 \diamond p)_k = 1 - \frac{T_s}{\tau} + \frac{mglT_s^2}{J}p_{k-2}. \quad (19)$$

Table 2. Parameters of the unbalanced disc.

Parameter	Symbol	Value	Unit
Sampling time	T_s	75	ms
Motor constant	K_m	15.3145	-
Disc inertia	J	$2.2 \cdot 10^{-4}$	Nm ²
Lumped mass	m	0.07	kg
Distance mass to center	l	0.42	mm
Gravitational acceleration	g	9.8	m · s ⁻²
Back EMF constant	τ	0.5971	-

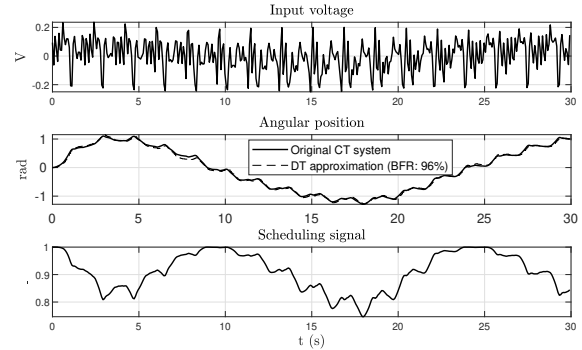


Fig. 2. Estimation dataset from the unbalanced disc system (14), compared to the output of its DT approximation (18).

Figure 2 shows the quality of the approximation in terms of the *Best Fit Rate* (BFR) for a simulation. The BFR, expressed as a percentage, is defined as follows:

$$\text{BFR} := 100\% \cdot \max\left(0, 1 - \frac{\|y - \hat{y}\|_2}{\|y - \text{mean}(y)\|}\right). \quad (20)$$

The values of all numerical parameters used in this example are listed in Table 2.

For the identification procedure, five datasets of 400 samples each are generated: four for estimation with varying levels of additive White Gaussian output noise and one for validation (without noise). These datasets are generated by simulating (14) for a random-phase multi-sine input u containing 10 frequencies equally spaced in a passband from 0 to 0.75 times the Nyquist frequency. The signal is scaled to have a maximum amplitude of 0.25 V. Zero-order-hold is used to interpolate between the samples. The SNRs for the estimation datasets are 0, 10, 20 and 40 dB. A noise-free example of the estimation dataset and a comparison with the DT LPV-IO approximation (18) can be seen in Figure 2. Four LPV-IO model structures are identified: ARX, ARMAX, OE and BJ. It is assumed that the true model structure (18) is not known exactly and the following more generic template structure is used instead (see (10)):

- $(\mathcal{A}(q^{-1}) \diamond p)_k = 1 + (1 + p_{k-1})q^{-1} + (1 + p_{k-2})q^{-2}$
- $(\mathcal{B}(q^{-1}) \diamond p)_k = (\mathcal{F}(q^{-1}) \diamond p)_k = (\mathcal{A}(q^{-1}) \diamond p)_k$
- $\mathcal{C}q^{-1} = \mathcal{D}(q^{-1}) = 1 + q^{-1}$

The models are identified using the estimation dataset in the following order: (1) The LPV-ARX model is identified by calling `lpvarx`. (2) The LPV-ARMAX and LPV-OE models are identified by calling `lpvpolyest`. The LPV-ARX model is used as initialization. (3) The LPV-BJ model is identified by calling `lpvpolyest`. The LPV-OE model is used as initialization. For each method, the default options were used, except the number of gradient descent iterations performed by `lpvpolyest`: it

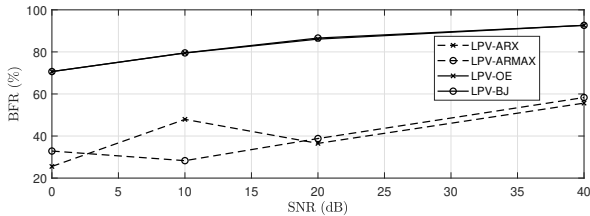


Fig. 3. Validation BFR of the model estimates for different output SNRs of the estimation dataset.

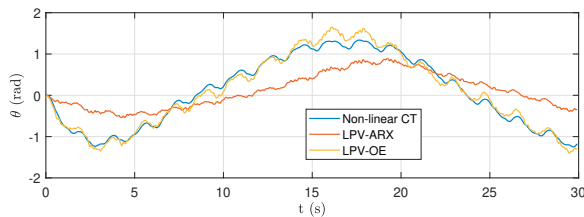


Fig. 4. Comparison between the output of the validation dataset and the DT LPV-ARX and LPV-OE models estimated with SNR = 20 dB.

was set to 400. The quality of the resulting models is evaluated using the validation dataset by calculating the BFR between the output of the non-linear CT system (14) and the output of the DT model estimate.

4.3 Results

The achieved BFR of the model estimates can be seen in Figure 3. For each SNR, the LPV-OE and LPV-BJ models have a substantially higher BFR than the LPV-ARX and LPV-ARMAX models, which is in line with the expectation since output noise was used. The low BFR of the LPV-ARX and LPV-ARMAX models, even for high SNRs, is likely the result of a high parameter sensitivity due to the proximity of the poles of the frozen dynamics of (16) to the unit circle. For example, at $p = 0$ ($\theta = \pm\pi$ rad), the magnitude of the two poles of the resulting LTI dynamics are 1 and 0.92. When $p = 1$ ($\theta = 0$), both poles have a magnitude of 0.96. The output of the LPV-ARX and LPV-OE models estimated with SNR = 20 dB are compared to the original non-linear CT dynamics (14) in Figure 4.

5. CONCLUSION

This paper introduces a software package for MATLAB called the LPVCORE toolbox to model, simulate, and estimate linear parameter-varying input-output, state-space and linear fractional representations in a global setting. To support the global setting, the package defines parameter-varying matrix functions, which is the key difference to other software suites that use a grid or linear fractional based representation. This paper provides an overview of the core modelling tools combined with the simulation and identification methods. The software and examples are freely available on www.lpvcore.net.

The initial toolbox forms a solid basis for additional algorithms. In the near future, algorithms for LPV realization and a wide range of subspace identification methods will be implemented.

REFERENCES

Briat, C. (2015). *Linear Parameter-Varying and Time-Delay Systems: Analysis, Observation, Filtering & Control*. Springer.

- Butcher, M., Karimi, A., and Longchamp, R. (2008). On the consistency of certain identification methods for linear parameter varying systems. In *Proc. of the 17th IFAC World Congress*, 4018–4023. Seoul, Korea.
- Cox, P.B. (2018). *Towards Efficient Identification of Linear Parameter-Varying State-Space Models*. Phd thesis, Eindhoven University of Technology.
- Cox, P.B. and Tóth, R. (2021). Linear parameter-varying subspace identification: A unified framework. *Automatica*, 123, 109296.
- Den Boef, P., Tóth, R., and Cox, P.B. (2021). LPVcore: MATLAB toolbox for LPV modelling, identification and control. *Submitted to arXiv*.
- Golub, G.H. and Van Loan, C.F. (2013). *Matrix Computations*. The Johns Hopkins Uni. Press, 4th edition.
- Hjartarson, A., Seiler, P.J., and Packard, A. (2015). LPV-Tools: A toolbox for modeling, analysis, and synthesis of parameter varying control systems. In *Proc. of the 1st IFAC Workshop on Linear Parameter Varying Systems*, 139–145. Grenoble, France.
- Hoffmann, C. and Werner, H. (2015). A survey of linear parameter-varying control applications validated by experiments or high-fidelity simulations. *IEEE Trans. on Control Systems Technology*, 23(2), 416–433.
- Lopes dos Santos, P., Azevedo-Perdicoulis, T.P., Novara, C., Ramos, J.A., and Rivera, D.E. (eds.) (2011). *Linear Parameter-Varying System Identification: New Developments and Trends*. World Scientific.
- MATLAB (2014). *version 8.4.0 (R2014b)*. The MathWorks Inc., Natick, Massachusetts.
- Mohammadpour, J. and Scherer, C. (eds.) (2012). *Control of Linear Parameter Varying Systems with Applications*. Springer.
- Rabiei, M., Abbas, H.S., and Hassan, M.M. (2015). LPVOID: An LPV input/output systems identification toolbox using MATLAB. URL sites.google.com/site/mustafarabiei/.
- Rasmussen, C.E. and Williams, C.K.I. (2006). *Gaussian Processes for Machine Learning*. the MIT Press.
- Sename, O., Gáspár, P., and Bokor, J. (eds.) (2012). *Robust Control and Linear Parameter Varying Approaches: Application to Vehicle Dynamics*. Springer.
- Tóth, R. (2010). *Modeling and Identification of Linear Parameter-Varying Systems*. Springer.
- Tóth, R., Heuberger, P.S.C., and Van den Hof, P.M.J. (2012). Prediction error identification of LPV systems: present and beyond. In J. Mohammadpour and C.W. Scherer (eds.), *Control of Linear Parameter Varying Systems with Applications*, 27–60. Springer, Heidelberg.
- van Wingerden, J.W. and Verhaegen, M. (2009). Subspace identification of bilinear and LPV systems for open- and closed-loop data. *Automatica*, 45(2), 372–381.
- Vizer, D., Mercere, G., Prot, O., Laroche, E., and Lovera, M. (2013). Linear fractional lpv model identification from local experiments: an H_∞ -based optimization technique. In *52nd IEEE Conference on Decision and Control*, 4559–4564. IEEE.
- Wills, A. and Ninness, B. (2011). System identification of linear parameter varying state-space models. In P.L. dos Santos, C. Novara, D. Rivera, J. Ramos, and T. Perdicoulis (eds.), *Linear Parameter-Varying System Identification: New Developments and Trends*, 295–313. World Scientific Publishing, Singapore.