Direct identification of continuous-time LPV state-space models via an integral architecture [⋆]

Manas Mejari [a], Bojan Mavkov [b], Marco Forgione [a], Dario Piga [a]

[a] *Dalle Molle Institute for Artificial Intelligence, IDSIA USI-SUPSI, Via la Santa 1, CH-6962 Lugano-Viganello, Switzerland.*

[b] *Université Grenoble Alpes, CNRS, GIPSA-lab, F-38000, Grenoble.*

Abstract

In this paper, we present a block-structured architecture for direct identification of continuous-time *Linear Parameter-Varying* (LPV) state-space models. The proposed architecture consists of an LPV model followed by an *integral* block. This structure is used to approximate the continuous-time LPV system dynamics. The unknown LPV model matrices are estimated along with the state sequence by minimizing a properly constructed dual-objective criterion. A coordinate-descent algorithm is employed to optimize the desired objective, which alternates between computing the unknown LPV matrices and estimating the state sequence. Thanks to the linear parametric structure induced by the LPV model, the optimization variables within each coordinate-descent step can be updated analytically via ordinary least squares. Furthermore, in order to handle large-size datasets, we discuss how to perform optimization based on short-size subsequences. The effectiveness of the proposed methodology is demonstrated via an academic example and two case studies. The first case study consists of identifying an LPV model describing the behavior of an electronic bandpass filter from benchmark experimental data. The second case study involves identification of the plasma safety factor from a tokamak plasma simulator.

Key words: linear parameter-varying models, continuous-time identification, state-space identification, tokamak plasma.

1 Introduction

Direct identification of *continuous-time* (CT) dynamical systems from sampled data has gained significant importance in the past few years. This is due to the fact that direct CT identification has multiple advantages over the discrete-time case as discussed in [8] and [6]. Essentially, the majority of physical systems are naturally modelled in continuous-time, and thus, the estimated parameters of CT models usually have a physical interpretation. Direct CT identification methods can also deal with non-uniformly sampled data, while discrete-time models implicitly rely on a fixed sampling time. Moreover, CT identification methods are generally more robust to numerical issues that may arise when using discrete-time methods in the case of high-frequency sampled data. Successful applications and complete reviews of direct CT identification methods can be found in the book [7], in the contributions [6, 12, 21, 22] and the references therein. However, most of the available algorithms for CT identification are restricted to the *Linear Time-Invariant* (LTI) modelling framework.

Concerning the identification of *Linear Parameter-Varying* (LPV) models, most of the approaches have been developed for the discrete-time case, either in input-output [1, 23, 20, 18, 13] or in state-space representation [3, 19, 25]. Only few contributions have addressed CT identification of LPV state-space models [9, 2, 10]. In [9], CT identification of gray-box quasi-LPV models with an observer-based identification scheme is proposed. In [2], a local approach in the framework of subspace identification is developed. A frequency domain approach is proposed in [10] under the assumption of a periodic variation of the scheduling signal.

In this paper, we address the problem of direct identification of continuous-time *multi-input multi-output* (MIMO) LPV systems through an *integral* architecture. This architecture consists of an LPV model followed by an integral block, which is used to approximate the continuous state dynamics of an LPV system. The over-

[⋆] Corresponding author: M. Mejari.

Email addresses: `manas.mejari@supsi.ch` (Manas Mejari), `bojan.mavkov@univ-grenoble-alpes.fr` (Bojan Mavkov), `marco.forgione@supsi.ch` (Marco Forgione), `dario.piga@supsi.ch` (Dario Piga).

all methodology is based on the concept recently introduced by some of the authors in [15] for identification of non-linear systems through neural networks. In the current contribution, we adapt and specialize that methodology for LPV system identification. The main advantage of the specialized approach w.r.t. the generic one in [15] is the higher computational efficiency. Indeed, the most intensive steps of the training procedure presented in [15] are accelerated in this paper – leveraging particular properties of the LPV model structure – through a coordinate-descent algorithm with a closed-form formula for each update step.

Another possible approach to estimate continuous-time LPV models is to *simulate* the model using a numerical ODE solution scheme, and then minimize the *simulation error* w.r.t. to the unknown parameters. However, the resulting simulation error minimization problem is non-convex and highly nonlinear, and obtaining its solution can be computationally and numerically hard. With the integral architecture employed in this paper, we circumvent the need to run time simulations. Furthermore, we split the original optimization problem into two subproblems which can be solved *analytically* via ordinary least squares. In particular, a properly constructed cost function is minimized through a *coordinate-descent* algorithm, by successively estimating the state variables and the mapping functions, one at a time. At each step of the coordinate-descent algorithm, a least-square problem is formulated and the solution is computed analytically. Nonetheless, for large datasets, the computation of state sequence within the coordinate-descent algorithm may still be excessively time consuming or even intractable. To this end, we also present an approach based on short-size subsequences to estimate the state sequence in an efficient manner in the case of long training sequences.

The performance of the proposed identification algorithm is validated on several examples. First, an academic example is used to demonstrate the efficiency of the algorithm and by assessing its robustness w.r.t. different initial guesses of the state variables and different lengths of the training subsequences. The computational speed improvement resulting from the approach based on subsequences is demonstrated. In addition, the performance of the algorithm is tested on two more complex identification benchmarks. The first benchmark consists of experimental measurements from an electronic band-pass filter [11]. An LPV model describing the behaviour of the filter is identified and validated. The second benchmark consists of evolution of the plasma safety factor obtained from the tokamak plasma simulator RAPTOR [4]. The proposed algorithm is employed to identify the plasma safety factor via an LPV representation after an appropriate choice of the scheduling parameters.

The paper is organized as follows. The identification problem is formalized in Section 2. The description of the integral architecture and details for the implementation of the proposed estimation algorithm are provided in Section 3. Examples are reported in Section 4. Preliminary ideas of this work have been already presented in [17]. However, [17] includes neither the optimization based on short-size subsequences nor the two case studies discussed in Section 4.

2 Problem formulation

We consider a data-generating system $\mathcal{S}$ governed by the following continuous-time MIMO state-space LPV representation:

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathcal{A}(\mathbf{p}(t))\mathbf{x}(t) + \mathcal{B}(\mathbf{p}(t))\mathbf{u}(t), && \text{(1a)}\\ \mathbf{x}(0) &= \mathbf{x}_0, && \text{(1b)}\\ \mathbf{y}^{\mathrm{o}}(t) &= \mathcal{C}(\mathbf{p}(t))\mathbf{x}(t) + \mathcal{D}(\mathbf{p}(t))\mathbf{u}(t), && \text{(1c)} \end{aligned}$$

where $\mathbf{x}(t) \in \mathbb{R}^{n_x}$ and $\dot{\mathbf{x}}(t) \in \mathbb{R}^{n_x}$ are the state vector and its time derivative, respectively; $\mathbf{x}_0 \in \mathbb{R}^{n_x}$ is the initial state condition; $\mathbf{u}(t) \in \mathbb{R}^{n_u}$ is the system input; $\mathbf{p}(t) \in \mathbb{R}^{n_p}$ is the scheduling signal and $\mathbf{y}^{\mathrm{o}}(t) \in \mathbb{R}^{n_y}$ is the *noise-free* system output at time $t \in \mathbb{R}$. The matrix functions $\mathcal{A}(\cdot), \mathcal{B}(\cdot), \mathcal{C}(\cdot), \mathcal{D}(\cdot)$ are time-varying *affine* functions of the scheduling signal $\mathbf{p}(t)$ defined as:

$$\mathcal{A}(\mathbf{p}(t)) = A_0 + \sum_{i=1}^{n_p} A_i \mathbf{p}_i(t), \quad \mathcal{B}(\mathbf{p}(t)) = B_0 + \sum_{i=1}^{n_p} B_i \mathbf{p}_i(t),$$
$$\mathcal{C}(\mathbf{p}(t)) = C_0 + \sum_{i=1}^{n_p} C_i \mathbf{p}_i(t), \quad \mathcal{D}(\mathbf{p}(t)) = D_0 + \sum_{i=1}^{n_p} D_i \mathbf{p}_i(t),$$

where $\mathbf{p}_i(t)$ denotes the i-th component of the scheduling vector $\mathbf{p}(t)$ and $\{A_i, B_i, C_i, D_i\}_{i=0}^{n_p}$ are real-valued constant matrices of appropriate dimensions.

A training dataset $\mathcal{D}$ of length N is gathered from the LPV system $\mathcal{S}$ defined in (1) at time instants $\{t_0 = 0, t_1, \ldots, t_{N-1}\}$. The dataset consists of input, scheduling and *noisy* output samples: $\mathcal{D} = \{\mathbf{u}(t_k), \mathbf{p}(t_k), \mathbf{y}(t_k)\}_{k=0}^{N-1}$. The measured output $\mathbf{y}(t_k)$ is corrupted by a zero-mean noise η, *i.e.*, $\mathbf{y}(t_k) = \mathbf{y}^{\mathrm{o}}(t_k) + \eta(t_k)$.

Given the training dataset $\mathcal{D}$, our goal is to identify a continuous-time LPV state-space affine model, such that the model output matches closely with the measured system output $\mathbf{y}(t)$. To this end, we define an *integral* architecture for the identification of continuous-time LPV models in the following section.

3 Continuous-time identification of LPV state-space models

3.1 Integral architecture

In order to describe the continuous-time state dynamics in (1a), we define an LPV block $\mathcal{M}_x(\hat{\mathbf{x}}, \mathbf{u}, \mathbf{p})$, which is fed by the system input $\mathbf{u}(t)$, scheduling signal $\mathbf{p}(t)$ and (estimated) state $\hat{\mathbf{x}}(t)$ at time t, and returns the estimated state time-derivative $\dot{\hat{\mathbf{x}}}(t)$, *i.e.*,

$$\mathcal{M}_x(\hat{\mathbf{x}}, \mathbf{u}, \mathbf{p}) :$$
$$\dot{\hat{\mathbf{x}}} = \left(\hat{A}_0 + \sum_{i=1}^{n_p} \hat{A}_i \mathbf{p}_i(t)\right) \hat{\mathbf{x}}(t) + \left(\hat{B}_0 + \sum_{i=1}^{n_p} \hat{B}_i \mathbf{p}_i(t)\right) \mathbf{u}(t), \tag{2}$$

where $\hat{A}_i \in \mathbb{R}^{n_x \times n_x}$ and $\hat{B}_i \in \mathbb{R}^{n_x \times n_u}$ (for $i = 0, \ldots, n_p$) are the model matrices to be identified. Similarly, the output equation in (1c) is represented by another LPV block $\mathcal{M}_y(\hat{\mathbf{x}}, \mathbf{u}, \mathbf{p})$, which is fed by the estimated state $\hat{\mathbf{x}}(t)$, input $\mathbf{u}(t)$ and scheduling signal $\mathbf{p}(t)$ and it returns the model output $\hat{\mathbf{y}}(t)$ at time t, *i.e.*,

$$\mathcal{M}_y(\hat{\mathbf{x}}, \mathbf{u}, \mathbf{p}) :$$
$$\hat{\mathbf{y}}(t) = \left(\hat{C}_0 + \sum_{i=1}^{n_p} \hat{C}_i \mathbf{p}_i(t)\right) \hat{\mathbf{x}}(t) + \left(\hat{D}_0 + \sum_{i=1}^{n_p} \hat{D}_i \mathbf{p}_i(t)\right) \mathbf{u}(t), \tag{3}$$

where the matrices $\hat{C}_i \in \mathbb{R}^{n_y \times n_x}$ and $\hat{D}_i \in \mathbb{R}^{n_y \times n_u}$ (for $i = 0, \ldots, n_p$) have to be estimated from data.

To simplify the notations, we introduce the following matrices stacking the model parameters in their columns:

$$\Theta_x = \left[\hat{A}_0 \cdots \hat{A}_{n_p} \ \hat{B}_0 \cdots \hat{B}_{n_p}\right] \in \mathbb{R}^{n_x \times (n_p+1)(n_x+n_u)}$$

and

$$\Theta_y = \left[\hat{C}_0 \cdots \hat{C}_{n_p} \ \hat{D}_0 \cdots \hat{D}_{n_p}\right] \in \mathbb{R}^{n_y \times (n_p+1)(n_x+n_u)}.$$

The resulting continuous-time LPV state-space affine model is then given by:

$$\dot{\hat{\mathbf{x}}}(t) = \mathcal{M}_x(\hat{\mathbf{x}}(t), \mathbf{u}(t), \mathbf{p}(t); \Theta_x), \tag{4a}$$
$$\hat{\mathbf{x}}(0) = \hat{\mathbf{x}}_0, \tag{4b}$$
$$\hat{\mathbf{y}}(t) = \mathcal{M}_y(\hat{\mathbf{x}}(t), \mathbf{u}(t), \mathbf{p}(t); \Theta_y). \tag{4c}$$

We note that, from (2) and (3), the maps $\mathcal{M}_x(\cdot)$ and $\mathcal{M}_y(\cdot)$ are *linear* functions of the parameters Θ_x and Θ_y, respectively given by:

$$\mathcal{M}_x(\hat{\mathbf{x}}(t), \mathbf{u}(t), \mathbf{p}(t); \Theta_x) = \Theta_x \begin{bmatrix} \begin{bmatrix} 1 \\ \mathbf{p}(t) \end{bmatrix} \otimes \hat{\mathbf{x}}(t) \\ \begin{bmatrix} 1 \\ \mathbf{p}(t) \end{bmatrix} \otimes \mathbf{u}(t) \end{bmatrix}, \tag{5a}$$
$$\mathcal{M}_y(\hat{\mathbf{x}}(t), \mathbf{u}(t), \mathbf{p}(t); \Theta_y) = \Theta_y \begin{bmatrix} \begin{bmatrix} 1 \\ \mathbf{p}(t) \end{bmatrix} \otimes \hat{\mathbf{x}}(t) \\ \begin{bmatrix} 1 \\ \mathbf{p}(t) \end{bmatrix} \otimes \mathbf{u}(t) \end{bmatrix}, \tag{5b}$$

where $\otimes$ denotes the Kronecker product between the two vectors.

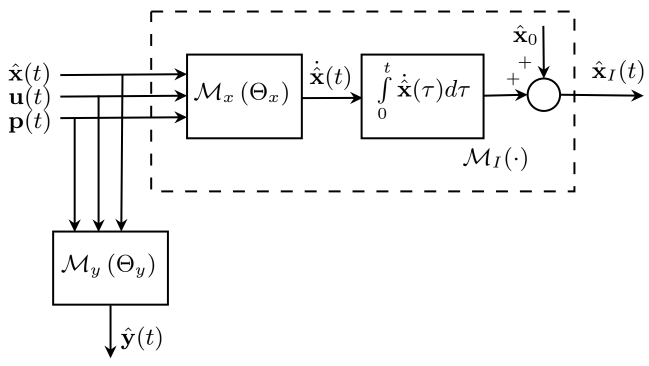

Fig. 1. Integral architecture for continuous-time LPV model identification.

In this paper, we adopt a method introduced in [15], which exploits the *integral form* of the Cauchy problem (4a)-(4b), by defining an *integral* LPV block $\mathcal{M}_I$ as:

$$\hat{\mathbf{x}}_I(t) = \mathcal{M}_I(\hat{\mathbf{x}}(t), \mathbf{u}(t), \mathbf{p}(t); \Theta_x) \tag{6}$$

with

$$\mathcal{M}_I(\hat{\mathbf{x}}(t), \mathbf{u}(t), \mathbf{p}(t); \Theta_x) =$$
$$\hat{\mathbf{x}}(0) + \int_0^t \mathcal{M}_x(\hat{\mathbf{x}}(\tau), \mathbf{u}(\tau), \mathbf{p}(\tau); \Theta_x) d\tau.$$

The block diagram in Fig. 1 is a representation of (6), along with the output equation (4c) producing $\hat{\mathbf{y}}(t)$.

If the state sequence $\hat{\mathbf{x}}(t)$ feeding the LPV model block $\mathcal{M}_I(\cdot)$ is actually generated by the model given in (4), then the state $\hat{\mathbf{x}}_I(t)$ exactly matches with $\hat{\mathbf{x}}(t)$, i.e.,

$$\hat{\mathbf{x}}(t) = \hat{\mathbf{x}}_I(t) \qquad \forall t \in [t_0 \ t_{N-1}]. \tag{7}$$

3.2 Fitting criterion

In the proposed methodology, the LPV model matrices Θ_x, Θ_y and the state signal $\hat{\mathbf{x}}(t), t \in [t_0, \ t_{N-1}]$, are *free* optimization parameters. They are jointly optimized according to a *dual* objective constructed according to the following rationale. First, the estimated model output $\hat{\mathbf{y}}$ should match the output measurements in the training dataset $\mathcal{D}$. This objective is achieved by introducing a *fitting term* in the cost function which penalizes the distance between the model output $\hat{\mathbf{y}}(t_k)$ and the measured output $\mathbf{y}(t_k), \ k = 0, 1, \ldots, N-1$. Second, the state signal $\hat{\mathbf{x}}$ should be compatible with the model dynamics (4). This can be achieved through an additional *regularization term* which penalizes the distance between $\hat{\mathbf{x}}_I(t)$ and $\hat{\mathbf{x}}(t)$, where $\hat{\mathbf{x}}_I$ is defined as in (6).

The following minimization problem is thus formulated:

$$\min_{\hat{\mathbf{x}}(\cdot), \Theta_x, \Theta_y} J(\hat{\mathbf{x}}(\cdot), \Theta_x, \Theta_y), \tag{8a}$$

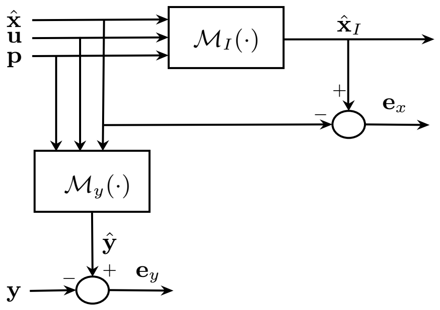

Fig. 2. Block diagram representing computation of the cost function for the proposed integral LPV architecture.

where

$$J = \underbrace{\sum_{k=0}^{N-1} \|\overbrace{\hat{\mathbf{y}}(t_k) - \mathbf{y}(t_k)}^{\mathbf{e}_y}\|^2}_{J_{\mathbf{y}}} + \underbrace{\alpha \int_{t_0}^{t_{N-1}} \|\overbrace{\hat{\mathbf{x}}_I(\tau) - \hat{\mathbf{x}}(\tau)}^{\mathbf{e}_x}\|^2 \, d\tau}_{J_{\mathbf{x}}}, \tag{8b}$$

with

$$\hat{\mathbf{y}}(t_k) = \mathcal{M}_y(\hat{\mathbf{x}}(t_k), \mathbf{u}(t_k), \mathbf{p}(t_k);\ \Theta_y), \tag{8c}$$

$$\hat{\mathbf{x}}_I(t) = \hat{\mathbf{x}}(0) + \int_0^t \mathcal{M}_x(\hat{\mathbf{x}}(\tau), \mathbf{u}(\tau), \mathbf{p}(\tau);\ \Theta_x)\, d\tau. \tag{8d}$$

The hyper-parameter $\alpha > 0$ acts as a tuning knob balancing the relative importance of the fitting cost J_y and the regularization cost J_x, and it can be chosen through cross-validation. The operations required to compute the cost J in (8b) are depicted in Fig. 2.

3.3 Optimization algorithm

It is important to note that the optimization problem (8) is *infinite-dimensional* and thus computationally intractable. Indeed, the continuous-time state signal $\hat{\mathbf{x}}(t) \in \mathbb{R}^{n_x}$, $t \in [t_0\ t_{N-1}]$ is one of the problem's decision variables. Following the rationale in [15], we employ numerical techniques to transform (8) into a finite-dimensional problem. In particular, the state signal $\hat{\mathbf{x}}(t)$ is approximated using a finite-dimensional parametrization. For simplicity of exposition, we represent the state signal with a *piecewise constant* parametrization, where $\hat{\mathbf{x}}(t)$ is constant in the intervals $[t_k\ t_{k+1}]$, $k = 0, 1, \ldots, N-1$. In general, more complex parametrizations for $\hat{\mathbf{x}}$ such as piecewise linear or polynomial could be used. Furthermore, we approximate the integrals in (8b) and (8d) by applying a numerical integration scheme. For simplicity, in this work we apply the classical *rectangular approximation* rule for the numerical integration of (8b) and (8d). Other quadrature rules such as trapezoidal or Gaussian quadrature could be alternatively considered.

Overall, the *piecewise constant* parametrization of the signals $\hat{\mathbf{x}}(t)$, $\mathbf{u}(t)$, $\mathbf{p}(t)$ with the *rectangular quadrature* of the integrals leads to the following approximation:

$$\int_{t_0}^{t_{N-1}} \|\hat{\mathbf{x}}_I(\tau) - \hat{\mathbf{x}}(\tau)\|^2 d\tau \approx \sum_{k=1}^{N-1} \|\hat{\mathbf{x}}_I(t_k) - \hat{\mathbf{x}}(t_k)\|^2 \Delta t_k,$$

where $\Delta t_k = t_k - t_{k-1}$, and (8d) can be approximated with the following Riemann sum:

$$\hat{\mathbf{x}}_I(t_k) \approx \hat{\mathbf{x}}(0) + \sum_{j=0}^{k-1} \Delta t_{j+1} \mathcal{M}_x(\hat{\mathbf{x}}(t_j), \mathbf{u}(t_j), \mathbf{p}(t_j);\ \Theta_x).$$

Note that, for computational convenience, the sum in the equation above can be constructed recursively as:

$$\hat{\mathbf{x}}_I(t_{k+1}) = \hat{\mathbf{x}}_I(t_k) + \overbrace{\Delta t_{k+1} \mathcal{M}_x(\hat{\mathbf{x}}(t_k), \mathbf{u}(t_k), \mathbf{p}(t_k)); \Theta_x)}^{\Delta \mathbf{x}_k}.$$

From the linear dependence of the map $\mathcal{M}_x(\cdot)$ on Θ_x in (5a), the equation above can be rewritten as:

$$\hat{\mathbf{x}}_I(t_{k+1}) \ = \ \hat{\mathbf{x}}_I(t_k) + \Delta t_{k+1} \Theta_x \begin{bmatrix} \begin{bmatrix} 1 \\ \mathbf{p}(t_k) \end{bmatrix} \otimes \hat{\mathbf{x}}(t_k) \\ \begin{bmatrix} 1 \\ \mathbf{p}(t_k) \end{bmatrix} \otimes \mathbf{u}(t_k) \end{bmatrix}. \tag{9}$$

Similarly, as the map $\mathcal{M}_y(\cdot)$ is a linear function of Θ_y, substituting (5b) in (8c) yields:

$$\hat{\mathbf{y}}(t_k) = \Theta_y \begin{bmatrix} \begin{bmatrix} 1 \\ \mathbf{p}(t_k) \end{bmatrix} \otimes \hat{\mathbf{x}}(t_k) \\ \begin{bmatrix} 1 \\ \mathbf{p}(t_k) \end{bmatrix} \otimes \mathbf{u}(t_k) \end{bmatrix}.$$

With the above considerations, the cost function J in (8a) is minimized w.r.t. the parameters $\{\hat{\mathbf{x}}, \Theta_x, \Theta_y\}$ by using the coordinate-descent approach described in Algorithm 1. With a slight abuse of notation, the optimization variable $\hat{\mathbf{x}}$ in Algorithm 1 denotes the finite-dimensional representation of the state signal $\hat{\mathbf{x}}$, i.e., $\hat{\mathbf{x}} = \{\hat{\mathbf{x}}(t_0), \ldots, \hat{\mathbf{x}}(t_{N-1})\}$.

Given an initial guess $\hat{\mathbf{x}}^{(0)}$ of the state sequence, at each iteration $n \geq 1$, Algorithm 1 alternates between two steps: Step 1.1 and Step 1.2. In particular, at Step 1.1, model parameters Θ_x, Θ_y are computed by solving (8a) for a *fixed* state sequence $\hat{\mathbf{x}}^{(n-1)}$ obtained at the iteration $(n-1)$. Subsequently, at Step 1.2, the state sequence $\hat{\mathbf{x}}^{(n)}$ is estimated by minimizing the cost (8a) for *fixed* model parameters $\Theta_x^{(n)}$ and $\Theta_y^{(n)}$ obtained from Step 1.1 at the n-th iteration. The procedure continues until a maximum number of iterations is reached, or a certain convergence criterion is met (Step 2).

We stress that the solutions at Step 1.1 and 1.2 of Algorithm 1 may be obtained *analytically* through linear least

Algorithm 1 Coordinate-descent optimization for the estimation of states $\hat{\mathbf{x}}$ and model parameter matrices Θ_x, Θ_y

Input: Training dataset $\mathcal{D} = \{\mathbf{u}(t_k), \mathbf{y}(t_k), \mathbf{p}(t_k)\}_{k=0}^{N-1}$; initial guess $\hat{\mathbf{x}}^{(0)}$; tuning parameter α; tolerance ϵ, maximum number of iteration $n_{\max}$.

```
1. Iterate for $n = 1, \dots$
   1.1. $\Theta_x^{(n)}, \Theta_y^{(n)} \leftarrow \arg\min_{\Theta_x,\Theta_y} J(\hat{\mathbf{x}}^{(n-1)}, \Theta_x, \Theta_y)$
   1.2. $\hat{\mathbf{x}}^{(n)} \leftarrow \operatorname{argmin}_{\hat{\mathbf{x}}} J(\hat{\mathbf{x}}, \Theta_x^{(n)}, \Theta_y^{(n)})$
2. Until $\|\hat{\mathbf{x}}^{(n)} - \hat{\mathbf{x}}^{(n-1)}\| \leq \epsilon$ or $n = n_{\max}$
```

Output: Estimates $\{\hat{\mathbf{x}}(t_k)\}_{k=0}^{N-1}$ and Θ_x, Θ_y.

squares. For example, at Step 1.1, the updated model parameters Θ_x and Θ_y (for fixed $\hat{\mathbf{x}}$) are given by:

$$\Theta_x' = (\mathbf{\Phi}_x'\mathbf{\Phi}_x)^{-1}\mathbf{\Phi}_x' X, \quad \Theta_y' = (\mathbf{\Phi}_y'\mathbf{\Phi}_y)^{-1}\mathbf{\Phi}_y' Y, \tag{10}$$

with $X = \left[(\hat{\mathbf{x}}(t_1)-\hat{\mathbf{x}}(t_0)) \; \dots \; (\hat{\mathbf{x}}(t_{N-1})-\hat{\mathbf{x}}(t_0)) \right]'$ and $Y = \left[(\hat{\mathbf{y}}(t_0))' \; \dots \; (\hat{\mathbf{y}}(t_N))' \right]'$. The matrices $\mathbf{\Phi}_x$ and $\mathbf{\Phi}_y$ are constructed such that their k-th *block row* is given by:

$$\mathbf{\Phi}_x[k,:] = \sum_{j=0}^{k-1} \Delta t_{j+1} \begin{bmatrix} \begin{bmatrix} 1 \\ \mathbf{p}(t_j) \end{bmatrix} \otimes \hat{\mathbf{x}}(t_j) \\ \begin{bmatrix} 1 \\ \mathbf{p}(t_j) \end{bmatrix} \otimes \mathbf{u}(t_j) \end{bmatrix},$$

$$\mathbf{\Phi}_y[k,:] = \sum_{j=0}^{k-1} \begin{bmatrix} \begin{bmatrix} 1 \\ \mathbf{p}(t_j) \end{bmatrix} \otimes \hat{\mathbf{x}}(t_j) \\ \begin{bmatrix} 1 \\ \mathbf{p}(t_j) \end{bmatrix} \otimes \mathbf{u}(t_j) \end{bmatrix},$$

where we introduce the notation $\mathbf{\Phi}_x[k,:]$ and $\mathbf{\Phi}_y[k,:]$ to refer to the k-th *block row* of respective matrices. Similarly, at Step 1.2, given the model parameters Θ_x and Θ_y, the state sequence $\hat{\mathbf{x}}$ can be computed via ordinary least-squares.

3.4 *Subsequence optimization algorithm*

Note that the size of the optimization variables $\hat{\mathbf{x}} = \{\hat{\mathbf{x}}(t_0), \dots, \hat{\mathbf{x}}(t_{N-1})\}$ at Step 1.2 of Algorithm 1 increases with the size of training data samples. For applications involving large training datasets, solving this optimization problem may become computationally intractable or Algorithm 1 can be very sensitive to the choice of the initial guess $\hat{\mathbf{x}}^{(0)}$ because of the large dimension of the optimization domain. In order to tackle this issue, we describe an approach based on shorter subsequences to compute the state sequence $\hat{\mathbf{x}}$ in an efficient manner.

Consider a subsequence of length T (typically, $T << N$). For a generic time instance t_i and for fixed model parameters Θ_x, Θ_y, we minimize the cost

$$J_T = \sum_{k=i}^{i+T-1} \|\hat{\mathbf{y}}(t_k) - \mathbf{y}(t_k)\|^2 + \alpha \int_{t_i}^{t_{i+T-1}} \|\hat{\mathbf{x}}_I(\tau) - \hat{\mathbf{x}}(\tau)\|^2 \, d\tau. \tag{11}$$

over the time interval $[t_i \;\; t_{i+T-1}]$. By approximating the integral in (11) with the rectangular quadrature, we may compute the subsequence variables $\hat{\mathbf{x}}_{t_i:t_{i+T-1}} = \{\hat{\mathbf{x}}(t_i), \dots, \hat{\mathbf{x}}(t_{i+T-1})\}$, by optimizing the following cost

$$\begin{aligned} &J_T(\hat{\mathbf{x}}_{\text{prev}}(t_{i-1}), \hat{\mathbf{x}}_{t_i:t_{i+T-1}}; \Theta_x, \Theta_y) \\ &= \sum_{k=i}^{i+T-1} \|\hat{\mathbf{y}}(t_k) - \mathbf{y}(t_k)\|^2 + \alpha \sum_{k=i}^{i+T-1} \|\hat{\mathbf{x}}_I(t_k) - \hat{\mathbf{x}}(t_k)\|^2 \Delta t_k \end{aligned} \tag{12}$$

with

$$\begin{aligned} \hat{\mathbf{y}}(t_k) &= \mathcal{M}_y(\hat{\mathbf{x}}(t_k), \mathbf{u}(t_k), \mathbf{p}(t_k);\; \Theta_y), \\ \hat{\mathbf{x}}_I(t_k) &= \hat{\mathbf{x}}_{\text{prev}}(t_{i-1}) \\ &\quad + \sum_{j=i-1}^{k} \mathcal{M}_x(\hat{\mathbf{x}}(t_j), \mathbf{u}(t_j), \mathbf{p}(t_k); \Theta_x) \Delta t_{j+1}. \end{aligned}$$

Note that $\hat{\mathbf{x}}_{\text{prev}}(t_{i-1})$ denotes the state optimized over the *previous* subsequence. Thus, $\hat{\mathbf{x}}_{\text{prev}}(t_{i-1})$ acts as the initial state for the current subsequence in order to enforce consistency of the state variables over the training dataset. The loss in (12) is minimized in a *rolling-horizon* manner as described in step 1.2 of Algorithm 2. In particular, at the n-th iteration, for fixed $\Theta_x^{(n)}, \Theta_y^{(n)}$ obtained from Step 1.1, the subsequence $\hat{\mathbf{x}}_{t_i:t_{i+T-1}}$ is computed by minimizing (12) starting from $t_i = 0$. Next, the time index is shifted T steps forward to compute the succeeding subsequence with $\hat{\mathbf{x}}_{\text{prev}}$ set to the last state of the previous subsequence. The process is repeated over all N training data samples. Algorithm 2 is run until a maximum number of iterations $n_{\max}$ is reached or a certain stopping criterion is met (Step 2). Note that the number of optimization variables at $\hat{\mathbf{x}}$ at Step 1.2 of Algorithm 2 grows linearly with T.

4 Case studies

In this section, we demonstrate the effectiveness of the proposed method via an academic example and two more complex case studies. The first case study concerns the identification of an electronic bandpass filter from an experimental dataset and the second one involves identification of plasma safety factor from a nonlinear tokamak plasma simulator. The sub-problems within the coordinate-descent steps are solved analytically via ordinary least squares. All computations are carried out on an i7 1.9-GHz Intel core processor with 32 GB of RAM running MATLAB R2019a.

Algorithm 2 Coordinate-descent optimization with short-size subsequences for estimation of state $\hat{\mathbf{x}}$ and model parameter matrices Θ_x, Θ_y.

Input: Training dataset $\mathcal{D} = \{\mathbf{u}(t_k), \mathbf{y}(t_k), \mathbf{p}(t_k)\}_{k=0}^{N-1}$; initial guess $\hat{\mathbf{x}}^{(0)}$; subsequence of length T; tuning parameter α; tolerance ϵ, maximum number of iteration $n_{\max}$.

```
1. Iterate for $n = 1, \ldots$
   1.1. Optimize $\Theta_x, \Theta_y$
        $\Theta_x^{(n)}, \Theta_y^{(n)} \leftarrow \arg\min_{\Theta_x, \Theta_y} J(\hat{\mathbf{x}}^{(n-1)}, \Theta_x, \Theta_y)$
   1.2. Optimize $\hat{\mathbf{x}}$
        1.2.1. set $i \leftarrow 0$, $\hat{\mathbf{x}}_{\text{prev}}(t_0) \leftarrow \hat{\mathbf{x}}_0$
        1.2.2. solve short-sequence optimization (12)

$\hat{\mathbf{x}}_{t_i:t_{i+T-1}} \leftarrow \underset{\hat{\mathbf{x}}_{t_i:t_{i+T-1}}}{\text{argmin}} \; J_T(\hat{\mathbf{x}}_{\text{prev}}(t_{i-1}), \hat{\mathbf{x}}_{t_i:t_{i+T-1}}, \Theta_x^{(n)}, \Theta_y^{(n)})$

        1.2.3. $i \leftarrow i + T$
        1.2.4. $\hat{\mathbf{x}}_{\text{prev}}(t_{i-1}) \leftarrow \hat{\mathbf{x}}_{t_{i-1}}$
        1.2.5. Go to step 1.2.2 until $i < N - T - 1$.
        1.2.6. set $\hat{\mathbf{x}}^{(n)} \leftarrow \hat{\mathbf{x}}_{t_0:t_{N-1}}$
2. Until $\|\hat{\mathbf{x}}^{(n)} - \hat{\mathbf{x}}^{(n-1)}\| \leq \epsilon$ or $n = n_{\max}$
```

Output: Estimates $\{\hat{\mathbf{x}}(t_k)\}_{k=0}^{N-1}$ and Θ_x, Θ_y.

4.1 Academic example

An MIMO LPV state-space model with input dimension $n_u = 2$, scheduling signal dimension $n_p = 2$, state dimension $n_x = 2$, and output dimension $n_y = 2$ is considered as a data-generating system, with:

$$\begin{bmatrix} A_0 & A_1 & A_2 \end{bmatrix} = \left[\begin{array}{cc|cc|cc} 0 & 0.5 & 0.4 & 0 & 0.2 & 0 \\ -0.5 & 0 & 0 & 0.3 & 0 & 0 \end{array}\right],$$

$$B_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, C_0 = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, D_0 = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}.$$

The data are generated by simulating [1] the continuous-time LPV system $\mathcal{S}$ in (1) and sampling the outputs, inputs and scheduling signals with a sampling time of 0.01 s. The input trajectories vary in the domain $\mathbf{u}(t) \in [-2, 2] \times [-2, 2]$ and the scheduling signal varies within the box $\mathbf{p}(t) \in [-1, 1] \times [-1, 1]$. The output measurements are corrupted by a zero-mean white Gaussian noise η with distribution $\eta(t) \sim \mathcal{N}(0, \Sigma)$ where Σ is diagonal. In order to analyze the statistical properties of the proposed identification algorithm, a Monte-Carlo study with 50 runs is performed. At each Monte-Carlo run, a different realization of input $\mathbf{u}(t)$, scheduling signal $\mathbf{p}(t)$ and noise $\eta(t)$ is considered. The noise covariance matrix Σ is chosen such that for each Monte-Carlo run, the *signal-to-noise* ratio (SNR) is $\text{SNR}_{\mathbf{y}}^{[i]} = 10 \log \frac{\sum_{k=0}^{N-1} (\mathbf{y}_i(t_k) - \eta_i(t_k))^2}{\sum_{k=0}^{N-1} (\eta_i(t_k))^2} = 15$ dB for both output channels. Here, the subscript i denotes the output channel and thus $\text{SNR}_{\mathbf{y}}^{[i]}$ corresponds to the SNR of the i-th channel. For each Monte-Carlo run, a training dataset of $N = 2000$ samples and a *noise-free* validation dataset of $N_{\text{val}} = 500$ samples is generated. The quality of the estimated LPV model is assessed via the *Best Fit Rate* (BFR) index defined as $\text{BFR}_{\mathbf{y}}^{[i]} =$

$$\max\left\{1 - \sqrt{\frac{\sum_{k=0}^{N_{\text{val}}-1} (\mathbf{y}_i(t_k) - \hat{\mathbf{y}}_i(t_k))^2}{\sum_{k=0}^{N_{\text{val}}-1} (\mathbf{y}_i(t_k) - \bar{\mathbf{y}}_i)^2}}, 0\right\} \times 100\%,$$

with $\hat{\mathbf{y}}_i$ being the simulated model output and $\bar{\mathbf{y}}_i$ is the sample mean of the output over the validation set.

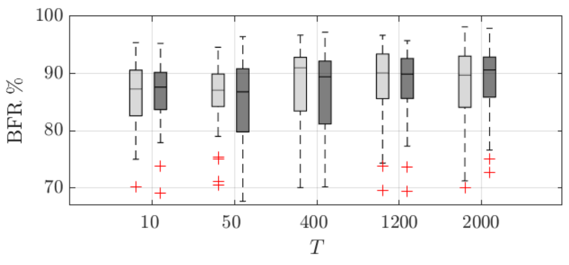

Fig. 3. Monte-Carlo analysis: boxplots of achieved BFR indices for output channels $\mathbf{y}_1$ (light) and $\mathbf{y}_2$ (dark) for different subsequence lengths

For identification, we consider an LPV state-space affine model structure (4) with state dimension set to the true system dimension $n_x = 2$. The LPV model matrices Θ_x, Θ_y and the state sequence $\hat{\mathbf{x}}$ are estimated by running the coordinate-descent Algorithm 2 for $n_{\max} = 30$ iterations, with initial guess $\hat{\mathbf{x}}^{(0)}$ of the state sequence chosen randomly from a uniform distribution in the interval $[0, 0.01)$. The regularization parameter α is set to 1. In order to asses the effect of the chosen subsequence length T, we carry out simulation studies with 5 different subsequences lengths $T = \{10, 50, 400, 1200, 2000\}$.

The boxplots of the achieved BFR indices over different Monte-Carlo runs are plotted in the Fig. 3. The estimated models achieve a good performance in terms of BFR index for all T, with only a few outliers. The average computation time per iteration of the coordinate-descent algorithm for different lengths T is plotted in Fig. 4. The computation time grows with increasing values of the sequence length T. From these figures, we observe that in general the performance improves with increase in T. Overall, faster execution times can be achieved by setting a short sequence at the cost of a slight decrease in the accuracy of the estimated models.

Furthermore, the estimated model outputs over the val-

[1] The trajectories of the system $\mathcal{S}$ in (1) are generated using MATLAB's `ode45` function which employs the 4-th order Runge-Kutta method with an adaptive step size.

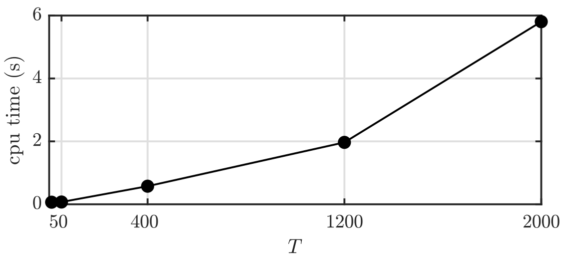

Fig. 4. Computation time per iteration *vs* subsequence length

Table 1
Academic example: BFR achieved on validation data.

	CT LPV-SS	DT NL-ARX	DT LPV-ARX
$\mathrm{BFR}_{\mathbf{y}}^{[1]}$	90%	88%	88%
$\mathrm{BFR}_{\mathbf{y}}^{[2]}$	91%	93%	90%

idation dataset are plotted in Fig. 5 for the case of $T = 400$. Only a subset of the simulated output has been plotted for the sake of better visualization. To assess the convergence properties of the coordinate-descent algorithm, the mean and the standard deviation of the cost J (over the Monte-Carlo runs) is plotted in Fig. 6 against the iterations. It can be observed that convergence of the coordinate-descent algorithm is achieved in about 30 iterations, and the reconstructed outputs match closely with the true ones.

Finally, the proposed algorithm is compared with two discrete-time approaches, which identify a *non-linear auto-regressive with exogenous* input (NL-ARX) model and a linear parameter-varying ARX (LPV-ARX) model in input-output form. The NL-ARX and LPV-ARX models are identified by minimizing the prediction error w.r.t. model parameters using the MATLAB System Identification Toolbox [14]. Note that, for the LPV-ARX model, the solution is obtained via ordinary least squares. For both cases, we consider a second-order model structure, *i.e.*, the output at each time instance depends on the past two outputs and inputs samples. For NL-ARX, we choose sigmoid non-linearities. In LPV-ARX model, the dependency of the model coefficients on scheduling parameters is considered to be affine. The obtained BFRs are reported in Table 1.

4.2 Identification of an electronic bandpass filter

We consider the identification of an LPV model describing the behaviour of a second-order electronic bandpass filter, which is implemented using an n-type JFET transistor in parallel with a variable resistor, as shown in Fig. 7. The resonant frequency of the bandpass filter varies according to the gate-source voltage of the transistor, which is chosen as a scheduling signal $\mathbf{p}(t)$ for the LPV model to be identified.

From the available repository [11], we consider the dataset termed MS_Harm_h20_N15640_RMS70_P2P700.

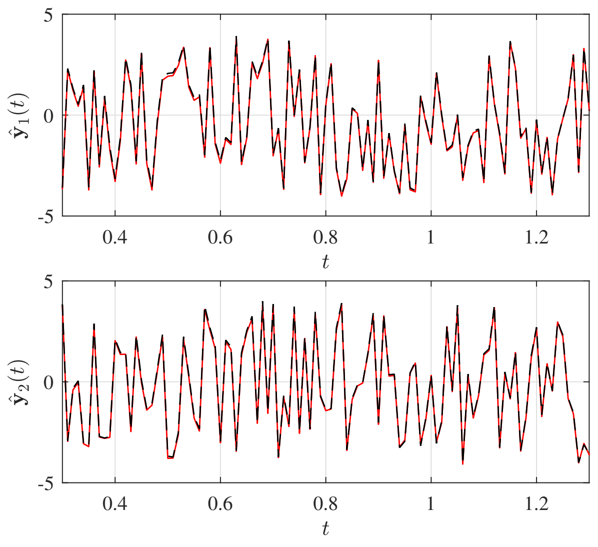

Fig. 5. Validation: True output (red) *vs* estimated outputs (dashed black) for output channel 1 (Left panel) and output channel 2 (Right panel) with $T = 400$.

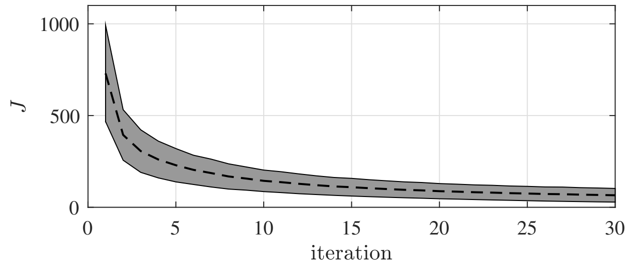

Fig. 6. Training: Mean (dashed black) $\pm$ std deviation (shaded gray) of the cost function *vs* number of iterations.

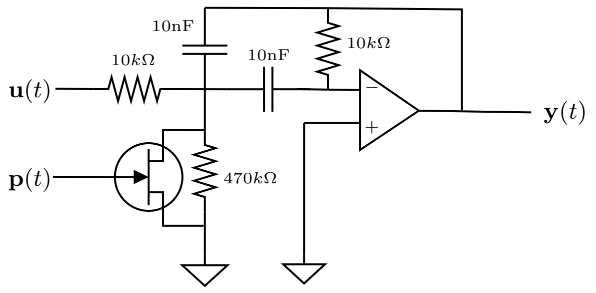

Fig. 7. Electric schematic of the electronic bandpass filter.

The dataset consists of 6 different realizations of the input and 2 different realizations of the scheduling signal. Each realization has $N = 46920$ samples of input, output and scheduling signal measurements. The reader is referred to [11] for a detailed description of the applied excitation input signal $\mathbf{u}(t)$ and scheduling variable $\mathbf{p}(t)$.

For identification, we consider the affine LPV state-space representation (4) with model order set to $n_x = 3$. One of the realization of the signals consisting of $N = 46920$ samples is used for training. Note that the size of the state-sequence $\hat{\mathbf{x}}$ to be optimized is very large, namely

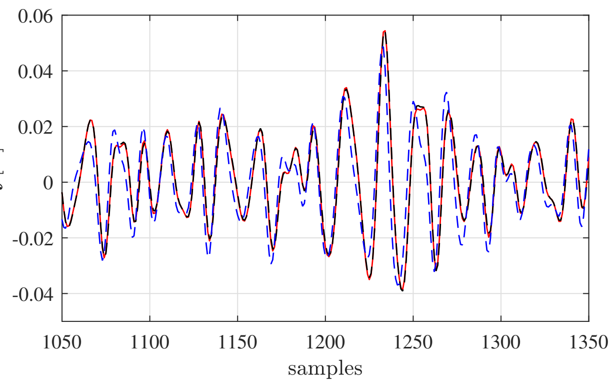

Fig. 8. Electronic bandpass filter: real output (red) *vs* simulated outputs with LPV model (dashed black) and LTI model (dashed blue).

Table 2
Electronic bandpass filter: BFR achieved on validation data.

CT LPV-SS	DT NL-ARX	DT LPV-ARX
93%	63%	87%

46920×3. Thus, the optimization problem at Step 1.2 of Algorithm 1 is computationally intractable on the considered platform. This requires to employ the short-size subsequence algorithm discussed in Section 3.4.

The LPV model is identified by running Algorithm 2 for $n_{\max} = 50$ iterations with subsequence length $T = 2000$. The hyper-parameter α is set to $\alpha = 1$. As initial guess, the state sequence $\hat{\mathbf{x}}^{(0)}$ is set to the states of an LTI state-space model, identified via the N4SID subspace method [24]. The average computation time required for one iteration of the coordinate-descent algorithm to process the entire batch of training data is 118 seconds.

The performance of the identified model is evaluated on an independent validation dataset (not used for training) of size $N_{\text{val}} = 15640$ samples. For a better visualization, part of the simulated output of the identified continuous-time LPV model is plotted in Fig. 8 along with the real output. For comparison, the simulated output of a continuous-time LTI model identified with N4SID algorithm is also plotted. The achieved BFR index for the LPV model identified with the proposed approach is 93% *vs* 62% achieved by the LTI model. We also compare the proposed approach with the discrete-time prediction error method identifying a third-order NL-ARX model (with sigmoid non linearity, cf. [14]) and an LPV-ARX model (with affine dependence) in input-output form. The results are reported in Table 2, which shows that the CT LPV-SS model identified with the proposed algorithm is able to reconstruct the output more accurately than the NL-ARX and LPV-ARX models.

4.3 Identification of the plasma safety factor from the RAPTOR code

As a last case study, we consider identification of the plasma safety factor from the *RApid Plasma Transport simulatOR* (RAPTOR) simulator described in [4].

4.3.1 System description

The RAPTOR is a predictive transport code simulating the 1D plasma coupled poloidal flux diffusion and the electron temperature transport.

The safety factor q, and its reciprocal (ι), are among the key parameters to analyze the plasma stability and performance. The evolution of these parameters is proportional to the spatial derivative of the poloidal magnetic flux $\psi(\rho,t)$, as: $\iota = \frac{1}{q} = \frac{\partial \Psi}{\partial \phi} = \frac{1}{2\pi B_0 \rho}\frac{\partial \psi}{\partial \rho}$, where $\rho = \sqrt{\phi/\pi B_0}$. Normalized ρ_n is defined as ρ/ρ_e, where ρ_e is the toroidal flux enclosed by the last closed flux surface. Here $\phi(\rho)$ is the toroidal flux and B_0 is the toroidal magnetic field at the center of the vacuum vessel. The evolution of the poloidal flux in the RAPTOR code is modelled using the following equation [4]:

$$\frac{\partial \psi}{\partial t} = \eta_{\parallel}\frac{R_0 J^2}{\mu_0 \rho}\frac{\partial}{\partial \rho}\left(\frac{G_2}{J}\frac{\partial \psi}{\partial \rho}\right) - \eta_{\parallel}\frac{V'}{2\pi\rho}(j_{bs} + j_{aux}), \quad (14)$$

where μ_0 and R_0 are constant parameters. The geometric profile quantities $J(\rho), V'(\rho), G_2(\rho)$ are considered to be constant under the assumption that the toroidal flux distribution does not change. The parameter $\eta_{\parallel}(\rho,t)$ is the parallel electrical resistivity of the plasma. This variable is time varying and related to the electron temperature T_e according to $\eta_{\parallel} \sim T_e^{-3/2}$. The auxiliary sources current density j_{aux} is generated by external sets of current drive actuators. In this simulation configuration, two clusters of electron cyclotron current drive (ECCD) gyrotrons are used. The total j_{aux} is calculated as sum of each individual cluster j_{eccd}. The contribution of each cluster is calculated using the scaling law [4]:

$$j_{eccd}(\rho,t) = \frac{T_e}{n_e} j_{dis,i}(\rho) P_{ec,i}(t), \quad (15)$$

which is the product of the weighted Gaussian distributions $j_{dis}(\rho)$ (given in [5]) with the current-drive efficiency $\left(\frac{T_e}{n_e}\right)$ and the input power $P_{ec}(t)$. The bootstrap current density j_{bs} comes from a self-generated plasma current, its magnitude in this setup is much lower than the one of j_{aux}. The boundary conditions of (14) are:

$$\frac{\partial \psi(0,t)}{\partial \rho} = 0, \qquad \frac{G_2(\rho_e)}{\mu_0}\frac{\partial \psi(\rho_e,t)}{\partial \rho} = I_p(t), \quad (16)$$

where $I_p(t)$ is the total plasma current.

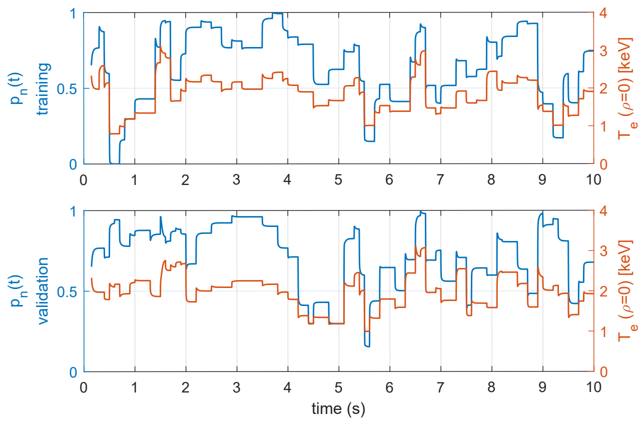

Fig. 9. Plot of $p_n(t)$ and $T_e(\rho = 0, t)$.

4.3.2 LPV modelling

A lumped version of the dynamics in (14) is modelled using an affine LPV representation. The safety factor inverse ι in three discretized points is selected as the output of the system $\mathbf{y}(t) = \{\iota(\rho_n = 0, t), \iota(\rho_n = 0.1, t), \iota(\rho_n = 0.2, t)\}$. The control inputs are the power requests to the ECCD-clusters $\mathbf{u}(t) = \{P_{ec1}(t), P_{ec2}(t)\}$. The first clusters are one with counter-current drive ($P_{ec1}(t)$), and the second with co-current drive ($P_{ec2}(t)$). The total current $I_p(t)$ is kept to the constant value 120 kA after the initial ramp-up phase of 80 ms. The simulations are executed over 10 s with sampling time of 1 ms, and the samples from the first 0.15 s are excluded from the training data.

The choice of the scheduling parameters is inspired by the physical equation (14). The first scheduling parameter $\mathbf{p}_1(t) = p_n(t)$ is chosen based on the time-varying profile $\eta_{\|}(\rho, t)$, which is described as proposed in [16]:

$$\eta_{\|}(\rho, t) = p_n(t)\eta_{\|,min} + (1 - p_n(t))\eta_{\|,max}, \qquad (17)$$

where $\eta_{\|,min}$ and $\eta_{\|,max}$ are fixed bounding profiles (minimum and maximum value, respectively) of $\eta_{\|}$. The second scheduling term $\mathbf{p}_2(t)$ comes from the multiplication of the scaling efficiency factor $(\frac{T_e}{n_e})$ in the model of j_{eccd} from (15), and $\eta_{\|}$ in (14). Thus, the second scheduling parameter is chosen as $\mathbf{p}_2(t) = p_n(t) \cdot T_e(\rho = 0, t)$. The temperature T_e at the core of the plasma ($\rho = 0$) is selected to define the scheduling parameter because of the peak location of the deposition of $j_{dis,i}(\rho)$ of both ECCD clusters. The time evolution of the signals $p_n(t)$ and $T_e(\rho = 0, t)$ is given in Fig. 9.

With the chosen scheduling parameters, an LPV affine model (4) is considered with model order set to $n_x = 3$. The model is identified by running Algorithm 2 for $n_{\max} = 80$ iterations with subsequence length $T = 500$. The state variables are initialized as $\hat{\mathbf{x}}^{(0)} = \mathbf{y}$. The estimated model output in the validation dataset is shown in Fig. 10. The results are compared with a CT LTI state-space model identified using the N4SID algorithm. The BFR indices for the CT LTI model for the three output channels are $\{85, 83, 78\}\%$ respectively. The proposed algorithm is also compared with a third-order DT NL-ARX and to an LPV-ARX model obtained via prediction error minimization. The achieved BFR indexes for the CT LPV model identified with the proposed approach and DT NL-ARX, LPV-ARX are reported in Table 3. The obtained results show that the LPV-SS model estimated using the proposed algorithm closely matches with the actual evolution of the safety factor from the RAPTOR simulator, and has a better accuracy compared to all the other methods we have tested.

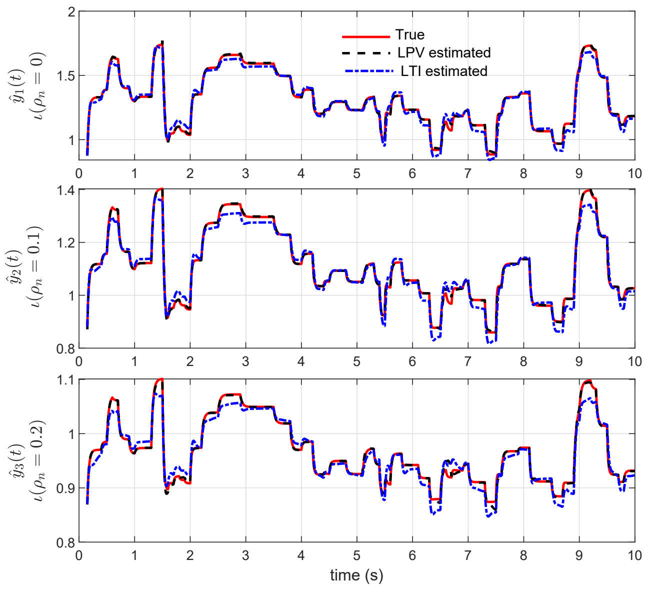

Fig. 10. The outputs of the RAPTOR simulator: True output (red) *vs* simulated outputs with LPV model (dashed black) and LTI model (dashed blue).

5 Conclusion

This paper has presented an integral architecture for direct identification of continuous-time LPV models in state-space form. A coordinate-descent algorithm tailored for the proposed integral architecture is presented in combination with a subsequence optimization heuristic for efficient computation of state sequence and model parameters. The main advantage of the proposed algorithm is the high computational efficiency, as the solution to the sub-problems at each step of the coordinate-

Table 3
BFR achieved on validation data of the RAPTOR simulator.

	output 1	output 2	output 3
CT LPV-SS	96%	96%	94%
DT NL-ARX	84%	81%	77%
DT LPV-ARX	87%	85%	71%

descent algorithm can be obtained in closed form. Current research activities are focused on the introduction of regularization criteria in order to enforce smoothness of the estimated hidden state, thus improving the overall accuracy of the proposed estimation algorithm.

References

[1] B. A. Bamieh and L. Giarré. Identification of linear parameter-varying models. *International Journal of Robust Nonlinear Control*, 12(9):841–853, 2002.

[2] M. Bergamasco and M. Lovera. Subspace identification of continuous-time state-space LPV models. *Linear Parameter-Varying System Identification*, pages 201–230, 2012.

[3] P. B. Cox and R. Tóth. Linear parameter-varying subspace identification: A unified framework. *Automatica*, 123:109296, 2021.

[4] F. Felici and O. Sauter. Non-linear model-based optimization of actuator trajectories for tokamak plasma profile control. *Plasma Physics and Controlled Fusion*, 54(2):025002, 2012.

[5] F. Felici, O. Sauter, S. Coda, BP. Duval, TP. Goodman, JM. Moret, JI. Paley, TCV Team, et al. Real-time physics-model-based simulation of the current density profile in tokamak plasmas. *Nuclear Fusion*, 51(8):083052, 2011.

[6] H. Garnier. Direct continuous-time approaches to system identification. overview and benefits for practical applications. *European Journal of control*, 24:50–62, 2015.

[7] H. Garnier and L. Wang. *Identification of Continuous-time Models from Sampled Data.* Springer Publishing Company, 2008.

[8] H. Garnier and P. C. Young. The advantages of directly identifying continuous-time transfer function models in practical applications. *International Journal of Control*, 87(7):1319–1338, 2014.

[9] P. Gáspár, Z. Szabó, and J. Bokor. Gray-box continuous-time parameter identification for LPV models with vehicle dynamics applications. In *Proc. of IEEE Mediterrean Conference on Control and Automation*, pages 393–398, 2005.

[10] J. Goos and R. Pintelon. Continuous-time identification of periodically parameter-varying state space models. *Automatica*, 71:254 – 263, 2016.

[11] J. Lataire, E. Louarroudi, R. Pintelon, and Y. Rolain. Benchmark data on a linear time- and parameter-varying system. In *Proc. of the 17th IFAC Symposium on System Identification*, pages 1477–1482, Beijing, China, 2015.

[12] J. Lataire, R. Pintelon, D. Piga, and R. Tóth. Continuous-time linear time-varying system identification with a frequency-domain kernel-based estimator. *IET Control Theory Applications*, 11(4):457–465, 2017.

[13] V. Laurain, R. Tóth, D. Piga, and M. A. H. Darwish. Sparse RKHS estimation via globally convex optimization and its application in LPV-IO identification. *Automatica*, 115, 2020.

[14] Lennart Ljung. *System identification toolbox: User's guide.* The Matlab user's guide, 2019.

[15] B. Mavkov, M. Forgione, and D. Piga. Integrated neural networks for nonlinear continuous-time system identification. *IEEE Control Systems Letters*, 4(4):851–856, 2020.

[16] B. Mavkov, E. Witrant, C. Prieur, E Maljaars, F Felici, O Sauter, et al. Experimental validation of a lyapunov-based controller for the plasma safety factor and plasma pressure in the TCV tokamak. *Nuclear Fusion*, 58(5):056011, 2018.

[17] M. Mejari, B. Mavkov, M. Forgione, and D. Piga. An integral architecture for identification of continuous-time state-space LPV models. In *Proc. of the 4th IFAC Workshop on Linear Parameter Varying Systems*, Milan, Italy, 2021.

[18] M. Mejari, V.V. Naik, D. Piga, and A. Bemporad. Identification of hybrid and linear parameter-varying models via piecewise affine regression using mixed integer programming. *International Journal of Robust Nonlinear Control*, 30:5802–5819, 2020.

[19] M. Mejari and M. Petreczky. Realization and identification algorithm for stochastic LPV state-space models with exogenous inputs. *Proc. of the 3rd IFAC Workshop on Linear Parameter Varying Systems*, 52(28):13 – 19, 2019.

[20] M. Mejari, D. Piga, and A. Bemporad. A bias-correction method for closed-loop identification of Linear Parameter-Varying systems. *Automatica*, 87:128–141, 2018.

[21] A. Padilla, H. Garnier, P. C. Young, F. Chen, and J. I. Yuz. Identification of continuous-time models with slowly time-varying parameters. *Control Engineering Practice*, 93:104165, 2019.

[22] D. Piga. Finite-horizon integration for continuous-time identification: bias analysis and application to variable stiffness actuators. *International Journal of Control*, 93(10):2378–2391, 2020.

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

[24] B. van Overschee, P. de Moor. *Subspace Identification for Linear Systems.* Kluwer Academic Publishers, Springer US, 1996.

[25] V. Verdult and M. Verhaegen. Kernel methods for subspace identification of multivariable LPV and bilinear systems. *Automatica*, 41(9):1557–1565, 2005.