Fixed-size LS-SVM LPV System Identification for Large Datasets

International Journal of Control, Automation, and Systems VV(X) (YYYY) 1-13
http://dx.doi.org/10.1007/s12555-xxx-xxxx-x
ISSN:1598-6446 eISSN:2005-4092
http://www.springer.com/12555

Fixed-size LS-SVM LPV System Identification for Large Datasets

Luca Cavanini, Riccardo Felicetti, Francesco Ferracuti*, and Andrea Monteriù

Abstract: In this paper, we propose an efficient method for handling large datasets in Linear Parameter-Varying (LPV) model identification. The method is based on Least-Squares Support Vector Machine (LS-SVM) identification in the primal space. To make the identification computationally feasible, even for very large datasets, we propose estimating a finite-dimensional feature map. To achieve this, we propose a two-step method to reduce the computational effort. First, we define the training set as a fixed-size subsample of the entire dataset, considering collision entropy for subset selection. The second step involves approximating the feature map through the eigenvalue decomposition of the kernel matrices. This paper considers both AutoRegressive with eXogenous input (ARX) and State-Space (SS) model forms. By comparing the problem formulation in the primal and dual spaces in terms of accuracy and computational complexity, the main advantage of the proposed technique is the reduction in space and time complexity during the training stage, making it preferable for handling very large datasets. To validate our proposed primal approach, we apply it to estimate LPV models using provided inputs, outputs, and scheduling signals for two nonlinear benchmarks: the parallel Wiener-Hammerstein system and the Silverbox system. The performances of our proposed approach are compared with the dual LS-SVM approach and the Kernel Principal Component Regression.

Keywords: Linear parameter-varying, support vector machines, system identification, ARX.

1. INTRODUCTION

System identification in general is concerned with the identification of appropriate models from only the measurements of input/output signals. Black-box modeling-based approaches try to identify a model when a priori knowledge is not available. In particular, black-box models allow gathering nonlinear relationships using different approaches including Artificial Intelligence (AI), and, specifically, machine learning techniques ([1–4]). Linear Parameter-Varying (LPV) system identification has received considerable attention in the last few years from the control community. LPV system identification is recognized as a powerful tool to handle the complexity of nonlinear systems via its scheduling variable [5–7]. Among nonparametric approaches, the Least Squares Support Vector Machines (LS-SVM) framework ([8]) presents a way to bypass the difficulties associated with the selection of basis functions in LPV system identification ([9]). The LS-SVM solves a linear Least Squares (LS) problem in a computationally efficient way and it is capable of capturing difficult nonlinear dependencies. A Kernel method using LS-SVM to identify Auto Regressive eXogenous (ARX) LPV models has been proposed in [10] and a similar approach is also used to identify LPV State-Space (SS) models ([11]). These approaches boil down to solving a linear system in the dual space. It is easier, in general, to solve a linear system than a quadratic program (as it happens in non-LS SVM), but a drawback is the loss of sparseness in the representation of the estimate that characterizes conventional (non-LS) SVM. Sparseness in SVMs means that several support vectors are associated with a zero coefficient, so they do not contribute in any way to the testing stage and can thus be omitted. In the context of LPV system identification by LS-SVM framework, such sparseness can be introduced in different ways. The authors of [12] scaled the estimated coefficients by polynomial weights, which are shrunk towards zero to enforce sparsity in the final LPV model. The authors of [13] introduced a term in the objective function to enforce sparsity in the estimate of the LPV model and the problem is solved in the dual space. The authors of [14] proposed a regularized LS-SVM that shrinks the derivative of the functions with respect to the scheduling signals to zero. The authors of [15] introduced sparseness by sequentially pruning the support vector spectrum to speed up the testing stage ([8]). The authors of [16] proposed a low-rank matrix approximation for efficient online updating. In [17], the authors presented a block-structured ar-

Manuscript received —; revised —; accepted —. Recommended by Associate Editor — under the direction of Editor —.

Luca Cavanini is with Industrial Systems and Control Ltd, Glasgow, UK (e-mail: l.cavanini@isc-ltd.com). Riccardo Felicetti, Francesco Ferracuti, and Andrea Monteriù are with Department of Information Engineering, Università Politecnica delle Marche, Ancona, Italy (e-mails: {r.felicetti, f.ferracuti, a.monteriu}@univpm.it).
* Corresponding author.

chitecture for direct identification of continuous-time LPV state-space models.

The LPV system identification by LS-SVM framework is solved in the dual form in the literature ([10, 11]). In fact, in the dual space, the definition and the computation of high-dimensional feature maps can be avoided by using the kernel trick, thus making the identification problem computationally tractable. However, as highlighted in [18], solving the dual problem in kernel-based learning is more convenient in the case the number of observations is small and the number of predictor variables is large: this almost always occurs in data mining problems. However, this does not hold in system identification, especially in the case of large datasets, where the number of data points makes the identification problem intractable in the dual space as well.

The main contribution of this paper is the introduction of fixed-size LS-SVM ([1, 2]) in the field of LPV system identification. Indeed, with the fixed-size solution, it is possible to compute a sparse approximation by using only a subsample of selected support vectors from the entire data set. Hence, in this paper, we fill the gap in the literature by proposing an efficient method for LPV model identification via LS-SVM in the primal space that can deal with large datasets. The main advantage of fixed-size LS-SVM is to reduce the computational complexity (both temporal and spatial) of the training stage, and also to introduce sparsity in the weighting vector to decrease the computational complexity in the prediction stage. We also compare the estimation of LPV models by LS-SVM in the primal space with the dual space-based approach ([10]) and the Kernel Principal Component Regression (KPCR) for LPV system identification ([3]). Two benchmark datasets, the so-called parallel Wiener-Hammerstein system, and the Silverbox system, are considered to test the estimation of LPV-ARX and LPV-SS systems in the primal space ([19, 20]).

The paper is organized as follows. In Section 2, the LPV-ARX and the LPV-SS model identification problems are formulated. Section 3 describes the proposed LPV-ARX and LPV-SS model identification method in the primal space, including the techniques employed to reduce the dimension in the primal space. Section 4 is devoted to comparing the primal and the dual space identification, focusing on computational complexity, for both the LPV-ARX and LPV-SS model identification. Section 5 defines the case studies and reports the simulation results for two numerical examples as well as two well-known benchmarks for model identification, i.e., the parallel Wiener-Hammerstein system and the Silverbox system. Finally, Section 6 concludes the paper by discussing the main advantages of the proposed primal space LS-SVM LPV identification strategy.

2. PROBLEM DEFINITION

2.1. LPV-ARX model

A SISO discrete-time LPV system in ARX form is represented as follows:

$$\begin{aligned} y(k)+\sum_{i=1}^{n_a} a_i(\mathbf{p}(k))y(k-i) = \\ \sum_{i=1}^{n_b} b_{i-1}(\mathbf{p}(k))u(k-i+1)+e(k) \end{aligned} \tag{1}$$

where $k \in \mathbb{Z}$ is the discrete time index, n_a is the output order, n_b is the input order, $u(k) : \mathbb{Z} \to \mathbb{U} \subseteq \mathbb{R}$ is the input signal, $y(k) : \mathbb{Z} \to \mathbb{Y} \subseteq \mathbb{R}$ is the output signal, $\mathbf{p}(k) : \mathbb{Z} \to \mathbb{P} \subseteq \mathbb{R}^{n_p}$ is the scheduling parameter vector with $\mathbb{P}$ being compact and $e(k) : \mathbb{Z} \to \mathbb{E} \subseteq \mathbb{R}$ is a white noise process. The time-varying parameters $a_i(\mathbf{p}(k)) : \mathbb{P} \to \mathbb{R}$ and $b_i(\mathbf{p}(k)) : \mathbb{P} \to \mathbb{R}$ have a static dependence on the scheduling parameter vector $\mathbf{p}(k)$. Without loss of generality and to simplify the notation, please note that in the proposed approach a_i and b_i depend only on the instantaneous value of the scheduling parameters vector $\mathbf{p}(k)$. The number of parameters to be estimated is

$$n_g = n_a + n_b. \tag{2}$$

A usual assumption in LPV system identification is that the coefficients $a_i(\mathbf{p}(k))$, $b_i(\mathbf{p}(k))$ are linear combinations of a set of basis functions ([10]):

$$\begin{aligned} a_i(\mathbf{p}(k)) &= \sum_{v=1}^{n_H} \omega_{i,v}\phi_{i,v}(\mathbf{p}(k)) \\ &= \boldsymbol{\omega}_i^T \boldsymbol{\phi}_i(\mathbf{p}(k)), \quad i = 1,\dots,n_a \\ b_i(\mathbf{p}(k)) &= \sum_{v=1}^{n_H} \omega_{\tilde{i},v}\phi_{\tilde{i},v}(\mathbf{p}(k)) \\ &= \boldsymbol{\omega}_{\tilde{i}}^T \boldsymbol{\phi}_{\tilde{i}}(\mathbf{p}(k)), \quad i = 1,\dots,n_b \end{aligned} \tag{3}$$

where $\tilde{i} = n_a+1,\dots,n_g$, $\boldsymbol{\phi}_i(\mathbf{p}(k)) : \mathbb{R}^{n_p} \to \mathbb{R}^{n_H}$ denotes an undefined, potentially infinite dimensional ($n_H = \infty$) feature map with static dependence on $\mathbf{p}(k)$, and $\boldsymbol{\omega}_i \in \mathbb{R}^{n_H}$ is a vector of weights. By using (3), the LPV-ARX model can be rewritten in regression form:

$$\hat{y}(\tilde{k}) = \sum_{i=1}^{n_g} \boldsymbol{\omega}_i^T \boldsymbol{\phi}_i(\mathbf{p}(\tilde{k}))x_i(\tilde{k}) \tag{4}$$

where k is the discrete time index related to the training dataset and $\tilde{k}$ is related to the testing dataset, and

$$x_i(\tilde{k}) = \begin{cases} y(\tilde{k}-i) & \text{if } i = 1,\dots,n_a \\ u(\tilde{k}-i+n_a+1) & \text{if } i = n_a+1,\dots,n_g \end{cases} \tag{5}$$

Equation (4) is the primal model representation of the LPV-ARX system. As the dimension of $\boldsymbol{\omega}_i$ is large (potentially infinite), the direct estimation of these weights in primal form becomes feasible only if sparsity is enforced.

The goal of the identification process in primal space is to find an approximation of the feature maps ϕ_i for $i = 1,\ldots,n_g$ and subsequently calculate the weight vectors ω_i given the identification dataset $\mathcal{D}_N = \{u(k),\mathbf{p}(k),y(k)\}_{k=1}^N$, where N is the number of observations.

2.2. LPV-SS model

A MIMO discrete-time LPV system in a state-space innovation noise model can be represented as ([21]):

$$\begin{aligned}\mathbf{x}(k+1) &= \mathbf{A}(\mathbf{p}(k))\mathbf{x}(k)+\mathbf{B}(\mathbf{p}(k))\mathbf{u}(k)\\ &\quad +\mathbf{K}(\mathbf{p}(k))\mathbf{e}(k)\\ \mathbf{y}(k) &= \mathbf{C}(\mathbf{p}(k))\mathbf{x}(k)+\mathbf{D}(\mathbf{p}(k))\mathbf{u}(k)+\mathbf{e}(k)\end{aligned} \tag{6}$$

where $k \in \mathbb{Z}$ is the discrete time index and $\mathbf{A}(\mathbf{p}(k)) : \mathbb{P} \to \mathbb{R}^{n_x\times n_x}$, $\mathbf{B}(\mathbf{p}(k)) : \mathbb{P} \to \mathbb{R}^{n_x\times n_u}$, $\mathbf{C}(\mathbf{p}(k)) : \mathbb{P} \to \mathbb{R}^{n_y\times n_x}$, $\mathbf{D}(\mathbf{p}(k)) : \mathbb{P} \to \mathbb{R}^{n_y\times n_u}$, $\mathbf{K}(\mathbf{p}(k)) : \mathbb{P} \to \mathbb{R}^{n_x\times n_y}$ are smooth functions of the time-varying scheduling parameter vector $\mathbf{p}(k) : \mathbb{Z} \to \mathbb{P} \subseteq \mathbb{R}^{n_p}$ with $\mathbb{P}$ being compact. The vectors $\mathbf{u}(k) : \mathbb{Z} \to \mathbb{U} \subseteq \mathbb{R}^{n_u}$, $\mathbf{y}(k) : \mathbb{Z} \to \mathbb{Y} \subseteq \mathbb{R}^{n_y}$, $\mathbf{x}(k) : \mathbb{Z} \to \mathbb{X} \subseteq \mathbb{R}^{n_x}$ represent the inputs, outputs, and states of the system at time k, while the vector $\mathbf{e}(k) : \mathbb{Z} \to \mathbb{E} \subseteq \mathbb{R}^{n_y}$ is a white noise process independent of $\mathbf{u}(k)$. By substituting $\mathbf{e}(k) = \mathbf{y}(k) - \mathbf{C}(\mathbf{p}(k))\mathbf{x}(k) - \mathbf{D}(\mathbf{p}(k))\mathbf{u}(k)$ in (6), the above set of equations can be rewritten as

$$\begin{aligned}\mathbf{x}(k+1) &= \tilde{\mathbf{A}}(\mathbf{p}(k))\mathbf{x}(k)+\tilde{\mathbf{B}}(\mathbf{p}(k))\mathbf{u}(k)\\ &\quad +\mathbf{K}(\mathbf{p}(k))\mathbf{y}(k)\\ \mathbf{y}(k) &= \mathbf{C}(\mathbf{p}(k))\mathbf{x}(k)+\mathbf{D}(\mathbf{p}(k))\mathbf{u}(k)+\mathbf{e}(k)\end{aligned} \tag{7}$$

where $\tilde{\mathbf{A}}(\mathbf{p}(k)) = \mathbf{A}(\mathbf{p}(k)) - \mathbf{K}(\mathbf{p}(k))\mathbf{C}(\mathbf{p}(k))$ and $\tilde{\mathbf{B}}(\mathbf{p}(k)) = \mathbf{B}(\mathbf{p}(k)) - \mathbf{K}(\mathbf{p}(k))\mathbf{D}(\mathbf{p}(k))$.

By using (7), the LPV-SS model can be rewritten in regression form:

$$\hat{\mathbf{x}}(k+1) = \mathbf{W}_\mathbf{x}\varphi_x^T(\mathbf{p}(k)) \tag{8}$$

$$\hat{\mathbf{y}}(k) = \mathbf{W}_\mathbf{y}\varphi_y^T(\mathbf{p}(k)) \tag{9}$$

where $\mathbf{W}_\mathbf{x} = [\mathbf{W}_{\mathbf{xx}}\ \mathbf{W}_{\mathbf{xu}}\ \mathbf{W}_{\mathbf{xy}}] \in \mathbb{R}^{n_x\times 3n_H}$ and $\mathbf{W}_\mathbf{y} = [\mathbf{W}_{\mathbf{yx}}\ \mathbf{W}_{\mathbf{yu}}] \in \mathbb{R}^{n_y\times 2n_H}$ are weighting matrices. The functions $\varphi_x^T(\mathbf{p}(k))$: $\mathbb{P} \to \mathbb{R}^{3n_H\times 1}$ and $\varphi_y^T(\mathbf{p}(k))$: $\mathbb{P} \to \mathbb{R}^{2n_H\times 1}$ are defined by

$$\begin{aligned}\varphi_x^T(\mathbf{p}(k)) &= \mathrm{col}\big(\phi_{xx}(\mathbf{p}(k))\mathbf{x}(k),\, \phi_{xu}(\mathbf{p}(k))\mathbf{u}(k),\\ &\qquad \phi_{xy}(\mathbf{p}(k))\mathbf{y}(k)\big)\\ \varphi_y^T(\mathbf{p}(k)) &= \mathrm{col}\,(\phi_{yx}(\mathbf{p}(k))\mathbf{x}(k),\, \phi_{yu}(\mathbf{p}(k))\mathbf{u}(k))\end{aligned} \tag{10}$$

where $\phi_{xx}(\mathbf{p}(k)), \phi_{yx}(\mathbf{p}(k)) : \mathbb{P} \to \mathbb{R}^{n_H\times n_x}$, $\phi_{xy}(\mathbf{p}(k)) : \mathbb{P} \to \mathbb{R}^{n_H\times n_y}$, and $\phi_{xu}(\mathbf{p}(k)), \phi_{yu}(\mathbf{p}(k)) : \mathbb{P} \to \mathbb{R}^{n_H\times n_u}$ are unknown feature maps. Equations (8)–(9) are the primal model representation of the LPV-SS system.

The goal of the identification process in primal space is to find an approximation of the above mentioned feature maps and subsequently calculate the weight matrices $\mathbf{W}_{\mathbf{xx}}, \mathbf{W}_{\mathbf{xu}}, \mathbf{W}_{\mathbf{xy}}, \mathbf{W}_{\mathbf{yx}}, \mathbf{W}_{\mathbf{yu}}$ given the identification dataset $\mathcal{D}_N = \{\boldsymbol{u}(k),\mathbf{p}(k),\boldsymbol{y}(k)\}_{k=1}^N$, where N is the number of observations.

3. LS-SVM IDENTIFICATION FOR LPV SYSTEMS IN PRIMAL SPACE

In this section, we illustrate LPV-ARX and LPV-SS identification in the primal space. From a computational point of view, identification in the primal space suffers from high-dimensionality of the feature maps, as the kernel trick holds in the dual space only: a tractable approximation of the feature map is needed, otherwise the unknown weights $\omega_i \in \mathbb{R}^{n_H}$, for $i = 1,\ldots,n_g$, must be calculated (see (3)). Considering large-scale problems, it has been motivated by [8] to choose a working set of fixed size $m \ll N$ where the value m is related to the Nyström subsample and an entropy-based subset selection. The LS-SVM approaches that make use of such fixed-size working set are known in the literature as fixed-size LS-SVM ([2]). Hence, first of all, we introduce two techniques to reduce the computational effort, namely, subset selection and feature map approximation. Then, we illustrate ARX and SS for LPV system identification in the primal space.

3.1. Subset Selection and Sparseness

In order to choose a subsample $n \ll N$ of the training set, a straightforward solution is to take a random subset of n samples among the N available samples. However, there is no guarantee that it represents the entire dataset in terms of coverage of the state space. On the other hand, it is impractical to try all the subsamples of the desired size n, train the LS-SVM model, and take the best result, due to the combinatorial explosion of the number of subsamples and also due to the time complexity of training several LS-SVM models. To overcome this issues, we propose to limit the number of subsamples to be tested and we choose the best subsample in terms of entropy (as proposed in [2, 8]) instead of training an LS-SVM on each subsample. Indeed, the n support vectors can be selected by maximizing the collision entropy (also known as quadratic Rényi entropy):

$$H_q = -\log \int p(x)^2 dx \tag{11}$$

where p is the density of the selected support vectors. In fact, the larger the collision entropy, the more the samples are well spread over the entire data region, thus leading to an improved identification. Moreover, the quadratic Rényi entropy can be conveniently approximated ([22]) by using

$$\int \hat{p}(x)^2 dx = \frac{1}{n^2}\mathbf{1}^T\mathbf{K}(\mathbf{p})\mathbf{1} \tag{12}$$

where $\mathbf{1} = (1,\dots,1)^T$ and $\mathbf{K}(\mathbf{p}) = \sum_{i=1}^{n_g} \mathbf{K}_i^g(\mathbf{p})$. The $(k,\tilde{k})$-th entry of the matrix $\mathbf{K}_i^g(\mathbf{p}) \in \mathbb{R}^{N\times N}$ is

$$K_i^g(\mathbf{p}(k),\mathbf{p}(\tilde{k})) = \phi_i(\mathbf{p}(k))^T \phi_i(\mathbf{p}(\tilde{k})) \tag{13}$$

for $k,\tilde{k} = 1,\dots,N$, where $K_i^g(\mathbf{p}(k),\mathbf{p}(\tilde{k})) : \mathbb{P}\times\mathbb{P} \to \mathbb{R}$.

Considering the Radial Basis Function (RBF) as kernel function, therefore the entries of $\mathbf{K}_i^g(\mathbf{p})$ are

$$K_i^g(\mathbf{p}(k),\mathbf{p}(\tilde{k})) = \exp\left(-\frac{\|\mathbf{p}(k)-\mathbf{p}(\tilde{k})\|_2^2}{\sigma_i^2}\right) \tag{14}$$

where σ_i are the kernel function calibration parameters.

3.2. Approximation of the Feature Map

Approximation to the feature maps ϕ_i can be obtained by means of an eigenvalue decomposition of the kernel matrices $\mathbf{K}_i^g(\mathbf{p})$ ([2]). The Fredholm integral equation of the first kind defines the eigenvalues and eigenfunctions of the kernel function, and the Nyström method ([23, 24]) approximates the integral by means of the sample average. Based on this approximation, an explicit expression for the j-th entry of the approximated feature maps is

$$\hat{\phi}_i^j(\mathbf{p}(\tilde{k})) = \frac{1}{\sqrt{\lambda_{i,j}}} \sum_{k=1}^{n} u_{i,j,k} K_i^g(\mathbf{p}(k),\mathbf{p}(\tilde{k})) \tag{15}$$

where $\hat{\phi}_i^j(\mathbf{p}(k)) : \mathbb{P} \to \mathbb{R}$, $\lambda_{i,j}$ is the j-th largest eigenvalue of the kernel matrix $\mathbf{K}_i^g(\mathbf{p})$, and $u_{i,j,k}$ is the k-th element of the j-th eigenvector of the same kernel function. Please note that $\lambda_{i,j}$ and $u_{i,j,k}$ are calculated in the training stage and saved, and they are used again to calculate (15) in the prediction stage. Also, note that using n training samples to compute the approximation of $\hat{\phi}_i(\mathbf{p}(\tilde{k}))$ will yield at most n components. Choosing a fixed-size $m_i \ll n$ for $i = 1,\dots,n_g$, the fixed-size approximation of the feature map is then

$$\hat{\phi}_i(\mathbf{p}(\tilde{k})) = \left[\hat{\phi}_i^1(\mathbf{p}(\tilde{k})),\dots,\hat{\phi}_i^{m_i}(\mathbf{p}(\tilde{k}))\right]^T \tag{16}$$

This finite-dimensional approximation can be used in the primal model representation to finally estimate the weight vectors $\boldsymbol{\omega}_i$ of dimension $m_i \ll n_H$.

3.3. LPV-ARX system identification in primal space

The discrete-time SISO LPV-ARX system described in the primal form in Eq. (4) can be rewritten in compact form:

$$\hat{y}(\tilde{k}) = \boldsymbol{\omega}^T \boldsymbol{\varphi}(\mathbf{p}(\tilde{k})) \tag{17}$$

where $\boldsymbol{\omega} \in \mathbb{R}^{n_g\cdot n_H}$ and $\boldsymbol{\varphi}(\mathbf{p}(\tilde{k})) = \mathrm{col}\big(\phi_1(\mathbf{p}(\tilde{k}))x_1(\tilde{k}),\dots,\phi_{n_g}(\mathbf{p}(\tilde{k}))x_{n_g}(\tilde{k})\big) \in \mathbb{R}^{n_g\cdot n_H}$.

Once the feature maps $\phi_i(\mathbf{p}(\tilde{k}))$ are estimated by using an approximation, either using the full sample or using a sparse approximation based on a subsample, the model can be estimated in the primal space.

Consider any m_i-dimensional approximation based on the subsample of the feature maps given by (16), where $m_i \ll n_H$. Defining $\boldsymbol{e} = [e(1),\dots,e(N)]^T$, the solution to the following optimization problem

$$\begin{aligned} &\arg\min_{\boldsymbol{\omega},\boldsymbol{e}} \mathcal{J}(\boldsymbol{\omega},\boldsymbol{e}) = \frac{1}{2}\boldsymbol{\omega}^T\boldsymbol{\omega} + \frac{\gamma}{2}\boldsymbol{e}^T\boldsymbol{e} \\ &\text{s.t. } e(k) = y(k) - \sum_{i=1}^{n_g} \boldsymbol{\omega}_i^T \hat{\phi}_i(\mathbf{p}(k)) x_i(k),\ k = 1,\dots,N \end{aligned} \tag{18}$$

can be written as the following linear system

$$\left(\hat{\boldsymbol{\Phi}}^T\hat{\boldsymbol{\Phi}} + \frac{\mathbf{I}}{\gamma}\right)\boldsymbol{\omega} = \hat{\boldsymbol{\Phi}}^T\mathbf{Y} \tag{19}$$

where γ is the regularization parameter, $\mathbf{Y} = [y(1),\dots,y(N)]^T$, $\hat{\boldsymbol{\Phi}}$ is feature matrix of dimension $N\times m$

$$\hat{\boldsymbol{\Phi}}^T = \begin{pmatrix} \hat{\phi}_1(\mathbf{p}(1))x_1(1) & \dots & \hat{\phi}_1(\mathbf{p}(N))x_1(N) \\ \vdots & \ddots & \vdots \\ \hat{\phi}_{n_g}(\mathbf{p}(1))x_{n_g}(1) & \dots & \hat{\phi}_{n_g}(\mathbf{p}(N))x_{n_g}(N) \end{pmatrix}, \tag{20}$$

$$m = \sum_{i=1}^{n_g} m_i, \tag{21}$$

and $\mathbf{I}$ is the $m\times m$ identity matrix. In the primal model representation, once the approximation feature maps $\hat{\phi}_i(\mathbf{p}(\tilde{k}))$ are estimated, then the weights vector $\boldsymbol{\omega}$ can be calculated solving the linear system (19). The approximation solution allows us to estimate a smaller weights vector $\boldsymbol{\omega}$ of dimension $m \ll n_g \cdot n_H$.

The output estimation follows from (17) (or, equivalently, (4)), where the feature map $\boldsymbol{\varphi}(\mathbf{p}(k))$ in (17) is calculated by the product between $\hat{\phi}_i(\mathbf{p}(k))$ and $x_i(k)$. The estimation of the model coefficients in the primal form in (3) can be calculated as:

$$\hat{a}_i(\mathbf{p}(\tilde{k})) = \boldsymbol{\omega}_i^T \hat{\phi}_i(\mathbf{p}(\tilde{k})), \quad i = 1,\dots,n_a \tag{22}$$

$$\hat{b}_i(\mathbf{p}(\tilde{k})) = \boldsymbol{\omega}_i^T \hat{\phi}_{\tilde{i}}(\mathbf{p}(\tilde{k})), \quad i = 1,\dots,n_b \tag{23}$$

where $\tilde{i} = n_a + 1,\dots,n_g$.

3.4. LPV-SS system identification in primal space

The discrete-time LPV-SS system in the primal form can be rewritten in a compact form (see Eqs. (8),(9)) similar to the LPV system in ARX form (see Eqs. (4) and (17)). So the learning procedure in the primal space is based on the approximation of five feature maps, i.e., $\phi_{xx}(\mathbf{p}(k)), \phi_{yx}(\mathbf{p}(k)), \phi_{xu}(\mathbf{p}(k)), \phi_{yu}(\mathbf{p}(k)), \phi_{xy}(\mathbf{p}(k))$ and the calculation of weights matrices $\mathbf{W_x}$ and $\mathbf{W_y}$. Similarly to the LPV-ARX case, the state-space matrices can be calculated by means of the weight matrices $\mathbf{W_x}$ and $\mathbf{W_y}$ and the approximated feature maps:

$$\tilde{\mathbf{A}}_e(\tilde{k}) = \mathbf{W_{xx}}\hat{\phi}_{xx}(\mathbf{p}(\tilde{k})) \tag{24}$$

$$\tilde{\mathbf{B}}_e(\tilde{k}) = \mathbf{W}_{\mathbf{xu}}\hat{\phi}_{xu}(\mathbf{p}(\tilde{k})) \tag{25}$$

$$\mathbf{C}_e(\tilde{k}) = \mathbf{W}_{\mathbf{yx}}\hat{\phi}_{yx}(\mathbf{p}(\tilde{k})) \tag{26}$$

$$\mathbf{D}_e(\tilde{k}) = \mathbf{W}_{\mathbf{yu}}\hat{\phi}_{yu}(\mathbf{p}(\tilde{k})) \tag{27}$$

$$\mathbf{K}_e(\tilde{k}) = \mathbf{W}_{\mathbf{xy}}\hat{\phi}_{xy}(\mathbf{p}(\tilde{k})) \tag{28}$$

4. LS-SVM IDENTIFICATION FOR LPV SYSTEMS: PRIMAL VERSUS DUAL SPACE

In this section, we briefly resume the identification procedure of LPV-ARX and LSV-SS models, which is described in [10, 11], in order to highlight the advantages and disadvantages of the primal and dual formulations.

4.1. LPV-ARX system identification in dual space

The dual model representation is proposed and investigated in [10, 13, 25, 26]. The output estimation in dual space is given by

$$\hat{y}(\tilde{k}) = \sum_{i=1}^{n_g}\left(\sum_{k=1}^{N}\left(\alpha(k)x_i(k)\phi_i^T(\mathbf{p}(k))\right)\phi_i(\mathbf{p}(\tilde{k}))x_i(\tilde{k})\right) \tag{29}$$

where $\boldsymbol{\alpha} = \left[\alpha(1),\ldots,\alpha(N)\right]^T \in \mathbb{R}^N$ is the vector of Lagrangian multipliers. Eq. (29) is the dual model representation of the LPV system. In the dual form, the goal of the identification process is to estimate the vector of Lagrangian multipliers $\boldsymbol{\alpha}$. Rewriting Eq. (29) in compact form, the dual space solution can be obtained by ridge regression ([10]).

Once the vector of Lagrangian multipliers $\boldsymbol{\alpha}$ has been calculated, the model coefficients are obtained as follows

$$\hat{a}_i(\mathbf{p}(\tilde{k})) = \sum_{k=1}^{N}\alpha(k)x_i(k)K_i^g(\mathbf{p}(k),\mathbf{p}(\tilde{k})),\ i=1,\ldots,n_a \tag{30}$$

$$\hat{b}_i(\mathbf{p}(\tilde{k})) = \sum_{k=1}^{N}\alpha(k)x_{\tilde{i}}(k)K_{\tilde{i}}^g(\mathbf{p}(k),\mathbf{p}(\tilde{k})),\ i=1,\ldots,n_b \tag{31}$$

where $\tilde{i} = n_a+1,\ldots,n_g$.

4.2. LPV-SS system identification in dual space

Similar to the case of LPV-ARX identification via LS-SVM, minimizing an LS-SVM-based cost function, the problem can be solved in the dual form by introducing the Lagrangian multipliers $\boldsymbol{\alpha}(k) \in \mathbb{R}^{n_x}$ and $\boldsymbol{\beta}(k) \in \mathbb{R}^{n_y}$. The output estimation in the dual space is given by

$$\hat{\mathbf{x}}(\tilde{k}+1) = \left(\sum_{k=1}^{N}\boldsymbol{\alpha}(k)\boldsymbol{\varphi}_x(\mathbf{p}(k))\right)\boldsymbol{\varphi}_x^T(\mathbf{p}(\tilde{k})) \tag{32}$$

$$\hat{\mathbf{y}}(\tilde{k}) = \left(\sum_{k=1}^{N}\boldsymbol{\beta}(k)\boldsymbol{\varphi}_y(\mathbf{p}(k))\right)\boldsymbol{\varphi}_y^T(\mathbf{p}(\tilde{k})) \tag{33}$$

where

$$\mathbf{W}_{\mathbf{x}} = \left(\sum_{k=1}^{N}\boldsymbol{\alpha}(k)\boldsymbol{\varphi}_x(\mathbf{p}(k))\right) \tag{34}$$

$$\mathbf{W}_{\mathbf{y}} = \left(\sum_{k=1}^{N}\boldsymbol{\beta}(k)\boldsymbol{\varphi}_y(\mathbf{p}(k))\right). \tag{35}$$

In the dual form, the goal of the identification process is to estimate the vectors of Lagrangian multipliers $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. Rewriting Equation (32) in compact form, the dual space solution can be obtained by ridge regression ([21]).

Once the vector of Lagrangian multipliers $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ have been calculated, the state-space matrices are obtained as follows

$$\tilde{\mathbf{A}}_e(\mathbf{p}(\tilde{k})) = \sum_{k=1}^{N}\boldsymbol{\alpha}(k)\mathbf{x}(k)^T K_{xx}^g(\mathbf{p}(k),\mathbf{p}(\tilde{k})) \tag{36}$$

$$\tilde{\mathbf{B}}_e(\mathbf{p}(\tilde{k})) = \sum_{k=1}^{N}\boldsymbol{\alpha}(k)\mathbf{u}(k)^T K_{xu}^g(\mathbf{p}(k),\mathbf{p}(\tilde{k})) \tag{37}$$

$$\mathbf{C}_e(\mathbf{p}(\tilde{k})) = \sum_{k=1}^{N}\boldsymbol{\beta}(k)\mathbf{x}(k)^T K_{yx}^g(\mathbf{p}(k),\mathbf{p}(\tilde{k})) \tag{38}$$

$$\mathbf{D}_e(\mathbf{p}(\tilde{k})) = \sum_{k=1}^{N}\boldsymbol{\beta}(k)\mathbf{u}(k)^T K_{yu}^g(\mathbf{p}(k),\mathbf{p}(\tilde{k})) \tag{39}$$

$$\mathbf{K}_e(\mathbf{p}(\tilde{k})) = \sum_{k=1}^{N}\boldsymbol{\alpha}(k)\mathbf{y}(k)^T K_{xy}^g(\mathbf{p}(k),\mathbf{p}(\tilde{k})) \tag{40}$$

where $K_{xx}^g(\mathbf{p}(k),\mathbf{p}(\tilde{k})) : \mathbb{P}\times\mathbb{P}\to\mathbb{R}$ is the $(k,\tilde{k})$-th entry of the kernel matrix $\mathbf{K}_{xx}^g(\mathbf{p})$. The same consideration holds for the other kernel matrices $\mathbf{K}_{xu}^g(\mathbf{p}), \mathbf{K}_{yx}^g(\mathbf{p}), \mathbf{K}_{yu}^g(\mathbf{p}), \mathbf{K}_{xy}^g(\mathbf{p})$.

4.3. Computational effort of LPV models in primal and dual space

4.3.1 Training stage in primal space

If no techniques are employed to reduce space and time complexity, and so the entire dataset is employed ($N = n = m$), the time complexity for the ridge regression (19) in the primal space is $\mathcal{O}\left(N^3\right)$, where N is the number of training datapoints, while the space complexity is $\mathcal{O}\left(N^2\right)$. In the case of a large dataset, it becomes prohibitive due to the size of the kernel matrix.

As highlighted in the previous sections, the proposed procedure in primal space involves two steps to reduce space and time complexity. The first step is the subset selection based on entropy, to obtain a dataset of dimension n starting from dimension N. Selecting the best n samples from N datapoints is too demanding for a large dataset, as an exhaustive search requires to explore $C_n(k)$ possible combinations. So, we employ the subset selection procedure proposed in [2], that is a non-exhaustive search that randomly selects a point from the training dataset and replaces randomly a point in the working set. If the entropy increases, then the point is accepted for the working

set. Such non-exhaustive search is characterized by $\mathcal{O}(n)$ space complexity, because at most $n+1 \ll N$ samples are loaded at the same time, while time complexity can be fixed in advance, as the search is not exhaustive. The time complexity for calculating each eigenvalue and eigenvector pair for a square matrix of size n is $\mathcal{O}\left(n^3\right)$. Hence, given a reduced dataset with n samples, the time complexity for calculating the feature map (16) is $\mathcal{O}(n^3)$ and the one for solving the linear system (19) is $\mathcal{O}\left(n^2N\right)$ ([2]), so the overall time complexity becomes $\mathcal{O}\left(n^2N\right)$ while the space complexity is $\mathcal{O}\left(n^2\right)$.

The second step to reduce computational complexity is the low-dimensional approximation of the feature maps, to obtain an additional reduction of the number of support vectors, from n to m. Definitively, the number of support vectors that are kept for learning is m, where $m \ll n \ll N$. So, the time complexity to calculate the approximated feature map becomes $\mathcal{O}\left(n^3\right)$ and the solution of the linear system (19) is further reduced from $\mathcal{O}\left(n^2N\right)$ (if only the first step is performed) to $\mathcal{O}\left(n^3+mnN\right) = \mathcal{O}\left(mnN\right)$, while the space complexity is reduced to $\mathcal{O}\left(m^2\right)$. Please note that the time complexity for calculating the m_i largest eigenvalues (and their respective eigenvectors) can be lower if dedicated algorithms are employed, such as the Lanczos algorithm ([27]).

Please note that the subset selection step is strictly necessary in case of large datasets. In fact, calculating the feature map with a very large dataset of size N is practically infeasible due to both space complexity and computational complexity. This holds even if we resort to an approximation of the feature map, i.e., even if only the largest eigenvalues and the corresponding eigenvectors of the feature map are to be calculated.

4.3.2 Training stage in dual space

In the dual space, N Lagrangian multipliers must be estimated by ridge regression. So, the time complexity for the ridge regression (19) in the dual space is $\mathcal{O}\left(N^3\right)$, where N is the number of training datapoints, while the space complexity is $\mathcal{O}\left(N^2\right)$. The subset selection in Section 3.1 could be performed also in the dual space, thus reducing the time complexity to $\mathcal{O}\left(n^3\right)$ and the space complexity to $\mathcal{O}\left(n^2\right)$.

4.3.3 Prediction stage in primal space

The prediction stage in primal space is resumed in Algorithm 1. The input signals are available or they are computed in advance (i.e., during the training stage), so they do not involve additional computation. In particular, let us compute in advance

$$\mu_{i,k} = \sum_{j=1}^{m_i} \frac{\omega_i^j u_{i,j,k}}{\sqrt{\lambda_{i,j}}} \;\forall i = 1,\ldots,n_g, \forall k = 1,\ldots,n. \quad (41)$$

The pseudocode for the prediction stage for fixed-size LS-SVM in primal space is reported in Algorithm 1.

Algorithm 1 Prediction in primal space

Input: $\mathbf{p}(k), \mathbf{p}(\tilde{k}), m, n_a, n_g, x_i(\tilde{k}), -1/\sigma_i^2, \mu_{i,k}$
$\forall i = 1,\ldots,n_g, \forall k = 1,\ldots,n$
for $i = 1,\ldots,n_g$ **do**
  if $i \le n_a$ **then** ▷ (14),(15),(16),(22),(41)
    $\hat{a}_i(\mathbf{p}(\tilde{k})) \leftarrow \sum_{k=1}^{n} \left[\mu_{i,k} \exp\left(-\frac{\|\mathbf{p}(k)-\mathbf{p}(\tilde{k})\|_2^2}{\sigma_i^2}\right)\right]$
  else ▷ (14),(15),(16),(23),(41)
    $\hat{b}_{i-n_a}(\mathbf{p}(\tilde{k})) \leftarrow \sum_{k=1}^{n} \left[\mu_{i,k} \exp\left(-\frac{\|\mathbf{p}(k)-\mathbf{p}(\tilde{k})\|_2^2}{\sigma_i^2}\right)\right]$
  end if
end for
$\hat{y}(\tilde{k}) \leftarrow \sum_{i=1}^{n_a} \hat{a}_i(\mathbf{p}(\tilde{k}))x_i(\tilde{k}) + \sum_{i=n_a+1}^{n_g} \hat{b}_{i-n_a}(\mathbf{p}(\tilde{k}))x_i(\tilde{k})$

The computation of each $K_i^g(\mathbf{p}(k), \mathbf{p}(\tilde{k}))$ from (14) requires one exponentiation (n_e FLoating point Operations Per Seconds (FLOPs)), one vector subtraction (n_p FLOPs), one scalar product ($2n_p - 1$ FLOPs), and one multiplication (1 FLOP), for a total of $n_e + 3n_p$ FLOPs. The number $n_e \ge 1$ of FLOPs for each exponentiation depends on the actual CPU. The expressions for $\hat{a}_i$ and $\hat{b}_i$ are obtained by substituting (14), (15), and (16) in both (22) and (23). The online computation boils down to calculating $K_i^g(\mathbf{p}(k), \mathbf{p}(\tilde{k}))$ for $k = 1,\ldots,n$ (i.e., $n(n_e+3n_p)$ FLOPs), plus a weighted sum with pre-computed weights ($2n-1$ FLOPs). The final prediction is a scalar product ($2n_g - 1$ FLOPs). The total number of FLOPs for the prediction in primal stage is then:

$$\begin{aligned}&\sum_{i=1}^{n_g} [n(n_e+3n_p)+2n-1] + 2n_g - 1\\ &= nn_g(3n_p+n_e+2) + n_g - 1\end{aligned} \quad (42)$$

where (21) has been exploited.

4.3.4 Prediction stage in dual space

For comparison, the prediction stage in dual space is resumed in Algorithm 2. As in the case of primal space, the computation of each $K_i^g(\mathbf{p}(k), \mathbf{p}(\tilde{k}))$ from (14) requires $n_e + 3n_p$ FLOPs. Then, the weighted sum is, essentially, a scalar product ($2N-1$ FLOPs). The computation of the final prediction is a scalar product as well ($2n_g - 1$ FLOPs). The total number of FLOPs for the prediction in primal space is then:

$$\begin{aligned}&\sum_{i=1}^{n_g} [N(n_e+3n_p)+2N-1] + 2n_g - 1\\ &= Nn_g(3n_p+n_e+2) + n_g - 1\end{aligned} \quad (43)$$

As already stated, the subset selection in Section 3.1 could be performed also in the dual space. In this case, (43) be-

Algorithm 2 Prediction in dual space

Input: $\mathbf{p}(k), \mathbf{p}(\tilde{k}), m, n_a, n_g, x_i(\tilde{k}), -1/\sigma_i^2, \alpha(k)x_i(k)$
$\forall i = 1,\ldots,n_g, \forall k = 1,\ldots,m$
for $i = 1,\ldots,n_g$ **do**
if $i \leq n_a$ **then** ▷ (14),(30)

$$\hat{a}_i(\mathbf{p}(\tilde{k})) \leftarrow \sum_{k=1}^{N}\left[\alpha(k)x_i(k)\exp\left(-\frac{\|\mathbf{p}(k)-\mathbf{p}(\tilde{k})\|_2^2}{\sigma_i^2}\right)\right]$$

else ▷ (14),(31)

$$\hat{b}_{i-n_a}(\mathbf{p}(\tilde{k})) \leftarrow \sum_{k=1}^{N}\left[\alpha(k)x_i(k)\exp\left(-\frac{\|\mathbf{p}(k)-\mathbf{p}(\tilde{k})\|_2^2}{\sigma_i^2}\right)\right]$$

end if
end for

$$\hat{y}(\tilde{k}) \leftarrow \sum_{i=1}^{n_a}\hat{a}_i(\mathbf{p}(\tilde{k}))x_i(\tilde{k}) + \sum_{i=n_a+1}^{n_g}\hat{b}_{i-n_a}(\mathbf{p}(\tilde{k}))x_i(\tilde{k})$$

comes

$$\begin{aligned}&\sum_{i=1}^{n_g}[n(n_e+3n_p)+2N-1]+2n_g-1\\&= nn_g(3n_p+n_e+2)+n_g-1\end{aligned} \tag{44}$$

Comparing (44) with (42), we highlight that exactly the same number of FLOPs are needed in the prediction stage for both the primal and the dual approach.

4.4. Comparison with KPCR for LPV System Identification

In the following, we compare the proposed primal solution, the solution based on KPCR ([3]), and the solution based on dual space ([10]). KPCR is a special case of LS-SVM estimation based on primal space. In particular, in KPCR based solution only one $\mathbf{K}_1^g(\mathbf{p})$ kernel matrix is estimated (see Eq. (24) in ([3]), where $\mathbf{K}_1^g(\mathbf{p})$ is evaluated as Hadamard product of two kernel matrices of which one is a linear kernel), while in the primal or dual space formulation of LS-SVM based solution, one kernel matrix $\mathbf{K}_i^g(\mathbf{p})$ can be evaluated differently for each σ_i. Indeed, in [3], the parameters, differently from Eq. (4), are estimated as:

$$\begin{aligned}a_i(\mathbf{p}(k)) &= \sum_{v=1}^{n_H}\omega_{i,v}\phi_v(\mathbf{p}(k))\\&= \omega_i^T\phi(\mathbf{p}(k)), \quad i=1,\ldots,n_a\\b_i(\mathbf{p}(k)) &= \sum_{v=1}^{n_H}\omega_{\tilde{i},v}\phi_v(\mathbf{p}(k))\\&= \omega_{\tilde{i}}^T\phi(\mathbf{p}(k)), \quad i=1,\ldots,n_b\end{aligned} \tag{45}$$

where $\tilde{i} = n_a+1,\ldots,n_g$.

In [3] the authors considered that the nonlinear functions ϕ_i are unknown and the output predictor equation is based on the kernel matrices and the matrix $\hat{\mathbf{\Lambda}}$ (see Eq. (41) in [3]). The $\hat{\mathbf{\Lambda}}$ is evaluated similarly to the ridge regression solution in dual space (in KPCR based solution, the regularization parameter γ is omitted) and, finally, the prediction is evaluated in the dual space. Differently from [3] and [10], in this work, the approximation functions $\hat{\phi}_i(\mathbf{p}(k))$ are evaluated and they can be used directly for the prediction (i.e., to calculate $\hat{y}(\tilde{k})$ in the primal space).

5. SIMULATION RESULTS

In the following, the illustrative examples proposed in ([3, 10, 21]) are reported, as well as the parallel Wiener-Hammerstein system and the Silverbox system ([19, 20]).

5.1. Numerical example (LPV-ARX)

In this section, the following simulated SISO system is considered

$$\begin{aligned}y(k)+a_1(p(k))y(k-1) &= b_1(p(k))u(k-1)\\&\quad +b_2(p(k))u(k-2)+e(k)\end{aligned} \tag{46}$$

with

$$a_1(p(k)) = 0.1\frac{\sin(\pi^2 p(k))}{\pi^2 p(k)} \tag{47}$$

$$b_1(p(k)) = \begin{cases}-0.5 & p(k) < -0.5\\ p(k) & -0.5 \leq p(k) \leq 0.5\\ 0.5 & p(k) > 0.5\end{cases} \tag{48}$$

$$b_2(p(k)) = -0.2p(k)^2 \tag{49}$$

The input and scheduling signals, $u(k)$ and $p(k)$, were zero mean independent white noise sequences uniformly distributed in the interval [-1,1] and the equation error, $e(k)$, was a zero mean Gaussian white noise with a variance determined by the Signal-to-Noise Ratio (SNR) level.

The aim of this example is to compare the results of LPV-ARX estimation in the primal space with respect to LPV-ARX estimation in the dual space ([10]) and LPV-ARX estimation based on Kernel Principal Component Regressor ([3]).

To investigate the performance under fairly severe noise conditions, the SNR is set to 10 dB, and 1500 input-output data points were generated. The first 750 data points were the training data, and the other half was used for testing. To quantify the model quality, the Best Fit Rate (BFR) is considered to compare the three algorithms:

$$\text{BFR} = 100\% \cdot \max\left(0, 1-\frac{\|y-\hat{y}\|_2^2}{\|y-\bar{y}\|_2^2}\right) \tag{50}$$

where y is the output, $\bar{y}$ is the mean of y, and $\hat{y}$ is the simulated output with the estimated models.

Using the gathered data sets, the different approaches have been applied by using RBF kernels for $\mathbf{K}_1^g(\mathbf{p})$, $\mathbf{K}_2^g(\mathbf{p})$, $\mathbf{K}_3^g(\mathbf{p})$ having σ_1=σ_2=σ_3=0.7. The regularization parameter, based on trial-and-error, has been tuned to $\gamma = 10^4$.

We have performed 100 simulations employing the same data points, but considering different generations

of random series $u(k)$, $p(k)$, $e(k)$. The results obtained in terms of BFR by using the identification in the primal space are 97.53% $\pm$ 0.172%. Figure 1 displays the real and estimated values of the parameter $a_1(p(k))$ for SNR=10 dB. Solid blue line is the real value, dashed red line is the average of 100 simulations and dash-dotted yellow/violet line is the average plus/minus standard deviation.

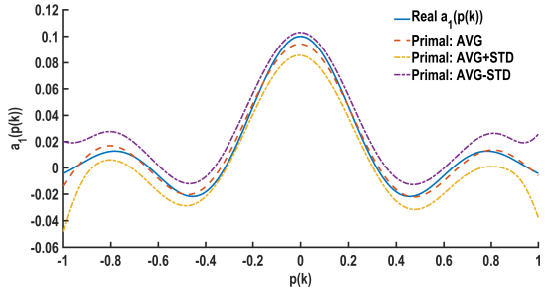


Fig. 1. LPV-ARX numerical example: parameter $a_1(p(k))$, SNR=10 dB.

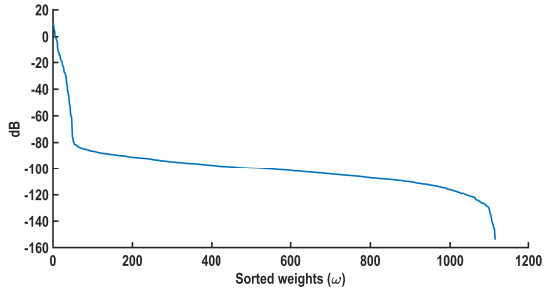


Fig. 2. LPV-ARX numerical example: amplitude in dB of sorted weights (ω).

Figure 2 shows the amplitudes (i.e., absolute value) in dB of sorted largest weights ω by using the whole dataset for identification. We highlight that they are sparse, hence only a few weights can be kept while keeping a high BFR.

Figure 3 shows the effect of varying the number of selected support vectors m by using the fixed-size procedure shown in subsection 3.1. Even in this case, we note that the increase in BFR is large at the beginning, when the first support vectors are added, while increasing m further (e.g., from 100 to 200) entails a minor increase in BFR. The solid blue line is the average of 100 simulations and solid black lines represent the average plus/minus standard deviation.

Finally, we compare the proposed primal solution, the solution based on KPCR ([3]) and the solution based on dual space ([10]). As we expected, all three solutions give the same results in terms of BFR since in this particular case σ_1=σ_2=σ_3 and as long as the regularization parameter

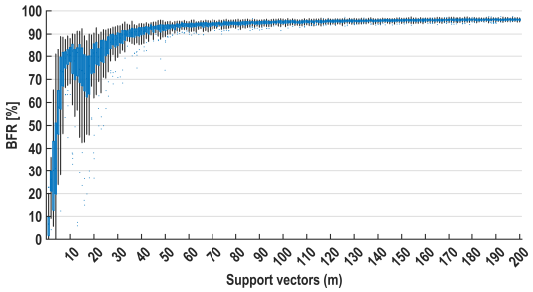


Fig. 3. LPV-ARX numerical example: BFR with respect to the different numbers of selected support vectors (m).

γ is considered in the KPCR formulation. Table 1 displays the minimum, average and maximum BFR values of the 100 simulations. This table shows that all estimated models accurately simulated the system, revealing the same high BFR.

Table 1. BFRs of LPV-ARX numerical example.

Method	BFR [%] Min-Avg-Max	BFR [%] Std
LS-SVM (primal)	97.08-97.53-97.95	0.17
LS-SVM (dual)	97.08-97.53-97.95	0.17
KPCR	97.08-97.53-97.95	0.17

5.2. Silverbox system

The Silverbox system can be seen as an electronic implementation of the Duffing oscillator. It is built as a second-order linear time-invariant system with a third-degree polynomial static nonlinearity around it in feedback ([20]).

The dataset consists of 131,072 samples. Since the system is a SISO model, in this work, we consider as a scheduling parameter the previous output, i.e., the output delayed by a single sampling time interval, such that the result of the identification is a quasi-LPV (qLPV) model.

The working strategy for using the data in terms of training, validation and testing is:

- Training and validation samples: data points from 40,001 to end. Models are estimated using this part of the data. Bayesian optimization algorithm and 10-fold Cross Validation (CV) is used to set the hyperparameters (i.e., in this specific case, σ, γ, n_a, n_b). The BFR on the validation set is computed for hyperparameter selection.
- Testing samples: the "head of the arrow", data points from 1 to 40,000. After defining the optimal model

(using the validation BFR), the prediction for the test set is done.

The hyperparameters to be set by the validation stage are σ, γ, n_a, n_b. In this work, it was considered $\sigma_i = \sigma, \forall i$ and the maximum values of n_a, n_b equal to 3. The identification is repeated by changing the number of selected support vectors m and 20 Monte Carlo simulations are carried out where the subset selection procedure is repeated by changing the random seed. Figure 4 shows the results of LPV models identified in the primal space and, in particular, the statistic of BFR with respect to the number of the support vectors $m = 1,\ldots,20$. The results for $m = 21,\ldots,200$ are comparable and thus omitted. The BFR is evaluated by means of a one-step prediction and the LPV models are identified as prediction models. Moreover, the figure shows the comparison between the selection procedure in subsection 3.1 and a trivial selection solution based on random sample selection. The dashed line is a baseline, i.e., the BFR obtained by keeping constant the output value at instant $\tilde{k}$ to predict the value to the instants $\tilde{k}+1$, i.e., $\hat{y}(\tilde{k}+1) = y(\tilde{k})$. The selection procedure based on entropy gives better results in terms of average BFR and smaller variance with respect to the trivial random selection. It is worth noting that using the entropy-based subset selection methodology increases the training time with respect to the random-based subset selection methodology but improves the BFR. Please note that the computational complexity to perform a single prediction remains the same: the additional computational effort is limited to the training state. Figure 5 shows the identified parameters considering the support vectors size $m = 200$.

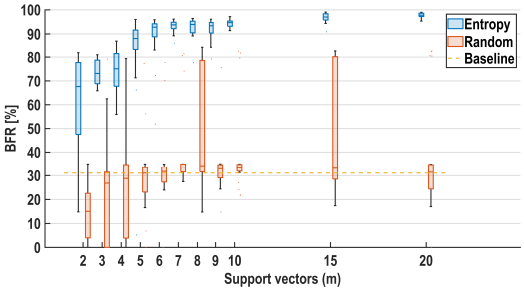


Fig. 4. LPV-ARX system identification of Silverbox system in the primal space: BFR with respect to the different numbers of selected support vectors (m).

To compare the LPV identification by LS-SVM in the primal space with a conventional linear time-invariant (LTI) identification, we also report a discrete-time state-space model identification. The LTI identification is performed using the System Identification toolbox in Matlab, considering the Prediction-Error Minimization (PEM) algorithm to update the parameters of the initial model. All training and validation data are used for LTI model identification. Table 2 shows the testing results for different model orders. Please note the BFR of LTI identification is worse in any case with respect to LPV identification by LS-SVM, even when a few support vectors m are employed in the latter.

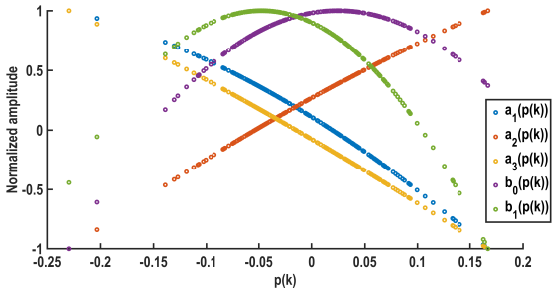


Fig. 5. LPV-ARX system identification of Silverbox system in the primal space: identified parameters $a_1(p(k))$, $a_2(p(k))$, $a_3(p(k))$, $b_0(p(k))$ and $b_1(p(k))$ with $m = 200$.

Table 2. State-space model identification of Silverbox system: BFR with respect to the different model orders.

Model order	BFR [%]
1	10.01
2	73.56
3	73.56
4	73.66
5	73.60

5.3. Parallel Wiener–Hammerstein system

A Parallel Wiener-Hammerstein (PWH) system is obtained by connecting multiple Wiener-Hammerstein systems in parallel. Each parallel branch contains a static nonlinearity that is sandwiched in between two LTI blocks ([19]).

The dataset consists of a total of 3,473,408 samples. Since the system is a SISO model, we consider as a scheduling parameter the previous output (as done in subsection 5.2), so we obtain a qLPV model. Signals are sampled at 78 kHz and the measured input and output signals contain 16,384 measured samples per period. Twenty independent random phase realizations of the multisines are used at each input level. The input signal is applied at 5 different RMS values that are linearly distributed between 100 mV and 1 V.

The working strategy for using the data in terms of training, validation and testing is:

- Training and validation samples: a total of 3,276,800 data points. Models are estimated using this part of the data. Bayesian optimization algorithm and 10-fold CV are used to set the hyperparameters (i.e., in this specific case, σ, γ, n_a, n_b). The BFR on the validation set is computed for hyperparameter selection.
- Testing samples: a total of 196,608 data points. The dataset presents signals generated by an input signal that is a filtered Gaussian noise with an envelope that grows linearly over time. Here, these data are used in testing to assess the model quality over a broad amplitude range of the input in one signal. After defining the optimal model (using the validation BFR), the prediction for the test set is done.

The hyperparameters to be set by the validation stage are σ, γ, n_a, n_b. In this work, it was considered $\sigma_i = \sigma, \forall i$ and the maximum values of n_a, n_b equal to 3. The identification is repeated by changing the number of selected support vectors m and 20 Monte Carlo simulations are carried out where the subset selection procedure is repeated by changing the random seed. The figure 6 shows the results of LPV models identified in the primal space and, in particular, the statistic of BFR with respect to the number of selected support vectors $m = 1, \ldots, 20$. The results for $m = 21, \ldots, 200$ are comparable and thus omitted. The BFR is evaluated by means of a one-step prediction and the LPV models are identified as prediction models. Figure 6 shows the comparison between the selection procedure based on entropy and the trivial selection solution based on random sample selection. The selection procedure based on entropy gives better results in terms of average BFR and smaller variance mostly in the case of higher m values.

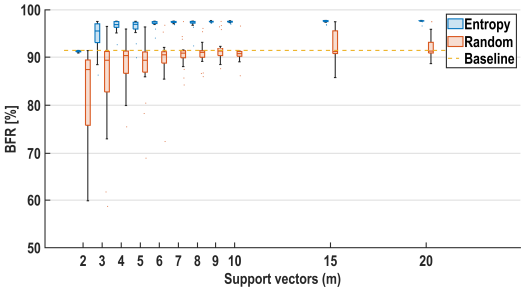


Fig. 6. LPV-ARX system identification of PWH system in the primal space: BFR with respect to the different numbers of selected support vectors (m).

Figure 7 shows the identified parameters considering $m = 200$. As in the previous example, we compare the results with a LTI discrete-time state-space model, identified by the System Identification Toolbox from MathWorks© MATLAB suite. All training and validation data are used for LTI model identification. Table 3 shows the testing

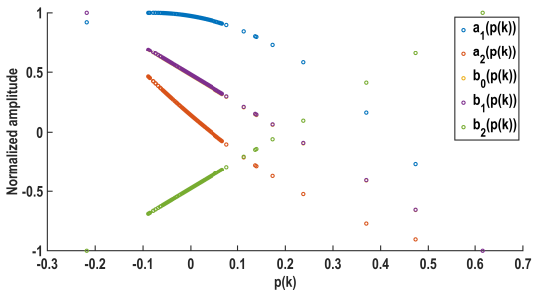


Fig. 7. LPV-ARX system identification of PWH system in the primal space: identified parameters $a_1(p(k))$, $a_2(p(k))$, $b_0(p(k))$, $b_1(p(k))$ and $b_2(p(k))$ with $m = 200$.

results for different model orders. The highest BFR is achieved for the fifth order. Again, the results are worse with respect to LPV identification by LS-SVM.

Table 3. State-space model identification of PWH system: BFR with respect to the different model orders.

Model order	BFR [%]
1	43.34
2	43.93
3	80.55
4	83.78
5	83.79

5.4. Numerical example (LPV-SS)

The aim of this example is to compare the results of LPV-SS estimation in the primal space with respect to LPV-SS estimation in the dual space. Consider the discrete-time state-space data-generating system described in [21] by the following matrices

$$\mathbf{A}(p(k)) = \begin{bmatrix} \text{sat}(p(k)) & 1 & 0 & 0 \\ \frac{1}{2} & \frac{p(k)^3}{8} & \frac{4}{10} & \frac{1}{5} \\ \frac{3}{10} & 0 & \frac{p(k)^2}{5} & \frac{1}{8} \\ 0 & 0 & \frac{1}{2} & \frac{1}{5} \end{bmatrix} \tag{51}$$

$$\mathbf{B}(p(k)) = \begin{bmatrix} \frac{p(k)^4}{5} \\ 0 \\ \frac{1}{5} \\ 0 \end{bmatrix} \tag{52}$$

$$\mathbf{C}(p(k)) = \begin{bmatrix} \frac{p(k)^2}{5} & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{53}$$

$$\mathbf{K}(p(k)) = \begin{bmatrix} \frac{\tanh(p(k))}{3p(k)} & 0 \\ 0 & 0 \\ 0 & \sin(2\pi p(k)) + \cos(2\pi p(k)) \\ 0 & 1 \end{bmatrix} \tag{54}$$

where $\text{sat}(p(k))$ is a saturation function with limits at $\pm$ 0.5 and unity slope. The data-generating system with initial condition $x_0 = [0000]^T$ has been simulated with uniformly distributed input signal $u(k)$ in the interval $[-1, 1]$, $p(k) = \sin(0.3k)$ and a zero mean Gaussian white noise $e(k)$ with a variance determined by the SNR. The noise variance has been chosen to guarantee a SNR of 20 dB. For this purpose, 1,100 input-output data points were generated. The first 600 data points were the training and validation data, and the remaining 500 were used for testing. Bayesian optimization algorithm and 10-fold CV are used to set the hyperparameters. The BFR on the validation set is computed for hyperparameter selection. The identification is repeated by changing the number of selected support vectors m and 20 Monte Carlo simulations are carried out where the subset selection procedure is repeated by changing the random seed. Figure 8 shows the results of LPV-SS models identified in the primal space, and in particular, the BFR statistics related to the state prediction with respect to the number of selected support vectors m. The BFR is evaluated by means of a one-step prediction and the LPV models are identified as simulation models.

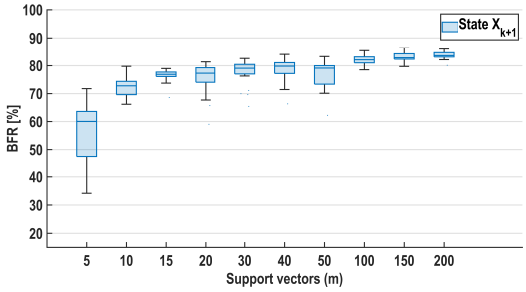


Fig. 8. LPV-SS system identification of numerical example in the primal space: BFR related to the state prediction with respect to the different numbers of selected support vectors (m).

Figure 9 shows the results of LPV-SS models identified in the primal space, and in particular, the BFR statistics related to the output with respect to the number of selected support vectors m.

Regarding the fitting, the results in terms of BFR are better than those reported in [21], where the authors considered 800 datapoints for training and the last 300 datapoints for testing, obtaining the best results with a BFR of $86.31\% \pm 0.022\%$. In contrast, in our approach, we have obtained a BFR up to $90.49\% \pm 0.53\%$ in the primal space, as reported in Table 4. It is worth highlighting that in our case, we have used the measured states, whereas in [21], the states were estimated. As for the speed of identification, in the method from [21], the inversion of two square matrices (see Eqs. (31a) and (31b) in [21]) of dimensions $N \cdot n_x \times N \cdot n_x$ and $N \cdot n_y \times N \cdot n_y$ is required. This leads to a space complexity of $\mathcal{O}((N \cdot n_x)^2 + (N \cdot n_y)^2)$ and a time complexity of $\mathcal{O}((N \cdot n_x)^3 + (N \cdot n_y)^3)$. As a result, the approach based on the dual space formulation becomes practically infeasible with a very large dataset of size N due to both space and computational complexity. On the other hand, our proposed approach based on the primal space formulation requires calculating the eigenvalues of 5 kernel matrices, namely $\mathbf{K}^g_{xx}(\mathbf{p}), \mathbf{K}^g_{xu}(\mathbf{p}), \mathbf{K}^g_{yx}(\mathbf{p}), \mathbf{K}^g_{yu}(\mathbf{p}), \mathbf{K}^g_{xy}(\mathbf{p})$. The space complexity for this approach is $\mathcal{O}(5 \cdot n^2)$ and the time complexity is $\mathcal{O}(5 \cdot n^3)$, with $n \ll N$, which makes it preferable for large datasets.

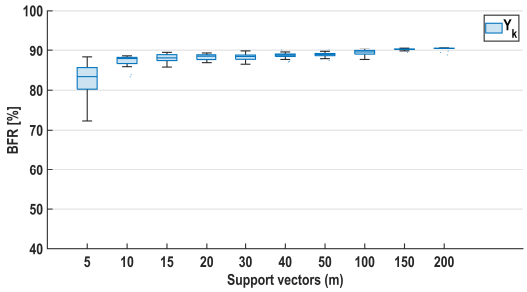


Fig. 9. LPV-SS system identification of numerical example in the primal space: BFR related to the output with respect to the different numbers of selected support vectors (m).

Table 4. LPV-SS numerical example in the primal space: BFR related to the output with respect to the different numbers of selected support vectors (m).

Support vectors (m)	BFR [%] Avg	BFR [%] Std
10	87.27	1.46
20	88.26	0.75
50	88.89	0.76
100	89.43	0.72
200	90.49	0.53

6. CONCLUSIONS

The main contribution of this paper lies in formulating a LS-SVM-based method for the identification of LPV-ARX and LPV-SS models in the primal space. Firstly, the problem formulation of LPV-ARX and LPV-SS identifica-

tion in the primal and the dual space has been shown and compared. Secondly, the LPV identification has been designed to handle very large data sets, computing a sparse approximation of feature maps by using only a subsample of selected support vectors from the entire data set.

As for the training stage, training a primal or a dual model with a large dataset is infeasible, as the time complexity is $\mathcal{O}\left(N^3\right)$ and space complexity is $\mathcal{O}\left(N^2\right)$. Training in the primal space is easier because both subset selection and feature map approximation can be performed, so that time complexity in primal space can be reduced to $\mathcal{O}\left(mnN\right)$, in contrast to $\mathcal{O}\left(n^2N\right)$ of the dual approach. As for the space complexity, it is $\mathcal{O}\left(m^2\right)$ for the primal approach, instead of $\mathcal{O}\left(n^2\right)$ of the dual approach. As $m \ll n$, both the time and the space complexity in the primal space are lower in the training stage.

As for the prediction stage, instead, the primal and the dual approach are equivalent because they require the same number of FLOPs. This equivalence holds when the training subset size is consistent for both the primal and dual methods, otherwise, the larger the training subset, the higher the computational effort (i.e., without distinction between primal and dual).

The advantage of the solution in the primal space is that one can reduce the computational time and space in the training stage in nonlinear estimation problems while keeping comparable performances. This approach has been extensively tested in numerical examples and two nonlinear benchmarks, i.e., the parallel Wiener-Hammerstein system and the Silverbox system.

In future work, the authors will investigate the possibility to consider random feature expansion such as Random Kitchen Sinks ([28]) or Fastfood ([29]), which is a scheme to approximate Gaussian kernels of the kernel regression algorithm for large data set in a computationally efficient way. Random feature expansion is more practical for applications that have large training sets. Finally, the proposed solution will be extended to the case of discontinuous scheduling variables where the introduction of discontinuous kernel functions could be needed.

CONFLICT OF INTEREST

The authors declare that there is no competing financial interest or personal relationship that could have appeared to influence the work reported in this paper.

REFERENCES

[1] M. Espinoza, K. Pelckmans, L. Hoegaerts, J. A. Suykens, and B. De Moor, "A comparative study of LS-SVM's applied to the silver box identification problem," IFAC Proceedings Volumes, vol. 37, no. 13, pp. 369–374, 2004.

[2] K. De Brabanter, P. Dreesen, P. Karsmakers, K. Pelckmans, J. De Brabanter, J. Suykens, and B. De Moor, "Fixed-size LS-SVM applied to the Wiener-Hammerstein benchmark," IFAC Proceedings Volumes, vol. 42, no. 10, pp. 826–831, 2009.

[3] P. L. dos Santos and T. A. Perdicoúlis, "A kernel principal component regressor for LPV system identification," IFAC-PapersOnLine, vol. 52, no. 28, pp. 7–12, 2019.

[4] L. Cavanini, L. Ciabattoni, F. Ferracuti, E. Marchegiani, and A. Monteriù, "A comparative study of driver torque demand prediction methods," IET Intelligent Transport Systems, 2022.

[5] S. Ijaz, M. T. Hamayun, H. Anwaar, L. Yan, and M. K. Li, "Lpv modeling and tracking control of dissimilar redundant actuation system for civil aircraft," International Journal of Control, Automation and Systems, vol. 17, pp. 705–715, 2019.

[6] F. Ma, J. Li, L. Wu, and D. Yuan, "Tensor product based polytopic lpv system design of a 6-dof multi-strut platform," International Journal of Control, Automation and Systems, vol. 20, no. 1, pp. 137–146, 2022.

[7] J. Che, Y. Zhu, M. V. Basin, and D. Zhou, "Active fault-tolerant control for discrete-time markov jump lpv systems via time-varying hidden markov model approach," International Journal of Control, Automation and Systems, vol. 20, no. 6, pp. 1785–1799, 2022.

[8] J. A. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. P. Vandewalle, Least squares support vector machines. World scientific, 2002.

[9] L. Cavanini, G. Ippoliti, and E. F. Camacho, "Model predictive control for a linear parameter varying model of an uav," Journal of Intelligent & Robotic Systems, vol. 101, no. 3, pp. 1–18, 2021.

[10] R. Tóth, V. Laurain, W. X. Zheng, and K. Poolla, "Model structure learning: A support vector machine approach for LPV linear-regression models," in 2011 50th IEEE Conference on Decision and Control and European Control Conference. IEEE, 2011, pp. 3192–3197.

[11] S. Z. Rizvi, J. Mohammadpour, R. Tóth, and N. Meskin, "A kernel-based approach to MIMO LPV state-space identification and application to a nonlinear process system," IFAC-PapersOnLine, vol. 48, no. 26, pp. 85–90, 2015.

[12] M. Mejari, D. Piga, and A. Bemporad, "Regularized least square support vector machines for order and structure selection of LPV-ARX models," in 2016 European Control Conference (ECC), 2016, pp. 1649–1654.

[13] D. Piga and R. Tóth, "LPV model order selection in an LS-SVM setting," in 52nd IEEE Conference on Decision and Control, 2013, pp. 4128–4133.

[14] R. Duijkers, R. Tóth, D. Piga, and V. Laurain, "Shrinking complexity of scheduling dependencies in LS-SVM based LPV system identification," in 53rd IEEE Conference on Decision and Control. IEEE, 2014, pp. 2561–2566.

[15] L. Cavanini, F. Ferracuti, S. Longhi, E. Marchegiani, and A. Monteriù, "Sparse Approximation of LS-SVM for LPV-ARX Model Identification: Application to a Powertrain Subsystem," in 2020 AEIT International Conference of Electrical and Electronic Technologies for Automotive (AEIT AUTOMOTIVE), 2020, pp. 1–6.

[16] L. Cavanini, F. Ferracuti, S. Longhi, and A. Monteriù, "LS-SVM for LPV-ARX Identification: Efficient Online Update by Low-Rank Matrix Approximation," in 2020 International Conference on Unmanned Aircraft Systems (ICUAS), 2020, pp. 1590–1595.

[17] M. Mejari, B. Mavkov, M. Forgione, and D. Piga, "Direct identification of continuous-time LPV state-space models via an integral architecture," Automatica, vol. 142, p. 110407, 2022.

[18] J. A. Suykens, C. Alzate, and K. Pelckmans, "Primal and dual model representations in kernel-based learning," Statistics Surveys, vol. 4, pp. 148–183, 2010.

[19] M. Schoukens, A. Marconato, R. Pintelon, G. Vandersteen, and Y. Rolain, "Parametric identification of parallel Wiener–Hammerstein systems," Automatica, vol. 51, pp. 111–122, 2015.

[20] T. Wigren and J. Schoukens, "Three free data sets for development and benchmarking in nonlinear system identification," in 2013 European Control Conference (ECC), 2013, pp. 2933–2938.

[21] 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.

[22] M. Girolami, "Orthogonal series density estimation and the kernel eigenvalue problem," Neural computation, vol. 14, no. 3, pp. 669–688, 2002.

[23] E. J. Nyström, "Über die praktische Auflösung von Integralgleichungen mit Anwendungen auf Randwertaufgaben," Acta Mathematica, vol. 54, pp. 185–204, 1930.

[24] C. Williams and M. Seeger, "Using the Nyström method to speed up kernel machines," Advances in neural information processing systems, vol. 13, 2000.

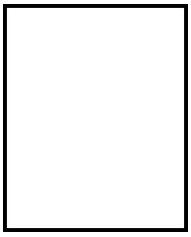
[25] R. Tóth, H. Hjalmarsson, and C. R. Rojas, "Order and structural dependence selection of LPV-ARX models revisited," in 2012 IEEE 51st IEEE Conference on Decision and Control (CDC), Dec 2012, pp. 6271–6276.

[26] R. Tóth, P. Heuberger, and P. Van den Hof, "LPV system identification using series expansion models," in Linear parameter-varying system identification: new developments and trends. World Scientific, 2012, pp. 259–294.

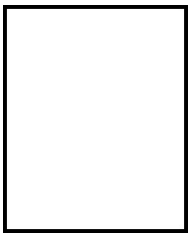
[27] J. Chen and Y. Saad, "Lanczos vectors versus singular vectors for effective dimension reduction," IEEE Transactions on Knowledge and Data Engineering, vol. 21, no. 8, pp. 1091–1103, 2008.

[28] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," Advances in neural information processing systems, vol. 20, 2007.

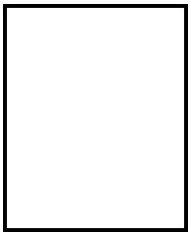
[29] Q. Le, T. Sarlós, and A. Smola, "Fastfood: Approximating kernel expansions in loglinear time," in Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28, ser. ICML'13. JMLR.org, 2013, p. III–244–III–252.



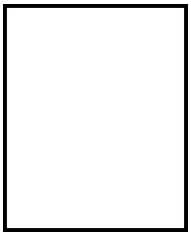
Luca Cavanini received his Ph.D. degree in automation, information and management engineering from Università Politecnica delle Marche, Ancona, Italy, in 2018. He works as a Technical Consultant at Industrial Systems and Control Ltd. His activityincludes model predictive control, autonomous mobile robotics and artificial intelligence, and machine learning techniques for control systems.



Riccardo Felicetti received his Master's Degree cum laude in Computer and Automation Engineering and the Ph.D. degree cum laude in Information Engineering from Università Politecnica delle Marche, in 2016 and 2021, respectively. Since 2019, he has been a postdoctoral research fellow with Università Politecnica delle Marche. His main research interests are fault detection and diagnosis, fault tolerant control, and optimization with applications to unmanned vehicles and energy management systems.



Francesco Ferracuti received his Ph.D. degree in automation, information and management engineering from Università Politecnica delle Marche, Ancona, Italy, in 2014. He is a research at Università Politecnica delle Marche. His research interests include model-based and data-driven fault diagnosis, signal processing, statistical pattern recognition, and machine learning and their applications in industry.



Andrea Monteriù received his M.Sc. degree cum laude in Electronic Engineering and the Ph.D. degree in Artificial Intelligence Systems from Università Politecnica delle Marche, Italy, in 2003 and 2006, respectively. Currently, he is an associate professor at Università Politecnica delle Marche. His research interests mainly focus on the areas of fault diagnosis and fault tolerant control applied on robotic, and unmanned and artificial intelligent systems.