2014 European Control Conference (ECC)
June 24-27, 2014. Strasbourg, France

Parameter Set-mapping using Kernel-based PCA for Linear Parameter-varying Systems*

Syed Z. Rizvi†, Javad Mohammadpour†, Roland Tóth‡, and Nader Meskin§

Abstract— **This paper proposes a method for reduction of scheduling dependency in linear parameter-varying (LPV) systems. In particular, both the dimension of the scheduling variable and the corresponding scheduling region are shrunk using kernel-based principal component analysis (PCA). Kernel PCA corresponds to linear PCA that is performed in a high-dimensional feature space, allowing the extension of linear PCA to nonlinear dimensionality reduction. Hence, it enables the reduction of complicated coefficient dependencies which cannot be simplified in a linear subspace, giving kernel PCA an advantage over other linear techniques. This corresponds to mapping the original scheduling variables to a set of lower dimensional variables via a nonlinear mapping. However, to recover the original coefficient functions of the model, this nonlinear mapping is needed to be inverted. Such an inversion is not straightforward. The reduced scheduling variables are a nonlinear expansion of the original scheduling variables into a high-dimensional feature space, an inverse mapping for which is not available. Therefore, we cannot generally assert that such an expansion has a "pre-image" in the original scheduling region. While certain pre-image approximation algorithms are found in the literature for Gaussian kernel-based PCA, we aim to generalize the pre-image estimation algorithm to other commonly used kernels, and formulate an iterative pre-image estimation rule. Finally, we consider the case study of a physical system described by an LPV model and compare the performance of linear and kernel PCA-based LPV model reduction.**

I. Introduction

Principal component analysis (PCA) is a mathematical tool that extracts a set of linearly uncorrelated variables from an observation of possibly correlated variables using orthogonal transformations. PCA is an eigenvector-based multivariate data analysis technique. The extracted variables are called principal components and are arranged in descending order of their variance in the data, measured by their corresponding eigenvalues. This means that the data components with very small variance can be neglected without losing useful information, a property which makes PCA an extremely attractive option for dimensionality reduction purpose [1].

*This work was made possible by the NPRP grant # NPRP 5-574-2-233 from Qatar National Research Fund (QNRF), a member of Qatar Foundation. The statements made here are solely the responsibility of the authors.

†Syed Z. Rizvi and Javad Mohammadpour are with the Complex Systems Control Laboratory (CSCL), College of Engineering, The University of Georgia, Athens, GA 30602, USA. Corresponding author email: szrizvi@uga.edu

‡Roland Tóth is with the Control Systems Group, Dept. of Electrical Engineering, Eindhoven University of Technology, P.O.Box 513, 5600 MB Eindhoven, The Netherlands. Email: r.toth@tue.nl

§Nader Meskin is with the Dept. of Electrical Engineering, Qatar University, Doha, Qatar, email: nader.meskin@qu.edu.qa

The ability of PCA to reduce the data dimension makes it ideal for tightening the scheduling region in linear parameter-varying (LPV) models. LPV systems are a class of systems, in which nonlinear models can be represented as a linear dynamic relation of the input and output variables, where this relation is dependent on a measurable time-varying signal, the so called scheduling variable, which expresses the varying operating conditions of the system. This allows one to apply linear optimal control techniques to nonlinear systems represented by LPV models. However, in practice, LPV controller design often encounters significant difficulties due to the high number of scheduling variables and conservatism and overbounding in the scheduling region [2]. For instance, in polytopic LPV systems, the complexity of controller synthesis has an *exponential* dependence on the number of scheduling variables. Hence, a large number of scheduling signals results in a high computational complexity for controller synthesis. Due to this, one of the objectives in deriving an LPV model for even a highly complex system is to limit the number of scheduling variables to a very few as described in [3], [4], [5]. PCA has proven to be an effective tool for solving this problem and a handful of papers have successfully demonstrated the use of PCA for finding tighter scheduling regions for LPV systems (see [2], [7]).

With the emergence of kernel-based techniques, a new avenue of data processing appeared in the last decade. Kernels are nonlinear functions that enable us to perform linear operations in high-dimensional feature spaces, where it is easier to separate components in the data. These nonlinear functions operate in the original data space and do not require mapping of the original data to the feature space, where the actual extraction takes place. This so called *kernel trick* has made component extraction much more efficient and realizable [8]. It is thus natural to investigate the use of kernel-based PCA for attaining efficient LPV models with reduced scheduling region.

In this paper, we explore the use of kernel-based PCA for LPV model reduction. It should be noted that model reduction in the LPV case refers to both reduction of the number of state variables (model order) and scheduling dependency as these facets of complexity are strongly related [5]. In particular, reduction of state-order can result in an increased complexity, while reduction of dependency is often available via the introduction of extra state variables [5]. Here, we address the problem of reduction of the complexity of the dependency with a proposed kernel PCA approach while the current state order is preserved. We analyze the advantages of kernel-based PCA over linear PCA, as well as the difficulties

associated with it in obtaining a pre-image of the reduced parameters in the feature space. The paper is arranged as follows: Section II gives an introduction to LPV systems and formulates the problem of interest. Section III revisits the use of PCA for LPV systems. In Section IV, kernel-based PCA is applied to determine a tighter LPV scheduling region, and the mathematical advantages and pitfalls are explored. Section V considers a practical example of a mechanical system approximated by kernel-based PCA reduced LPV model. Some concluding remarks are made in Section VI. Throughout this paper, unless otherwise specified, for any given vector $\beta \in \mathbb{R}^n$, we use the notation β_i^j to denote the j^{th} component of the i^{th} measurement vector β_i.

II. Preliminaries and Problem Formulation

Consider a nonlinear system Q shown in Figure 1. The system describes (possibly) nonlinear dynamic relation between measurable signals $\mu : \mathbb{R} \to \mathbb{R}^s$. Let $\mathfrak{B}$ be the set of all trajectories of μ compatible with Q. By introducing an auxiliary variable θ, one can reformulate the system representation of Q as shown in Figure 1 (b), where given the trajectory of θ, all relations between μ are linear. By applying this reformulation of cutting the dependency of θ on Q, and assuming that all trajectories of θ are allowed in the resulting θ-dependent linear structure which is now possible in Q, the reformulated system will form a possible behavior $\mathfrak{B}'$. This behavior $\mathfrak{B}'$ will contain $\mathfrak{B}$, giving us a linear, but θ-dependent description of Q. The reformulated system represents an LPV system. This enables us to use several linear control techniques and convex controller synthesis for the nonlinear system described by an LPV representation. There can be several different relations between the scheduling variable θ and the original variables μ. Variable θ might be a free variable w.r.t. Q, However, often the auxillary variable depends on other signals. In such a case, the resulting system is referred as a quasi parameter-varying system [9].

LPV systems in continuous-time are often described by a state-space representation as follows:

$$\begin{aligned} \dot{x}(t) &= A(\theta(t))x(t) + B(\theta(t))u(t), \\ y(t) &= C(\theta(t))x(t) + D(\theta(t))u(t), \end{aligned} \tag{1}$$

where $x(t) \in \mathbb{R}^{n_x}$, $u(t) \in \mathbb{R}^{n_u}$, and $y(t) \in \mathbb{R}^{n_y}$ represent the state, input, and output variables at time t, respectively. The system matrix, as well as the input, output and feedthrough matrices are continuous functions of the time-varying parameters $\theta(t) \in \mathbb{R}^l$, commonly known as the scheduling variables. The scheduling variables are in turn a continuous function of measurable signals $\mu(t) \in \mathbb{R}^s$, available from the system. This dependence can be written as

$$\theta(t) = p(\mu(t)), \quad p : \mathbb{R}^s \to \mathbb{R}^l. \tag{2}$$

The LPV system is considered affine in the scheduling variables if

$$Q(\theta) = \sum_{i=1}^{l} \theta^i Q_i, \tag{3}$$

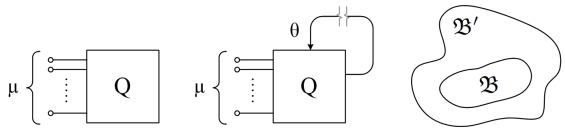

Fig. 1. (a) Original nonlinear system representation. (b) LPV representation. (c) Resulting behaviors

where $Q(\theta)$ is given by

$$Q(\theta) = \begin{bmatrix} A(\theta) & B(\theta) \\ C(\theta) & D(\theta) \end{bmatrix}. \tag{4}$$

Now, consider the compact set $R_\theta \subset \mathbb{R}^l : \theta(t) \in R_\theta, \forall t > 0$ defined by vertices

$$R_\theta := \text{Co}\{\theta_{v1} \cdots \theta_{vN}\}, \tag{5}$$

where $N = 2^l$ defines the number of vertices in the scheduling region and Co denotes minimal convex hull. Given the fact that θ can be obtained by a convex combination of vertices θ_{vi}, and that $Q(\theta)$ depends affinely on the scheduling variables, it follows that the system can be represented by a linear combination of multiple LTI systems at the vertices. Such a representation/system is referred to as a polytopic LPV system, giving a highly useful system representation for LPV control design.

Given the system (1), the problem of LPV model reduction can be described as follows: Find a mapping defined by

$$\rho(t) = q(\mu(t)), \quad q : \mathbb{R}^s \to \mathbb{R}^m, \tag{6}$$

where $m \leq l$, such that the system matrices in

$$\begin{aligned} \dot{x}(t) &= \tilde{A}(\rho(t))x(t) + \tilde{B}(\rho(t))u(t), \\ y(t) &= \tilde{C}(\rho(t))x(t) + \tilde{D}(\rho(t))u(t), \end{aligned} \tag{7}$$

approximate (1) sufficiently well. We use kernel-based PCA to find a tighter scheduling region, and seek to find a relatively small m, that can approximate the original system with the one in (7), without incurring a significant loss of useful information in the scheduling variable data. The later sections in this paper will formulate an accuracy measure for this purpose.

III. PCA for LPV Modeling

Before formulating a kernel-based PCA algorithm for model reduction, we quickly revisit the standard linear PCA as applied to the LPV model reduction problem. For the basic details of PCA, we refer the reader to [10]. To apply PCA to LPV scheduling variable data, one first needs to generate and collect data by means of measurement or simulation, such that the data covers all regions of operation within the operating range. Given the LPV system (1) and assuming that the scheduling signals have been sampled at time instants $t = 1, 2, \cdots, n$, scheduling variables $\theta_i \in \mathbb{R}^l$ for $i = 1, 2, \cdots, n$ are computed and represented by the following $l \times n$ matrix:

$$\Theta = \begin{bmatrix} \theta_1 & \cdots & \theta_n \end{bmatrix} = \begin{bmatrix} p(\mu_1) & \cdots & p(\mu_n) \end{bmatrix},$$

where $n \geq l$. PCA can be applied either by performing a singular value decomposition (SVD) on the data matrix Θ, or by solving an eigenvalue problem for the covariance matrix $\Theta\Theta^{\mathrm{T}}$. Here, we describe the procedure based on eigenvalue decomposition problem in order to maintain uniformity with the kernel version of PCA. The covariance matrix is given by $\bar{C} = \Theta_c \Theta_c^{\mathrm{T}}$, where $\Theta_c = \mathscr{N}(\Theta) = \Theta - \Theta_{\mathrm{mean}}$ is the data centered around the origin. We then solve an eigenvalue problem for the covariance matrix $\bar{C}$, such that $\bar{C}v_i = \lambda_i v_i$, where λ_i and v_i are the i^{th} eigenvalue and eigenvector, respectively. The eigenvectors are then sorted in descending order of their corresponding non-zero eigenvalues, and the m principal components for a test point θ_t at time t are extracted using

$$\rho_t \quad = \quad q(\mu_t) = V_m^{\mathrm{T}} p(\mu_t) = V_m^{\mathrm{T}} \theta_t,$$

where V_m denotes an $l \times m$ matrix whose columns contain the first m significant eigenvectors associated with the m significant eigenvalues. The approximation of the actual parameter $\hat{\theta}_t$ corresponding to ρ_t can be computed as

$$\hat{\theta}_t = \mathscr{N}^{-1}(V_m \rho_t), \tag{8}$$

where $\mathscr{N}^{-1}(v_m \rho_t) = v_m \rho_t + \Theta_{mean}$ denotes scaling back of the data by adding the mean along each dimension of Θ.

IV. Kernel PCA for LPV Modeling

Kernel-based PCA, more simply known as kernel PCA, is an extension of the traditional PCA approach, in which linear dot-product operation is performed, albeit in a higher dimensional feature space [1]. Due to the high dimension of the feature space, separation of features or components in the data is much easily realizable. The beauty of kernel-based methods primarily lies in the now-famous *kernel trick*, which allows performing linear operations in the feature space without explicitly mapping the parameters into the feature space. In LPV systems, this can lead to reduced over-parametrization by reducing scheduling variables more effectively.

Let us assume that the scheduling variables of the LPV system (1) are mapped into the feature space as $\Phi(\theta_1), \Phi(\theta_2), \cdots, \Phi(\theta_n)$. We also assume that the mapped parameters are centered, *i.e.*, $\sum_{j=1}^{n} \Phi(\theta_j) = 0$. Since it is not possible to obtain the mean of data we do not explicitly calculate, we will assume at this point that the data is centered in the feature space and explore an implicit alternative to centering the data, later in this section. To perform traditional PCA on this data, we derive the covariance matrix as follows:

$$\bar{C} = \frac{1}{n} \sum_{j=1}^{n} \Phi(\theta_j) \Phi^{\mathrm{T}}(\theta_j). \tag{9}$$

In order to select the principal components, the relation $\lambda_i v_i = \bar{C} v_i$ should hold, where λ_i and v_i denote the i^{th} eigenvalue and eigenvector, respectively. We can, therefore, consider the following

$$\lambda (v \cdot \Phi(\theta_i)) = (\bar{C} v \cdot \Phi(\theta_i)), \quad \forall\, i = 1, \cdots, n, \tag{10}$$

where $(a \cdot b)$ denotes dot product given by $a^{\mathrm{T}} b$, and that there exists coefficients α_w for $w = 1, \cdots, n$ such that

$$v = \sum_{w=1}^{n} \alpha_w \Phi(\theta_w). \tag{11}$$

This means that the eigenvectors of the matrix $\bar{C}$ belong to the span of $\Phi(\theta_j)$ for $j = 1, \cdots, n$. Substituting (9) and (11) in (10), we obtain [12]

$$\begin{aligned} &\lambda \sum_{w=1}^{n} \alpha_w (\Phi(\theta_w) \cdot \Phi(\theta_i)) \\ &= \left\{ \left(\frac{1}{n} \sum_{j=1}^{n} \Phi(\theta_j) \Phi^{\mathrm{T}}(\theta_j) \right) \left(\sum_{w=1}^{n} \alpha_w \Phi(\theta_w) \right) \right\} \cdot \Phi(\theta_i) \\ &= \frac{1}{n} \sum_{j=1}^{n} \sum_{w=1}^{n} \alpha_w \left(\Phi(\theta_j) \cdot \Phi(\theta_w) \right) \left(\Phi(\theta_j) \cdot \Phi(\theta_i) \right) \quad \forall i = 1, \cdots, n \end{aligned} \tag{12}$$

The eigenvalue problem in (12) only involves dot products of mapped vectors in the feature space and does not explicitly require computing $\Phi(\cdot)$. We define an $n \times n$ matrix K as

$$K_{ij} = (\Phi(\theta_i) \cdot \Phi(\theta_j)) = k(\theta_i, \theta_j), \tag{13}$$

where matrix $K \in \mathbb{R}^{n \times n}$ is known as the Gram matrix, or kernel matrix, and k is a nonlinear kernel function. Substituting (13) in (12) gives us the final problem of finding the solutions to

$$\lambda \sum_{w=1}^{n} \alpha_w k(\theta_w, \theta_i) = \frac{1}{n} \sum_{j=1}^{n} \sum_{w=1}^{n} \alpha_w k(\theta_j, \theta_w) k(\theta_j, \theta_i), \tag{14}$$

where $i = 1, \cdots, n$, α_r^i denotes the i^{th} component of α_r. Writing (14) in matrix form, we obtain

$$n \lambda \alpha = K \alpha \tag{15}$$

for all nonzero eigenvalue. The solutions α_r belonging to nonzero eigenvalues need to be normalized by requiring that the corresponding vectors in the resulting feature space F are normalized. This translates to the condition $(v_r \cdot v_r) = 1$. Using (11), (13) and (15), we obtain

$$1 = \sum_{i,j=1}^{n} \alpha_r^i \alpha_r^j (\Phi(\theta_i) \cdot \Phi(\theta_j)) = (\alpha_r \cdot K \alpha_r) = \lambda_r (\alpha_r \cdot \alpha_r). \tag{16}$$

This translates to dividing each eigenvector α_r of K by $\sqrt{\lambda_r}$ in order to normalize it. Lastly, for principal component extraction, we compute the projections of the image of a test-point θ_t at a given time t onto the eigenvectors v_r in the feature space as

$$\rho_t^r = (v_r \cdot \Phi(\theta_t)) = \sum_{j=1}^{n} \alpha_r^j (\Phi(\theta_j) \cdot \Phi(\theta_t)) = \sum_{j=1}^{n} \alpha_r^j \, k(\theta_j, \theta_t), \tag{17}$$

where ρ_t^r is the r^{th} component of the said projection of $\Phi(\theta_t)$ on v_r. It is noteworthy that the above equation does not explicitly require the computation of $\Phi(\theta_j)$. What is needed is only the dot product in feature space F. This can be computed by using nonlinear kernel functions k. This way, the *kernel trick* allows us to project the image of a point onto the eigenvectors in F without explicitly having to calculate the data image in F.

Finally, it can be recalled that the data was initially, and rather simplistically, assumed to be centered in F. This

however is not always the case, and care must be taken to center the data in F first. However, we cannot in general center the scheduling variables data in F, since we cannot compute the mean of the data that we explicitly do not calculate. We therefore, work around it and obtain a centered Gram matrix as explained in [1].

$$\tilde{K}_{ij} = K - 1_n K - K 1_n + 1_n K 1_n, \tag{18}$$

where $1_n \in \mathbb{R}^{n\times n}$ with each entry being $1/n$. Kernel functions can be chosen from a variety of different functions that exist in the literature, and have been used for classification, feature extraction, and other applications [13]. These include but are not limited to the polynomial kernel given as

$$k(\theta_i, \theta_j) = (\theta_i \cdot \theta_j + 1)^d, \tag{19}$$

the radial basis function kernel given as

$$k(\theta_i, \theta_j) = e^{-\frac{||\,\theta_i - \theta_j\,||^2}{\sigma^2}}, \tag{20}$$

and the sigmoid kernel given as

$$k(\theta_i, \theta_j) = \tanh(a\ \theta_i \cdot \theta_j + b), \tag{21}$$

where d, σ, a, and b in (19), (20), and (21) refer to the degree of polynomial, the spread of radial basis function, and the slope and bias in sigmoid functions, respectively.

A. Accuracy of the estimated LPV model

Using linear or kernel PCA, one can reduce the LPV model to have affine dependence on lesser number of scheduling variables. The accuracy of the estimated model can be gauged from the fraction of total data variation

$$a(m) = \frac{\sum_{i=1}^{m} \lambda_i}{\sum_{i=1}^{v} \lambda_i}, \tag{22}$$

where m is the number of reduced parameters, and λ_i denotes the i^{th} eigenvalue of the kernel matrix $\tilde{K}$. In case of linear PCA, the eigenvalue is that of the covariance matrix [6]. Since the kernel and covariance matrices are square matrices of dimensions n and l, respectively, v is consequently equal to n and l for kernel and linear PCA, respectively. Therefore, by choosing the number of scheduling variables, a trade-off can be achieved between accuracy and complexity.

B. The pre-imaging problem

In the LPV model reduction problem, it is important that we are able get the original scheduling variables back by using the reduced set of scheduling variables. This is because of the fact that for LPV control design, the controller matrices are scheduled based on the reduced variables. The control matrices however depend on the open-loop LPV system matrices which are a function of the original scheduling variables. Therefore, one can also say that the original scheduling variables are required for calculating the new representation of the system. The kernel PCA, however, suffers from an inability to reconstruct the original patterns in a straightforward way. So to say, given reduced parameter vector ρ_i extracted from θ_i using kernel PCA, there is no systematic way to reconstruct the original parameters θ_i, since feature space algorithms express their solutions as expansions in terms of mapped data points. Therefore, one cannot in general say that a pre-image exists under the map Φ, for which $\Phi(\theta_i) = \Psi$. Moreover, this problem of finding the pre-image might be unsolvable in the general case, since the pre-image might not always exist [11]. Schölkopf *et al.* therefore argued in [11] that rather than finding the exact pre-image, it is possible to find an estimate of the pre-image. This is done by considering the fact that we may seek to approximate the pre-image of a feature space expansion $\Psi = \sum_{i=1}^{n} \gamma_i \Phi(\theta_i)$ by its estimate $\Psi' = \beta\Phi(\tilde{\theta}_i)$, where $\tilde{\theta}$ denotes the approximated pre-image. One can, therefore, attempt to minimize the distance between Ψ and Ψ'. Somewhat equivalently, one can minimize the distance between Ψ and the orthogonal projection of Ψ onto the span of $\Phi(\tilde{\theta})$, *i.e.*,

$$||\frac{(\Psi\cdot\Phi(\tilde{\theta}))}{(\Phi(\tilde{\theta})\cdot\Phi(\tilde{\theta}))}\Phi(\tilde{\theta}) - \Psi||^2 = ||\Psi||^2 - \frac{(\Psi\cdot\Phi(\tilde{\theta}))^2}{(\Phi(\tilde{\theta})\cdot\Phi(\tilde{\theta}))} \tag{23}$$

This can be achieved by maximizing

$$H(\tilde{\theta}) = \frac{(\Psi\cdot\Phi(\tilde{\theta}))^2}{(\Phi(\tilde{\theta})\cdot\Phi(\tilde{\theta}))}, \tag{24}$$

and once the maximum is achieved, we set the variable β as $\beta = \Psi\cdot\Phi(\tilde{\theta})\ /(\ \Phi(\tilde{\theta})\cdot\Phi(\tilde{\theta}))$ [11]. To find the extremum, we solve

$$\nabla_{\tilde{\theta}} H(\tilde{\theta}) = \nabla_{\tilde{\theta}} \frac{(\Psi\cdot\Phi(\tilde{\theta}))^2}{(\Phi(\tilde{\theta})\cdot\Phi(\tilde{\theta}))} = \nabla_{\tilde{\theta}} \frac{(\sum_{i=1}^{n}\gamma_i k(\theta_i,\tilde{\theta}))^2}{k(\tilde{\theta},\tilde{\theta})} = 0 \tag{25}$$

1) pre-image for Gaussian kernels: For the Gaussian kernel, since $k(\tilde{\theta},\tilde{\theta}) = 1$, (25) becomes

$$\nabla_{\tilde{\theta}} \left(\sum_{i=1}^{n} \gamma_i e^{-\frac{||\theta_i - \tilde{\theta}||^2}{2\sigma^2}} \right)^2 = 0, \tag{26}$$

giving us

$$\tilde{\theta} = \frac{\sum_{i=1}^{n} \gamma_i\, k(\theta_i,\tilde{\theta})\theta_i}{\sum_{i=1}^{n} \gamma_i\, k(\theta_i,\tilde{\theta})}. \tag{27}$$

2) pre-image for polynomial kernels: Similarly, solving (25) for polynomial kernel gives us

$$\tilde{\theta} = \frac{\left(\sum_{i=1}^{n} \gamma_i\, k^{d-1}(\theta_i,\tilde{\theta})\theta_i\ \right) k^d(\tilde{\theta},\tilde{\theta})}{\left(\sum_{i=1}^{n} \gamma_i\, k^{d}(\theta_i,\tilde{\theta})\theta_i\ \right) k^{d-1}(\tilde{\theta},\tilde{\theta})}, \tag{28}$$

where $k^d(\cdot,\cdot)$ indicates a polynomial kernel function of degree d, $\gamma_i = \sum_{j=1}^{m} \rho^j \alpha_j^i$, with ρ^j being the j^{th} component of the reduced variable ρ[14]. Similar relations can be worked out for other kernels as well.

A generalized iterative update rule

To formulate an iterative update equation for the estimate $\tilde{\theta}$, we can generalize (27) and (28) as $\tilde{\theta} = f(\tilde{\theta})$. Defining a new function

$$g(\tilde{\theta}) = \tilde{\theta} - f(\tilde{\theta}) = 0, \tag{29}$$

Algorithm 1 Applying kernel PCA for LPV model reduction

Step 1: Obtain a set of measurable signals using measurements or simulations, covering the expected range of operation
Step 2: Compute the trajectories of the corresponding scheduling variables θ_i for $i = 1, \cdots, n$
Step 3: Compute matrix K using (13)
Step 4: Center the data in feature space to find $\tilde{K}$ using (18)
Step 5: Solve (15) by diagonalizing $\tilde{K}$
Step 6: Normalize the eigenvectors using (16)
Step 7: For online LPV control: For each set of observed scheduling variable θ_t at time t, get the r^{th} component of reduced-dimension parameter ρ_t^r by using (17)
Step 8:
while $g(\tilde{\theta})$ is not minimized, **do**
Compute pre-image $\tilde{\theta}_t$ from ρ_t using (30)
end while

we can reduce the problem at hand to finding the root $\tilde{\theta}$ of $g(\tilde{\theta})$ such that $g(\tilde{\theta}) = 0$. Note that $g(\tilde{\theta})$ is a vector-valued function of vector $\tilde{\theta}$. To find the root of $g(\tilde{\theta})$, one can use iterative root-finding algorithms like Newton's method. An alternative to root finding is to use a steepest descent method with variable or fixed step size, in order to minimize $g(\tilde{\theta})$. Steepest descent methods iteratively estimate the pre-image $\tilde{\theta}$ according to the following equation:

$$h(\bar{k}) = -[J_g(\tilde{\theta}(\bar{k}))]^{\mathrm{T}} g(\tilde{\theta}(\bar{k})),$$
$$\tilde{\theta}(\bar{k}+1) = \tilde{\theta}(\bar{k}) + \eta h(\bar{k}), \tag{30}$$

where $\bar{k}$ is the iteration index, η is the step size, $g[\tilde{\theta}(\bar{k})]$ is as defined in (29), $f(\tilde{\theta})$ would be formulated according to the kernel function in use, and $J_g(\cdot)$ is the Jacobian matrix. A small step size η ensures stability at the cost of slower convergence; conversely, a larger η speeds up convergence but may cause instability of the iterative approximation. The success or failure of steepest descent methods are highly dependent on the nonlinearities in the data, the number of minima in the function, the choice of the kernel function, a "decent" initial condition for $\tilde{\theta}$ and an initial estimate of the Jacobian. Certain variable step size variants of the steepest descent methods vary the step size at each iteration to ensure a stable convergence to the best solution. In case of numerical instabilities, which is rare, the algorithm simply requires a restart with different starting values. Finally, the kernel PCA-based method for LPV model reduction is summarized in Algorithm 1.

Remark: The pre-image mapping from reduced scheduling variables ρ_i to approximated scheduling variables $\tilde{\theta}_i$ is based on solving a nonlinear problem; therefore, it is noteworthy here that the LPV model approximated using kernel PCA is no longer affine in the approximated variables $\tilde{\theta}_i$.

V. Numerical Example

A dynamic model of a two-link planar robotic manipulator is considered here; the schematic diagram is shown in Figure 2. Detailed nonlinear model and the associated parameters have been taken from [15]. The lower arm of the robot is known as the shoulder, while the upper part is simply known as the arm, which is attached to the actuator. The two joints are connected via a gear servo mechanism. Following the ideas in [2] and [15], the derived LPV model is given as

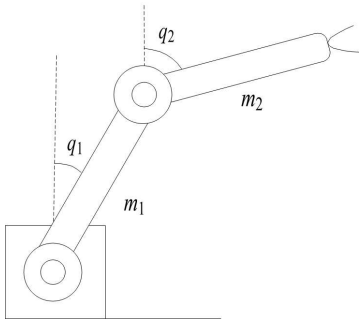

Fig. 2. Schematic of 2-DoF robotic manipulator

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ cd\theta_3 & -be\theta_4 & \theta_5 & b\theta_6 \\ -bd\theta_7 & ae\theta_8 & \theta_9 & \theta_{10} \end{bmatrix}, C = \begin{bmatrix} I & \emptyset \end{bmatrix}$$
$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ cnk_m\theta_1 & -bnk_m\theta_2 \\ -bnk_m\theta_2 & ank_m\theta_1 \end{bmatrix}, D = \emptyset,$$
$$h = ac - b^2\cos^2(\Delta),$$
$$\theta_1 = \frac{g}{h}, \theta_2 = \frac{g\cos(\Delta)}{h}, \theta_3 = \frac{\mathrm{sinc}(q_1)}{h},$$
$$\theta_4 = \cos\Delta\frac{\mathrm{sinc}(q_2)}{h}, \theta_6 = \frac{-c\sin(\Delta)\dot{q_2} + \cos(\Delta)f}{h},$$
$$\theta_5 = \frac{-b^2\sin(\Delta)\ \cos(\Delta)\dot{q_1} - (c + b\cos(\Delta))f}{h},$$
$$\theta_7 = \frac{\cos(\Delta)\mathrm{sinc}(q_1)}{h}, \theta_8 = \frac{\mathrm{sinc}(q_2)}{h},$$
$$\theta_9 = \frac{ab\ \sin(\Delta)\dot{q_1} + f(a + b\cos(\Delta))}{h},$$
$$\theta_{10} = \frac{b^2\sin(\Delta)\cos(\Delta)\dot{q_2} - af}{h}, \tag{31}$$

where $\cos(\Delta) = \cos(q_1 - q_2)$, $\sin(\Delta) = \sin(q_1 - q_2)$, and I and $\emptyset$ are identity and zero matrices of appropriate dimensions, respectively. The state vector is given by $x = [q_1\ q_2\ \dot{q_1}\ \dot{q_2}]^{\mathrm{T}}$. Angles of the two links with respect to the vertical reference make up the first two states, while angular velocities make up the other two states of the LPV model. This LPV model has $s = 4$ states and a total of $l = 10$ scheduling variables. The objective here is to reduce the number of scheduling variables in order to reduce the over-parametrization in the given LPV model. We seek to find, using PCA, a smaller number of reduced parameters m, with $m \leq l$ that gives an LPV model that can approximate the original model (31).

A set of trajectories is generated with open-loop simulation of the LPV model using sinusoidal inputs u and scheduling variables are computed. We apply linear and kernel PCA on

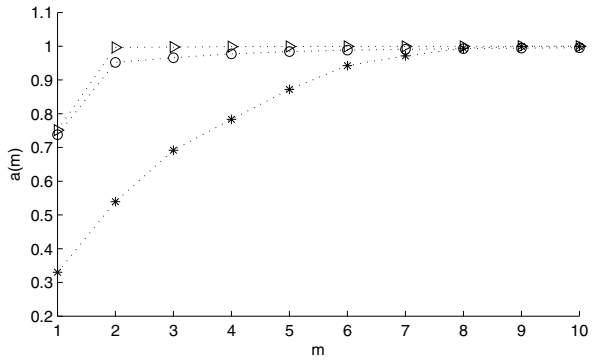

Fig. 3. Accuracy of estimates (22) as a function of the number of scheduling variables m for (*) linear PCA, (o) kernel PCA with polynomial kernel, and ($\triangleright$) kernel PCA with sigmoid kernel

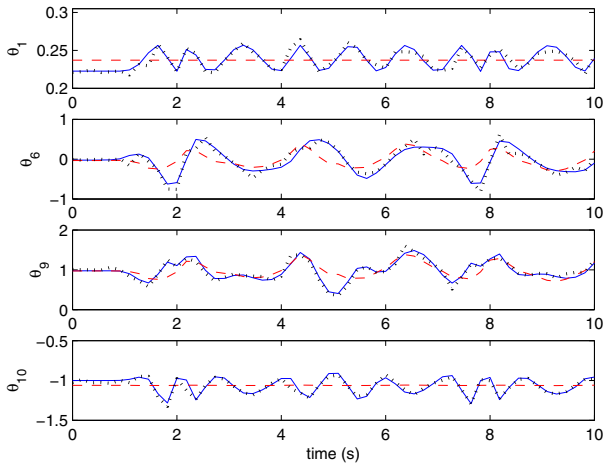

Fig. 4. Scheduling variables θ_i (blue solid line), PCA-based approximation $\hat{\theta}_i$ (red dashed line), and kernel PCA-based approximation $\tilde{\theta}_i$ (black dotted line) with $m = 1$.

the data. For the kernel PCA case, we choose a polynomial kernel of degree 12 and sigmoid kernel with $a = 0.5$, $b = -13$, based on trial and error. Gaussian kernel-based PCA provides an accuracy measure close to that of linear PCA with this problem; hence, it is not discussed here. We then compare the accuracy (22) of the approximated LPV models as a function of the number of LPV variables, m. This is shown in Figure 3. Kernel PCA for both kernels gives around 78% accuracy with one scheduling variable, much higher than 33% accuracy of the linear PCA. Both polynomial and sigmoid kernels give an almost equal measure of accuracy; therefore, here we present the results for polynomial kernel PCA only. Having reduced the scheduling variables from $l = 10$ to $m = 1$, we re-estimate the actual scheduling variables using pre-imaging. We employ the steepest descent estimation (30) with step size $\eta = 0.001$. Initial estimate is chosen randomly. Results for a few parameters are shown in Figure 4. Other parameters show a very similar trend to those plotted here. These results show a good fit between the actual (solid line) and kernel PCA re-estimated (dotted line) scheduling variables. As observed, linear PCA (dashed line) simply fails to estimate θ_1 and θ_{10}, while showing some trend in the other parameters.

VI. Concluding Remarks

In this paper, we have explored the use of kernel-based PCA for the purpose of dimensionality reduction of the scheduling variables in LPV representations. Reducing the number of scheduling variables is a problem of paramount importance, since it directly translates to the reduction in computational complexity for LPV controller design. Though parameters mapped into feature space cannot be systematically mapped back to the scheduling region as in the case of linear PCA, they can still be estimated efficiently using an iterative update rule derived in this paper. While an iterative approach to estimation is not as quick and straightforward as projecting the parameters on an eigenvector space, the improvement in dimensionality reduction makes it a worthy trade-off. Results in this paper provide encouraging insights into the use of kernel PCA for LPV model reduction.

References

[1] B. Schölkopf, A. Smola, and K.R. Mller, "Kernel principal component analysis," in *Advances in kernel methods-support vector learning*, Boston, MA: MIT Press, 1999, pp. 327-352.

[2] A. Kwiatkowski, and H. Werner, "PCA-based parameter set mappings for LPV models with fewer parameters and less overbounding," *IEEE Trans. Control Syst. Tech.*, vol. 16, no. 4, pp. 781-788, Jul. 2008.

[3] P. Gaspar, Z. Szabo, and J. Bokor, "The design of an integrated control system in heavy vehicles based on an LPV method," *in 44th IEEE Conf. Decision and Control, and 2005 European Control Conf.*, 2005, pp. 6722-6727.

[4] M.M. Siraj and R. Tóth, "On the problem of model reduction of LPV systems," in *Proc. of the 31st Benelux Meeting*, Heijderbos, The Netherlands, 2012.

[5] R. Tóth, H. Abbas and H. Werner, "On the state-space realization of LPV input-output models: practical approaches," *IEEE Trans. Control Syst. Tech.*, vol. 20, no. 1, pp. 139-153, Jan. 2012.

[6] S.M. Hashemi, H.S. Abbas, and H. Werner, "LPV modeling and control of a 2-DOF robotic manipulator using PCA-based parameter set mapping," in *48th IEEE Conf. Decision and Control, and 28th Chinese Control Conf.*, Shanghai, China, 2009, pp. 7418-7423.

[7] M. Meisami-Azad, J. Mohammadpour, K.M. Grigoriadis, M.P. Harold, and M.A. Franchek, "PCA-based linear parameter varying control of SCR aftertreatment systems," in *American Control Conf.*, San Francisco, CA, 2011, pp. 1543-1548.

[8] B. Schölkopf, A.J. Smola, *Learning with Kernel*, Cambridge, MA: MIT press, 2002.

[9] R. Tóth, J.C. Willems, P.S.C. Heuberger, and P.M.J. Van den Hof, "The Behavioral Approach to Linear Parameter-Varying Systems," *IEEE Trans. Auto. Control*, vol. 56, no. 11, Nov. 2011.

[10] I.T. Jolliffe, *Principal Component Analysis*, 2nd ed. NY: Springer, 2002.

[11] B. Schölkopf, S. Mika, C.J.C Burges, P. Knirsch, K.R. Müller, G. Ratsch, and A.J. Smola, "Input space versus feature space in kernel-based methods," *IEEE Trans. Neural Networks*, vol. 10, no. 5, pp. 1000-1016, Sep. 1999.

[12] J.M. Lee, C. Yoo, S.W. Choi, P.A. Vanrolleghem, I.B. Lee, "Nonlinear process monitoring using kernel principal component analysis," *Chemical Engineering Science*, vol. 59, pp. 223-234, 2004.

[13] B.E. Boser, I.M. Guyon, and V. Vapnik, "A training algorithm for optimal margin classifiers," in *Fifth ACM Annual Workshop on Computational Learning Theory*, Pittsburg, PA, 1992.

[14] S. Mika, B. Schölkopf, A.J. Smola, K.R. Mller, M. Scholz, and G. Rätsch, "Kernel PCA and de-noising in feature spaces," *NIPS*, vol. 11, pp. 536-542, 1998.

[15] Z. Yu, H. Chen, and P. Woo, "Gain scheduled LPV H_∞ control based on LMI approach for a robotic manipulator," *J. Rob. Syst.*, vol. 19, no. 12, pp. 585-593, 2002.