Model Reduction in Linear Parameter-Varying Models using Autoencoder Neural Networks

Syed Z. Rizvi, Farshid Abbasi, and Javad Mohammadpour Velni

***Abstract*— This paper presents a method for model reduction of systems represented by linear parameter-varying (LPV) models seeking to reduce the scheduling variables space by employing autoencoder (AE) neural networks. The reduction of scheduling variables results in an exponential decrease in computational complexity for gain-scheduled LPV controller synthesis. Autoencoders rely on minimizing regularized sparse least-squares cost function that seeks to fit scheduling variables reproduced from the new lower-dimensional variables to the original measurements. This way, unlike other unsupervised nonlinear reduction methods, AEs do not require separately solving for a pre-image in the original scheduling space. Moreover, unlike principal component analysis (PCA), AEs can employ nonlinear encoding, and are thus suitable for LPV models. When needed, AEs can add multiple layers for encoding and decoding, thus enabling deep learning methods for the purpose of model reduction. A case study of a mechanical system is considered and results are compared with linear dimensionality reduction techniques. The reduced model is used for $\mathcal{H}_\infty$ LPV controller synthesis and results are compared.**

I. Introduction

The need for accurate modeling of complex systems leads to derivation of high-dimensional and nonlinear mathematical models. This makes the applicability of such models in controller synthesis problems very limited. To tackle this, researchers have focused on various approaches to obtain reduced-order models representing an estimation of the underlying full-order linear [1], as well as nonlinear systems represented by nonlinear partial differential equations [2].

Linear parameter-varying (LPV) models are a class of linear models, using which, nonlinear systems can be represented by linear dynamic relations of the input and output variables; the relation itself is dependent on measurable time-varying signals called the *scheduling variables*. These scheduling variables express the varying operating conditions of the system. This allows for the applicability of linear optimal control techniques to nonlinear systems represented by LPV models. However, in practice, LPV controller design often encounters significant difficulties due to high number of scheduling variables, conservatism and over-bounding in the scheduling region [3]. 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 variables 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 [4], [5].

An autoencoder (AE) is a neural network that is trained to replicate its input at its output. A first layer encodes the original signals to a reduced-dimension variable, while a second layer learns a decoding map, mapping the reduced variable back to the actual measurements. This way, dimensionality reduction of the original variables is carried out with the intention that the reduced variables should have easy mapping back to the actual variables. A single-layer autoencoder with linear transfer function is nearly equivalent to *principal component analysis* (PCA); nearly in this context means that the reduced variables computed by the autoencoder will not be the same as PCA but would span the same subspace. The advantage of AEs is that unlike PCA, AEs are not restricted to a linear map. Moreover, since AEs are trained by minimizing estimation error, once trained, there is no need to solve for a pre-image as is the case in several kernel-based nonlinear PCA variants [6], [7]. Adding multiple layers, autoencoders can also be used as *deep neural networks* [8], making them suitable for dimensionality reduction of scheduling variables in LPV models with complex nonlinearities. The proposed framework in this paper uses AE-based method [9] employing nonlinear activation functions in order to obtain a reasonable low-dimensional transformation of the scheduling variables. Once a mapping is obtained for the reduced scheduling variables, another optimization problem is solved in order to optimize the reduced model for controller synthesis.

The paper is arranged as follows: Section II gives an introduction to LPV systems and defines the problem statement. Section III describes autoencoders and formultes the reduction problem. Model reduction using linear decomposition is revisited in Section IV. Numerical case study of a mechanical system is studied in Section V. Finally, concluding remarks are made in Section VI. Throughout this paper, for a given vector $\beta_i \in \mathbb{R}^n$, we use the notation $\beta_{i,j}$ to denote the j^{th} entry of β_i.

II. Problem Statement

A continuous-time nonlinear system Q is considered and shown in Figure 1. The system describes possibly nonlinear dynamic relation between the measurable signals $\mu : \mathbb{R} \to \mathbb{R}_\mu,\ \mathbb{R}_\mu \subseteq \mathbb{R}^s$. If $\mathfrak{B}$ represents the set of all trajectories of μ compatible with Q, then by introducing an auxiliary variable θ, we can reformulate the representation of Q as shown in Figure 1(b), where given the trajectory of $\theta : \mathbb{R} \to \mathbb{R}_\theta,\ \mathbb{R}_\theta \subseteq \mathbb{R}^l$, all relations between μ are linear.

†The authors are with School of Electrical & Computer Engineering, College of Engineering, The University of Georgia, Athens, GA 30602, USA. Corresponding author email: javadm@uga.edu

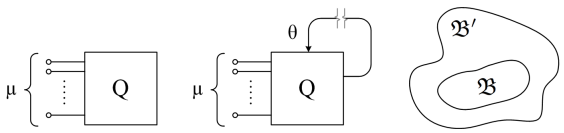

Fig. 1. (a) Original nonlinear system representation. (b) LPV representation. (c) Resulting behaviors

Assuming that all trajectories of θ are allowed independently from μ, the reformulated system will have a solution set $\mathfrak{B}' = \{(\theta, \mu) | \theta : \mathbb{R} \to \mathbb{R}_\theta, \mu : \mathbb{R} \to \mathbb{R}_\mu\}$ and the pair (θ, μ) satisfies the reformulated model equations. The behavior $\mathfrak{B}'$ contains $\mathfrak{B}$, giving a linear, but θ-dependent description of Q. The reformulated system represents an LPV system.

Consider an LPV state-space (LPV-SS) model in continuous-time described 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 LPV-SS matrices are continuous functions of the time-varying scheduling variables $\theta_t \in \mathbb{R}^l$. The scheduling variables are 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 model is considered affine in the scheduling variables if

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

where $Q(\theta_t)$ is given by

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

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

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

where $N = 2^l$ defines the number of vertices in the scheduling region and Co denotes minimal convex hull. The scheduling variables θ_t can be obtained by a convex combination of vertices θ_{vi}. Moreover, the LPV model $Q(\theta_t)$ depends affinely on the scheduling variables. These two conditions lead to the corollary that the system can be represented by a linear combination of multiple *linear time-invariant* (LTI) systems at the vertices. Such a system representation is referred to as a polytopic LPV system, giving a highly useful 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

$$\rho_t = q(\mu_t), \quad q : \mathbb{R}^s \to \mathbb{R}^m, \tag{6}$$

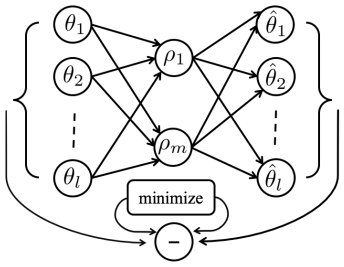

Fig. 2. A two-layer autoencoder.

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

$$\begin{aligned} \dot{x}_t &= \hat{A}(\rho_t)x_t + \hat{B}(\rho_t)u_t, \\ y_t &= \hat{C}(\rho_t)x_t + \hat{D}(\rho_t)u_t, \end{aligned} \tag{7}$$

have an affine or rational dependence on ρ_t and approximate (1) sufficiently well. While linear and nonlinear unsupervised methods for dimensionality reduction like principal component analysis (PCA) and its nonlinear variants have been used for the reduction of scheduling space in the past, in this paper, we attempt the use of autoencoders. In the next sections of this paper, we explain how autoencoders obtain the mapping $q : \mathbb{R}^s \to \mathbb{R}^m$, and discuss the advantages it presents to us over linear as well as nonlinear and kernelized versions of PCA.

III. LPV Model Reduction using Autoencoders

An autoencoder is a special neural network that is designed and trained to replicate its input at the output. An autoencoder can be broken into two parts: an encoder and a decoder that can each have multiple layers. A simple autoencoder with an encoding and decoding layer is illustrated in Figure 2. One can write one neuron (encoder) of an artificial neural network for the case of reducing the scheduling variables $\theta_t \in \mathbb{R}^l$ to $\rho_t \in \mathbb{R}^m$ in the following form:

$$\rho_t = h^{(1)}(W^{(1)\top}\theta_t + b^{(1)}), \tag{8}$$

where $W^{(1)} = [W_1^{(1)} \dots W_m^{(1)}] \in \mathbb{R}^{l \times m}$ is the weight matrix and $W_i^{(1)} = [w_{1i}^{(1)} \dots w_{li}^{(1)}]^\top$. Function $h^{(1)}(\cdot)$ is the transfer function and $b^{(1)} \in \mathbb{R}^m$ is the bias in the first layer. The second, or the decoding layer, can be described by the following equation:

$$\hat{\theta}_t = h^{(2)}(W^{(2)\top}\rho_t + b^{(2)}), \tag{9}$$

where $\hat{\theta}_t$ is a reconstruction of θ_t and $W^{(2)} \in \mathbb{R}^{m \times l}$ and $b^{(2)} \in \mathbb{R}^l$ represent the weighting matrix and the bias vector for the decoding layer.

An optimization problem is defined seeking to minimize the error between estimated scheduling variables $\hat{\theta}_t$ via reduced model and the actual measurements θ_t, for $t = 1, \dots, n$. The autoencoder seeks to minimize the following mean squared error cost with regularization for weights and

sparsity:

$$\min_{W_1,W_2} \mathcal{J} = \frac{1}{n}\sum_{i=1}^{l}\sum_{j=1}^{n}(\Theta_{ij}-\hat{\Theta}_{ij})^2 + \frac{\psi}{2}\sum_{k=1}^{g}\sum_{i=1}^{l}\sum_{j=1}^{m}(w_{ij}^{(k)})^2 + \beta\sum_{j}^{m}\Big(\nu\log(\frac{\nu}{\hat{\nu}_i}) + (1-\nu)\log(\frac{1-\nu}{1-\hat{\nu}_i})\Big), \quad (10)$$

where g is the number of layers, n denotes the number of observations for training, l is the dimension of the original scheduling variable θ_t, m is the dimension of reduced variable ρ_t, and ψ and β denote the regularization coefficients on the weights and sparsity, respectively. Additionally, Θ_{ij} denotes $\{i,j\}^{\text{th}}$ entry of the scheduling variable data matrix $\Theta \in \mathbb{R}^{l\times n}$ while $\hat{\Theta}_{ij}$ denotes the estimate of Θ_{ij} as decoded by the autoencoder. The first term in (10) is the mean squared error between the scheduling variable and its estimate, while the second term is regularization on the weights. The third term in the cost function is the sparsity regularization term known as Kullback-Leibler divergence function that attempts to enforce a constraint on the sparsity of the output from the hidden layer [10]. Variables $\hat{\nu}_i$ and ν denote the average output activation value of the neuron i and its desired value, respectively. The average activation value is defined as

$$\hat{\nu}_i = \frac{1}{n}\sum_{j=1}^{n} h(W_i^{(1)\top}\theta_j + b_i^{(1)}). \quad (11)$$

The goal of the model reduction problem is therefore to find the mapping $\rho_t = h^{(1)}(\theta_t) = h^{(1)}(p(\mu_t)) = q(\mu_t)$ as given in (8) with $\rho_t \in \mathbb{R}^m$ and $m < l$, such that the LPV-SS matrices have an affine or rational dependence on ρ. The problem is attempted to solve such that the cost (10) is minimized.

A. Decoding versus pre-imaging

When using linear techniques such as PCA for LPV model reduction, the reduced LPV model has mapping LPV-SS matrices that are automatically affine functions of the reduced scheduling variables. This will be revisited in the next section briefly. In other nonlinear variants of PCA, like kernel PCA, there is no systematic way of reconstructing the estimates of original variables θ_t, and in order to find the estimate of this so-called pre-image $\hat{\theta}_t$, one has to solve a nonlinear optimization problem [6], [7], [11]. However, unlike kernelized PCA, in the case of autoencoders, the mapping (9) already exists once an autoencoder is trained. Since the autoencoder is trained to replicate its input at the output and to minimize the squared error between its input and output, the activation function for the decoding layer already maps the reduced variables ρ_t to the estimates $\hat{\theta}_t$. Therefore, once trained, the autoencoder has both, the encoding function that maps the original variables to the reduced variables, and the decoding function that estimates the original scheduling variables.

Since $\hat{\theta}_t$ denotes an estimate of θ_t, we can write the following for the autoencoder-based reduced LPV model

$$\hat{Q}(\rho_t) = \begin{bmatrix}\hat{A}(\rho_t) & \hat{B}(\rho_t)\\ \hat{C}(\rho_t) & \hat{D}(\rho_t)\end{bmatrix} \approx \begin{bmatrix}A(\hat{\theta}_t) & B(\hat{\theta}_t)\\ C(\hat{\theta}_t) & D(\hat{\theta}_t)\end{bmatrix} = Q(\hat{\theta}_t). \quad (12)$$

Using (3)-(4) and (12), we can relate the reduced model to the full-order LPV model as follows:

$$\hat{Q}(\rho_t) \approx Q(\hat{\theta}_t) = Q_0 + \sum_{i=1}^{l} Q_i\hat{\theta}_{t,i} = Q_0 + \sum_{i=1}^{l} Q_i h^{(2)}(W^{(2)\top}\rho_t + b^{(2)})_i, \quad (13)$$

where $h^{(2)}(W^{(2)\top}\rho_t + b^{(2)})$ is the decoding function given in (9). From (13), one can realize that the reduced model is niether affine, and in most cases, nor is it rationally dependent on the reduced scheduling variables ρ_t.

B. Optimizing the reduced LPV model for controller synthesis

Suppose that the autoencoder-based reduced model is represented by state-space matrices $\breve{A}(\rho_t)$, $\breve{B}(\rho_t)$, $\breve{C}(\rho_t)$, and $\breve{D}(\rho_t)$ that have affine dependence on ρ_t. According to Apkarian et al. [12], the vertex property implies that for such an LPV representation, the state-space matrices for any ρ_t belong to a matrix polytope as

$$\begin{bmatrix}\breve{A}(\rho_t) & \breve{B}(\rho_t)\\ \breve{C}(\rho_t) & \breve{D}(\rho_t)\end{bmatrix} \in \mathcal{P} := \text{Co}\left\{\begin{bmatrix}\breve{A}_{v_i} & \breve{B}_{v_i}\\ \breve{C}_{v_i} & \breve{D}_{v_i}\end{bmatrix}, i = 1,..,2^m\right\},$$

where $\breve{A}_{v_i}$ denotes the matrix corresponding to $\breve{A}(\rho_t)$ at the i^{th} vertex of the polytope. Therefore, for any given ρ_t, the system matrices can be described as a convex combination of the matrices at the vertices of the polytope. For a polytopic LPV representation with affine dependence, we only need to solve the controller synthesis problem, detailed in the next section, with respect to the vertices of the scheduling region polytope to ensure closed-loop system stability and quadratic $\mathcal{H}_\infty$ performance for all variations of ρ_t within the polytope. Scherer showed in [13] that a similar property for LPV controller synthesis holds if dependence on ρ_t is rational. For all other dependencies, controller synthesis problem will have to be solved over a fine grid across the polytope, compromising both performance and computational complexity. To ensure affine or rational dependence on ρ_t, we recast the projection *w.r.t.* the system matrices as

$$\min_{R_i,P_i,S_i\ i=0,1,\cdots,m} \quad \mathcal{I} = \frac{1}{n}\sum_{t=1}^{n}\|Q(\theta_t) - R(\rho_t)\|_{\text{F}}^2, \quad (14)$$

where $Q(\theta_t)$ is as defined in (3)-(4), $\|\cdot\|_{\text{F}}$ represents the Frobenius norm, and

$$R(\rho_t) = R_0 + \sum_{i=1}^{m} R_i\rho_{t,i} = \begin{bmatrix}\breve{A}(\rho_t) & \breve{B}(\rho_t)\\ \breve{C}(\rho_t) & \breve{D}(\rho_t)\end{bmatrix} \quad (15)$$

for an affine approximation, or

$$R(\rho_t) = \left\{ P_0 + \sum_{i=1}^{m} P_i \cdot (\rho_{t,i})^j \right\}^{-1} \left\{ S_0 + \sum_{i=1}^{m} S_i \cdot (\rho_{t,i})^j \right\} = \begin{bmatrix} \breve{A}(\rho_t) & \breve{B}(\rho_t) \\ \breve{C}(\rho_t) & \breve{D}(\rho_t) \end{bmatrix} \tag{16}$$

for a rational approximation. Variable $j \in \mathbb{Z}_+$ is a non-negative integer. This gives the state-space matrices $\breve{A}$, $\breve{B}$, $\breve{C}$, and $\breve{D}$ of the reduced LPV model. It is important to note that this optimization problem is solved offline.

IV. Revisiting Model Reduction using Linear Decomposition

For the sake of comparison, we also revisit LPV model reduction using linear dimensionality reduction techniques like PCA. PCA transforms highly correlated data into a set of linearly uncorrelated and mutually orthogonal variables called principal components. Its main advantage lies in its linear mapping, resulting in the reduced LPV-SS matrices retaining their affine dependence. On the other hand, since in LPV models, the scheduling variables capture nonlinearities, linear methods might not be as effective as other nonlinear dimensionality reduction techniques.

For linear PCA, given the data set $\Theta = \begin{bmatrix} \theta_1, \ldots, \theta_n \end{bmatrix} = \begin{bmatrix} p(\mu_1), \ldots, p(\mu_n) \end{bmatrix} \in \mathbb{R}^{l \times n}$, PCA solves an eigenvalue problem for the covariance matrix. The reduced variables for any test point θ_t, at a given time t, are given by

$$\rho_t = V_m^\top p(\mu_t) = V_m^\top \theta_t,$$

where V_m denotes an $l \times m$ matrix whose columns contain the m eigenvectors associated with the first m significant eigenvalues. The approximation of the actual variable $\tilde{\theta}_t$, corresponding to ρ_t, can be computed as $\tilde{\theta}_t = \mathcal{C}^{-1}(V_m \rho_t)$, where $\mathcal{C}^{-1}(V_m \rho_t) = V_m \rho_t + \theta_{\text{mean}}$. The mapping matrices $\tilde{A}(\cdot)$, $\tilde{B}(\cdot)$, $\tilde{C}(\cdot)$, and $\tilde{D}(\cdot)$ of the PCA-based reduced model are related to the reconstructed scheduling variable $\tilde{\theta}_t$ by

$$\tilde{Q}(\rho_t) = \begin{bmatrix} \tilde{A}(\rho_t) & \tilde{B}(\rho_t) \\ \tilde{C}(\rho_t) & \tilde{D}(\rho_t) \end{bmatrix} \approx \begin{bmatrix} A(\tilde{\theta}_t) & B(\tilde{\theta}_t) \\ C(\tilde{\theta}_t) & D(\tilde{\theta}_t) \end{bmatrix} = Q(\tilde{\theta}_t). \tag{17}$$

We can now write the following:

$$\tilde{Q}(\rho_t) \approx Q(\tilde{\theta}_t) = Q_0 + \sum_{i=1}^{l} Q_i \left(V_m \rho_t + \theta_{\text{mean}} \right)_i,$$

giving us an affine LPV-SS model in the reduced variables $\rho_t \in \mathbb{R}^m$. For further details, see [3], [11].

V. Numerical Example

We consider the case study of a two-link planar robotic arm. The schematic of this robotic manipulator is shown in Figure 3. The nonlinear model has been taken from [14]. The lower and upper links of the robot are known as the shoulder and the arm; the arm is attached to the actuator. The two joints connect via a gear servo. The authors in [3] have re-written the robotic manipulator model in an LPV form, which is reproduced here as

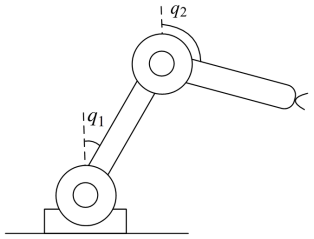

Fig. 3. Schematic of 2-DoF robotic manipulator

$$\begin{aligned}
A &= \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ cd\theta_3 & -be\theta_4 & \theta_5 & b\theta_6 \\ -bd\theta_7 & ae\theta_8 & \theta_9 & \theta_{10} \end{bmatrix}, \; C = \begin{bmatrix} I & \emptyset \end{bmatrix}, \\
B &= \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ cnk_m\theta_1 & -bnk_m\theta_2 \\ -bnk_m\theta_2 & ank_m\theta_1 \end{bmatrix}, \; D = \emptyset, \\
h &= ac - b^2\cos^2(\Delta), \\
\theta_1 &= \frac{g}{h}, \theta_2 = \frac{g\cos(\Delta)}{h}, \theta_3 = \frac{\text{sinc}(q_1)}{h}, \\
\theta_4 &= \cos\Delta \frac{\text{sinc}(q_2)}{h}, \theta_6 = \frac{-c\sin(\Delta)\dot{q_2} + \cos(\Delta)f}{h}, \\
\theta_5 &= \frac{-b^2\sin(\Delta)\ \cos(\Delta)\dot{q_1} - (c + b\cos(\Delta))f}{h}, \\
\theta_7 &= \frac{\cos(\Delta)\text{sinc}(q_1)}{h}, \theta_8 = \frac{\text{sinc}(q_2)}{h}, \\
\theta_9 &= \frac{ab\ \sin(\Delta)\dot{q_1} + f(a + b\cos(\Delta))}{h}, \\
\theta_{10} &= \frac{b^2\sin(\Delta)\cos(\Delta)\dot{q_2} - af}{h},
\end{aligned} \tag{18}$$

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. States of the robot are given by $x = [q_1\ q_2\ \dot{q_1}\ \dot{q_2}]^{\text{T}}$. The first two states represent the angles of the links with respect to vertical reference. Angular velocities are represented by the other two states of the model. The LPV model ends up with a total of $l = 10$ scheduling variables. The objective here is to reduce the number of scheduling variables to m, with $m \leq l$ that gives an LPV model suitable for gain-scheduled $\mathcal{H}_\infty$ LPV controller synthesis

A simulation model is used to generate the data from the robotic manipulator model. The scheduling variables data is collected and fed to an autoencoder for training and validation. A *satlin* and a *logsig* function is chosen as activation function for the encoding and decoding layers of the autoencoder, respectively. Values of regularization weight, sparsity weight, and other parameters, tuned by grid search to minimize the mean squared error (MSE), are tabulated in Table I. The solution for autoencoder can be nontrivial when we constrain the dimension of reduced

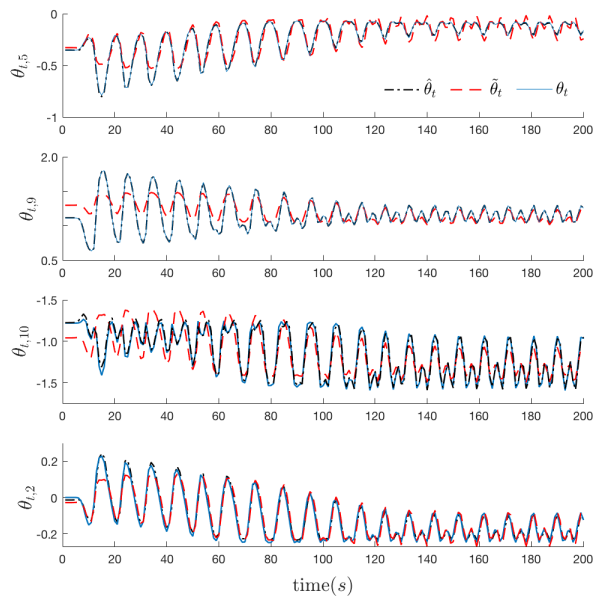

Fig. 4. Scheduling variables $\theta_{t,i}$ (blue solid line), PCA-based approximation $\tilde{\theta}_{t,i}$ (red-dashed line), and autoencoder-based approximation $\hat{\theta}_{t,i}$ (black-dotted line) with $m = 1$.

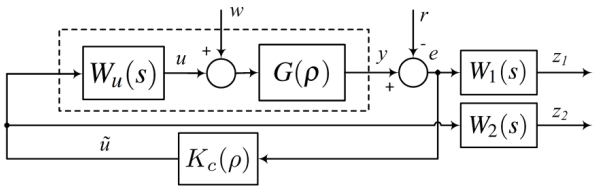

Fig. 5. Generalized configuration of closed-loop system composed of reduced LPV model, LPV controller, and design weights

variable. Here, we want to restrict the number of reduced scheduling variables, and hence, the number of neurons in the hidden layer to $m = 1$. Once trained, the decoding results give us the estimates $\hat{\theta}_t$ of the original scheduling variables θ_t, which are shown in Figure 4. For the sake of comparison, results for PCA-based model reduction with $m = 1$ are also compared. Since autoencoders do not solve an eigenvalue decomposition problem, one cannot compare the percentage data variance associated with each data component as in the case of PCA. Therefore, for the purpose of comparison, fitness scores based on MSE are compared between the true and estimated scheduling variables after running 100 simulations in a Monte-Carlo run. Simulation fitness statistics are tabulated in Table II.

To examine the closed-loop performance of LPV controllers designed using the reduced LPV models, we explore LPV controller design. We seek to design a controller described by the following state-space representation

$$K(\rho): \begin{cases} \dot{\zeta}_t = A_K(\rho)\zeta_t + B_K(\rho)y_t, \\ u_t = C_K(\rho)\zeta_t + D_K(\rho)y_t. \end{cases} \tag{19}$$

The corresponding closed-loop combination of the reduced

TABLE I

AE ACTIVATION FUNCTIONS AND PARAMETERS

1^{st} layer act. func.	satlin	$h^{(1)}(\theta) = \begin{cases} 0, \text{if } \theta \le 0 \\ \theta, \text{if } 0 < \theta < 1 \\ 1, \text{if } \theta \ge 1 \end{cases}$
Weight regularization	ψ	1×10^{-5}
Sparsity regularization	β	0.2
2^{nd} layer act. func.	logsig	$h^{(2)}(\rho) = \frac{1}{1+e^{-\rho}}$

TABLE II

MONTE-CARLO SIMULATION STATISTICS FOR SCHEDULING VARIABLES ESTIMATION FITNESS

	PCA	Autoencoder
mean MSE	2.7×10^{-3}	8.1×10^{-4}
std MSE	1.23×10^{-4}	1.301×10^{-4}

plant and the controller (19) is given by

$$G_{cl}(\rho): \begin{cases} \dot{\eta}_t = A_{cl}(\rho)\eta_t + B_{cl}(\rho)w_t, \\ z_t = C_{cl}(\rho)\eta_t + D_{cl}(\rho)w_t, \end{cases} \tag{20}$$

where $\eta_t = \begin{bmatrix} x_t^\top & \zeta_t^\top \end{bmatrix}^\top$, and all matrix functions are affine in ρ. LPV gain-scheduled controller design theory makes use of the definition of induced $\mathcal{L}_2$ gain performance, which implies that the closed-loop LPV system (20) has an induced $\mathcal{L}_2$ gain performance less than γ if there exists a symmetric positive-definite matrix X such that

$$\begin{bmatrix} A_{cl}^\top(\rho)X + XA_{cl}(\rho) & XB_{cl}(\rho) & C_{cl}^\top(\rho) \\ \star & -\gamma I & D_{cl}^\top(\rho) \\ \star & \star & -\gamma I \end{bmatrix} \prec 0 \tag{21}$$

for all admissible trajectories of ρ. The vertex property shows that the inequality (21) holds for all systems within the polytope, if it holds true at the vertices. This directly translates to computational advantage of LPV model reduction. For a reduced LPV model with a lower number of scheduling variables, the number of LMIs that need to be solved in order to design an LPV controller reduces exponentially. For details of the vertex property, see [12].

A generalized LPV control loop is shown in Figure 5. For the control of the robotic manipulator, the block $G(\rho)$ in Figure 5 represents the reduced LPV model of the manipulator. The block $K(\rho)$ represents an LPV controller described in (19), which is scheduled based on the reduced scheduling variables ρ. Variable r denotes reference trajectories while $\tilde{u}$ denotes the input to the plant augmented with $W_u(s)$; this is a low-pass augmenting filter which is a design requirement [12]. Filters $W_1(s)$ and $W_2(s)$ are loop-shaping filters, which are tuned to be first order low pass filter and a static gain, respectively. The pole of the filter is chosen so that steady state error is minimized. The overall design objective is to minimize the induced $\mathcal{L}_2$ gain from external disturbance $w_t = \begin{bmatrix} r_t^1 & r_t^2 & w_t^1 & w_t^2 \end{bmatrix}^\top$ to the controlled outputs $z_t = \begin{bmatrix} z_t^1 & z_t^2 \end{bmatrix}^\top$.

Using the autoencoder-based reduced LPV model, we solve the controller synthesis problem. An 8^{th} order con-

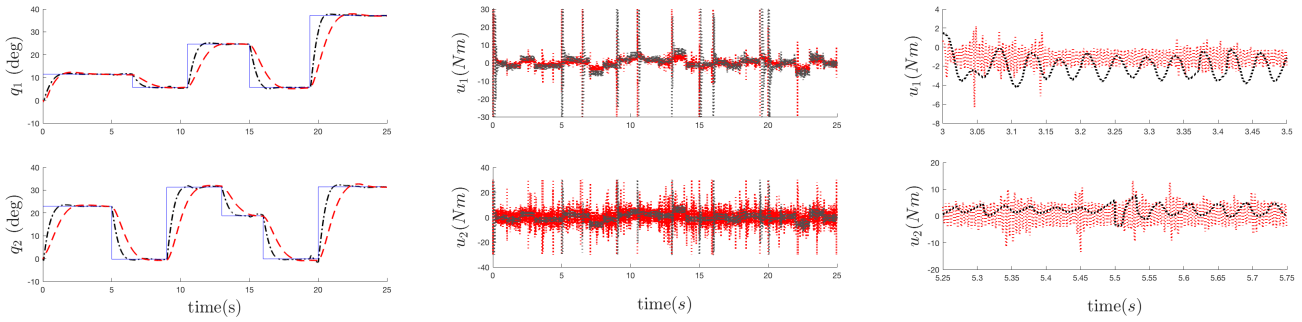

Fig. 6. (Left) Closed-loop trajectory-tracking results showing reference trajectory and system outputs; (center) closed-loop trajectory-tracking results showing controller outputs; (right) a closer look at the controller outputs. In all figures, red-dashed lines represent the case when controller is designed using PCA-based reduced model, black-dotted lines represent the autoencoder case, and reference trajectories are shown by blue solid lines.

troller is designed and a minimum value of $\gamma = 0.076$ is obtained. We also add process noise w to the system input such that a *signal-to-noise ratio* (SNR) of 20dB is maintained. To meet physical constraints on the robot link motors, a saturation limit of $|u_i| < 30$ N-m, for $i = 1, 2$, is imposed on the controller outputs.

A similar controller is designed using a PCA-based reduced model with $m = 1$. An optimal best-case value of $\gamma = 0.1909$ is achieved. Figure 6 shows reference tracking results. The reference trajectories are represented by blue solid line; PCA and autoencoder-based controlled outputs are shown by red dashed and black dotted lines, respectively. Other plots in Figure 6 show the controller outputs, and their close-up views. In general, we observe that the autoencoder-based control design gives better closed-loop performance in terms of lesser rise and settling times. The outputs of the controller also seem to chatter less, are more smooth, and have an overall lesser average power.

VI. Concluding Remarks

In this work, we have developed a new method for LPV model reduction using AEs. While model reduction in the LPV case can refer to either reduction in the number of state variables or reduction of scheduling dependency, here, we addressed the problem of reduction in the complexity of the scheduling dependency while preserving the model order. This results in a direct and exponential reduction in computational complexity while synthesizing an LPV $\mathcal{H}_\infty$ controller. Although in the past, studies involving both linear and nonlinear unsupervised dimensionality reduction techniques like PCA and its kernelized variants have been been carried out, the use of AEs adds attractive aspects to this exercise. As noted earlier, while PCA can only represent linear transformation, the activation functions in AEs introduce nonlinearities in the encoding and decoding. Additionally, in comparison to nonlinear kernel-based PCA, once the AEs train for reduction, they do not have to solve for a pre-image separately. At the same time, AEs retain the advantage of nonlinear mapping that is absent in linear techniques like PCA. Finally, for more challenging data with difficult nonlinearities, AEs have the capability to add more layers for encoding and decoding, paving the way for applicability of *deep learning* methods in LPV model reduction. A case study for a mechanical robotic arm was examined. Results were compared with standard PCA-based results as well. Improvements were observed in both scheduling variables estimation and closed-loop tracking performance.

References

[1] A. Astolfi, "Model reduction by moment matching for linear and nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 55, no. 10, pp. 2321–2336, 2010.

[2] P. Holmes, *Turbulence, coherent structures, dynamical systems and symmetry*. Cambridge university press, 2012.

[3] 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, 2008.

[4] P. Gaspar, Z. Szabo, and J. Bokor, "The design of an integrated control system in heavy vehicles based on an LPV method," in *Proc. of the 44$^{\text{th}}$ IEEE Conf. Decision and Control, and 2005 European Control Conf.*, 2005, pp. 6722–6727.

[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, 2012.

[6] S. Rizvi, J. Mohammadpour, R. Tóth, and N. Meskin, "Parameter set-mapping using kernel-based PCA for linear parameter-varying systems," in *Proc. of the 13$^{\text{th}}$ European Control Conf.*, 2014, pp. 2744–2749.

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

[8] L. Deng, D. Yu *et al.*, "Deep learning: methods and applications," *Foundations and Trends in Signal Processing*, vol. 7, no. 3–4, pp. 197–387, 2014.

[9] Y. Bengio *et al.*, "Learning deep architectures for AI," *Foundations and trends in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[10] B. Olshausen and D. Field, "Sparse coding with an overcomplete basis set: A strategy employed by v1?" *Vision research*, vol. 37, no. 23, pp. 3311–3325, 1997.

[11] S. Z. Rizvi, J. Mohammadpour, R. Tóth, and N. Meskin, "A kernel-based PCA approach to model reduction of linear parameter-varying systems," *IEEE Trans. Control Syst. Tech.*, vol. 24, no. 5, pp. 1883–1891, Sept 2016.

[12] P. Apkarian, P. Gahinet, and G. Becker, "Self-scheduled $\mathcal{H}_\infty$ control of linear parameter-varying systems: a design example," *Automatica*, vol. 31, no. 9, pp. 1251–1261, 1995.

[13] C. Scherer, "LPV control and full block multipliers," *Automatica*, vol. 37, no. 3, pp. 361–375, 2001.

[14] Z. Yu, H. Chen, and P. Woo, "Gain scheduled LPV $\mathcal{H}_\infty$ control based on LMI approach for a robotic manipulator," *J. Rob. Syst.*, vol. 19, no. 12, pp. 585–593, 2002.