Closed-Loop Identification of LPV Models Using Cubic Splines with Application to an Arm-Driven Inverted Pendulum

Sudchai Boonto and Herbert Werner

Abstract—A method for the identification of MIMO input-output LPV models in closed-loop is proposed. The model is assumed to display both linear and non-linear behavior in which the latter is dependent on the scheduling parameters, and cubic splines are used to represent the non-linear dependence. For the estimation of both linear and non-linear parameters, the separable least square method is employed. The linear parameters are obtained by a least square identification algorithm, while the non-linear parameters are obtained using a recursive Levenberg-Marquardt algorithm. To identify such a model in closed-loop, we use a non-linear version of a two-step method. A neural network ARX model will be used in the first step for two purposes. Firstly, to generate noise-free input signal to get an unbiased model and secondly to generate noise-free scheduling signal for consistent identification. The proposed method is applied to an arm-driven inverted pendulum. The resulting model is compared with a linear time-invariant model, and with an LPV model that depends polynomially on the scheduling parameters. Experimental results indicate that the cubic spline model outperforms the other ones in terms of accuracy.

I. INTRODUCTION

Linear Parameter Varying (LPV) systems [1] have received considerable attention over the last decade. This is because they can be used to model a wide class of nonlinear systems. At the same time LPV models allow the extension of linear design techniques to non-linear systems. An increasing number of applications of LPV gain-scheduling control is being reported, for example [2], [3]. However, a difficulty with this approach is that it is not trivial to construct an LPV model for a given non-linear plant. In most of the reported applications, it is assumed that an accurate physical model of the plant is available. In practice to obtain such a first principle model can be quite challenging; for this reason, the black-box identification of LPV models from measured input-output data is receiving increasing attention. Two main directions can be distinguished: One of them is based on state-space models, see e.g. [4] and more recent [5], while the other is based on input-output models, see e.g. [6]. Using the latter method with a suitable parameterisation the prediction error can be written in linear regressor form, and the problem can be turned into a least square problem. This approach has been used in several practical applications, e.g. [7], [2]. On the other hand, most available controller design techniques are based on state space models; a transformation of input-output LPV models into state-space form is therefore necessary, see [8].

For input-output LPV identification [6], the functional dependence on scheduling parameters can be expressed as a linear combination of basis functions. One that is frequently used is a polynomial in scheduling variables. This approach is simple but rather restricted, see [9], and difficult to tune. The only tuning parameter of this parameterisation is the order of the polynomial; however high order polynomials may cause oscillation. Here we consider a cubic spline function that was proposed in [10]. There are many advantages in using cubic splines instead of polynomials. One of them is that splines are defined by the position of their knots and react locally. Moving a single knot does not affect the whole function. Unlike polynomial basis functions, cubic spline functions are non-linear in their parameters, therefore non-linear optimization techniques have to be used. In this paper, we use a Separable Least Squares (SLS) algorithm to improve the numerical condition of the non-linear optimizations.

There are many situations in which identification in open-loop is difficult or even impossible, for example unstable plants and plants with integral behaviour. In these cases closed-loop system identification methods have to be used [11]. To identify an input-output discrete-time LPV model in a closed-loop, a two-step method, proposed in [12] for LTI systems, will be extended to use with LPV systems. This can be done by using a neural network as a noise filter.

The consistency of the identification is also taken care of in this work. This issue has been discussed in [13], it is shown that the scheduling parameters have to be noise free for the case of output-dependent LPV models. Again this can be solved by using a neural network as a noise filter.

The proposed method is applied to an Arm-Driven Inverted Pendulum (ADIP) [3], [14]. This plant is a single-input multiple-output (SIMO) system. A linear controller is used to stabilize the system in an initial range. A multi-sine signal is then used to excite all possible input-output levels for both outputs of the systems. Application of the proposed method and simulation results are shown.

The paper is organized as follows. In section II, the model class considered here is defined and the estimation method is presented. Section III presents the experimental results and discussion. Conclusions are drawn in Section IV.

II. INPUT-OUTPUT LPV-ARX MODELS

In this paper, we consider the following input-output discrete-time LPV model in ARX form, as discussed in [6],
\[ A(q^{-1}, p(k))y(k) = B(q^{-1}, p(k))u(k - d) + e(k) \tag{1} \]

where \( q^{-1} \) is the backward shift operator, \( p(k) \in \mathbb{R}^q \) represents the scheduling variables, \( y(k) \in \mathbb{R}^{n_y} \) is the system output at time \( k \), and \( u(k) \in \mathbb{R}^{n_u} \times 1 \) is the system input. The polynomial matrices \( A(q^{-1}, p(k)) \) and \( B(q^{-1}, p(k)) \) are defined by

\[
\begin{align*}
A(q^{-1}, p(k)) &= I_{n_y} + A_1(p(k))q^{-1} + \cdots + A_{n_a}(p(k))q^{-n_a} \\
B(q^{-1}, p(k)) &= B_1(p(k))q^{-1} + \cdots + B_{n_b}(p(k))q^{-n_b}
\end{align*}
\tag{2}
\]

where \( A_i \in \mathbb{R}^{n_y \times n_y}, B_i \in \mathbb{R}^{n_y \times n_u}. \) The elements of the coefficient matrices \( A_i \) and \( B_i \) are functions of \( p(k) \), which can be expressed as a linear combinations of a set of known fixed basis functions \( f_i \) of \( p(k) \) in the following way:

\[
\rho_i(p(k)) = \rho_i^0 + \rho_i^1 f_1(p(k)) + \cdots + \rho_i^{N-1} f_{N-1}(p(k)) \tag{3}
\]

where \( \rho_i(p(k)) \) represents either \( A_i(p(k)) \) or \( B_i(p(k)) \), \( \{\rho_i^j, j = 1, 2, \ldots, N-1\} \) are constant values, and \( \{f_j(p(k)), j = 1, 2, \ldots, N-1\} \) are functions of the online measurable variables \( p(k) \).

As in [6], the model (1) can be rewritten in linear regressor form. For MIMO systems, we form the predictor output as

\[
\hat{y}(k) = \Theta^T \varphi(k), \tag{4}
\]

where the regressor vector is given by

\[
\varphi(k) = \varphi(k) \otimes \sigma(p(k)). \tag{5}
\]

Here \( \otimes \) denotes a Kronecker product and

\[
\varphi^T(k) = [-y^T(k-1) - y^T(k-2) \cdots - y^T(k-n_a) \cdots u^T(k-1) u^T(k-2) \cdots u^T(k-n_b)]
\]

\[
\sigma(p(k)) = [1 \ p(k) \cdots f_m(p(k))],
\]

\( \Theta \in \mathbb{R}^{n_y \times (n_a + n_b)} \) is a matrix containing all coefficients to be identified:

\[
\Theta^T = [a_1 \cdots a_m \cdots a_1 \cdots a_{n_a} b_1^1 \cdots b_{n_b}^1 \cdots b_1 \cdots b_{n_b}]
\]

where \( a_i, b_i \) are the columns of the corresponding matrices \( A_i \) and \( B_i \), respectively. Given a data set

\[
Z^N \triangleq \{u(0), p(0), y(0), \ldots, u(N), p(N), y(N)\},
\tag{6}
\]

the least-squares (LS) parameter estimate [11]

\[
\text{minimizing } V_N(\theta, Z^N) = \frac{1}{N} \sum_{k=1}^{N} \| y(k) - \Theta^T \varphi(k) \|^2 \tag{7}
\]

is

\[
\hat{\Theta}_N = \left[ \frac{1}{N} \sum_{k=1}^{N} \varphi(k) \varphi^T(k) \right]^{-1} \frac{1}{N} \sum_{k=1}^{N} \varphi(k) y(k),
\tag{8}
\]

or in short matrix form:

\[
\hat{\Theta}_N = Y^T \Phi^T (\Phi \Phi)^{-1} \triangleq Y^T \Phi^1, \tag{9}
\]

where \( Y^T = [y(1) \ y(2) \cdots \ y(N)] \)

\( \Phi^T = [\varphi(1) \ \varphi(2) \cdots \ \varphi(N)] \)

and \( \Phi^1 = \Phi^T (\Phi \Phi)^{-1} \) denotes a left pseudo-inverse.

**Remark**: The structure in (4) has the advantage of structure sharing the same regressor. The form \( \hat{y} = \varphi^T(k) \theta \) or considering each output as a separate model is also possible.

A. **Polynomial basis functions and Spline functions**

In general, the exact basis functions \( f_i(p(k)) \) may not be known in advance. The basis functions can be assumed to be polynomial in the scheduling parameter \( p(k) \) [15, 7]:

\[
f_i(p(k)) = \rho_i^0 + \rho_i^1 p(k) + \rho_i^2 p(k)^2 + \cdots + \rho_i^{n_a} p(k)^{n_a},
\]

where \( n_a \) is the degree of the polynomial. It is well known that any continuous functions on a closed interval can be approximated as closely as desired by a polynomial function if the order of the polynomial is high enough. However, polynomial functions with a given order are not easily tunable, very sensitive to data variation when the order is high, and oscillatory. Moreover, changing one coefficient of the polynomial will change the shape of the entire function.

As an alternative, one can use a cubic spline function, which has the following advantages [16]:

(i) They can represent nonlinear functions with significantly less oscillation than polynomials;

(ii) a cubic spline reacts locally to a parameter change;

(iii) cubic splines can capture sharp corners better than polynomials.

The shape of cubic spline functions can be tuned by changing the knots as function parameters. A given nonlinear dependence of a model on a scheduling variable can be parameterized in cubic spline form [17] by assigning a set of knots \( \{\eta_1, \eta_2, \ldots, \eta_m\} \) to the scheduling variable \( p(k) \), which are real numbers that satisfy

\[
\eta_{m} = \eta_1 < \eta_2 < \ldots < \eta_m = \eta_{\text{max}} \tag{10}
\]

Then, the cubic spline function associated with \( p(k) \) is given by

\[
f_i(p(k)) = \rho_i^0 + \rho_i^1 p(k) + \sum_{j=2}^{m} \rho_i^{j+1} |p(k) - \eta_j|^3. \tag{11}
\]

Fixing the values of the knots beforehand, we can solve for the coefficient \( \rho_i \) by using (8).

B. **Separable Least-Squares Algorithms**

An LPV model with a cubic spline functional dependence on scheduling parameters is represented by linear coefficients \( \Theta \) and cubic spline knot positions \( \eta \). Given a set of knots, the linear coefficients can be simply calculated by LS. On the other hand, given the linear coefficients the knot values can be obtained by a nonlinear optimization. Thus the estimation problem can be cast as a Separable Least-Squares problem [18]. The performance index we are interested in here is the
sum of squared prediction errors, which can be written in matrix form:

\[ V_N(\Theta, \eta, Z^N) = \frac{1}{2N} \| (Y(k) - \Theta \Phi(\eta)) \|_2^2. \]  

(12)

where \( y(k) \) is a vector of measured output data, \( \Phi(\eta) \) is a regressor matrix and \( \Theta \Phi(\eta) \) is a vector of predicted outputs. The objective is then to find the minimizing value,

\[ [\hat{\Theta}^T, \hat{\eta}^T]^T = \arg \min_{\Theta, \eta} V_N(\Theta, \eta, Z^N). \]  

(13)

For the above problem, if \( \eta \) is known, the corresponding \( \Theta \) that minimizes \( V_N \) can be computed from (8). The problem now can be solved in two steps, where the first step is finding \( \eta \) that minimizes the function

\[ \hat{V}_N(\eta) = \frac{1}{2N} \| (I - Y^T \Phi(\eta)) \|_2^2 \]  

(14)

The second step involves finding \( Y^T \Phi \). That the original nonlinear problem can be indeed being solved by the above two-step procedure can be proven with the help of the following theorem:

**Theorem 2.1 (Golub and Percya [18]):** Assume that the matrix \( \Phi(\eta) \) has constant rank over an open set \( \Omega \subset \mathbb{R}^n \).

(i) If \( \hat{\eta} \in \Omega \) is a minimizer of \( \hat{V}(\eta) \) and \( \hat{\Theta} = \Phi(\hat{\eta})^T Y \), then \((\hat{\Theta}, \hat{\eta})\) is also a minimizer of \( V(\Theta, \eta) \).

(ii) If \( (\Theta, \eta) \) is a minimizer of \( V(\Theta, \eta) \) for \( \eta \in \Omega \), then \( \hat{\eta} \) is a minimizer of \( \hat{V}(\eta) \) in \( \Omega \) and \( \hat{V}(\hat{\eta}) = V(\hat{\Theta}, \hat{\eta}) \). Furthermore, if there is a unique \( \Theta \) among the minimizing pairs of \( V(\Theta, \eta) \), then \( \hat{\Theta} \) must satisfy

\[ \Theta = \Phi(\hat{\eta})^T Y. \]

In this work, the recursive Levenberg-Marquardt algorithm [11] is used to compute the nonlinear parameters. A recursive algorithm is required when the data set is large. To construct a Jacobian matrix of a large sampled-data set may exceed the capacity of 32-bit computer systems.

The update step of the recursive Levenberg-Marquardt algorithm is described by

\[ \hat{\eta}(k + 1) = \hat{\eta}(k) - \lambda R(k)^{-1} J_s^T \varepsilon(k) \]  

(15a)

\[ R(k) = \lambda R(k - 1) + (1 - \lambda) J_s J_s^T, \]  

(15b)

where \( R = (J_s^T J_s + \delta_k I) \), \( \varepsilon(k) \) is a prediction error at time \( k \), \( I \) is the identity matrix, \( \delta_k \) is a time-varying regularization parameter, \( \lambda \) is a forgetting factor and \( J_s \) is the Jacobian matrix of \( \eta \). See [19] for more details.

The Jacobian matrix \( J_s \) can be computed as

\[ J_s(k) = \frac{\partial \hat{y}(\Theta, \eta, k)}{\partial \eta_i}, \]  

(16)

The dependence of \( J_s \) on \( \theta_i \) must be taken into account. This can be done by first computing \( J_t \) and \( J_n \), the Jacobian with respect to the linear and nonlinear parameters, respectively. Then \( J_s \) can be obtained [20], [21] by

\[ J_s = (I - P_t) J_n \]  

(17)

where \( P_t = J_t (J_t^T J_t)^{-1} J_t^T \) is an orthogonal projection onto the columns of the linear Jacobian \( J_t \).

**Remark:** Since the algorithm contains nonlinear optimization, a good initial parameter estimate is required. In this paper we do not consider this problem - in practice we will use random guesses of the initial values of the nonlinear parameters, and then select the best of the resulting models.

### C. Estimation Algorithm

Finally the algorithm that realizes the proposed recursive algorithm for a LPV cubic spline model can be summarized as follows.

**Algorithm 1 SLS algorithm for LPV-SP**

**Require:** \( \gamma, R(0), \delta, \hat{\eta}(0), \) ischange = 1

**for** \( k = 1 \) **to** \( N \)** **do**

Use LS to find \( \Theta(k) \)

**if** ischange = 1 **then**

Compute Jacobian \( J_s \) using (17)

**end if**

Update \( R(k) \) by (15a) and (15b)

Estimate \( \hat{\eta}(k) \) using Levenberg-Marquardt

Calculate a temporary \( \hat{\Theta}(k) \) with new \( \hat{\eta} \)

Calculate the cost function, \( V_N(\hat{\Theta}(k), \hat{\eta}) \)

**if** \( V_N \) is decreased **then**

ischange = 1

**if** The change is greater than a threshold **then**

\[ \delta = \delta/\gamma \]

**end if**

\( \Theta(k) = \hat{\Theta}(k) \)

**else**

ischange = 0

Restore old values of \( \hat{\eta}(k + 1) = \hat{\eta}(k) \)

**end if**

**end for**

### III. Closed-Loop Input-Output LPV System Identification with Two-Stage Method

In this section, we discuss a method which can be used to identify an input-output LPV model of a nonlinear unstable system which has to be stabilized in closed-loop form. The LPV model considered here is an output-dependent system. Two main issues are discussed, i.e. bias model and consistency of identification. The first issue comes from the correlation between input signal and disturbance noise via the feedback loop [22]. The latter issue is because the scheduling signal is not noise free [13].

A closed-loop system is shown in Figure 1; \( r_2(k) \) is an external signal designed by the user, while \( u(k) \) and \( y(k) \) are signals measured in experiments. If we use a direct method to estimate the model, directly on the basis of input and output, data this may result in a biased model [22]. To deal with this bias, we will use the two–step method [12]. From Figure 1, the cause for a biased model is the correlation between the inputs \( u(k) \) and the disturbances \( e(k) \) acting on
the output, which also appears in the input via feedback. For linear systems we can use a linear sensitivity function to do the filter task. In this work, we propose to use a nonlinear filter, e.g. a neural network [23], to remove the disturbance term from the input for the first step. To do this, we can use a Neural Network AutoRegressive with eXogenous input (NNARX) [23] structure as a predictor model to predict the noise free input $u(k)$ by using $r_2(k)$ and $\tilde{y}(k)$ as inputs to the network as shown in Figure 2.

\[
\begin{align*}
\tilde{y}(k-1) & \quad \vdots \\
\tilde{y}(k-n) & \\
r_2(k-1) & \quad \vdots \\
r_2(k-m) & \\
\end{align*}
\]

Fig. 2. NNARX model structure for predicting noise free $u(k)$

The consistency of input-output LPV identification is investigated in [13]: to get a consistent identification, the scheduling parameters have to be noise-free for the case of an output dependent LPV model. In [13] an instrumental variable (IV) method is proposed to solve the problem. Instead of using IV method we can also use a neural network to remove the disturbance noise for the output as well. As above we can use an NNARX network to identify the whole closed–loop system by using $r_2(k)$ and $\tilde{y}(k)$ as an input signal and $y(k)$ as a predicted output signal as shown in Figure 3.

\[
\begin{align*}
\tilde{y}(k-1) & \quad \vdots \\
\tilde{y}(k-n) & \\
r_2(k-1) & \quad \vdots \\
r_2(k-m) & \\
\end{align*}
\]

Fig. 3. NNARX model structure for predicting noise free $y(k)$

**Remark 1:** We can use more complex neural network structures, for example NNARMAX, when a more flexible disturbance model is required. Moreover we can take the NNARX network as a MIMO model which can be used to predict both noise free input and noise free scheduling signals at the same time. However, the cost of computation will be increased.

**Remark 2:** Since the objective of using an NNARX network is to generate noise-free signals, we can use a large number of hidden neurons and also a high order NNARX model. However, the network should not be too complex to avoid over fitting.

Finally we can use the direct method with the noise free data set to get the input–output LPV model. The configuration for the direct method is shown in Figure 4.

\[
\begin{array}{c}
\text{LPV} \\
\uparrow \quad f(\cdot) \\
\quad \downarrow \\
y(k)
\end{array}
\]

Fig. 4. Configuration for the direct method identification.

**IV. APPLICATION TO THE ARM–DRIVEN INVERTED PENDULUM (ADIP)**

To illustrate the improvement of the MIMO LPV model with cubic splines (LPV-SP) over the polynomial model (LPV-Poly), experiments on an arm-driven inverted pendulum (ADIP) are carried out. In the experiments, we use a laboratory version of this plant manufactured by Quanser Inc. [14]. Its structure is shown in Figure 5. The pendulum is the top link hinged on the rotated arm - the bottom link - which is driven by a DC motor. The plant input is the DC voltage applied to the motor that drives the arm; controlled outputs are the angular position $\varphi_1$ of the arm, and the position of the pendulum $\varphi_2$ which is to be held at 0 degree position. When a wide range of the angle $\varphi_1$ is considered, the plant displays strong nonlinearity and an LTI model is not sufficient for controller design.
A. Experiment Setup

The closed-loop configuration is shown in Figure 6, where \( K \) is a stabilizing discrete-time LTI \( H_\infty \) controller based on a nominal model provided by Quanser [14], and an inner PI loop is used to control the angular velocity of the arm. The angular velocity \( \dot{\phi}_1 \) is measured by means of a differentiator filter FD. Then the plant to be identified here includes the PI controller and the filter FD, shown as the dashed box in Figure 6.

B. Experimental Results

The ADIP is excited by a multisine input signal with 2 periods and 24000 samples, with a sampling time of 10 ms, see more details in [24]. The first half of the data set is used for identification and the second half is for validation. For the LPV-SP model, linear LS is used in the linear step, and the recursive Levenberg-Marquardt algorithm in the nonlinear step of the SLS algorithm. In this paper we use a trial-and-error method to select the structure and the order of the model for all cases. A more systematic method has been developed in [25]. The best linear model for our sampled data set is \( n_a = [\frac{2}{9} \ 0] \), \( n_b = 3 \) and one step delay, while both LPV-poly and LPV-SP have the same structure \( n_a = [\frac{2}{9} \ 0] \), \( n_b = 5 \) and 2 steps delay.

The scheduling variable is a previous sample of the angular position \( \phi_1 \), which is the main source of the nonlinearity of this plant. The scheduling function for the polynomial case is given in vector form by

\[
\sigma_{\text{poly}}(p(k)) = \begin{bmatrix} 1 & \phi_1(k-1) & \phi_1(k-1)^2 & \phi_1(k-1)^3 \end{bmatrix}
\]

and for cubic spline

\[
\sigma_{\text{sp}}(p(k)) = \begin{bmatrix} 1 & \phi_1(k-1) & |\phi_1(k-1) - \eta_1|^3 & |\phi_1(k-1) - \eta_2|^3 \end{bmatrix}.
\]

For both cases one step delay of the scheduling variable is necessary to avoid an algebraic loop. The non–linear part of the spline function is taken to contain only three terms to reduce the dimension of the initial value space. The search is initialized randomly, and the best model is taken as solution. The value of \( \lambda \) in the nonlinear step is 0.999, and the initial value of \( \eta \) is [0.01 0.9]. Figure 7 and 8 show the comparison of the validation data and simulated (open-loop) output of all models. For the output \( \phi_1 \) both LPV models outperform the linear model while the output \( \phi_2 \) is difficult to judge from the plot. The LPV-SP model is slightly better than LPV-Poly over the whole range - this is because the cubic spline function has more freedom to fit the nonlinear scheduling function. The price to pay for this improvement is the computation time in the nonlinear step of the SLS algorithm - the increase in computation time depends on the number of nonlinear parameters. To assess the quality of the models, we use the Mean Square Error (MSE), the Best Fit (% BFT) quality of the model [11] and the Percent Variance Accounted for (% VAF) [26]. These criterions are defined...
as:

$$\text{MSE} = \frac{1}{N} \sum_{k=0}^{N-1} (y(k) - \hat{y}(k))^2,$$  
(18)

$$\text{BFT} = 100\% \times \max \left( 1 - \frac{\|y(k) - \hat{y}(k)\|_2}{\|y(k) - \bar{y}\|_2}, 0 \right),$$  
(19)

$$\text{VAF} = 100\% \times \max \left( 1 - \frac{\text{var}(y(k) - \hat{y}(k))}{\text{var}(y(k))}, 0 \right),$$  
(20)

where $N$ is the data size, $y$ is the measured data and $\hat{y}$ is the simulation data. The validation result is shown in Table I.

<table>
<thead>
<tr>
<th>Model</th>
<th>MSE</th>
<th>% BFT</th>
<th>% VAF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear $\varphi_1$</td>
<td>0.0416</td>
<td>40.5387</td>
<td>65.2372</td>
</tr>
<tr>
<td>LPV-Poly $\varphi_1$</td>
<td>0.0184</td>
<td>60.5190</td>
<td>84.4756</td>
</tr>
<tr>
<td>LPV-SP $\varphi_2$</td>
<td>0.0066</td>
<td>76.3063</td>
<td>94.5185</td>
</tr>
<tr>
<td>Linear $\varphi_2$</td>
<td>6.1780 $\times 10^{-5}$</td>
<td>25.3413</td>
<td>47.6410</td>
</tr>
<tr>
<td>LPV-Poly $\varphi_2$</td>
<td>4.3792 $\times 10^{-5}$</td>
<td>36.6924</td>
<td>70.3316</td>
</tr>
<tr>
<td>LPV-SP $\varphi_2$</td>
<td>4.4426 $\times 10^{-5}$</td>
<td>36.6924</td>
<td>70.3316</td>
</tr>
</tbody>
</table>

Table I confirms again that a LPV-SP model gives better results for the output $\varphi_1$ in all tests, while the output $\varphi_2$ is nearly the same as for the LPV-Poly model.

V. CONCLUSIONS

In this work we have shown that LPV models with cubic spline dependence on scheduling parameters can be identified using the separable least squares method in closed loop. This model class has advantages over LPV models with polynomial dependence, such as the tunability via the knot positions. Experimental results show a successful application of this approach to the ADIP system; the proposed method outperforms LTI as well as LPV-polynomial models.

REFERENCES


