Received: 30 April 2019 | Revised: 2 July 2020 | Accepted: 3 August 2020

DOI: 10.1002/rnc.5198

SPECIAL ISSUE ARTICLE

WILEY

Identification of hybrid and linear parameter-varying models via piecewise affine regression using mixed integer programming

Manas Mejari[1] | **Vihangkumar V. Naik**[2] | **Dario Piga**[1] | **Alberto Bemporad**[3]

[1]IDSIA, Dalle Molle Institute for Artificial Intelligence SUPSI-USI, Manno, Switzerland

[2]ODYS S.r.l., Lucca, Italy

[3]IMT School for Advanced Studies, Lucca, Italy

Correspondence
Manas Mejari, IDSIA, Dalle Molle Institute for Artificial Intelligence SUPSI-USI, 6928 Manno, Switzerland.
Email: manas.mejari@idsia.ch

Summary

This article presents a two-stage algorithm for *piecewise affine* (PWA) regression. In the first stage, a moving horizon strategy is employed to simultaneously estimate the model parameters and to classify the training data by solving a small-size mixed-integer quadratic programming problem. In the second stage, linear multicategory separation methods are used to partition the regressor space. The framework of PWA regression is adapted to the identification of *PWA AutoRegressive with eXogenous input* (PWARX) models as well as *linear parameter-varying* (LPV) models. The performance of the proposed algorithm is demonstrated on an academic example and on two benchmark experimental case studies. The first experimental example concerns modeling the placement process in a pick-and-place machine, while the second one consists in the identification of an LPV model describing the input-output relationship of an electronic bandpass filter with time-varying resonant frequency.

KEYWORDS

hybrid system identification, linear parameter-varying models, mixed integer programming, PWA regression

1 | INTRODUCTION

Piecewise affine (PWA) models are defined by a collection of simple affine functions, each associated to a polyhedral region of the input (or regressor) space. PWA regression aims at fitting a PWA model to a set of training data. Such modeling paradigm can be used to describe systems which change their dynamics due to, for example: saturations, thresholds, dead-zones, logic rules, abrupt changes of the working environment (for example, robot manipulators which alternate between free and contact motion), and so on. Thanks to their universal approximation property, PWA maps are able to approximate any nonlinear function with arbitrary accuracy,[1] making PWA models also suitable to describe nonlinear systems that do not necessarily exhibit a switching behavior. Furthermore, because of the equivalence between PWA and hybrid linear models[2-4] (such as *mixed logical dynamical* and linear complementarity models), well-settled tools for analysis and control of hybrid systems can be applied to systems represented in PWA form.[3,5]

Learning PWA models from data is an NP-hard problem,[6] which requires to estimate both the parameters defining the local affine functions and the partition of the regressor space. Several algorithms/heuristics have been developed in the last years[7,8] for PWA regression or, in general, for data-driven modeling of hybrid systems. Among these

methods, we mention set-membership approaches[9,10] which assume that the noise corrupting the output observations is bounded (with known bound) in amplitude or energy; algorithms based on sparse optimization,[11-13] which formulate an over-parametrized least-squares problem with a LASSO-like regularization term penalizing the number of switches, and mixed-integer programming method.[14] The computation complexity of the latter algorithm increases with the number of affine submodels (or "modes") defining the PWA models and with the size of the training dataset, as the number of integer optimization variables proportionally increases with both these parameters. Hence, this approach is limited to small/medium-size identification problems. However, although computationally expensive, this approach gives a global optimal solution. A trade-off between optimality and complexity can be made using the two-stage cluster-based heuristics,[15-20] which work as follows. In the first stage, the training samples are clustered and the affine submodel parameters are estimated. Specifically, clusters are formed by assigning each regressor to a specific submodel according to a given criterion. At a second stage, linear classification techniques (eg, multicategory support vector machines with linear kernels) are used to separate the clusters, thus partitioning the input domain into polyhedral regions.

In this article, we present a two-stage regularized moving-horizon algorithm for PWA regression which combines the advantages of the cluster-based method[19] and the integer programming approach.[14] In the first stage, the training input-output pairs are processed iteratively. At this stage, the parameters of the local models are estimated recursively along with the sequence of active modes. More specifically, at each index k, a moving-horizon window of length N_p containing past training samples from index $k - N_p + 1$ to index k is considered. A small-size *mixed-integer quadratic-programming* (MIQP) problem is formulated to find both the optimal sequence of active modes within the considered window and the parameters of the local affine submodels. According to a moving-horizon strategy, once the solution of the formulated MIQP problem is computed, only the optimal active mode at index k is kept (namely, the last active mode within the considered horizon). Then, the window is shifted forward to process the next input-output pair. The length N_p of the window balances the trade-off between computation complexity (namely, number of binary optimization variables) against estimation accuracy (namely, number of training data points contained in the window). Thus, the length N_p of the horizon acts as a knob to combine the advantages of the two-stage algorithm[19] (computational efficiency and iterative processing of the training samples) and the advantages of the mixed-integer programming approach[14] (nondecoupled optimization over the active modes and the submodel parameters). In order to overcome the limitation of short horizon length, past data outside of the window is also taken into account in estimating the model parameters by adding a regularization term in the formulated MIQP problem. At the second stage, the regressor space is partitioned into polyhedral regions based on the optimal sequence of active modes estimated in the first stage. The partitions are computed using linear multicategory discrimination techniques.

The proposed PWA regression algorithm is properly adapted for the identification of *PWA autoRegressive with eXogenous inputs* (PWARX) models and *linear parameter-varying* (LPV) models. The latter class represents a natural extension of *linear time-invariant (LTI)* models, where the dynamic relation between input and output signals is linear and can vary over time according to a measurable signal, the so-called scheduling variable denoted by p.[21] For the identification of LPV models, the proposed PWA regression algorithm is used to approximate the time-varying coefficients of the LPV model as a PWA function of the scheduling variable p. We remark that in conventional parametric LPV identification approaches,[22-24] the underlying p-dependence of the model coefficients are parameterized as a linear combination of known nonlinear basis functions (eg, polynomial) of the scheduling variable p. The selection of appropriate basis functions is still an open issue, which has been partly solved by nonparametric methods,[25-27] where "kernel functions" are employed to capture the underlying scheduling-variable dependencies. However, major challenges still lie in tuning the hyper-parameters defining the kernels, as well as in the computational complexity of the algorithms. In the proposed PWA approach, the underlying scheduling dependencies of the model coefficients are reconstructed via PWA maps, without the need of tuning any critical kernel hyper-parameters.

The effectiveness of the presented PWA regression algorithm is demonstrated via an academic example using synthetic data and two benchmark case studies using experimental data. Specifically, the first benchmark case study concerns the identification of a PWARX model describing the dynamic behavior of a placement process in a pick-and-place machine.[28] In the second case study, experimental data are used to estimate an LPV model describing the input-output behavior of an electronic bandpass filter with time-varying resonant frequency.[29]

We remark that a preliminary version of the material presented in this manuscript appeared in a conference paper by the authors.[30] This article discusses the extension to the identification of PWARX and LPV models, besides providing a richer case study analysis. The rest of this article is organized as follows. The PWA regression problem is formulated in Section 2. First, a general PWA regression problem is presented, while more specific problems of identification of PWARX and LPV models are discussed in Sections 2.2 and 2.3, respectively. Section 3 presents the developed moving-horizon identification approach, with a description of the strategy used to simultaneously cluster the training observations and update the model parameters (Section 3.1), and the multicategory discrimination algorithm for partitioning the input space (Section 3.2). The three case studies are discussed in Section 4.

2 | PROBLEM FORMULATION

In this section, we first provide an introduction to the general problem of PWA regression. Then, we discuss its application to the identification of (i) dynamical models in a PWARX form and (ii) LPV models whose dependence of the model coefficients on the scheduling signal is approximated by a PWA map.

2.1 | PWA regression

Consider a vector-valued PWA function $f : \mathcal{X} \to \mathbb{R}^{n_y}$ defined as

$$f(x) = \begin{cases} A_1 \begin{bmatrix} 1 \\ x \end{bmatrix} & \text{if } x \in \mathcal{X}_1, \\ \quad \vdots & \\ A_s \begin{bmatrix} 1 \\ x \end{bmatrix} & \text{if } x \in \mathcal{X}_s, \end{cases} \tag{1a}$$

where $x \in \mathcal{X} \subseteq \mathbb{R}^{n_x}$, $s \in \mathbb{N}$ is the number of modes (ie, the number of affine functions defining f), $A_i \in \mathbb{R}^{n_y \times (n_x+1)}$ are parameter matrices, and the sets $\mathcal{X}_i \subseteq \mathbb{R}^{n_x}$ (with $i = 1, \dots, s$) are polyhedra defined by linear inequality constraints, namely:

$$\mathcal{X}_i \doteq \{x \in \mathbb{R}^{n_x} : \ H_i x \leq D_i\}, \tag{1b}$$

with H_i and D_i being real-valued matrices that form a complete polyhedral partition[1] of the space $\mathcal{X}$. Note that f is not assumed to be continuous over the boundaries of the polyhedra $\{\mathcal{X}_i\}_{i=1}^s$. Therefore, f might take multiple values at the boundaries of $\{\mathcal{X}_i\}_{i=1}^s$. In order to avoid ambiguities when evaluating f on overlapping boundaries, some inequalities in (1b) can be replaced by strict inequalities.

We want to address the following problem of PWA regression.

Problem 1 (PWA regression). Consider the data-generating system:

$$y(k) = f_o(x(k)) + e_o(k), \tag{2}$$

where $e_o(k) \in \mathbb{R}^{n_y}$ is a zero-mean additive random noise independent of the input $x(k)$, and $f_o : \mathcal{X} \to \mathbb{R}^{n_y}$ is an unknown function, possibly discontinuous, and not necessarily PWA. The PWA regression problem aims at finding a PWA approximation f of the unknown function f_o based on a set of N input-output pairs $\{x(k), y(k)\}_{k=1}^N$.

Computing a PWA estimate f of the true function f_o thus requires: (i) choosing the number of modes s, (ii) computing the parameter matrices $\{A_i\}_{i=1}^s$ defining the local affine functions, and (iii) finding the polyhedral partition $\{\mathcal{X}_i\}_{i=1}^s$ of the input space $\mathcal{X}$.

[1] A collection $\{\mathcal{X}_i\}_{i=1}^s$ is a complete partition of the set $\mathcal{X}$ if $\cup_{i=1}^s \mathcal{X}_i = \mathcal{X}$ and $\mathring{\mathcal{X}}_i \cap \mathring{\mathcal{X}}_j = \emptyset$, $\forall i \neq j$, with $\mathring{\mathcal{X}}_i$ denoting the interior of $\mathcal{X}_i$.

Note that the polyhedra $\{\mathcal{X}_i\}_{i=1}^s$ should be estimated directly from the available data $\{x(k), y(k)\}_{k=1}^N$, along with the parameter matrices $\{A_i\}_i^s$. In the simpler case when the partition $\{\mathcal{X}_i\}_{i=1}^s$ is given or a-priori chosen by the user, Problem 1 reduces to independently estimating s affine models through simple linear regression.

Remark 1. When choosing the number of modes s, one must take into account the tradeoff between complexity of the overall PWA model and data fitting. Indeed, for small values of s, the PWA map f may not be able to capture the shape of the nonlinear function f_o. On the other hand, increasing the number of modes also increases the degrees of freedom in the description of the PWA map f, which may cause overfitting and poor generalization to unseen data (ie, the final estimate is sensitive to the noise corrupting the observations), besides increasing the complexity of the estimation procedure and of the resulting PWA model. In this work, we assume that s is fixed by the user. The value of s can be chosen via cross validation, with an upper-bound dictated by the maximum tolerable complexity of the model.

The PWA regression Problem 1 is quite general and, as described in the following paragraphs, covers a more specific case of identification of dynamical systems with a PWARX structure. Furthermore, as discussed in Section 2.3, the problem can be extended to the identification of LPV models, where the unknown model coefficients are described by PWA functions of the scheduling signal.

2.2 | Identification of PWARX models

Model (1a) represents a *multi-input multi-output* (MIMO) PWARX discrete-time dynamical system if the regressor vector $x(k)$ at the sampling step k is a collection of past input and output observations of a dynamical system, that is,

$$x(k) = [y'(k-1)\ y'(k-2)\ \ \dots\ \ y'(k-n_a)\ u'(k-1)\ u'(k-2)\ \cdots\ u'(k-n_b)]', \tag{3}$$

where $u(k) \in \mathbb{R}^{n_u}$ and $y(k) \in \mathbb{R}^{n_y}$ denote the observed input and output signals at time k, respectively, and n_b and n_a are the input and output lags defining the dynamical order of the model.

2.3 | Identification of LPV models

LPV systems extend the concept of LTI systems, with a linear dynamic relation between input and output signals. However, unlike LTI systems, such a linear relation can change over time according to a measurable time-varying signal $p(k) \in \mathbb{R}^{n_p}$, the so-called *scheduling variable*. This can be an endogenous signal of the process or a collection of external variables, such as space coordinates, measurable disturbances that change the dynamics of the system, or parameters used to describe changing operating conditions.

The identification problem of MIMO LPV systems can be formulated as the PWA regression Problem 1 by adopting the following MIMO LPV-ARX form

$$y(k) = a_0(p(k)) + \sum_{j=1}^{n_a} a_j(p(k))y(k-j) + \sum_{j=1}^{n_b} a_{j+n_a}(p(k))u(k-j) + e(k), \tag{4}$$

where $a_j(p(k)), j = 0, \dots, n_a + n_b$, are PWA functions of $p(k)$ defined as:

$$a_j(p(k)) = \begin{cases} A_1^j(p(k)) & \text{if } p(k) \in \mathcal{P}_1, \\ \vdots & \\ A_s^j(p(k)) & \text{if } p(k) \in \mathcal{P}_s, \end{cases} \tag{5}$$

and $p(k) \in \mathcal{P} \subseteq \mathbb{R}^{n_p}$ is the value of the scheduling variable at time k. By imposing that each entry of the matrix $A_i^j(p(k))$ depends affinely on the scheduling variable $p(k)$, the LPV identification problem reduces to the construction of a PWA mapping of the p-dependent coefficient functions $\{a_j(p(k))\}_{j=0}^{n_a+n_b}$ over a polyhedral partition $\{\mathcal{P}_i\}_{i=1}^s$ of the scheduling variable set $\mathcal{P}$. In this case, the model is estimated from an N-length sequence of regressor/output observations $\{x(k), y(k)\}_{k=1}^N$,

where the regressor is defined as $x(k) = [y'(k-1) \ \dots \ y'(k-n_a) \ u'(k-1) \ \dots \ u'(k-n_b)]' \otimes [1 \ p'(k)]'$, with $\otimes$ denoting the Kronecker product.

Remark 2. In case the submodel matrices $A_i^j(p(k))$ in (5) (with $i=1, \dots, s$ and $j=0, \dots, n_a+n_b$) do not depend on the scheduling signal $p(k)$ (ie, the p-dependent functions $a_j(p(k))$ are piecewise constant), the LPV model (4)-(5) becomes a switched linear model, where each subsystem is LTI and the scheduling signal $p(k)$ governs the switches.

Remark 3. Since the polyhedral partition $\{\mathcal{P}_i\}_{i=1}^s$ is not fixed a priori, the underlying dependencies of the functions $a_j(p(k))$ on the scheduling variable $p(k)$ are directly reconstructed from data. This represents one of the main advantages w.r.t. to widely used parametric LPV identification approaches,[21,22] which would require to parametrize $a_j(p(k))$ as a linear combination of some a-priori specified basis functions, like polynomial or trigonometric functions. Indeed, the choice of the number and of the type of basis functions is a critical issue to be addressed to guarantee both flexibility of the model and generalization properties.

3 | PWA REGRESSION ALGORITHM

In this section, we describe the algorithm for PWA regression, which consists of two stages:

S1. Recursive *estimation* of the submodel parameters $\{A_i\}_{i=1}^s$ (Equation (1a)) and simultaneous *clustering* of the regressor $\{x(k)\}_{k=1}^N$ (for PWARX models) or scheduling variables $\{p(k)\}_{k=1}^N$ (for LPV models).
S2. *Computation of a polyhedral partition* of the regressor space $\mathcal{X}$ (for PWARX models) or of the scheduling variable space $\mathcal{P}$ (for LPV models) using multicategory linear separation methods.

In the following sections, stages **S1** and **S2** are described in detail.

3.1 | Stage 1: Recursive estimation and simultaneous clustering

Stage **S1** is carried out through a regularized moving-horizon identification algorithm. The training regressor/output pairs $\{x(k), y(k)\}$ are processed iteratively. At each time sample k, a moving-horizon window of length N_p containing regressor/output pairs from time $k-N_p+1$ to time k is considered. The model parameters A_i and the active mode $\sigma(k)$ at time k are estimated simultaneously by solving the mixed-integer programming problem:

$$\min_{\substack{\{A_i\}_{i=1}^s \\ \{\delta_i(k-t)\}_{i=1,t=0}^{s,N_p-1}}} \quad \sum_{i=1}^{s}\sum_{t=0}^{N_p-1}\left\| \left(y(k-t) - A_i \begin{bmatrix} 1 \\ x(k-t) \end{bmatrix} \right) \delta_i(k-t) \right\|^2 \tag{6a}$$

$$+\gamma_1 \sum_{t=1}^{k-N_p} \left\| y(t) - A_{\sigma(t)} \begin{bmatrix} 1 \\ x(t) \end{bmatrix} \right\|^2 \tag{6b}$$

$$+\gamma_2 \sum_{i=1}^{s}\sum_{t=0}^{N_p-1} \|(x(k-t)-c_i)\delta_i(k-t)\|^2 \tag{6c}$$

$$\text{s.t.} \quad \delta_i(k-t) \in \{0,1\}, \sum_{i=1}^{s}\delta_i(k-t) = 1, \ t = 0, \dots, N_p-1, \tag{6d}$$

where $\gamma_1, \gamma_2 \geq 0$ are regularization parameters which can be tuned by the user via cross-validation, and c_i denotes the *centroid* of cluster C_i defined as $c_i = \frac{1}{|C_i|}\sum_{x(k)\in C_i} x(k)$ with $|C_i|$ denoting the cardinality of C_i. The active mode $\sigma(k) \in \{1, \dots, s\}$ represents the cluster $C_{\sigma(k)}$ which the regressor $x(k)$ is assigned to, and it is extracted from the optimizer of problem (6), that is,

$$\sigma(k-t) = i^*, \quad \text{with} \ \ i^* : \delta_{i^*}(k-t) = 1, \quad t = 0, \dots, N_p-1. \tag{7}$$

According to a moving-horizon strategy, once the solution of the mixed-integer programming problem (6) is obtained, only the active mode $\sigma(k)$ at time k is kept (namely, the last active mode within the considered horizon) among the computed sequence $(\sigma(k-N_p+1), \dots, \sigma(k))$. Then, the N_p-length window is shifted forward to process the next pair $\{x(k+1), y(k+1)\}$. In processing last N_p measurements, the active modes $\sigma(N-N_p+1), \dots \sigma(N)$ are simply the integer solution of the MIQP problem (6) solved at $k=N$.

Problem (6) computes the model parameters A_i and the sequence of active modes within the considered horizon that best match the data up to time k. Note that the term (6a) involves only the observations within the N_p-step time window, and both the model parameters and the sequence of active modes $\{\sigma(t)\}_{t=k-N_p+1}^{k}$ are optimized. The term (6b) takes into account the history of the observations outside the considered time window and it basically acts as a regularization term on the parameters A_i. Indeed, the sequence of active modes from time 1 up to $k-N_p$ is not optimized in (6b), but it is fixed to the values $\{\sigma(t)\}_{t=1}^{k-N_p}$ obtained from the solutions computed at the previous sampling instances of the moving horizon algorithm. Instead, as previously remarked, the sequence of active modes is optimized only within the considered time horizon in (6a) and in (6c). The term (6c) penalizes the distance of the regressor vector $x(k)$ from the centroid c_i of the cluster $\mathcal{C}_i$. This regularization term is added to improve the clustering performance, as it takes into account the assumption that regressors "close" to each other belong to the same clusters.

Note that decreasing N_p reduces the number of binary decision variables δ_i in problem (6), thus reducing its computational complexity. However, this introduces suboptimality as it limits the information used to cluster the regressor $x(k)$. In fact, the active modes are optimized only within a limited time window, while the other modes are fixed in (6b) to the previous optimized values. The length N_p of the horizon thus acts as a knob to control the tradeoff between complexity of the optimization problem (6) versus accuracy in estimating the model parameters A_i and in clustering the regressor $x(k)$.

3.1.1 | Recursive update of the objective function

Note that, at a first glance, the regularization cost (6b) requires to use, and thus to store, the whole time-history of observations up to time $k-N_p$ (ie, the sequence of regressor/output pairs $\{x(t), y(t)\}_{t=1}^{k-N_p}$). Nevertheless, once a new observation is available at time k, the term (6b) can be recursively updated, as described in the following paragraph. This allows us to avoid storing the whole time history and to reduce the number of operations required to construct the term (6b) at every sampling instance k.

To this end, the regularization term (6b) is rewritten as

$$\sum_{t=1}^{k-N_p} \mathrm{tr}\left(\left(y(t) - A_{\sigma(t)}\begin{bmatrix}1\\ x(t)\end{bmatrix}\right)\left(y(t) - A_{\sigma(t)}\begin{bmatrix}1\\ x(t)\end{bmatrix}\right)'\right) \tag{8a}$$

$$= \mathrm{tr}\left(\sum_{t=1}^{k-N_p}\left(y(t) - A_{\sigma(t)}\begin{bmatrix}1\\ x(t)\end{bmatrix}\right)\left(y(t) - A_{\sigma(t)}\begin{bmatrix}1\\ x(t)\end{bmatrix}\right)'\right)$$

$$= \mathrm{tr}\left(\sum_{t=1}^{k-N_p} A_{\sigma(t)}\begin{bmatrix}1\\ x(t)\end{bmatrix}\begin{bmatrix}1\\ x(t)\end{bmatrix}' A'_{\sigma(t)}\right) - 2\mathrm{tr}\left(\sum_{t=1}^{k-N_p} A_{\sigma(t)}\begin{bmatrix}1\\ x(t)\end{bmatrix} y(t)'\right) + \mathrm{tr}\left(\sum_{t=1}^{k-N_p} y(t)y(t)'\right), \tag{8b}$$

with $\mathrm{tr}(\cdot)$ denoting the matrix trace.

We then introduce the following matrices

$$H_i(k-N_p) = \sum_{t=1}^{k-N_p}\begin{bmatrix}1\\ x(t)\end{bmatrix}\begin{bmatrix}1\\ x(t)\end{bmatrix}'\delta_i(t), \tag{9a}$$

$$F_i(k-N_p) = \sum_{t=1}^{k-N_p}\begin{bmatrix}1\\ x(t)\end{bmatrix} y(t)'\delta_i(t), \tag{9b}$$

with

$$\delta_i(t) = \begin{cases} 1 & \text{if } \sigma(t) = i, \\ 0 & \text{otherwise.} \end{cases} \tag{9c}$$

Substituting (9) into the cost (8b), (6b) can be rewritten as

$$\begin{aligned} &\sum_{t=1}^{k-N_p} \left\| y(t) - A_{\sigma(t)} \begin{bmatrix} 1 \\ x(t) \end{bmatrix} \right\|^2 \\ &= \text{tr}\left(\sum_{t=1}^{k-N_p} A_{\sigma(t)} \begin{bmatrix} 1 \\ x(t) \end{bmatrix} \begin{bmatrix} 1 \\ x(t) \end{bmatrix}' A'_{\sigma(t)} \right) - 2\text{tr}\left(\sum_{t=1}^{k-N_p} A_{\sigma(t)} \begin{bmatrix} 1 \\ x(t) \end{bmatrix} y(t)' \right) + \text{tr}\left(\sum_{t=1}^{k-N_p} y(t)y(t)' \right), \\ &= \text{tr}\left(\sum_{i=1}^{s} A_i H_i(k-N_p) A'_i \right) - 2\text{tr}\left(\sum_{i=1}^{s} A_i F_i(k-N_p) \right) + \text{tr}\left(\sum_{t=1}^{k-N_p} y(t)y(t)' \right). \end{aligned} \tag{10}$$

Note that the matrices $H_i(k-N_p)$ and $F_i(k-N_p)$ can be computed recursively as

$$H_i(k-N_p) \quad = H_i(k-N_p-1) + \begin{bmatrix} 1 \\ x(k-N_p) \end{bmatrix} \begin{bmatrix} 1 \\ x(k-N_p) \end{bmatrix}' \delta_i(k-N_p), \tag{11a}$$

$$F_i(k-N_p) \quad = F_i(k-N_p-1) + \begin{bmatrix} 1 \\ x(k-N_p) \end{bmatrix} y(k-N_p)' \delta_i(k-N_p). \tag{11b}$$

Thus, at each sampling instance, the matrices $H_i(k-N_p)$ and $F_i(k-N_p)$ can be recursively updated through (11), without the need to store the whole time history $\{x(t), y(t)\}_{t=1}^{k-N_p-1}$.

3.1.2 | MIQP formulation

We cast (6) as an MIQP problem by defining the vector $z_i(k) \in \mathbb{R}^{n_y}$ such that

$$z_i(k) = \begin{cases} y(k) - A_i \begin{bmatrix} 1 \\ x(k) \end{bmatrix} & \text{if } \delta_i(k) = 1 \\ 0 & \text{if } \delta_i(k) = 0 \end{cases}, \tag{12a}$$

which can be represented as

$$z_i(k) = \left(y(k) - A_i \begin{bmatrix} 1 \\ x(k) \end{bmatrix} \right) \delta_i(k), \tag{12b}$$

where $\delta_i(k) \in \{0, 1\}$. Let the elements of the vector $y(k) - A_i \begin{bmatrix} 1 \\ x(k) \end{bmatrix}$ be bounded by an upper bound M and a lower bound m, such that

$$m\mathbf{1}_{n_y} \leq y(k) - A_i \begin{bmatrix} 1 \\ x(k) \end{bmatrix} \leq M\mathbf{1}_{n_y}, \tag{12c}$$

where $\mathbf{1}_{n_y}$ is an n_y-dimensional vector of ones.

By utilizing (12a) along with recursive update terms (10), problem (6) can be equivalently written as the following MIQP problem

$$\min_{\substack{\{A_i\}_{i=1}^{s} \\ \{\delta_i(k-t)\}_{i=1,t=0}^{s,N_p-1} \\ \{z_i(k-t)\}_{i=1,t=0}^{s,N_p-1}}} \sum_{t=0}^{N_p-1}\sum_{i=1}^{s}(z_i(k-t))'z_i(k-t) + \gamma_1 \mathrm{tr}\left(\sum_{i=1}^{s} A_i H_i(k-N_p)A_i'\right) - \gamma_1 2\mathrm{tr}\left(\sum_{i=1}^{s} A_i F_i(k-N_p)\right) \tag{13a}$$

$$+\gamma_2 \sum_{i=1}^{s}\sum_{t=0}^{N_p-1} \|(x(k-t)-c_i)\delta_i(k-t)\|^2, \tag{13b}$$

$$\text{s.t.} \quad z_i(k-t) \le M\mathbf{1}_{n_y}\delta_i(k-t), \tag{13c}$$

$$z_i(k-t) \ge m\mathbf{1}_{n_y}\delta_i(k-t), \tag{13d}$$

$$z_i(k-t) \le y(k-t) - A_i\begin{bmatrix}1\\ x(k-t)\end{bmatrix} - m\mathbf{1}_{n_y}(1-\delta_i(k-t)), \tag{13e}$$

$$z_i(k-t) \ge y(k-t) - A_i\begin{bmatrix}1\\ x(k-t)\end{bmatrix} - M\mathbf{1}_{n_y}(1-\delta_i(k-t)), \tag{13f}$$

$$\sum_{i=1}^{s}\delta_i(k-t) = 1,\ t = 0, \dots, N_p-1, \tag{13g}$$

$$\delta_i(k-t) \in \{0,1\},\ i = 1, \dots, s. \tag{13h}$$

In order to solve small-scale MIQP problems (13h), we can use, for example, the accelerated dual gradient projection (GPAD)[31] coupled with *Branch and Bound* (B&B) method, denoted as miqpGPAD.[32] The B&B algorithm provides a structured methodology to possibly avoid exploring all integer combinations. This algorithm in a basic form relies on sequentially partitioning the integer feasible solution space into small subsets or subregions, and to search the space of all feasible solutions. An optimization problem is required to be solved corresponding to each subregion. Employing a B&B algorithm for MIQP problems relies on efficient solution of the quadratic programming (QP) problems obtained from relaxation of integer constraints.

The GPAD algorithm involves only basic arithmetic operations, which makes it very simple to code and particularly suitable for implementation on embedded platforms. These features make the GPAD-based MIQP solver particularly suitable for online applications, where a model needs to be updated once new data becomes available. The details on the efficient implementation of miqpGPAD algorithm are provided in Section 3 of the accompanying technical report.[33]

In addition, a generic framework to warm-start the binary variables can be utilized for efficient solution of MIQP problems (13h),[34] as described in the following. The shifted binary values optimized at the previous step k can be used as the binary warm start for subsequent MIQP at time step $k+1$. Specifically, let $\{\delta^*(k-N_p+1|k)\ \dots\ \delta^*(k|k)\}$ be the sequence of optimal solution computed for the binary variables at time k, then we set $\delta(k-N_p+1|k+1) = \delta^*(k-N_p+2|k), \dots, \delta(k-1|k+1) = \delta^*(k|k)$ as a binary warm-start. For example, let $N_p = 3$ and suppose the optimized binary sequence at time k is $(1,0,0)$. Then, at time $k+1$, $(0,0,\star)$ is set as a binary warm start. This further increases efficiency for the recursive solution of the formulated MIQP problems.

3.1.3 | Summary and iterative refinement

The steps of Stage **S1** described so far for the estimation of submodel parameters $\{A_i\}_{i=1}^{s}$ and for simultaneous *clustering* of the regressor $\{x(k)\}_{k=1}^{N}$ are summarized in Algorithm 1.

Algorithm 1. Recursive estimation of model parameters and simultaneous clustering of the regressors

Input: Observations sequence $\{x(k), y(k)\}_{k=1}^{N}$; number of modes s; horizon N_p; initial clusters $\mathcal{C}_i$ and centroids c_i.

1. **let** $H_i(0) \leftarrow 0$, $F_i(0) \leftarrow 0$, $\mathcal{C}_i \leftarrow \emptyset$, $i = 1, \dots, s$;
2. **let** $k \leftarrow N_p$;
3. **solve** the MIQP problem (13);
4. **let** $\{\delta_i^*(t)\}_{i=1,t=1}^{s,N_p}$ be the optimal parameters minimizing (13);
5. **for** $t = 1, \dots, N_p$ **do**
   5.1. **let** $i^*(t)$ be the index such that $\delta_{i^*}^*(t) = 1$;
   5.2. **let** $\sigma(t) \leftarrow i^*(t)$;
   5.3. **let** $\mathcal{C}_{\sigma(t)} \leftarrow \mathcal{C}_{\sigma(t)} \cup \{x(t)\}$;
6. **end for**
7. **for** $k = N_p + 1, \dots, N$ **do**
   7.1. **update** the matrices $H_i(k - N_p)$ and $F_i(k - N_p)$ through (11);
   7.2. **solve** the MIQP problem (13);
   7.3. **let** $A_i^*(k)$ be the optimal parameters minimizing (13), $i = 1, \dots, s$;
   7.4. **let** $\{\delta_i^*(k - t)\}_{t=0}^{N_p-1}$ be the optimal parameters minimizing (13), $i = 1, \dots, s$;
   7.5. **let** i^* be the index such that $\delta_{i^*}^*(k) = 1$;
   7.6. **let** $\sigma(k) \leftarrow i^*$;
   7.7. **let** $\mathcal{C}_{\sigma(k)} \leftarrow \mathcal{C}_{\sigma(k)} \cup x(k)$;
   7.8. **let** $\delta c_{\sigma(k)} \leftarrow \frac{1}{|\mathcal{C}_{\sigma(k)}|}(x(k) - c_{\sigma(k)})$;
   7.9. **update** the centroid $c_{\sigma(k)}$ of cluster $\mathcal{C}_{\sigma(k)}$ as $c_{\sigma(k)} \leftarrow c_{\sigma(k)} + \delta c_{\sigma(k)}$;
8. **end for**;

Output: Estimated parameters $A_1^*(N), \dots, A_s^*(N)$; clusters $\mathcal{C}_1, \dots, \mathcal{C}_s$; sequence of active modes $\{\sigma(k)\}_{k=1}^{N}$.

At the beginning of Algorithm 1, MIQP problem (13) is solved (step 3), without considering the second and third terms in the cost (13b) by initializing $H_i(0) = 0$ and $F_i = 0$ (step 1). Thus, from step 2 to step 6, a mini-batch identification problem is solved to estimate the sequence of active modes $\sigma(t)$ from time 1 up to time N_p and to assign the regressor $\{x(t)\}_{t=1}^{N_p}$ to the cluster $\{\mathcal{C}_{\sigma(t)}\}_{t=1}^{N_p}$.

Then, the observations $\{x(k), y(k)\}$ are processed iteratively. At each time k, the model parameters A_i are updated (step 7.3), the active mode $\sigma(k)$ is estimated (steps 7.4-7.6), and the regressor $x(k)$ is consequently assigned to cluster $\mathcal{C}_{\sigma(k)}$ (step 7.7) and the cluster's centroid $c_{\sigma(k)}$ is computed (steps 7.8-7.9). Note that, in steps 7.8 and 7.9, the centroids are recursively updated. The expression for the recursive update can be easily derived from the definition of centroids.

We remark that, at the initial iterations of Algorithm 1 (ie, for $\bar{k} \ll N$), the estimate of mode sequence $\{\sigma(t)\}_{t=1}^{\bar{k}}$ may not be accurate since the learning phase is based on a "small" set of observations $\{x(t), y(t)\}_{t=1}^{\bar{k}}$. Such inaccurate estimate of the mode sequence $\{\sigma(t)\}_{t=1}^{\bar{k}}$ (namely, poor classification of the pairs $\{x(t), y(t)\}_{t=1}^{\bar{k}}$) also influences the estimate of the active modes and of the model parameters A_i at the next time samples $k > \bar{k}$, as the regularization cost (6b) depends on the estimated mode sequence $\{\sigma(t)\}_{t=1}^{\bar{k}}$. In order to reduce the effect of initial classification error, Algorithm 1 can be run multiple times. More specifically, at the n_qth run of Algorithm 1, the following cost is considered instead of (6a)-(6b):

$$\sum_{i=1}^{s}\sum_{t=0}^{N_p-1}\left\|\left(y(k-t) - A_i\begin{bmatrix}1\\ x(k-t)\end{bmatrix}\right)\delta_i(k-t)\right\|^2 + \tag{14a}$$

$$\gamma_1\sum_{t=1}^{k-N_p}\left\|y(t) - A_{\sigma(t,n_q)}\begin{bmatrix}1\\ x(t)\end{bmatrix}\right\|^2 + \tag{14b}$$

$$\gamma_2 \sum_{q=1}^{n_q-1} \lambda^{n_q-q-1} \sum_{t=1}^{N-N_p} \left\| y(t) - A_{\sigma(t,q)} \begin{bmatrix} 1 \\ x(t) \end{bmatrix} \right\|^2, \tag{14c}$$

$$\gamma_3 \sum_{i=1}^{s} \sum_{t=0}^{N_p-1} \|(x(k-t) - c_i)\delta_i(k-t)\|^2, \tag{14d}$$

where $\sigma(t,q)$ $(q=1, \dots, n_q)$ denotes the estimate of the active mode at time t obtained at the qth run of Algorithm 1. Note that (14c) is a regularization term based on the past runs of Algorithm 1, while (14b) plays the same role of (6b), which regularizes the parameters A_i based on the estimate $\{\sigma(t,n_q)\}_{t=1}^{k-N_p}$ obtained at the current run n_q of Algorithm 1. A forgetting factor $\lambda \in \mathbb{R} : 0 < \lambda \leq 1$ is also included in (14c) to exponentially downweight the estimates $\{\sigma(t,q)\}_{t=1}^{N}$ obtained at past runs of Algorithm 1.

Using the same ideas presented in Section 3.1.1, the cost (14) can be recursively updated for each new sample $\{x(k), y(k)\}$. This avoids the need to store the whole time history of estimates $\{\sigma(k,q)\}_{k=1,q=1}^{N,n_q-1}$ obtained at the previous runs of Algorithm 1. Thus, the cost (14) can be written as

$$\sum_{i=1}^{s} \sum_{t=0}^{N_p-1} \left\| \left(y(k-t) - A_i \begin{bmatrix} 1 \\ x(k-t) \end{bmatrix} \right) \delta_i(k-t) \right\|^2 + \tag{15a}$$

$$\gamma_1 \left\{ \operatorname{tr}\left(\sum_{i=1}^{s} A_i H_i(k-N_p, n_q) A_i' \right) - 2\operatorname{tr}\left(\sum_{i=1}^{s} A_i F_i(k-N_p, n_q) \right) + \operatorname{tr}\left(\sum_{t=1}^{k-N_p} y(t) y(t)' \right) \right\} + \tag{15b}$$

$$\gamma_2 \left\{ \sum_{q=1}^{n_q-1} \operatorname{tr}\left(\sum_{i=1}^{s} A_i \lambda^{n_q-q-1} H_i(N-N_p, q) A_i' \right) - 2 \sum_{q=1}^{n_q-1} \operatorname{tr}\left(\sum_{i=1}^{s} A_i \lambda^{n_q-q-1} F_i(N-N_p, q) \right) + \sum_{q=1}^{n_q-1} \operatorname{tr}\left(\lambda^{n_q-q-1} \sum_{t=1}^{N-N_p} y(t) y(t)' \right) \right\}, \tag{15c}$$

$$\gamma_3 \sum_{i=1}^{s} \sum_{t=0}^{N_p-1} \|(x(k-t) - c_i)\delta_i(k-t)\|^2, \tag{15d}$$

where $H_i(N-N_p, q)$ and $F_i(N-N_p, q)$ $(q=1, \dots, n_q)$ in (15c) are defined similarly to (9), and are computed based on the estimates $\sigma(t,q)$ calculated at the q-th previous run of Algorithm 1. Specifically,

$$H_i(N-N_p, q) = \sum_{t=1}^{N-N_p} \begin{bmatrix} 1 \\ x(t) \end{bmatrix} \begin{bmatrix} 1 \\ x(t) \end{bmatrix}' \delta_i(t,q), \tag{16a}$$

$$F_i(N-N_p, q) = \sum_{t=1}^{N-N_p} \begin{bmatrix} 1 \\ x(t) \end{bmatrix} y(t)' \delta_i(t,q), \tag{16b}$$

with

$$\delta_i(t,q) = \begin{cases} 1 & \text{if } \sigma(t,q) = i, \\ 0 & \text{otherwise.} \end{cases} \tag{16c}$$

When a new input/output pair $\{x(k), y(k)\}$ at time k is processed, the matrices $H_i(k-N_p, n_q)$ and $F_i(k-N_p, n_q)$ in (15b) can be recursively updated through (11) while only the matrices $H_i(N-N_p, q)$ and $F_i(N-N_p, q)$ (with $q=1, \dots, n_q-1$) are needed to construct the terms (15c).

3.2 | Stage 2: Computing the input partition

Once the model parameters $\{A_i\}_{i=1}^{s}$ and the sequence of active modes $\{\sigma(k)\}_{k=1}^{N}$ are estimated, the partition $\{\mathcal{X}_i\}_{i=1}^{s}$ of the regressor space $\mathcal{X}$ is computed. This is done by separating the computed clusters $\{C_i\}_{i=1}^{s}$ using linear multicategory discrimination.

In the following paragraphs, we briefly describe the linear multicategory discrimination algorithm recently proposed by some of the authors[19] and used to compute the partition of the regressor space.

3.2.1 | Linear multicategory discrimination: problem formulation

According to the formulation introduced by Bennet and Mangasarian,[35] the following piecewise-affine separator function $\phi : \mathbb{R}^{n_x} \to \mathbb{R}$ is used to separate the clusters $C_1, \dots, C_s$:

$$\phi(x) = \max_{i=1,\dots,s} \left(\begin{bmatrix} x' & -1 \end{bmatrix} \begin{bmatrix} \omega^i \\ \gamma^i \end{bmatrix} \right), \tag{17}$$

where $\omega^i \in \mathbb{R}^{n_x}$ and $\gamma^i \in \mathbb{R}$ are unknown parameters to be computed.

For $i = 1, \dots, s$, let m_i denote the cardinality of the cluster C_i and let $M_i \in \mathbb{R}^{m_i \times n_x}$ denote the matrix obtained by stacking the regressors $x(k)'$ belonging to C_i in its rows.

If the clusters $\{C_i\}_{i=1}^s$ are linearly separable, then the separator function ϕ in (17) satisfies the following conditions:

$$\begin{bmatrix} M_i & -\mathbf{1}_{m_i} \end{bmatrix} \begin{bmatrix} \omega^i \\ \gamma^i \end{bmatrix} > \begin{bmatrix} M_i & -\mathbf{1}_{m_i} \end{bmatrix} \begin{bmatrix} \omega^j \\ \gamma^j \end{bmatrix}, \quad i,j = 1, \dots, s,\ i \neq j, \tag{18}$$

or equivalently,

$$\begin{bmatrix} M_i & -\mathbf{1}_{m_i} \end{bmatrix} \begin{bmatrix} \omega^i \\ \gamma^i \end{bmatrix} \geq \begin{bmatrix} M_i & -\mathbf{1}_{m_i} \end{bmatrix} \begin{bmatrix} \omega^j \\ \gamma^j \end{bmatrix} + \epsilon, \quad i,j = 1, \dots, s,\ i \neq j,$$

where $\epsilon \in \mathbb{R}^{m_i}$ with $\epsilon_k > 0,\ k = 1, \dots m_i$ and $\mathbf{1}_{m_i}$ is an m_i-dimensional vector of ones. Multiplying both sides of the above inequality *element-wise* with $1/\epsilon_k$, we can obtain the following equivalent inequality with normalized coefficients ω^i and γ^i,

$$\begin{bmatrix} M_i & -\mathbf{1}_{m_i} \end{bmatrix} \begin{bmatrix} \omega^i \\ \gamma^i \end{bmatrix} \geq \begin{bmatrix} M_i & -\mathbf{1}_{m_i} \end{bmatrix} \begin{bmatrix} \omega^j \\ \gamma^j \end{bmatrix} + \mathbf{1}_{m_i}, \quad i,j = 1, \dots, s,\ i \neq j. \tag{19}$$

From (19), the piecewise-affine separator ϕ thus satisfies the conditions:

$$\begin{cases} \phi(x) = \begin{bmatrix} x' & -1 \end{bmatrix} \begin{bmatrix} \omega^i \\ \gamma^i \end{bmatrix}, \ \forall x \in C_i,\ i = 1, \dots, s, \\ \phi(x) \geq \begin{bmatrix} x' & -1 \end{bmatrix} \begin{bmatrix} \omega^j \\ \gamma^j \end{bmatrix} + 1, \ \forall x \in C_i,\ i \neq j. \end{cases} \tag{20}$$

Thus, based on (20), each polyhedron $\{\mathcal{X}_i\}_{i=1}^s$ is defined as

$$\mathcal{X}_i = \left\{ x \in \mathbb{R}^{n_x} : \begin{bmatrix} x' & -1 \end{bmatrix} \begin{bmatrix} \omega^i - \omega^j \\ \gamma^i - \gamma^j \end{bmatrix} \geq 1,\ j = 1, \dots, s,\ j \neq i \right\}.$$

In order to compute the parameters $\{\omega^i, \gamma^i\}_{i=1}^s$, the following convex optimization problem is solved

$$\min_{\xi} \frac{\kappa}{2} \sum_{i=1}^{s} (\|\omega^i\|_2^2 + (\gamma^i)^2) + \sum_{i=1}^{s} \sum_{\substack{j=1 \\ j \neq i}}^{s} \frac{1}{m_i} \left\| \left(\begin{bmatrix} M_i & -\mathbf{1}_{m_i} \end{bmatrix} \begin{bmatrix} \omega^j - \omega^i \\ \gamma^j - \gamma^i \end{bmatrix} + \mathbf{1}_{m_i} \right)_+ \right\|_2^2, \tag{21}$$

where ξ is the set of optimization variables, that is, $\xi = \begin{bmatrix}(\omega^1)' & \cdots & (\omega^s)' & \gamma^1 & \cdots & \gamma^s\end{bmatrix}'$, and $(\cdot)_+$ denotes a nonnegative vector, that is, $(x)_+$ denotes a vector whose ith element is $x_i = \max\{x_i, 0\}$. Problem (21) minimizes the averaged squared 2-norm of the violation of the inequalities in (19). The regularization parameter $\kappa > 0$ makes sure that the optimization problem (21) is strongly convex and has a unique global minimizer.

3.2.2 | Recursive multicategory discrimination

As an alternative to the batch approach for multicategory discrimination discussed in the previous paragraph, or in addition to it, for refining the partition function ϕ on-line based on streaming data, a recursive approach using *on-line* convex programming can be used to solve (21).

Let us treat the data-points $x \in \mathbb{R}^{n_x}$ as random vectors and let us assume that there exists an oracle function $i^\star : \mathbb{R}^{n_x} :\to \{1, \dots, s\}$ that assigns the corresponding mode $i^\star(x)$ to a given input $x \in \mathbb{R}^{n_x}$. By definition, the function $i^\star$ describes the clusters in the data-point space $\mathbb{R}^{n_x}$. Let us also assume that the following probabilities

$$\pi_i = \text{Prob}[i^\star(x) = i] = \int_{\mathbb{R}^{n_x}} \delta(i, i^\star(x)) p(x) dx,$$

are known for all $i = 1, \dots, s$, where $\delta(i,j) = 1$ if $i = j$, zero otherwise.

By denoting with $E_x[\cdot]$ the expected value w.r.t. the random variable x, problem (21) can be then generalized as the following convex regularized stochastic optimization problem

$$\xi^* = \min_{\xi} E_{x \in \mathbb{R}^{n_x}}[\ell(x, \xi)] + \frac{\kappa}{2}\|\xi\|_2^2, \tag{22}$$

with

$$\ell(x, \xi) = \sum_{\substack{j=1 \\ j \neq i^\star(x)}}^{s} \frac{1}{\pi_{i^\star(x)}} (x'(\omega^j - \omega^{i^\star(x)}) - \gamma^j + \gamma^{i^\star(x)} + 1)_+^2.$$

The objective function in (22) penalizes the expected violation of the condition in (19) when $i = i^\star(x)$. The weights π_i can be estimated offline from a data subset, specifically $\pi_i = \frac{m_i}{N}$, and can be then updated iteratively. Nevertheless, numerical experiments have shown that uniform weights $\pi_i = \frac{1}{s}$ work equally well. Problem (22) can be solved using online convex optimization algorithms, such as the adapted version of the *averaged stochastic gradient descent* (ASGD) method.[19]

4 | CASE STUDIES

The performance of the presented PWA regression algorithm is demonstrated through three case studies, two of which use experimental data. All computations are carried out on an i5 1.8-GHz Intel core processor with 8 GB of RAM running MATLAB R2018a.

4.1 | Numerical example

In this example, we use synthetic data generated by the following *single-input single-output* PWARX dynamical system:

$$y(k) = \begin{cases} A_1 \begin{bmatrix} 1 \\ x(k) \end{bmatrix} + e_o(k) & \text{if } -0.3x_1(k) + 0.6x_3(k) + 0.3 > 0, \\ A_2 \begin{bmatrix} 1 \\ x(k) \end{bmatrix} + e_o(k) & \text{if } -0.3x_1(k) + 0.6x_3(k) + 0.3 < 0, \end{cases} \tag{23}$$

SNR	21 dB	13 dB
BFR	$87.59 \pm 4.82\%$	$82.24 \pm 4.97\%$
MF	$94.54 \pm 1.45\%$	$85.28 \pm 10.17\%$

TABLE 1 Example 1: mean $\pm$ standard deviation of BFR and MF index over 100 Monte-Carlo runs for SNR equal to 21 and 13 dB

	True value	mean $\pm$ std (SNR=21 dB)	mean $\pm$ std (SNR=13 dB)
	0.20	0.2137 ± 0.0116	0.2709 ± 0.0916
	0.50	0.4972 ± 0.0194	0.4997 ± 0.0568
A_1	-0.10	-0.0946 ± 0.0149	-0.0681 ± 0.0480
	1.00	0.9891 ± 0.0096	0.9431 ± 0.0949
	0.20	0.2041 ± 0.0147	0.2159 ± 0.0412
	-0.30	-0.2744 ± 0.0194	-0.1071 ± 0.1745
	0.80	0.7867 ± 0.0168	0.7331 ± 0.0690
A_2	0.10	0.1065 ± 0.0118	0.1108 ± 0.0521
	0.40	0.4125 ± 0.0106	0.4858 ± 0.1153
	0.05	0.0564 ± 0.0124	0.0779 ± 0.0428

TABLE 2 Example 1: mean and standard deviation (std) of the estimated PWARX model parameters over 100 Monte-Carlo runs for SNR equal to 21 and 13 dB

where $A_1 = [0.2\ 0.5\ -0.1\ 1\ 0.2]$, $A_2 = [-0.3\ 0.8\ 0.1\ 0.4\ 0.05]$. The 4-dimensional regressor vector $x(k)$ is defined as $x(k) = [y(k-1)\ y(k-2)\ u(k-1)\ u(k-2)]' \in \mathbb{R}^4$, where $u(k)$ is the external input of the dynamical system at time k, and it is randomly generated from a uniform distribution taking values between -2 and 2. Two experiments are carried out with varying noise conditions where the noise e_o corrupting the output signal is generated by a zero-mean white Gaussian process with variance 0.01 for the first experiment and 0.09 for the second experiment.

In order to evaluate the statistical properties of the proposed identification approach, a Monte-Carlo study with 100 runs is performed (Table 1, 2). At each Monte-Carlo run, a different realization of input $u(k)$ and noise $e_o(k)$ process is generated. A training dataset of length $N = 1000$ and a noise-free validation dataset of length $N_{\text{val}} = 300$ are constructed. The average values of the *signal-to-noise ratio* (SNR)

$$\text{SNR} = 10 \log \frac{\sum_{k=1}^{N} (y(k) - e_o(k))^2}{\sum_{k=1}^{N} e_o^2(k)},$$

over the 100 runs are 21 and 13 dB for the two set of Monte Carlo experiments. For the identification of PWARX model (23), Algorithm 1 is run for $n_q = 3$ iterations with $s = 2$ modes, model orders $n_a = n_b = 2$, horizon length $N_p = 8$, and weights, $\gamma_1 = 0.1$, $\gamma_2 = 0.01$, $\gamma_3 = 0$, $\lambda = 0.1$ in (14). The formulated MIQP problem (with objective function (14)) contains 26 real and 16 binary variables, 64 inequality and 8 equality constraints, and solved using both the miqpGPAD algorithm and the commercial solver GUROBI. The average CPU time to solve the resultant MIQP for processing a single training sample is 90 ms, while GUROBI takes 20 ms. It is worth remarking that miqpGPAD is a simple library-free solver, providing comparable performance with respect to GUROBI for the given problem. These features make the miqpGPAD solver particularly suitable for embedded online applications, where a model needs to be updated once new data becomes available.

In the second stage, the linear multicategory discrimination problem (21) is solved in order to compute polyhedral partitions of the regressor space. A small value of the regularization hyper-parameter κ is set ($\kappa = 10^{-5}$) just to guarantee strong convexity of problem (21). On average, 25 ms are required to compute the solution of problem (21) through the regularized piecewise-smooth Newton method[36] with Armijo's line search.

The mean and standard deviation of the estimated model parameters A_1 and A_2 over the 100 Monte-Carlo runs are reported in Table 2, which shows that the true parameters A_1 and A_2 are correctly identified.

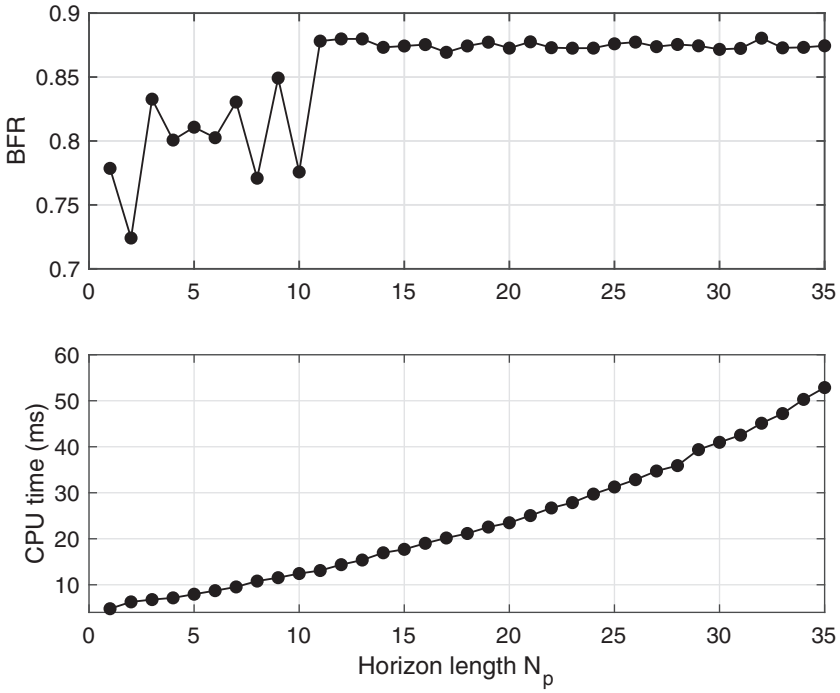

FIGURE 1 Example 1: Top panel: Best fit rate; Bottom panel: Average CPU time taken by GUROBI to solve a single MIQP problem

The quality of the estimated PWARX model is assessed on validation data and quantified by the *best fit rate* (BFR) index defined as

$$\text{BFR} = \max\left\{1 - \sqrt{\frac{\sum_{k=1}^{N_{\text{val}}}(y(k)-\hat{y}(k))^2}{\sum_{k=1}^{N_{\text{val}}}(y(k)-\bar{y})^2}}, 0\right\} \times 100\%, \tag{24}$$

where $\hat{y}(k)$ is the estimated model output and $\bar{y}$ is the sample mean of the output signal in the validation dataset. The accuracy in reconstructing the active mode sequence is expressed by the *mode-fit* (MF) index

$$\text{MF} = \left(\frac{1}{N_{\text{val}}}\sum_{k=1}^{N_{\text{val}}}\mathbf{I}(\hat{\sigma}(k) = i^{\star}(k))\right) \times 100\%, \tag{25}$$

where $\mathbf{I}(\cdot)$ is the indicator function, and $\hat{\sigma}(k)$ and $i^{\star}(k)$ are the estimated and true mode at time k, respectively.

The mean of the BFR and MF index achieved over the 100 Monte-Carlo runs are reported in Table 1.

Furthermore, in order to asses the effect of the horizon length N_p on accuracy and computational complexity of the approach, Algorithm 1 is run with different values of N_p. The dataset associated to a SNR equal to 21 dB is used. It is observed in Figure 1 that increasing the value of N_p increases the BFR, at the cost of a larger computational time. Nevertheless, note that for values of N_p larger than 15, no significant increase in the BFR is observed.

Finally, the effectiveness of the iterative refinement described in Section 3.1.3 is assessed by running Algorithm 1 for $n_q = 1, 2, 3$, and 4 iterations, with $N_p = 8$, weights $\gamma_1 = 0.1$, $\gamma_2 = 0.01$, $\gamma_3 = 0$, and forgetting factor $\lambda = 0.1$ in (14). The achieved BFR, MF, and the norm of error between true and estimated model parameters $||A - \hat{A}||$ are reported in Table 3, which shows how iterative refinement improves the accuracy of the estimated model. This also shows that, using iterative refinement, higher accuracy can be achieved with lower values of horizon length N_p. Note that, after three iterations, no improvement is observed.

4.2 | Case study: Identification of a pick-and-place machine

As a second case study, we consider a data-driven identification problem of the electronic component placement process in a pick-and-place machine. The experimental setup[28] consists of an electronic component placed on a mounting head,

TABLE 3 Example 1: Iterative refinement

runs(n_q)	BFR	MF	$\|\|A_1 - \hat{A}_1\|\|$	$\|\|A_2 - \hat{A}_2\|\|$
1	77.09 %	89.33 %	0.0954	0.4553
2	83.64 %	91.67 %	0.0550	0.2187
3	87.03 %	92.67 %	0.0443	0.0876
4	87.02 %	92.67 %	0.0443	0.0882

Note: BFR, MF, and $||A - \hat{A}||$ reported by running Algorithm 1 with $N_p = 8$.

TABLE 4 Identification of pick-and-place machine: Best fit rates (BFR) for the identified PWARX models with different number of discrete operating modes s

	$s=1$	$s=2$	$s=3$
BFR	76.98%	88.25%	89.33%

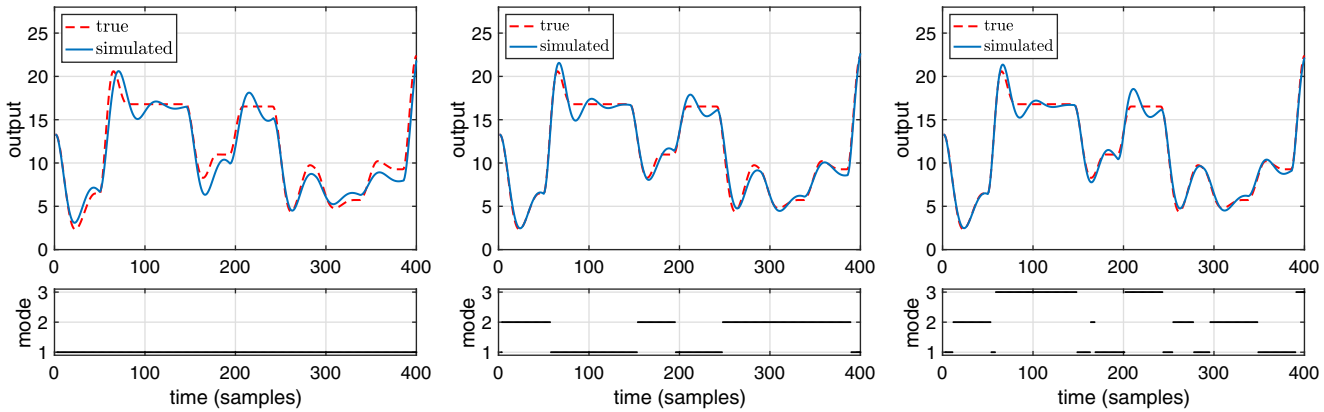

FIGURE 2 Validation results for pick-and-place machine using PWARX models with $s = 1$ (left), $s = 2$ (center), and $s = 3$ (right) operating modes. Top: simulated versus true output. Bottom: estimated mode sequence [Colour figure can be viewed at wileyonlinelibrary.com]

which is pushed down to an impact surface, and then released. The system consists of two operating modes: *free* mode and *impact* mode. In the free mode, the movement of the electronic component is unconstrained, while in the impact mode, the mounting head is in contact with the impacting surface. An input-output dataset over a time interval of 15 seconds with sampling frequency of 150 Hz is gathered.

Due to the switching between free and impact modes, a PWARX model is a good candidate to describe the behavior of the process. Specifically, we consider the PWARX model structure in (1). The regressor is defined as in (3), for dynamical orders $n_a = 2$ and $n_b = 2$. Three different number of submodels s are considered (namely, $s = 1$, $s = 2$, and $s = 3$). Note that, for $s = 1$, the PWARX model is actually a simple linear ARX model.

The model parameters are identified from a training dataset of $N = 1000$ samples, by using Algorithm 3 with horizon length $N_p = 15$ and weights $\gamma_1 = 0.1$, $\gamma_2 = 0$, $\gamma_3 = 1$ in (14). The resultant MIQP is solved through the GUROBI solver.[37] The average execution time to solve a single MIQP problem is 46.1 ms. In the second stage, the linear multicategory discrimination problem (21) is solved to compute a polyhedral partition of the regressor space. The parameter κ is set to 10^{-5}. The execution time to solve this problem is 63 ms.

The performance of the estimated models is assessed on a validation dataset of length $N_{\text{val}} = 400$, and quantified in terms of BFR (24). The BFRs achieved by the three different identified models are reported in Table 4. The estimated model output and the true output trajectories are shown in Figure 2, along with the estimated sequence of discrete operating modes. It can be observed from Table 4 and Figure 2 that both the PWARX models with $s = 2$ and $s = 3$ operating modes outperform the performance of the linear ARX model.

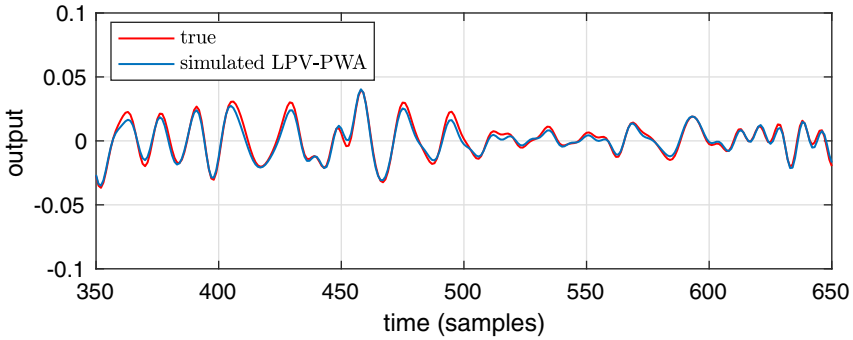

FIGURE 3 Electronic bandpass filter: true versus simulated output [Colour figure can be viewed at wileyonlinelibrary.com]

4.3 | Case study: Identification of an electronic bandpass filter

As a third case study, we consider the identification of an LPV model describing the behavior of an electronic bandpass filter.[29] The benchmark dataset used in this case study has been downloaded from the https://www.kth.se/social/group/system-identificatio/page/17th-ifac-symposium-on-system-identifica/.[38]

The experimental setup consists of an electronic second-order bandpass filter, which is implemented using an *n-type JFET* transistor in parallel with a variable resistor. The resonant frequency of the filter varies according to the gate-source voltage of the transistor, which is chosen as a scheduling signal $p(k)$ for the LPV model to be identified.

The input signal $u(k)$ applied to the system is the sum of cosines, that is, $u(k) = \sum_{\tau=1}^{N_{\max}} \bar{A}_\tau \cos(\omega_\tau k + \psi_\tau)$, where $\bar{A}_\tau$ is the amplitude of the τth cosine, ψ_τ is the phase angle randomly generated from a uniform distribution in the interval $(0, 2\pi)$, and $N_{\max}$ is selected such that $\omega_{N_{\max}} \approx 2\pi 20$ kHz. The scheduling variable $p(k)$ is a harmonic signal consisting of sum of sines with random amplitudes and 3 frequency bins, that is, $p(k) = V_{\mathrm{DC}} + \sum_{\tau=1}^{3} B_\tau \cos(\omega_\tau k + \phi_\tau)$, where $V_{\mathrm{DC}} = -0.7V$ is the offset voltage, amplitudes B_τ are zero-mean, randomly generated from a normal distribution. The reader is referred to Lataire et al[29] for a detailed description of the experimental setup and of the available datasets.

For the identification of the electronic bandpass filter, we consider the LPV model (4)-(5) with model orders $n_a = 2$, $n_b = 2$ and with number of PWA submodels $s = 6$. The model is identified via the PWA regression Algorithm 1 with horizon length $N_p = 10$, $\gamma_1 = 10^{-4}$, $\gamma_2 = 0$, $\gamma_3 = 0.1$ in (14) and using a training dataset with $N = 30\,000$ samples. The values of n_a, n_b, N_p, γ_1, γ_2, γ_3 are tuned via cross-validation. The average time to process one training sample solve (namely, to solve the resulting MIQP (6) through the commercial solver GUROBI) is 51 ms. In the second stage, the linear multicategory discrimination problem (21) is solved to compute a polyhedral partition of the scheduling space. The hyper-parameter κ is set equal to 10^{-5}. The execution time to solve this problem is 57 ms.

The performance is evaluated on an independent validation dataset of size $N_{\mathrm{val}} = 10\,000$ samples. This validation set was used neither for training nor for tuning all the hyper-parameters of the presented approach. The achieved BFR is 81.5%. For the sake of better visual representation, only few samples of the simulated output *vs* true output are depicted in Figure 3. The obtained results show that the estimated PWA-LPV model with PWA approximations of LPV coefficients is able to accurately reproduce the behavior of the electronic bandpass filter better.

5 | CONCLUSIONS

In this article, a novel two-stage algorithm for PWA regression is presented and properly adapted for the identification of PWARX and LPV models. The underlying idea of the approach is to iteratively process training data according to a regularized moving-horizon strategy and then solve, at each iteration, a small-scale MIQP problem. The proposed method combines the advantages of the mixed-integer programming approach for PWA regression[14] (namely, simultaneous estimation of submodel parameters and operating modes) and recursive algorithm[19] (namely, iterative processing of training samples and computational efficiency).

Future research activities are devoted to the formulation of the problem in a Bayesian framework, in order to compute (or approximate) the posterior probability distribution of the model parameters, with the final aim of providing confidence uncertainty intervals on the estimated parameters and on the predicted output.

ORCID

Manas Mejari https://orcid.org/0000-0002-0641-1156
Vihangkumar V. Naik https://orcid.org/0000-0002-2214-8052
Dario Piga https://orcid.org/0000-0001-7691-4886
Alberto Bemporad https://orcid.org/0000-0001-6761-0856

REFERENCES

1. Breiman L. Hinging hyperplanes for regression, classification, and function approximation. *IEEE Trans Inf Theory*. 1993;39(3):999-1013.
2. Heemels WPMH, De Schutter B, Bemporad A. Equivalence of hybrid dynamical models. *Automatica*. 2001;37(7):1085-1091.
3. Bemporad A, Ferrari-Trecate G, Morari M. Observability and controllability of piecewise affine and hybrid systems. *IEEE Trans Autom Control*. 2000;45(10):1864-1876.
4. Herceg M, Kvasnica M, Jones CN, Morari M. Multi-parametric toolbox 3.0. Paper presented at: Proceedings of the European Control Conference; 2013;502-510; IEEE: Zurich, Switzerland.
5. Bemporad A, Morari M. Control of systems integrating logic, dynamics, and constraints. *Automatica*. 1999;35(3):407-427.
6. Lauer F. On the complexity of piecewise affine system identification. *Automatica*. 2015;62:148-153.
7. Garulli A, Paoletti S, Vicino A. A survey on switched and piecewise affine system identification. Paper presented at: Proceedings of the 16th IFAC Symposium on System Identification; 2012:344-355; Brussels, Belgium.
8. Paoletti S, Juloski AL, Ferrari-Trecate G, Vidal R. Identification of hybrid systems a tutorial. *Eur J Control*. 2007;13(2):242-260.
9. Bemporad A, Garulli A, Paoletti S, Vicino A. A bounded-error approach to piecewise affine system identification. *IEEE Trans Autom Control*. 2005;50(10):1567-1580.
10. Ozay N, Lagoa C, Sznaier M. Set membership identification of switched linear systems with known number of subsystems. *Automatica*. 2015;51:180-191.
11. Bako L. Identification of switched linear systems via sparse optimization. *Automatica*. 2011;47(4):668-677.
12. Ohlsson H, Ljung L. Identification of switched linear regression models using sum-of-norms regularization. *Automatica*. 2013;49(4):1045-1050.
13. Piga D, Tóth R. An SDP approach for ℓ_0-minimization: application to ARX model segmentation. *Automatica*. 2013;49(12):3646–3653.
14. Roll J, Bemporad A, Ljung L. Identification of piecewise affine systems via mixed-integer programming. *Automatica*. 2004;40(1):37-50.
15. Nakada H, Takaba K, Katayama T. Identification of piecewise affine systems based on statistical clustering technique. *Automatica*. 2005;41(5):905-913.
16. Juloski AL, Weiland S, Heemels WPMH. A Bayesian approach to identification of hybrid systems. *IEEE Trans Autom Control*. 2005;50(10):1520-1533.
17. Ferrari-Trecate G, Muselli M, Liberati D, Morari M. A clustering technique for the identification of piecewise affine systems. *Automatica*. 2003;39(2):205-217.
18. Bako L, Boukharouba K, Duviella E, Lecoeuche S. A recursive identification algorithm for switched linear/affine models. *Nonlinear Anal Hybrid Syst*. 2011;5(2):242-253.
19. Breschi V, Piga D, Bemporad A. Piecewise affine regression via recursive multiple least squares and multicategory discrimination. *Automatica*. 2016;73:155-162.
20. Breschi V, Bemporad A, Piga D. Identification of hybrid and linear parameter varying models via recursive piecewise affine regression and discrimination. Paper presented at: Proceedings of the 15th European Control Conference; 2016; IEEE: Aalborg, Denmark.
21. Tóth R. *Modeling and Identification of Linear Parameter-Varying Systems Lecture Notes in Control and Information Sciences*. Vol 403. Heidelberg, Germany: Springer; 2010.
22. Bamieh BA, Giarré L. Identification of linear parameter-varying models. *Int J Robust Nonlinear Control*. 2002;12(9):841-853.
23. Wingerden JW, Verhaegen M. Subspace identification of bilinear and LPV systems for open-and closed-loop data. *Automatica*. 2009;45(2):372-381.
24. Piga D, Cox P, Tóth R, Laurain V. LPV system identification under noise corrupted scheduling and output signal observations. *Automatica*. 2015;53:329-338.
25. Tóth R, Laurain V, Zheng W X, Poolla K. Model structure learning: a support vector machine approach for LPV linear-regression models. Paper presented at: Proceedings of the 50th IEEE Conference on Decision and Control; 2011:3192-3197; Orlando, FL.
26. Hsu K, Vincent T L, Poolla K. Nonparametric methods for the identification of linear parameter varying systems. Paper presented at: Proceedings of the International Symposium on Computer-Aided Control System Design; 2008:846-851; San Antonio, Texas.
27. Piga D, Tóth R. LPV model order selection in an LS-SVM setting. Paper presented at: Proceedings of the 52nd IEEE Conference on Decision and Control; 2013:4128-4133; Florence, Italy.
28. Juloski AL, Heemels WPMH, Ferrari-Trecate G. Identification of hybrid systems a tutorial. *Control Eng Pract*. 2004;12(10):1241-1252.
29. Lataire J, Louarroudi E, Pintelon R, Rolain Y. Benchmark data on a linear time- and parameter-varying system. Paper presented at: Proceedings of the 17th IFAC Symposium on System Identification; 2015:1477-1482; Beijing, China.
30. Naik V. V., Mejari M, Piga D, Bemporad A. Regularized moving-horizon piecewise affine regression using mixed-integer quadratic programming. Paper presented at: Proceedings of the 25th Mediterranean Conference on Control and Automation; 2017:1349-1354; Valletta, Malta.

31. Patrinos P, Bemporad A. An accelerated dual gradient-projection algorithm for embedded linear model predictive control. *IEEE Trans Automat Control*. 2014;59(1):18-33.
32. Naik V. V., Bemporad A. Embedded mixed-integer quadratic optimization using accelerated dual gradient projection. Paper presented at: Proceedings of the 20th IFAC World Congress; 2017; Toulouse, France.
33. Mejari M, Naik V. V., Piga D, Bemporad A. Identification of Hybrid and LPV Models via Piecewise Affine Regression using Mixed Integer Programming. Technical report TR-IDSIA-2019-04; 2019. www.dariopiga.com/TR/TR_IDSIA_2019_04.pdf.
34. Bemporad A, Naik V. V. A numerically robust mixed-integer quadratic programming solver for embedded hybrid model predictive control. Paper presented at: Proceedings of the 6th IFAC Conference on Nonlinear Model Predictive Control; 2018:412-417; Madison, Wisconsin.
35. Bennett KP, Mangasarian OL. Multicategory discrimination via linear programming. *Optim Methods Softw*. 1994;3:27-39.
36. Bemporad A, Bernardini D, Patrinos P. A *Convex Feasibility Approach to Anytime Model Predictive Control*. IMT Institute for Advanced Studies, Lucca; 2015. http://arxiv.org/abs/1502.07974.
37. Gurobi Optimization, Inc Gurobi optimizer reference manual; 2014. https://www.gurobi.com/
38. Lataire J, Louarroudi E, Pintelon R, Rolain Y. Benchmark Data on a Linear Time- and Parameter-varying system. *IFAC-PapersOnLine*. 2015;48(28):1477-1482. http://dx.doi.org/10.1016/j.ifacol.2015.12.342.

How to cite this article: Mejari M, Naik VV, Piga D, Bemporad A. Identification of hybrid and linear parameter-varying models via piecewise affine regression using mixed integer programming. *Int J Robust Nonlinear Control*. 2020;30:5802–5819. https://doi.org/10.1002/rnc.5198