Prediction error methods in learning jump ARMAX models

Valentina Breschi, Alberto Bemporad, Dario Piga, Stephen Boyd

Abstract— **Jump models describe systems that change their dynamics over time. Identifying jump models amounts both to learn the behavior of the system at each operating mode and to reconstruct the active mode sequence from data. This paper focuses on the identification of jump autoregressive moving-average models with exogenous inputs (JARMAX), combining prediction error methods with a coordinate descent algorithm for fitting jump models. The resulting identification algorithm alternates between minimizing the sum of prediction errors with respect to the parameters of the ARMAX models, and minimizing a discrete loss function with respect to the sequence of active modes.**

I. Introduction

Many real-world systems are characterized by multiple operating regimes. Examples include switching electronic circuits, thermostats, vehicles, electrical household appliances, etc. The dynamics of these systems can be described through jump models, which are finite collections of (usually simple) dynamical submodels with continuous states, each one characterizing the local behavior of the system. The time evolution of the operating mode (usually referred to as *discrete state*) might be governed by *deterministic* rules, as in hybrid automata [14], [23], mixed logical dynamical models [5] and piecewise-affine (PWA) models [7], or by *stochastic* jumps, as in hybrid stochastic automata [4] and Markov jump models [11].

Although jump models are widely used in different applications, such as motion reconstruction [17], speech recognition [20] and tracking maneuvering targets [22], how to learn jump models from data is still an active topic of research. The main difficulty in learning jump models is that the discrete state is supposed unknown, that is it must be reconstructed from input/output observations. Several algorithms were proposed in the literature to learn specific classes of jump models and, in particular, to identify switched linear dynamical models. *PieceWise affine AutoRegressive with eXogenous inputs* (PWARX) models are considered in [9], [12], [21], with PWARX defined by partitioning the state+input domain into polyhedral regions and by associating a linear/affine ARX model with each region. In [9], [12], [21] both the partition of the input+state space and the parameters of the local ARX models are estimated from data. Algorithms for the identification of *Switched AutoRegressive with eXogenous inputs* (SARX) models are presented in [1], [18] - [19]. In SARX models, in which the switching between ARX models does not depend on the input/state of the system, the problem is to estimate the parameters of the ARX submodels and switching instant from data. In *Markov Jump Linear Models* (MJLMs), the mechanism driving the switches of the discrete state is governed by a Markov chain, while the dynamics of continuous states are described by linear models. If the submodels are known, the problem of learning MJLM reduces to the estimation of a *Hidden Markov Model* (HMM), that can be solved through the Baum-Welch method [20]. Algorithms for learning MJMLs with unknown dynamics are proposed in [10], [13], while the problem of discrete hybrid automata identification, where mode switches are driven by a finite state machine, is addressed in [8].

All the above approaches are based on the assumption that output data are generated by local ARX subsystems, so that standard algorithms for linear regression (e.g., least squares) can be employed to retrieve a consistent estimate of the parameters of the local linear models, provided that the discrete state sequence is correctly reconstructed. In this work we relax this assumption and focus on the identification of *Jump AutoRegressive Moving Average with eXogenous inputs* (JARMAX). Due to the ARMAX structure of each submodel, we combine the *Prediction Error Method* (PEM) [15] and the general framework for fitting jump models in [3]. The resulting learning algorithm alternates between two steps: (i) estimation of the parameters of the ARMAX sub-models, and (ii) inference of the active mode sequence. Parameter estimation is carried out using an instance of PEM tailored to ARMAX models with switching coefficients, while mode inference is performed using discrete *Dynamic Programming* (DP) [2]. The approach presented in this paper is thus a characterization to JARMAX models of the general methodology that we introduced in [3]. Contrarily to [16], we do not fix the structure of the time-varying parameters of the ARMAX model a priori, but rather embed the time-varying nature of the ARMAX structure in the hidden mode. Furthermore, in presenting the method, we do not make assumptions on the mechanism driving the mode-transition dynamics, as different models for the evolution of the discrete state can be considered by simply changing the loss function.

The paper is organized as follows. The identification problem is formulated in Section II. The solution algorithm is described in Section III, and strategies to infer the output

This work was partially supported by the H2020-723248 project *DAEDALUS - Distributed control and simulation platform to support an ecosystem of digital automation developers*.

This work was partially supported by the Lombardia region and the Cariplo foundation, under the project *Learning to Control (L2C)*, no. 2017-1520.

V. Breschi is with the Dipartimento di Elettronica e Informazione, Politecnico di Milano, Milano, Italy. valentina.breschi@polimi.it

A. Bemporad is with IMT School for Advanced Studies Lucca, Lucca, Italy alberto.bemporad@imtlucca.it

D. Piga is with IDSIA Dalle Molle Institute for Artificial Intelligence SUPSI-USI, Manno, Switzerland. dario.piga@supsi.ch

S. Boyd is with Department of Electrical Engineering, Stanford University, Stanford, CA. boyd@stanford.edu

and the mode sequence based on the estimated JARMAX model are presented in Section IV. Simulation results are reported in Section V. Conclusions, possible extensions and directions for future works are given in Section VI.

II. Setting and goal

Consider a sequence of T data pairs $\{u_t, y_t\}_{t=1}^T$ generated by a single-input single-output dynamical system, where $t \in \mathbb{N}$ is the *time* index, $u_t \in \mathcal{U} \subseteq \mathbb{R}$ is the input and $y_t \in \mathcal{Y} \subseteq \mathbb{R}$ is the output at time t. Our modeling assumption is that output data is generated by the *Jump Autoregressive Moving-Average* system *with eXogenous input* (JARMAX):

$$A(q, \theta_{s_t})y_t = B(q, \theta_{s_t})u_t + C(q, \theta_{s_t})e_t, \tag{1a}$$

where $\{e_t\}_{t=1}^T$ is a white noise sequence. The variable $s_t \in \mathcal{K} = \{1, \dots, K\}$ denotes the hidden mode (discrete state) at time t, and $A(q, \theta_i)$, $B(q, \theta_i)$, $C(q, \theta_i)$, with $i \in \mathcal{K}$, are polynomials in the time shift operator q ($q^d u_t = u_{t+d}$) given by

$$A(q, \theta_i) = 1 + a_1^i q^{-1} + \ldots + a_{n_a}^i q^{-n_a} \tag{1b}$$

$$B(q, \theta_i) = b_1^i q^{-1} + \ldots + b_{n_b}^i q^{-n_b} \tag{1c}$$

$$C(q, \theta_i) = 1 + c_1^i q^{-1} + \ldots + c_{n_c}^i q^{-n_c}, \tag{1d}$$

where n_a, n_b and n_c indicate the order of the ARMAX submodels. We denote by θ_i, $i \in \mathcal{K}$, the collection of parameters characterizing the i-th submodel, i.e.,

$$\theta_i = \left[\, a_1^i \,\dots\, a_{n_a}^i \; b_1^i \,\dots\, b_{n_b}^i \; c_1^i \,\dots\, c_{n_c}^i \,\right]'. \tag{2}$$

In this paper, we aim at identifying from data both the vector $\Theta = [\theta_1' \;\dots\; \theta_K']' \in \mathbb{R}^{Kn_\Theta}$ of parameters of the ARMAX submodels and the mode sequence $\{s_t\}_{t=1}^T$. This problem clearly reduces to the estimation of a jump ARX model if $C(q, \theta_i) = 1$ for all $i = 1, \dots, K$, or to the identification of a single ARMAX model if $K = 1$ (i.e., the coefficients of the ARMAX structure do not switch over time). In this case, the identification problem can be solved via prediction error methods (PEM) [15].

III. Learning JARMAX models

To identify the JARMAX model (1) we embed PEM into the coordinate descent algorithm for jump model fitting recently proposed by the authors in [3], as detailed next.

A. Loss functions

Let $U = (u_1, \dots, u_T)$ and $Y = (y_1, \dots, y_T)$ be the collections of input and output, respectively. The learning algorithm in [3] relies on the minimization of the *fitting loss*

$$J(U, Y, \Theta, S) = \ell(U, Y, \Theta, S) + r(\Theta) + \mathcal{L}(S) \tag{3}$$

with respect to the parameters Θ and the mode sequence $S = (s_0, s_1, \dots, s_T)$, where s_0 is the (unknown) *initial mode*.

The *mode sequence loss* $\mathcal{L} : \mathcal{K}^{T+1} \to \mathbb{R} \cup \{+\infty\}$ accounts for the temporal order of the data and it is defined as

$$\mathcal{L}(S) = \mathcal{L}^{\text{init}}(s_0) + \sum_{t=1}^T \mathcal{L}^{\text{mode}}(s_t) + \mathcal{L}^{\text{trans}}(s_t, s_{t-1}). \tag{4}$$

Specifically, the *initial mode loss* $\mathcal{L}^{\text{init}} : \mathcal{K} \to \mathbb{R} \cup \{+\infty\}$ takes into account prior knowledge on the initial mode, the *mode loss* $\mathcal{L}^{\text{mode}} : \mathcal{K} \to \mathbb{R} \cup \{+\infty\}$ is used to penalize some modes rather than others (such as to break symmetries), and the *mode transition loss* $\mathcal{L}^{\text{trans}} : \mathcal{K}^2 \to \mathbb{R} \cup \{+\infty\}$ penalizes mode transitions. For a detailed discussion on possible choices for $\mathcal{L}(S)$ we refer the reader to [3].

The *loss function* $\ell : \mathcal{U} \times \mathcal{Y} \times \mathbb{R}^{Kn_\Theta} \times \mathcal{K}^{T+1} \to \mathbb{R} \cup \{+\infty\}$, specified in the next section, accounts for the mismatch between predicted and recorded outputs. The function $r : \mathbb{R}^{Kn_\Theta} \to \mathbb{R} \cup \{+\infty\}$ is a regularization term on Θ. In the following we use $r(\Theta) = \delta \|\Theta\|_2^2$, with $\delta > 0$ to prevent overfitting.

B. Prediction error method

According to the PEM formulation [15], the prediction error at time t for system (1) is given by

$$\begin{aligned}\varepsilon_t(\theta_{s_t}) = e_t = y_t - \hat{y}_t(\theta_{s_t}|t-1) = \\ = \frac{A(q, \theta_{s_t})}{C(q, \theta_{s_t})} y_t - \frac{B(q, \theta_{s_t})}{C(q, \theta_{s_t})} u_t,\end{aligned} \tag{5}$$

where $\hat{y}_t(\theta_{s_t}|t-1)$ is the predictor of the output at time t, given input and output data up to time $t-1$. By manipulating the definition for the prediction error in (5), the predictor is thus given by

$$\hat{y}_t(\theta_{s_t}|t-1) = \left(1 - \frac{A(q, \theta_{s_t})}{C(q, \theta_{s_t})}\right) y_t + \frac{B(q, \theta_{s_t})}{C(q, \theta_{s_t})} u_t. \tag{6}$$

Both in equation (5) and equation (6), the time-domain operators $\frac{A(q,\theta_{s_t})}{C(q,\theta_{s_t})}$ and $\frac{A(q,\theta_{s_t})}{C(q,\theta_{s_t})}$ have the following meaning:

$$\begin{aligned}\varepsilon_t(\theta_{s_t}) = \frac{A(q, \theta_{s_t})}{C(q, \theta_{s_t})} y_t - \frac{B(q, \theta_{s_t})}{C(q, \theta_{s_t})} u_t \\ \Rightarrow C(q, \theta_{s_t})\varepsilon_t(\theta_{s_t}) = A(q, \theta_{s_t})y_t - B(q, \theta_{s_t})u_t.\end{aligned} \tag{7}$$

Based on the same reasoning behind prediction error methods [15], in selecting the objective function (3) we choose the loss

$$\ell(U, Y, \Theta, S) = \sum_{t=1}^T \varepsilon_t^2(\theta_{s_t}). \tag{8}$$

The chosen cost function $J(U, Y, \Theta, S)$ is optimized with respect to Θ and S as outlined in Algorithm 1. The algorithm minimizes J in (3) (with loss function ℓ as in (8)) by a coordinate descent approach. At each iteration k, the parameters Θ (Step 1.1) and the mode sequence S (Step 1.2) are updated as

$$\Theta^k \leftarrow \arg\min_\Theta \; \ell(U, Y, \Theta, S^{k-1}) + r(\Theta) \tag{9a}$$

$$S^k \leftarrow \arg\min_S \; \ell(U, Y, \Theta^k, S) + \mathcal{L}(S), \tag{9b}$$

respectively, where S^{k-1} denotes the estimate of the mode sequence obtained at iteration $k-1$. The algorithm terminates when no improvement in the cost J is observed.

1) Computation of Θ:

Using the definition of the loss ℓ given in (8), for a fixed mode sequence S^{k-1} the update of the estimate Θ^k in (9a) requires the solution of the nonconvex optimization problem

$$\min_\Theta \sum_{t=1}^T \varepsilon_t^2(\theta_{s_t^{k-1}}) + r(\Theta), \tag{10}$$

Algorithm 1 JARMAX model fitting

Input: Training set $U = (u_1, \dots, u_T)$, $Y = (y_1, \dots, y_T)$, number K of submodels; order of the filters n_a, n_b, n_c; initial guesses Θ^0 and $S^0 = (s_0^0, \dots, s_T^0)$; tolerance $\epsilon_J > 0$; maximum number $k_{\max}$ of iterations.

```
1. iterate for $k = 1, \dots$
  1.1. $\Theta^k \leftarrow$ solution of $\min_\Theta \ell(U, Y, \Theta, S^{k-1}) + r(\Theta)$ from initial guess $\Theta^{k-1}$;
  1.2. $S^k \leftarrow$ solution of $\min_S \ell(U, Y, \Theta^k, S) + \mathcal{L}(S)$;
2. until $J(U, Y, \Theta^{k-1}, S^{k-1}) - J(U, Y, \Theta^k, S^k) \le \epsilon_J$ or $k = k_{\max}$
```

Output: Estimated model parameters $\Theta^\star = \Theta^k$ and mode sequence $S^\star = S^k$.

with the time-domain evolution of the prediction error $\varepsilon_t(\theta_{s_t^{k-1}})$ described by (7).

Problem (10) can be solved by any unconstrained nonlinear optimization method, such as the steepest descent or Gauss-Newton method with line search [6, Chapter 9]. The gradient $\nabla_\Theta J$ of the cost J is given by

$$\nabla_\Theta J(\Theta) = 2\sum_{t=1}^{T} \varepsilon_t(\theta_{s_t^{k-1}}) \frac{\partial \varepsilon_t(\theta_{s_t^{k-1}})}{\partial \Theta} + 2\delta\Theta, \tag{11a}$$

while the Hessian $\nabla_\Theta^2 J$ can be approximated as

$$\nabla_\Theta^2 J(\Theta) \approx 2\sum_{t=1}^{T} \frac{\partial \varepsilon_t(\theta_{s_t^{k-1}})}{\partial \Theta} \left(\frac{\partial \varepsilon_t(\theta_{s_t^{k-1}})}{\partial \Theta}\right)' + 2\delta I. \tag{11b}$$

For a fixed mode sequence S^{k-1}, based on the time evolution of the prediction error in (7), the partial derivatives of the prediction error $\varepsilon_t(\theta_{s_t^{k-1}})$ with respect to the components of Θ can be computed by solving the following difference equations:

$$C(q, \theta_{s_t^{k-1}}) \frac{\partial \varepsilon_t(\theta_{s_t^{k-1}})}{\partial a_j^i} = y_{t-j} \mathbf{1}_{[s_t^{k-1}=i]}, \quad \text{for } j = 1, \dots, n_a \tag{12a}$$

$$C(q, \theta_{s_t^{k-1}}) \frac{\partial \varepsilon_t(\theta_{s_t^{k-1}})}{\partial b_j^i} = -u_{t-j} \mathbf{1}_{[s_t^{k-1}=i]}, \quad \text{for } j = 1, \dots, n_b \tag{12b}$$

$$C(q, \theta_{s_t^{k-1}}) \frac{\partial \varepsilon_t(\theta_{s_t^{k-1}})}{\partial c_j^i} = -\varepsilon_{t-j}(\theta_{s_t^{k-1}}) \mathbf{1}_{[s_t^{k-1}=i]}, \quad \text{for } j = 1, \dots, n_c, \tag{12c}$$

where $\mathbf{1}_{[s_t^{k-1}=i]}$ denotes the indicator function

$$\mathbf{1}_{[s_t^{k-1}=i]} = \begin{cases} 1 & \text{if } s_t^{k-1} = i, \\ 0 & \text{otherwise.} \end{cases}$$

2) Computation of S:

To compute the active mode sequence S^k at Step 1.2 of Algorithm 1, we use *dynamic programming* (DP) [2]. Let $h \in \mathcal{K}^{n_c+1}$, $h = [h_0\ h_1\ \dots\ h_{n_c}]'$, denote a subsequence of current and past n_c modes. For each time instant $t \in \{1, \dots, T\}$, the prediction error has to be simulated through (5), i.e.,

$$\varepsilon_t(\theta_{h_0}, h) = -\sum_{j=1}^{n_c} c_j^{h_0} \varepsilon_{t-j}(\theta_{h_j}) + y_t + \sum_{j=1}^{n_a} a_j^{h_0} y_{t-j} - \sum_{j=1}^{n_b} b_j^{h_0} u_{t-j}, \tag{13}$$

for all possible values of h_0 and past histories, requiring the evaluation of an exponential number of configurations. Instead of solving problem (9b) exactly, we introduce the approximated DP scheme described next.

At each step t and for a generic value of the mode vectors h, consider the approximated prediction error

$$\tilde{\varepsilon}_t(\theta_{h_0}, h) = -\sum_{j=1}^{n_c} c_j^{h_0} \tilde{\varepsilon}_{t-j}(\theta_{h_j}, \hat{h}^j) + y_t + \sum_{j=1}^{n_a} a_j^{h_0} y_{t-j} - \sum_{j=1}^{n_b} b_j^{h_0} u_{t-j}, \tag{14}$$

where $\hat{h}^j$ is the optimal subsequence of n_c+1 modes starting from h_j obtained at the $t-j$-th DP step. The term $\tilde{\varepsilon}_t(\theta_{h_0}, h)$ in (14) is an approximation of the actual prediction error at time t constructed from (7) for $s_{t-j} = h_j$, $j = 0, \dots, n_c$.

For each time $t = 0, 1, \dots, T$ we can compute K^{n_c+1} the cost matrices $M_h(t)$ via the following modified Viterbi algorithm:

$$M_h(0) = \mathcal{L}^{\text{init}}(h_0), \quad \forall h \in \mathcal{K}^{n_c+1} \tag{15a}$$

$$M_h(t) = \mathcal{L}^{\text{mode}}(h_0) + \tilde{\varepsilon}_t^2(\theta_{h_0}, h) + \min_{\substack{g \,\in\, \mathcal{K}^{n_c+1}:\ g_j = h_{j+1} \\ j = 0, \dots, n_c}} M_g(t-1) + \mathcal{L}^{\text{trans}}(h_0, g_0), \quad t = 1, \dots, T, \quad \forall h \in \mathcal{K}^{n_c+1}. \tag{15b}$$

Matrix $M_h(t)$ is the minimum of the loss cumulated up to time t when the current mode is $s_t = h_0$ and the previous n_c modes are $s_{t-j} = h_j$, $j = 1, \dots, n_c$. Setting $g_{0:n_c-1} = h_{1:n_c}$ in (15b) constrains the sequence g of current+past active $n_c - 1$ modes at time $t-1$ to coincide with the past n_c of h. By saving the optimal indexes

$$U_h(t) = \arg\min_{\substack{g \,\in\, \mathcal{K}^{n_c+1}:\ g_j = h_{j+1} \\ j = 0, \dots, n_c}} M_g(t-1) + \mathcal{L}^{\text{trans}}(h_0, g_0), \quad t = 1, \dots, T \quad \forall h \in \mathcal{K}^{n_c+1}, \tag{15c}$$

backwards iterations can be performed to retrieve the optimal mode sequence as follows:

$$h(T) = \arg\min\nolimits_{h \in \mathcal{K}^{n_c+1}} M_h(T) \tag{15d}$$

$$h(t) = U_h(t+1) \tag{15e}$$

$$s_t = h_0(t), \quad t = 0, \dots, T-1. \tag{15f}$$

The result returned by Algorithm 1 depends on the initial guess S^0 of the mode sequence. Due to the nonconvexity of problem (10), it may also depend on the initial guess Θ_0, unless (10) is solved to global optimality. The effect of the initial guess can be mitigated, and the quality of the estimated model improved, by running Algorithm 1 N times, starting from different initial guesses and selecting the one which leads to the smallest value of the fitting cost J in (3).

Algorithm 2 JARMAX batch inference

```
Input: Model $\Theta^\star$, set of inputs $\tilde{U}_t$, past outputs $\tilde{Y}_{t-1}$.
1. $\hat{S}_t \leftarrow \arg\min_{S_t}\Big(\mathcal{L}(S_t)+\ell(\tilde{U}_{t-1},\tilde{Y}_{t-1},\Theta^\star,S_{t-1}) + \min_{y\in\mathcal{Y}_t}\{\varepsilon_t^2(\theta^\star_{s_t},y)\}\Big)$
2. $\hat{y}_t \leftarrow \arg\min_{y\in\mathcal{Y}_t}\varepsilon_t^2(\theta^\star_{\hat{s}_t},y)$
Output: Estimated output $\hat{y}_t$ and mode sequence $\hat{S}_t$.
```

IV. Inference

Given a new set of inputs $\tilde{U}_t=(\tilde{u}_1,\dots,\tilde{u}_t)$ and outputs $\tilde{Y}_{t-1}=(\tilde{y}_1,\dots,\tilde{y}_{t-1})$, not used to train the JARMAX model, the parameters $\Theta^\star$ estimated through Algorithm 1 can be used to predict the output $\tilde{y}_t$ and infer the state sequence $\tilde{S}_t=(\tilde{s}_0,\dots,\tilde{s}_t)$. To this aim, we exploit the ideas in [3] to predict $\tilde{y}_t$ and retrieve $\tilde{S}_t$ from $\tilde{U}_t$ and $\tilde{Y}_{t-1}$.

A. *One-step ahead prediction (batch mode)*

In one-step ahead prediction we want to predict $\tilde{y}_t$ and estimate the mode sequence S_t given the partial sequences of inputs $\tilde{U}_t=(\tilde{u}_1,\dots,\tilde{u}_t)$ and past outputs $\tilde{Y}_{t-1}=(\tilde{y}_1,\dots,\tilde{y}_{t-1})$. Given $\Theta^\star$, the estimates $\hat{y}_t$ and $\hat{S}_t$ can be obtained by considering the same objective function used to learn JARMAX as follows:

$$(\hat{y}_t,\hat{S}_t)=\arg\min_{y,S_t} J_t(\tilde{U}_t,\{\tilde{Y}_{t-1},y\},\Theta^\star,S_t) \quad \text{s.t. } y\in\mathcal{Y}_t, \tag{16}$$

where $\mathcal{Y}_t\subseteq\mathcal{Y}$ is a feasible set constructed based on additional information we might have on the output ($\mathcal{Y}_t=\mathcal{Y}$ if no additional information is available). The loss function J_t at time t is defined as in (3), i.e.,

$$J_t=\ell(\tilde{U}_t,\{\tilde{Y}_{t-1},y\},\Theta^\star,S_t)+\mathcal{L}(S_t), \tag{17}$$

where the regularization $r(\Theta^\star)$ is omitted as it does not depend on y and S_t.

Based on the selected loss function ℓ given in (8), the function $\ell(\tilde{U}_t,\{\tilde{Y}_{t-1},y\},\Theta^\star,S_t)$ in J_t is equal to

$$\begin{aligned}\ell(\tilde{U}_t,\{\tilde{Y}_{t-1},y\},\Theta^\star,S_t)&=\left[\sum_{\tau=1}^{t-1}\varepsilon_\tau^2(\theta^\star_{s_\tau})+\varepsilon_t^2(\theta^\star_{s_t},y)\right]\\&=\ell(\tilde{U}_{t-1},\tilde{Y}_{t-1},\Theta^\star,S_{t-1})+\varepsilon_t^2(\theta^\star_{s_t},y),\end{aligned} \tag{18}$$

where the dependency of the prediction error ε_t on the (unknown) output y is explicitly indicated to emphasize that this term is unknown, as $\tilde{y}_t$ is not available at time t.

According to the method proposed in [3], problem (16) is solved as summarized in Algorithm 2. Specifically, Step 1 is still solved though discrete DP, with the terminal penalty $\mathcal{L}^{mode}(s_t)+\min_{y\in\mathcal{Y}_t}\{\varepsilon_t^2(\theta^\star_{s_t},y)\}$ accounting for the fact that $\tilde{y}_t$ is not known. From (5) it can be noticed that, in the case of unconstrained output (i.e., $\mathcal{Y}_t=\mathbb{R}$), the term $\min_{y\in\mathcal{Y}_t}\{\varepsilon_t^2(\theta^\star_{s_t},y)\}$ achieves its minimum, equal to 0, at $y=\hat{y}_t(\theta_{s_t}|t-1)$, with $\hat{y}_t(\theta_{s_t}|t-1)$ defined as in (6).

B. *One-step ahead prediction (recursive mode)*

We now detail an approximated incremental version of Algorithm 2, which provides a one-step ahead prediction of $\tilde{y}_t$ without the need to run Algorithm 2 each time a new data sample is available.

Consider again the function to be minimized at Step 1 of Algorithm 2. We introduce the partial cost $\mathcal{L}_t(s)$ at time t and for $s\in\mathcal{K}$, given by

$$\mathcal{L}_t(s)=\mathcal{L}(\{S_{t-1},s\})+\ell(\tilde{U}_{t-1},\{\tilde{Y}_{t-2},\tilde{y}_{t-1}\},\Theta^\star,S_{t-1}), \tag{19}$$

where

$$\begin{aligned}&\ell(\tilde{U}_{t-1},\{\tilde{Y}_{t-2},\tilde{y}_{t-1}\},\Theta^\star,S_{t-1})=\\&\quad=\ell(\tilde{U}_{t-2},\tilde{Y}_{t-2},\Theta^\star,S_{t-2})+\varepsilon_{t-1}^2(\theta^\star_{s_{t-1}},\tilde{y}_{t-1}).\end{aligned} \tag{20}$$

The dependency of the prediction error ε_{t-1} on the output $\tilde{y}_{t-1}$ is explicitly shown in (20) to underline that ε_{t-1} can be computed based on $\tilde{y}_{t-1}$, which is available only from time t on. Instead, the loss $\ell(\tilde{U}_{t-2},\tilde{Y}_{t-2},\Theta^\star,S_{t-2})$

$$\ell(\tilde{U}_{t-2},\tilde{Y}_{t-2},\Theta^\star,S_{t-2})=\sum_{\tau=1}^{t-2}\varepsilon_\tau^2(\theta^\star_{s_\tau})$$

can be computed at time $t-1$ based on the outputs $\tilde{Y}_{t-2}=(\tilde{y}_1,\dots,\tilde{y}_{t-2})$.

From the definition of the mode sequence loss given in (4), $\mathcal{L}(\{S_{t-1},s\})$ can be written as

$$\begin{aligned}\mathcal{L}(\{S_{t-1},s\})&=\mathcal{L}^{\text{init}}(s_0)+\mathcal{L}^{\text{mode}}(s)+\mathcal{L}^{\text{trans}}(s,s_{t-1})\\&\quad+\sum_{\tau=1}^{t-1}\left[\mathcal{L}^{\text{mode}}(s_\tau)+\mathcal{L}^{\text{trans}}(s_\tau,s_{\tau-1})\right]\\&=\mathcal{L}(S_{t-1})+\mathcal{L}^{\text{mode}}(s)+\mathcal{L}^{\text{trans}}(s,s_{t-1}).\end{aligned} \tag{21}$$

Substituting (21) into (19) and noticing that

$$\mathcal{L}(S_{t-1})+\sum_{\tau=1}^{t-2}\varepsilon_\tau(\theta^\star_{s_\tau})^2=\mathcal{L}_{t-1}(s_{t-1}) \tag{22}$$

we obtain the following recursive formula for $\mathcal{L}_t(s)$

$$\begin{aligned}\mathcal{L}_t(s)=&\mathcal{L}_{t-1}(s_{t-1})+\mathcal{L}^{\text{mode}}(s)+\mathcal{L}^{\text{trans}}(s,s_{t-1})+\\&+\varepsilon_{t-1}^2(\theta^\star_{s_{t-1}},\tilde{y}_{t-1}).\end{aligned} \tag{23}$$

For a given s, we consider the value s_{t-1} that minimizes $\mathcal{L}_t(s)$. This leads to define

$$\begin{aligned}\bar{\mathcal{L}}_t(s)&=\mathcal{L}^{\text{mode}}(s)+\\&+\min_{j\in\mathcal{K}}\left\{\bar{\mathcal{L}}_{t-1}(j)+\mathcal{L}^{\text{trans}}(s,j)+\varepsilon_{t-1}^2(\theta^\star_j,\tilde{y}_{t-1})\right\}.\end{aligned} \tag{24}$$

We can thus infer the current mode $\hat{s}_t$ as follows:

$$\hat{s}_t=\arg\min_s\left(\bar{\mathcal{L}}_t(s)+\min_{y\in\mathcal{Y}_t}\left\{\varepsilon_t^2(\theta^\star_s,y)\right\}\right). \tag{25}$$

In computing the prediction error $\varepsilon_{t-1}(\theta^\star_j,\tilde{y}_{t-1})$ through (7), the prediction errors $\varepsilon_\tau(\theta^\star_{\hat{s}_\tau})$, with $\tau=t-2,\dots,t-n_c-1$, obtained previously are used. This introduces an approximation in the minimization of the cost J_t in (16) with respect to the whole mode sequence S_t.

Algorithm 3 JARMAX recursive inference

```
Input: ARMAX models' parameters $\Theta^\star$; current input $\tilde{u}_t$; past input/output pairs $(\tilde{u}_1, \tilde{y}_1), \dots, (\tilde{u}_{t-1}, \tilde{y}_{t-1})$, arrival cost $\bar{\mathcal{L}}_{t-1}(j)$, $j = 1, \dots, K$

1. Update $\bar{\mathcal{L}}_t(s)$ as in (24),    $s = 1, \dots, K$;
2. Compute
       $(\hat{y}_t, \hat{s}_t) = \arg\min_{y,s} \left(\varepsilon_t^2(\theta_s^\star, y) + \bar{\mathcal{L}}_t(s)\right)$ s.t. $y \in \mathcal{Y}_t$

Output: Estimated output $\hat{y}_t$ and active mode $\hat{s}_t$
```

Remark 1: When the output $\tilde{y}_t$ is available, we can predict the mode s_t using the measured output as in (25), i.e.,

$$\hat{s}_t = \arg\min_s \left(\varepsilon_t(\theta_s^\star, \tilde{y}_{t-1})^2 + \mathcal{L}_t(s)\right), \qquad (26)$$

where the prediction error ε_t has to be computed considering all the possible sequences $(\hat{s}_0, \dots, \hat{s}_{t-1}, s)$, for $s = 1, \dots, K$. ■

V. Simulation Example

We test the approaches discussed in the previous sections on a simulation example, with a dataset of length $T = 30000$ available for training and $\tilde{T} = 10000$ samples used for testing. The data are generated by a JARMAX system described as in (1) with $K = 3$ modes. All the ARMAX submodels are described by second order filters $A(q, \theta_i)$, $B(q, \theta_i)$ and $C(q, \theta_i)$ (i.e., $n_a = n_b = n_c = 2$), for $i = 1, 2, 3$, whose coefficients are reported in Table I. The input u is a sequence of random *i.i.d.* samples uniformly distributed in the interval $[-1,\ 1]$, and the noise e_t corrupting the output observations is a Gaussian-distributed random variable with zero mean and standard deviation $\sigma_e = 0.45$. This corresponds to the *Signal-to-Noise-Ratio* (SNR)

$$\text{SNR} = 10 \log \frac{\sum_{t=1}^{\tilde{T}} (\tilde{y}_t - e_t)^2}{\sum_{t=1}^{\tilde{T}} e_t^2} = 7.5 \text{ dB} \qquad (27)$$

The true mode sequence S is obtained changing the operating condition every 500 samples, starting from $s_0 = 3$.

The cost J in (3) is defined by setting $\delta = 0.001$, $\mathcal{L}^{\text{init}}(j) = \mathcal{L}^{\text{mode}}(j) = 0$, for all $j = 1, \dots, K$, and

$$\mathcal{L}^{\text{trans}}(i, j) = \tau \mathbf{1}_{[i=j]}, \ \ i, j = 1, 2, 3, \qquad (28)$$

where $\tau \in \mathbb{R}$ is tuned via cross validation and it is set to 2.5. Algorithm 1 is run $N = 5$ times from randomly generated initial mode sequences S^0, by setting Θ^0 equal to a zero matrix, selecting the outcome that provides the minimum value of the cost J. The maximum number of iterations k_{max} and the tolerance ϵ_J are set to 50 and 10^{-16}, respectively.

The update of Θ^k in Step 1.1 of Algorithm 1 is computed by solving problem (9a) via Gauss-Newton with exact line search based on 100 equally-spaced grid points in the interval $[0\ 1]$. The Gauss-Newton method is stopped if either the maximum number $j_{\max} = 50$ of iterations is reached, or if the following terminal condition

$$(\nabla_\Theta J(\Theta^{k,j}))'(\nabla_\Theta^2 J(\Theta^{k,j}))^{-1} \nabla_\Theta J(\Theta^{k,j}) \leq \epsilon_{GN}, \qquad (29)$$

TABLE I

True vs estimated parameters

		True	JARMAX	JARX
a_1^j	$j=1$	0.9	0.902	-0.659
	$j=2$	0.4	0.395	-0.191
	$j=3$	-0.4	-0.403	0.421
a_2^j	$j=1$	0.3	0.302	-0.072
	$j=2$	0.8	0.800	-0.727
	$j=3$	0.2	0.193	-0.363
b_1^j	$j=1$	1	1.001	1.001
	$j=2$	0.1	0.095	0.102
	$j=3$	0.5	0.500	0.496
b_2^j	$j=1$	-0.5	-0.507	-0.746
	$j=2$	0.5	0.500	0.489
	$j=3$	0.9	0.893	0.879
c_1^j	$j=1$	0.4	0.393	-
	$j=2$	0.7	0.683	-
	$j=3$	0.1	0.098	-
c_2^j	$j=1$	0.4	0.394	-
	$j=2$	0.2	0.200	-
	$j=3$	-0.7	-0.700	-

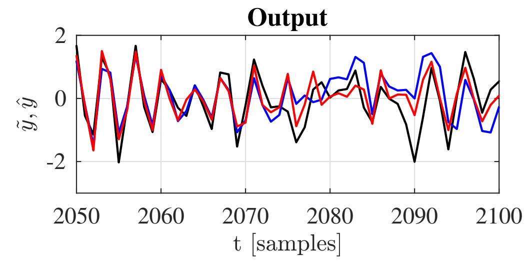

Fig. 1. Output signal: true noise-free $\tilde{y}_t$ (black); predicted $\hat{y}_t$ with JARX model (blue), predicted $\hat{y}_t$ with JARMAX model (red).

is satisfied, with $\epsilon_{GN} = 10^{-8}$.

The true and estimated parameters of the JARMAX model are compared in Table I, which shows that the parameters of the local ARMAX models are accurately identified. In the same table, we also report the parameters obtained by training a *jump ARX* (JARX) model ($n_c = 1$). It can clearly be noticed that the estimated parameters are biased, due to the inconsistent noise model structure. The trajectory of the output signal is plotted in Fig. 1, which shows a better match between the true output and the one predicted by the JARMAX model with respect to the one achieved by the JARX model. By assessing the quality of the predicted output through the *mean square error* (MSE), i.e.,

$$\text{MSE} = \frac{1}{\tilde{T}} \sum_{t=1}^{\tilde{T}} (\tilde{y}_t - \hat{y}_t)^2, \qquad (30)$$

we obtain that MSE $= 0.22$ for the JARMAX model and MSE $= 0.29$ for the JARX model.

To evaluate the quality of the inferred mode sequence $\hat{S}$ over the test set, we consider the mismatch index

$$\ell_S^{true} = \frac{100}{\tilde{T}} \sum_{t=1}^{\tilde{T}} \mathbf{1}_{[\hat{s}_t \neq \tilde{s}_t]}, \qquad (31)$$

with the true mode sequence $\tilde{S}_{\tilde{T}}$ on the test set assumed available only for validation purposes. We obtain $\ell_S^{true} = 0.93\%$ and $\ell_S^{true} = 1.82\%$ using a JARMAX and a JARX model, respectively. As shown in Fig. 2, the JARX model

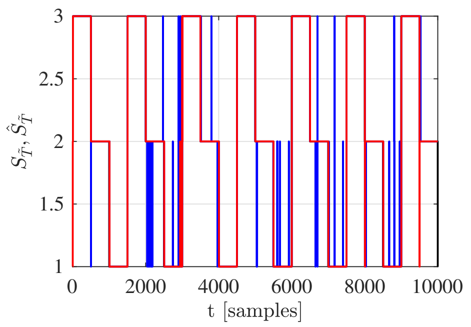

Fig. 2. Test set: mode sequence. True $S_{\tilde{T}}$ (black); estimated $\hat{S}_{\tilde{T}}$ with JARX model (blue), estimated $\hat{S}_{\tilde{T}}$ with JARMAX model (red). Black and red lines are almost overlapped.

TABLE II

MONTE CARLO SIMULATION: JARMAX MODEL. TRUE VS ESTIMATED PARAMETERS (MEAN $\pm$ STANDARD DEVIATION)

		True	Estimated
a_1^j	$j=1$	0.9	0.901 $\pm$ 0.009
	$j=2$	0.4	0.397 $\pm$ 0.005
	$j=3$	-0.4	-0.400 $\pm$ 0.009
a_2^j	$j=1$	0.3	0.301 $\pm$ 0.007
	$j=2$	0.8	0.802 $\pm$ 0.006
	$j=3$	0.2	0.200 $\pm$ 0.007
b_1^j	$j=1$	1	1.000 $\pm$ 0.009
	$j=2$	0.1	0.104 $\pm$ 0.008
	$j=3$	0.5	0.500 $\pm$ 0.009
b_2^j	$j=1$	-0.5	-0.499 $\pm$ 0.014
	$j=2$	0.5	0.507 $\pm$ 0.009
	$j=3$	0.9	0.901 $\pm$ 0.007
c_1^j	$j=1$	0.4	0.397 $\pm$ 0.010
	$j=2$	0.7	0.692 $\pm$ 0.013
	$j=3$	0.1	0.099 $\pm$ 0.008
c_2^j	$j=1$	0.4	0.393 $\pm$ 0.010
	$j=2$	0.2	0.195 $\pm$ 0.016
	$j=3$	-0.7	-0.700 $\pm$ 0.009

exhibits multiple sudden jumps, which are not representative of the behavior of the actual data-generating system.

Finally, robustness of the learning method is assessed performing a Monte Carlo simulation with 20 realizations of the input and noise signals. The mean and the standard deviations for the estimated parameters of the JARMAX model are reported in Table II, which shows that the true parameters are always within the uncertainty intervals defined by the standard deviation.

VI. CONCLUSIONS

We have proposed a new algorithm for inference and learning of *Jump AutoRegressive Moving Average with eXogenous inputs* (JARMAX) models. The proposed approach is built upon the optimization-based formulation for jump model fitting presented in [3], using objective functions derived from prediction error methods. A coordinate descent algorithm is employed to alternate between estimation of the ARMAX submodels' parameters (via unconstrained nonlinear optimization) and the sequence of active modes (via dynamic programming).

Extensions of this work include the identification of jump models with more general Box-Jenkins submodels, and the use of different types of cost J to address identification of other classes of jump models, such as PWA models with local ARMAX structures. The analysis of the statistical properties of the estimated models is the subject of ongoing research.

REFERENCES

[1] L. Bako. Identification of switched linear systems via sparse optimization. *Automatica*, 47(4):668–677, 2011.
[2] R. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1957.
[3] A. Bemporad, V. Breschi, D. Piga, and S. P. Boyd. Fitting jump models. *Automatica*, 96:11 – 21, 2018.
[4] A. Bemporad and S. Di Cairano. Model-predictive control of discrete hybrid stochastic automata. *IEEE Transactions on Automatic Control*, 56(6):1307–1321, June 2011.
[5] A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407 – 427, 1999.
[6] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.
[7] L. Breiman. Hinging hyperplanes for regression, classification, and function approximation. *IEEE Transactions on Information Theory*, 39(3):999–1013, 1993.
[8] V. Breschi, D. Piga, and A. Bemporad. Learning hybrid models with logical and continuous dynamics via multiclass linear separation. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 353–358, Dec 2016.
[9] V. Breschi, D. Piga, and A. Bemporad. Piecewise affine regression via recursive multiple least squares and multicategory discrimination. *Automatica*, 73:155–162, 2016.
[10] D. Chen, L. Bako, and S. Lecuche. A recursive sparse learning method: Application to jump markov linear systems. *IFAC Proceedings Volumes*, 44(1):3198 – 3203, 2011. 18th IFAC World Congres.
[11] O.L.V. Costa, M.D. Fragoso, and R.P. Marques. *Discrete-Time Markov Jump Linear Systems*. Probability and Its Applications. Springer London, 2006.
[12] G. Ferrari-Trecate, M. Muselli, D. Liberati, and M. Morari. A clustering technique for the identification of piecewise affine systems. *Automatica*, 39(2):205–217, 2003.
[13] E. Fox, E.B. Sudderth, M.I. Jordan, and A.S. Willsky. Bayesian nonparametric inference of switching dynamic linear models. *IEEE Transactions on Signal Processing*, 59(4):1569–1585, 2011.
[14] T.A. Henzinger. The theory of hybrid automata. In *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science*, LICS '96, pages 278–, Washington, DC, USA, 1996. IEEE Computer Society.
[15] L. Ljung. *System identification: theory for the user*. Prentice-Hall Englewood Cliffs, NJ, 1999.
[16] R.B. Mrad. Non-linear systems representation using armax models with time-dependent coefficients. *Mechanical Systems and Signal Processing*, 16(5):803 – 815, 2002.
[17] B. North, A. Blake, M. Isard, and J. Rittscher. Learning and classification of complex dynamics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(9):1016–1034, Sep 2000.
[18] H. Ohlsson and L. Ljung. Identification of switched linear regression models using sum-of-norms regularization. *Automatica*, 49(4):1045–1050, 2013.
[19] D. Piga and R. Tóth. An SDP approach for ℓ_0-minimization: Application to ARX model segmentation. *Automatica*, 49(12):3646–3653, 2013.
[20] L.R. Rabiner. Readings in speech recognition. chapter A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, pages 267–296. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.
[21] J. Roll, A. Bemporad, and L. Ljung. Identification of piecewise affine systems via mixed-integer programming. *Automatica*, 40(1):37–50, 2004.
[22] X. Rong Li and V.P. Jilkov. Survey of maneuvering target tracking. part v. multiple-model methods. *IEEE Transactions on Aerospace and Electronic Systems*, 41(4):1255–1321, Oct 2005.
[23] F.D. Torrisi and A. Bemporad. Hysdel-a tool for generating computational hybrid models for analysis and synthesis problems. *IEEE Transactions on Control Systems Technology*, 12(2):235–249, March 2004.