Automatica 120 (2020) 109126

Contents lists available at ScienceDirect

Automatica

journal homepage: www.elsevier.com/locate/automatica

Estimation of jump Box–Jenkins models[☆]

Dario Piga [a,*], Valentina Breschi [b], Alberto Bemporad [c]

[a] *IDSIA Dalle Molle Institute for Artificial Intelligence Research - USI/SUPSI, Manno, Switzerland*
[b] *Dipartimento di Elettronica e Informazione, Politecnico di Milano, Milano, Italy*
[c] *IMT School for Advanced Studies Lucca, Lucca, Italy*

ARTICLE INFO

Article history:
Received 24 May 2019
Received in revised form 27 April 2020
Accepted 15 June 2020
Available online xxxx

Keywords:
Box–Jenkins models
Jump models
Hybrid models
Prediction error methods
Maximum-a-posteriori estimation

ABSTRACT

Jump *Box–Jenkins* (BJ) models are a collection of a finite set of linear dynamical submodels in BJ form that switch over time, according to a Markov chain. This paper addresses the problem of *maximum-a-posteriori* estimation of jump BJ models from a given training input/output dataset. The proposed solution method estimates the coefficients of the BJ submodels, the state transition probabilities of the Markov chain regulating the switching of operating modes, and the corresponding mode sequence hidden in the dataset. In particular, the posterior distribution of all the unknown variables characterizing the jump BJ model is derived and then maximized using a coordinate ascent algorithm. The resulting estimation algorithm alternates between Gauss–Newton optimization of the coefficients of the BJ submodels, a method derived based on an instance of *prediction error methods* tailored to BJ models with switching coefficients, and approximated dynamic programming for optimization of the sequence of active modes. The quality of the proposed estimation approach is evaluated on a numerical example based on synthetic data and in a case study related to segmentation of honeybee dances.

1. Introduction

1.1. Hybrid modelling framework

The behaviour of many real-world systems is characterized by the interaction between *discrete* (logical) and *continuous* (physical) states. Examples include power electronic circuits, electrical household appliances, robot grasping (alternation of free and contact motion), just to cite a few. Hybrid models allow describing such nonlinear and complex systems as multiple, but simple, submodels with continuous state variables, each one characterizing the local behaviour of the system at a given operating mode (Bemporad & Morari, 1999; Henzinger, 1996; Sontag, 1981; Torrisi & Bemporad, 2004).

A similar hybrid modelling framework is also used in temporal segmentation of time series, where segments are clustered according to some (usually hidden) temporal patterns. Examples of time series segmentation include: human motion and action recognition, which consists of recognizing the activities of a person (e.g., walking, running, jumping) from video frames or motion capture sensors (Fox, Hughes, Sudderth, & Jordan, 2014; Gong, Medioni, & Zhao, 2014; Ozay, Lagoa, & Sznaier, 2015); audio segmentation for automatic speech recognition (Ostendorf, Digalakis, & Kimball, 1996); stock market analysis under different volatility regimes (Fu, Chung, Ng, & Luk, 2001; Nguyen, 2018); and many others. In these applications, the discrete state is given by the cluster where the segment belongs to, while each local submodel describes the behaviour of the time series within the temporal segment.

Depending on the modelling application, the time evolution of the *discrete state* can be governed by *deterministic* rules, as in hybrid automata (Henzinger, 1996; Torrisi & Bemporad, 2004), mixed logical dynamical models (Bemporad & Morari, 1999), and piecewise-affine (PWA) models (Sontag, 1981), or by *stochastic* jumps, as in discrete hybrid stochastic automata (Bemporad & Di Cairano, 2011), hidden Markov models (HMM) (Rabiner, 1990) and, more in general, Markov jump models (Bemporad, Breschi, Piga, & Boyd, 2018; Costa, Fragoso, & Marques, 2006).

1.2. Paper contribution

This paper addresses the estimation of jump dynamical models where the time transition of the discrete state is described by a stochastic Markov chain and in each operating mode the dynamics are described by Box–Jenkins (BJ) model. BJ structures have the advantage of being more general and flexible than simple *autoregressive* (AR) models, due the presence of both the moving

[☆] The material in this paper was partially presented at the 58th IEEE Conference on Decision and Control, December 11–13, 2019, Nice, France. This paper was recommended for publication in revised form by Associate Editor Alessandro Chiuso under the direction of Editor Torsten Söderström.

[*] Corresponding author.

E-mail addresses: dario.piga@supsi.ch (D. Piga), valentina.breschi@polimi.it (V. Breschi), alberto.bemporad@imtlucca.it (A. Bemporad).

https://doi.org/10.1016/j.automatica.2020.109126
0005-1098/© 2020 Elsevier Ltd. All rights reserved.

average and the autoregressive term, and have widely proven their efficiency in time series analysis and forecasting (Box, Jenkins, Reinsel, & Ljung, 2015). The main difficultly in data-driven modelling of jump BJ models is caused by the breaking of the Markovian structure, as the output observations are not anymore conditionally independent given the current mode and the regressor containing past measured data. Furthermore, even in the simpler case of non-switching (namely, time-invariant) BJ models, the estimation of the model coefficients usually requires solving a nonconvex optimization problem (Ljung, 1999).

In this paper, we derive the posterior distribution of all the parameters defining the jump BJ model, namely the coefficients Θ of the time-invariant BJ submodels, the transition matrix M governing the switching of the (stochastic) discrete state, the variance σ_v^2 of the noise affecting the output observations, and the hidden sequence $\mathcal{S}^T$ of active modes. Then, the posterior distribution is maximized by a coordinate-ascent optimization algorithm which alternates between two steps: (i) maximization w.r.t. the Θ, σ_v^2 and the entries of the transition matrix M; (ii) inference of the active mode sequence $\mathcal{S}^T$. The maximization w.r.t. the model coefficients Θ is carried out by using a Gauss–Newton method, derived based on an instance of *prediction error methods* (Ljung, 1999) tailored to BJ models with switching coefficients. The noise variance σ_v^2 and the transition matrix M maximizing the posterior distribution are computed analytically. Finally, inference of the discrete-state sequence $\mathcal{S}^T$ is performed using a suboptimal moving-horizon approach, which can be interpreted as an approximation of the Viterbi algorithm for discrete *dynamic programming* (DP). It is also shown that such a moving-horizon approach is actually optimal and equivalent to the Viterbi algorithm for the special case of switching autoregressive models.

This paper extends the works in Bemporad et al. (2018), Breschi, Bemporad, Piga, and Boyd (2018) in two main directions: (i) it considers general jump Box–Jenkins models, while only ARX (resp. ARMAX) models can be handled through the formulation in Bemporad et al. (2018) (resp. Breschi et al., 2018); (ii) the approach in Bemporad et al. (2018), Breschi et al. (2018) is based on the minimization of a deterministic cost function, where the hyper-parameters which weight the mode transitions and the norm of the submodel parameters must be tuned by the user. On the other hand, in the MAP formulation considered in this paper such regularization hyper-parameters depend on the entries of the transition matrix M and on the noise variance σ_v^2, and they are optimized along with the mode sequence and parameters of the local models.

1.3. Related works

The presence of unobserved time-varying discrete state makes the data-driven modelling of hybrid and jump models a challenging problem which has attracted the attention of many researchers both from the *system identification* and the *machine learning* community. In order to facilitate the learning task, it is commonly assumed that the outputs are generated by a process satisfying a Markovian assumption. For example, in hidden Markov models the output observations are conditionally independent given the discrete state (Baum, Petrie, Soules, & Weiss, 1970; Fridman, 1994; Rabiner, 1990), which in turn is assumed to be generated by a first-order Markov chain. However, HMMs do not consider the dynamics of the continuous state. This limitation is relaxed in switching autoregressive models, where the current output observation is conditionally independent given both the current discrete state and past (noisy) signal measurements (Bako, 2011; Breschi, Piga, & Bemporad, 2016; Ferrari-Trecate, Muselli, Liberati, & Morari, 2003; Naik, Mejari, Piga, & Bemporad, 2017; Ohlsson & Ljung, 2013; Ozay et al., 2015; Piga, Bemporad, & Benavoli, 2020; Piga & Tóth, 2013). However, simple autoregressive models may lead to inaccurate results both in modelling dynamical systems and in time-series analysis, due to the assumption of noise only influencing the process output.

Only few contributions relax the Markovian assumption in data-driven modelling of switching models in an input–output form, and consider output-error model structures (Canty, O'Mahony, & Cychowski, 2012; Goudjil, Pouliquen, Pigeon, & Gehan, 2017; Rosenqvist & Karlström, 2005), *i.e.*, a special case of the BJ structure considered in this paper. Besides considering a less flexible model structure than BJ, the estimation approaches in Canty et al. (2012), Goudjil et al. (2017), Rosenqvist and Karlström (2005) neither reconstruct nor exploit the stochastic information regarding the switching of the discrete state. This information can be useful, for example, in applications where the knowledge of the discrete state at the previous time step is useful to predict the current discrete state (e.g., in stochastic jump models that switch rarely, given that the probability of remaining in the same mode is very high). In Fox, Sudderth, Jordan, and Willsky (2011), Bayesian estimation of more general switching state-space models is considered. The advantage of Fox et al. (2011) w.r.t. our approach is that the size of the discrete state (or equivalently, the number of linear time-invariant dynamical systems) is not fixed a-priori, but automatically reconstructed from data using an approximated Dirichlet process. However, Gibbs sampling is used to retrieve the continuous state and the unobserved sequence of the discrete state. Thus, the sampling space increases with the size of the training dataset, and many samples may be needed to obtain accurate results with large datasets. On the other hand, no Monte Carlo sampling is used by our approach and, at each step of the coordinate ascent optimization algorithm, the maximization is performed either analytically or iteratively via a Gauss–Newton method.

For completeness, we remark that it is always possible to transform Box–Jenkins models (which are described in an input–output form) into equivalent state-space representations. Thus, jump Box–Jenkins systems could be seen as *Markov Jump Linear Systems* (MJLS) (Costa et al., 2006), with a state-space representation of each local model. Although several methods have been developed for identification of MJLS (Cinquemani, Porreca, Ferrari-Trecate, & Lygeros, 2007; Kun Huang, Wagner, & Yi Ma, 2004; Özkan, Lindsten, Fritsche, & Gustafsson, 2015; Yang, Qin, Pan, Yang, & Li, 2017; Zheng, Derrode, & Pieczynski, 2019), these methods cannot be directly employed to estimate Box–Jenkins systems. In fact, these approaches are developed for state-space models and thus do not take into account the particular structure of the dynamical matrices coming from the state-space realization of Box–Jenkins systems.

Preliminary ideas of this work were presented in Breschi, Piga, and Bemporad (2019). With respect to Breschi et al. (2019), this paper contains rigorous mathematical derivations of the posterior distribution of the model parameters, a more detailed description of the proposed numerical optimization algorithms, a discussion on the problem of inferring the hidden mode sequence and the output signal from past input–output samples, and a more exhaustive evaluation on the performance of the method.

1.4. Paper outline

The paper is organized as follows. Jump Box–Jenkins systems are introduced in Section 2. In Section 3, the posterior distribution of the model parameters is derived, under properly assumed prior distributions. The coordinate ascent algorithm used to compute the maximum of the posterior distribution is described in Section 4. Specifically, Section 4.1 describes the Gauss–Newton method used to optimize w.r.t. the coefficients of the linear dynamical BJ submodels given the mode sequence, while the moving-horizon approach used to optimize w.r.t. the mode sequence (given the other model parameters) is described in Section 4.2. In Section 5, the problem of inferring the hidden mode sequence and the output signal, given past input–output samples, is addressed. Two case studies are reported in Section 6. In the first one, synthetic data are used and numerical analyses are carried out to assess the performance of the proposed identification algorithm. Then, the benchmark example originally proposed in Oh, Rehg, Balch, and Dellaert (2008), and concerning automated segmentation of a honeybee waggle dance, is considered. Conclusions and directions for future research are finally discussed in Section 7.

1.5. Notation

The following notation will be used throughout the paper. The set of integers is denoted by $\mathbb{Z}$, the set of positive real number by $\mathbb{R}^+$, the set of real matrices of dimension $n \times m$ by $\mathbb{R}^{n,m}$. Given a matrix M, $[M]_{i,:}$ denotes its ith row and $[M]_{i,j}$ its entry in position (i,j). For a random matrix $M \in \mathbb{R}^{n,m}$, we refer to $p(M)$ as the probability distribution of $vec(M)$, where $vec(M) \in \mathbb{R}^{nm}$ is the vector obtained by stacking the columns of M on top of one another. The indicator function $\mathbb{I}$ of a logic condition Q is

$$\mathbb{I}(Q) = \begin{cases} 1 & \text{if } Q \text{ is true,} \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

For $\alpha > 0$, $\Gamma(\alpha)$ denotes the *Gamma* function

$$\Gamma(\alpha) = \int_0^{+\infty} x^{\alpha-1} e^{-x}\, dx. \tag{2}$$

2. System description

Let $\mathcal{U}^T = \{u_t\}_{t=1}^T$ be an input sequence, $u_t \in \mathbb{R}$, exciting a single-input single-output dynamical system, and $\mathcal{Y}^T = \{y_t\}_{t=1}^T$, $y_t \in \mathbb{R}$ the corresponding noise-corrupted outputs

$$y_t = y_t^o + v_t. \tag{3a}$$

where $v_t \in \mathbb{R}$ is the noise term. We model the system as a collection of a finite number K of linear submodels, where the noise-free output y_t^o satisfies the equation

$$y_t^o = G(q^{-1}, \theta^{s_t}) u_t, \tag{3b}$$

while v_t is modelled as (coloured) noise satisfying

$$v_t = H(q^{-1}, \theta^{s_t}) e_t, \tag{3c}$$

where e_t is a zero-mean Gaussian random variable generated by a white noise stationary process with variance σ_e^2. The linear filters $G(q^{-1}, \theta^{s_t})$, $H(q^{-1}, \theta^{s_t})$ have time-varying coefficients θ^{s_t}, where the superscript $s_t \in \mathcal{K} = \{1, \ldots, K\}$ denotes the hidden active mode (or equivalently, discrete state) at time t, and q is the time-shift operator (*i.e.*, $q^{-d} u_t = u_{t-d}$, for $d \in \mathbb{Z}$). Furthermore, data are assumed to be generated by an open-loop experiment, thus the input sequence u is independent of the stochastic process generating the noise e_t.

For simplicity of notation, the initial conditions of the signals y, u, v, e are assumed zero.

2.1. Continuous-state dynamics

For a fixed mode $i \in \mathcal{K}$, the dynamical model (3b)–(3c) is assumed to have a *Box–Jenkins* structure, with $G(q^{-1}, \theta^i)$ and $H(q^{-1}, \theta^i)$ being rational functions of the time operator q^{-1}, *i.e.*,

$$G(q^{-1}, \theta^i) = \frac{B(q^{-1}, \theta^i)}{A(q^{-1}, \theta^i)} = \frac{b_1^i q^{-1} + \cdots + b_{n_b}^i q^{-n_b}}{1 + a_1^i q^{-1} + \cdots + a_{n_a}^i q^{-n_a}}, \tag{4a}$$

$$H(q^{-1}, \theta^i) = \frac{C(q^{-1}, \theta^i)}{D(q^{-1}, \theta^i)} = \frac{1 + c_1^i q^{-1} + \cdots + c_{n_c}^i q^{-n_c}}{1 + d_1^i q^{-1} + \cdots + d_{n_d}^i q^{-n_d}}, \tag{4b}$$

In (4b) n_a, n_b n_c and n_d indicate the dynamical order of the BJ submodel, and the parameter vector $\theta^i \in \mathbb{R}^{n_\theta}$ describing the ith Box–Jenkins submodel is given by

$$\theta^i = \begin{bmatrix} a_1^i & \ldots & a_{n_a}^i & b_1^i & \ldots & b_{n_b}^i & c_1^i & \ldots & c_{n_c}^i & d_1^i & \ldots & d_{n_d}^i \end{bmatrix}'. \tag{5}$$

Note that, as commonly assumed in *Prediction Error Methods* (PEM) for LTI system identification (Ljung, 1999), G is a strictly proper filter (*i.e.*, $G(0, \theta^i) = 0$) and H is monic (*i.e.*, $H(0, \theta^i) = 1$) for all $i \in \mathcal{K}$.

It is worth stressing that, because of the time-varying nature of the filters $G(q^{-1}, \theta^{s_t})$ and $H(q^{-1}, \theta^{s_t})$, the rational functions in (4) cannot be simply treated as the ratio of *Z-transforms*: the time-domain equations in (3b)–(3c) have the following meaning in terms of difference equations:

$$y_t^o = G(q^{-1}, \theta^{s_t}) u_t \to y_t^o = -\sum_{k=1}^{n_a} a_k^{s_t} y_{t-k}^o + \sum_{k=1}^{n_b} b_k^{s_t} u_{t-k}, \tag{6a}$$

$$v_t = H(q^{-1}, \theta^{s_t}) e_t \to v_t = -\sum_{k=1}^{n_a} d_k^{s_t} v_{t-k} + \sum_{k=1}^{n_c} c_k^{s_t} e_{t-k}. \tag{6b}$$

Remark 1. In the specific case of switching ARX models, the polynomials $C(q^{-1}, \theta^i)$ and $D(q^{-1}, \theta^i)$ reduce to

$$C(q^{-1}, \theta^i) = 1, \quad D(q^{-1}, \theta^i) = A(q^{-1}, \theta^i), \quad \forall i \in \mathcal{K}. \tag{7a}$$

By summing Eqs. (6), for $C(q^{-1}, \theta^i)$ and $D(q^{-1}, \theta^i)$ in (7a), we thus obtain

$$A(q^{-1}, \theta^{s_t})(y_t^o + v_t) = B(q^{-1}, \theta^{s_t}) u_t + e_t. \tag{7b}$$

Using the definition of the noisy output y_t in (3), Eq. (7b) can be rewritten in terms of the well known expression of switching ARX models (Garulli, Paoletti, & Vicino, 2012):

$$A(q^{-1}, \theta^{s_t}) y_t = B(q^{-1}, \theta^{s_t}) u_t + e_t. \quad \blacksquare \tag{7c}$$

To compact the notation, in the following we denote by Θ the vector stacking the parameters θ^i, *i.e.*, $\Theta = [\theta^{1'} \ \ldots \ \theta^{K'}]' \in \mathbb{R}^{n_\Theta}$, with $n_\Theta = K n_\theta$. Furthermore, the dependence of $G(q^{-1}, \theta^{s_t})$ and $H(q^{-1}, \theta^{s_t})$ on the time-shift operator q^{-1} will be made explicit only when necessary.

2.2. Discrete-state dynamics

The discrete state s_t is not observed, and it is supposed to be generated by a discrete-time stochastic Markov process with state transition matrix M, *i.e.*,

$$p(s_t | s_{t-1}, s_{t-2}, \ldots, s_0) = p(s_t | s_{t-1}) = [M]_{s_{t-1}, s_t}, \qquad t = 1, \ldots, T, \tag{8}$$

with

$$[M]_{i,j} \geq 0, \quad i, j = 1, \ldots, K, \tag{9}$$

$$\sum_{j=1}^{K} [M]_{i,j} = 1, \quad i = 1, \ldots, K, \tag{10}$$

and s_0 is the (unknown) initial discrete state.

In the following, the sequence of discrete states up to time T, including the initial state s_0, is denoted by $\mathcal{S}^T$, *i.e.*, $\mathcal{S}^T = \{s_\tau\}_{\tau=0}^T$.

3. Learning problem

Under the assumption that the size K of the discrete state (namely, the number of local subsystems) is known, the following unknown variables characterize the Box–Jenkins model (3)–(4):

- $\Theta \in \mathbb{R}^{n_\Theta}$: collection of parameters θ^i defining each Box–Jenkins submodel (4);
- $\sigma_e^2 \in \mathbb{R}^+$: variance of noise e_t;
- $M \in \mathbb{R}^{K,K}$: state transition matrix;
- $\mathcal{S}^T$: sequence of (hidden) discrete states.

The objective of this paper is to compute the parameters Θ, σ_e^2, M and the discrete-state sequence $\mathcal{S}^T$ that maximize the joint posterior distribution $p(\Theta, \sigma_e^2, M, \mathcal{S}^T|\mathcal{Y}^T, \mathcal{U}^T)$ given the training output and input sequences $\mathcal{Y}^T$ and $\mathcal{U}^T$.

Remark 2. In case the number K of local linear models is not known a priori, it can be chosen, for instance, via holdout cross validation. This requires one to split the available training dataset into two subsets, one used to estimate the model parameters and the other one to assess the performance of the estimated model. Different values of K should be considered, with an upper-bound dictated by the maximum tolerated complexity of the resulting Jump Box–Jenkins model (3). The value of K leading to the best performance is then chosen. ■

3.1. Priors over the unknown parameters

In order to compute the posterior distribution of the unknown variables $\Theta, \sigma_e^2, M, \mathcal{S}^T$, the following priors are assumed.

1. The unknown variables $\Theta, \sigma_e^2, M, \mathcal{S}^T$ are statistically independent of the input data $\mathcal{U}^T$, *i.e.*,
$$p(\Theta, \sigma_e^2, M, \mathcal{S}^T|\mathcal{U}^T) = p(\Theta, \sigma_e^2, M, \mathcal{S}^T), \tag{11}$$
and the joint prior $p(\Theta, \sigma_e^2, M, \mathcal{S}^T)$ factorizes as
$$p(\Theta, \sigma_e^2, M, \mathcal{S}^T) = p(\Theta, \sigma_e^2)p(M, \mathcal{S}^T). \tag{12}$$
2. The joint probability distribution $p(\Theta, \sigma_e^2)$ is a *Gaussian Inverse-Gamma* with parameters $\lambda, \alpha_0, \beta_0 > 0$, and it factorizes as $p(\Theta, \sigma_e^2) = p(\Theta|\sigma_e^2)p(\sigma_e^2)$, with
$$p(\Theta|\sigma_e^2) = \mathcal{N}\left(0, \sigma_e^2\lambda^2 I_{n_\Theta}\right), \tag{13a}$$
$$p(\sigma_e^2) = \Gamma^{-1}(\alpha_0, \beta_0). \tag{13b}$$
In (13b) $\Gamma^{-1}(\alpha_0, \beta_0)$ denotes the *Inverse-Gamma* distribution with parameters α_0 and β_0, having probability density function:
$$p\left(\sigma_e^2; \alpha_0, \beta_0\right) = \frac{\beta_0^{\alpha_0}}{\Gamma(\alpha_0)}(\sigma_e^2)^{-\alpha_0-1}e^{-\frac{\beta_0}{\sigma_e^2}}. \tag{13c}$$
A Gaussian Inverse-Gamma prior is assumed for $p(\Theta, \sigma_e^2)$, as it represents the conjugate prior of a Gaussian likelihood with unknown mean and variance (Bishop, 2006, Ch. 2). This choice allows us to obtain an analytical expression for the conditional posterior of Θ and σ_v^{-2} given the mode sequence $\mathcal{S}^T$.
3. The joint probability distribution of $\mathcal{S}^T$ and M factorizes as
$$p(\mathcal{S}^T, M) = p(\mathcal{S}^T|M)p(M), \tag{14}$$
where the components of the ith row $M_{i,:}$ follow a Dirichlet distribution with parameters $\alpha_1, \ldots, \alpha_K$ and probability density function
$$p(M_{i,1}, \ldots, M_{i,K}) = \frac{\Gamma(\alpha_1 + \cdots + \alpha_K)}{\Gamma(\alpha_1)\cdots\Gamma(\alpha_K)}\prod_{j=1}^{K} M_{i,j}^{\alpha_j-1}, \tag{15}$$
where $p(M_{i,1}, \ldots, M_{i,K})$ is defined over the simplex
$$M_{i,j} \geq 0, \quad \sum_{j=1}^{K} M_{i,j} = 1, \quad i = 1, \ldots, K. \tag{16}$$
Furthermore, the rows of the transitions matrix M are assumed to be statistically independent with each others, *i.e.*, $p(M) = \prod_{i=1}^{K} p(M_{i,:})$. Thus,
$$p(M) = \left(\frac{\Gamma(\alpha_1 + \cdots + \alpha_K)}{\Gamma(\alpha_1)\cdots\Gamma(\alpha_K)}\right)^K \prod_{i,j=1}^{K} M_{i,j}^{\alpha_j-1}. \tag{17}$$
Based on the modelling assumptions discussed in Section 2.2, the probability distribution $p(\mathcal{S}^T|M)$ is equal to
$$p(\mathcal{S}^T|M) = p(s_0)\prod_{t=1}^{T} p(s_t|s_{t-1}) = p(s_0)\prod_{t=1}^{T} M_{s_{t-1},s_t} \tag{18a}$$
$$= p(s_0)\prod_{i,j}^{K}\prod_{t=1}^{T} M_{i,j}^{\mathbb{I}(s_{t-1}=i\ \&\ s_t=j)} \tag{18b}$$
$$= p(s_0)\prod_{i,j}^{K} M_{i,j}^{\#(s_{t-1}=i\ \&\ s_t=j)}, \tag{18c}$$
where # counts the number of times the joint event $s_{t-1} = i$ and $s_t = j$ occurs in the sequence $\mathcal{S}^T$, *i.e.*,
$$\#(s_{t-1} = i\ \&\ s_t = j) = \sum_{t=1}^{T} \mathbb{I}(s_{t-1} = i\ \&\ s_t = j),$$
and $p(s_0)$ is the probability of the initial state s_0. In order to keep the derivations in the paper simple, we assume that $p(s_0)$ is known and uniform, *i.e.*,
$$p(s_0) = \frac{1}{K}, \quad \text{for all} \quad s_0 = 1, \ldots, K. \tag{18d}$$
The extension to the case of unknown $p(s_0)$ is straightforward.
By combining (17) and (18), the joint probability distribution $p(S, M)$ is given by:
$$p(S, M) = \frac{1}{K}\left(\frac{\Gamma(\alpha_1 + \cdots + \alpha_K)}{\Gamma(\alpha_1)\cdots\Gamma(\alpha_K)}\right)^K \times \prod_{i,j}^{K} M_{i,j}^{\#(s_{t-1}=i\ \&\ s_t=j)+\alpha_j-1}. \tag{19}$$

3.2. Likelihood

We now analyse the likelihood of the model parameters $\Theta, \sigma_e^2, M, \mathcal{S}^T$ given the input $\mathcal{U}^T$ and output dataset $\mathcal{Y}^T$.

Following the same ideas used in prediction-error-method for identification of LTI systems (Ljung, 1999), the output observation y_t is factorized into a predictor $\hat{y}_{t|t-1}$ and a prediction error ε_t as follows
$$y_t = \hat{y}_{t|t-1} + \varepsilon_t, \tag{20}$$
where the output predictor $\hat{y}_{t|t-1}$ only depends on past input and output observations $\mathcal{U}^{t-1} = \{u_h\}_{h=1}^{t-1}$ and $\mathcal{Y}^{t-1} = \{y_h\}_{h=1}^{t-1}$, and on the mode sequence up to time t (*i.e.*, $\mathcal{S}^t$).

In order to guarantee that the predictor $\hat{y}_{t|t-1}$ and the prediction error ε_t are conditionally independent given past observations $\mathcal{U}^{t-1}, \mathcal{Y}^{t-1}$ and current discrete mode s_t, the prediction error is chosen as

$$\begin{aligned}\varepsilon_t = e(t) =& H^{-1}(\theta^{s_t})v_t = H^{-1}(\theta^{s_t})(y_t - y_t^o) = \\ =& H^{-1}(\theta^{s_t})(y_t - G(\theta^{s_t})u_t),\end{aligned} \tag{21}$$

where the time-domain operator $H^{-1}(\theta^{s_t})$ has the following meaning in terms of difference equations:

$$\varepsilon_t = H^{-1}(\theta^{s_t})v_t \to C(\theta^{s_t})\varepsilon_t = D(\theta^{s_t})v_t. \tag{22}$$

Thus, the predictor $\hat{y}_{t|t-1}$ can be reconstructed from (20) and (21), and it is given by

$$\begin{aligned}\hat{y}_{t|t-1} =& y_t - \varepsilon_t = y_t - H^{-1}(q^{-1}, \theta^{s_t})(y_t - y_t^o) = \\ =& y_t - H^{-1}(q^{-1}, \theta^{s_t})(y_t - G(q^{-1}, \theta^{s_t})u_t)\end{aligned} \tag{23}$$

Note that, since $H^{-1}(0, \theta^{s_t}) = 1$ and $G(0, \theta^{s_t}) = 0$, then $\hat{y}_{t|t-1}$ only depends on past input and output data, and thus $\hat{y}_{t|t-1}$ is independent of e_t. Although not explicitly indicated in the definition of the predictor (23), as already mentioned, we stress that $\hat{y}_{t|t-1}$ depends on the whole mode sequence up to time t because of the recursive definition in (23).

Thus,

$$p(y_t|\Theta, \sigma_e^2, \mathcal{S}^t, \mathcal{Y}^{t-1}, \mathcal{U}^{t-1}) = \mathcal{N}\left(y_t; \hat{y}_{t|t-1}, \sigma_e^2\right). \tag{24}$$

Remark 3. Let us consider the switching ARX model introduced in Remark 1. Using the definition of the prediction error and predictor in (21) and (23), or simply model equation (7c), the following well known expression for the prediction error and predictor are obtained:

$$\varepsilon_t = e(t) = A(q^{-1}, \theta^{s_t})y_t - B(q^{-1}, \theta^{s_t})u_t, \tag{25a}$$

$$\hat{y}_{t|t-1} = \left(1 - A(q^{-1}, \theta^{s_t})\right) y_t - B(q^{-1}, \theta^{s_t})u_t. \quad \blacksquare \tag{25b}$$

Let us factorize the likelihood $p(\mathcal{Y}^T|\Theta, \sigma_e^2, M, \mathcal{S}^T, \mathcal{U}^T)$ as

$$p(\mathcal{Y}^T|\Theta, \sigma_e^2, M, \mathcal{S}^T, \mathcal{U}^T) \tag{26a}$$

$$= \prod_{t=1}^{T} p(y_t|\Theta, \sigma_e^2, M, \mathcal{S}^T, \mathcal{Y}^{t-1}, \mathcal{U}^T) = \tag{26b}$$

$$= \prod_{t=1}^{T} p(y_t|\Theta, \sigma_e^2, \mathcal{S}^t, \mathcal{Y}^{t-1}, \mathcal{U}^{t-1}) = \tag{26c}$$

$$= \prod_{t=1}^{T} \mathcal{N}\left(y_t; \hat{y}_{t|t-1}, \sigma_e^2\right), \tag{26d}$$

where the last equality comes from (24) and (26c) holds because of system causality and conditional independence of y_t of the future mode sequence, given $\mathcal{S}^t$ and past input/output samples.

Thus

$$p(\mathcal{Y}^T|\Theta, \sigma_e^2, M, \mathcal{S}^T, \mathcal{U}^T) = \tag{27a}$$

$$= \frac{1}{(2\pi\sigma_e^2)^{T/2}} e^{-\frac{1}{2\sigma_e^2}\sum_{t=1}^{T}(y_t - \hat{y}_{t|t-1})^2} = \tag{27b}$$

$$= \frac{1}{(2\pi\sigma_e^2)^{T/2}} e^{-\frac{1}{2\sigma_e^2}\sum_{t=1}^{T}\left(H^{-1}(\theta^{s_t})(y_t - G(\theta^{s_t})u_t)\right)^2}. \tag{27c}$$

3.3. Posterior distribution

Using Bayes' rule, the posterior distribution of the model parameters is factorized, up to the proportionality normalization constant $\frac{1}{p(\mathcal{Y}^T|\mathcal{U}^T)}$, as

$$\begin{aligned}& p(\Theta, \sigma_e^2, M, \mathcal{S}^T|\mathcal{Y}^T, \mathcal{U}^T) \propto \\ & \propto p(\mathcal{Y}^T|\Theta, \sigma_e^2, M, \mathcal{S}^T, \mathcal{U}^T)p(\Theta, \sigma_e^2, M, \mathcal{S}^T).\end{aligned} \tag{28}$$

Algorithm 1 Maximum-a-posteriori estimation of Jump Box–Jenkins models.

Input: Training set $(\mathcal{U}^T, \mathcal{Y}^T)$; initial guess on the mode sequence $\mathcal{S}^{T(0)} = (s_0^{(0)}, \ldots, s_T^{(0)})$; maximum number $k_{\max}$ of iterations.

```
1. **iterate for** $k = 1, \ldots$
    1.1 Compute $\Theta^{(k)}, \sigma_e^{2(k)}, M^{(k)}$ as the solution of
        $\operatorname{argmax}_{\Theta,\sigma_e^2,M} p(\Theta, \sigma_e^2, M, \mathcal{S}^{T(k-1)}|\mathcal{Y}^T, \mathcal{U}^T)$;
    1.2 Compute $\mathcal{S}^{T(k)}$ as the solution of
        $\operatorname{argmax}_{\mathcal{S}^T} p(\Theta^{(k)}, \sigma_e^{2(k)}, M^{(k)}, \mathcal{S}^T|\mathcal{Y}^T, \mathcal{U}^T)$;
2. **until** $k = k_{\max}$ or $\mathcal{S}^{T(k)} = \mathcal{S}^{T(k-1)}$
```

Output: Estimated parameters $\Theta^\star = \Theta^{(k)}$, $\sigma_e^{2\star} = \sigma_e^{2(k)}$, $M^\star = M^{(k)}$ and mode sequence $\mathcal{S}^{T\star} = \mathcal{S}^{T(k)}$.

Thus, from the likelihood (27) and the priors assumed in Section 3.1, the posterior (28) becomes

$$p(\Theta, \sigma_e^2, M, \mathcal{S}^T|\mathcal{Y}^T, \mathcal{U}^T) \propto \tag{29a}$$

$$\propto e^{-\frac{1}{2\sigma_e^2}\sum_{t=1}^{T}\left(H^{-1}(\theta^{s_t})(y_t - G(\theta^{s_t})u_t)\right)^2} \tag{29b}$$

$$\times \frac{1}{(\sigma_e^2)^{T/2+n_\Theta/2+\alpha_0+1}} e^{-\frac{1}{2\lambda^2\sigma_e^2}\Theta'\Theta} e^{-\frac{\beta_0}{\sigma_e^2}} \tag{29c}$$

$$\times \prod_{i,j}^{K} M_{i,j}^{\#(s_{t-1}=i\ \&\ s_t=j)+\alpha_j-1}, \tag{29d}$$

where all the proportionality terms independent of the variables $\Theta, \sigma_e^2, M, \mathcal{S}^T$ do not appear in Eq. (29) as they do not affect the maximization of the posterior distribution $p(\Theta, \sigma_e^2, M, \mathcal{S}^T|\mathcal{Y}^T, \mathcal{U}^T)$.

The following section addresses the computation of the *maximum-a-posteriori* (MAP) estimate of the unknown variables $\Theta, \sigma_e^2, M, \mathcal{S}^T$.

4. Optimization algorithm

The posterior distribution $p(\Theta, \sigma_e^2, M, \mathcal{S}^T|\mathcal{Y}^T, \mathcal{U}^T)$ is maximized with respect to the model parameters Θ, σ_e^2, M and the mode sequence $\mathcal{S}^T$ through the iterative coordinate ascent approach outlined in Algorithm 1. At each iteration k, Algorithm 1 alternates between:

- Step 1.1: maximization over Θ, σ_e^2, M, for a fixed mode sequence $\mathcal{S}^{T(k-1)}$ computed at the previous iteration $k-1$;
- Step 1.2: maximization w.r.t. $\mathcal{S}^T$, for fixed parameters $\Theta^{(k)}$, $\sigma_e^{2(k)}, M^{(k)}$ computed at Step 1.1.

The final solution of Algorithm 1 depends on the initial guess $\mathcal{S}^{T(0)}$, and there is no guarantee that the computed solution is the global maximizer of the posterior $p(\Theta, \sigma_e^2, M, \mathcal{S}^T|\mathcal{Y}^T, \mathcal{U}^T)$. To improve the quality of the solution, Algorithm 1 can be run N times, starting from different initial sequences $\mathcal{S}^{T(0)}$ and then selecting the best result.

It is worth remarking that since Algorithm 1 implements a coordinate-ascent approach, the posterior distribution $p(\Theta, \sigma_e^2, M, \mathcal{S}^T|\mathcal{Y}^T, \mathcal{U}^T)$ is not decreasing at every iteration. This condition holds when Steps 1.1 and 1.2 are solved at the global optimum or simply by choosing at the $k+1$-th iteration $\Theta^{(k)}$, $\sigma_e^{2(k)}$, $M^{(k)}$ and $\mathcal{S}^{T(k)}$ as initial conditions for Θ, σ_e^2, M and $\mathcal{S}^T$, respectively.

The use of coordinate-descent optimization for fitting jump models has been recently proposed by the authors in Bemporad et al. (2018). However, Bemporad et al. (2018) do not address the estimation of jump models in a probabilistic framework, and the fitting problem is formulated as a suitable-constructed cost function, which is minimized only w.r.t. the parameters Θ and the mode sequence $\mathcal{S}^T$. Furthermore, the method in Bemporad et al. (2018) assumes a structure where the output observations are conditionally independent given the current mode and past signal measurements. Thus, jump Box–Jenkins models cannot be handled with the approach in Bemporad et al. (2018). Therefore, although coordinate-ascent (or equivalently descent) optimization is the high-level algorithm both in this paper and Bemporad et al. (2018), the single steps in coordinate-ascent optimization are completely different from this paper and Bemporad et al. (2018).

In the following sections we show how to solve the maximization problems in Steps 1.1 and 1.2.

4.1. Optimizing parameters Θ, σ^2, M

We first note that, for a fixed mode sequence $\mathcal{S}^{T(k-1)}$, the optimization of the posterior $p(\Theta, \sigma_e^2, M, \mathcal{S}^{T(k-1)}|\mathcal{Y}^T, \mathcal{U}^T)$ (Eq. (29)) w.r.t. Θ, σ_e^2 and w.r.t. M can be performed independently.

4.1.1. Optimizing Θ and σ_e^2

The optimization over the parameters Θ is performed numerically, via a Gauss–Newton approach, as described in the following.

First, note that maximizing the posterior (29) over Θ for a fixed mode sequence $\mathcal{S}^{T(k-1)}$ is equivalent to solve the minimization problem

$$\min_{\Theta} J(\Theta) \triangleq \sum_{t=1}^{T} \varepsilon_t(\Theta, \mathcal{S}^{T(k-1)})^2 + \frac{1}{\lambda^2}\Theta'\Theta, \tag{30a}$$

where the time evolution of the prediction error ε_t is described in (21), and equivalently rewritten as

$$C(\theta^{s_t^{(k-1)}})\varepsilon_t(\Theta, \mathcal{S}^{T(k-1)}) = D(\theta^{s_t^{(k-1)}})\left(y_t - y_t^{\mathrm{o}}\right) \tag{31a}$$

with

$$y_t^{\mathrm{o}} : A(\theta^{s_t^{(k-1)}})y_t^{\mathrm{o}} = B(\theta^{s_t^{(k-1)}})u_t. \tag{31b}$$

To alleviate the notation, the dependence of the mode sequence $\mathcal{S}^{T(k-1)}$ and of the single mode $s_t^{(k-1)}$ on the iteration $k-1$ will be dropped in the rest of the paragraph.

Note that, as typical in Bayesian inference, the term $\frac{1}{\lambda^2}\Theta'\Theta$ in (30) acts as a quadratic regularization and it is due to the prior distribution on Θ (Eq. (13a)).

Problem (30) can be solved by a Gauss–Newton algorithm by properly extending numerical algorithms for PEM estimation to linear models with time-varying coefficient Specifically, at each Gauss–Newton iteration h, Θ is updated from the value $\Theta^{(h-1)}$ computed at the previous iteration as follows:

$$\Theta^{(h)} = \Theta^{(h-1)} - \alpha\left(\nabla^2_{\Theta}J(\Theta^{(h-1)})\right)^{-1}\nabla_{\Theta}J(\Theta^{(h-1)}), \tag{32}$$

where $\alpha > 0$ is a scaling factor which can be computed through *line search* (Boyd & Vandenberghe, 2004, Ch. 9), $\nabla_{\Theta}J(\Theta)$ is the gradient of the cost J, *i.e.*,

$$\nabla_{\Theta}J(\Theta) = 2\sum_{t=1}^{T}\varepsilon_t(\Theta, \mathcal{S}^T)\frac{\partial\varepsilon_t(\Theta, \mathcal{S}^T)}{\partial\Theta} + 2\lambda^{-2}\Theta, \tag{33a}$$

and $\nabla^2_{\Theta}J(\Theta)$ is the Hessian of J. As discussed in Ljung (1999, Ch. 10.2), the Hessian $\nabla^2_{\Theta}J(\Theta)$ is approximated as follows, based on the Levenberg–Marquardt modification of the Gauss–Newton algorithm:

$$\nabla^2_{\Theta}J(\Theta) \approx 2\sum_{t=1}^{T}\frac{\partial\varepsilon_t(\Theta, \mathcal{S}^T)}{\partial\Theta}\left(\frac{\partial\varepsilon_t(\Theta, \mathcal{S}^T)}{\partial\Theta}\right)' + 2\lambda^{-2}I, \tag{33b}$$

with I being the identity matrix of proper dimension.

For fixed $\Theta = \Theta^{(h-1)}$ and $\mathcal{S}^T = \mathcal{S}^{T(k-1)}$, the prediction error $\varepsilon_t(\Theta, \mathcal{S}^T)$ (required to calculate the gradient $\nabla_{\Theta}J(\Theta)$ in (33a)) can be computed recursively through the difference equations in (31b).

As for the computation of the gradient $\frac{\partial\varepsilon_t(\Theta,\mathcal{S}^T)}{\partial\Theta}$ of the prediction error ε_t at time t, proper modifications and extensions of the numerical algorithms commonly used in PEM for LTI systems are needed to take into account the hybrid nature of jump BJ models, characterized by time-varying linear dynamical filters $G(q^{-1}, \theta^{s_t})$ and $H(q^{-1}, \theta^{s_t})$. By taking the partial derivatives on the left and right side of (31a) w.r.t. each element of the parameter vector Θ, we can derive recursive formulas for $\frac{\partial\varepsilon_t(\Theta)}{\partial c_j^i}$ (with $j = 1, \ldots, n_c$ and $i = 1, \ldots, K$) and $\frac{\partial\varepsilon_t(\Theta)}{\partial d_j^i}$ (with $j = 1, \ldots, n_d$ and $i = 1, \ldots, K$):

$$C(\theta^{s_t})\frac{\partial\varepsilon_t(\Theta)}{\partial c_j^i} = -\varepsilon_{t-j}(\Theta)\mathbb{I}(s_t = i),$$

$$C(\theta^{s_t})\frac{\partial\varepsilon_t(\Theta)}{\partial d_j^i} = \left(y_{t-j} - y_{t-j}^{\mathrm{o}}\right)\mathbb{I}(s_t = i),$$

with y_t^{o} simulated (for fixed Θ) using (31b).

As for the gradient $\frac{\partial\varepsilon_t(\Theta)}{\partial a_j^i}$ (with $j = 1, \ldots, n_a$ and $i = 1, \ldots, K$) and $\frac{\partial\varepsilon_t(\Theta)}{\partial b_j^i}$ (with $j = 1, \ldots, n_b$ and $i = 1, \ldots, K$), we have

$$C(\theta^{s_t})\frac{\partial\varepsilon_t(\Theta)}{\partial a_j^i} = -D(\theta^{s_t})\frac{\partial y_t^{\mathrm{o}}}{\partial a_j^i},$$

$$C(\theta^{s_t})\frac{\partial\varepsilon_t(\Theta)}{\partial b_j^i} = -D(\theta^{s_t})\frac{\partial y_t^{\mathrm{o}}}{\partial b_j^i},$$

where $\frac{\partial y_t^{\mathrm{o}}}{\partial a_j^i}$ and $\frac{\partial y_t^{\mathrm{o}}}{\partial b_j^i}$ can be computed taking the partial derivatives of the left and right side of (31b), thus obtaining the recursive equations:

$$A(\theta^{s_t})\frac{\partial y_t^{\mathrm{o}}}{\partial a_j^i} = -y_{t-j}^{\mathrm{o}}\mathbb{I}(s_t = i), \tag{34a}$$

$$A(\theta^{s_t})\frac{\partial y_t^{\mathrm{o}}}{\partial b_j^i} = u_{t-j}\mathbb{I}(s_t = i). \tag{34b}$$

Remark 4. Consider again the switching ARX model discussed in Remarks 1 and 3. In this case, the objective function $J(\Theta)$ in (30a) is quadratic in the unknown variables Θ. Specifically, using the expression of the prediction error in (21), the cost $J(\Theta)$ is given by

$$J(\Theta) = \sum_{t=1}^{T}\left(A(q^{-1}, \theta^{s_t})y_t - B(q^{-1}, \theta^{s_t})u_t\right)^2 + \frac{1}{\lambda^2}\Theta'\Theta, \tag{35}$$

and its (global) minimum can be computed analytically using standard least-squares formulas. ■

The Gauss–Newton algorithm is run until a maximum number of iterations is reached or when the following condition on the weighted norm of the gradient $\nabla_{\Theta}J(\Theta^{(h)})$ is satisfied:

$$(\nabla_{\Theta}J(\Theta^{(h)}))'(\nabla^2_{\Theta}J(\Theta^{(h)}))^{-1}\nabla_{\Theta}J(\Theta^{(h)}) \le \epsilon_J. \tag{36}$$

Once the Gauss–Newton algorithm is terminated and the parameters $\Theta^{(k)}$ maximizing the posterior distribution $p(\Theta, \sigma_e^2, M, \mathcal{S}^{T(k-1)}|\mathcal{Y}^T, \mathcal{U}^T)$ (for a fixed mode sequence $\mathcal{S}^{T(k-1)}$) are obtained, the optimal parameter $\sigma_e^{2(k)}$ maximizing $p(\Theta, \sigma_e^2, M, \mathcal{S}^{T(k-1)}|\mathcal{Y}^T, \mathcal{U}^T)$ in (29) can be computed analytically and it is given by:

$$\sigma_e^{2(k)} = \frac{\beta_0 + \frac{1}{2}\lambda^{-2}\Theta^{(k)'}\Theta^{(k)} + \frac{1}{2}\sum_{t=1}^T \left(\varepsilon_t(\Theta^{(k)}, \mathcal{S}^{T(k-1)})\right)^2}{\frac{T+n_\Theta}{2} + \alpha_0 + 1}.$$

4.1.2. Optimizing the transition matrix M

The optimization of the posterior distribution $p(\Theta, \sigma_e^2, M, \mathcal{S}^{T(k-1)}|\mathcal{Y}^T, \mathcal{U}^T)$ in (29) with respect to the entries of the transition matrix M can be performed separately for each row $M_{i,:}$, using the method of *Lagrange multipliers* to take into account the equality constraint $\sum_{j=1}^K M_{i,j} = 1$. Specifically, after taking the *log* of (29), the *Lagrangian* is given by

$$\mathcal{L}(M_{i,:}, \gamma) = \gamma \left(1 - \sum_{j=1}^K M_{i,j}\right) + \sum_{j=1}^K (\#(s_{t-1} = i \;\&\; s_t = j) + \alpha_j - 1)\log(M_{i,j}), \tag{37}$$

where γ is the Lagrange multiplier. Taking and zeroing the partial derivatives of $\mathcal{L}$ w.r.t. $M_{i,j}$ and γ we obtain

$$\frac{\partial L}{\partial \gamma} = 0 \rightarrow \sum_{j=1}^K M_{i,j} = 1, \tag{38a}$$

$$\frac{\partial L}{\partial M_{i,j}} = 0 \rightarrow M_{i,j} = \frac{\#(s_{t-1} = i \;\&\; s_t = j) + \alpha_j - 1}{\gamma}. \tag{38b}$$

By substituting (38b) into (38a) we obtain

$$\gamma = \sum_{j=1}^K \#(s_{t-1} = i \;\&\; s_t = j) + \alpha_j - 1. \tag{39}$$

Finally, by substituting (39) into (38b), the following optimal values for the transition probabilities $M_{i,j}$ are obtained:

$$\begin{aligned} M_{i,j} =& \frac{\#(s_{t-1} = i \;\&\; s_t = j) + \alpha_j - 1}{\sum_{j=1}^K \#(s_{t-1} = i \;\&\; s_t = j) + \alpha_j - 1} = \\ =& \frac{\#(s_{t-1} = i \;\&\; s_t = j) + \alpha_j - 1}{\sum_{j=1}^K \#(s_{t-1} = i) + \sum_{j=1}^K(\alpha_j - 1)}. \end{aligned} \tag{40}$$

Note that (40) has a very intuitive interpretation. Indeed, the MAP estimate of $M_{i,j}$ is equal to the sampling frequency (up to the additive term $\alpha_j - 1$ due the prior on $M_{i,j}$) given by the number of times a switch from mode i to mode j occurs in the sequence $\mathcal{S}^T$, divided by the number of times mode i is active.

4.2. Optimization of the mode sequence $\mathcal{S}^T$

In order to optimize the posterior distribution $p(\Theta^{(k)}, \sigma_e^{2(k)}, M^{(k)}, \mathcal{S}^T|\mathcal{Y}^T, \mathcal{U}^T)$ over the mode sequence $\mathcal{S}^T$, the log of the posterior is taken and only the terms depending on $\mathcal{S}^T$ are considered, *i.e.*

$$\log\left(p(\Theta, \sigma_e^2, M, \mathcal{S}^T|\mathcal{Y}^T, \mathcal{U}^T)\right) \propto \tag{41a}$$

$$\propto -\frac{1}{2\sigma_e^2}\sum_{t=1}^T \left(H^{-1}(\theta^{s_t})\left(y_t - G(\theta^{s_t})u_t\right)\right)^2 \tag{41b}$$

$$+ \sum_{t=1}^T \sum_{i,j}^K \mathbb{I}(s_{t-1} = i \;\&\; s_t = j)\log M_{i,j}, \tag{41c}$$

where the dependence of Θ, σ_e^2, M on the iteration k is dropped to alleviate the notation. Using the definition of the prediction error ε_t in (21), Eq. (41) is equivalent to:

$$\log\left(p(\Theta, \sigma_e^2, M, \mathcal{S}^T|\mathcal{Y}^T, \mathcal{U}^T)\right) \propto \tag{42a}$$

$$\propto \sum_{t=1}^T \underbrace{-\frac{1}{2\sigma_e^2}\varepsilon_t\left(\Theta, \mathcal{S}^t\right)^2}_{\mathcal{L}_t(\mathcal{S}^t)} \tag{42b}$$

$$+ \sum_{t=1}^T \underbrace{\sum_{i,j}^K \mathbb{I}(s_{t-1} = i \;\&\; s_t = j)\log M_{i,j}}_{\mathcal{L}^{\text{trans}}(s_{t-1}, s_t)}. \tag{42c}$$

Note that, in the general Box–Jenkins modelling framework considered in the paper, the prediction error $\varepsilon_t(\Theta, \mathcal{S}^t)$ depends on the whole sequence $\mathcal{S}^t$ of discrete states up to time t, according to the recursive definition (22). Thus, maximizing the log of the posterior distribution (42) with respect to the mode sequence $\mathcal{S}^T$ requires to try all possible instances of $\mathcal{S}^T$. This leads to a combinatorial problem with K^{T+1} possible feasible solutions. To reduce the complexity of this combinatorial problem, a sub-optimal solution is computed. The main idea, discussed in the following, is to simulate the prediction error $\varepsilon_t(\Theta, \mathcal{S}^t)$ by fixing part of the sequence $\mathcal{S}^t$ and consequently trying a smaller number of instances of $\mathcal{S}^t$.

According to this idea, a sub-optimal *moving-horizon* approach is proposed to maximize the objective function (42), which is rewritten in the compact form

$$Q_T\left(\mathcal{S}^T\right) = \sum_{t=1}^T \mathcal{L}_t(\mathcal{S}^t) + \sum_{t=1}^T \mathcal{L}^{\text{trans}}(s_{t-1}, s_t). \tag{43}$$

At each time step t, we consider a moving-horizon window of length T_c containing all possible subsequences $\{s_{t+1}, s_{t+2}, \ldots, s_{t+T_c}\} = \mathcal{S}_{t+1}^{t+T_c}$ from time $t+1$ to time $t+T_c$. Then, for all possible sequences $\mathcal{S}_{t+1}^{t+T_c}$, we compute the discrete mode s_t maximizing the objective in (43) truncated to time $t + T_c$, and defined as

$$Q_{t+T_c}\left(\mathcal{S}^{t+T_c}\right) = \sum_{h=1}^{t+T_c} \mathcal{L}_h(\mathcal{S}^h) + \sum_{h=1}^{t+T_c} \mathcal{L}^{\text{trans}}(s_{h-1}, s_h). \tag{44}$$

For each possible instance of $\mathcal{S}^{t+T_c}$, the past discrete states up to time $t-1$ (*i.e.*, $\mathcal{S}^{t-1}$) in (44) are fixed to the previously optimized values.

Here below we describe in detail the approach used to maximize the objective function (43) w.r.t. $\mathcal{S}^T$. We remind that the sequence $\mathcal{S}^T$ includes the initial discrete state s_0.

(*i*) First, for all possible T_c-length sequences in $\mathcal{S}_1^{T_c}$, the optimal initial mode

$$\tilde{s}_0(\mathcal{S}_1^{T_c}) = \arg\max_{s_0} Q_{T_c}\left(\mathcal{S}_0^{T_c}\right) \tag{45}$$

is computed. Note that an optimal initial mode $\tilde{s}_0(\mathcal{S}_1^{T_c})$ is associated to each sequence $\mathcal{S}_1^{T_c}$;

(*ii*) A step forward is made and all possible sequences $\mathcal{S}_2^{T_c+1}$ are considered. The optimal mode at time $t = 1$ is then computed as

$$\begin{aligned} \tilde{s}_1(\mathcal{S}_2^{T_c+1}) = \quad & \arg\max_{s_1} Q_{T_c+1}(\mathcal{S}_0^{T_c+1}) \\ & \text{s.t. } s_0 = \tilde{s}_0(\mathcal{S}_1^{T_c}). \end{aligned} \tag{46}$$

As in the previous step, an optimal mode $\tilde{s}_1$ is associated to all possible sequences $\mathcal{S}_2^{T_c+1}$. Note also that, for each sequence $\mathcal{S}_1^{T_c}$, the initial mode s_0 is fixed in (46) to the optimal value $\tilde{s}_0(\mathcal{S}_1^{T_c})$ previously computed in (45);

(*iii*) The procedure is repeated up to time $t = T - T_c - 1$. Specifically, for a generic time step $t \le T - T_c - 1$, the optimal mode $\tilde{s}_t$ associated to each possible sequence $\mathcal{S}_{t+1}^{T_c+t}$ is defined recursively

$$\begin{aligned}\tilde{s}_t(\mathcal{S}_{t+1}^{t+T_c}) = \quad & \arg\max_{s_t} Q_{t+T_c}(\mathcal{S}_0^{t+T_c}) \\ & \text{s.t. } s_k = \tilde{s}_k(\mathcal{S}_{k+1}^{k+T_c}), \\ & \quad k = 0, \ldots, t-1.\end{aligned} \tag{47}$$

(*iv*) At time $t = T - T_c$, the optimal sub-sequence is computed $(\mathcal{S}_{T-T_c}^T)^\star$ by solving

$$\begin{aligned}(\mathcal{S}_{T-T_c}^T)^\star = \quad & \arg\max_{\mathcal{S}_{T-T_c}^T} Q_T(\mathcal{S}_0^T) \\ & \text{s.t. } s_k = \tilde{s}_k(\mathcal{S}_{k+1}^{k+T_c}), \\ & \quad k = 0, \ldots, T-T_c-1.\end{aligned} \tag{48}$$

The remaining modes $s_{T-T_c-1}^\star, \ldots, s_0^\star$ are obtained via backward recursion from (47), by setting $s_t^\star = \tilde{s}_t((\mathcal{S}_{t+1}^{t+T_c})^\star)$ for $t = T - T_c - 1, \ldots, 0$.

Note that, at each time sample $t = 1, \ldots, T - T_c - 1$, the cost Q_{T_c+t} is computed K^{T_c+1} times, namely for all possible sequences $\mathcal{S}_t^{t+T_c}$. Then, the maximum over s_t is taken according to (47). Thus, the complexity of the proposed optimization strategy is $O\left((T - T_c)K^{T_c+1}\right)$. The length of horizon window T_c acts as a knob to tradeoff between complexity and suboptimality in the optimization of the posterior distribution w.r.t. the whole mode sequence $\mathcal{S}^T$.

Remark 5. In the specific and simple case of ARX models, the objective function Q_T in (43) to be maximized is given by

$$Q_T\left(\mathcal{S}^T\right) = \sum_{t=1}^{T}\left(\mathcal{L}_t(s_t) + \mathcal{L}^{\text{trans}}(s_{t-1}, s_t)\right), \tag{49}$$

with

$$\begin{aligned}\mathcal{L}_t(s_t) &= -\frac{1}{2\sigma_e^2}\varepsilon_t\left(\Theta, s_t\right)^2 = \\ &= -\frac{1}{2\sigma_e^2}\left(A(q^{-1}, \theta^{s_t})y_t - B(q^{-1}, \theta^{s_t})u_t\right)^2.\end{aligned} \tag{50}$$

Since the prediction error ε_t (and thus the term $\mathcal{L}_t(s_t)$) does not depend on the whole discrete-state sequence $\mathcal{S}^t$, but only on the current mode s_t, the (global) maximum of the objective Q_T w.r.t. the mode sequence $\mathcal{S}^T$ can be computed with polynomial complexity by the Viterbi algorithm for discrete dynamic programming (Viterbi, 1967), which coincides with the proposed moving-horizon approach for $T_c = 1$. ■

5. Inference

Assume that optimal parameters $\Theta^\star$, $M^\star$ and $\sigma_e^{2\star}$ have been estimated by running Algorithm 1 on a training dataset and consider a new dataset of inputs $\tilde{\mathcal{U}}^{t-1} = \{\tilde{u}_\tau\}_{\tau=1}^{t-1}$ and outputs (up to time $t-1$) $\tilde{\mathcal{Y}}^{t-1} = \{\tilde{y}_\tau\}_{\tau=1}^{t-1}$. We want now to use the estimated jump BJ model to infer the mode sequence $\hat{\mathcal{S}}^t = (\hat{s}_0, \hat{s}_1, \ldots, \hat{s}_t)$ and predict the output $\hat{y}_t$ at time t.

According to the probabilistic framework considered in the paper, inference is performed by maximizing the joint distribution $p(y_t, S^t|\Theta^\star, \sigma_e^{2\star}, M^\star, \tilde{\mathcal{U}}^{t-1}, \tilde{\mathcal{Y}}^{t-1})$ w.r.t. y_t and S^t. First, $p(y_t, S^t|\Theta^\star, \sigma_e^{2\star}, M^\star, \tilde{\mathcal{U}}^{t-1}, \tilde{\mathcal{Y}}^{t-1})$ is factorized (up to a proportionality constant) as

$$\begin{aligned}& p(y_t, S^t|\Theta^\star, \sigma_e^{2\star}, M^\star, \tilde{\mathcal{U}}^{t-1}, \tilde{\mathcal{Y}}^{t-1}) = \\ & \propto p(y_t|S^t, \Theta^\star, \sigma_e^{2\star}, M^\star, \tilde{\mathcal{U}}^{t-1}, \tilde{\mathcal{Y}}^{t-1}) \\ & \quad \times p(S^{t-1}|\Theta^\star, \sigma_e^{2\star}, M^\star, \tilde{\mathcal{U}}^{t-1}, \tilde{\mathcal{Y}}^{t-1}) \\ & \quad \times p(s_t|S^{t-1}, \Theta^\star, \sigma_e^{2\star}, M^\star, \tilde{\mathcal{U}}^{t-1}, \tilde{\mathcal{Y}}^{t-1}) \\ & = p(y_t|S^t, \Theta^\star, \sigma_e^{2\star}, M^\star, \tilde{\mathcal{U}}^{t-1}, \tilde{\mathcal{Y}}^{t-1}) \\ & \quad \times p(S^{t-1}|\Theta^\star, \sigma_e^{2\star}, M^\star, \tilde{\mathcal{U}}^{t-1}, \tilde{\mathcal{Y}}^{t-1})p(s_t|s_{t-1}, M^\star).\end{aligned} \tag{51}$$

By substituting (24) and (42c) into (51) and taking the log of $p(y_t, S^t|\Theta^\star, \sigma_e^{2\star}, M^\star, \tilde{\mathcal{U}}^{t-1}, \tilde{\mathcal{Y}}^{t-1})$, we obtain

$$\begin{aligned}& \tilde{Q}_t(y_t, \mathcal{S}^t) = \log\left(p(y_t, S^t|\Theta^\star, \sigma_e^{2\star}, M^\star, \tilde{\mathcal{U}}^{t-1}, \tilde{\mathcal{Y}}^{t-1})\right) \\ & \propto -\frac{1}{2\sigma_e^{2\star}}\|y_t - \hat{y}_{t|t-1}(\Theta^\star, S^t)\|_2^2 + \\ & \quad + \sum_{i,j}^{K}\mathbb{I}(s_{t-1} = i \,\&\, s_t = j)\log M_{i,j}^\star \\ & \quad + \sum_{\tau=1}^{t-1} -\frac{1}{2\sigma_e^{2\star}}\varepsilon_\tau\left(\Theta^\star, \mathcal{S}^\tau\right)^2 \\ & \quad + \sum_{\tau=1}^{t-1}\sum_{i,j}^{K}\mathbb{I}(s_{\tau-1} = i \,\&\, s_\tau = j)\log M_{i,j}^\star.\end{aligned} \tag{52}$$

At each time t, inference is thus performed by maximizing $\tilde{Q}_t(y_t, \mathcal{S}^t)$, *i.e.*,

$$\{\hat{y}_t, \hat{\mathcal{S}}^t\} = \arg\max_{y_t, \mathcal{S}^t} \tilde{Q}_t(y_t, \mathcal{S}^t). \tag{53}$$

Note that, for any given sequence $\hat{\mathcal{S}}^t$, the output $\hat{y}_t$ maximizing $\tilde{Q}_t(y_t, \mathcal{S}^t)$ is the predictor $\hat{y}_{t|t-1}(\Theta^\star, \hat{\mathcal{S}}^t)$. According to (23) and the definition of filters $H(q^{-1}, \theta^{\hat{s}_t})$ and $G(q^{-1}, \theta^{\hat{s}_t})$, the predictor $\hat{y}_{t|t-1}(\Theta^\star, \hat{\mathcal{S}}^t)$ is given by

$$\begin{aligned}\hat{y}_{t|t-1} = \hat{y}_t^o &+ \sum_{i=1}^{n_c} c_i^{\star\hat{s}_t}(\tilde{y}_{t-i} - \hat{y}_{t-i|t-1-i}) + \\ &+ \sum_{j=1}^{n_d} d_j^{\star\hat{s}_t}(\hat{y}_{t-j}^o - \tilde{y}_{t-j}),\end{aligned} \tag{54a}$$

where $c_i^{\star\hat{s}_t}$ (resp. $d_j^{\star\hat{s}_t}$) denotes the element c_i (resp. d_j) of the parameter vector Θ^* and associated to mode $\hat{s}_t$.

Note that the predictor $\hat{y}_{t|t-1}$ in (54a) is independent of the (unknown) output at time t, while it depends on the simulated noiseless output, which is recursively computed as

$$\hat{y}_t^o = -\sum_{i=1}^{n_a} a_i^{\star\hat{s}_t}\hat{y}_{t-i}^o + \sum_{j=1}^{n_b} b_j^{\star\hat{s}_t}\tilde{u}_{t-j}. \tag{54b}$$

As for the sequence $\hat{S}^t$ solving (53), a sub-optimal solution can be computed using the same moving-horizon method presented in Section 4.2.

5.1. Recursive inference

Instead of solving problem (53) every time a new input–output pair is available, inference on $\hat{y}_t$ and $\hat{S}^t$ can be performed recursively. Indeed, the predictor $\hat{y}_{t|t-1}$ in (54) depends on past computed predictors $\hat{y}_{t-i|t-1-i}$. As for the sequence S^t, by simply noticing that the cost $\tilde{Q}_t(y_t, \mathcal{S}^t)$ in (52) can be written in the recursive form

$$\begin{aligned}\tilde{Q}_t(y_t, \mathcal{S}^t) \propto \tilde{Q}_{t-1}&(y_{t-1}, \mathcal{S}^{t-1}) \\ &- \frac{1}{2\sigma_e^{2\star}}\|y_t - \hat{y}_{t|t-1}(\Theta^\star, S^t)\|_2^2 + \\ &+ \sum_{i,j}^{K}\mathbb{I}(s_{t-1} = i \,\&\, s_t = j)\log M_{i,j}^\star,\end{aligned}$$

the moving-horizon algorithm discussed in Section 4.2 can be used to infer a (sub-optimal) mode sequence $\hat{S}^t$ by exploiting

Table 1
True parameters vs estimated parameters (jBJ=jump Box–Jenkins, jARX=jump ARX model).

	$i=1$			$i=2$			$i=3$			$i=4$		
	True	jBJ	jARX	True	jBJ	jARX	True	jBJ	jARX	True	jBJ	jARX
a^i_1	0.60	0.59	0.09	−0.50	−0.50	−0.06	0.10	0.08	0.07	−0.70	−0.68	−0.59
a^i_2	0.10	0.09	0.01	−0.10	−0.10	−0.43	−0.10	−0.14	−0.34	0.40	0.37	0.41
a^i_3	0.30	0.30	0.50	−0.10	−0.13	−0.15	0.10	0.09	−0.43	0.20	0.22	0.17
b^i_1	0.80	0.79	0.82	−1.00	−1.00	−1.01	−0.20	−0.18	−0.18	4.00	4.00	4.00
b^i_2	−0.80	−0.81	−1.16	1.00	0.97	0.54	−0.20	−0.16	−0.17	1.00	1.09	1.41
b^i_3	0.80	0.82	1.41	0.10	0.15	0.67	0.60	0.62	0.66	−0.10	−0.09	0.36
c^i_1	0.99	1.09	–	−0.20	−0.12	–	0.20	0.19	–	0.20	−0.01	–
c^i_2	0.80	0.87	–	0.20	0.14	–	−0.10	−0.12	–	−0.30	−0.34	–
c^i_3	0.10	0.16	–	−0.20	−0.18	–	0.5	0.49	–	0.30	0.43	–
d^i_1	0.10	0.18	–	−0.25	−0.15	–	0.25	0.21	–	−0.25	−0.46	–
d^i_2	0.70	0.69	–	−0.25	−0.32	–	−0.40	−0.41	–	0.40	0.47	–
d_3	0.10	0.16	–	−0.25	−0.25	–	−0.30	−0.30	–	0.10	0.07	–

the computations already performed at time $t-1$ to compute $\tilde{Q}_{t-1}(y_{t-1}, \mathcal{S}^{t-1})$. This leads to an additional approximation, since the sequence of prediction errors is not simulated all over again, but the predictors and prediction errors obtained at the previous time steps are used along with the inferred sequence $\mathcal{S}^{t-1}$ to update the predicted output and then compute s_t.

6. Case studies

The quality of the models estimated using the proposed approach is shown on a simulation example using synthetic data and on an experimental case study addressing *unsupervised* segmentation of honeybee dances.

Algorithm 1 is run under the assumption that the number of modes K and the order n_a, n_b, n_c and n_d of the BJ sub-models (as in classical linear system identification algorithms) are fixed by the user. The hyper-parameters of the *Gaussian Inverse-Gamma* prior distribution in (13) are set to $\lambda = 10^4$ and $\alpha_0 = \beta_0 = 1$, while all the parameters of the Dirichlet distribution (15) are set to $\alpha_k = 1$ for all $k = 1, \dots, K$. Since no prior knowledge on the variables $\Theta, \sigma_e^2, M, \mathcal{S}^T$ is assumed to be available, these hyper-parameters are chosen to have broad (large variance) "uninformative" prior distributions. Algorithm 1 is iterated for a maximum of $k_{max} = 50$ iterations.

In both the considered examples, Algorithm 1 is initialized with different initial guesses on the mode sequence $\mathcal{S}^{T(0)}$, and the jump BJ model is selected as the outcome that returns the maximum value of the posterior distribution (29).

In optimizing the posterior w.r.t. Θ (Section 4.1.1), the parameters Θ are updated via the Gauss–Newton method (32). The parameter α in (32) is chosen through *exact line search*, that is performed over 100 equally-spaced grid points in the interval [0 1]. The Gauss–Newton method is terminated when either the maximum number of iterations $h_{max} = 50$ is reached or when the terminal condition (36) is satisfied, with tolerance $\epsilon_J = 10^{-8}$.

At the kth iteration of Algorithm 1, the mode sequence $\mathcal{S}^{T(k)}$ is updated using the suboptimal moving-horizon approach presented in Section 4.2, with horizon length $T_c = \max(n_a, n_c, n_d)$.

The true mode sequence $\mathcal{S}^T$ is used only for validation purposes to evaluate the accuracy of the reconstructed sequence $\mathcal{S}^{T\star}$, which is measured through the following *label accuracy index*

$$L_T^{true} = \frac{1}{T}\sum_{t=1}^{T} \mathbb{I}(s_t^\star = s_t) \cdot 100\ \%. \tag{55}$$

The quality of the predicted output is assessed in terms of the *best fit rate* (BFR) index defined as

$$\text{BFR} = \max\left\{1 - \frac{\|\tilde{y} - \hat{y}\|_2}{\|\tilde{y} - m_{\tilde{y}}\|_2}, 0\right\} \cdot 100\ \%, \tag{56}$$

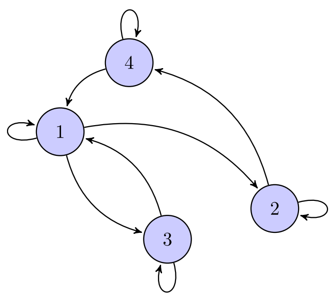


Fig. 1. Numerical example. Directed graph indicating admissible mode transitions.

with $\tilde{y}$ and $\hat{y}$ denoting the vector stacking the measured and predicted output, respectively, and $m_{\tilde{y}}$ being the vector stacking the sample mean of the measured output.

All the tests were run on a MacBook Pro 2.8 GHz Intel i7 in MATLAB R2018b.

6.1. Numerical example using synthetic data

As a data-generating system, let us consider a jump Box–Jenkins system described by (3), with $K = 4$ modes, so to evaluate whether the algorithm is able to reconstruct the four submodels. The linear filters of each BJ sub-model are characterized by $n_a = n_b = n_c = n_d = 3$, with coefficients reported in Table 1. The operating mode switches every 100 samples according to the directed graph reported in Fig. 1, starting from $s_0 = 3$.

The system is excited by a white noise input sequence of length $T = 20,000$, uniformly distributed in the interval $[-1,\ 1]$. The noise term e_t corrupting the output signal is a zero-mean Gaussian random variable with standard deviation $\sigma_e = 0.9$. This corresponds to a *Signal-to-Noise-Ratio* (SNR) equal to

$$\text{SNR} = 10 \log \frac{\sum_{t=1}^{T}(y_t - e_t)^2}{\sum_{t=1}^{T} e_t^2} = 8.7\ \text{dB}. \tag{57}$$

Table 1 shows the true coefficients Θ of the jump BJ (jBJ) data-generating system, along with the estimated parameters. These model parameters are computed by executing Algorithm 1 for five different randomly generated initial sequences $\mathcal{S}^{T(0)}$. For the sake of comparison, the estimated parameters of a jump ARX (jARX) model are reported in the same table. This estimate is obtained by using the approach proposed in this paper for $C(q^{-1}, \theta^i) = 1$ and $D(q^{-1}, \theta^i) = A(q^{-1}, \theta^i)$. Due to the inconsistent noise structure, it can be noticed that the estimated parameters for the jump ARX model are biased.

The label accuracy indexes L_T^{true} attained with the jump BJ model and the jump ARX model on the training set are equal to

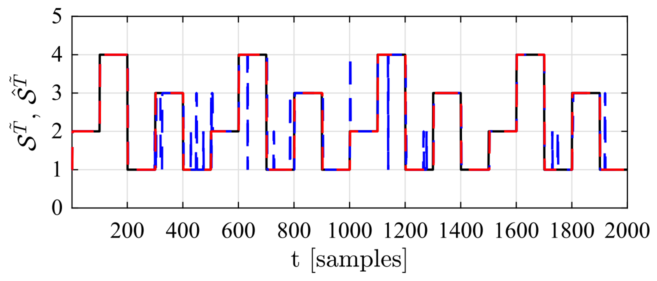


Fig. 2. Numerical example. True mode sequence $\mathcal{S}^{\tilde{T}}$ (black) *vs* sequence estimated with the jump ARX model (dashed blue) and the jump BJ model (dashed red). Black and dashed red lines are almost overlapping. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

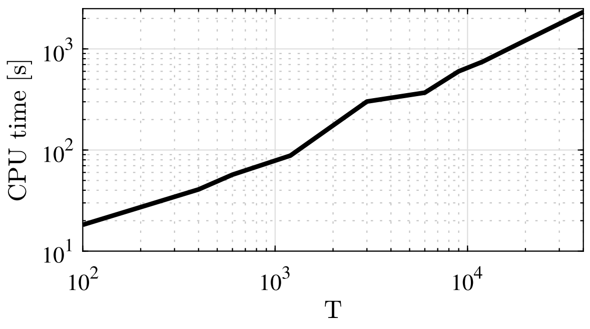


Fig. 4. Numerical example. Length of the dataset T vs CPU time required to fit the jump BJ model.

99.4% and 98.5%, respectively. Thus, a proper choice of the local models' structure seems to have its main benefits in a correct estimation of the local submodels, rather than in reconstructing the true sequence of the discrete state. Since no approximation is introduced in computing the mode sequence for the jump ARX model, these label accuracy indexes also show that the sub-optimal method introduced in Section 4.2 is a viable solution to trade-off between reconstruction accuracy and computational complexity.

To further compare the accuracy of the jump BJ and jump ARX models, both are used to reconstruct the hidden mode sequence and predict the output in a validation dataset of length $\tilde{T} = 2000$, not used for training. Validation data are generated with new noise and input sequences, with input uniformly distributed in the interval $[-5,\ 5]$, and discrete state starting from $s_0 = 2$ and then evolving according to the switching rules described by the direct graph in Fig. 1. By using the previously identified jump models (either BJ or ARX), the mode and the output sequences are reconstructed as explained in Section 5.

Label accuracy indexes $L_{\tilde{T}}^{true} = 98.6\%$ and $L_{\tilde{T}}^{true} = 96.3\%$ are attained for the jump BJ and the jump ARX model, respectively. These results are due to the differences between the reconstructed sequences shown in Fig. 2, where it can be noticed that a jump ARX model leads to spikes that are inconsistent with the behaviour of the data-generating system.

The outputs predicted by the two models are compared in Fig. 3, along with the attained prediction errors of BFR=85.6% and 73.5% attained for the jBJ model and the jARX model, respectively. Although the difference between the inferred sequences is negligible within the considered interval (see Fig. 2), the jump BJ model generally leads to more accurate prediction of the output signal with respect to a jump ARX model. This is due to inconsistency of the jump ARX model structure.

The effect of the approximation introduced by the recursive inference presented in Section 5.1 is evaluated by iteratively estimating the mode sequence and predicting the output signal in the considered validation set. The label accuracy index obtained with the jBJ model is $L_{\tilde{T}}^{true} = 98.6\%$ and it is similar to the one attained by performing batch inference. Therefore, at least in the considered case, the two procedures are equally effective in inferring the mode sequence. However, the best fit rate decrease from BFR=85.6% to 80.3% when the output is predicted iteratively. For a jARX model, the label accuracy index and the best fit rate drop to $L_{\tilde{T}}^{true} = 93.4\%$ and BFR=71.2%, respectively.

The robustness of the proposed estimation method is assessed by performing a Monte Carlo simulation with 25 realizations of the initial state $s_0 \in \{1, 2, 3, 4\}$ and of the input and noise signals. Table 2 reports the mean values and standard deviations for the estimated parameters, showing that the true value of the system's parameters lies within the uncertainty intervals defined by the standard deviation. The achieved mean label accuracy index $\bar{L}_T^{true}$ is 99.4%.

Finally, the computational CPU time required to run Algorithm 1 is evaluated for datasets of different length T. The CPU time increases with T according to the trend shown in Fig. 4, and most of the computational time is spent in maximizing the posterior distribution w.r.t. the model parameters Θ via the Gauss–Newton method.

6.2. Experimental case study: segmenting honeybee dance

The effectiveness of the proposed approach is shown on the benchmark example described in Oh et al. (2008). The goal is to segment a honeybee dance sequence into the $K = 3$ possible operating regimes, namely "turn right", "waggle" and "turn left". The available dataset consists of the 2D coordinates of the bee's body (x_t, y_t) and its head angle θ_t. Six dance time-trajectories are available and shown in Fig. 5(a), where the manually labelled groundtruth operating modes are highlighted with different colours.

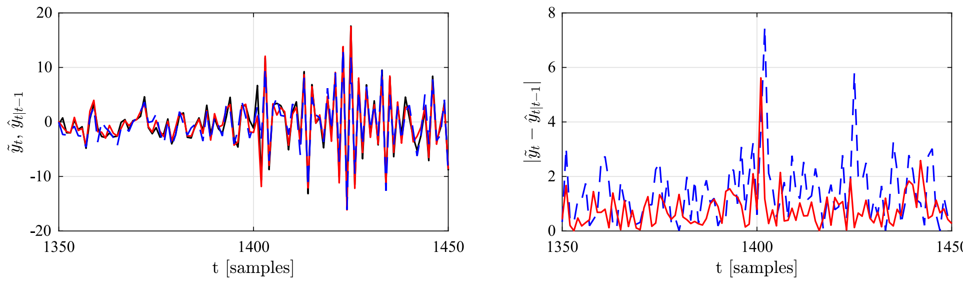


Fig. 3. Numerical example. [Left] True output (black) *vs* predicted output with estimated jump ARX (dashed blue) and jump BJ model (red). [Right] Absolute value of the prediction error attained by the jump ARX (dashed blue) and the jump BJ model (red). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 2
Numerical example. Monte Carlo simulation: true *vs* estimated jump BJ model parameters (mean ± standard deviation).

	$i=1$		$i=2$		$i=3$		$i=4$	
	True	mean ± std	True	mean ± std	True	mean ± std	True	mean ± std
a_1^i	0.60	0.60 ± 0.02	−0.50	−0.51 ± 0.06	0.10	0.10 ± 0.04	−0.70	−0.69 ± 0.05
a_2^i	0.10	0.10 ± 0.02	−0.10	−0.08 ± 0.08	−0.10	−0.10 ± 0.05	0.40	0.38 ± 0.10
a_3^i	0.30	0.30 ± 0.01	−0.10	−0.09 ± 0.08	0.10	0.10 ± 0.03	0.20	0.19 ± 0.07
b_1^i	0.80	0.80 ± 0.02	−1.00	−0.80 ± 1.00	−0.20	−0.20 ± 0.02	4.00	3.80 ± 1.00
b_2^i	−0.80	−0.80 ± 0.03	1.00	1.02 ± 0.12	−0.20	−0.21 ± 0.03	1.00	1.02 ± 0.18
b_3^i	0.80	0.81 ± 0.03	0.10	0.09 ± 0.07	0.60	0.59 ± 0.03	−0.10	−0.09 ± 0.04
c_1^i	0.99	1.04 ± 0.18	−0.20	−0.12 ± 0.13	0.20	0.20 ± 0.03	0.20	0.22 ± 0.21
c_2^i	0.80	0.84 ± 0.15	0.20	0.15 ± 0.10	−0.10	−0.09 ± 0.02	−0.30	−0.26 ± 0.14
c_3^i	0.10	0.13 ± 0.12	−0.20	−0.16 ± 0.10	0.5	0.49 ± 0.03	0.30	0.25 ± 0.12
d_1^i	0.10	0.16 ± 0.19	−0.25	−0.18 ± 0.11	0.25	0.25 ± 0.03	−0.25	−0.21 ± 0.18
d_2^i	0.70	0.69 ± 0.01	−0.25	−0.25 ± 0.14	−0.40	−0.40 ± 0.02	0.40	0.38 ± 0.12
d_3^i	0.10	0.15 ± 0.13	−0.25	−0.25 ± 0.09	−0.30	−0.30 ± 0.02	0.10	0.09 ± 0.08

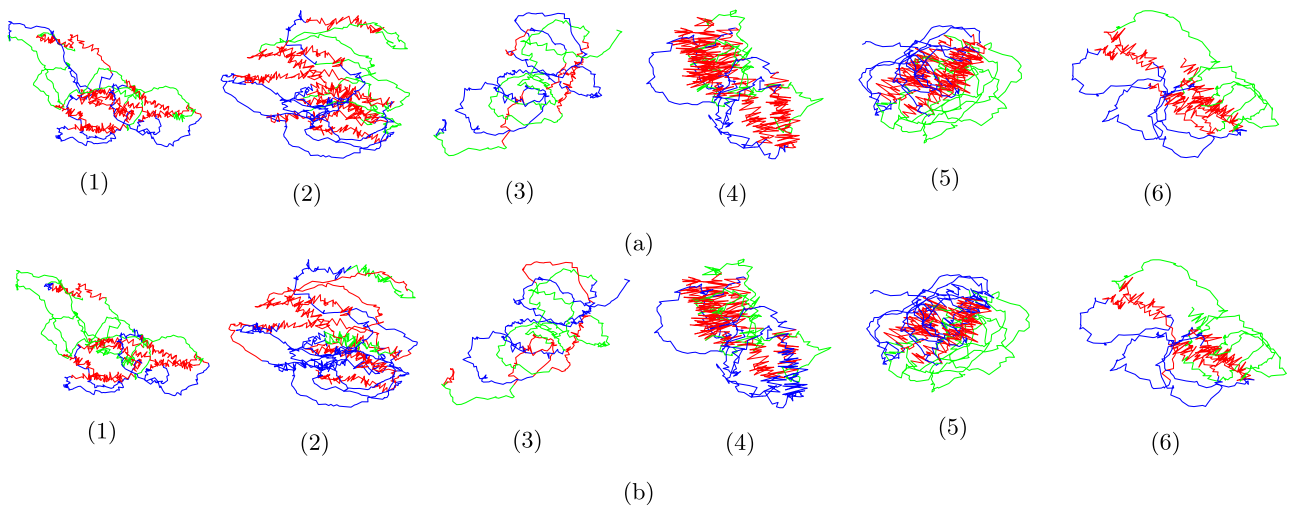


Fig. 5. Benchmark example. Honeybee dance trajectories for sequences 1 to 6. True (upper panel) vs estimated motion patterns (lower panel), with "turn right" motion (blue), "waggle" dance (red) and "turn left" motion (green). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

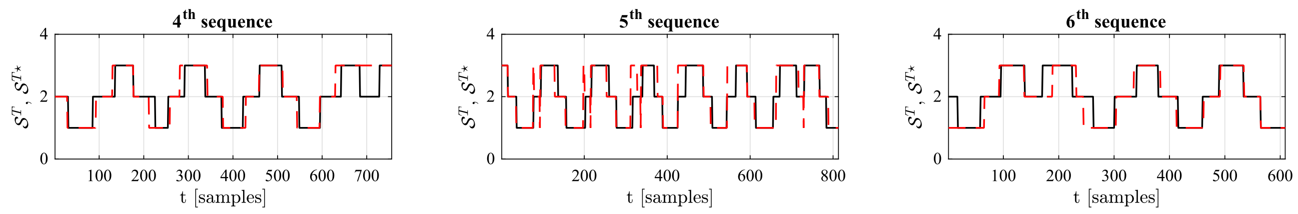


Fig. 6. Honeybee dance example: true (black) vs estimated mode sequence (dashed red) for the 4th, 5th and 6th trajectories.

Algorithm 1 is run for each of the six sequences, setting $K = 3$ and by considering as an output signal the cosine of the bee's head angle (namely, $\mathcal{Y}^T = \{\cos(\theta_t)\}_{t=1}^T$). The remaining data (namely $\sin(\theta_t)$ and (x_t, y_t)) are treated as inputs of the model, after normalizing and detrending the 2D coordinates.

For each of the six sequences, first-order BJ submodels (*i.e.*, $n_a = n_b = n_c = n_d = 1$) are estimated. Algorithm 1 is initialized with two possible mode sequences $\mathcal{S}^{T(0)}$. The first one is constructed by cyclically changing the mode every 50 samples, and the other one is obtained by randomly permuting the elements of the first sequence.

The trajectories of the honeybee dance for all the six sequences are plotted in Fig. 5(b), along with the reconstructed motion patterns. The same information is also reported in Fig. 6 for the 4th, 5th and 6th dance sequence. By comparing the trajectories in Fig. 5, it can be seen that the proposed method generally detects the actual motion pattern for the first sequence and for the 4th, 5th and 6th dance sequence. Instead, Algorithm 1 returns less accurate estimates of the motion patterns for the second and the third sequence, which are characterized by significant variations in the head angle during waggle dances, which makes segmentation more challenging. Nonetheless, a drop in performance for the same datasets is also experienced when using the approach for identification of jump models presented in Fox et al. (2011). Indeed, as shown in Table 3, the label accuracy indexes L_T^{true} (55)

Table 3
Benchmark example. Label accuracy index $L_T^{true}\%$ achieved with Algorithm 1 vs median label accuracy obtained in Fox et al. (2011).

Sequence	1	2	3	4	5	6
Algo 1	79.0	61.0	59.4	83.8	83.7	85.1
Algo (Fox et al., 2011)	46.5	44.1	45.6	83.2	93.2	88.7

attained by using Algorithm 1 is comparable, and in some cases greater, than the median label accuracy achieved in Fox et al. (2011).

7. Conclusions

This paper has proposed a maximum-a-posteriori (MAP) estimation algorithm to fit jump Box–Jenkins (BJ) models to an input/output data set collected from a dynamical system. The posterior distribution of the unknown parameters characterizing the model is computed analytically and then maximized by combining an extension of the prediction-error methods tailored to BJ models with time-varying coefficients and suboptimal moving-horizon dynamic programming, which is used to reconstruct the discrete-state sequence with limited computational complexity. Although convergence to the global optimum is not guaranteed, numerical evidence shows the effectiveness of the overall approach.

Extensions of this work include autotuning of the number K of operating modes and derivation of confidence intervals of the estimated parameters.

References

Bako, L. (2011). Identification of switched linear systems via sparse optimization. *Automatica*, *47*(4), 668–677.

Baum, L., Petrie, T., Soules, G., & Weiss, N. (1970). A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *The Annals of Mathematical Statistics*, *41*(1), 164–171.

Bemporad, A., Breschi, V., Piga, D., & Boyd, S. (2018). Fitting jump models. *Automatica*, *96*, 11–21.

Bemporad, A., & Di Cairano, S. (2011). Model-predictive control of discrete hybrid stochastic automata. *IEEE Transactions on Automatic Control*, *56*(6), 1307–1321.

Bemporad, A., & Morari, M. (1999). Control of systems integrating logic, dynamics, and constraints. *Automatica*, *35*(3), 407–427.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Box, G. E. P., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). *Time series analysis: forecasting and control*. John Wiley & Sons.

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. New York, NY, USA: Cambridge University Press.

Breschi, V., Bemporad, A., Piga, D., & Boyd, S. (2018). Prediction error methods in learning jump ARMAX models. In *Proc. 57th IEEE conf. on decision and control* (pp. 2247–2252). Miami Beach, FL, USA.

Breschi, V., Piga, D., & Bemporad, A. (2016). Piecewise affine regression via recursive multiple least squares and multicategory discrimination. *Automatica*, *73*, 155–162.

Breschi, V., Piga, D., & Bemporad, A. (2019). Maximum-a-posteriori estimation of jump Box–Jenkins models. In *2019 IEEE 58th conference on decision and control* (pp. 1532–1537). Nice, France.

Canty, N., O'Mahony, T., & Cychowski, M. T. (2012). An output error algorithm for piecewise affine system identification. *Control Engineering Practice*, *20*(4), 444–452.

Cinquemani, E., Porreca, R., Ferrari-Trecate, G., & Lygeros, J. (2007). A general framework for the identification of jump Markov linear systems. In *2007 46th IEEE conference on decision and control* (pp. 5737–5742). New Orleans, LA, USA.

Costa, O. L. V., Fragoso, M. D., & Marques, R. P. (2006). Discrete-time Markov jump linear systems. In *Probability and its applications*. Springer London.

Ferrari-Trecate, G., Muselli, M., Liberati, D., & Morari, M. (2003). A clustering technique for the identification of piecewise affine systems. *Automatica*, *39*(2), 205–217.

Fox, E. B., Hughes, M. C., Sudderth, E. B., & Jordan, M. I. (2014). Joint modeling of multiple time series via the beta process with application to motion capture segmentation. *The Annals of Applied Statistics*, *8*(3), 1281–1313.

Fox, E., Sudderth, E. B., Jordan, M. I., & Willsky, A. S. (2011). Bayesian Nonparametric inference of switching dynamic linear models. *IEEE Transactions on Signal Processing*, *59*(4), 1569–1585.

Fridman, M. (1994). *Hidden Markov model regression: Technical report*, Minneapolis, MN: Institute of Mathematics, University of Minnesota.

Fu, T., Chung, F., Ng, V., & Luk, R. 2001. Evolutionary segmentation of financial time series into subsequences. In *Proceedings of the 2001 congress on evolutionary computation (Vol. 1)*. (pp. 426–430). Seoul, South Korea.

Garulli, A., Paoletti, S., & Vicino, A. (2012). A survey on switched and piecewise affine system identification. In *16th IFAC symposium on system identification IFAC Proceedings Volumes*, *45*(16), 344–355.

Gong, D., Medioni, G., & Zhao, X. (2014). Structured time series analysis for human action segmentation and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *36*(7), 1414–1427.

Goudjil, A., Pouliquen, M., Pigeon, E., & Gehan, O. (2017). Identification algorithm for MIMO switched output error model in presence of bounded noise. In *IEEE 56th annual conference on decision and control* (pp. 5286–5291).

Henzinger, T. A. (1996). The theory of hybrid automata. In *Proceedings of the 11th annual IEEE symposium on logic in computer science* (p. 278). Washington, DC, USA: IEEE Computer Society.

Kun Huang, Wagner, A., & Yi Ma (2004). Identification of hybrid linear time-invariant systems via subspace embedding and segmentation (SES). In *2004 43rd IEEE conference on decision and control (CDC) (IEEE Cat. No.04CH37601) (Vol. 3)* (pp. 3227–3234). Nassau, Bahamas.

Ljung, L. (1999). *System identification: Theory for the user*. Prentice-Hall Englewood Cliffs, NJ.

Naik, V. V., Mejari, M., Piga, D., & Bemporad, A. (2017). Regularized moving-horizon piecewise affine regression using mixed-integer quadratic programming. In *25th Mediterranean conference on control and automation* (pp. 1349–1354).

Nguyen, N. (2018). Hidden Markov model for stock trading. *International Journal of Financial Studies*, *6*(2), 36.

Oh, S. M., Rehg, J. M., Balch, T., & Dellaert, F. (2008). Learning and inferring motion patterns using parametric segmental switching linear dynamic systems. *International Journal of Computer Vision*, *77*(1–3), 103–124.

Ohlsson, H., & Ljung, L. (2013). Identification of switched linear regression models using sum-of-norms regularization. *Automatica*, *49*(4), 1045–1050.

Ostendorf, M., Digalakis, V. V., & Kimball, O. A. (1996). From hmm's to segment models: A unified view of stochastic modeling for speech recognition. *IEEE Transactions on Speech and Audio Processing*, *4*(5), 360–378.

Ozay, N., Lagoa, C., & Sznaier, M. (2015). Set membership identification of switched linear systems with known number of subsystems. *Automatica*, *51*, 180–191.

Özkan, E., Lindsten, F., Fritsche, C., & Gustafsson, F. (2015). Recursive maximum likelihood identification of jump Markov nonlinear systems. *IEEE Transactions on Signal Processing*, *63*(3), 754–765.

Piga, D., Bemporad, A., & Benavoli, A. (2020). Rao-blackwellized sampling for batch and recursive Bayesian inference of piecewise affine models. *Automatica*, *117*.

Piga, D., & Tóth, R. (2013). An SDP approach for ℓ_0-minimization: Application to ARX model segmentation. *Automatica*, *49*(12), 3646–3653.

Rabiner, L. R. (1990). A tutorial on hidden Markov models and selected applications in speech recognition. In *Readings in speech recognition* (pp. 267–296). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Rosenqvist, F., & Karlström, A. (2005). Realisation and estimation of piecewise-linear output-error models. *Automatica*, *41*(3), 545–551.

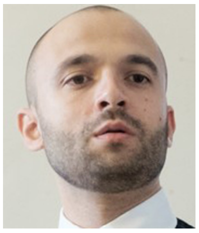
Sontag, E. (1981). Nonlinear regulation: The piecewise linear approach. *IEEE Transactions on Automatic Control*, *26*(2), 346–358.

Torrisi, F., & Bemporad, A. (2004). HYSDEL – A tool for generating computational hybrid models for analysis and synthesis problems. *IEEE Transactions on Control Systems Technology*, *12*(2), 235–249.

Viterbi, A. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, *13*(2), 260–269.

Yang, Y., Qin, Y., Pan, Q., Yang, Y., & Li, Z. (2017). LMMSE estimation of Markovian jump linear systems with random parameters and estimate feedback. In *20th international conference on information fusion* (pp. 1–7). Xian, China.

Zheng, F., Derrode, S., & Pieczynski, W. (2019). Parameter estimation in switching Markov systems and unsupervised smoothing. *IEEE Transactions on Automatic Control*, *64*(4), 1761–1767.



Dario Piga received his Ph.D. in Systems Engineering from the Politecnico di Torino (Italy) in 2012. He was Assistant Professor at the IMT School for Advanced Studies Lucca (Italy) and since March 2017 he has been Senior Researcher at the IDSIA Dalle Molle Institute for Artificial Intelligence in Lugano (Switzerland) and Adjunct Professor at the SUPSI University of Applied Sciences and Arts of Southern Switzerland. His main research interests include system identification, deep learning, Bayesian filtering and non-convex optimization, with applications to process control and smart manufacturing.

Valentina Breschi received her Master's degree in Electrical and System Engineering in 2014 from the University of Florence, Italy. She received her Ph.D. in Control Engineering in 2018 from IMT School for Advanced Studies Lucca. In 2017 she was with the Department of Aerospace Engineering, University of Michigan, Ann Arbor. In 2018–2020 she held a postdoctoral position at Politecnico di Milano, where she is currently a junior Assistant Professor. Her main research interests include systems identification, data-driven control, data-analysis and policy design for mobility systems.

Alberto Bemporad received his Master's degree in Electrical Engineering in 1993 and his Ph.D. in Control Engineering in 1997 from the University of Florence, Italy. In 1996/97 he was with the Center for Robotics and Automation, Department of Systems Science & Mathematics, Washington University, St. Louis. In 1997–1999 he held a postdoctoral position at the Automatic Control Laboratory, ETH Zurich, Switzerland, where he collaborated as a senior researcher until 2002. In 1999–2009 he was with the Department of Information Engineering of the University of Siena, Italy, becoming an Associate Professor in 2005. In 2010–2011 he was with the Department of Mechanical and Structural Engineering of the University of Trento, Italy. Since 2011 he is Full Professor at the IMT School for Advanced Studies Lucca, Italy, where he served as the Director of the institute in 2012–2015. He spent visiting periods at Stanford University, University of Michigan, and Zhejiang University. In 2011 he cofounded ODYS S.r.l., a company specialized in developing model predictive control systems for industrial production. He has published more than 350 papers in the areas of model predictive control, hybrid systems, optimization, automotive control, and is the co-inventor of 16 patents. He is author or coauthor of various MATLAB toolboxes for model predictive control design, including the Model Predictive Control Toolbox (The Mathworks, Inc.) and the Hybrid Toolbox. He was an Associate Editor of the IEEE Transactions on Automatic Control during 2001–2004 and Chair of the Technical Committee on Hybrid Systems of the IEEE Control Systems Society in 2002–2010. He received the IFAC High- Impact Paper Award for the 2011–14 triennial and the IEEE CSS Transition to Practice Award in 2019. He is an IEEE Fellow since 2010.