Incremental Segmentation of ARX Models [⋆]

Glen Chou, Necmiye Ozay, Dmitry Berenson

Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48105 USA (e-mail: gchou,necmiye,dmitryb@umich.edu).

Abstract: We consider the problem of incrementally segmenting auto-regressive models with exogenous inputs (ARX models) when the data is received sequentially at run-time. In particular, we extend a recently proposed dynamic programming based polynomial-time algorithm for offline (batch) ARX model segmentation to the incremental setting. The new algorithm enables sequential updating of the models, eliminating repeated computation, while remaining optimal. We also show how certain noise bounds can be used to detect switches automatically at run-time. The efficiency of the approach compared to the batch method is illustrated on synthetic and real data.

Keywords: Switching autoregressive models, change detection, dynamic programming

1. INTRODUCTION AND RELATED WORK

Time series segmentation is a fundamental problem due to its relevance in a plethora of domains as broad as econometrics (Yin et al. (2011)), biological systems (Omranian et al. (2015); Gersch (1970), Bodenstein and Praetorius (1977)), and land cover monitoring (Mithal et al. (2012)). In particular, segmentation of ARX models - a model for representing time series data with input - has received a great amount of attention from the dynamical systems (Ozay et al. (2008b); Ohlsson et al. (2010); Ozay et al. (2012)) and computer vision (Ozay et al. (2008a)) communities, largely due to its application to anomaly and change detection problems.

Many ARX segmentation algorithms exist for the "batch" setting, in which all data is observed at once. With ℓ_∞ norm bounded noise, a greedy algorithm (Ozay et al. (2008b, 2012)) finds an exact solution. Convex relaxation-based methods have been proposed for ℓ_1 and ℓ_2 norm bounded noise (Ozay et al. (2008b, 2012); Ohlsson et al. (2010); Piga and Tóth (2013)), but the solution of the relaxed problems may be far from the solution of the original problem for some cases. These methods also require the tuning of a regularization parameter to trade off between the number of switches and the quality of the fit. For arbitrary, unbounded noise descriptions, there exist probabilistic approaches (Li et al. (2003)) but these tend to have weaker (if any) optimality guarantees. Contrary to previous approaches, Ozay (2016) proposes a Dynamic Programming (DP) method for solving the batch ARX segmentation problem for broader noise descriptions than ℓ_∞ bounded noise. The algorithm is more computationally-efficient than the relaxation-based methods - it runs in polynomial time - and solves the problem exactly: given a fixed number of switches, it returns the parameters and switches minimizing the fit error.

The "incremental" setting, in which the time series data is received in a streaming manner, is another area rich with applications like analysis of mobile sensor data (Guo et al. (2012)) and fuel monitoring (Gustafsson (2000)). In these cases, we want to perform the segmentation incrementally and update the model as we receive more data.

While several segmentation algorithms exist for generic time series (Keogh et al. (2001), Qi et al. (2015)) due to the reduced structure compared to ARX models, the approaches tend to fit the data based on a heuristically chosen set of basis functions and do not consider the case when input data is also provided. Less work has been done in the incremental ARX segmentation setting. Vidal (2008) tackles the problem by viewing the individual simple ARX models for each segment together as part of a single more complex ARX model in a lifted space, but the convergence of this method is sensitive to noise. The greedy algorithm in Ozay et al. (2008b, 2012) works for the incremental case, but is limited to ℓ_∞ bounded noise.

In this paper, we generalize the dynamic programming batch ARX model segmentation method to the streaming data setting, maintaining exactness of the solution while eliminating repeated computation. Our contributions are twofold: 1) to extend the DP method in Ozay (2016) to the incremental case, 2) for various noise descriptions, to provide a lower bound on the number of switches to detect switches automatically at run-time. We then demonstrate the speed and accuracy of our incremental algorithm and switch detection on both synthetic and real datasets.

2. PRELIMINARIES AND PROBLEM SETUP

2.1 Preliminaries and Notation

In this paper, we consider time-varying affine autoregressive exogenous models of the form:

$$y_t = \sum_{i=1}^{n_a} a_t^i y_{t-i} + \sum_{i=1}^{n_c} c_t^i u_{t-i} + k_t + \eta_t \tag{1}$$

[⋆] This work was supported by DARPA grant N66001-14-1-4045 and ONR grant N000141712050.

where u_t, y_t and η_t denote the input, output and noise at time step t, respectively, and where $t \in \{t_0, \ldots, t_F\}$, with $t_0 = \max(n_a, n_c)$. The parameters at time step t $\mathbf{p}_t \doteq [a_t^1, \ldots, a_t^{n_a}, c_t^1, \ldots, c_t^{n_c}, k_t]^\top$ are unknown. When the parameter vector is constant (i.e., $\mathbf{p}_t = \mathbf{p}^*$ for all t), we recover the time-invariant ARX models in Ljung (1999). We define the regressor vectors as $\mathbf{r}_t \doteq [y_{t-1}, \ldots, y_{t-n_a}, u_{t-1}, \ldots, u_{t-n_c}, 1]^\top$ for $t \in \{t_0, \ldots, t_F\}$. Then, the model (1) can be written as

$$y_t = \mathbf{p}_t^\top \mathbf{r}_t + \eta_t. \tag{2}$$

Instead of model parameters varying at each time, we focus on models where $\mathbf{p}_t$ is piecewise constant in t. A *switch point* (or switch for short) is a time t such that $\mathbf{p}_t \neq \mathbf{p}_{t+1}$. The i^{th} switch point τ_i is then given by $\tau_i = \min_{t \in \{t_0, \ldots, t_c\}}$ subject to $\left\| [||\mathbf{p}_{t_0+1} - \mathbf{p}_{t_0}||, \ldots, ||\mathbf{p}_{t_c} - \mathbf{p}_{t_c-1}||]^\top \right\|_0 = i+1$. The i^{th} *segment* $\mathbf{s}_i$ is the time interval $\{\tau_i, \ldots, \tau_i'\}$, where $\tau_i' := \tau_{i+1} - 1$. We also overload τ by defining the function $\tau(t) := \tau_i$ if $t \in \{\tau_i, \ldots, \tau_i'\}$ that maps the time step t to the the start of the segment that t lies within. The mapping τ' is defined similarly, i.e., $\tau'(t) := \tau_i'$ if $t \in \{\tau_i, \ldots, \tau_i'\}$.

The following notation is used through out the paper:

$(\mathbf{U}, \mathbf{Y}) := (u_{t_0:t_F}, y_{t_0:t_F})$	a stream of input-output data received incrementally over time horizon $t_0 \ldots t_F$. $\mathbf{Y}_{i,j}$ denotes $[y_i, \ldots, y_j]^\top$; $\mathbf{Y}_\mathbf{s}$ denotes vector indexing .
$\boldsymbol{\eta} := \eta_{t_0:t_F}$	a process noise sequence, also indexed similarly.
$\bar{\mathbf{p}}_{i,j}; \bar{\mathbf{p}}_\mathbf{s}$	least squares fit on $\mathbf{Y}_{i,j}; \mathbf{Y}_\mathbf{s}$
$\bar{\mathbf{p}}_{t_i:t_j}$	$\{\mathbf{p}_{t_i}, \ldots, \mathbf{p}_{t_j}\}$
t_c	current time step
$H := t_F - t_0 + 1$	the total time horizon
$H_c := t_c - t_0 + 1$	the current time horizon
d	the dimension of the regressor $\mathbf{r}_t \Leftrightarrow n_a + n_c + 1$
$\mathbf{R} := [\mathbf{r}_{t_0} \ldots \mathbf{r}_{t_F}]^\top$	the vertically stacked matrix of regressors; similar indexing.
$\mathbf{I}$	identity matrix of suitable dimension
$\|\cdot\|_0$	the ℓ_0-quasinorm, equal to the number of non-zero entries in the vector
$\|\cdot\|$	ℓ_2 norm
$\mathbf{Y}^* = \mathbf{Y} - \boldsymbol{\eta}$	"cleaned" output;
$\mathbf{p}_t^*$	true parameter at time t

2.2 Problem Statement

We focus on the incremental segmentation of ARX models. That is, we are given a data stream of input-output pairs $(\mathbf{U}, \mathbf{Y})$ generated by a time-varying ARX system. We assume the underlying process is corrupted by noise. The data appears in streaming fashion, i.e., one at a time, and is of finite duration. We want to incrementally fit a "simple" time-varying ARX model to the data as it is received. Formally, we want to solve the following problem:

Problem 1. (Incremental segmentation). At each time t_c, given data (u_{t_c}, y_{t_c}), and a possibly time-varying upper bound m_{t_c} on the number of segments, for each $0 \leq m \leq m_{t_c}$, find a set of parameters $\bar{\mathbf{p}}_{t_0:t_c}$ such that

$$\begin{aligned} \bar{\mathbf{p}}_{t_0:t_c}(m) &= \arg\min_{\mathbf{p}_{t_0:t_c}} \sum_{t=t_0}^{t_c} (y_t - \mathbf{p}_t^\top \mathbf{r}_t)^2 \\ \text{s.t.} \quad & \left\| [||\mathbf{p}_{t_0+1} - \mathbf{p}_{t_0}||, \ldots, ||\mathbf{p}_{t_c} - \mathbf{p}_{t_c-1}||]^\top \right\|_0 \leq m. \end{aligned} \tag{3}$$

Note that this problem amounts to finding an ARX model at each time with minimum squared fitting error with at most $m+1$ segments. We denote this minimum error, which is the optimal objective value of Problem 1, as $E(m, t_c)$.

The problem can naïvely be solved by repeatedly applying the DP batch approach proposed in Ozay (2016), at each time step. However this requires the bulk of the computations to be repeated redundantly. In this paper, we develop an algorithm that leverages the computation of $\bar{\mathbf{p}}_{t_0:t_c}(m)$ in computing $\bar{\mathbf{p}}_{t_0:t_c+1}(m)$ to eliminate redundancy. Next, we briefly summarize the DP solution to the batch problem.

We start by noting that batch problem is equivalent to waiting until $t_c = t_F$ and solving for $E(m_{t_F}, t_F)$. As shown in Ozay (2016), $E(m_{t_F}, t_F)$ can be efficiently computed using a dynamic programming recursion (see Bellman and Roth (1969) for an overview of dynamic programming). The algorithm exploits the fact that the number of segments is quadratic in the time horizon, and hence by exhaustively computing and storing the fit error and parameters of a time-invariant ARX model for each segment, the problem can be solved by stitching together the $m_{t_F} + 1$ segments that result in the minimum error.

Let $e_{i,j} := \min_\mathbf{p} \sum_{t=i}^{j} (y_t - \mathbf{r}_t^\top \mathbf{p})^2$ be the error obtained by fitting a single time-invariant ARX model to data $(u_{i:j}, y_{i:j})$. The dynamic programming recursion is as follows:

$$E(0, t) = e_{t_0,t} \text{ for all } t \in [t_0, t_F], \tag{4}$$

and for $m = 1, \ldots, m_{t_F}$ and for $t = \max(m+1, t_0), \ldots, N$,

$$E(m, t) = \min_{m \leq j \leq t-1} \left[E(m-1, j) + e_{j+1,t} \right]. \tag{5}$$

Intuitively, this recursion says that if we have the "best" segmentations with $m-1$ switch points over $\{t_0, \ldots, t_j\}$, where $j \in \{m, \ldots, t-1\}$, and if we can also compute the fit errors $e_{j+1,t}$ for all $t = \max(m+1, t_0), \ldots, N$, we can compute the optimal error and switch point for m switch points by minimizing the objective in Equation (5) with respect to j. This is due to the principle of optimality (see Ozay (2016)).

We also note that $E(m, t)$ for different values of m can be interpreted as a time-varying Pareto frontier describing the trade-off in error and number of switches.

Complexity analysis: To compute $E(m, t_F)$, we need to compute $\frac{H(H-1)}{2}$ time invariant errors $e_{i,j}$, each of which involves an $O(Hd^2)$ linear regression operation (i.e. solving $\mathbf{R}_{i,j}\bar{\mathbf{p}} = \mathbf{Y}_{i,j}$ for the parameter $\bar{\mathbf{p}}$). The dynamic programming recursion requires $O(mH)$ additions and an $O(mH)$ pairwise comparison to compute (5), which must be dominated by the regressions (since $m < H$), yielding an overall complexity of $O(H^3 d^2)$.

```
Input: $u_{t_c+1}; y_{t_c+1}; m_{t_c+1}; E(0:m_{t_c}, t_0:t_c); e_{i,j}$
Output: $E(0:m_{t_c+1}, t_c+1), \bar{\mathbf{p}}_{t_0:t_c+1}$
1: update $e_{i,t_c+1}$ for all $i < t_c+1$ using (6) and (7)
2: using $E(0:m_{t_c}, t_0:t_c)$, run recursion in (4) and (5)
   to compute $E(0:m_{t_c+1}, t_c+1)$
```

Algorithm 1. Incremental update from t_c to t_c+1

3. INCREMENTAL ALGORITHM

3.1 Incremental Algorithm

We want to extend the batch algorithm to the incremental setting in a scalable fashion. Naïvely, one could repeatedly solve the batch problem from scratch upon receiving each new (u_{t_c+1}, y_{t_c+1}), but such a method would have a $O(H^4d^2)$ complexity. We would like to do better by reusing information used in solving $E(m, t_c)$ to solve $E(m, t_c+1)$. The recursion in the batch algorithm is in m; we next observe a similar recursion in the update rules in terms of t.

Theorem 1. To exactly compute $E(m, t_c+1)$ using $E(m, t_c)$, it is sufficient to compute e_{i,t_c+1}, for all $i \in \{t_0, \ldots, t_c\}$ and $E(j, t_c+1)$, for all $j \in \{1, \ldots, m_{t_c+1}\}$.

Proof. This follows by noting that $E(m_{t_c}, t_c+1) = \min_{m_{t_c} \le j \le t_c}[E(m_{t_c}-1, j) + e_{j,t_c+1}]$. □

Additionally, we note that we can avoid computing e_{i,t_c+1}, for all $i < t_c$ from scratch by updating the parameters associated with e_{i,t_c}, for all $i < t_c$ using Recursive Least Squares (RLS) (Åstrom and Wittenmark (1994)):

$$\bar{\mathbf{p}}_{i,t_c+1} = \bar{\mathbf{p}}_{i,t_c} + \frac{\boldsymbol{\pi}(t_c) r_{t_c+1}}{1 + r_{t_c+1}^\top \boldsymbol{\pi}(t_c) r_{t_c+1}} (y_{t_c+1} - r_{t_c+1}^\top \bar{\mathbf{p}}_{i,t_c})$$

$$\boldsymbol{\pi}(t_c+1) = \boldsymbol{\pi}(t_c) - \frac{\boldsymbol{\pi}(t_c) r_{t_c+1} r_{t_c+1}^\top \boldsymbol{\pi}(t_c)}{1 + r_{t_c+1}^\top \boldsymbol{\pi}(t_c) r_{t_c+1}} \qquad (6)$$

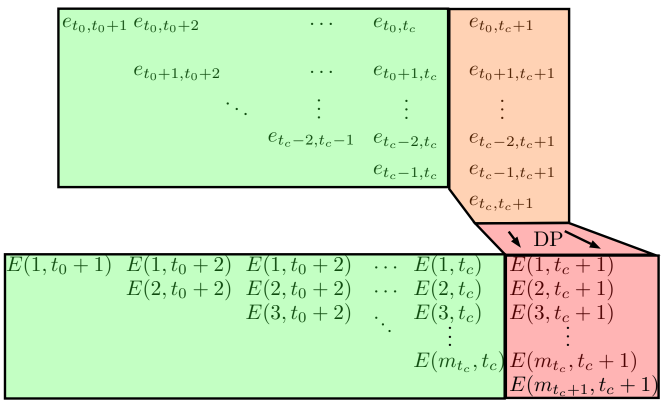
and recovering the errors

$$e_{i,t_c+1} = \|\mathbf{Y}_{i,t_c+1} - \mathbf{R}_{i,t_c+1}\bar{\mathbf{p}}_{i,t_c+1}\|^2, \qquad (7)$$

where $\boldsymbol{\pi}(t_c) := (\mathbf{R}_{i,t_c}\mathbf{R}_{i,t_c}^\top)^{-1}$ is an intermediate quantity updated through time.

The pseudocode of the incremental algorithm can be found in Algorithm 1 and a visualization of the computation can be found in Figure 1. A minor implementation detail is that in order to initialize RLS, one needs to accumulate enough data such that $\pi(t_c)$ is nonsingular; it usually suffices to get d samples. It is possible to modify Algorithm 1 to consider only a recent window of n data points: an approximation for the full-length Pareto frontier. An additional benefit of this extension is that we can avoid having to store the data $(\mathbf{U}, \mathbf{Y})$ for long data streams. The analysis of such an extension is the subject of future work.

Complexity analysis: For each time step, the online algorithm needs to compute $\frac{H_c(H_c-1)}{2}$ time-invariant errors, each of which requires $O(d^2)$ to update using RLS. The dynamic programming update still requires $O(mH_c)$ computation, which is dominated by the regressions. Hence, we have an overall complexity of $O(H_c^2 d^2)$ per time step.



e_{t_0,t_0+1} e_{t_0,t_0+2} $\cdots$ e_{t_0,t_c}
e_{t_0,t_c+1}
e_{t_0+1,t_0+2} $\cdots$ e_{t_0+1,t_c}
e_{t_0+1,t_c+1}
e_{t_c-2,t_c-1} e_{t_c-2,t_c}
e_{t_c-2,t_c+1}
e_{t_c-1,t_c}
e_{t_c-1,t_c+1}
e_{t_c,t_c+1}
DP
$E(1,t_0+1)$ $E(1,t_0+2)$ $E(1,t_0+2)$ $\cdots$ $E(1,t_c)$
$E(2,t_0+2)$ $E(2,t_0+2)$ $\cdots$ $E(2,t_c)$
$E(3,t_0+2)$ $E(3,t_c)$
$E(m_{t_c},t_c)$
$E(1,t_c+1)$
$E(2,t_c+1)$
$E(3,t_c+1)$
$E(m_{t_c},t_c+1)$
$E(m_{t_c+1},t_c+1)$

Fig. 1. Visualization of the incremental algorithm at time step t_c+1. Green indicates no new computation; orange indicates computation via update; red indicates computation from scratch.

3.2 Switch detection heuristics

The incremental algorithm assumes that an upper bound m_{t_c} on the number of segments is given at each time. In this section, we discuss how some side information on the noise characteristics can be used to determine if a switch has occurred in the data at run time.

First, we show that if we are given the noise vector $\boldsymbol{\eta}$, we can compute $E(m^*, t_c)$. We also provide upper bounds on $E(m^*, t_c)$ for when we lack $\boldsymbol{\eta}$ but have bounds on the noise in the ℓ_∞ and ℓ_2 sense, which can be used as necessary conditions to introduce a switch, therefore leading to a lower bound on the true number $m^*_{t_c}$ of switches.

Lemma 2. Given noisy data and assuming all switches are correctly identified, the total squared error $E(m^*, t_c)$ using the true number of switches m^* is given by

$$E(m^*, t_c) = \boldsymbol{\eta}^\top(\mathbf{I} - \mathrm{diag}(\mathbf{P}_{\mathbf{s}_1}, \ldots, \mathbf{P}_{\mathbf{s}_{m^*+1}}))\boldsymbol{\eta} \qquad (8)$$

where $\mathbf{P}_{\mathbf{s}_i} = \mathbf{R}_{\mathbf{s}_i}^\top(\mathbf{R}_{\mathbf{s}_i}\mathbf{R}_{\mathbf{s}_i}^\top)^{-1}\mathbf{R}_{\mathbf{s}_i}$.[1]

Proof. Note that the parameter estimation error for least squares (with correct switches) is given by $\bar{\mathbf{p}}_t - \mathbf{p}_t^* = (\mathbf{R}_{\tau(t),\tau'(t)}\mathbf{R}_{\tau(t),\tau'(t)}^\top)^{-1}\mathbf{R}_{\tau(t),\tau'(t)}\boldsymbol{\eta}_{\tau(t),\tau'(t)}$, where we take $\tau'(t) = t_c$ for $t > \tau(t)$. Now, using this fact and summing the error over segments gives:

$$\begin{aligned}
E(m^*, t_c) &= \sum_{i=\mathbf{s}_1}^{\mathbf{s}_{m^*+1}} \|\mathbf{Y}_i^* + \boldsymbol{\eta}_i - \mathbf{R}_i^\top \bar{\mathbf{p}}_i\|^2 \\
&= \sum_{i=\mathbf{s}_1}^{\mathbf{s}_{m^*+1}} \|\mathbf{Y}_i^* + \boldsymbol{\eta}_i + \mathbf{R}_i^\top \mathbf{p}_i^* - \mathbf{R}_i^\top \mathbf{p}_i^* - \mathbf{R}_i^\top \bar{\mathbf{p}}_i\|^2 \\
&= \sum_{i=\mathbf{s}_1}^{\mathbf{s}_{m^*+1}} \|\boldsymbol{\eta}_i\|^2 - 2\boldsymbol{\eta}_i^\top \mathbf{P}_i \boldsymbol{\eta}_i + \|\mathbf{P}_i\boldsymbol{\eta}_i\|^2 \\
&= \|\boldsymbol{\eta}\|^2 - \sum_{i=\mathbf{s}_1}^{\mathbf{s}_{m^*+1}} \|\mathbf{P}_i\boldsymbol{\eta}_i\|^2 \\
&= \boldsymbol{\eta}^\top(\mathbf{I} - \mathrm{diag}(\mathbf{P}_{\mathbf{s}_1}, \ldots, \mathbf{P}_{\mathbf{s}_{m^*+1}}))\boldsymbol{\eta},
\end{aligned}$$

[1] In case $(\mathbf{R}_{\mathbf{s}_i}\mathbf{R}_{\mathbf{s}_i}^\top)$ is not invertible due to the true segment being short or not persistently exciting, an error bound can still be computed using pseudo-inverses.

where we also use the fact that given the correct switches on the noiseless data, regression will return zero error. □

Using this result and some algebra, we can derive bounds for different types of side information in terms of ℓ_2 noise bounds:

(i) [*Root mean square (RMS) noise bound*]: If for a given t_c, $\sqrt{\frac{1}{t_c - t_0 + 1}\sum_{t=t_0}^{t_c} \eta_t^2} \le \eta_{\max}$, then $E(m^*_{t_c}, t_c) \le B^{\text{run}}_{\ell_2}(t_c)$, where

$$B^{\text{run}}_{\ell_2}(t_c) := (t_c - t_0 + 1)\eta^2_{\max}. \tag{9}$$

This follows from noting that the noise specification is equivalent to $\sum_{t=t_0}^{t_c} \eta_t^2 < (t_c - t_0 + 1)\eta^2_{\max}$. If the switches are detected correctly and thus have no systematic fit error, we expect $E(m^*, t_c) = \sum_{t=t_0}^{t_c} e_t^2 \le \sum_{t=t_0}^{t_c} \eta_t^2$.

(ii) [*Square noise bound*]: If only a bound on the total ℓ_2 norm of the noise is known, i.e., $||\boldsymbol{\eta}|| \le \eta_{\max}$, then $E(m^*_{t_c}, t_c) \le B^{\text{tot}}_{\ell_2}(t_c)$, where

$$B^{\text{tot}}_{\ell_2}(t_c) := \eta^2_{\max}. \tag{10}$$

The derivation follows the same logic as for the RMS noise bound, except $\sum_{t=t_0}^{t_c} \eta_t^2 < \eta^2_{\max}$.

A few remarks on these bounds are in order. For all upper bounds (9), (10), we never add a segment when one does not occur. This is due to the fact that the error computed by the algorithm is a lower bound on the true error. If this lower bound exceeds the computed upper bounds ($E(m, t_c) > B$), the true noise necessarily exceeds it. Therefore, $E(m, t_c)$ being monotonically non-increasing in m suggests that m should be incremented if it is known that the true noise satisfies the side information. Similar bounds can be obtained for ℓ_∞ type noise descriptions as well.

It is also worth noting that the side information is only used for deriving the bounds on error, and is considered as a soft/heuristic constraint. That is, the problem solved still aims to minimize the least squares error and the resulting segmentation at t_c might violate the noise bounds for $t < t_c$: the bound is only enforced at t_c. The side information is not fully integrated by enforcing the noise bounds at each time (e.g., as in a set membership identification framework), which is possible but not pursued in this paper.

4. RESULTS

First, we demonstrate our incremental method returns the same solution as the batch method by comparing our results with Ozay (2016).

Example 1: Accuracy of incremental method In this example, we fit MATLAB earthquake data from `quake.mat` and compare with the segmentation results on the same data in Ozay (2016). As in Ozay (2016), we use a second order auto-regressive model $y_t = a_t^1 y_{t-1} + a_t^2 y_{t-2} + \eta_t$ and allocate one switch. As shown in Figure 2, we find that the switch point and parameter are identical. We also perform a fit using these parameters, which is calculated as $y_t^{\text{fit}} = a_t^1 y_{t-1} + a_t^2 y_{t-2}$, where the fit for each time step is computed using the true data from the prior two time steps. A comparison of this fit compared to the true data is provided in Figure 3. As we can see, the fit retains

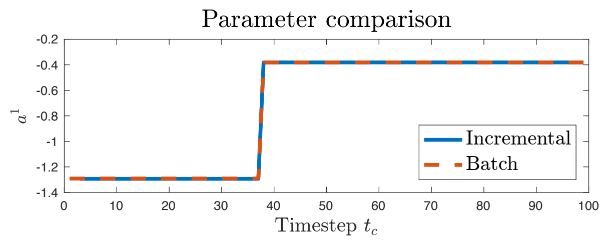


Fig. 2. Comparison for a^1 on `quake.mat` using $m = 1$.

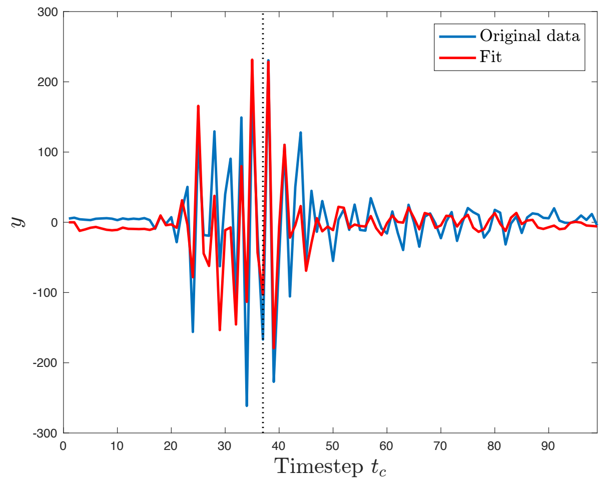


Fig. 3. Comparison for data fit on `quake.mat` using $m = 1$.

the characteristics of the underlying data and chooses a reasonable switch point.

Example 2: Computation time We demonstrate that running our incremental method on a data stream received over horizon $\{t_0, \ldots, t_F\}$ has computation time similar to running the batch method in one shot on the complete data set. We also demonstrate that running the batch algorithm naïvely (i.e. running it repeatedly from scratch upon receiving new data in the incremental setting) results in massive computation cost.

The computation times for segmenting data generated by the time-varying ARX system

$$y_t = \begin{cases} [-0.4,\ -0.1,\ 0,\ 0.1,\ 0.2]\, r_t + \eta_t & t_0 \le t < \dfrac{t_F - t_0}{2} \\ -[0.1,\ 0.3,\ 0.1,\ -0.2,\ 0.2]\, r_t + \eta_t & \dfrac{t_F - t_0}{2} \le t \le t_F \end{cases}$$

where $r_t = [y_{t-1}, y_{t-2}, y_{t-3}, u_{t-1}, u_{t-2}]^\top$, are shown in Figure 4. Results were collected on a 2.3 GHz 2012 Macbook Pro.

As we can see, the naïve algorithm is intractable, and the incremental algorithm performs similarly to the single-shot batch algorithm at smaller horizons but scales slightly better with longer horizons. This is likely due to recursive least squares being a more attractive alternative to matrix inversion at very long horizons.

Example 3: Demonstration on synthetic data We demonstrate the switch detection bounds on synthetic input-output pairs generated by the time-varying ARX system

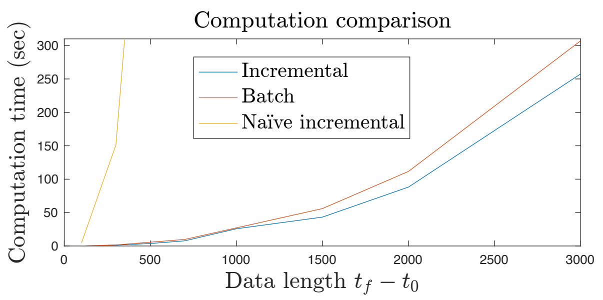


Fig. 4. Comparison of computation times for our incremental algorithm, the batch algorithm in Ozay (2016), and naïve incremental algorithm.

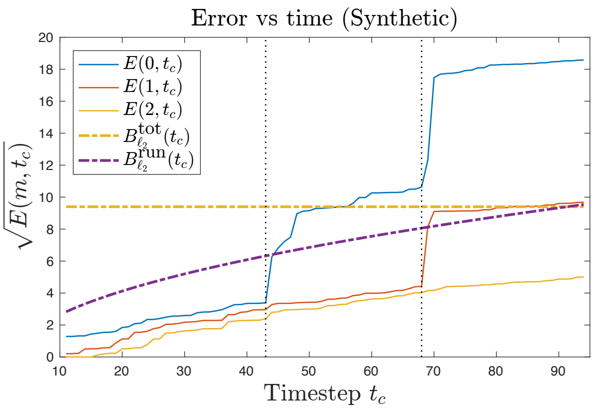


Fig. 5. Error for Example 3; switch detection bounds. True switches are displayed as vertical lines.

$$y_t = \begin{cases} [0.8,\ 0.9,\ -0.9,\ 0.8,\ 0.2]\, r_t + \eta_t & t < 43 \\ -[0.7,\ 0.1,\ 0.5,\ -0.5,\ 0.8]\, r_t + \eta_t & 43 \le t < 68 \\ -[0.3,\ 0.3,\ 0.2,\ 0.5,\ 0.3]\, r_t + \eta_t & 68 \le t \le 100 \end{cases}$$

where $r_t = [y_{t-1}, y_{t-2}, y_{t-3}, u_{t-1}, u_{t-2}]^\top$. Both the input and noise are randomly selected according to a uniform distribution $u_t \sim U[-1,1]$ and $\eta_t \sim U[-1,1]$, respectively.

We show in Figure 5 the $E(m_{t_c}, t_c)$ generated by the incremental algorithm. If our algorithm is using a particular bound, it detects switches at the time step where that bound intersects with $E(m_{t_c}, t_c)$. In this example, we see that $B_{\ell_2}^{\mathrm{run}}(t_c)$ is tighter than $B_{\ell_2}^{\mathrm{tot}}(t_c)$. This is expected, since the more information each bound leverages, the tighter it should be. In particular, we note that for this data stream, $B_{\ell_2}^{\mathrm{run}}(t_c)$ can detect each of the switches within $1-2$ time steps of them occurring, and additionally, a switch will go undetected only if it increases the error by less than $\approx$ 2 to 3. $B_{\ell_2}^{\mathrm{tot}}(t_c)$ takes longer to detect switches and a switch can incur a higher error without being detected, but we note that even with this bound, both switches are eventually detected.

Example 4: Demonstration on real data We demonstrate the performance of $B_{\ell_2}^{\mathrm{tot}}(t_c)$ and $B_{\ell_2}^{\mathrm{run}}(t_c)$ on a real dataset: noisy 4-dimensional friction sensor measurements of a car tire rolling on various materials from Erdogan et al. (2010). As we were given no information on the sensor's noise profile, we estimated ϵ_{t_c} as the average variance of the signal in the first 25 time steps.

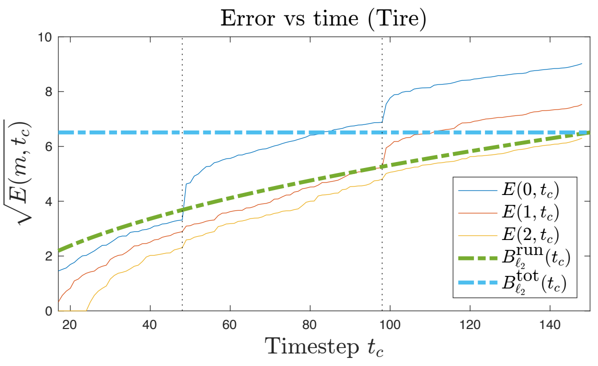


Fig. 6. Error for Example 4; switch detection bounds. True switches are displayed as vertical lines.

We show in Figure 6 the result of switch detection on the tire data; the two true switches are both detected by the bounds. Using $B_{\ell_2}^{\mathrm{run}}(t_c)$ enables the algorithm to detect switches within 1 time step of their occurrence, and a switch will go undetected only if it increases the error by less than ≈ 0.5. Again, $B_{\ell_2}^{\mathrm{tot}}(t_c)$ takes longer to detect switches, but both switches are still detected.

5. CONCLUSION

In this paper, we considered the problem of incrementally identifying time-varying ARX systems from an input/output data stream that is received sequentially in real time. We generalize a dynamic programming approach previously proposed in Ozay (2016) to the incremental setting, analyze its computational complexity, and provide several error bounds that enable the algorithm to detect switch-points in the data at run-time and we show that these error bounds can work well in practice. These error bounds provide necessary conditions on the existence of switches and provide a lower bound on the number of switches. Moreover, our simulation results demonstrate computational gains obtained by the incremental algorithm. We leave to future work a full probabilistic analysis of our incremental algorithm when some statistics on the noise are known, which will facilitate an interpretation based on sequential hypothesis testing. Research currently under way seeks to apply the algorithm for robotics applications and to adapt the incremental algorithm to the setting in which the model order is also unknown.

Acknowledgments: We thank Dr. Gürkan Erdoğan and Prof. Francesco Borrelli for providing the tire data.

REFERENCES

Åstrom, K.J. and Wittenmark, B. (1994). *Adaptive Control.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition.

Bellman, R. and Roth, R. (1969). Curve fitting by segmented straight lines. *Journal of the American Statistical Association*, 64(327), 1079–1084.

Bodenstein, G. and Praetorius, H.M. (1977). Feature extraction from the electroencephalogram by adaptive segmentation. *Proceedings of the IEEE*, 65(5), 642–652.

Erdogan, G., Borrelli, F., Tebano, R., Audisio, G., Lori, G., and Sannazzaro, J. (2010). Development of a new lateral stability control system enhanced with accelerometer based tire sensors. In *DSCC 2010*, 44182, 841–848.

Gersch, W. (1970). Spectral analysis of eeg's by autoregressive decomposition of time series. *Mathematical Biosciences*, 7(1), 205 – 222.

Guo, T., Yan, Z., and Aberer, K. (2012). An adaptive approach for online segmentation of multi-dimensional mobile data. MobiDE '12, 7–14.

Gustafsson, F. (2000). Adaptive filtering and change detection.

Keogh, E., Chu, S., Hart, D., and Pazzani, M. (2001). An online algorithm for segmenting time series. In *Proceedings 2001 IEEE ICDM*, 289–296.

Li, A., He, S., and Zheng, Q. (2003). Real-time segmenting time series data. In *APWeb 2003*, 178–186.

Ljung, L. (ed.) (1999). *System Identification: Theory for the User*. Prentice Hall, Upper Saddle River, NJ, USA.

Mithal, V., O'Connor, Z., Steinhaeuser, K., Boriah, S., Kumar, V., Potter, C.S., and Klooster, S.A. (2012). Time series change detection using segmentation: A case study for land cover monitoring. In *2012 CIDU*, 63–70.

Ohlsson, H., Ljung, L., and Boyd, S.P. (2010). Segmentation of arx-models using sum-of-norms regularization. *Automatica*, 46(6), 1107–1111.

Omranian, N., Mueller-Roeber, B., and Nikoloski, Z. (2015). Segmentation of biological multivariate time-series data. *Scientific Reports*, 5, 8937 EP –. Article.

Ozay, N. (2016). An exact and efficient algorithm for segmentation of ARX models. In *ACC 2016*, 38–41.

Ozay, N., Sznaier, M., and Camps, O.I. (2008a). Sequential sparsification for change detection. In *CVPR 2008*.

Ozay, N., Sznaier, M., Lagoa, C.M., and Camps, O.I. (2008b). A sparsification approach to set membership identification of a class of affine hybrid systems. In *CDC 2008*, 123–130.

Ozay, N., Sznaier, M., Lagoa, C.M., and Camps, O.I. (2012). A sparsification approach to set membership identification of switched affine systems. *IEEE Trans. Automat. Contr.*, 57(3), 634–648.

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

Qi, J., Zhang, R., Ramamohanarao, K., Wang, H., Wen, Z., and Wu, D. (2015). Indexable online time series segmentation with error bound guarantee. *World Wide Web*, 18(2), 359–401.

Vidal, R. (2008). Recursive identification of switched ARX systems. *Automatica*, 44(9), 2274–2287.

Yin, J., Si, Y.W., and Gong, Z. (2011). Financial time series segmentation based on turning points. In *Proceedings 2011 ICSSE*, 394–399.