Online Topology Estimation for Vector Autoregressive Processes in Data Networks

Bakht Zaman, Luis M. Lopez-Ramos, Daniel Romero, and Baltasar Beferull-Lozano

***Abstract*— An important problem in data sciences pertains to inferring causal interactions among a collection of time series. Upon modeling these as a vector autoregressive (VAR) process, this paper deals with estimating the model parameters to identify the underlying causality graph. To exploit the sparse connectivity of causality graphs, the proposed estimators minimize a group-Lasso regularized functional. To cope with real-time applications, big data setups, and possibly time-varying topologies, two online algorithms are presented to recover the sparse coefficients when observations are received sequentially. The proposed algorithms are inspired by the classic recursive least squares (RLS) algorithm and offer complementary benefits in terms of computational efficiency. Numerical results showcase the merits of the proposed schemes in both estimation and prediction tasks.**

A.1 Introduction

Network data analysis emerges naturally in a plethora of applications such as wireless sensor networks, transportation, social, and biological networks, to name a few. A prominent task in this context is inferring graphs that provide the causal relations among a collection of time series such as those encountered in econometrics and sensor data analysis. Identifying these causal interactions is a central problem in many disciplines such as neuroscience, econometrics, bio-informatics, meteorology. Revealing these interactions facilitates tasks such as prediction of time series and data completion.

The problem of inferring graphs capturing dependencies among variables has recently received a great attention in the literature. The simplest approach is to place an edge between two vertices if the sample correlation between the associated variables exceeds a threshold [13]. However, such an approach cannot distinguish mediated from unmediated interactions, thus motivating the methods of partial correlations [13], [14]. Since these methods are still unable to determine directionality in the dependencies, Granger proposed a means to infer the direction of causation by building upon the principle that the cause precedes the effect [28]. An alternative notion of interaction is adopted in the literature of structural equation models by incorporating the influence of exogenous variables; see e.g. [25],[26] and references therein. Unfortunately, these models do not generally capture the temporal structure present in time series. Further approaches for topology identification include [29, 30, 31] though their batch nature cannot track temporal changes in the topology.

The goal of this paper is to track the temporal dynamics of causal relations among time series associated with different variables. To this end, the framework of vector autoregressive (VAR) processes is invoked. These processes are extensively adopted to model linear dependencies among time series [43]. In a P-th order VAR model, the

current data are a noisy superposition of the data at the P previous time instants. The parameters of the VAR model reveal the topology of the causality graph, which motivates their estimation. An estimator based on minimizing a convex criterion regularized by a group-Lasso penalty is presented in [66] to estimate VAR parameters and hence the graph topology. This approach relies on the assumption that the connectivity is sparse, in the sense that the number of edges is small.

When the samples of the time series become available one by one, or when the size of the data challenges the available processing and memory capabilities, online estimation of the model parameters offers a great advantage compared to batch approaches as presented in [97], [98]. Online estimation is also advantageous when the data model is time-varying. Some authors have addressed estimation of time-varying AR models [89, 90, 91]. However, to the best of our knowledge, no online approach for tracking VAR parameters, and thus the associated network topology, has been considered in the literature.

This paper proposes two online estimators for the parameters of a VAR signal model to track the topology of the causality graph. Sparse estimates, where each time series is influenced by a small number of other time series, are enforced by means of a group-Lasso regularized objective in [66]. The first algorithm applies the approximate recursive least squares (RLS) approach in [99], whereas the second solves an optimization problem at each time instant by means of block coordinate descent (BCD). The complexity of these algorithms grows at different rates with the problem size (P and the number of time series), so that their benefits are complementary depending on the specific problem setting.

The contribution of this paper is twofold:

- Online estimation of the group-sparse parameters of VAR process by means of two algorithms with different orders of computational complexity.
- A performance comparison of the different approaches through numerical simulations.

The remainder of this paper is structured as follows: Section A.2 introduces the model and formulates the problem, and the proposed algorithms are presented in Section B.2.2. Section A.4 provides numerical tests and wraps up the paper.

A.2 Model and problem formulation

Consider a collection of N time series, where $f_n[t]$, $t = 0, 1, \ldots, T-1$, denotes the value of the n-th time series at time t. The goal is to determine a directed graph $\mathcal{G} \triangleq (\mathscr{V}, \mathscr{E})$, where $\mathscr{V} = \{1, \ldots, N\}$ is the vertex set and $\mathscr{E} \subset \mathscr{V} \times \mathscr{V}$ is the edge set capturing the causation relations among time series. Specifically, $(n, n') \in \mathscr{E}$ iff $f_{n'}[t]$ causes $f_n[t+\tau]$ for some $\tau \in [1, P]$. To this end, the VAR model is adopted, which prescribes that

$$\boldsymbol{f}[t] \triangleq [f_1[t], \ldots, f_N[t]]^\top = \boldsymbol{u}[t] + \sum_{p=1}^{P} \mathbf{A}_p \boldsymbol{f}[t-p], \tag{A.1}$$

where $\boldsymbol{u}[t] \triangleq \left[u_1[t], u_2[t], ..., u_N[t]\right]^\top$ denotes noise and $\boldsymbol{A}_p \in \mathbb{R}^{N\times N}, p = 1, \ldots, P$, are the VAR parameters. From this expression, it follows that

$$f_n[t] = u_n[t] + \sum_{n'=1}^{N}\sum_{p=1}^{P} a_{n,n'}^{(p)} f_{n'}[t-p], \quad n = 1...N. \tag{A.2}$$

Then, if $f_{n'}[t]$ does not cause $f_n[t+\tau]$ for any $\tau \in [1, P]$, it holds that $a_{n,n'}^{(p)} = 0\ \forall p$, where $a_{n,n'}^{(p)}$ stands for the (n, n')-th entry of $\boldsymbol{A}_p$. This implies that (A.2) can be equivalently expressed as

$$f_n[t] = u_n[t] + \sum_{n':(n,n')\in\mathscr{E}}\sum_{p=1}^{P} a_{n,n'}^{(p)} f_{n'}[t-p], n = 1\ldots N. \tag{A.3}$$

Therefore, one can trivially obtain $\mathscr{E}$, and consequently $\mathcal{G}$, if $\{\boldsymbol{A}_p\}_{p=1}^P$ are known. From (E.2), it follows that $f_n[t]$ is the result of filtering the neighboring time series through a linear time-invariant (LTI) filter and adding these filtered signals together with noise. One can therefore interpret a sparse VAR model in terms of a graph whose edges correspond to an LTI filter between the adjacent vertices. The problem of topology identification reduces therefore to estimating $\{\boldsymbol{A}_p\}_{p=1}^P$ given $\{\boldsymbol{f}[t]\}_{t=0}^{T-1}$.

A.3 Online topology identification

After presenting the estimation criterion in Sec. A.3.0.1, this section describes the proposed algorithms in Secs. A.3.1 and A.3.2.

A.3.0.1 Estimation criterion

A natural approach to estimate $\{\boldsymbol{A}_p\}_{p=1}^P$ is to minimize the following objective:

$$\underset{\{\boldsymbol{A}_p\}_{p=1}^P}{\arg\min}\ \mathcal{L}\left(\{\boldsymbol{A}_p\}_{p=1}^P\right) + \lambda \sum_{n=1}^{N}\sum_{\substack{n'=1\\ n'\neq n}}^{N} \mathbb{1}\Big\{\sum_{p=1}^{P}|a_{n,n'}^{(p)}|\Big\}, \tag{A.4}$$

where $\mathcal{L}(\cdot)$ is given by

$$\begin{aligned}\mathcal{L}\left(\{\boldsymbol{A}_p\}_{p=1}^P\right) &\triangleq \sum_{\tau=P}^{T-1}\left\|\boldsymbol{f}[\tau] - \sum_{p=1}^{P}\boldsymbol{A}_p\boldsymbol{f}[\tau-p]\right\|_2^2 \\ &= \sum_{n=1}^{N}\sum_{\tau=P}^{T-1}\left(f_n[\tau] - \sum_{n'=1}^{N}\sum_{p=1}^{P} a_{n,n'}^{(p)} f_{n'}[\tau-p]\right)^2\end{aligned}$$

and it is a quadratic empirical loss function promoting data fit; and $\mathbb{1}$ is an indicator function satisfying $\mathbb{1}\{x\} = 0$ if $x = 0$ and $\mathbb{1}\{x\} = 1$ if $x \neq 0$. The second term in (A.4) equals the cardinality of $\mathscr{E}$, i.e., the number of edges, times the regularization parameter $\lambda > 0$; and therefore promotes a group-sparse structure in $\{\boldsymbol{A}_p\}_{p=1}^P$ to exploit thus the prior information that the number of edges in $\mathscr{E}$ is small. Self-connections are

not regularized. The parameter λ controls the tradeoff between the data fit and sparsity, and can be adjusted e.g. via cross-validation [11].

For notational convenience, let us introduce the variables $\boldsymbol{a}_{n,n'} \triangleq [a_{n,n'}^{(1)}, a_{n,n'}^{(2)}, ..., a_{n,n'}^{(P)}]^\top \in \mathbb{R}^P$, $\boldsymbol{a}_n \triangleq [\boldsymbol{a}_{n,1}^\top, \boldsymbol{a}_{n,2}^\top, ..., \boldsymbol{a}_{n,N}^\top]^\top \in \mathbb{R}^{NP}$, as well as

$$\boldsymbol{g}[\tau] \triangleq \mathrm{vec}\big(\big[\boldsymbol{f}[\tau-1], \ldots, \boldsymbol{f}[\tau-P]\big]^\top\big) \in \mathbb{R}^{NP}. \tag{A.5}$$

Then, $\mathcal{L}\left(\{\boldsymbol{A}_p\}_{p=1}^P\right)$ can be expressed as $\sum_{n=1}^N \mathcal{L}^{(n)}(\boldsymbol{a}_n)$, where

$$\mathcal{L}^{(n)}(\boldsymbol{a}_n) \triangleq \sum_{\tau=P}^{T} \left(f_n[\tau] - \boldsymbol{g}^\top[\tau]\boldsymbol{a}_n\right)^2. \tag{A.6}$$

With this notation, (A.4) can be expressed as

$$\{\hat{\boldsymbol{a}}_n\}_{n=1}^N = \underset{\{\boldsymbol{a}_n\}_{n=1}^N}{\arg\min} \sum_{n=1}^N \left[\mathcal{L}^{(n)}(\boldsymbol{a}_n) + \lambda \sum_{\substack{n'=1 \\ n'\neq n}}^N \mathbb{1}\{\|\boldsymbol{a}_{n,n'}\|_2\}\right].$$

Since the above problem is non-convex, [66] proposed recovering sparse coefficients by minimizing the following group-Lasso regularized functional

$$\{\hat{\boldsymbol{a}}_n\}_{n=1}^N = \underset{\{\boldsymbol{a}_n\}_{n=1}^N}{\arg\min} \sum_{n=1}^N \left[\mathcal{L}^{(n)}(\boldsymbol{a}_n) + \lambda \sum_{\substack{n'=1 \\ n'\neq n}}^N \|\boldsymbol{a}_{n,n'}\|_2\right] \tag{A.7}$$

which clearly separates across $\boldsymbol{a}_n$ as

$$\hat{\boldsymbol{a}}_n = \underset{\boldsymbol{a}_n}{\arg\min}\ \mathcal{L}^{(n)}(\boldsymbol{a}_n) + \lambda \sum_{\substack{n'=1 \\ n'\neq n}}^N \|\boldsymbol{a}_{n,n'}\|_2. \tag{A.8}$$

The batch estimation criterion in (A.8) requires all data $\{\boldsymbol{f}[t]\}_{t=0}^{T-1}$ before an estimate can be obtained. The rest of this section proposes an online criterion that provides an estimate per each time-slot when new data is received, and furthermore enables tracking topology changes. To this end, the objective in (A.8) is replaced with a time-dependent objective as follows:

$$\hat{\boldsymbol{a}}_n[t] = \underset{\boldsymbol{a}_n[t]}{\arg\min}\ \mathcal{L}^{(n)}(\boldsymbol{a}_n[t], t) + \lambda \sum_{\substack{n'=1 \\ n'\neq n}}^N \|\boldsymbol{a}_{n,n'}[t]\|_2, \tag{A.9}$$

where $\hat{\boldsymbol{a}}_n[t]$ is the estimate of $\boldsymbol{a}_n$ at time t,

$$\boldsymbol{a}_n[t] \triangleq [\boldsymbol{a}_{n,1}^\top[t], \boldsymbol{a}_{n,2}^\top[t], ..., \boldsymbol{a}_{n,N}^\top[t]]^\top \in \mathbb{R}^{NP} \tag{A.10}$$

contains the optimization variables at time t, and

$$\mathcal{L}^{(n)}(\boldsymbol{a}_n[t], t) \triangleq \sum_{\tau=P}^{t} \gamma^{t-\tau} \left(f_n[\tau] - \boldsymbol{g}^\top[\tau]\boldsymbol{a}_n[t]\right)^2 \tag{A.11}$$

is a time-dependent version of the empirical loss function in (A.6), where $0 < \gamma \leq 1$ is a user-selected forgetting factor. The latter weights recent samples more heavily than the older ones and is introduced to facilitate tracking topology changes. Observe that (A.11), and therefore (A.9), only depend on data up to time t, and therefore $\hat{\boldsymbol{a}}_n[t]$ can be obtained right after $\{\boldsymbol{f}[\tau]\}_{\tau=0}^{t}$ have been received. The rest of this section proposes two algorithms to solve (A.9) in an online fashion.

A.3.1 Regularized RLS (R-RLS)

A solver based on RLS is proposed in this section. To this end, consider the following valid subgradient of the (non differentiable) regularization term in (A.9)

$$\boldsymbol{h}(\boldsymbol{a}_n[t]) \triangleq [\nabla^{s\top}_{\boldsymbol{a}_{n,1}[t]} \|\boldsymbol{a}_{n,1}[t]\|_2, \ldots, \nabla^{s\top}_{\boldsymbol{a}_{n,n-1}[t]} \|\boldsymbol{a}_{n,n-1}[t]\|_2, \boldsymbol{0}, \nabla^{s\top}_{\boldsymbol{a}_{n,n+1}[t]} \|\boldsymbol{a}_{n,n+1}[t]\|_2, \ldots, \nabla^{s\top}_{\boldsymbol{a}_{n,N}[t]} \|\boldsymbol{a}_{n,N}[t]\|_2]^\top, \quad \text{(A.12)}$$

where

$$\nabla^{s}_{\boldsymbol{x}} \|\boldsymbol{x}\|_2 = \begin{cases} \frac{\boldsymbol{x}}{\|\boldsymbol{x}\|_2}, & \boldsymbol{x} \neq \boldsymbol{0} \\ \boldsymbol{0}, & \boldsymbol{x} = \boldsymbol{0}. \end{cases} \quad \text{(A.13)}$$

On the other hand, let

$$\boldsymbol{\Phi}[t] \triangleq \sum_{\tau=P}^{t} \gamma^{t-\tau} \boldsymbol{g}[\tau]\boldsymbol{g}^\top[\tau], \quad \text{(A.14)}$$

$$\boldsymbol{r}_n[t] \triangleq \sum_{\tau=P}^{t} \gamma^{t-\tau} f_n[\tau]\boldsymbol{g}[\tau], \quad \text{(A.15)}$$

respectively denote a weighted sample auto-correlation matrix of $\boldsymbol{g}[\tau]$ and a weighted sample cross-correlation of $f_n[\tau]$ and $\boldsymbol{g}[\tau]$. Note that $\boldsymbol{\Phi}[t]$ and $\boldsymbol{r}_n[t]$ can be updated recursively as

$$\boldsymbol{\Phi}[t] = \gamma\boldsymbol{\Phi}[t-1] + \boldsymbol{g}[t]\boldsymbol{g}^\top[t], \quad \text{(A.16)}$$

$$\boldsymbol{r}_n[t] = \gamma\boldsymbol{r}_n[t-1] + f_n[t]\boldsymbol{g}[t]. \quad \text{(A.17)}$$

In view of these equations, it can be shown that the algorithm in [99] reduces to **Algorithm 6** when solving (A.9). This algorithm offers an approximate solution since it relies on the assumption that the estimated coefficients do not change abruptly between consecutive time steps.

The complexity of **Algorithm 6** is dominated by the N computations of $\boldsymbol{Q}[t]\boldsymbol{h}(\hat{\boldsymbol{a}}_n[t-1])$ (which are $\mathcal{O}(N^2P^2)$), and therefore the overall complexity is $\mathcal{O}(N^3P^2)$.

A.3.2 Online Block Coordinate Descent (OBCD)

When N is very large, the computational burden of **Algorithm 6** can become prohibitive given its cubic-order complexity with respect to N. To alleviate this limitation, this section proposes an online method with quadratic complexity in N. The proposed method is based on performing a single iteration of BCD to minimize (A.9). A related approach

Algorithm 6 Group-sparse R-RLS algorithm

Input: $\sigma, P, \lambda, \gamma, \{\boldsymbol{f}[\tau]\}_{\tau=0}^{t}$
Output: $\{\hat{\boldsymbol{a}}_n[t]\}_{n=1}^{N}$
Initialization: $\hat{\boldsymbol{a}}_n[P-1] = \boldsymbol{0}, \boldsymbol{Q}[P-1] = \sigma^{-1}\boldsymbol{I}$

1: **for** $t = P, P+1, \ldots,$ **do**
2: $\quad \boldsymbol{k}[t] = \dfrac{\boldsymbol{Q}[t-1]\boldsymbol{g}[t]}{\gamma + \boldsymbol{g}^{\top}[t]\boldsymbol{Q}[t-1]\boldsymbol{g}[t]}$
3: $\quad \boldsymbol{Q}[t] = \gamma^{-1}\boldsymbol{Q}[t-1] - \gamma^{-1}\boldsymbol{k}[t]\boldsymbol{g}^{\top}[t]\boldsymbol{Q}[t-1]$
4: $\quad$ **for** $n = 1, 2, \ldots, N$ **do**
5: $\quad\quad e_n[t] = f_n[t] - \boldsymbol{g}^{\top}[t]\hat{\boldsymbol{a}}_n[t-1]$
6: $\quad\quad \hat{\boldsymbol{a}}_n[t] = \hat{\boldsymbol{a}}_n[t-1] + e_n[t]\boldsymbol{k}[t] + \lambda(\gamma-1)\boldsymbol{Q}[t]\boldsymbol{h}(\hat{\boldsymbol{a}}_n[t-1])$
7: $\quad$ **end for**
8: **end for**

for solving batch group-Lasso problems was proposed in [100]. Note that although (A.9) can be solved directly by off-the-shelf convex optimization solvers, their complexity is high and therefore an algorithm tailored to (A.9) is preferable.

Block coordinate descent is based on iteratively minimizing a given objective with respect to a group of variables while keeping the rest of groups fixed to their values in previous iterations. Fortunately, at each minimization step the function is differentiable in all points except the zero, while the minimization step at the zero vector is simple to be performed.

The right-hand side of (A.9) can be rewritten in terms of the recursively computed $\boldsymbol{\Phi}[t]$ and $\boldsymbol{r}_n[t]$ as

$$\arg\min_{\boldsymbol{a}_n[t]} \boldsymbol{a}_n^{\top}[t]\boldsymbol{\Phi}[t]\boldsymbol{a}_n[t] - 2\mathbf{r}_n^{\top}[t]\boldsymbol{a}_n[t] + \lambda\sum_{\substack{n'=1\\ n'\neq n}}^{N} \|\boldsymbol{a}_{n,n'}[t]\|_2$$

For each t and n, the proposed algorithm performs N block updates: the i-th update modifies the i-th group $\boldsymbol{a}_{n,i}[t]$ whereas all other entries in $\boldsymbol{a}_n[t]$ are kept fixed.

Upon appropriately permuting the entries of $\boldsymbol{a}_n[t], \boldsymbol{\Phi}[t]$, and $\boldsymbol{r}_n[t]$, the minimization of the above objective with respect to the i-th group can be expressed as

$$\hat{\boldsymbol{a}}_{n,i}[t] = \arg\min_{\boldsymbol{a}_{n,i}[t]} \begin{bmatrix}\boldsymbol{a}_{n,\bar{i}}^{\top}[t] & \boldsymbol{a}_{n,i}^{\top}[t]\end{bmatrix} \begin{bmatrix}\boldsymbol{\Phi}_{\bar{i}\bar{i}}[t] & \boldsymbol{\Phi}_{\bar{i}i}[t]\\ \boldsymbol{\Phi}_{i\bar{i}}[t] & \boldsymbol{\Phi}_{ii}[t]\end{bmatrix} \begin{bmatrix}\boldsymbol{a}_{n,\bar{i}}[t]\\ \boldsymbol{a}_{n,i}[t]\end{bmatrix} - 2\begin{bmatrix}\mathbf{r}_{n,i}^{\top}[t] & \mathbf{r}_{n,\bar{i}}^{\top}[t]\end{bmatrix} \begin{bmatrix}\boldsymbol{a}_{n,\bar{i}}[t]\\ \boldsymbol{a}_{n,i}[t]\end{bmatrix} + \lambda\Big(\sum_{n'=1}^{N} \|\boldsymbol{a}_{n,n'}[t]\|_2\Big)\mathbb{1}\{i-n\} \quad \text{(A.18)}$$

where $\hat{\boldsymbol{a}}_{n,i}[t]$ collects the entries of the i-th group in $\hat{\boldsymbol{a}}_n[t]$; $\boldsymbol{a}_{n,\bar{i}}[t]$ collects the entries in the complementary set of i-th group; and similar definitions apply for $\mathbf{r}_{n,\bar{i}}[t]$, $\boldsymbol{\Phi}_{\bar{i}\bar{i}}[t]$, and $\boldsymbol{\Phi}_{i\bar{i}}[t]$ $(= \boldsymbol{\Phi}_{\bar{i}i}^{\top}[t]$ because of symmetry). Ignoring the constant terms, the right-hand side of (A.18) can be rewritten as

$$\hat{\boldsymbol{a}}_{n,i}[t] = \arg\min_{\boldsymbol{a}_{n,i}[t]} \boldsymbol{a}_{n,i}^{\top}[t]\boldsymbol{\Phi}_{ii}[t]\boldsymbol{a}_{n,i}[t] + 2(\boldsymbol{\Phi}_{i\bar{i}}[t]\boldsymbol{a}_{n,\bar{i}}[t] - \mathbf{r}_{n,i}[t])^{\top}\boldsymbol{a}_{n,i}[t] + \lambda\|\boldsymbol{a}_{n,i}[t]\|_2 \quad \text{(A.19)}$$

When $i = n$, the last term is zero and therefore (A.19) constitutes a conventional least-squares equation and its solution is $\hat{\boldsymbol{a}}_{n,n}[t] = \boldsymbol{\Phi}_{nn}^{\dagger}\boldsymbol{p}_n$, where $\boldsymbol{M}_i \triangleq \boldsymbol{\Phi}_{ii}$ and $\boldsymbol{p}_i \triangleq \boldsymbol{\Phi}_{i\bar{i}}\boldsymbol{a}_{n,\bar{i}} - \mathbf{r}_{n,i}^{\top}[t]$.

Conversely, when $i \neq n$, we solve the optimization using Newton's method. This requires the cost function to be twice differentiable at every point. In our case, this holds at every point except the zero vector; fortunately, when solving (A.19) this case can be circumvented. Note first that it can be proven [100] that $\mathbf{0}$ will be an optimizer of (A.19) iff $\|\mathbf{p}\|_2 \leq \lambda$. Second, if Newton's method is initialized at an $\boldsymbol{a}^{(0)}$ that yields a negative objective and every iteration effectively reduces the objective, then the optimization is done over a sub-level set where the gradient and the Hessian are always well defined.

Consequently, the proposed solver first checks if $\mathbf{0}$ is the optimal solution to (A.19), and if it is not, $\boldsymbol{a}^{(0)}$ is initialized as $\boldsymbol{a}^{(0)} = \left((\lambda \|\mathbf{p}\|_2 - \|\mathbf{p}\|_2^2)/(\mathbf{p}^{\top}\mathbf{M}\mathbf{p})\right)\mathbf{p}$ which is the solution to a line search over the half line that starts at $\mathbf{0}$ in the steepest descent direction. Afterwards, standard Newton iterations are performed until convergence as detailed in **Algorithm 8**.

Algorithm 7 further generalizes [101, Algorithm 3] which can only accommodate groups of size 1 (regular Lasso). Regarding complexity, **Algorithm 8** is called $N(N-1)$ times and its complexity is dominated by the inversion of the $P \times P$ Hessian. Consequently, OBCD entails a complexity of $\mathcal{O}(N^2P^3)$ per time instant.

Algorithm 7 Online Block Coordinate Descent

Input: $\lambda, \gamma, \sigma, \{\boldsymbol{f}[\tau]\}_{\tau=0}^{t}$,
Output: $\{\hat{\boldsymbol{a}}_n[t]\}_{n=1}^{N}$
Initialization: $\hat{\boldsymbol{a}}_n[P-1] = \mathbf{0}, \boldsymbol{\Phi}[P-1] = \sigma^2\boldsymbol{I}, \mathbf{r}_n[P-1] = \mathbf{0}$, and $\mathbf{g}[P-1]$ as in (B.4)

for $t = P, P+1, \ldots,$ **do**
  Obtain $\boldsymbol{\Phi}[t]$ as in (A.16)
  for $n = 1, 2, \ldots, N$ **do**
    for $i = 1, 2, \ldots, N$ **do**
      Obtain $\mathbf{r}_n[t]$ as in (A.17)
      Set $\boldsymbol{a}_{n,j}[t] = \boldsymbol{a}_{n,j}[t-1] \forall j \neq i$
      Update $\boldsymbol{a}_{n,i}[t]$ via (A.19)
    end for
  end for
end for
end for

A.4 Numerical Experiments

The performance of the proposed online algorithms is compared with the batch group-Lasso approach by numerical tests in this section. A network is simulated by a random graph with $N = 15$ nodes, and an edge set randomly generated by an Erdos-Renyi model with edge probability p_e constant for every pair of nodes except for self-loops, which have

Algorithm 8 Solve (A.19) via Newton's method

Input: $\boldsymbol{\Phi}_{ii}[t], \boldsymbol{\Phi}_{i\bar{i}}[t], \lambda, \mathbf{r}_{n,i}[t], \boldsymbol{a}_{n,\bar{i}}[t]$ **Output:** $\hat{\boldsymbol{a}}_{n,i}[t]$

$\mathbf{M} = \boldsymbol{\Phi}_{ii}[t]$; $\mathbf{p} = \boldsymbol{\Phi}_{i\bar{i}}[t]\boldsymbol{a}_{n,\bar{i}} - \mathbf{r}_{n,i}[t]$

if $\|\mathbf{p}\|_2 \leq \lambda$ **then return** $\hat{\boldsymbol{a}}_n[t] = \mathbf{0}$

else $\boldsymbol{a}^{(0)} = \left((\lambda \|\mathbf{p}\|_2 - \|\mathbf{p}\|_2^2)/(\mathbf{p}^\top \mathbf{M}\mathbf{p})\right)\mathbf{p}$

for $k = 0, 1, \ldots$ until convergence **do**

$$\mathbf{H} = \mathbf{M} + \lambda \left(\frac{\mathbf{I}}{\|\boldsymbol{a}^{(k)}\|_2} - \frac{\boldsymbol{a}^{(k)}\boldsymbol{a}^{\top(k)}}{\|\boldsymbol{a}^{(k)}\|_2^3} \right)$$

$$\mathbf{g} = \mathbf{M}\boldsymbol{a}^{(k)} + \mathbf{p} + \frac{\lambda \boldsymbol{a}^{(k)}}{\|\boldsymbol{a}^{(k)}\|_2}$$

$\boldsymbol{a}^{(k+1)} = \boldsymbol{a}^{(k)} - \mathbf{H}^\dagger \mathbf{g}$

end forreturn $\hat{\boldsymbol{a}}_{n,i}[t] = \boldsymbol{a}^{(k)}$

edge probability one. A VAR process with order $P = 5$ is generated by drawing the active coefficients of $\boldsymbol{A}_p$ from a Gaussian distribution, setting the rest of the coefficients to zero, and normalizing the result so that the largest-magnitude eigenvalue of $\boldsymbol{A}_p$ is less than $1/P$, thus guaranteeing a stable VAR process. A time series of T time instants is generated according to (E.1) with $\boldsymbol{u} \sim \mathcal{N}(\mathbf{0}, 0.02\boldsymbol{I})$. The regularization parameter is chosen as $\lambda = 0.02$.

Two error measures are used to compare the performance of the developed methods. In the first case, the estimated VAR coefficients $\{\hat{\boldsymbol{a}}_n[t]\}$ are directly compared to the true coefficients and the evolution of the normalized mean squared deviation (NMSD) defined as $\mathbb{E}[\|\sum_n(\hat{\boldsymbol{a}}_n[t] - \boldsymbol{a}_n)\|_2^2]/\mathbb{E}[\sum_n\|\boldsymbol{a}_n\|_2^2]$ is represented in the top pane of Fig. A.1. In the second case, the coefficients are used to predict the process in the next time instant and the normalized mean square error (NMSE) is depicted in the bottom pane. To reduce computational burden, the two error measures for the batch approach are evaluated for $T = 50, 100, 150, \ldots, 650$, considering all available data up to time T. The dashed line is added to improve visualization. These results suggest that both algorithms have similar convergence rates and their estimate approaches the batch solution after processing a large number of samples. Although OBCD shows a slight advantage over R-RLS, a main factor to choose one approach or the other is the computational efficiency. As a short wrap-up, recall that OBCD has $\mathcal{O}(N^2P^3)$ computation, and R-RLS has $\mathcal{O}(N^3P^2)$. This makes the former more suitable for large networks, whereas the latter enjoys fast performance for large filter order.

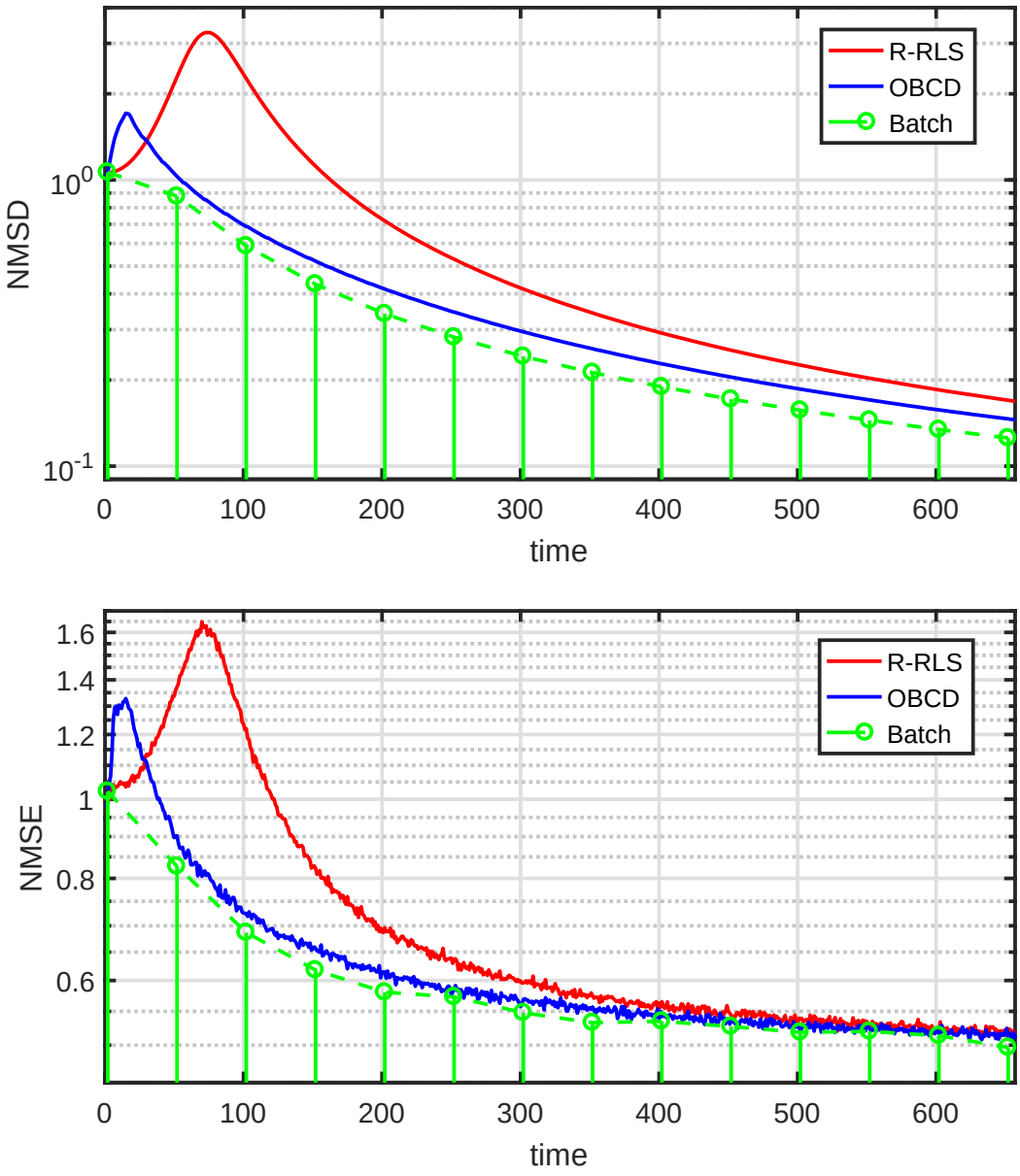

Figure A.1: Normalized Mean Squared Deviation (top) and Normalized Mean Squared Error (bottom).