
Fitting ARMA Time Series Models without Identification: A Proximal Approach

Yin Liu

Department of Integrated Systems Engineering
The Ohio State University, OH, US

Sam Davanloo Tajbakhsh

Abstract

Fitting autoregressive moving average (ARMA) time series models requires model identification before parameter estimation. Model identification involves determining the order of the autoregressive and moving average components which is generally performed by inspection of the autocorrelation and partial autocorrelation functions or other offline methods. In this work, we regularize the parameter estimation optimization problem with a non-smooth hierarchical sparsity-inducing penalty based on two path graphs that allow performing model identification and parameter estimation simultaneously. A proximal block coordinate descent algorithm is then proposed to solve the underlying optimization problem efficiently. The resulting model satisfies the required stationarity and invertibility conditions for ARMA models. Numerical results supporting the proposed method are also presented.

1 INTRODUCTION

ARIMA time series models have a multitude of applications, e.g., in epidemiological surveillance (Zhang et al., 2014), water resource management (Wang et al., 2015), transportation systems (Billings and Yang, 2006), drought forecasting (Han et al., 2010), stock price forecasting (Adebiyi et al., 2014), business planning (Calheiros et al., 2014), and power systems (Chen et al., 2009), to name a few. Even the emergence of deep neural networks and their customized architec-

tures for time series modeling, e.g., Recurrent Neural Nets (RNN) and Long Short-Term Memory (LSTM) has not decreased the popularity of ARIMA models (Makridakis et al., 2018).

Fitting ARMA(p, q) time series models requires a two-step process: 1. Model identification, 2. Parameter estimation. The model identification step determines the order of the autoregressive (AR) component (p) and the moving average (MA) component (q). Next, given the underlying ARMA model, the parameters are estimated by solving an optimization problem for the maximum likelihood or least square estimates (Box et al., 2015; Del Castillo, 2002). We should note that ARMA models are used to model stationary processes; however, there exists a more general class of ARIMA models for *homogeneous nonstationary* processes (which are stationary in the mean). Such processes become stationary after d times differencing; hence, the corresponding ARIMA(p, d, q) model includes differencing of order d . The results of this paper are mainly for stationary processes with potential extension to the homogeneous nonstationary processes.

Model identification is primarily based on visual inspection of the sample autocorrelation function (ACF) and partial autocorrelation (PACF) plots. For the AR(p) process, the sample ACF follows an exponential decay, and the sample PACF cuts off after lag p , while for the MA(q) process, the sample ACF cuts off after lag q and the sample PACF decays exponentially (Del Castillo, 2002). When the process involves both AR and MA components, it is more difficult to identify the correct orders. After model identification, parameters are estimated by minimizing a loss function (e.g., negative log-likelihood or least square). Some works, e.g., Box et al. (2015), recommended an iterative approach between the model identification and parameter estimation which involves inspection of the residuals from the fitted model to make sure that they are indeed white noise.

In many of today’s applications, ARMA models should be fitted to many times series data $\{y_t^j\}_{j=1}^J$ with J be-

ing very large, e.g., the demand data for more than thousands of products. If demand is uncorrelated across different products, fitting vector ARMA models is unnecessary, and separate models are preferable. In such scenarios, model identifications become a significant challenge in the modeling process. This work proposes a novel approach to fit ARMA models that allows automating the fitting procedure by merging the model identification step into the parameter estimation. Indeed, with the aid of a single tuning parameter, the proposed algorithm allows data to identify an appropriate model.

1.1 CONTRIBUTIONS

The contributions of this work are as follows:

- We develop a novel approach to fit ARMA time series models that identifies the model by tuning a single continuous parameter (λ). This approach merges model identification with parameter estimation by introducing a hierarchical sparsity-inducing penalty into the optimization problem. The sparsity-inducing penalty *preserves the hierarchical model structure*, e.g., it does not allow the second AR parameter to be nonzero when the first AR parameter is zero.
- We propose an efficient proximal block coordinate descent (BCD) algorithm to solve the underlying nonsmooth and nonconvex optimization problem to a stationary point – see Algorithm 2. The proximal map of the nonsmooth hierarchical sparsity-inducing penalty is shown to be separable on the AR and MA components.
- The proposed approach automates the ARMA time series modeling, does not require offline model identification and allows ARMA time series modeling for a large number of time series data.

1.2 Related Work

Model identification to determine the order of the time series model through regularization with ℓ_1 -norm (also known as Lasso regularization) is performed for univariate AR models in Wang et al. (2007); Nardi and Rinaldo (2011). Extensions of such methods for vector AR (VAR) models are also considered in Hsu et al. (2008). By smart tuning of the regularization parameter, Ren and Zhang (2010) proposed an adaptive Lasso regularizer for VAR models with provable asymptotic properties – see also Chan and Chen (2011) for an adaptive ARMA model selection. This line of research utilizes ℓ_1 -penalty to induce sparsity in the parameters

of the time series model to select a subset of the model parameters. However, naive usage of ℓ_1 -penalty results in models that lack the hierarchical structure. Hierarchically structured models are those in which higher-order parameters (in both the AR and MA components) are allowed to be nonzero when lower-order parameters are nonzero (as a necessary condition). This is similar to regression modeling where for better interpretability, one prefers to have higher-order interactions in the model only if the lower-level ones are included in the model.

To keep the benefits and simplicity of fitting ARMA models using Lasso-type penalties and also to enforce the desired hierarchical structure in the identified model, few works looked into hierarchical sparsity-inducing penalties for time series modeling. Nicholson et al. (2014) consider a hierarchical lag structure (HLag) for VAR models utilizing the group lasso with nested groups and use an iterative soft-thresholding algorithm to solve the underlying problem. Furthermore, Wilms et al. (2017) consider a vector ARMA model and propose to measure the complexity of the model based on a user-defined strongly convex function that can then be used as a regularizer for model identification. Their parameter estimation is a two-phase process: first, the unobservable errors are estimated by fitting a pure VAR(∞) model; next, the approximate lagged errors are used as the covariates for the MA component which results in a least-square problem regularized with ℓ_1 or HLag penalty.

1.3 Notations

Lowercase boldface letters denote vectors, and uppercase Greek letters denote sets, except for \mathcal{B} which denotes the back-shift operator. The set of all real and complex numbers are denoted by \mathbb{R} and \mathbb{C} , respectively. Given a set $g \subseteq \mathcal{G}$, $|g|$ denotes its cardinality and g^c denotes its complement. Given $\beta \in \mathbb{R}^d$ and $g \subseteq \{1, \dots, d\}$, $\beta_g \in \mathbb{R}^{|g|}$ is a vector with its elements selected from β over the index set g .

2 PROBLEM DEFINITION

We consider a stationary ARMA(p, q) time series process with a zero mean as

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \dots + \phi_p y_{t-p} - \theta_1 \epsilon_{t-1} - \theta_2 \epsilon_{t-2} - \dots - \theta_q \epsilon_{t-q} + \epsilon_t, \quad (1)$$

where ϕ_ℓ , with $\ell = 1, \dots, p$ are the parameters of the AR component, and θ_ℓ , with $\ell = 1, \dots, q$ are the parameters of the MA component, and ϵ_t is a white noise with zero mean and variance σ^2 . The process (1) can

also be written as

$$P_{\phi}^p(\mathcal{B})y_t = P_{\theta}^q(\mathcal{B})\epsilon_t, \quad (2)$$

where \mathcal{B} is the *back-shift operator*, i.e., $\mathcal{B}y_t = y_{t-1}$, and

$$P_{\alpha}^d(z) \triangleq 1 - \alpha_1 z - \alpha_2 z^2 - \dots - \alpha_d z^d, \quad (3)$$

is a polynomial of degree d with the parameter α . The process (2) is *stationary* if the AR component is stationary which is the case if all roots of the $P_{\phi}^p(z)$ polynomial are outside the unit circle; furthermore, the process is *invertible* if the MA component is invertible which is the case if all roots of the $P_{\theta}^d(z)$ polynomial are outside the unit circle (Del Castillo, 2002). Requiring the two polynomials to have roots outside of the unit circle in the \mathcal{B} space translates to some constraints on $\phi = [\phi_1, \dots, \phi_p]^{\top}$ and $\theta = [\theta_1, \dots, \theta_q]^{\top}$, i.e., $\phi \in \mathcal{X}_{\phi}^p \subseteq \mathbb{R}^p$ and $\theta \in \mathcal{X}_{\theta}^q \subseteq \mathbb{R}^q$, where \mathcal{X}_{α}^d is defined as

$$\mathcal{X}_{\alpha}^d \triangleq \{\alpha \in \mathbb{R}^d : \forall z \in \mathbb{C}, P_{\alpha}^d(z) = 0 \Rightarrow |z| > 1\}. \quad (4)$$

We note that there is also another (maybe more common) representation for \mathcal{X}_{α}^d based on the monic polynomial

$$\bar{P}_{\alpha}^d(z) \triangleq z^d + \alpha_1 z^{d-1} + \dots + \alpha_{d-1} z + \alpha_d, \quad (5)$$

of degree d , where it can be shown that

$$\mathcal{X}_{\alpha}^d = \{\alpha \in \mathbb{R}^d : \forall z \in \mathbb{C}, \bar{P}_{\alpha}^d(z) = 0 \Rightarrow |z| < 1\}. \quad (6)$$

Note that the new representation requires roots of the polynomial to be *inside* the unit circle. For an arbitrary d , the geometrical complexity of \mathcal{X}_{α}^d makes projection onto this set very difficult (Combettes and Trussell, 1992). Indeed, Combettes and Trussell (1992) discussed that \mathcal{X}_{α}^d is open, bounded, and *not necessarily convex* – see also Moses and Liu (1991); Blondel et al. (2012). To deal with the openness of \mathcal{X}_{α}^d , it is common to approximate it with a closed set from inside – see (12). However, projection onto this set or its approximation *may not be unique* due to its potential nonconvexity. A method for projection onto the \mathcal{X}_{α}^d set was developed in Moses and Liu (1991). While their scheme is easy to implement, the convergence of this iterative method is slower than the steepest descent method – see also Combettes and Trussell (1992). To conclude, imposing stationarity and invertibility of the model is performed by projecting ϕ and θ onto (inner approximate of) \mathcal{X}_{ϕ}^p and \mathcal{X}_{θ}^q , respectively, which may not be unique.

The above discussion is for an ARMA model that is already identified, i.e., p and q are known. For a model

that is not identified, we also need

- if $\phi_{\ell} = 0$ then $\phi_{\ell'} = 0$, $\forall \ell < \ell'$,
(or equivalently) if $\phi_{\ell'} \neq 0$ then $\phi_{\ell} \neq 0$, $\forall \ell < \ell'$,
- if $\theta_{\ell} = 0$ then $\theta_{\ell'} = 0$, $\forall \ell < \ell'$,
(or equivalently) if $\theta_{\ell'} \neq 0$ then $\theta_{\ell} \neq 0$, $\forall \ell < \ell'$,
(7)

i.e., the sparsity of ϕ and θ follow hierarchical structures.

Before discussing how these sparsity structures are enforced, we will briefly discuss the loss function for fitting ARMA models. Given an identified model, i.e., p and q are known, fitting ARMA models are generally performed by finding the conditional maximum likelihood or conditional least-square estimates, which are close to each other assuming that ϵ_t in (1) follows a Normal distribution and the data size T is reasonably large. The conditional least-square estimate (for an identified model) requires solving

$$\begin{aligned} \min_{\phi, \theta} \mathcal{L}(\phi, \theta) &= \frac{1}{2} \sum_{t=\max\{p,q\}}^T \hat{\epsilon}_t^2 = \frac{1}{2} \sum_{t=\max\{p,q\}}^T (y_t - \hat{y}_{t|t-1}(\phi, \theta))^2 \\ \text{s.t. } &\phi \in \mathcal{X}_{\phi}^p, \quad \theta \in \mathcal{X}_{\theta}^q, \end{aligned} \quad (8)$$

where $\hat{y}_{t|t-1}(\phi, \theta)$ is the model prediction for y_t using the data $\{y_t\}_{t=1}^{t-1}$, and is called *conditional* since it depends on the p initial values for y_t and q initial values for ϵ_t . Note that in the absence of MA terms (i.e., $q = 0$), the objective function of (8) is convex in the parameters of the AR model ϕ . However, if $q > 0$ then the objective function of (8) is also nonconvex, and optimization routines are not guaranteed to converge to the global optimum (Hamilton, 1994; Box et al., 2015; Benidir and Picinbono, 1990; Georgiou and Lindquist, 2008). To sum up, in its most general case, problem (8) involves nonconvex minimization over a nonconvex set and, hence, it is difficult to solve.

This paper intends to provide a solution that preserves the hierarchical sparsity structure and is *not* concerned with the nonconvexities of the objective function and the feasible region. In the next section, we propose a method that allows learning p and q within the parameter estimation step.

3 PROPOSED METHOD

Before discussing the proposed method, we should briefly discuss the notion of *hierarchical sparsity*. Let $D = (\mathcal{S}, \mathcal{E})$ be a Directed Acyclic Graph (DAG) where $\mathcal{S} = \{s_1, \dots, s_n\}$ is the set of graph nodes and \mathcal{E} be the set of ordered pair of nodes denoting edges where each pair denotes an edge from the node in the first

element to the node in the second element. Each s_i is an index set of the parameters of the model such that $s_i \cap s_j = \emptyset$, $\forall (i, j)$ and $\cup_{i=1}^n s_i = \{1, \dots, d\}$ where d is the number of parameters. DAG shows the sparsity structures of interest in the parent/child relationship. Assuming one variable per node, the variable in a child node can only be nonzero if the variable in the parent node is nonzero. For instance, given a parameter $\beta \in \mathbb{R}^3$, the top plot in Figure 1 requires $\beta_1 \neq 0$ if $\beta_2 \neq 0$ ($\beta_2 = 0$ if $\beta_1 = 0$); similarly, $\beta_2 \neq 0$ if $\beta_3 \neq 0$ ($\beta_3 = 0$ if $\beta_2 = 0$). For a DAG that contains more than one variable per node (e.g. the bottom plot in Figure 1), two different hierarchies can be considered: 1. *Strong hierarchy*: the parameters in the child node can only be nonzero if *all* of the parameters in its parent node(s) are nonzero. 2. *Weak hierarchy*: the parameters in the child node can be nonzero if *at least one* of the parameters in its parent node(s) is nonzero (Bien et al., 2013). For more information about hierarchical sparsity structures refer to Zhao et al. (2009); Jenatton et al. (2011b,a); Bach et al. (2012); Yan et al. (2017).

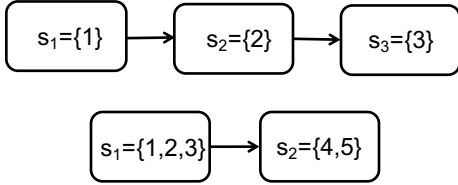


Figure 1: Path graphs showing hierarchical sparsities: **(Top)** A graph with a variable per node for $\beta \in \mathbb{R}^3$. **(Bottom)** A graph with multiple variables per node for $\beta \in \mathbb{R}^5$.

3.1 Hierarchical Sparsity for ARMA Models

In this work, we want to include the model identification of ARMA models in the parameter estimation step. We assume the knowledge about some upper bounds on the true p^* and q^* , i.e., $\bar{p} \geq p^*$ and $\bar{q} \geq q^*$, respectively. Considering ARMA(\bar{p}, \bar{q}), the estimated parameters should satisfy the condition (7). To do so, we define two path graphs as shown in Figure 2. Since this DAG consists of two path graphs and there is only one variable in each node, weak and strong hierarchies are equivalent. Enforcing the sparsity structure shown in Figure 2 *exactly* requires introducing binary variables into the optimization problem (8) and solving a Mixed Integer Program (MIP). For instance, to model the parent/child hierarchy between ϕ_1 and ϕ_2 , one needs to introduce a binary variable $z \in \{0, 1\}$ and two constraints as $z\epsilon \leq |\phi_1|$ and $|\phi_2| \leq z\mu$ for some reasonably small and large parameters ϵ and μ , respectively. Provided that the underlying optimization problem is already very difficult to solve, introducing $\bar{p} + \bar{q} - 2$ binary variable makes the problem even more challenging. Hence, despite the significant

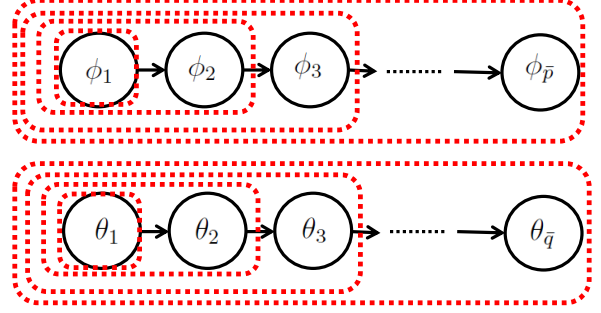


Figure 2: DAG for the ARMA(\bar{p}, \bar{q}) process. The red dotted rectangles illustrate the ascending grouping scheme for the LOG penalty.

recent advances in MIP algorithms (see e.g., Manzhou et al. (2019); Bertsimas et al. (2016); Mazumder and Radchenko (2017); Bertsimas and Van Parys (2017)), we use a convex nonsmooth regularizer that induces hierarchical sparsity structures of interest.

3.2 Latent Overlapping Group (LOG) Lasso

The hierarchical sparsity structure shown in Figure 2 is induced by regularizing the objective function in (8) by the LOG penalty – see Jacob et al. (2009). Let $\beta \triangleq [\phi, \theta] \in \mathbb{R}^{(\bar{p}+\bar{q})}$ denote all of the ARMA parameters. The LOG penalty function is defined as

$$\Omega_{\text{LOG}}(\beta) = \inf_{\nu^{(g)}, g \in \mathcal{G}} \left\{ \sum_{g \in \mathcal{G}} w_g \|\nu^{(g)}\|_2 \quad \text{s.t.} \quad \sum_{g \in \mathcal{G}} \nu^{(g)} = \beta, \nu_{g^c}^{(g)} = 0 \right\}, \quad (9)$$

where

$$\mathcal{G} = \left\{ \{1\}, \{1, 2\}, \dots, \{1, \dots, \bar{p}\}, \{\bar{p} + 1\}, \{\bar{p} + 1, \bar{p} + 2\}, \dots, \{\bar{p} + 1, \dots, \bar{p} + \bar{q}\} \right\},$$

$g \in \mathcal{G}$ is itself a set, $\nu^{(g)} \in \mathbb{R}^{(\bar{p}+\bar{q})}$ is a latent vector indexed by g , and w_g is the weight for the group g . $\nu_{g^c}^{(g)}$ selects the elements of $\nu^{(g)}$ based on the index g^c . The groups inside \mathcal{G} are shown with the red dotted rectangles in Figure 2, i.e., for each node, there is a group containing this node and all of its ascendants.

It is known that ℓ_2 -norm induces block sparsity; hence, the LOG penalty tries to find block sparse combinations of the latent variables that sum up to β (Jacob et al., 2009; Yan et al., 2017). For instance, for an ARMA model with $\bar{p} = 2$ and $\bar{q} = 2$, $\mathcal{G} = \{\{1\}, \{1, 2\}, \{3\}, \{3, 4\}\}$, the objective of the infimum is $|\nu_1^{\{1\}}| + \left\| [\nu_1^{\{1,2\}}, \nu_2^{\{1,2\}}] \right\| + |\nu_3^{\{3\}}| + \left\| [\nu_3^{\{3,4\}}, \nu_4^{\{3,4\}}] \right\|$ (where for simplicity $w_g = 1$, $\forall g \in \mathcal{G}$) and the con-

straints are

$$\begin{bmatrix} \nu_1^{\{1\}} \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} \nu_1^{\{1,2\}} \\ \nu_2^{\{1,2\}} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \nu_3^{\{3\}} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \nu_3^{\{3,4\}} \\ \nu_4^{\{3,4\}} \end{bmatrix} = \begin{bmatrix} \phi_1 \\ \phi_2 \\ \theta_1 \\ \theta_2 \end{bmatrix}.$$

3.3 The Proposed Hierarchically Sparse Learning Problem

The proposed Hierarchically Sparse (HS) learning problem is

$$\begin{aligned} \min_{\phi, \theta} \quad & \mathcal{L}(\phi, \theta) + \lambda \Omega_{\text{LOG}}(\phi, \theta) \\ \text{s.t.} \quad & \phi \in \mathcal{X}_\phi^p, \quad \theta \in \mathcal{X}_\theta^q, \end{aligned} \quad (\text{HS-ARMA})$$

where $\lambda > 0$ is a tuning parameter, \mathcal{X}_ϕ^p and \mathcal{X}_θ^q are defined in (4), and $\Omega_{\text{LOG}}(\cdot)$ is defined in (9). λ controls the tradeoff between the loss and penalty functions and, hence, allows model identification and parameter estimation simultaneously. Increasing λ results in sparser models where the resulted nested model satisfies the hierarchical sparsity structure shown in Figure 2. As discussed in Section 3.1, \bar{p} and \bar{q} are some upper bounds on the true p^* and q^* and are known a priori.

Given the convex nonsmooth function $\Omega_{\text{LOG}}(\cdot)$, we propose to solve (HS-ARMA) using a proximal method (Nesterov, 2013; Beck and Teboulle, 2009; Parikh et al., 2014). Similar to gradient methods which require iterative evaluation of the gradient, proximal methods require iterative evaluation of the proximal operator. Proximal operator of the $\Omega_{\text{LOG}}(\mathbf{b})$ at $\mathbf{b} \in \mathbb{R}^{(\bar{p}+\bar{q})}$ is defined as

$$\mathbf{prox}_{\lambda \Omega_{\text{LOG}}}(\mathbf{b}) \triangleq \underset{\beta \in \mathbb{R}^{(\bar{p}+\bar{q})}}{\text{argmin}} \left\{ \lambda \Omega_{\text{LOG}}(\beta) + \frac{1}{2} \|\beta - \mathbf{b}\|_2^2 \right\}. \quad (10)$$

Zhang et al. (2022) developed a two-block alternating direction method of multiplier (ADMM) with a sharing scheme (Boyd et al., 2011) to solve (10) – see Algorithm 1. The proposed algorithm can be parallelized over all groups in \mathcal{G} in the update of the first block; furthermore, it converges *linearly* – see Zhang et al. (2022) for more details.

Let $\Omega_{\text{LOG}}^{\text{AR}}$ and $\Omega_{\text{LOG}}^{\text{MA}}$ be the LOG penalties for the pure AR, i.e., ARMA(\bar{p} , 0), and pure MA, i.e., ARMA(0, \bar{q}), models, respectively. In Lemma 3.1 below, we show that the proximal operator of Ω_{LOG} is separable over ϕ and θ .

Lemma 3.1. *The proximal operator of the LOG penalty defined over the ARMA DAG is separable, i.e., $\mathbf{prox}_{\Omega_{\text{LOG}}}(\mathbf{b}_1, \mathbf{b}_2) = (\mathbf{prox}_{\Omega_{\text{LOG}}^{\text{AR}}}(\mathbf{b}_1), \mathbf{prox}_{\Omega_{\text{LOG}}^{\text{MA}}}(\mathbf{b}_2))$.*

Algorithm 1 Evaluating $\mathbf{prox}_{\lambda \Omega_{\text{LOG}}}(\mathbf{b})$

Require: $\mathbf{b}, \lambda, \alpha, w_g \forall g \in \mathcal{G}$
 1: $k = 0, U_{.g}^0 = \mathbf{0}, X_{.g}^{2,0} = \mathbf{0} \forall g \in \mathcal{G}$
 2: **while** stopping criterion not met **do**
 3: $k \leftarrow k + 1$
 4: $X_{.gg}^{1,k+1} \leftarrow \mathbf{prox}_{\lambda w_g \|\cdot\|_2}(X_{.gg}^{2,k} - U_{.gg}^k), \forall g \in \mathcal{G}$
 5: $X_{.g^c g}^{1,k+1} \leftarrow \mathbf{0}, \forall g \in \mathcal{G}$
 6: $\bar{\mathbf{x}}^{2,k+1} \leftarrow \frac{1}{|\mathcal{G}|+\rho} \left(\mathbf{b} + \frac{\rho}{|\mathcal{G}|} \sum_{g \in \mathcal{G}} (X_{.g}^{1,k+1} + U_{.g}^k) \right)$
 7: $X_{.g}^{2,k+1} \leftarrow \bar{\mathbf{x}}^{2,k+1} + X_{.g}^{1,k+1} + U_{.g}^k - (1/|\mathcal{G}|) \sum_{g \in \mathcal{G}} (X_{.g}^{1,k+1} + U_{.g}^k), \forall g \in \mathcal{G}$
 8: $U_{.g}^{k+1} = U_{.g}^k + (\alpha/\rho) \left(\frac{1}{|\mathcal{G}|} \sum_{g \in \mathcal{G}} (X_{.g}^{1,k+1} + U_{.g}^k) - \bar{\mathbf{x}}^2 \right), \forall g \in \mathcal{G}$
 9: **end while**
 10: $\beta = \sum_{g \in \mathcal{G}} X_{.g}^{1,k+1}$
Output: β

Proof. With a slight abuse of notation, let \mathcal{G}^{AR} be the set of groups for $\Omega_{\text{LOG}}^{\text{AR}}$ such that $\sum_{g \in \mathcal{G}^{\text{AR}}} \nu^{(g)} = \phi$ (the top path graph in Figure 2). Similarly, let \mathcal{G}^{MA} be the set of groups for $\Omega_{\text{LOG}}^{\text{MA}}$ such that $\sum_{g \in \mathcal{G}^{\text{MA}}} \omega^{(g)} = \theta$. Given that the objective of the infimum in the definition of Ω_{LOG} for the ARMA DAG is separable in \mathcal{G}^{AR} and \mathcal{G}^{MA} , we have $\Omega_{\text{LOG}}(\phi, \theta) = \Omega_{\text{LOG}}^{\text{AR}}(\phi) + \Omega_{\text{LOG}}^{\text{MA}}(\theta)$. Hence, the result follows from the separable sum property of the proximal operator. \square

Indeed, in Algorithm 2, the proximal operator of LOG is not evaluated in one step while the algorithm evaluates $\mathbf{prox}_{\lambda \Omega_{\text{LOG}}^{\text{AR}}}$ and $\mathbf{prox}_{\lambda \Omega_{\text{LOG}}^{\text{MA}}}$ sequentially in a Gauss-Seidel manner.

Algorithm 2 Proximal BCD to solve (HS-ARMA)

Require: $\lambda, \bar{p}, \bar{q}, \phi_0 \in \mathcal{X}_\phi^{\bar{p}}, \theta_0 \in \mathcal{X}_\theta^{\bar{q}}$
 1: $k = 1$
 2: **while** stopping criterion not met **do**
 3: $\phi^{k+1/2} \leftarrow \mathbf{prox}_{\lambda \Omega_{\text{LOG}}^{\text{AR}}}(\phi^k - \gamma_k \nabla_\phi \mathcal{L}(\phi^k, \theta^k))$
 (prox is calculated by Algorithm 1)
 4: $p \leftarrow \text{card}(\{i : \phi_i^{k+1/2} \neq 0\})$
 5: $\phi^{k+1} \leftarrow \text{Proj}_{\bar{\mathcal{X}}_\phi^p}(\phi^{k+1/2})$
 6: $\theta^{k+1/2} \leftarrow \mathbf{prox}_{\lambda \Omega_{\text{LOG}}^{\text{MA}}}(\theta^k - \gamma_k \nabla_\theta \mathcal{L}(\phi^{k+1}, \theta^k))$
 (prox is calculated by Algorithm 1)
 7: $q \leftarrow \text{card}(\{i : \theta_i^{k+1/2} \neq 0\})$
 8: $\theta^{k+1} \leftarrow \text{Proj}_{\bar{\mathcal{X}}_\theta^q}(\theta^{k+1/2})$
 9: $k \leftarrow k + 1$
 10: **end while**
Output: (ϕ_k, θ_k)

The algorithm to solve problem (HS-ARMA) is a two-block proximal block coordinate descent (BCD) with projection, shown in Algorithm 2. From (1), since $\epsilon_t = y_t - \phi^\top \mathbf{y}_{t-p}^{t-1} - \theta^\top \epsilon_{t-q}^{t-1}$ where $\mathbf{y}_{t-p}^{t-1} = [y_{t-1}, \dots, y_{t-p}]$

and $\boldsymbol{\epsilon}_{t-q}^{t-1} = [\epsilon_{t-1}, \dots, \epsilon_{t-q}]$, we have

$$\nabla_{\boldsymbol{\phi}} \mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\theta}) = - \sum_{t=\max\{\bar{p}, \bar{q}\}}^T (y_t - \boldsymbol{\phi}^\top \mathbf{y}_{t-p}^{t-1} - \boldsymbol{\theta}^\top \boldsymbol{\epsilon}_{t-q}^{t-1}) \mathbf{y}_{t-p}^{t-1}, \quad (11a)$$

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\theta}) = - \sum_{t=\max\{\bar{p}, \bar{q}\}}^T (y_t - \boldsymbol{\phi}^\top \mathbf{y}_{t-p}^{t-1} - \boldsymbol{\theta}^\top \boldsymbol{\epsilon}_{t-q}^{t-1}) \boldsymbol{\epsilon}_{t-q}^{t-1}. \quad (11b)$$

The gradient updates are passed to proximal operators as arguments which are indeed proximal gradient steps (Beck and Teboulle, 2009; Parikh et al., 2014). Note that the solution of the proximal operators is sparse vectors that conform to the hierarchical sparsity of Figure 2.

The solutions of the proximal gradient steps for the AR and MA components, i.e., $\boldsymbol{\phi}^{k+1/2}$ and $\boldsymbol{\theta}^{k+1/2}$ are not necessarily stationary or invertible, respectively. The stationarity and invertibility of AR and MA are regained by projection on $\mathcal{X}_{\boldsymbol{\phi}}^p$ and $\mathcal{X}_{\boldsymbol{\theta}}^q$ where p and q are the order of AR and MA components from the proximal steps. For the projection, we use the second definition of $\mathcal{X}_{\boldsymbol{\alpha}}^d$ in (6). Since $\mathcal{X}_{\boldsymbol{\alpha}}^d$ is an open set, following Combettes and Trussell (1992), we find its approximation with a closed set from inside as

$$\begin{aligned} \tilde{\mathcal{X}}_{\boldsymbol{\alpha}}^d(\delta) &\triangleq \{\boldsymbol{\alpha} \in \mathbb{R}^d : \forall z \in \mathbb{C}, \bar{P}_{\boldsymbol{\alpha}}^d(z) = 0 \\ &\Rightarrow -1 + \delta \leq z \leq 1 - \delta\}, \end{aligned} \quad (12)$$

where $\delta > 0$ determines the approximation gap. Euclidean projection on $\tilde{\mathcal{X}}_{\boldsymbol{\phi}}^p(\delta)$ and $\tilde{\mathcal{X}}_{\boldsymbol{\theta}}^q(\delta)$ sets guarantee stationarity and invertibility of $\boldsymbol{\phi}^{t+1}$ and $\boldsymbol{\theta}^{t+1}$, respectively. Note that these projections do *not* change the sparsity of the parameters.

Finally, note that ϵ_t in the objective of (HS-ARMA) is calculated based on ARMA(\bar{p}, \bar{q}). Hence, while the iterates $\boldsymbol{\phi}^{t+1}$ and $\boldsymbol{\theta}^{t+1}$ are feasible with respect to $\mathcal{X}_{\boldsymbol{\phi}}^p$ and $\mathcal{X}_{\boldsymbol{\theta}}^q$, respectively, we need to show $(\boldsymbol{\phi}^{k+1}, \boldsymbol{\theta}^{k+1}) \in \mathcal{X}_{\boldsymbol{\phi}}^{\bar{p}} \times \mathcal{X}_{\boldsymbol{\theta}}^{\bar{q}}$. This is established in Lemma 3.2 below.

Lemma 3.2. *For any $d \in \{1, 2, \dots\}$, we have $\mathcal{X}_{\boldsymbol{\alpha}}^d \subseteq \mathcal{X}_{\boldsymbol{\alpha}}^{d+1}$.*

Proof. Proof follows from the definition of $\mathcal{X}_{\boldsymbol{\alpha}}^d$ in (4), and that if $\boldsymbol{\alpha} \in \mathcal{X}_{\boldsymbol{\alpha}}^d$ then $[\boldsymbol{\alpha}, 0] \in \mathcal{X}_{\boldsymbol{\alpha}}^{d+1}$. \square

Therefore, $\{\mathcal{X}_{\boldsymbol{\alpha}}^d\}_{d=1}^{\bar{d}}$ is a sequence of nested sets as $\mathcal{X}_{\boldsymbol{\alpha}}^1 \subseteq \dots \subseteq \mathcal{X}_{\boldsymbol{\alpha}}^{\bar{d}}$. However, the reverse is not true, i.e., $\boldsymbol{\alpha} \in \mathcal{X}_{\boldsymbol{\alpha}}^d$ is *not sufficient* for $[\alpha_1, \dots, \alpha_{d-1}] \in \mathcal{X}_{\boldsymbol{\alpha}}^{d-1}$, which can be shown by counter examples.

3.4 A Note on the Optimization Problem (HS-ARMA)

Problem (HS-ARMA) requires nonconvex and non-smooth optimization over a nonconvex set. To be specific, if $q = 0$ the loss function is convex in $\boldsymbol{\phi}$; otherwise, $\mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\theta})$ is nonconvex in *both* $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$. Indeed, when $q > 0$ the objective function is a *polynomial* function of degree $T - \max\{p, q\}$. The $\Omega_{\text{LOG}}(\boldsymbol{\phi}, \boldsymbol{\theta})$ penalty is jointly convex but nonsmooth function. Finally, $\mathcal{X}_{\boldsymbol{\phi}}^p$ and $\mathcal{X}_{\boldsymbol{\theta}}^q$ are open nonconvex sets and their approximations $\tilde{\mathcal{X}}_{\boldsymbol{\phi}}^p$ and $\tilde{\mathcal{X}}_{\boldsymbol{\theta}}^q$ (defined in (12)) are closed but still nonconvex sets.

To deal with nonconvexities of $\tilde{\mathcal{X}}_{\boldsymbol{\phi}}^p$ and $\tilde{\mathcal{X}}_{\boldsymbol{\theta}}^q$, one may try to approximate them with some inscribed convex sets which require generalizations of the *potato peeling problem* (Goodman, 1981) and the algorithm in Chang and Yap (1986) to non-polygon geometries – see also Cabello et al. (2017). Note that optimization over the convex hulls of these sets may result in nonstationary or noninvertible solutions.

Under some convex approximations of the sets $\tilde{\mathcal{X}}_{\boldsymbol{\phi}}^p$ and $\tilde{\mathcal{X}}_{\boldsymbol{\theta}}^q$, the problem under investigation is a nonconvex nonsmooth optimization over a convex set. For such a setting, algorithms are settled with finding solutions that satisfy some necessary optimality conditions, e.g., stationary solutions which are those that lack a feasible descent direction. To the best of our knowledge, the only study that provides a method that converges to stationary points in this setting is Razaviyayn et al. (2013), which involves iterative minimization of a consistent majorizer of the objective function over the feasible set.

4 NUMERICAL STUDIES

The corresponding code is provided in https://github.com/Yin-LIU/ARMA_identify_proximal.

4.1 Synthetic Data Generation Process

To generate a stationary and an invertible ARMA(p^*, q^*) model, we first generate $p^* + q^*$ numbers uniformly at random on $[-1, -0.1] \cup [0.1, 1]$ for all parameters. The samples are then rejected if the stationary and invertibility conditions, based on (6), are not satisfied. Given an accepted sampled parameter $(\boldsymbol{\phi}^{*,i}, \boldsymbol{\theta}^{*,i})$, a realization of the time series with length $T = 4000$ is simulated with a zero mean and variance equal to one.

Table 1: The mean (standard deviation) of HS-ARMA estimation errors. Boldface numbers are the minimum mean error for each model (row). Parameter estimates are obtained by the proximal BCD Algorithm 2.

(p^*, q^*)	λ_0					
	0.5	1	2	3	5	10
(3,2)	0.62 (0.350)	0.38 (0.392)	0.54 (0.451)	0.59 (0.438)	0.63 (0.394)	0.77 (0.338)
(3,3)	0.64 (0.494)	0.74 (0.518)	0.85 (0.491)	0.92 (0.486)	1.05 (0.393)	1.05 (0.355)
(2,6)	0.79 (0.326)	0.58 (0.324)	0.46 (0.339)	0.64 (0.368)	0.92 (0.390)	1.04 (0.435)
(6,6)	0.69 (0.414)	0.79 (0.518)	1.10 (0.477)	1.25 (0.468)	1.32 (0.420)	1.29 (0.344)
(8,5)	0.87 (0.307)	0.99 (0.426)	1.22 (0.492)	1.41 (0.586)	1.57 (0.492)	1.48 (0.502)

4.2 Model Identification and Parameter Estimation Accuracy

To evaluate the estimation error of the proposed method, we simulate $n = 20$ realizations of ARMA models with orders (p^*, q^*) such that $p^* \leq \bar{p} = 10$ and $q^* \leq \bar{q} = 10$ following our discussion in Section 4.1. The tuning parameter of the Ω_{LOG} penalty is set as $\lambda = \lambda_0 \sqrt{T}$ with $\lambda_0 \in \{0.5, 1, 2, 3, 5, 10\}$ and w_g in its definition is set to $|g|^{1/2}$. The estimation error is calculated as $\epsilon_{\lambda_0} = \|(\hat{\phi}_{\lambda_0}, \hat{\theta}_{\lambda_0}) - (\phi^*, \theta^*)\|_2$, where (ϕ^*, θ^*) are the true and $(\hat{\phi}, \hat{\theta})$ are the estimated parameters based on Algorithm 2. Table 1 reports the mean and standard deviation of the estimation errors for different λ_0 values.

To provide a better understanding of the quality of parameter estimates and how they conform to the induced sparsity structure in Figure 2, we conducted another study. First, we sampled one realization from 10 different ARMA(3,2) models. Then, with $\bar{p} = \bar{q} = 5$ and $\lambda_0 \in \{0.5, 1, 2, 3, 5, 10\}$, the HS-ARMA parameter estimates $(\hat{\phi}_{\lambda_0}^i, \hat{\theta}_{\lambda_0}^i)$ are calculated using Algorithm 2 and reported along with the true parameters $(\phi^{*,i}, \theta^{*,i})$ in Table 2 in Appendix A, where i is the simulation index. Simple tuning of λ_0 allows the method to correctly identify the true orders (p^*, q^*) and the estimated parameters conform to the underlying sparsity structure. Furthermore, the estimation errors are reasonably small. We also compared the estimation errors with pre-identified models where their parameters are estimated using a package – see Figure 3. The mean of the HS-ARMA estimation error lies between those of the correctly and incorrectly identified (by one order in the AR component) models. For some samples with λ_0 around 2 or 3, the error of HS-ARMA is very close to the correctly identified ARMA model.

4.3 Prediction Performance

We also compare the prediction performance of the HS-ARMA with those of correctly and incorrectly identified models using 10 realizations of one

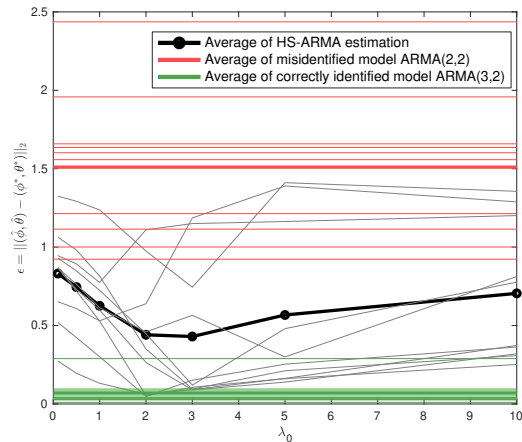


Figure 3: The estimation error of HS-ARMA and two pre-identified models. The three thicker lines are the mean estimation errors and the thinner lines represent estimation errors for each sample.

ARMA(3,2) model. For each realization, the estimated parameters with $\lambda_0 \in \{0.5, 1, 2, 3, 5\}$ are used to forecast the process for the next 20 time points. Note that $\lambda_0 = 10$ is omitted because the fitted parameters were too sparse. Figure 4 illustrates the Root Mean Square Error (RMSE) for these methods.

For some λ_0 , the RMSE of HS-ARMA is smaller than that of the correctly identified ARMA model. Furthermore, all HS-ARMA predictions for different λ_0 values have significantly lower RMSE compared to the incorrectly identified model.

4.4 Comparison with Other Methods

We also compare our method with the one proposed by Wilms et al. (2017) (the “bigtime” R package) which also considers the hierarchical lag penalty, namely H-Lag penalty, and the lasso ℓ_1 penalty. While the underlying optimization problem and the algorithm to solve it are fundamentally different than those proposed in this work, we believe the method in Wilms et al. (2017) specifically with the H-Lag penalty is the best benchmark for the proposed method. The parameter estimation method in Wilms et al. (2017) has two

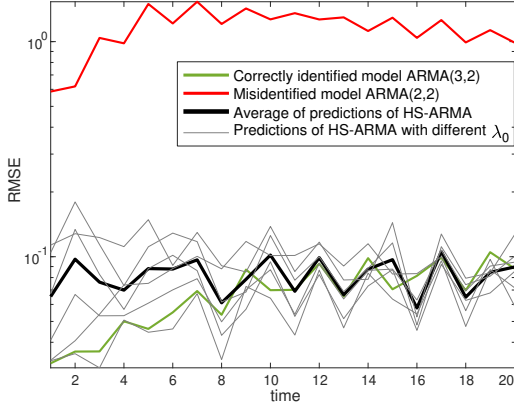


Figure 4: Prediction RMSEs for the HS-ARMA method vs. the correctly and incorrectly identified models. Each grey thin line is the RMSE of HS-ARMA with one λ_0 from $\{0.5, 1, 2, 3, 5\}$ from ten realizations and the black thick line is the average of the grey lines. The green and red lines are the RMSEs from the ten realizations for correctly and incorrectly identified models.

different phases. In the first phase, their method estimates a pure AR model, since every invertible ARMA process can be represented by an $AR(\infty)$ model. The estimated AR model is used to approximate the unobserved error terms which are used as the covariates of the MA component. In the second phase, a least-square objective regularized with a sparsity-inducing penalty is used to estimate the parameters of the final model.

We consider 9 different combinations of $ARMA(p, q)$ models with $p, q \in \{2, 5, 8\}$. For each combination, we construct four scenarios where the maximum absolute value of the roots of AR and MA components are chosen to be either 0.5 (invertible/stationary process) or 0.99 (close to none invertible/stationary process). In each scenario, we randomly generate an ARMA model and simulate 20 time series of length 200. Following the setting in Wilms et al. (2017), the maximum potential lag $\bar{p} = \bar{q} = \lfloor 0.75\sqrt{(T)} \rfloor = 10$, and AR and MA penalty coefficients λ_p and λ_q belong to 10 logarithmically spaced points between 1 and 100. The best combination of λ_p and λ_q parameters are determined by Bayesian Information Criterion (BIC). For each combination of (p, q) , the RMSEs are averaged over the final selected models. The results are presented in Figure 5. In most cases, H-Lag penalty has lower RMSEs compared to the ℓ_1 penalty; however, the proposed HS-ARMA method has the lowest RMSEs in all 9 cases.

4.5 Real Time Series Prediction

We also implement the proposed HS-ARMA method on the real dataset, which is the Netflix stock prices

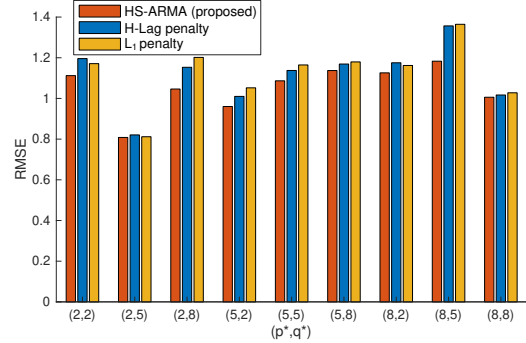


Figure 5: Comparison of the proposed HS-ARMA method with the hierarchical lag (H-Lag) and ℓ_1 penalty methods from Wilms et al. (2017).

from 4/15/2020 to 9/24/2021 with a total of 355 data points. To evaluate the influence of the penalty parameter of HS-ARMA, the model is fitted with different combinations of λ_{AR} and λ_{MA} . After the ARMA model is identified by HS-ARMA, we evaluate the performance of this model by different criteria, including AIC, AICc, and BIC. The results are presented in Figure 6. It is obvious that a larger penalty parameter will enforce the lower order of the ARMA model and the best model occurs when the value of the parameter is set properly. The advantage of HS-ARMA is that the search in the continuous λ parameter space can be performed more efficiently compared to a brute-force grid search. For instance, the proposed algorithm can be easily incorporated into a hyperparameter optimization method (e.g., Franceschi et al. (2018)) with a polynomial time solution to find the optimal λ values.

5 CONCLUDING REMARKS

This work presents a new learning framework that allows model identification and parameter estimation for ARMA time series models simultaneously. To do so, we use a hierarchical sparsity-inducing penalty, namely the Latent Overlapping Group (LOG) lasso, in the objective of the parameter estimation problem. While the addition of a nonsmooth (but convex) function to the objective of an already difficult nonconvex optimization seems restrictive, we propose a proximal block coordinate descent (BCD) algorithm that can solve the problem to a potential stationary point efficiently. Numerical simulation studies confirm the capabilities of the proposed learning framework to identify the true model and estimate its parameters with reasonably high accuracy.

We believe that this study sheds some light on the hard optimization problem behind the parameter estimation of ARMA time series models (see our discussion

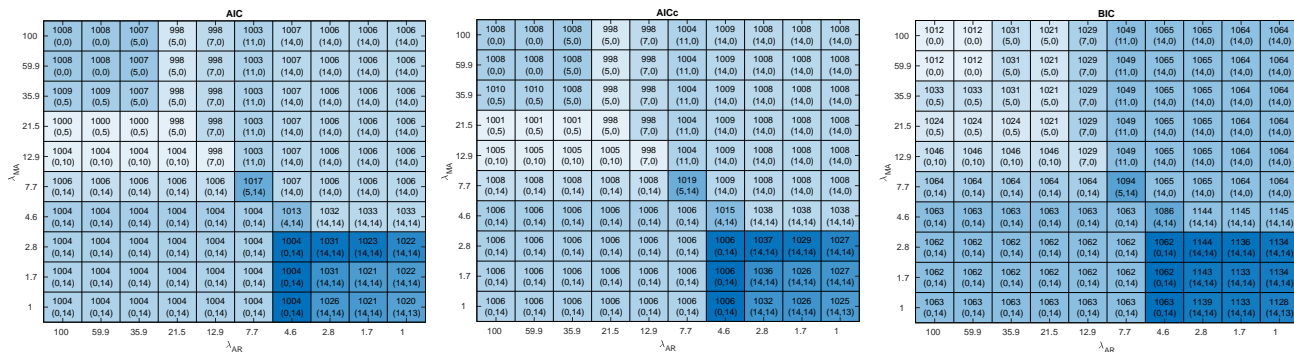


Figure 6: HS-ARMA fitting performance for the Netflix stock data. The identified $ARMA(p, q)$ models are shown inside the parentheses.

in Section 3.4). Furthermore, we hope it motivates future studies to look into the convergence analysis of the proposed proximal BCD or other algorithms for such problem structures. Finally, the proposed framework can be extended to fit vector ARMA (VARMA) models where the underlying path graphs would contain multiple variables per node (see e.g. the bottom plot in Figure 1), which we also leave for future studies.

Acknowledgements

We would like to express our gratitude to the reviewers for their valuable feedback.

References

Ayodele Ariyo Adebisi, Aderemi Oluyinka Adewumi, and Charles Korede Ayo. Comparison of arima and artificial neural networks models for stock price prediction. *Journal of Applied Mathematics*, 2014, 2014.

Francis Bach, Rodolphe Jenatton, Julien Mairal, Guillaume Obozinski, et al. Structured sparsity through convex optimization. *Statistical Science*, 27(4):450–468, 2012.

Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1): 183–202, 2009.

Messaoud Benidir and B Picinbono. Nonconvexity of the stability domain of digital filters. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 38(8):1459–1460, 1990.

Dimitris Bertsimas and Bart Van Parys. Sparse high-dimensional regression: Exact scalable algorithms and phase transitions. *arXiv preprint arXiv:1709.10029*, 2017.

Dimitris Bertsimas, Angela King, Rahul Mazumder, et al. Best subset selection via a modern optimiza-

tion lens. *The Annals of Statistics*, 44(2):813–852, 2016.

Jacob Bien, Jonathan Taylor, and Robert Tibshirani. A lasso for hierarchical interactions. *Annals of statistics*, 41(3):1111, 2013.

Daniel Billings and Jiann-Shiou Yang. Application of the arima models to urban roadway travel time prediction—a case study. In *2006 IEEE International Conference on Systems, Man and Cybernetics*, volume 3, pages 2529–2534. IEEE, 2006.

Vincent D Blondel, Mert Gurbuzbalaban, Alexandre Megretski, and Michael L Overton. Explicit solutions for root optimization of a polynomial family with one affine constraint. *IEEE transactions on automatic control*, 57(12):3078–3089, 2012.

George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

Sergio Cabello, Josef Cibulka, Jan Kyncl, Maria Saumell, and Pavel Valtr. Peeling potatoes near-optimally in near-linear time. *SIAM Journal on Computing*, 46(5):1574–1602, 2017.

Rodrigo N Calheiros, Enayat Masoumi, Rajiv Ranjan, and Rajkumar Buyya. Workload prediction using arima model and its impact on cloud applications? qos. *IEEE Transactions on Cloud Computing*, 3(4): 449–458, 2014.

Kung-Sik Chan and Kun Chen. Subset arma selection via the adaptive lasso. *Statistics and its Interface*, 4(2):197–205, 2011.

Jyun-Sheng Chang and Chee-Keng Yap. A polynomial solution for the potato-peeling problem. *Discrete & Computational Geometry*, 1(2):155–182, 1986.

- Peiyuan Chen, Troels Pedersen, Birgitte Bak-Jensen, and Zhe Chen. Arima-based time series model of stochastic wind power generation. *IEEE transactions on power systems*, 25(2):667–676, 2009.
- Patrick L Combettes and H Joel Trussell. Best stable and invertible approximations for arma systems. *IEEE Transactions on signal processing*, 40(12):3066–3069, 1992.
- Enrique Del Castillo. *Statistical process adjustment for quality control*, volume 369. Wiley-Interscience, 2002.
- Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazi, and Massimiliano Pontil. Bilevel programming for hyperparameter optimization and meta-learning. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1568–1577. PMLR, 10–15 Jul 2018.
- Tryphon T Georgiou and Anders Lindquist. A convex optimization approach to arma modeling. *IEEE transactions on automatic control*, 53(5):1108–1119, 2008.
- Jacob E Goodman. On the largest convex polygon contained in a non-convex n-gon, or how to peel a potato. *Geometriae Dedicata*, 11(1):99–106, 1981.
- James D Hamilton. *Time series analysis*, volume 2. Princeton New Jersey, 1994.
- Ping Han, Peng Xin Wang, Shu Yu Zhang, et al. Drought forecasting based on the remote sensing data using arima models. *Mathematical and computer modelling*, 51(11-12):1398–1403, 2010.
- Nan-Jung Hsu, Hung-Lin Hung, and Ya-Mei Chang. Subset selection for vector autoregressive processes using lasso. *Computational Statistics & Data Analysis*, 52(7):3645–3657, 2008.
- Laurent Jacob, Guillaume Obozinski, and Jean-Philippe Vert. Group lasso with overlap and graph lasso. In *Proceedings of the 26th annual international conference on machine learning*, pages 433–440. ACM, 2009.
- Rodolphe Jenatton, Jean-Yves Audibert, and Francis Bach. Structured variable selection with sparsity-inducing norms. *Journal of Machine Learning Research*, 12(Oct):2777–2824, 2011a.
- Rodolphe Jenatton, Julien Mairal, Guillaume Obozinski, and Francis Bach. Proximal methods for hierarchical sparse coding. *Journal of Machine Learning Research*, 12(Jul):2297–2334, 2011b.
- Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. Statistical and machine learning forecasting methods: Concerns and ways forward. *PloS one*, 13(3), 2018.
- Hasan Manzour, Simge Küçükyavuz, and Ali Shojajae. Integer programming for learning directed acyclic graphs from continuous data. *arXiv preprint arXiv:1904.10574*, 2019.
- Rahul Mazumder and Peter Radchenko. Thediscrete dantzig selector: Estimating sparse linear models via mixed integer linear optimization. *IEEE Transactions on Information Theory*, 63(5):3053–3075, 2017.
- Randolph L Moses and Duixian Liu. Determining the closest stable polynomial to an unstable one. *IEEE Transactions on signal processing*, 39(4):901–906, 1991.
- Yuval Nardi and Alessandro Rinaldo. Autoregressive process modeling via the lasso procedure. *Journal of Multivariate Analysis*, 102(3):528–549, 2011.
- Yu Nesterov. Gradient methods for minimizing composite functions. *Mathematical Programming*, 140(1):125–161, 2013.
- William B Nicholson, Ines Wilms, Jacob Bien, and David S Matteson. High dimensional forecasting via interpretable vector autoregression. *arXiv preprint arXiv:1412.5250*, 2014.
- Neal Parikh, Stephen Boyd, et al. Proximal algorithms. *Foundations and Trends® in Optimization*, 1(3):127–239, 2014.
- Meisam Razaviyayn, Mingyi Hong, and Zhi-Quan Luo. A unified convergence analysis of block successive minimization methods for nonsmooth optimization. *SIAM Journal on Optimization*, 23(2):1126–1153, 2013.
- Yunwen Ren and Xinsheng Zhang. Subset selection for vector autoregressive processes via adaptive lasso. *Statistics & probability letters*, 80(23-24):1705–1712, 2010.
- Hansheng Wang, Guodong Li, and Chih-Ling Tsai. Regression coefficient and autoregressive order shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(1):63–78, 2007.
- Wen-chuan Wang, Kwok-wing Chau, Dong-mei Xu, and Xiao-Yun Chen. Improving forecasting accuracy of annual runoff time series using arima based on eemd decomposition. *Water Resources Management*, 29(8):2655–2675, 2015.
- Ines Wilms, Sumanta Basu, Jacob Bien, and David S Matteson. Sparse identification and estimation of large-scale vector autoregressive moving averages. *arXiv preprint arXiv:1707.09208*, 2017.

Xiaohan Yan, Jacob Bien, et al. Hierarchical sparse modeling: A choice of two group lasso formulations. *Statistical Science*, 32(4):531–560, 2017.

Dewei Zhang, Yin Liu, and Sam Davanloo Tajbakhsh. A first-order optimization algorithm for statistical learning with hierarchical sparsity structure. *INFORMS Journal on Computing*, 34(2):1126–1140, 2022.

Xingyu Zhang, Tao Zhang, Alistair A Young, and Xiaosong Li. Applications and comparisons of four time series models in epidemiological surveillance data. *PLoS One*, 9(2), 2014.

Peng Zhao, Guilherme Rocha, and Bin Yu. The composite absolute penalties family for grouped and hierarchical variable selection. *The Annals of Statistics*, pages 3468–3497, 2009.

Checklist

1. For all models and algorithms presented, check if you include:
 - (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
 - (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [No]
 - (c) (Optional) Anonymized source code, with the specification of all dependencies, including external libraries. [Yes]
2. For any theoretical claim, check if you include:
 - (a) Statements of the full set of assumptions of all theoretical results. [Yes]
 - (b) Complete proofs of all theoretical results. [Yes]
 - (c) Clear explanations of any assumptions. [Yes]
3. For all figures and tables that present empirical results, check if you include:
 - (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
 - (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
 - (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
 - (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [No]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
 - (a) Citations of the creator If your work uses existing assets. [Yes]
 - (b) The license information of the assets, if applicable. [Yes]
 - (c) New assets either in the supplemental material or as a URL, if applicable. [Yes]
 - (d) Information about consent from data providers/curators. [Not Applicable]
 - (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]
5. If you used crowdsourcing or conducted research with human subjects, check if you include:
 - (a) The full text of instructions given to participants and screenshots. [Not Applicable]
 - (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
 - (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Fitting ARMA Time Series Models without Identification:
A Proximal Approach
Supplementary Materials

Appendix A EXPERIMENT RESULTS

Table 2: Model identification and parameter estimation accuracy of the HS-ARMA method for ten simulations from ARMA(3,2) (one realization each). Parameters are estimated using the proximal BCD Algorithm 2. Boldface columns denote the best identified models with the lowest estimation errors.

$(\phi^{*,1}, \theta^{*,1})$		λ_0						$(\phi^{*,2}, \theta^{*,2})$		λ_0					
		0.5	1	2	3	5	10			0.5	1	2	3	5	10
ϕ_1	-0.16	-0.23	-0.22	-0.25	-0.10	0.01	0.05	0.13	0.24	0.18	0.10	0.08	0.04	-0.14	
ϕ_2	-0.98	-0.53	-0.60	-0.83	-0.99	-0.99	-0.99	0.42	0.41	0.44	0.45	0.46	0.48	0.53	
ϕ_3	-0.22	-0.21	-0.24	-0.32	-0.16	-0.05	-0.01	-0.44	-0.51	-0.50	-0.42	-0.39	-0.34	-0.15	
ϕ_4	0.00	0.44	0.38	0.15	0.00	0.00	0.00	0.00	0.04	0.01	0.00	0.00	0.00	0.00	
ϕ_5	0.00	0.10	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
θ_1	-0.45	-0.38	-0.38	-0.35	-0.46	-0.51	-0.50	0.49	0.37	0.43	0.50	0.51	0.55	0.68	
θ_2	0.91	0.38	0.44	0.64	0.89	0.92	0.85	0.34	0.28	0.29	0.31	0.30	0.28	0.26	
θ_3	0.00	0.29	0.27	0.20	0.00	-0.03	0.00	0.00	-0.03	0.00	0.00	0.00	0.00	0.00	
θ_4	0.00	-0.45	-0.40	-0.20	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	
θ_5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
ϵ_{λ_0}		0.98	0.82	0.35	0.10	0.21	0.31		0.20	0.13	0.06	0.09	0.16	0.37	

$(\phi^{*,3}, \theta^{*,3})$		λ_0						$(\phi^{*,4}, \theta^{*,4})$		λ_0					
		0.5	1	2	3	5	10			0.5	1	2	3	5	10
ϕ_1	-0.64	-0.48	-0.46	-0.47	-0.55	-0.66	-0.75	-0.88	-0.12	-0.18	-0.37	-0.51	-0.15	-0.16	
ϕ_2	-0.70	-0.25	-0.33	-0.46	-0.57	-0.61	-0.40	-0.28	0.13	0.18	0.18	0.07	0.28	0.24	
ϕ_3	-0.56	-0.25	-0.29	-0.37	-0.46	-0.45	-0.24	0.37	0.35	0.41	0.47	0.43	0.29	0.27	
ϕ_4	0.00	0.33	0.26	0.14	0.04	0.00	0.01	0.00	-0.39	-0.34	-0.18	-0.11	-0.21	-0.07	
ϕ_5	0.00	0.18	0.11	0.00	0.00	0.00	0.00	0.00	0.07	0.03	0.00	0.00	0.00	0.00	
θ_1	-0.49	-0.61	-0.63	-0.62	-0.55	-0.44	-0.28	0.84	0.09	0.14	0.32	0.46	0.03	0.00	
θ_2	0.49	0.20	0.29	0.39	0.41	0.30	0.00	0.57	0.19	0.14	0.14	0.24	0.00	0.00	
θ_3	0.00	0.11	0.07	0.00	0.00	0.00	0.01	0.00	-0.19	-0.24	-0.19	-0.06	0.00	0.00	
θ_4	0.00	-0.15	-0.09	0.00	0.00	0.00	0.01	0.00	0.17	0.10	0.00	0.00	0.00	0.00	
θ_5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	
ϵ_{λ_0}		0.76	0.61	0.46	0.57	0.30	0.81		1.29	1.24	0.98	0.74	1.41	1.36	

Continued on next page

Table 2 – continued from previous page

		λ_0						λ_0							
$(\phi^{*,5}, \theta^{*,5})$		0.5	1	2	3	5	10	$(\phi^{*,6}, \theta^{*,6})$		0.5	1	2	3	5	10
ϕ_1	-0.19	-0.32	-0.37	-0.37	-0.51	-0.53	-0.53	-0.58	-0.21	-0.25	-0.43	-0.57	-0.57	-0.53	
ϕ_2	0.55	0.04	0.06	-0.12	-0.20	-0.21	-0.20	0.61	0.57	0.67	0.67	0.60	0.60	0.63	
ϕ_3	0.52	0.05	0.13	0.16	0.14	0.13	0.01	0.85	0.48	0.58	0.74	0.82	0.81	0.78	
ϕ_4	0.00	-0.16	-0.11	-0.10	-0.06	-0.01	0.00	0.00	-0.17	-0.20	-0.12	0.00	0.00	-0.01	
ϕ_5	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.19	0.09	0.00	0.00	0.00	0.00	
θ_1	-0.30	-0.17	-0.12	-0.13	0.00	0.00	0.00	0.24	-0.16	-0.12	0.05	0.19	0.18	0.08	
θ_2	-0.64	-0.18	-0.24	-0.01	0.00	0.00	0.00	0.23	0.40	0.29	0.21	0.18	0.13	0.01	
θ_3	0.00	0.22	0.16	0.00	0.00	0.00	0.00	0.00	-0.06	-0.05	-0.01	0.00	0.00	0.00	
θ_4	0.00	-0.03	-0.02	0.00	0.00	0.00	0.00	0.00	0.04	0.00	0.00	0.00	0.00	0.00	
θ_5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
ϵ_{λ_0}		0.89	0.77	1.11	1.15	1.16	1.20		0.73	0.60	0.26	0.09	0.14	0.32	

		λ_0						λ_0							
$(\phi^{*,7}, \theta^{*,7})$		0.5	1	2	3	5	10	$(\phi^{*,8}, \theta^{*,8})$		0.5	1	2	3	5	10
ϕ_1	-0.34	0.03	-0.02	0.05	0.24	0.41	0.39	0.92	0.83	0.83	0.88	0.93	0.93	0.93	
ϕ_2	-0.53	-0.54	-0.61	-0.68	-0.78	-0.84	-0.82	0.91	0.70	0.78	0.96	0.91	0.90	0.89	
ϕ_3	-0.65	-0.39	-0.40	-0.31	-0.13	0.00	0.00	-0.88	-0.50	-0.58	-0.83	-0.87	-0.87	-0.86	
ϕ_4	0.00	0.13	0.05	0.00	0.00	0.00	0.00	0.00	0.19	0.12	-0.04	0.00	0.00	-0.01	
ϕ_5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.27	-0.20	0.00	0.00	0.00	-0.01	
θ_1	0.32	-0.03	0.01	-0.04	-0.24	-0.47	-0.37	0.73	0.81	0.81	0.76	0.67	0.64	0.57	
θ_2	-0.49	-0.47	-0.42	-0.34	-0.19	-0.02	-0.01	0.21	0.58	0.50	0.21	0.13	0.10	0.01	
θ_3	0.00	0.12	0.09	0.05	0.00	0.00	0.00	0.00	0.25	0.18	0.00	0.00	0.00	0.00	
θ_4	0.00	-0.05	-0.02	-0.01	0.00	0.00	0.00	0.00	0.05	0.03	0.00	0.00	0.00	0.00	
θ_5	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
ϵ_{λ_0}		0.61	0.53	0.64	1.19	1.39	1.29		0.72	0.53	0.05	0.11	0.16	0.25	

		λ_0						λ_0							
$(\phi^{*,9}, \theta^{*,9})$		0.5	1	2	3	5	10	$(\phi^{*,10}, \theta^{*,10})$		0.5	1	2	3	5	10
ϕ_1	-0.76	-0.66	-0.64	-0.75	-0.81	-0.83	-0.89	0.67	0.21	0.26	0.37	0.56	0.81	1.04	
ϕ_2	-0.46	-0.22	-0.29	-0.44	-0.45	-0.42	-0.41	0.20	0.16	0.22	0.34	0.30	0.00	-0.33	
ϕ_3	-0.59	-0.38	-0.45	-0.58	-0.55	-0.49	-0.42	-0.41	-0.05	-0.10	-0.30	-0.44	-0.29	-0.09	
ϕ_4	0.00	0.19	0.15	0.00	0.00	0.00	0.00	0.00	-0.18	-0.21	-0.14	-0.01	0.00	0.00	
ϕ_5	0.00	0.11	0.05	0.00	0.00	0.00	0.00	0.00	-0.15	-0.09	0.00	0.00	0.00	0.00	
θ_1	-0.84	-0.90	-0.92	-0.82	-0.76	-0.74	-0.62	-0.57	-0.13	-0.18	-0.29	-0.47	-0.74	-0.90	
θ_2	0.17	0.06	0.15	0.15	0.09	0.03	0.00	-0.30	-0.23	-0.30	-0.43	-0.40	-0.14	0.00	
θ_3	0.00	0.09	0.03	0.00	0.00	0.01	0.00	0.00	-0.37	-0.32	-0.12	0.00	0.00	-0.01	
θ_4	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	-0.05	0.00	0.00	0.00	0.00	0.00	
θ_5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
ϵ_{λ_0}		0.42	0.30	0.05	0.15	0.25	0.36		0.85	0.72	0.45	0.12	0.48	0.78	