Lasso Regularization Paths for NARMAX Models via Coordinate Descent*

Antônio H. Ribeiro¹ and Luis A. Aguirre²

Abstract—We propose a new algorithm for estimating NARMAX models with \( L_1 \) regularization for models represented as a linear combination of basis functions. Due to the \( L_1 \)-norm penalty the Lasso estimation tends to produce some coefficients that are exactly zero and hence gives interpretable models. The proposed algorithm uses cyclical coordinate descent to compute the parameters of the NARMAX models for the entire regularization path and, to the best of the authors knowledge, it is first the algorithm to allow the inclusion of error regressors in the Lasso estimation. This is made possible by updating the regressor matrix along with the parameter vector. In comparative timings we find that this modification does not harm the global efficiency of the algorithm and can provide a more stable estimation than the discrete process used in subset selection, where regressors are either dropped or retained in the model without midterm solutions [1].

I. INTRODUCTION

The Lasso (least absolute shrinkage and selection operator) [1] proposed in 1996 by Robert Tibshirani is a popular method for regularizing least squares regression using \( L_1 \) penalization to achieve a sparse solution. Like subset selection methods (e.g. forward-stepwise regression [2], [3]) it allows the data analyst to control the model complexity in order to avoid overfitting the data. In the case of subset selection methods, the model complexity can be restricted by limiting the number regressors to enter the model. For the Lasso technique, the regularization weight can be used to control the degrees of freedom of the model. The greatest advantage of the second approach is that it provides a continuous parameter to adjust the model complexity, which can provide a more stable estimation than the discrete process used in subset selection, where regressors are either dropped or retained in the model without midterm solutions [1].

Following its publication, intense research activity has been devoted to study its variations and extensions (e.g. the grouped Lasso [4], the elastic net [5] and the graphical Lasso [6]). Furthermore, the Lasso and related regularization methods have been successfully applied to the more diverse areas (e.g. genomic analysis [7], spectrometry-based cancer diagnosis [8], prediction of soil carbon [9], identification of biomarkers [10]) and, recently, they have received significant attention for the identification of models for dynamic systems [11], [12], [13], [14], [15], [16], [17], [18].

*This work has been supported by the Brazilian agencies CAPES, CNPq and FAPEMIG.

¹ Graduate Program in Electrical Engineering, Federal University of Minas Gerais - Av. Antônio Carlos 6627, 31270-901, Belo Horizonte. antonio-ribeiro@ufmg.br

²Department of Electronic Engineering, Federal University of Minas Gerais. aguirre@ufmg.br

In this paper we study the application of Lasso regularization in the context of prediction error methods [19]. Prediction error methods is a framework for estimating parameters of models of dynamic systems by minimizing the error between the measured and predicted values. Different noise assumptions yield different estimators. Two famous models classes estimated within this framework are the so-called NARX and NARMAX model [1].

For linear-in-the-parameter functions, NARX models estimation can be formulated as a linear regression problem, and most of the methods applicable to linear models (e.g. Lasso regularization) can be direct extended to NARX models without any modification. NARX models, however, assume that the equation error is white noise and yield non-consistent estimation of parameters whenever this is not valid [20]. Since real processes are rarely disturbed as assumed by NARX models, the obtained models will often be subject to estimator bias.

Compared with NARX, NARMAX models are more flexible because they include an error model that is estimated together with the “process” model. NARMAX models are built from more realistic noise assumptions, which allows them to yield consistent results even in the presence of colored equation errors. Nevertheless, they also results in a nonlinear regression which requires the solution of non-convex optimization problems.

When no regularization term is present, there are two common approaches to deal with this non-convex problem: to use a nonlinear least squares solver (e.g. Levenberg-Marquardt [21]); or, for basis expansion representations, to approximate the NARMAX problem by a sequence of ordinary least squares problems. This procedure is known as Extended Least Squares [2]. However, neither of these procedures are applicable when the \( L_1 \) regularization term is present in the cost function.

In this paper we propose a novel method for estimating the parameters of NARMAX models subject to \( L_1 \) regularization for models represented as a linear expansion of basis functions. The procedure is based on the coordinate descent algorithm [22] and, to the best of the authors knowledge, this is the first time error regressors are included in the Lasso estimation. This is made possible because of the good use of the problem structure in order to solve a sequence of non-convex non-differentiable problems efficiently.

¹ The acronyms are defined as: NARX: Nonlinear Autoregressive With Exogenous Input; NARMAX: Nonlinear Autoregressive Moving Average With Exogenous Input.
In the original paper [1], the Lasso solution was obtained by solving quadratic programming problems. This approach, however, did not scale very well and was not very transparent [23]. The LARS (Least Angle Regression) algorithm [24] proposed in 2004 solve the entire regularization path with a similar computational cost to the least squares algorithm. A competing approach that has been proved the most efficient (according to benchmarks presented in [22]) is to use coordinate descent optimization to find the Lasso path, by solving “one-at-a-time” unidimensional optimization problems along the coordinates. This idea has been proposed very early [25] but its potential was only fully appreciated latter, after studies and efficient implementations [26], [27], [22] demonstrated its great potential.

We propose an adaptation of the coordinate descent algorithm that deals with the non-linearities of NARMAX estimation by updating the error model along with the solution. The necessity to constantly update the error model renders some of computational tricks proposed by [26] impossible, nevertheless the implementation is efficient and applicable to a large range of problems.

The rest of the paper is organized as follows: Section II and III provides the required background on, respectively, Lasso and prediction error methods. The proposed algorithm is described in Section IV along with computational considerations. Test results and implementation details are described in Section V and final comments are provided in Section VI.

II. LASSO AND THE PATHWISE COORDINATE OPTIMIZATION ALGORITHM

A. Lasso

Consider the usual setup for linear regression, being \( X \in \mathbb{R}^{N \times p} \) a matrix containing observations of independent variables and \( y \in \mathbb{R}^N \) a vector containing the correspondent dependent variable. Furthermore, we assume that all variables have been centered and have zero mean.

The Lasso solution of this regression problem is given by the solution of the following minimization problem:

\[
\min_{\theta} \frac{1}{2} \| y - X\theta \|_2^2 + \lambda \| \theta \|_1, \tag{1}
\]

where \( \theta \in \mathbb{R}^p \) is a vector containing parameters that we wish to estimate from observation data and \( \lambda \) weights the regularization term.

This formulation produces sparse solutions due to the non-differentiability of the function being minimized and the number of non-zero terms depends on the value of \( \lambda \). The larger the value of \( \lambda \), the lesser degrees of freedom are given to the solution.

The Lasso minimization problem can be interpreted as the Lagrangian formulation of a least-squares problem subject to constraints that require the parameter \( L_1 \)-norm \( \| \theta \|_1 \) to be limited by an upper bound. Alternatively, it can be viewed as the maximum a posteriori parameter estimation considering a Laplacian prior. Further discussion about the properties and interpretations of the Lasso solution can be found in [28].

B. Pathwise Coordinate Optimization

Consider a coordinate descent step for solving (1). Be \( \theta_j \) the \( j \)-th component of the parameter vector, suppose that all the components \( \theta_i \) for \( i \neq j \) are fixed and we want to optimize (1) with respect to \( \theta_j \). Simple manipulations shows that this yields the unidimensional optimization problem:

\[
\min_{\theta_j} \frac{1}{2} \| x_j \|_2^2 - (y - \sum_{i \neq j} x_i \theta_i) \theta_j + \lambda |\theta_j| + C, \tag{2}
\]

where \( x_i \) is the \( i \)-th column of \( X \), and, \( C \) is the term containing the remaining fixed components:

\[
C = \| y - \sum_{i \neq j} x_i \theta_i \|^2 + \lambda \sum_{i \neq j} |\theta_i|. \tag{3}
\]

The analytic solution of (2) can be found by minimizing the corresponding polynomial of degree 2 for three different situations: \( \theta_j > 0 \); \( \theta_j = 0 \); and, \( \theta_j < 0 \). This yields the optimal coordinate update:

\[
\theta_j \leftarrow \frac{1}{\| x_j \|^2} S \left( (y - \sum_{i \neq j} x_i \theta_i) x_j; \lambda \right) \tag{4}
\]

where \( S(z; \lambda) \) stands for the soft-thresholding operator:

\[
S(z; \lambda) = \begin{cases} 
z - \lambda & \text{if } z > \lambda \\
0 & \text{if } -\lambda < z < \lambda \\
z + \lambda & \text{if } z < -\lambda,
\end{cases} \tag{5}
\]

Thus the algorithm applies the update (4) cyclically along the coordinates until the solution converges. Conditions for convergence of the coordinate descent algorithm for non-differentiable problems are described in [29].

C. Computing the Update

In [22] two different ways of storing and updating the computation of \( \tilde{r}_j = (y - \sum_{i \neq j} x_i \theta_i) x_j \) are discussed:

- The so-called naive update approach keeps an updated value of the residual vector \( r = y - \sum_{i=1}^p x_i \theta_i \) stored and computes \( \tilde{r}_j \leftarrow (r + x_j \theta_j)^T x_j \). This yields a computational cost of \( O(N) \) per iteration and \( O(p \cdot N) \) for each complete cycle through all \( p \) variables.
- The covariance update keeps values of \( y^T x_j \) and \( x_j^T x_j \) stored, and computes \( \tilde{r}_j \leftarrow (y^T x_j - \sum_{i \neq j} x_i^T x_j \theta_i) \). That way, each time a new variable enters the model there is an associated computational cost of \( O(p \cdot N) \) due to the computation of all dot products \( x_i^T x_j \). For the remaining iterations, however, the cost of the iteration is \( O(p \cdot m) \), where \( m < p \) is the number of non-zero variables. Hence, for the covariance approach, \( O(N) \) computations are not required at each steps.

The adapted version of coordinate descent optimization for NARMAX models described in Section IV uses the naive update because the proposed modifications renders the use of the more efficient covariance update impossible.
D. Warm Start

In [26] it is pointed out the role of warm starts in the efficient computation of the entire regularization path. The procedure consists of, starting with \( \lambda = \lambda_{\text{max}} \), computing the solution for a decreasing sequence of values of \( \lambda \), using the estimated parameter vector at the last iteration as initial guess to be refined for the current value of \( \lambda \).

Here \( \lambda_{\text{max}} \) denotes the smallest value of \( \lambda \) for which the entire parameter vector \( \theta \) is zero. A minimum value \( \lambda_{\text{min}} < \lambda_{\text{max}} \) is selected and a decreasing sequence of \( K \) values (in log-scale) between \( \lambda_{\text{max}} \) and \( \lambda_{\text{min}} \) is constructed.

It follows from (4) that if \( \lambda > |y^T x_j| \) all the updates are going to be zero. Hence, we can set \( \lambda_{\text{max}} = \max_j |y^T x_j| \). Typical values of \( K \) and \( \lambda_{\text{min}} \) are, according to [22], \( K = 100 \) and \( \lambda_{\text{min}} = 0.001 \lambda_{\text{max}} \).

E. Active Set Convergence

Some speedup can be obtained by, instead of cycling along all the \( p \) variables every time, to organize the iterations around the active set (non-zero variables) [22]. That is, after a complete cycle through all the variables, we iterate only on the active set until convergence. A new complete cycle through the complete set follows, interrupting the processes if no change on active set is found.

III. NARMAX MODELS

Consider the data set \( Z = \{(u[k], y[k]), k = 1, 2, \ldots, N\} \), containing a sequence of sampled inputs-output pairs. Here \( u[k] \in \mathbb{R}^N \) is a vector containing all the inputs and \( y[k] \in \mathbb{R} \) is the single output variable at instant \( k \).

Furthermore consider the output \( y[k] \) is correlated with its own past values \( y[k-1], y[k-2], y[k-3], \ldots \), and with past input values \( u[k-\tau_d], u[k-\tau_d-1], \ldots \), where the integer \( \tau_d \geq 0 \) is used to indicate the possibility of input-output delay (actions applied to the input take some time to appear in the system output). This paper is focused on trying to find a difference equation model that best describe the observed data.

A. Optimal Predictor

To study the previously described problem it is assumed that for a given input sequence \( u[k] \) the output was generated by a “true system”, described by the following difference equation:

\[
y[k] = F(y[k-1], \ldots, y[k-n_y], u[k-\tau_d], \ldots, u[k-n_u], v[k-1], \ldots, v[k-n_v]; \theta^*) + v[k],
\]

where \( F \) and \( \theta^* \) are the “true” function and parameter vector describing the system, \( n_y, n_u, \) and \( n_v \) are the maximum input, output and error lag and \( \tau_d \) is the input-output delay. The assumption that a finite number of past terms can be used to describe the output is implicit in this model.

Furthermore, \( v[k] \in \mathbb{R} \) is a random variable that cause the deviation of the deterministic model from its true value. We assume here that \( v[k] \) is a white random process (which implies it has zero mean and that \( v[k] \) is uncorrelated with \( v[l] \) for \( l \neq k \)). The capability of the above model to represent colored noise comes from the presence of lagged error terms \( v[k-i] \) in function \( F \) arguments.

The simplified notation \( y[k-1], y[k-1] \) and \( u[k-1], u[k-1] \) will be used to represent the vectors \( [y[k-1], \ldots, y[k-n_y]], [v[k-1], \ldots, v[k-n_v]] \) and \( [u[k-\tau_d], \ldots, u[k-\tau_d+n_u]] \). Using this new notation, equation (6) could be compactly rewritten as:

\[
y[k] = F(y_{[k-1]}, u_{[k-1]}, y_{[k-1]}; \theta^*) + v[k].
\]

If the measured values of \( y \) and \( u \) are known at all instants previous to \( k \), the optimal prediction of \( y[k] \) is the following conditional expectation:

\[
y^*_{\text{Opt}}[k] = \mathbb{E} \{ y[k] \mid y_{[k-1]}, u_{[k-1]} \},
\]

where the notation \( y^*_{\text{Opt}}[k] \) is used to denote the optimal prediction. Since \( v[k] \) is a white process with zero mean, it follows that:

\[
y^*_{\text{Opt}}[k] = E\{ F(y_{[k-1]}, u_{[k-1]}, y_{[k-1]}; \theta^*) + v[k] \mid y_{[k-1]}, u_{[k-1]} \}
\]

It follows from (6) that \( v[k] = y[k] - y^*_{\text{Opt}}[k] \). Which yields the following recurrent definition for the optimal predictor:

\[
y^*_{\text{Opt}}[k] = F(y_{[k-1]}, u_{[k-1]}, y_{[k-1]} - y^*_{\text{Opt}}[k-1]; \theta^*).
\]

B. Parameter Estimation and Linear-in-the-Parameters Functions

The parameter vector \( \theta \) of a NARMAX model can be estimated by solving:

\[
\min_{\theta} \sum_{k=1}^{N} (y[k] - \hat{y}[k])^2.
\]

where \( y[k] \) is the measured output value and the prediction \( \hat{y}[k] \) is defined similarly to (8):

\[
\hat{y}[k] = F(y_{[k-1]}, u_{[k-1]}, y_{[k-1]} - \hat{y}_{[k-1]}; \theta).
\]

This problem is non-convex and cannot be written as an ordinary least-squares problem due to the recurrent definition of \( \hat{y}[k] \). Nonlinear least-squares algorithms (e.g. Levenberg-Marquardt algorithm [21]) can be used to find the solution. Alternatively, for specific representations, an algorithm called Extended Least Squares [2] is also applicable.

Consider that \( F \) can be written according to a basis expansion:

\[
F(y_{[k-1]}, u_{[k-1]}, y_{[k-1]}) = \sum_{i=1}^{p} \theta_i x_i(y_{[k-1]}, u_{[k-1]}, y_{[k-1]}),
\]

where the variable \( e \) is being used to indicate the difference \( e[k] = y[k] - \hat{y}[k] \) and \( x_i(e) \) is a linear or nonlinear transformation (e.g. a linear relation: \( x_i = y[k] - 5 \), monomial term: \( x_i = y[k]^2 \), \( k \) is a linear or \( k \) is a nonlinear transformation: \( x_i = \tanh(y[k]-1) \)).

\(^3\)The expectation is the optimal prediction in the sense that the expected squared prediction error is minimized [30, p.18, Sec. 2.4].
It follows that the minimization problem (9) can be rewritten as:

$$
\min_{\theta} \sum_{k=1}^{N} \left( y[k] - \sum_{i=1}^{p} \theta_i \cdot x_i(y[k-1], u[k-1], e[k-1]) \right)^2,
$$
or, in matricial form:

$$
\min_{\theta} \| y - X_{(y,u,e)} \theta \|^2,
$$

for which $y \in \mathbb{R}^N$ is a vector containing $y[k]$ as its elements; and, $X_{(y,u,e)} \in \mathbb{R}^{N \times p}$ is a matrix with the elements $x_i(y[k-1], u[k-1], e[k-1])$ organized in such way the index $k$ grows along the matrix rows and the index $i$ along the matrix columns.

Notice that $X_{(y,u,e)}$ depends on $e[k] = y[k] - \hat{y}[k]$, which, in turn, depends on the parameter vector $\theta$. Hence, the minimization problem cannot be solved using the ordinary least squares algorithm. An alternative procedure that works for this situation is the extended least squares (Algorithm 1) that estimate the parameters by consecutively solving linear least squares problems, approximating $e$ by the residual vector $r$.

**Algorithm 1 (Extended Least Squares).** Given a initial guess for the residual vector $r^{(0)}$: Do the following computations starting in $i = 0$ until $\| \theta^i - \theta^{i-1} \|_{\infty} < $ tolerance:

1. Compute the matrix $X_{(y,u,r^{(i)})}$.
2. Find $\theta^{i+1}$ that minimizes $\| y - X_{(y,u,r^{(i)})} \theta^i \|^2$.
3. Update $r^{(i+1)} \leftarrow y[k] - X_{(y,u,r^{(i)})} \theta^{i+1}$.
4. Set $i \leftarrow i + 1$.

C. NARX models

NARX models can be interpreted as an special case of NARMAX models for which there are no noise terms $e[k]$ to estimate. In this case (12) simplifies to an ordinary least squares problems.

While computationally more efficient to estimate, NARX models may yield biased results in the presence colored noise [20]. The extra flexibility to deal with different types of noise justify the choice of NARMAX models over NARX models in some situations despite its extra computational cost.

IV. COORDINATE OPTIMIZATION ALGORITHM APPLIED TO NARMAX MODELS

The Lasso regression problem that arises when estimating NARMAX models (for basis expansion representations) requires the solution of the following minimization problem:

$$
\min_{\theta} \| y - X_{(y,u,e)} \theta \|^2 + \lambda \| \theta \|_1.
$$

A coordinate descent algorithm to solve the above minimization problem for a decreasing sequence of $\lambda$ is presented in Algorithm 2. This algorithm is similar to what was described in Section II for linear problems. It considers, however, that the matrix $X_{(y,u,e)}$ varies along the iterations.

This algorithm keeps stored an updated version of the residual $r$ and updates the matrix $X_{(y,u,e)}$ by approximating $e$ by the current estimate of $r$.

As mentioned in Section II-C the more efficient covariance update is not applicable for NARMAX models because the matrix $X_{(y,u,e)}$ changes along the iterations. Hence, Algorithm 2 uses the naive update approach.

Besides using the naive update approach, the only modification that was made in order to cope with NARMAX models is the update of the $j$-th column of $X_{(y,u,e)}$ every iteration (step 1-a). Assuming that the update of each element of the matrix $X_{(y,u,e)}$ has a computational cost of $O(1)$, the cost of updating an entire column is $O(N)$ and therefore the asymptotic cost of each iteration is not altered by this step.

**Algorithm 2 (NARMAX Coordinate Optimization).** Set $r \leftarrow y$, $\theta \leftarrow 0$ and $\lambda \leftarrow \lambda_{\text{max}}$.

1) Compute the solution of (13) for the given value of $\lambda$ by repeating the following steps until a convergence criteria is met (e.g. $\| \theta^i - \theta^{i-1} \|_{\infty} < $ tolerance):

a) Update the $j$-th column of the matrix $X_{(y,u,e)}$ considering $e$ equals to the current estimate of the residual $r$. Call this column vector $x_j$

b) Find the next value of $\theta_j$ according to:

$$
\theta^+ \leftarrow \frac{1}{\| x_j \|^2} S (r + x_j \theta_j^T x_j; \lambda).
$$

c) Update the residual:

$$
r \leftarrow r - x_j (\theta^+ - \theta_j).$$

d) Update the parameter $\theta_j \leftarrow \theta^+$.

e) Update the index $j$. As discussed in Section II-C this update can be done such that $j$ circles through all the $p$ variables on a first step and, after that, iterates on the active set until convergence.

2) Store the estimated parameter vector and decrease the value of lambda $\lambda$. Keep the values of $\theta$, $r$ and $X_{(y,u,e)}$ to be used as warm start for the next iteration.

Practical aspects of this implementation are discussed in the next session.

V. IMPLEMENTATION AND TEST RESULTS

Next we present numerical examples illustrating the method. In these examples we focus exclusively on linear and polynomial representations. The algorithm was implemented in Julia and the code to run the examples is available in the GitHub repository: https://github.com/antonior92/NarmaxLasso.jl.

A. Example 1: Linear Model

The follow linear system:

$$
y[k] = 0.5y[k-1] - 0.5u[k-1] + 0.5v[k-1] + v[k],
$$

is simulated for a sequence of randomly generated inputs $u$. The values are draw from a standard Gaussian distribution and each generated value is held for $5$ samples. And $v$ is a white Gaussian process with standard deviation $\sigma_v = 0.3$.

A window of $2000$ samples is used for training and a different realization with $1000$ samples is used to validate the model.
Algorithm 2 is used to find the parameter vector that minimizes the value of \( \lambda \) which yields the smallest validation results. For high values of \( \lambda \) most of the parameters are zero and, as we decrease it, more and more terms are included. The values \( y[k-1] \), \( u[k-1] \) and \( e[k-1] \) are the first terms to enter in the model and, as \( \lambda \) approaches zero, other terms starts to enter in the model as well.

We simulate each of the obtained models in the validation window and select the value of \( \lambda \) that yields the smallest sum of absolute errors between the estimated model free-run simulation and the observed values. This value of \( \lambda \) is indicated by a dashed vertical line in the figure.

For this value of \( \lambda \) the estimated model is:

\[
y[k] = 0.48 y[k-1] - 0.50 u[k-1] + 0.44 e[k-1].
\]

If the noise terms \( e[k-i] \) were not included in the initial model (17), an identical procedure, would yield:

\[
y[k] = 0.78 y[k-1] - 0.17 y[k-2] + 0.01 y[k-5] - 0.46 u[k-1] + 0.09 u[k-2].
\]

B. Example 2: Nonlinear Model

Consider the non-linear system: [31]

\[
y[k] = (0.8 - 0.5 \exp(-y[k-1]^2))y[k-1] + u[k-1] - (0.3 + 0.9 \exp(-y[k-1]^2))y[k-2] + 0.2 u[k-2] + 0.1 u[k-2] + 0.1 e[k-1] + 0.3 e[k-2] + v[k],
\]

for which, the values of \( u \) are draw from a standard Gaussian distribution and held for 5 samples. And \( v \) is a white Gaussian process with standard deviation \( \sigma_v = 0.5 \).

A window of 1000 samples is used for training a polynomial model and one with 500 samples to validate it.

The following polynomial model will be adjusted to the training set:

\[
y[k] = \sum_i \theta_i (y[k-q_i])^{t_i} (u[k-t_i])^{r_i} (e[k-w_i])^{s_i}
\]

(18)

For which the monomials included as regressors are all possible monomials for which: \( 1 \leq q_i \leq n_y; 1 \leq t_i \leq n_u; 1 \leq w_i \leq n_e; \) and, \( l_i + r_i + s_i \leq n_{\text{order}} \). In this example, we have used \( n_y = 3, n_u = 3, n_e = 2 \) and \( n_{\text{order}} = 2 \), which yields a total number of regressors \( p = 44 \).
TABLE I: Algorithm 2 running timings (in seconds) for different settings. The data used for training was generated as described in Example 2.

| Parameter Settings | n_y, n_u, n_e | \(\lambda_{\min} = 10^{-2}\lambda_{\max}, K = 100\) | \(\lambda_{\min} = 10^{-4}\lambda_{\max}, K = 200\) | \(\lambda_{\min} = 10^{-6}\lambda_{\max}, K = 300\) |
|--------------------|---------------|-----------------------------------|-----------------------------------|-----------------------------------|
| \(n_{\text{order}}\) | \(p\) | \(p_u/p\) | \(N = 100\) | \(N = 100\) | \(N = 100\) | \(N = 100\) | \(N = 100\) | \(N = 100\) | \(N = 100\) | \(N = 100\) | \(N = 100\) |
| 2 | 27 | 0.00 | 0.02 | 0.03 | 0.34 | 1.55 | 1.65 | 12.72 | 2.73 | 2.89 | 39.57 |
| 3 | 83 | 0.00 | 0.00 | 0.03 | 0.63 | 1.59 | 1.64 | 20.41 | 9.37 | 28.29 | 488.03 |
| \(n_{\text{order}}\) | \(p\) | \(p_u/p\) | \(N = 100\) | \(N = 100\) | \(N = 100\) | \(N = 100\) | \(N = 100\) | \(N = 100\) | \(N = 100\) | \(N = 100\) | \(N = 100\) |
| 2 | 27 | 0.00 | 0.02 | 0.03 | 0.36 | 3.07 | 5.89 | 112.98 | 7.37 | 30.31 | 347.54 |
| 3 | 83 | 0.00 | 0.00 | 0.02 | 3.32 | 5.92 | 19.58 | 500.38 | 27.57 | 144.05 | 2724.84 |
| \(n_{\text{order}}\) | \(p\) | \(p_u/p\) | \(N = 100\) | \(N = 100\) | \(N = 100\) | \(N = 100\) | \(N = 100\) | \(N = 100\) | \(N = 100\) | \(N = 100\) | \(N = 100\) |
| 2 | 27 | 0.00 | 0.02 | 0.03 | 0.36 | 3.07 | 5.89 | 112.98 | 7.37 | 30.31 | 347.54 |
| 3 | 83 | 0.00 | 0.00 | 0.02 | 3.32 | 5.92 | 19.58 | 500.38 | 27.57 | 144.05 | 2724.84 |

Again, Algorithm 2 is used to find the Lasso solution for a sequence of decreasing values of \(\lambda\). Figure 2(a) illustrate the obtained regularization paths. The linear terms that appear in the estimated parameter vector as a function of \(\lambda\) significantly different from zero, the regressors correspond to the linear terms annotated in the Table. This is true for smaller values of \(\lambda\) that under similar conditions the time grows with both the number of points \(n_y\), \(n_u\) and \(n_e\) and order \(n_{\text{order}}\) is fitted to this data set. The total number of regressors is denoted by \(p\) and consist of all possible combinations of monomials within this lag and order constraints. The fraction of monomials that contains some error term is denoted by \(\frac{\|\theta - \hat{\theta}\|_2}{\|\theta\|_2}\). Furthermore, different parameters of \(\lambda_{\min}\) and \(K\) are used in the different experiments. The stop criteria used is \(\|\theta^k - \theta^{k-1}\|_\infty < 10^{-7}\). All timings were carried out on an Intel Core i7-4790K 4.00GHz processor.

The more obvious point that can be taken from Table I is that under similar conditions the time grows with both the data set length \(N\) and the number of regressors \(p\).

The run time also grows if we decrease \(\lambda_{\min}\) and increase the number of points \(K\). That is because: i) the increase on \(K\) produces more values of \(\lambda\) to be evaluated; and, ii) for smaller values of \(\lambda\) the number of non-zero parameters increase, and the speed up provided by iterating only on the active set (described in Section II-E) lose its effect. The importance of effect (ii) can be observed on the above table by noticing that under similar conditions the simultaneous variation of \(\lambda_{\min}\) and \(K\) often results in a much greater increase on the running time than what the increment on the number of points \(K\) could account for.

It follows from the above discussion that the algorithm computes the first terms to enter the active set very efficiently due to the sparse structure of the solution. Hence, a subset of the regressors parameters can usually be efficiently computed in few inexpensive iterations.

Algorithm 2 modifies the original coordinate descent algo-
rithm by introducing the step 1-(a), which require the regressor matrix columns to be updated along the iterations. The fraction $p_{e}/p$ gives the number of columns which actually requires to be updated. In Table III, it is possible to find entries that have increasing values of $p_{e}/p$ for similar configurations and the run time does not consistently grows with it. This is a good indicator that the modifications we introduced in the algorithm are not critical to the total computation time and that other aspects, as the correlation between the variables, may have a much greater influence on the total running time.

VI. Final Comments

In this paper, we proposed a new pathwise coordinate descent algorithm for estimating NARMAX models with $L_1$-norm regularization. To the best of authors' knowledge, it is the first algorithm to consider the inclusion of error terms in the Lasso regression problem. The time results on Section VC suggests that the proposed modification does not harm the global efficiency of the algorithm and that the computation is especially efficient when only the more important terms to enter the model are required, as it is often the case.

Like Extended Least Squares, the algorithm uses heuristics that make it very hard to establish mathematical convergence properties. Nevertheless, the algorithm have converged to meaningful solutions in all tested situations.

While we have focused on the Lasso, the procedure could easily be adapted to elastic net penalties, using a similar reasoning as the one used in [22]. The algorithm seems to be very promising and the results presented here suggest it might prove to be useful in a variety of identification problems.

ACKNOWLEDGMENTS

This work has been supported by the Brazilian agencies CAPES, CNPq and FAPEMIG.

REFERENCES

[1] R. Tibshirani, “Regression shrinkage and selection via the LASSO,” Journal of the Royal Statistical Society. Series B (Methodological), pp. 267–288, 1996.

[2] S. Chen, S. A. Billings, and W. Luo, “Orthogonal least squares methods and their application to non-linear system identification,” International Journal of Control, vol. 50, no. 5, pp. 1873–1896, 1989.

[3] S. A. Billings, S. Chen, and M. J. Korenberg, “Identification of MIMO non-linear systems using a forward-regression orthogonal estimator,” International Journal of Control, vol. 49, no. 6, pp. 2157–2189, 1989.

[4] L. Meier, S. van de Geer, and P. Bühlmann, “The Group Lasso for Logistic Regression,” Journal of the Royal Statistical Society. Series B (Statistical Methodology), vol. 70, no. 1, pp. 53–71, 2008.

[5] H. Zou and T. Hastie, “Regularization and Variable Selection via the Elastic Net,” Journal of the Royal Statistical Society. Series B (Statistical Methodology), vol. 67, no. 2, pp. 301–320, 2005.

[6] J. Friedman, T. Hastie, and R. Tibshirani, “Sparse inverse covariance estimation with the graphical lasso,” Biostatistics, vol. 9, no. 3, pp. 432–441, 2008.

[7] G. Nowak, T. Hastie, J. R. Pollack, and R. Tibshirani, “A fused lasso latent feature model for analyzing multi-sample aCGH data,” Biostatistics, vol. 12, no. 4, pp. 776–791, Oct. 2011.

[8] S. Banerjee, R. N. Zare, R. J. Tibshirani, C. A. Kundra, R. Nolley, R. Fan, J. D. Brooks, and G. A. Sonn, “Diagnosis of prostate cancer by desorption electrospray ionization mass spectrometric imaging of small metabolites and lipids,” Proceedings of the National Academy of Sciences, vol. 114, no. 13, pp. 3334–3339, Mar. 2017.

[9] C. H. Wilson, T. T. Caughlin, S. W. Rifai, E. H. Boughton, M. C. Mack, and S. L. Flory, “Multi-decadal time series of remotely sensed vegetation improves prediction of soil carbon in a subtropical grassland,” Ecological Applications, vol. 27, no. 5, pp. 1646–1656, Jul. 2017.

[10] N. Temès, F. Rotolo, G. Heinze, and S. Michiels, “Identification of biomarker-by-treatment interactions in randomized clinical trials with survival outcomes and high-dimensional spaces,” Biometrical Journal, vol. 59, no. 4, pp. 685–701, Jul. 2017.

[11] G. C. Calaﬁore, C. Novara, and M. Taragana, “Leading impulse response identification via the Elastic Net criterion,” Automatica, vol. 80, Supplement C, pp. 75–87, Jun. 2017.

[12] G. C. Calaﬁore, L. M. El Ghoul, and C. Novara, “Sparse parameterization of posynomial models,” Automatica, vol. 59, no. Supplement C, pp. 27–34, Sep. 2015.

[13] G. C. Calaﬁore, C. Novara, and M. Taragana, “Leading impulse response identification via the weighted elastic net criterion,” in 2016 IEEE 55th Conference on Decision and Control (CDC), Dec. 2016, pp. 2926–2931.

[14] S. L. Kukreja, J. Löfberg, and M. J. Brenner, “A least absolute shrinkage and selection operator (LASSO) for nonlinear system identification,” IFAC proceedings volumes, vol. 39, no. 1, pp. 814–819, 2006.

[15] G. C. Calaﬁore, L. E. Ghoul, and C. Novara, “Sparse Identification of Polynomial and Posynomial Models,” 9th IFAC World Congress, vol. 47, no. 3, pp. 3238–3243, Jan. 2014.

[16] M. Bonin, V. Seghezza, and L. Piroddi, “NARX model selection based on simulation error minimisation and LASSO,” IET control theory & applications, vol. 4, no. 7, pp. 1157–1168, 2010.

[17] J. Friedman, T. Hastie, and R. Tibshirani, “LASSO-enhanced simulation error minimization method for NARX model selection,” in American Control Conference (ACC), 2010. IEEE, 2010, pp. 4522–4527.

[18] C. Cantelmo and L. Piroddi, “Adaptive model selection for polynomial NARX models,” IET Control Theory & Applications, vol. 4, no. 12, pp. 2693–2706, Dec. 2010.

[19] L. Ljung, System Identification. Springer, 1998.

[20] O. Nelles, Nonlinear System Identification: From Classical Approaches to Neural Networks and Fuzzy Models. Springer Science & Business Media, 2001.

[21] D. W. Marquardt, “An algorithm for least-squares estimation of non-linear parameters,” Journal of the Society for Industrial and Applied Mathematics, vol. 11, no. 2, pp. 431–441, 1963.

[22] J. Friedman, T. Hastie, and R. Tibshirani, “Regularization paths for generalized linear models via coordinate descent,” Journal of statistical software, vol. 33, no. 1, p. 1, 2010.

[23] R. Tibshirani, “Regression shrinkage and selection via the lasso: A retrospective,” Journal of the Royal Statistical Society: Series B (Statistical Methodology), vol. 73, no. 3, pp. 273–282, 2011.

[24] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, “Least angle regression,” The Annals of Statistics, vol. 32, no. 2, pp. 407–499, Apr. 2004.

[25] W. J. Fu, “Penalized Regressions: The Bridge versus the Lasso,” Journal of Computational and Graphical Statistics, vol. 7, no. 3, pp. 397–416, 1998.

[26] J. Friedman, T. Hastie, H. Höfling, and R. Tibshirani, “Pathwise coordinate optimization,” The Annals of Applied Statistics, vol. 1, no. 2, pp. 302–332, Dec. 2007.

[27] J. Friedman, T. Hastie, and R. Tibshirani, “Glmnet: Lasso and elastic-net regularized generalized linear models,” R package version 1.0, 2009.

[28] T. Hastie, R. Tibshirani, and M. Wainwright, Statistical Learning with Sparsity: The Lasso and Generalizations. CRC press, 2015.

[29] P. Tseng, “Convergence of a block coordinate descent method for nondifferentiable minimization,” Journal of optimization theory and applications, vol. 109, no. 3, pp. 475–494, 2001.

[30] T. Hastie, R. Tibshirani, and J. Friedman, Elements of Statistical Learning. 2nd ed. Springer Science & Business Media, 2009.

[31] S. Chen, S. A. Billings, and P. M. Grant, “Non-linear system identification using neural networks,” International Journal of Control, vol. 51, no. 6, pp. 1191–1214, 1990.