RESEARCH ARTICLE

LASSO vector autoregression structures for very short-term wind power forecasting

Laura Cavalcante¹, Ricardo J. Bessa¹, Marisa Reis¹ and Jethro Browell²

¹ INESC Technology and Science (INESC TEC), Campus da FEUP, Rua Dr. Roberto Frias, Porto 4200-465, Portugal
² Royal College Building, University of Strathclyde, 204 George Street, Glasgow, Scotland

ABSTRACT

The deployment of smart grids and renewable energy dispatch centers motivates the development of forecasting techniques that take advantage of near real-time measurements collected from geographically distributed sensors. This paper describes a forecasting methodology that explores a set of different sparse structures for the vector autoregression (VAR) model using the least absolute shrinkage and selection operator (LASSO) framework. The alternating direction method of multipliers is applied to fit the different LASSO-VAR variants and create a scalable forecasting method supported by parallel computing and fast convergence, which can be used by system operators and renewable power plant operators. A test case with 66 wind power plants is used to show the improvement in forecasting skill from exploring distributed sparse structures. The proposed solution outperformed the conventional autoregressive and vector autoregressive models, as well as a sparse VAR model from the state of the art. Copyright © 2016 John Wiley & Sons, Ltd.

KEYWORDS

wind power; vector autoregression; scalability; sparse; renewable energy; parallel computing

Correspondence

Ricardo J. Bessa, INESC Technology and Science (INESC TEC), Campus da FEUP, Rua Dr. Roberto Frias, Porto 4200-465, Portugal.
E-mail: ricardo.j.bessa@inesctec.pt

Received 8 March 2016; Revised 11 July 2016; Accepted 23 August 2016

1. INTRODUCTION

Operating a power system with high integration levels of wind power is challenging and demands for a continuous improvement of wind power forecast tools.¹,² Furthermore, the participation of wind power in the electricity market also requires accurate forecasts in order to mitigate financial risks associated to energy imbalances.³,⁴

The recent advent of smart grid technologies will increase the monitoring capability of the electric power system.⁵ Furthermore, the investment in renewable energy dispatch centers enables real-time acquisition of time series measurements from wind power plants (WPPs).⁶ The availability of the most recent WPP measurements improves the forecast skill during the first lead times, commonly called very short-term horizon.⁷

For this time horizon, it is generally established that statistical models are more accurate than physical models, while for longer time horizons the most relevant inputs come from numerical weather prediction (NWP) models.⁷ Even recent advances in physical models, such as the high-resolution rapid refresh model developed by U.S. National Oceanic and Atmospheric Administration, are outperformed by statistical models that use recent WPP observations.⁸

In the state of the art, a broad family of statistical models are available for the very short-term horizon. Two examples are the conditional parametric autoregression (AR) and regime-switching models that incorporate online observed local variables (i.e., wind speed and direction) to reduce the wind power forecast error for 10 min-ahead forecasting.⁹ Another example is the use of automatic self-tuning Kalman filters that incorporate NWP information.¹⁰

In this context, information from WPP time series distributed in space can be used to improve the forecast skill of each WPP. The first results were presented by Gneiting et al. for 2 h-ahead wind speed forecasting.¹¹ The authors showed that a regime-switching space-time diurnal model that takes advantage of temporal and spatial correlation from geographically dispersed meteorological stations as off-site predictors can have a root mean square error (RMSE) 28.6% lower than the persistence forecasts. Expert knowledge and empirical results were used to select the predictors. In Hering and Genton,¹² two additional statistical models are proposed: trigonometric direction diurnal model and bivariate skew- model. These
results were generalized by Tastu et al. by studying the spatiotemporal propagation of wind power forecast errors.\textsuperscript{13} The authors showed evidences of cross-correlation functions with significant dependency in lags of a few hours.

These works motivated the appearance of recent research that explores information from neighboring WPP. Figure 1 groups the state-of-the-art methods applied to wind power by category.

The first group consists of machine learning methods, such as artificial neural networks. To the authors’ knowledge, there is little research concerning the application of machine learning models to this problem. In Kou et al.,\textsuperscript{14} it is described a online sparse Bayesian model based on warped Gaussian process to generate probabilistic wind power forecasts. A sparsification strategy is used to reduce the computational cost, and the model includes wind speed observations from nearby WPP and NWP data. Also in this category, but applied to solar power forecasting, in Vaz et al.,\textsuperscript{15} multilayer perceptron neural networks are used to combine measurements of neighboring PV systems, and in Bessa et al.,\textsuperscript{16} component-wise gradient boosting is used to explore PV observations from a smart grid.

The following limitations were identified for this first group: (i) a separated model is fitted to each location, which increases the computational time; (ii) the scalability of the solution decreases when the number of predictor increases and (iii) with the exception of the sparse Bayesian model, the others do not provide a sparse vector of coefficients.

The second group consists of random fields. To the authors’ knowledge, the only work that explores this theory is from Wytock and Kolter.\textsuperscript{17} The model is based on sparse Gaussian conditional random field and uses a new second-order active set method to solve the problem. The main limitation of the method is that it requires a copula transformation in order to have a Gaussian marginal distribution, which might not solve the boundary problem of variables with limited support (e.g., wind power between zero and rated power). Moreover, the computational time for a solution with high accuracy is around 160 min for a case study with seven WPPs.\textsuperscript{18}

The third group is related to classical time series theory. Tastu et al. extended their previous work in Tastu et al.\textsuperscript{13} to the multivariate framework,\textsuperscript{19} i.e., from an AR to a vector AR (VAR) model. The VAR coefficients are allowed to vary with external variables, average wind direction in this case. The main limitation is a non-sparse matrix of coefficients since feature selection is not performed. A similar methodology was applied in Tastu et al.\textsuperscript{20} to generate probabilistic forecast based on geographically distributed sensors. Also in this case, the predictors are manually selected based on cross-correlation analysis.

He et al. presents a two-stage approach:\textsuperscript{21} (i) offline spatial–temporal analysis carried out on historical data with multiple finite-state Markov chains and (ii) online forecasting by feeding a Markov chain with real-time measurements of the wind turbines. Similar to previous works, different sparse structures of the spatial–temporal relations are not fully explored. The same authors in He et al.\textsuperscript{22} propose a different approach based on VAR model fitting with sparsity-constrained maximum likelihood. The main limitation of this approach is that the sparse coefficients are not automatically defined, instead, expert knowledge and partial correlation analysis are employed.

Aiming to generate forecasts on a large spatial scale, e.g. hundreds of locations, Dowell and Pinson proposed the sparse-VAR (sVAR) approach for 5 min-ahead forecasts.\textsuperscript{23} The sVAR method generates probabilistic forecasts based on the logit-normal distribution,\textsuperscript{24} whose mean is estimated with a VAR model and variance by a modified exponential smoothing. A state-of-the-art technique from Davis et al.\textsuperscript{25} is employed to fit a VAR model with a sparse coefficient matrix. The work proposed in the present paper is closely related to the sVAR and provides the following original contributions:
1. Explores a set of different sparse structures for the VAR framework using the least absolute shrinkage and selection operator (LASSO) framework;26
2. Applies the alternating direction method of multipliers (ADMM)27 to fit the different LASSO-VAR variants;
3. Proposes a scalable forecasting method based on parallel computing, fast convergence optimization algorithm and matrix calculations.

The proposed method will be compared with the sVAR approach in terms of advantages and limitations, applied to a case study with 66 WPPs located in the same control area. It should be stressed that the proposed approach is compatible with previous works from the literature. For instance, it can be used for spatial–temporal correction of forecast errors,13 extended to conditional VAR,19 or used to generate probabilistic forecasts based on the logit-normal distribution.23

The paper is organized as follows. Section 2 presents the different sparse structures for the VAR model. Section 3 describes the application of the ADMM method to fit the VAR model in its different LASSO variants. The test case results are presented in Section 4. Section 5 presents the conclusion and future work.

2. SPARSE STRUCTURES FOR THE VAR MODEL

The VAR model allows a simultaneous forecast of the wind generation in several neighboring sites combining time series information. However, forecasting with VAR models may be intractable for high-dimensional data since the non-sparse coefficients matrix grows quadratically with the number of series included in the model. In order to overcome this limitation, in Hsu et al.28 it is proposed the combination of LASSO and VAR frameworks, which is further explored in this paper for very short-term forecasting of wind power.

2.1. Formulation of the forecasting problem

The VAR model allows us to model the joint dynamic behavior of a collection of WPPs by capturing the linear interdependencies between its time series. In this multivariate (or spatiotemporal) framework, the future trajectory of output from WPPs in the model, given by

\[ y_{it} = \mu + \sum_{l=1}^{p} \beta(l) \cdot y_{i,t-l} + \epsilon_t \quad (1) \]

where \( \beta(1), \ldots, \beta(p) \) are the model coefficients, \( \mu \) is a constant (or intercept) term, \( p \) is the order of the AR model, and \( \epsilon_t \) is a contemporaneous white noise (or residuals) with zero mean and constant variance \( \sigma^2 \).

Let \( \{Y_t\} = \{(y_{1,t}, y_{2,t}, \ldots, y_{k,t})\} \), denote a \( k \)-dimensional vector time series. Modeling it as a vector AR process of order \( p \) (VAR\(_p\)), we obtain an expression relating the future observations at each of the \( k \) WPPs to the past observations of all WPPs in the model, given by

\[ Y_t = \eta + \sum_{l=1}^{p} B(l) \cdot Y_{t-l} + \epsilon_t \quad (2) \]

in which \( \eta \) is a vector of constant terms, each \( B(l) \in \mathbb{R}^{k \times k} \) represents a coefficient matrix related to the lag \( l \) and \( \epsilon_t \sim (0, \Sigma_e) \) denotes a white noise disturbance term.

In order to obtain a compact matrix notation, let \( Y = (Y_1, Y_2, \ldots, Y_T) \) define the \( k \times T \) response matrix, \( B = (B(1), B(2), \ldots, B(p)) \) the \( k \times kp \) matrix of coefficients, \( Z = (Z_1, Z_2, \ldots, Z_T) \) the \( kp \times T \) matrix of explanatory (or predictors) variables, in which \( Z_t = (Y'_{t-1}, Y'_{t-2}, \ldots, Y'_{t-p}) \) and \( E = (\epsilon_1, \epsilon_2, \ldots, \epsilon_T) \) the \( k \times T \) error matrix. To simplify the notation, consider \( m = kp \). Then it is possible to express (2) as

\[ Y = \eta1' + BZ + E \quad (3) \]

with \( 1 \) denoting a \( T \times 1 \) vector of ones.

The matrix of unknown coefficients needs to be correctly estimated to obtain the model that ‘best’ characterizes the data. Commonly, this is achieved using the least squares statistical methodology by choosing the coefficients that minimize the sum of squared errors. The predictor that will be deduced gives, for a given sample, the in-sample forecasts of the variable of interest.
Usually this methodology is applied with centered variables instead of the original ones. This allows simplifications in the calculation, including the model handling without intercept term. The intercept can be easily estimated after the model has been fitted. As a result, and assuming centered variables \( Y \) and \( Z \), \( \eta \) will no longer appear in the least squares objective function.

The multi-period forecasts can be generated with two alternative strategies, iterative or direct approach.\(^{29}\) In this paper, a direct approach, in which a specific model is created for each lead time, is adopted to generate 6 h-ahead wind power forecasts.

### 2.2. Sparse structures with LASSO

This section presents a set of different sparse structures for the LASSO-V AR model, inspired by Nicholson et al.\(^{30}\) to capture the dynamics of the underlying system.

The LASSO framework is powerful and convenient to use when handling high-dimensional data. The loss function is a regularized version of least squares that introduces an \( L_1 \) penalty on the coefficients. The penalty function shrinks some of the coefficients to zero, performing variable selection and producing a sparse solution. Instead of assuming that all the predictors are contributing to the model, this framework extracts the most important predictors, i.e., those with the strongest contribution to the prediction of the target variable.

Let \( \| \cdot \|_r \) represents both vector and matrix \( L_r \) norms. The standard LASSO-V AR (sLV) loss function is expressed as\(^{30}\)

\[
\frac{1}{2} \| Y - BZ \|_2^2 + \lambda \| B \|_1
\]

where \( \lambda > 0 \) is a scalar regularization (or penalty) parameter controlling the amount of shrinkage.

The \( L_1 \) penalty works as a sparsity-inducing term over the individual entries of the coefficient matrix \( B \), zeroing some of them in an element-wise manner.

Since the same predictors are available for each target variable (each WPP), the VAR coefficients can be estimated with ordinary least squares applied independently for the regression of each individual target variable.\(^{31}\) The problem is then re-formulated for each row of the matrix \( Y \), with a different penalization parameter for each, resulting in a separable loss function for each variable.

The main advantage of this approach, here called row LASSO-V AR (rLV), is the possibility of distributed computing, since each equation can be solved in parallel. Its loss function can be expressed as

\[
\frac{1}{2} \| Y_i - B_i Z \|_2^2 + \lambda \| B_i \|_1
\]

where \( Y_i \) and \( B_i \), \( i = 1, \ldots, k \), correspond to the \( i \)th rows of the \( Y \) and \( B \) matrices, respectively.

An alternative to deal with model’s coefficients individually, which results in an unstructured sparsity pattern, is to make some simple modifications to the sLV penalty in order to capture different sparsity patterns accordingly to the inherent structure of the VAR.\(^{30}\) These modifications produce more interpretable models that offer great flexibility in the detection of the true underlying dynamics of the system, which is especially fruitful in the high-dimensional context.

To take into account characteristics such as lag selection, within-group sparsity, delineation between a component’s own lags and those of another component and evaluate which variables add forecast improvement, the following LASSO-V AR sparse structures are explored: lag-group LASSO-V AR (lLV), lag-sparse-group LASSO-V AR (lsLV), own/other-group LASSO-V AR (ooLV) and causality-group LASSO-V AR (cLV). These LASSO schemes look through the sparsity in distinct group structures trying to find the ideal sparsity pattern.

The lLV model considers the coefficients grouped by their time lags and looks for time lags that add forecast improvement. Its objective function is

\[
\frac{1}{2} \| Y - BZ \|_2^2 + \lambda \sum_{l=1}^p \| B^{(l)} \|_2
\]

where each \( B^{(l)} \) is a sub-matrix containing the lag \( l \) coefficients.

This structure can be relevant if the interest is to perform lag selection. However, although it is advantageous when all time series tend to exhibit similar dynamics, it might be too restrictive for certain applications since all the coefficients of some lags are not considered in the prediction, and sometimes inefficient by including the entire lag if only few coefficients are significant.

In an attempt to overcome some of these limitations, the lsLV model adds within-group (or lag) sparsity to the lLV through the loss function

\[
\frac{1}{2k} \| Y - BZ \|_2^2 + (1-\alpha) \lambda \sum_{l=1}^p \| B^{(l)} \|_2^2 + \alpha \| B \|_1
\]

Usually this methodology is applied with centered variables instead of the original ones. This allows simplifications in the calculation, including the model handling without intercept term. The intercept can be easily estimated after the model has been fitted. As a result, and assuming centered variables \( Y \) and \( Z \), \( \eta \) will no longer appear in the least squares objective function.
where $0 \leq \alpha \leq 1$ is a parameter regulating the trade-off between the group and within-group importance.

As can be easily seen, the ILV and the sLV are obtained considering $\alpha = 0$ and $\alpha = 1$, respectively. Here, as proposed in Nicholson et al., the within-group sparsity is estimated based on the number of time series/variables, and set as $\alpha = 1/(k+1)$. In this sense, as the number of variables increases, the greater the group-wise sparsity and smaller the sparsity within-group. This variant allows to explore the significance of each lag and, at the same time, access the importance of each coefficient within each lag.

The ooLV model concerns with the possibility that, in many settings, the prediction of a variable is more influenced by their own past observations than by past observations of other variables. To address this question in a lag context, the coefficients of each $B_l$ are grouped by the diagonal entries representing variable’s own lags, and by off-diagonal entries representing cross dependencies with other variables, using the loss function

$$
\frac{1}{2} \| Y - BZ \|_2^2 + \sqrt{k} \lambda \sum_{l=1}^p \left\| \text{diag} \left( B^{(l)} \right) \right\|_2 + \sqrt{k(k-1)} \lambda \sum_{l=1}^p \| B^{(l-)} \|_2 \tag{8}
$$

where $B^{(l-)} = \{ [B^{(l)}]_i : i \neq j \}$. Since the groups differ in cardinality, it is necessary to weight the penalty accordingly to avoid favoring the larger groups of off-diagonal entries.

If all time series do not share the same dynamics, one may be interested in finding which of them do. Recent studies have been addressing these question considering causal structures in multivariate series, also called Granger causality. The idea is that a time series $y_i$ is Granger-caused by other time series $y_j$ if knowing the past values of $y_j$ helps to improve the prediction of $y_i$. With the intention of learn a causal inference from the data, the cLV model groups the coefficients by the corresponding variables (that they affect). Its loss function is

$$
\frac{1}{2} \| Y - BZ \|_2^2 + \lambda \sum_{i \neq j} \left\| \left( B^{(1)} \right)_{ij} \left( B^{(2)} \right)_{ij} \cdots \left( B^{(p)} \right)_{ij} \right\|_2 \tag{9}
$$

The $L_2$ norm of $p$–tuple of $\left( B^{(l)} \right)_{ij}$ is a composite penalty that will force all $p$ matrices $B^{(l)}$’s to share the same sparsity pattern, as can be observed in Figure 2. This structure can be useful to detect which locations can promote the forecasts at some location.

For a better understanding of the presented LASSO-VAR variants, Figure 2 illustrates an example of corresponding generated sparsity patterns.

<table>
<thead>
<tr>
<th>LASSO-VAR Structures</th>
<th>Illustration</th>
</tr>
</thead>
<tbody>
<tr>
<td>sLV</td>
<td><img src="image" alt="sLV Illustration" /></td>
</tr>
<tr>
<td>rLV</td>
<td><img src="image" alt="rLV Illustration" /></td>
</tr>
<tr>
<td>ILV</td>
<td><img src="image" alt="ILV Illustration" /></td>
</tr>
<tr>
<td>IsLV</td>
<td><img src="image" alt="IsLV Illustration" /></td>
</tr>
<tr>
<td>ooLV</td>
<td><img src="image" alt="ooLV Illustration" /></td>
</tr>
<tr>
<td>cLV</td>
<td><img src="image" alt="cLV Illustration" /></td>
</tr>
</tbody>
</table>

**Figure 2.** Example of sparsity patterns produced by LASSO-VAR structures.
3. VAR MODEL FITTING

The LASSO structures described in the previous section are non-differentiable objective functions, which makes it challenging to solve since it is not possible to obtain a closed form solution. The ADMM is a recent powerful algorithm that circumvents this situation and has been successfully shown to be efficient and well suited to distributed convex optimization, in particular for solving many large-scale statistical problems. The method also offers a high-convergence performance that, for some problems, is comparable with recent competitive algorithms.

3.1. ADMM framework

The ADMM framework combines the decomposability offered by the dual ascent method with the superior convergence properties of the method of multipliers, which means that problems with non-differentiable objective functions can be easily addressed and it is possible to perform a parallel optimization (topic covered in Section 3.2).

To give an overview of the key elements of ADMM, first recall the LASSO-VAR objective function in (4) and rewrite it in ADMM form as

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \| Y - BZ \|_2^2 + \lambda \| H \|_1 \\
\text{subject to} & \quad B - H = 0
\end{align*}
\]

Especially, the ADMM form is obtained replicating the \( B \) variable in the \( H \) variable and adding an equality constraint imposing that the two variables are equal. This can be viewed as a splitting of the objective function in two distinct objective functions, \( f(B) = \frac{1}{2} \| Y - BZ \|_2^2 \) and \( g(H) = \lambda \| H \|_1 \).

The augmented Lagrangian of this problem is

\[
L_\rho(B, H, W) = \frac{1}{2} \| Y - BZ \|_2^2 + \lambda \| H \|_1 + W^T (B - H) + \frac{\rho}{2} \| B - H \|_2^2
\]

where \( W \) is the dual variable (or Lagrange multiplier), and \( \rho > 0 \) is called the penalty parameter (or augmented Lagrange multiplier).

It is common to rewrite this Lagrangian in a scaled form by combining its linear and quadratic terms

\[
L_\rho(B, H, U) = \frac{1}{2} \| Y - BZ \|_2^2 + \lambda \| H \|_1 + \rho \| B - H + U \|_2^2 - \frac{\rho}{2} \| U \|_2^2
\]

where \( U = (1/\rho)W \) is the scaled dual variable associated with the constraint \( B = H \). The last term will be ignored in the sequel since it is a constant and does not matter when dealing with minimizations.

The method of multipliers for this problem is

\[
(B^{k+1}, H^{k+1}) := \arg \min_{B, H} L_\rho(B, H, U^k)
\]

\[
U^{k+1} := W^k + B^{k+1} - H^{k+1}
\]

The method of multipliers greatly improves the convergence properties over dual ascent, converging under far more general conditions. However, it is unable to address decomposition. The ADMM goes beyond decomposition issue by performing alternating minimization of the augmented Lagrangian over \( B \) and \( H \) instead of the usual joint minimization. The ADMM algorithm for (10) consists of the following iterations:

\[
B^{k+1} := \arg \min_B \left( \frac{1}{2} \| Y - BZ \|_2^2 + \frac{\rho}{2} \| B - H^k + U^k \|_2^2 \right)
\]

\[
H^{k+1} := \arg \min_H \left( \lambda \| H \|_1 + \frac{\rho}{2} \| B^{k+1} - H + U^k \|_2^2 \right)
\]

\[
U^{k+1} := U^k + B^{k+1} - H^{k+1}
\]

The ADMM performs minimization with respect to \( B \) (with \( H \) and \( U \) fixed), in (14), followed by minimization with respect to \( H \) (with \( B \) and \( U \) fixed), in (15), and finally, it updates the scaled dual variable \( U \), in (16).

Unlike the method of multipliers, the ADMM essentially decouples the functions \( f \) and \( g \), which makes it possible to exploit the individual structure of the \( f \) and \( g \) so that \( B \)-minimization and \( H \)-minimization may be computed in an efficient and parallel manner.

The ADMM formulation for the other LASSO-VAR structures presented in Section 2.2 can be obtained by replacing the regularization term, \( g(H) = \lambda \| H \|_1 \), in (10) and subsequent expressions by the corresponding LASSO penalty function. Nevertheless, this change is not trivial, and several details should be taken into account for a practical implementation, which will be described in Section 3.3.
3.2. Distributed fitting of the LASSO structures

This section presents distributed ADMM-based methods applied, by examples or by predictors, to the different LASSO-VAR structures. The main goal is to divide the initial problem into small local subproblems and thus improve the computational performance by solving the problems in a distributed way, with each processor (or computer) handling a subproblem.

Finding a solution to the problem of minimize the LASSO-VAR loss function, (4), involves computing $BZ$ and $YZ^0$ in a distributed manner. In this context, there are two main scenarios (Figure 3): row-block distribution and column-block distribution, in which $Z$ is partitioned into $N$ row blocks (splitting across predictors) or column blocks (splitting across examples), correspondingly.

One must choose the adequate scenario according to the $Z$-matrix dimension, i.e., choosing a row-block distribution if it has a large number of rows and a modest number of columns, and a column-block distribution otherwise. To cope with these settings, the same approach used for the sharing problem (in case of row-block distribution) and for global consensus problem (in case of column-block distribution) will be followed. These scenarios are described, using the sLV, in the remainder of this section. The same procedure can be followed for the other LASSO-VAR structures.

3.2.1. Row-block distribution.

The matrix $B$ is partitioned in $N$ column blocks as $B = (B_1 \ldots B_N)$ with $B_i \in \mathbb{R}^{l \times m_i}$, and the data matrix $Z$ is partitioned in $N$ row blocks as $Z = (Z_1, \ldots, Z_N)$ with $Z_i \in \mathbb{R}^{m_i \times T}$, where $\sum_{i=1}^{N} m_i = m$ (Figure 4). (Thus, $BZ = \sum_{i=1}^{N} B_i Z_i$, i.e., $B_i Z_i$ can be thought of as a ‘partial’ prediction of $Y$.) However, the blocks have to be carefully constructed when using an AR model. One has to be sure that all the lags are being considered in each block division/subproblem. In this case, this is ensured partitioning first each $B$ and $Z$ lag in $N$ column and row subblocks, respectively, and stacking the formed subblocks in each lag corresponding to the same order partition, in order to obtain blocks containing information of all the lags considered in the model.

This kind of procedure is especially important for the structures in which the penalty considers the division by lags, such as the ILV and the IsLV structures. For a two-lag model, it corresponds to gathering $B_i^{(1)}$ with $B_i^{(2)}$ and $Z_i^{(1)}$ with $Z_i^{(2)}$, to obtain each block $B_i$ and $Z_i$ for $i = 1, \ldots, N$ (Figure 4). For simplification, the notation $BZ^{k+1}$ is used to express the mean $B^{k+1}Z$. 
Then, for sLV, the model fitting problem (4) becomes

$$\minimize \frac{1}{2} \left\| Y - \sum_{i=1}^{N} B_i Z_i \right\|_2^2 + \lambda \sum_{i=1}^{N} \| B_i \|_1$$

(17)

and can be expressed in the sharing problem form as

$$\minimize \frac{1}{2} \left\| Y - \sum_{i=1}^{N} H_i \right\|_2^2 + \lambda \sum_{i=1}^{N} \| B_i \|_1$$

subject to \( B_i Z_i - H_i = 0, i = 1, \ldots, N \)

(18)

with new variables \( H_i \in \mathbb{R}^{k \times T} \). Thus, the scaled form of ADMM is

$$
\begin{align*}
B_i^{k+1} &:= \arg \min_{B_i} \left( \lambda \| B_i \|_1 + \frac{\rho}{2} \left\| B_i Z_i - H_i^k + U_i^k \right\|_2^2 \right) \\
H^k &:= \arg \min_{H} \left( \frac{1}{2} \left\| Y - \sum_{i=1}^{N} H_i \right\|_2^2 + \frac{\rho}{2} \sum_{i=1}^{N} \left\| B_i^{k+1} Z_i - H_i^k + U_i^k \right\|_2^2 \right) \\
U^{k+1} &:= U^k + B_i^{k+1} Z_i - H_i^{k+1}
\end{align*}
$$

(19)

Carrying out the \( H \)-update and using a single dual variable, the resulting ADMM algorithm is

$$
\begin{align*}
B_i^{k+1} &:= \arg \min_{B_i} \left( \frac{\rho}{2} \left\| B_i^k Z_i + \bar{T}^k - \bar{B} Z^k - U^k - B_i Z_i \right\|_2^2 + \lambda \| B_i \|_1 \right) \\
\bar{T}^k &:= \frac{1}{N + \rho} \left( Y - \rho \bar{B} Z^{k+1} + \rho U^k \right) \\
U^{k+1} &:= U^k + \bar{B} Z^{k+1} - \bar{T}^{k+1}
\end{align*}
$$

(20)

Each \( B_i \)-update is a LASSO problem that can be solved using ADMM by adapting (10) as

$$\minimize \frac{1}{2} \left\| \hat{Y}_i - B_i Z_i \right\|_2^2 + \hat{\lambda} \| H_i \|_1$$

subject to \( B_i - H_i = 0 \)

(21)

where \( \hat{Y}_i = B_i^k Z_i + \bar{T}^k - \bar{B} Z^k - U^k \) and \( \hat{\lambda} = \lambda / \rho \).

3.2.2. Column-block distribution.

The matrices \( Y \) and \( Z \) are partitioned in \( N \) column blocks as \( Y = (Y_1 \ldots Y_N) \) and \( Z = (Z_1 \ldots Z_N) \), with \( Y_i \in \mathbb{R}^{k \times T_i} \) and \( Z_i \in \mathbb{R}^{m \times T_i} \), where \( \sum_{i=1}^{N} T_i = T \). (Thus, \( Y_i \) and \( Z_i \) represent the \( i \)th block of data and will be handled by the \( i \)th processor.) In this case, there is no need to perform a ‘special’ construction of the blocks (partitioning the lags into subblocks) since all the lags are being considered in each block division/subproblem.

Then, for sLV, the model fitting problem (4) becomes

$$\minimize \frac{1}{2} \sum_{i=1}^{N} \left\| Y_i - B_i Z_i \right\|_2^2 + \lambda \| B_i \|_1$$

(22)

and can be expressed in the consensus problem form as

$$\minimize \frac{1}{2} \sum_{i=1}^{N} \left\| Y_i - B_i Z_i \right\|_2^2 + \lambda \| H \|_1$$

subject to \( B_i - H = 0, i = 1, \ldots, N \)

(23)
with variables \( B_i \in \mathbb{R}^{k \times m} \) and \( H \in \mathbb{R}^{k \times m} \). Thus, the resulting ADMM algorithm for consensus is

\[
B_i^{k+1} := \arg \min_{B_i} \left( \frac{1}{2} \sum_{i=1}^{N} \| Y_i - B_i Z_i \|^2_2 + \frac{\alpha}{2} \| B_i - H^k + U_i^k \|^2_2 \right)
\]

\[
H^{k+1} := \arg \min_{H} \left( \lambda \| H \|_1 + \frac{\alpha}{2} \| H - B^{k+1} - U^k \|^2_2 \right)
\]

\[
U^{k+1} := U_i^k + B_i^{k+1} - H^{k+1}
\]

### 3.3. Practical implementation

For a practical implementation, some aspects of the ADMM algorithm should be further discussed. Firstly, the efficient computation of \( B \)-update and \( H \)-update must be exploited. In addition, there are two parameters that need to be set: the LASSO regularization parameter, \( \lambda \), and the ADMM penalty parameter, \( \rho \). For all LASSO variants, all the primal variables are initialized to zero and the overall stopping criterion (that tests if the sequence \( \{ B_i^{k+1} \}_{k=0,1,2,\ldots} \) stabilizes itself sufficiently to stop the algorithm) is set to \( \| B_i^{k+1} - B_i^k \| / \max(1, \min(\| B_i^k \|, \| B_i^k \|)) \leq \epsilon \), where \( \epsilon \) is the relative tolerance parameter. The only exception is the \( \ell_1 \)-structure, which is implemented based on the code suggested in Boyd et al.\(^{27} \) and considers a relative tolerance parameter, \( \epsilon_r \), and an absolute tolerance parameter, \( \epsilon_a \).

The selection of the optimal tolerance parameters depends on the application and the algorithm used. Here, in order to choose a reasonable value for the tolerance parameters, a trade-off between the model’s running time and its solution was conducted. In practice, for each LASSO-V AR structure, the problem was first solved with a relatively large tolerance, and after the solution process continued with a smaller tolerance value, depending on how important it is to accurately map the solution and have reduced running times.

#### 3.3.1. B-update step.

The \( B \)-update, i.e., the step (14), takes the form of a ridge regression (i.e., quadratically regularized least squares) problem, with analytical solution

\[
B_i^{k+1} := (Y_i Z_i^T + \rho (H^k - U_i^k)) (Z_i Z_i^T + \rho I)^{-1}
\]

This solution is shared by all the LASSO-V AR structures presented in Section 2.2.

The \( B \)-minimization step of the ADMM (14) reduces to solving a system of linear equations involving the matrix \( Z_i Z_i^T + \rho I \). As long as the parameter \( \rho \) remains constant throughout the algorithm, the factorization of the matrix \( Z_i Z_i^T + \rho I \) can be cached once at the outset, and subsequent iterations can be carried out cheaply only involving back-solving the system using this factorization. Furthermore, for ‘skinny’ \( Z \) matrices (i.e., \( m \geq n \)), one may apply the matrix inversion lemma to \( (Z_i Z_i^T + \rho I)^{-1} \) and instead compute the factorization of the smaller matrix \((I + (1/\rho)Z_i Z_i^T)^{-1}\).

In a column block distribution framework, the same technique used for (14) applies for \( B_i \)-update in (24), resulting in the analytical solution

\[
B_i^{k+1} := (Y_i Z_i^T + \rho (H^k - U_i^k)) (Z_i Z_i^T + \rho I)^{-1}
\]

The same techniques as earlier, i.e., the factorization of \( Z_i Z_i^T + \rho I \) and the matrix inversion lemma, can also be applied to obtain an efficient method for computing multiple updates.

#### 3.3.2. H-update step.

The \( H \)-update will result in \( L_1-L_2 \) minimization problems that can be solved by means of shrinkages. For the \( \ell_1 \)-structure, the solution to step (15) is given element-wise by

\[
H^{k+1} = S_1 \left( B^{k+1} + U^k, \lambda / \rho \right)
\]

where \( S_1 \) is the scalar soft thresholding operator, defined as

\[
S_1(x, a) = \frac{x}{|x|} \max \{0, |x| - a\}
\]

The shrinkage (28) is the proximity operator of the \( L_1 \) norm

\[
S_1(x, a) = \arg \min_w w \|_1 + \frac{1}{2} \| w - x \|_2^2
\]
Following the same procedure used for sLV, the $H$-update for rLV is given by

$$H^{k+1} = S_1 \left( B^{(k+1)} + U^k \lambda / \rho \right)$$ \hspace{1cm} (30)

For ILV instead of doing scalar soft thresholding, a matrix soft thresholding is performed, i.e.,

$$H^{k+1} = S_2 \left( B^{(0)l+1} + U^{(0)l} \lambda / \rho \right)$$ \hspace{1cm} (31)

where $S_2$ is the matrix soft thresholding operator, defined as

$$S_2(X, \alpha) = \frac{X}{\|X\|_2} \max \{0, \|X\|_2 - \alpha\}$$ \hspace{1cm} (32)

Similarly to the approach used in Chartrand and Wohlberg, the $H$-update solution for IsLV, can be obtained combining scalar and matrix soft thresholding and is given block-wise by

$$H^{k+1} = S_2 \left( S_1 \left( B^{(0)l+1} + U^{(0)l} \right), (1 - \alpha)\lambda / \rho \right), \alpha\lambda / \rho \right)$$ \hspace{1cm} (33)

For ooLV, two vector soft thresholding are performed, one for diagonal entries and other for off-diagonal entries, in the following way:

$$H^{k+1} = S_2 \left( \text{diag} \left( B^{(0)l+1} + U^{(0)l} \right), \sqrt{k\lambda / \rho} \right) I + S_2 \left( \left( B^{(0)l+1} + U^{(0)l} \right)^-, \sqrt{k(k-1)\lambda / \rho} \right) (I - I)$$ \hspace{1cm} (34)

where $1$ denotes a $k \times k$ matrix of ones and $I$ denotes the $k \times k$ identity matrix.

Finally, for cLV, the $H$-update solution can be obtained performing a vector soft thresholding and is given row-wise by

$$H^{k+1} = S_2 \left( \left( B^{(1)l+1} + U^{(1)l} \right)_{ij}, \left( B^{(2)l+1} + U^{(2)l} \right)_{ij}, \ldots, \left( B^{(P)l+1} + U^{(P)l} \right)_{ij} \right), \lambda / \rho \right)$$ \hspace{1cm} (35)

In the column-block distribution framework, the procedure is the same but adjusting the soft thresholding operator to each LASSO-V AR structure. For instance, for the sLV, the solution to $H$-update in (24) is given by

$$H^{k+1} = S_1 \left( B^{k+1} + U^k \lambda / (N\rho) \right)$$ \hspace{1cm} (36)

Further details about soft thresholding operator can be found in Boyd et al.\textsuperscript{27}

3.3.3. Parameters estimation.

The optimal parameter values for each LASSO-V AR structure are estimated through $k$-fold cross validation, where the pair $(\lambda, \rho)$ that gives rise to the lowest square error is selected for each lead time.

To perform cross validation, a grid of $\lambda$ and $\rho$ values are considered. For $\lambda$, a decreasing sequence of values are computed, spaced logarithmically from the value $\lambda_{\text{max}}$ yielding to the sparsest solution (diagonal coefficient matrices). The $\lambda_{\text{max}}$ is calculated following the rule described in Songsiri.\textsuperscript{25}

Selecting $\rho$ as a function of $\lambda$ gives a desirable performance and the computational effort appears to be smaller. Hence, $\rho = \rho^\ast \lambda$ is assumed, with $\rho^\ast$ being an auxiliary constant, and a range of constant values $\rho^\ast$ is considered to perform cross validation.

3.3.4. Computational implementation.

The ADMM algorithm can be implemented with several distributed processing systems. In Singh and Reddy,\textsuperscript{35} the technologies/platforms are classified into two types: (i) horizontal scaling that involves distributing the workload by several servers (multiple independent machines)-decentralized and distributed cluster (cloud) computing framework; (ii) vertical scaling that involves installing more processors, memory and faster hardware inside a single machine. For horizontal scaling, Message Passing Interface was the first communication protocol to distribute and exchange the data between peers, Apache Hadoop with MapReduce as the data processing scheme emerged later, and Apache Spark is the prevalent solution. The main limitation of MapReduce is that it is not designed for iterative processes due to disk I/O limitations,\textsuperscript{35} which degrades the performance of the ADMM iterations. Spark has the unique feature to perform in-memory computations that overcomes the MapReduce limitations for iterative processes and currently is the adequate solution to implement ADMM in cluster (cloud) computing. Recent developments, such as HaLoop and Twister, seek to improve MapReduce performance for iterative tasks.\textsuperscript{36}

The most popular vertical scale up technologies are High Performance Computing Clusters, multicore processors, Graphics Processing Unit (GPU) and field programmable gate arrays (FPGA). The ADMM algorithm can be implemented
4. APPLICATION AND CASE STUDY

4.1. Experimental setup

4.1.1. Dataset description.

The proposed LASSO-V AR frameworks are tested on hourly mean wind power data that comprises 1 year of observations from 66 WPPs located in the same control area. The individual rated power ranges between 2 MW and 220 MW. All data have been normalized by the nominal power so that its values are between 0 and 1. The time series data, as depicted in Figure 5 for two WPPs, exhibits a high cross-autocorrelation in lags different from zero and one.

The first 9 months of the year (January–September) are used as a training set on which the implementation of the fitting procedure is optimized by cross-validation. The remaining 3 months (October–December) are then used to evaluate the performance of the models, which results are presented in Section 4.2.

In all models, two lags are used and forecasts from \( t + h, h = 1, \ldots, 6 \), i.e., six steps ahead, are considered. For a deeper visualization of the 132 × 6549 input matrix \( Z \), for \( h = 1 \), part of it is represented as follows (the horizontal dashed line divides the two lags):

\[
Z = \begin{bmatrix}
0.0792 & 0.1238 & 0.0545 & 0.1584 & 0.1584 & \cdots & 0.5594 & 0.5446 & 0.6584 & 0.7129 & 0.6337 \\
0.2222 & 0.3796 & 0.4769 & 0.4167 & 0.4630 & \cdots & 0.2269 & 0.2870 & 0.4259 & 0.4120 & 0.3935 \\
0.2541 & 0.2818 & 0.1823 & 0.0276 & 0.0221 & \cdots & 0.5138 & 0.7624 & 0.7790 & 0.5746 & 0.3757 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0.2727 & 0.3223 & 0.3884 & 0.2727 & 0.3140 & \cdots & 0.0331 & 0.2893 & 0.6446 & 0.6281 & 0.3636 \\
0.2711 & 0.2042 & 0.2359 & 0.3204 & 0.3697 & \cdots & 0.2482 & 0.1849 & 0.1725 & 0.1743 & 0.2113 \\
0.3294 & 0.3048 & 0.2739 & 0.1194 & 0.1840 & \cdots & 0.2142 & 0.4003 & 0.6166 & 0.5485 & 0.1419 \\
0.0396 & 0.0792 & 0.1238 & 0.0545 & 0.1584 & \cdots & 0.5547 & 0.5594 & 0.3446 & 0.6584 & 0.7129 \\
0.1713 & 0.2222 & 0.3796 & 0.4769 & 0.4167 & \cdots & 0.2361 & 0.2269 & 0.2870 & 0.4259 & 0.4120 \\
0.5967 & 0.2541 & 0.2818 & 0.1823 & 0.0276 & \cdots & 0.2431 & 0.5138 & 0.7624 & 0.7790 & 0.5746 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0.1818 & 0.2727 & 0.3223 & 0.3884 & 0.2727 & \cdots & 0.2397 & 0.0331 & 0.2893 & 0.6446 & 0.6281 \\
0.2324 & 0.2711 & 0.2042 & 0.2359 & 0.3204 & \cdots & 0.2254 & 0.2482 & 0.1849 & 0.1725 & 0.1743 \\
0.2633 & 0.3294 & 0.3048 & 0.2739 & 0.1194 & \cdots & 0.2135 & 0.2142 & 0.4003 & 0.6166 & 0.5485 \\
\end{bmatrix}
\]

Figure 5. Cross-autocorrelation plot between two wind power plants.
Table I. Tolerance parameters used for all the ADMM-based LASSO-VAR algorithms (distributed and non-distributed).

<table>
<thead>
<tr>
<th></th>
<th>rlLV</th>
<th>sLV</th>
<th>ILV</th>
<th>cLV</th>
<th>IslLV</th>
<th>ooLV</th>
<th>VAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-distributed</td>
<td>$\epsilon_\beta = 1e^{-4}$; $\epsilon_r = 1e^{-2}$</td>
<td>1e-5</td>
<td>1e-5</td>
<td>1e-4</td>
<td>1e-5</td>
<td>1e-5</td>
<td>1e-5</td>
</tr>
<tr>
<td>Distributed</td>
<td>$\epsilon_\beta = 1e^{-4}$; $\epsilon_r = 5e^{-2}$</td>
<td>2e-3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table II. Average RMSE and MAE (two lowest values in bold) across all sites for lead time $t+1$ (values normalized by rated power).

<table>
<thead>
<tr>
<th></th>
<th>rlLV</th>
<th>sLV</th>
<th>ILV</th>
<th>cLV</th>
<th>IslLV</th>
<th>ooLV</th>
<th>VAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMSE</td>
<td>11.1406</td>
<td>11.1387</td>
<td>11.1979</td>
<td><strong>11.1310</strong></td>
<td>11.1691</td>
<td><strong>11.1300</strong></td>
<td>11.2281</td>
</tr>
<tr>
<td>MAE</td>
<td>75650</td>
<td>75636</td>
<td>76431</td>
<td><strong>75538</strong></td>
<td>76258</td>
<td><strong>75368</strong></td>
<td>76563</td>
</tr>
</tbody>
</table>

The ADMM tolerances are presented in Table I. For the distributed LASSO-VAR algorithms, the problem is split into eight subproblems. The calculations are performed on a HP with 8 Intel Core i7-2600 CPU @ 3.40 GHz processor and 8 GB of RAM, and the algorithm is programmed in MATLAB R2012a.

4.1.2. State-of-the-art benchmark model.
An alternative method for estimating a sparse VAR model has been proposed in Davis et al.\textsuperscript{25} and is applied to very-short-term wind power forecasting in Dowell and Pinson.\textsuperscript{23} This approach, henceforth referred to as sVAR, is based on a parameter-ranking procedure to determine the sparse structure for the VAR model and maximum likelihood estimation to estimate the parameter values. Fitting the sVAR model is a two-stage procedure. The first stage determines the temporal order, $p$, of the sVAR model and makes a first estimation of the sparsity structure. The second stage refines the selection of parameters made in stage one.

The sVAR is implemented here as part of our case study so that its performance may be compared with that of the proposed LASSO-VAR method.

4.2. Results and discussion
In this section, several results obtained from the experiment described in Section 4.1 are provided and discussed.

4.2.1. Forecast accuracy.
The forecasting skill of the LASSO-VAR structures is evaluated with the RMSE and mean absolute error (MAE) calculated for the $t+h$ lead time with the following expressions:

$$\text{RMSE}_{t+h} = \sqrt{\frac{1}{k} \sum_{i=1}^{k} (\hat{Y}_{t+h|i} - Y_{t+h})^2},$$

$$\text{MAE}_{t+h} = \frac{1}{k} \sum_{i=1}^{k} |\hat{Y}_{t+h|i} - Y_{t+h}|,$$

with the forecast $\hat{Y}_{t+h|i}$ made at time instant $t$ and observed $(Y_{t+h})$ value normalized by the WPP rated power.

These skill scores were calculated separately for each model using the full dataset of errors. For lead time $t+1$, Table II shows the global RMSE and MAE scores for the six LASSO-VAR structures and also for the VAR model.

The forecast error metrics show that all the LASSO-VAR structures outperform the VAR model in terms of RMSE and MAE. As expected, the sLV and rLV exhibit very similar scores and the structure with the worst performance is the lLV. The two best scores, highlighted in bold, are achieved by cLV and ooLV structures.

The improvement of the VAR and LASSO-VAR structures over the AR model and persistence (defined as $\hat{Y}_{t+h|i} = \hat{Y}_t$), in terms of RMSE, for each lead time is plotted in Figures 6 and 7, respectively. These plots clearly show that the cLV and ooLV structures achieve the highest improvements for all lead times except for lead time $t+5$ whose top is led by ooLV and IslLV. Also, all the LASSO-VAR structures show better improvement than the VAR model, more pronounced after the second lead time.

In general, the improvement over AR is higher for the first three lead times ranging between 5.48\% and 6.91\% on average for the structures with the best performance (cLV and ooLV).
It is noteworthy that the rLV and sLV also have a very pleasant demeanor for all lead times, with an improvement deviation from the best model, between 0.07% (for the first lead time) and 0.5% (for the fifth lead time). Another interesting conclusion is that the improvement decays with the lead time, meaning that the spatial–temporal information is more relevant for the first three lead times and with a peak at lead time two that corroborates the information depicted by the cross-correlation plot (Figure 5).

The improvement over persistence increases with the time horizon till lead time $t + 4$, which is the maximum value, decreasing very slightly in the remaining lead times. As expected, the improvement over persistence is higher compared with the improvement over the AR model.

The results concerning the forecasting performance of cLV and ooLV structures against sVAR (benchmark model from the state of the art) are presented next.

The Figure 8 compares the cLV and ooLV with the sVAR model representing, for the first lead time, the improvement over the sVAR model for each WPP. The results show that, for the first lead time, the sVAR only has a better performance than cLV and ooLV at one WPP. Apart from that WPP, the improvement over sVAR ranges between 0.08% and 9.48% for cLV and between 0.17% and 9.85% for ooLV. More specifically, 95% of the WPP show an improvement over 2% and 55% of them achieved an improvement over 5%.

The Diebold–Mariano (DM) test is applied to assess the statistical significance of the forecast error improvement in each WPP. The null hypothesis is 'no difference in the accuracy of two competing forecasts', and if the $p$-value is less than a significance level (i.e., 0.05 in this paper), then the observed result would be highly unlikely under the
null hypothesis. Figure 9 depicts the $p$-values obtained for the first lead time, which shows $p$-values under the significance level for the vast majority of WPP, stressing that the difference in accuracy between the competing forecasts is significant.

Finally, for illustrative purposes, a visualization of the real wind power generated and the forecast wind power output provided by the ooLV model for the first lead time during a 10 day period is shown in Figure 10 for one WPP.

4.2.2. Analysis of the sparsity patterns.

In order to understand the joint dynamic behavior of this group of WPP, the sparsity patterns (i.e., coefficients’ matrix) obtained by the LASSO-V AR structures and sVAR for the first lead time are depicted in Figure 11. The darker shade represents coefficients that are larger in magnitude.

The figures show that the models that result in an unstructured sparsity pattern (rLV and sLV) give rise to the most sparse matrices, with about 40% of null values. Immediately afterwards, with 25% of sparsity, is the lsLV, which has a structured blockwise sparsity, but with unstructured sparsity within each block. The structured cLV gives rise to nearly 16% of sparsity, revealing that, in average, about 11 sites were not considered to the forecast of each WPP.

Both lLV and ooLV structures present non-sparse coefficient matrices. This means that both lags (in lLV) and both diagonal and non-diagonal entries (in ooLV) were considered relevant in the model. However, while the lLV accounts for the same penalty for each lag, the ooLV assign different penalties to the diagonal and non-diagonal entries. This might justify the good performance of the ooLV, that, in general, produces non-diagonal coefficients of smaller magnitude than the ones produced by lLV. As expected, it can be observed that all figures agree that the diagonal coefficients of the first lag should have a higher magnitude, revealing that a variable’s own first lag are more likely to improve the forecast than the other entries.

The top performance of ooLV highlights the great importance of the predictors corresponding to the first lag in the prediction, relatively to the remaining coefficients, that should also be taken into account but with a smaller contribution.
Also, the good results obtained by cLV demonstrates the importance of learning the causal inference from the data in order to find out which are the locations that add improvement to the prediction.

The sparsity obtained by the sVAR model (around 96%) is much higher than the one obtained by any of the LASSO-VAR structures, and it is possible to distinguish a great prevalence of the diagonal entries compared with the non-diagonal entries. However, a higher sparsity does not always indicate a better performance and other aspects, such as forecast improvement and computational time, should be considered when choosing the most appropriate model for each case.

### 4.2.3. Computational performance.

In this subsection, the computational performance of the distributed ADMM algorithm is assessed. Since the input matrix has a modest number of rows and a large number of columns ($Z \in \mathbb{R}^{132 \times 6549}$ for first lead time), a column-block distribution scenario is followed. In the implementation of the distributed algorithm, the $B$-update and $U$-update are performed in parallel.

The skill scores obtained for each ADMM-based distributed LASSO-VAR algorithm are very similar to the corresponding non-distributed version. Consequently, the running times and number of iterations of both distributed and non-distributed versions are evaluated and compared to investigate which advantages one may enjoy when a parallel computation is performed.

The results are depicted in Table III that shows that the use of distributed ADMM results in a decrease of the running time for all structures except for the rLV and ooLV, and in a decrease of the iteration number for all structures except sLV and ooLV.

To properly analyse these results, the computational times are separated to distinguish the time elapsed in the cycle where the iterations are made and the time elapsed out of this cycle where auxiliary calculations, such as the Cholesky factorization, are performed. As one knows, the running time of each cycle depends on the number of iterations taken, then it is more appropriate to display the time that each structure takes to perform one iteration, i.e., a per iteration cost, in order to make the algorithms comparable. The running times for each structure, using both versions of the ADMM (distributed and non-distributed), by iteration and for auxiliary calculations, are represented in Figure 12.

Concerning the time by iteration, it is possible to note that all structures show very similar results in both versions, except the sLV in which the running time is greatly reduced when a distributed version is used. As a result, the decreasing of the total running time for all structures, except for sLV, is very sensitive to the number of iterations taken in the distributed version. This seems to justify the results in Table III relating to the structures sLV, ILV, cLV, lsLV, and ooLV, taking into account that the time spent out of the cycle is similar in both versions. Also, it can be observed that the time spent by rLV out of the cycle greatly increases in the distributed version highlighting the impact of the increasing number of calculations (proportional to the number of WPPs) in this structure since it is applied to predict each location separately. This explains that, although the number of iterations is slightly decreased in this structure, the total elapsed time has increased.

Taking a deeper look over the fitting times of the non-distributed versions of the structures with better performances, we observe that the cLV is quite slow, possibly due to the combination of a high number of iterations with a high per iteration time, while ooLV stands out as the fastest. Although not at the top performances and of being the most time-consuming by iteration, the sLV also offers good results and its fast convergence results in a relatively low fitting time. Thereafter,
if the objective is to find a model with a good performance and a low running time, the best choice should lie with the ooLV.

Regarding now the distributed versions, the rLV should not be considered since there is no advantage in its application. It is important to point out that, except the sLV, all the other structures have similar per iteration times in both versions indicating that the implementation of its distributed version only brings advantage if the number of iterations in this version is less than the one achieved in the non-distributed version. To deal with this, the tolerance of the termination criterion must be chosen with some caution. The sLV structure is the only one that shows a significant decrease in the time per iteration, getting a competitive total running time despite the number of iterations be higher. Accordingly, if one looks for a balance between the performance, the running time and the scalability, the sLV is probably the more adequate choice.
Table III. Total running times (in seconds) and number of iterations (between parenthesis) for six lead times of the LASSO-VAR structures using distributed and non-distributed ADMM.

<table>
<thead>
<tr>
<th>LASSO-VAR structures</th>
<th>Standard ADMM</th>
<th>Distributed ADMM</th>
</tr>
</thead>
<tbody>
<tr>
<td>rLV</td>
<td>3.02 (4162)</td>
<td>5.29 (3610)</td>
</tr>
<tr>
<td>sLV</td>
<td>1.37 (84)</td>
<td>0.54 (163)</td>
</tr>
<tr>
<td>lLV</td>
<td>0.86 (397)</td>
<td>0.46 (128)</td>
</tr>
<tr>
<td>cLV</td>
<td>6.92 (472)</td>
<td>2.70 (164)</td>
</tr>
<tr>
<td>lsLV</td>
<td>0.80 (323)</td>
<td>0.37 (120)</td>
</tr>
<tr>
<td>ooLV</td>
<td>0.51 (109)</td>
<td>0.57 (142)</td>
</tr>
</tbody>
</table>

Figure 12. Running times by iteration (left) and outside cycle (right) of the LASSO-VAR structures using distributed and non-distributed ADMM.

Finally, it is important to emphasize that the proposed methodology provides higher computational performance compared with competitive models (such as the sVAR, with a running time of about 39 h), which is a key requirement for the large-scale application of the method.

5. CONCLUSIONS

This paper describes a forecasting technique that combines VAR and several variants of the LASSO framework to fully explore information from wind power time series distributed in space. The proposed methodology explores competing sparse structures for the VAR coefficients matrix and uses the ADMM optimization framework to guarantee fast convergence and parallel computation.

For a real case study with 66 WPPs, all the different sparse structures of the LASSO-VAR model shows a better performance than the persistence, AR and VAR models, and a sparse-VAR method from the state of the art. The ooLV and sLV structures turn out to be the best choices for, respectively, non-distributed and distributed implementations.

It is important to stress that one of the goals is to estimate a sparse matrix in which only the relevant predictors are selected to contribute for the forecasts with a small computational effort. However, the sparsity of the matrix depends on the number of relevant variables under the dynamic behavior considered by the structure, and a more sparse matrix does not always mean better performance. For instance, it is interesting to note that the ooLV does not produce a sparse matrix of coefficients, but its results stand out in terms of both forecasting skill and computational time.

As we have shown, the skill of the LASSO-VAR structures depends on the dynamic behavior of the data, and there is not a uniformly best structure. Therefore, each case must be carefully explored, and sometimes a trade-off between the computational expense and predictive performance must be made in order to choose the LASSO-VAR structure that meets our objectives.

This work was motivated by the desire to explore different sparse structures and propose a forecasting solution with high scalability. Despite the focus in point forecast, this methodology can be directly applied to forecast the mean and variance of a logit-normal distribution and generate probabilistic forecasts. Future work should explore alternatives to the ADMM algorithm (e.g., coordinate descent algorithm), include exogenous variables in the VAR and consider the possibility of a dynamic sparse structure.

Finally, the forecasting skill improvement is valuable for different transmission and distribution system operators (TSO and DSO) operational planning tasks. In this framework, TSO can improve the intra-day congestion forecast by combining spatial–temporal nodal feed-in generation and load at the transmission network level collected by the real-time supervisory control and data acquisition (SCADA), as well as to increase renewable resources integration and reduce sys-
tem costs against uncertainty (see Xie et al.\textsuperscript{40} for unit commitment and economic dispatch results). For DSO, improved accuracy and the possibility of updating forecast data regularly have a positive impact in voltage control functions with PV power electronic inverters and on-load tap-changer transformers, showed in Ziadi et al.\textsuperscript{41} to mitigate voltages violations and minimize the total losses. Moreover, renewable energy traders can decrease their imbalance costs, and the ADMM algorithm can be extended to construct data privacy-preserving distributed learning where several power plants collaborate to improve their forecasts without having to exchange actual generation data.\textsuperscript{42} A functional requirement for all end-users is the access to power observations in some specific times during the operation day, where a data update rate between 10 and 60 min in the ideal solution. This defines some requirements for communication latency and bandwidth.

ACKNOWLEDGEMENTS

This work was made in the framework of the SusCity project (contract no. ‘MITP-TB/CS/0026/2013’) financed by national funds through Fundação para a Ciência e a Tecnologia (FCT), Portugal. Jethro Dowell is supported by the University of Strathclyde’s EPSRC Doctoral Prize, grant number EP/M508159/1.

REFERENCES


