Erasmus University Rotterdam

Erasmus School of Economics

Master Econometrics and Management Science

Model selection for Vector Autoregressive processes using the Multi-Step Elastic Net

Author:

Nasser Oemar

333665

Supervisor:

Dr. Annika Schnücker

Second reader:

Dr. Hanno Reuvers

April, 2020

Abstract

Several penalized variable estimation techniques such as the (adaptive) elastic net have been proposed for modeling VAR data in order to improve dimension reduction and forecast performance. Literature has proven that the multi-step adaptive elastic net gains in sparsity performance, however, this has never been investigated within a VAR framework.

The aim is therefore to analyze if the multi-step adaptive elastic net (*maenet*) is able to provide the accurate VAR model compared to its single step variants used as benchmark methods such as the elastic net (*enet*), adaptive elastic net with ridge weights (*aenetR*) and the adaptive elastic net with lasso weights (*aenetL*).

I compare them in terms of estimation bias, sparsity and forecast performance.

Simulation results show that the multi-step adaptive elastic net is able to consistently find sparser VAR models compared to the benchmark methods with a gain in efficiency and accuracy as the probability of selecting the correct model increases. In addition, the coefficient estimates are closer to those of the true model. Forecast performance is one of the best in small samples, but approximately equal in large samples. Overall *maenet* performs well in high-dimensional small samples in terms of selecting the right variables and forecast perfomance.

Empirical results also show a sparser model compared to the single-step adaptive elastic net methods.

The gain in performance in especially small samples makes this model interesting for many fields, such as macroeconomics and finance.

Acknowledgements

Looking towards the end of my study Econometrics and Management Science, I can honestly tell that I have enjoyed my study so far. It has truly provided me interesting, though but certainly beautiful insights and tools to use in my career.

I declare my sincere gratitude to my family who has always supported me in my writing process. Even in periods where I faced some disturbing cicrcumstances, they still believed in me and motivated me to follow my dreams. Writing this thesis comes at the cost of lots of sacrifices, efforts and research work and also a job to get the bills paid.

I would also like to thank my supervisor dr. Annika Schnücker of the Erasmus University Rotterdam, who supported me during my writing process. She understood my personal situation and provided me the necessary and useful feedback.

[illegible]

M.N. Oemar

Contents

List of Figures

List of Tables

1 Introduction

In the field of macroeconomics and policy analysis, it is of keen interest to know what the dynamic relation between variables is. Vector autoregressive (VAR) models are the multivariate expansions of the univariate autoregressive (AR) models. Autoregressive models describe the intertemporal dynamics between the current value of a variable with its past values and an idiosyncratic part. Vector autoregressive models describe the dynamics of the variables as a function of their own lagged values, lagged values of other variables and an idiosyncratic part. Sims (1980) advocates the use of VAR models for macroeconomic analysis in order to quantify economic relations. Moreover, VAR models are amongst others appropriate for macroeconomic data, because macroeconomic data are correlated over time and as such dependent on past values. Macroeconomic variables are also likely to be intertemporally related to other macroeconomic variables. This makes the VAR model an appropriate estimation model for macroeconomic data.

The specification of VAR models consists of two steps. The first step is to analyze the k variables that should be included in the VAR model. The second step being a crucial choice for VAR models is the selection of the lag order. If the selected lag order is higher than than the true one, it results in overfitting and consecutively increases forecast errors. On contrary, if the selected lag order is smaller than the true one, it leads to underfitting resulting in autocorrelated error terms. The effect of autocorrelation is visible in the consistency of the estimated parameter, because the rate of convergence of the estimated parameter to the true parameter is then affected as shown in the paper of Sharma (1987). This in turn affects the efficiency of the prediction performance as well.

In the perspective of producing good predictions, it is necessary to select the best possible model in terms of forecast performance. Several attempts have been done for model selections. The first way for selecting the lag length was by using the information criteria such as the Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC). These conventional methods ought to find the model that balances a good fit between the data and sparsity. A proposition to improve this information criteria methods is to evaluate them sequentially. Two alternative methods are firstly the top-down approach and secondly the bottom-up approach. This procedure is dependent on the search path you opt for, which might provide sub-optimal models. Another approach for selecting the lag length is by means of hypothesis testing, where coefficients are tested on their significance.

Common to all approaches so far is that it is time-consuming and computationally intensive. This is where the work of Hsu et al. (2008) kicks in. They translate lag selection for VAR models to a variable selection problem. They use the lasso method of Tibshirani (1996) for model selection in the VAR framework. The lasso estimation is also based on minimizing the sum of squared residuals like the OLS estimation is based on. Now the addition here is that the sum of absolute coefficients is bounded by a prespecified value implying variable selection as some coefficients are put to zero. In this way the lasso method performs model selection by excluding abundant variables out of the model. The authors show an improvement in the forecast performance in finite samples. The paper of Zou and Hastie (2005) shows that the limitations of the lasso method are visible when the parameter space gets larger and more correlated.

That is where the elastic net estimation method shows an improvement compared to the lasso method. The paper of Zou and Hastie (2005) shows that the elastic net method, which combines the lasso and ridge regression penalties in a convex way, outperforms the lasso method in case the amount of parameters, in this case the pk^2 lagged variables, exceeds the amount of n observations and can obtain the oracle properties. Zou (2006) explains that if an estimator has the oracle properties, this estimator performs as if the true model were provided on beforehand. Technically said, the asymptotic distribution of an oracle estimator is the same as the asymptotic distribution of the maximum likelihood estimator (MLE) based on the true model. The lasso method is only capable of selecting at most n variables. In addition, if variables are highly correlated, then the lasso method randomly select one variable out of them, whilst the elastic net method is able to conduct grouped selection on which will be technically elaborated further in Section 3. The ridge regression part of the elastic net groups correlated variables together, which is the so-called de-correlation step such that a variable is not easily eliminated from the model while having predictive power. The lasso part of the elastic net asserts the variable selection by excluding redundant variables from the model. The paper of Zou and Hastie (2005) poses that the elastic net method works well in settings of relatively low amount of observations and a high set of variables.

For macroeconomic variables oftentimes the amount of observations n might be relatively low due to the frequency of data observations compared to the set of k macroeconomic variables. In light of this, it is of keen interest to investigate the elastic net procedure for selecting the lag order as well as evaluating the resulting forecast performance of the VAR models. Additionally, the grouping power of the elastic net method is useful for macroeconomic models. Suppose that you have a model with macroeconomic and financial variables, then it is likely that the financial and macroeconomic variables exhibit a group pattern with probably relatively strong intraclass correlations. The grouping property of the elastic net method as explained by Zou (2006) and Furman (2014), clusters correlated variables together. Finally, the oracle property of the elastic net method as mentioned earlier, is very useful in finding the correct sparsity pattern as the sample size increases.

As far as my knowledge of the literature concerns, only the paper of Furman (2014) explicitly discusses the elastic net approach in a VAR framework. In particular, they discussed the adaptive elastic net which allows heterogeneous weights for the lasso penalization on coefficients. This means that each coefficient is penalized differently according to their individual assigned penalty weight. Zou (2006) shows that this adaption in the lasso part results in the oracle property. However, in macroeconomics there is usually no large sample size. This means that the estimator is not likely to reach the asymptotics for the correct sparsity pattern and coefficient estimates. A relative finite small sample size implies a model that is not as sparse as that of the true model. Thus there is still a risk of having false positives in the model. This means that coefficients that should be zero and be left out of the model are unjustifiably estimated as a nonzero value and incorporated in the model causing overfitting. The paper of Xiao and Xu (2015) therefore presented the multi-step adaptive elastic net method for acquiring a more sparse model, which means decreasing the false positive rate, whilst maintaining the prediction accuracy as good as the adaptive elastic method provides or even better. When I link this concept to the VAR model, it becomes interesting. This is because you want to find the sparsest VAR model as generated by the true data generating process (DGP). Especially considering the consequences of underfitting and overfitting, this is of importance.

Combining the multi-step adaptive elastic net approach in terms of VAR model selection for finding the correct amount of p lags and sparsity in a relatively small finite sample, will be of added value for the current VAR literature, particularly in the field of macroeconomics, and has as far as my knowledge extents, not been investigated before.

The aim of this paper is to analyze the performance of the multi-step adaptive elastic net on VAR models in finite samples. I evaluate its performance in terms of variable selection, forecast performance and level of sparsity compared to the single-step elastic net, single-step adaptive elastic net based on ridge weights and the single-step adaptive elastic net based on lasso weights. I investigate the performance based on a simulation study and an empirical application.

The results of this paper show that the multi-step adaptive elastic net is able to find consistently sparser VAR models compared to the benchmark methods, such as the single step elastic net and its adaptive variants, with a gain in efficiency and accuracy as the probability of selecting the correct model increases. In addition, the coefficient estimates are closer to that of the true model. Forecast performance is one of the best in small samples, but approximately equal in large samples.

The remainder of this paper is set out as follows. Section 2 introduces the dataset for the empirical analysis. Section 3 introduces the model that I use for the analysis. Section 3.6 presents the simulation study for this model. Section 5 discusses the empirical analysis and section 7 concludes. The part appended to this study can be found in section 8 where the mathematical derivation of the used model is presented.

2 Data

The dataset that I use for this study comes from the *Economic Research department* of the Federal Reserve Bank of Saint Louis (FRED). I use the dataset with a timespan from the second quarter of 1959 until the first quarter of 2020 which amounts to about 244 observations. The variables in the dataset are grouped into financial and macroeconomic indicators. For the division of the groups, they largely followed the paper of Stock and Watson (2012).

I work with quarterly macroeconomic data of *Group 1: National Income and Product Accounts* henceforth abbreviated as *NIPA*. Since this group consists of 23 variables and all variables are explained in the appendix of the FRED database[1], I only summarize the categories of variables that I include rather than mentioning them all individually. Categories of variables I consider are *GDP, personal's available income, consumption and investment, government's expenditures, investments and receipts, exports and imports* and *output of several sectors.*

First of all, I would normally check for stationarity of the variables as this is the prerequisite prior to estimate a VAR model. However, the appendix of the FRED database already provides the integrating order of the variables as well as the transformation methods to get them stationary. That means that I do not have to conduct the Augmented Dickey-Fuller test for finding the integrating order I(d) as conducted by Engle and Granger (1991). It is for the estimation purpose important that the shape of the distribution remains the same over the time-dimension such that asymptotic theory can be held valid as is explained by Davidson (2009).

For the majority of the variables I take logarithmic differences c.q.

[1]https://research.stlouisfed.org/econ/mccracken/fred-databases/

$\Delta log(y_t)$. For some variables I applied the first numerical difference between two subsequent observations c.q. Δy_t. The remainder set of variables is already stationary and do therefore not need to be transformed.

2.1 Missing data

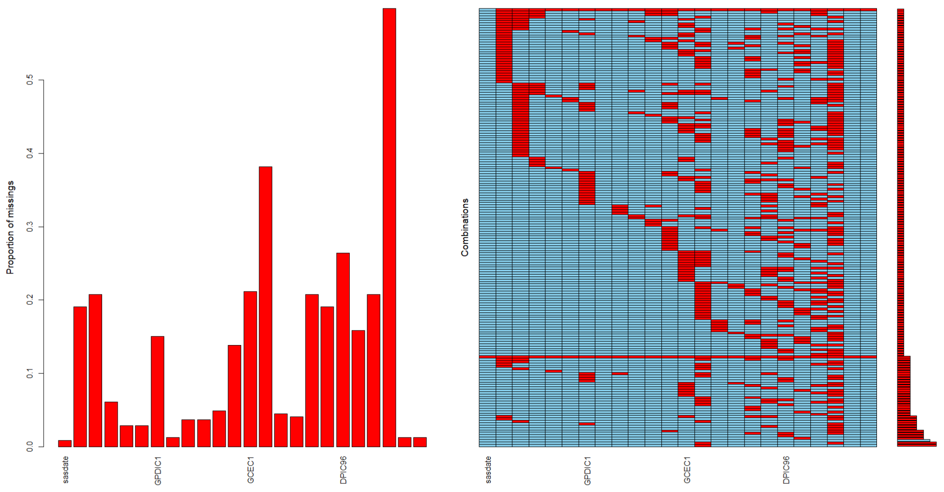


Fig. 1: Aggregate plot of variables in order to depict the missing values.

Figure 1 represents the overview of missing values within variables in terms of proportion missing and in terms of a combined overview. This plot is a result of the *VIM* package in *R*. If the missing observation extends over the set of variables or at least over the largest part of the variables, I opt for removing that observation. I consider the other missing values to follow the Missing at Random mechanism (MAR). This implies that I attempt to estimate the missing value by the information of other observed variables. I use a Bayesian approach for this problem. I draw imputed values from the predictive distribution conditioned on the observed variables in the model. In order to decrease the risk of biased estimates, multiple imputations are recommended as explained by Royston (2004). I therefore impute these missing values five times and take the average of these predicted values afterwards. I apply the method *predictive mean modelling* in the *mice* package in *R* for estimating the missing values, as I only deal with numerical variables. If the proportion of missing values is more than 50% for a variable, then I exclude this variable from the model. In my case, this is for the variable *Real Output from the Manufacturing Sector* (*OUTMS*). For the missing values where no imputation value can be estimated, I decide to delete the whole observation from the dataset. In the end, I remain with $T = 127$ complete observations for $k = 23$ variables that I use for further analysis.

3 Methodology

3.1 VAR model

The VAR model is given by

$$\mathbf{Y}_t = \mathbf{\Pi}_0 + \sum_{l=1}^{p} \mathbf{\Pi}_l \mathbf{Y}_{t-l} + \nu_t. \tag{1}$$

For each time period t, $\mathbf{Y}_t$ is a $k \times 1$ vector of dependent variables. The vector of constants is denoted by $\mathbf{\Pi}_0$ a $k \times 1$ vector, where $\mathbf{Y}_{t-l}$ is a $k \times 1$ vector of l periods lagged variables of $\mathbf{Y}_t$. It is more convenient to neglect the $k \times 1$ vector of constants $\mathbf{\Pi}_0$ for the remainder part of the research, because this study considers a standardized dataset. Subsequently, $\mathbf{\Pi}_l$ with a dimension of $k \times k$ represents the matrix of coefficients from the lag l past values of the k variables $\mathbf{Y}_{t-l}$ on the current value of the k variables $\mathbf{Y}_t$. Finally, ν_t is a $k \times 1$ vector of white noise error terms with $\nu_t \sim \mathcal{N}\ (0, \mathbf{\Sigma}_\nu)$, which is assumed to be independent from the explanatory variables $\mathbf{Y}_{t-l}$ and a multivariate normal distribution with covariance matrix $\mathbf{\Sigma}_\nu$. I let the past count for lag *1* until lag p, where p is subjective and dependent on the user input. This might be the first guess based on the serial correlation of the data. Another method is to consider the time-series properties of the dataset. The frequency at which the data is represented might give a proper indication for what maximum lag order could be considered.

In fact, it would be appropriate to estimate a VAR model by seemingly unrelated regressions, henceforth abbreviated as SUR. I hereby use the finding of among others the paper of Basu, Michailidis, et al. (2015). They found that a joint estimation of the lag coefficient matrices $\mathbf{\Pi}_l$ and the inverse of the covariance matrix $\mathbf{\Sigma}_\nu$ does not contribute evidently for better forecasts.

For making equation (1) more convenient in terms of an aggregated matrix, I also introduce the index $i \in \mathbb{R}^k$ to refer to variable i out of k variables and therefore $\mathbf{Y}_i$ is a $k \times 1$ vector and I represent the aggregated matrix accordingly.

All $k \times 1$ vectors of dependent variables $\mathbf{Y}_t$ over all T periods are collected into one big matrix $\mathbf{Y}$, thus the $k \times T$ matrix $\mathbf{Y} = \begin{bmatrix} \mathbf{Y}_1 & ,..., & \mathbf{Y}_T \end{bmatrix}$. I also collect all p times $k \times k$ lag coefficient matrices $\mathbf{\Pi}_l$ into one big matrix $\mathbf{\Pi}$ with dimension $k \times kp$, thus $\mathbf{\Pi} = \begin{bmatrix} \mathbf{\Pi}_1 & ,..., & \mathbf{\Pi}_p \end{bmatrix}$. In addition, I create a new explanatory variable $\mathbf{Q}_t$ that enables us to collect the vector $\mathbf{Y}_{t-l}$ for all p lags. This results in a $kp \times 1$ vector of $\mathbf{Q}_t = \begin{bmatrix} \mathbf{Y}_{t-1}^T & ,..., & \mathbf{Y}_{t-p}^T \end{bmatrix}^T$. Then I collect all $\mathbf{Q}_t$ for T periods into one big matrix $\mathbf{Q}$ with dimension $kp \times T$. This results in $\mathbf{Q} = \begin{bmatrix} \mathbf{Q}_1 & ,..., & \mathbf{Q}_T \end{bmatrix}$. Finally, I collect all T residual vectors ν_t into one $k \times T$ matrix such that $\mathbf{V} = \begin{bmatrix} \nu_1 & ,..., & \nu_T \end{bmatrix}$. After rewriting all terms accordingly, I get the following equation.

$$\underset{k\times T}{\mathbf{Y}} = \underset{k\times kp}{\mathbf{\Pi}} \times \underset{kp\times T}{\mathbf{Q}} + \underset{k\times T}{\mathbf{V}} \tag{2}$$

In vector notation I get the following representation with $\mathbf{Y}_i$ on the rows.

$$\begin{bmatrix} \mathbf{Y}_1^T \\ \vdots \\ \mathbf{Y}_k^T \end{bmatrix} = \begin{bmatrix} \mathbf{\Pi}_{1,1} & \dots \mathbf{\Pi}_{1,h} & \dots \mathbf{\Pi}_{1,kp} \\ \vdots & \ddots & \vdots \\ \mathbf{\Pi}_{k,1} & \dots \mathbf{\Pi}_{k,h} & \dots \mathbf{\Pi}_{k,kp} \end{bmatrix} \times \begin{bmatrix} \mathbf{Q}_1^T \\ \vdots \\ \mathbf{Q}_h^T \\ \vdots \\ \mathbf{Q}_{kp}^T \end{bmatrix} + \begin{bmatrix} \mathbf{V}_1^T \\ \vdots \\ \mathbf{V}_k^T \end{bmatrix}$$

For the remainder of this paper, I introduce the index numbers $m \in \mathbb{R}^k$ and $n \in \mathbb{R}^{kp}$. This means that $\mathbf{\Pi}_{m,n}$ refers to the effect of the nth lagged row of the lagged matrix $\mathbf{Q}$ on the current value of the mth variable.

3.2 Multi-step Adaptive elastic net on VAR models

3.2.1 Penalized methods

The idea of penalized estimation models as described by among others Zou and Hastie (2005) and Tibshirani (1996) in general is that extra bias has been introduced by the penalty terms λ at the benefit of having lower variance of the parameters' estimations. In this way a sparser model can be reached, which consists of less variables in the model and hence a lower amount of variance. The minimization formula of the sum of squared residuals or the log likelihood is affected by the penalty term. If redundant variables remain in the model, it basically generates extra variance without extra predictive power.

3.2.2 Ridge regression method

The ridge regression bounds the square of the coefficients by the L_2-norm. This implies that the sum of all squared elements in the $\mathbf{\Pi}$ matrix is smaller than an arbitrary non-zero value z representing the L_2-norm boundary on the coefficients. This implies $\sum_{m=1}^{k}\sum_{n=1}^{kp}\mathbf{\Pi}_{m,n}^2 < z$. The advantage of the ridge penalization is that it can shrink parameters and create grouping effects. This means that correlated variables in the model attain coefficients that are more in the vicinity of each other. This is the so-called de-correlation step. However, the disadvantage is that the model will not get a more parsimonious representation, since all variables will remain in the model. This means that the ridge regression is not able to select variables in the model. Especially in terms of selecting the amount of p lags and right sparsity, this is of crucial importance.

3.2.3 Lasso method

The lasso method improves in this by putting irrelevant coefficients to zero and as such eliminate variables from the model. The lasso method penalizes the absolute value of the coefficients by the L_1-norm. This implies that $\sum_{m=1}^{k}\sum_{n=1}^{kp}|\mathbf{\Pi}_l| < s$, where s is a non-zero value. In this way, lags of variables that are superfluous in the model can be simply left out. As the lasso estimation do not attain the oracle properties, the adaptive lasso method is proposed. The adaptive lasso method applies heterogeneous shrinking parameters across coefficients. This means that the strength of penalty differs across coefficients. The intuition behind this is that smaller coefficients are less relevant and are therefore penalized more heavily than the bigger coefficients.

3.2.4 Adaptive elastic net

The elastic net penalization method, also called a hybrid penalization method, is a weighted convex combination of the ridge penalization and the lasso penalization. Despite the fact that the lasso method improves in the variable selection part, it behaves worse in cases that the amount of variables k, p lags (k^2p parameters) in the model exceeds the amount of observations n. Also, in cases of multicollinearity. Zou and Hastie (2005) show that forecasts by ridge regressions outperformed the predictions of the lasso method. These two issues are relevant for the VAR setting, especially for macroeconomic variables. This is because the macroeconomic data frequency is mostly on a low-frequency basis, e.g. on a quarterly basis. In addition, the set of variables within the field of macroeconomics can be large and they are likely to exhibit coherence. The paper of Furman (2014) shows that the adaptive elastic net is a good method to estimate VAR models consistently, even in cases where the parameter space is large and where the lagged variables exhibit a certain degree of correlation.

3.2.5 Multi-step elastic net

Thus far Zou (2006) has shown that the adaptive lasso estimation method performs better than the regular lasso method in terms of coefficient estimation, because the heterogeneous weights make sure that relevant variables are not equally heavily biased as the irrelevant ones. The adaptive elastic net improves compared to the adaptive lasso by incorporating an extra regularization parameter, which makes sure that the variable selection is done more carefully, especially in the case of multicollinearity. This means that in high-dimensional settings, more regularization parameters are required to obtain better estimations as mentioned by Xiao and Xu(2015). This can be achieved by iterating the adaptive elastic net over different stages where the tuning parameters and weights change in each iteration r.

Xiao and Xu (2015) show that by iterating r times the adaptive elastic net for variable selection, a sparser model can be attained. Particularly in small sample cases, a sparser model is recommended. This is because in a finite small sample, the model cannot learn sufficiently from the data to estimate the coefficients in the unsparse model efficiently. This would obviously result in poor forecasts.

If the chosen maximum lag order is bigger than the maximum lag order according to the DGP, then the elastic net and lasso methods are able to shrink some of the coefficient matrix to zero on an individual level. However, they do not always select the correct lag order as a group. Therefore it results in an unstructured sparse coefficient matrix $\mathbf{\Pi}$ where some coefficients belong to the active set $\mathcal{S}_1$ and others to the inactive set $\mathcal{S}_0$. The indicator value *1* stands for coefficients that are not shrunk towards a zero value. The indicator value of *0* stands for coefficients that are shrunk towards zero. Figure 1 shows the graphical illustration of the problem. The DGP I use as example, consists of two variables with $n = 100$ observations and a VAR(1) coefficient matrix $\mathbf{\Pi}_{DGP}$ of

$$\mathbf{\Pi}_{DGP} = \begin{pmatrix} 0.7 & 0.2 \\ 0.2 & 0.7 \end{pmatrix}, \mathbf{\Sigma}_{\nu} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{3}$$

Fig. 2: Sparsity plot generated of the VAR coefficients estimated by the enet-VAR model generated by own code. I set the user defined maximum lag length arbitrarily on 4. The DGP is, however, a VAR(1) process.

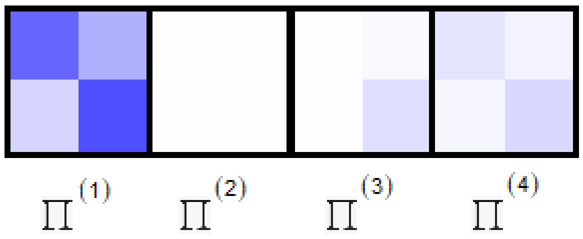


Figure 1 shows that 6 out of 16 (= 37.50%) coefficients of $\underset{3x12}{\mathbf{\Pi}} = \underset{3x3}{\mathbf{\Pi}_1} \cap \underset{3x3}{\mathbf{\Pi}_2} \cap \underset{3x3}{\mathbf{\Pi}_3} \cap \underset{3x3}{\mathbf{\Pi}_4}$ are shrunk to zero and thus fall in the inactive set $\mathcal{S}_0$. They are graphically depicted as white blocks. my aim is to find $\mathbf{\Pi}_1 \in \mathcal{S}_1$ and $\mathbf{\Pi}_2 \cup \mathbf{\Pi}_3 \cup \mathbf{\Pi}_4 \in \mathcal{S}_0$. However, I find that only some coefficients, in this case $\mathbf{\Pi}_2$ and $\mathbf{\Pi}_{kp=5} \in \mathbf{\Pi}_3$ are turned to zero.

The simulation study in section 3.6 elaborates on this matter in more detail with several high dimensional multivariate DGPs. The weighting term $\omega_{m,n}^{(r)}$ updates in each iteration r. Following the paper of Zou and Hastie (2005) and Zou and Zhang (2009) and after some rewriting, the optimization problem adopted to the multi-step elastic net function can be described in the following way for the VAR framework.

$$\mathbf{\Pi}_{elastic}^{(r)} = (1 + \frac{\lambda_2^{(r)}}{kT}) \times$$
$$\underset{\mathbf{\Pi}}{\text{argmin}} \left\{ \sum_{t=1}^{T} \sum_{m=1}^{k} \sum_{n=1}^{kP} ((\mathbf{Y}_{m,t} - \mathbf{\Pi}_{m,n}^{(r)} \mathbf{Q}_{n,t})^2 + \lambda_1^{(r)} \omega_{m,n}^{(r)} |\mathbf{\Pi}_{m,n}^{(r)}| + \lambda_2^{(r)} \mathbf{\Pi}_{m,n}^{2,(r)}) \right\}, \tag{4}$$

where λ_1 is the penalty term for the lasso method and λ_2 the penalty term for the ridge regression. The term $\omega_{m,n}$ defines the heterogeneous lasso penalization weights on the coefficients. This term takes care for heavier penalization of the smaller coefficients, and, softer penalization of the larger coefficients. In order to cope with the double shrinkage, I need to inflate the elastic net estimator. This is because the elastic net is estimated in a two stage procedure as explained by Zou and Hastie (2005). The first stage consists of estimating the ridge coefficients for each value of λ_2. The second stage consists of estimating the lasso coefficients conditioned on the ridge coefficients, λ_2 and λ_1. This double shrinkage incurs biased coefficients leading to poor forecasts. Following the paper of Furman (2014) I pre-multiply the elastic net equation 4 by the factor $(1+\frac{\lambda_2}{kT})$. Parameter k is the amount of equations that is being estimated at once in the VAR model, and, T the amount of time-series observations. The papers of Zou and Zhang (2009) and Furman (2014) elaborate more on this matter.

The initial weights for the adaptive lasso and adaptive elastic net are determined by $|\mathbf{\Pi}_{m,n}^{initial}|^{-\delta}$, where $|\mathbf{\Pi}_{m,n}^{initial}|$ are the initial estimates of the coefficients by OLS or ridge regression, and, δ a positive constant number. The paper of Zou and Zhang (2009) suggests that I can let $\delta \in (0.5, 1, 2)$, however I remain with the proposition of Xiao and Xu (2015) to let $\delta = 1$.

The additional step of the multi-step adaptive elastic net is that the adaptive elastic net for variable selection is conducted iteratively whereby the weights $\omega_{m,n}^{(r)}$ are updated in each iteration r. The weight in iteration $r > 1$ is determined by $|\mathbf{\Pi}_{m,n}^{(r-1)} \times \lambda^{(r-1)}\alpha^{(r-1)}|$. In order to make the estimation of the tuning parameters more feasible, I should rephrase equation 4 and introduce the term α which is the fraction of the total penalty belonging to the lasso method. That is, $\alpha = \frac{\lambda_1}{\lambda_1+\lambda_2}$, and, $(1-\alpha) = \frac{\lambda_2}{\lambda_1+\lambda_2}$. In addition, I aggregate λ_1 and λ_2 to λ. This means that $\lambda\alpha = \lambda_1$ and $\lambda(1-\alpha) = \lambda_2$.

The advantage of this re-parametrization is that I can better specify the two-dimensional searching grid of the tuning parameters, as α is specified on the interval [0,1]. So, instead of finding the parameters $\lambda_1 \in (0,\infty)$ and $\lambda_2 \in (0,\infty)$ in a two-dimensional grid, I simplified it to a two-dimensional searching grid for which $\lambda \in (0,\infty)$ and $\alpha \in (0,1)$. This allows us to reformulate equation (4) into,

$$\mathbf{\Pi}^{(r)}_{elastic} = (1+\frac{\lambda_2^{(r)}}{kT})\times$$
$$\underset{\mathbf{\Pi}}{\text{argmin}}\left\{\sum_{t=1}^{T}\sum_{m=1}^{k}\sum_{n=1}^{kP}((\mathbf{Y}_{m,t}-\mathbf{\Pi}^{(r)}_{m,n}\mathbf{Q}_{n,t})^2+\lambda^{(r)}(\alpha^{(r)}\omega^{(r)}_{m,n}|\mathbf{\Pi}^{(r)}_{m,n}|+(1-\alpha^{(r)})\mathbf{\Pi}^{2,(r)}_{m,n})\right\}, \tag{5}$$

The overview below shows that different values for parameters λ, ω and α lead to different estimation models. The models can vary between OLS, lasso, adaptive lasso, elastic net and adaptive elastic net. In case I iterate $r>1$ times over the adaptive elastic net VAR model, then I get the adaptive elastic net VAR model.

$$\begin{cases} if\lambda = 0,\ \text{r} = 0,\ \omega^{(r)}_{m,n} = 0 \text{ and } \alpha = 0 \rightarrow \textbf{VAR OLS} \\ if\lambda > 0,\ \text{r} = 0,\ \omega^{(r)}_{m,n} = 1 \text{ and } \alpha = 1 \rightarrow \textbf{VAR lasso} \\ if\lambda > 0,\ \text{r} = 1,\ \omega^{(r)}_{m,n} \neq 1 \text{ and } \alpha = 1 \rightarrow \textbf{VAR adaptive lasso} \\ if\lambda > 0,\ \text{r} = 0 \text{ and } \alpha = 0 \rightarrow \textbf{VAR ridge regression} \\ if\lambda > 0,\ \text{r} = 0,\ \omega^{(r)}_{m,n} = 1 \text{ and } \alpha \in (0,1) \rightarrow \textbf{VAR elastic net} \\ if\lambda > 0,\ \text{r} = 1,\ \omega^{(r)}_{m,n} \neq 1 \text{ and } \alpha \in (0,1) \rightarrow \textbf{VAR adaptive elastic net} \\ if\lambda > 0,\ \text{r} \in (1,..,R],\ \omega^{(r)}_{m,n} \neq 1 \text{ and } \alpha \in (0,1) \rightarrow \textbf{VAR multi-step aenet} \end{cases}$$

3.2.6 Optimization algorithm

There are two major methods to estimate the elastic net model. On the one hand there is the LARS algorithm as discussed by Efron et al. (2004), which calculates the whole solution path for the coefficients. On the other hand, there is the coordinate descent method as proposed by Friedman, Hastie, Höfling, et al. (2007) and Friedman, Hastie, and Tibshirani (2010). This method does not calculate the whole solution path of coefficients, but rather iteratively updates each coefficient $\Pi_{m,n}$ at a time while fixing the others until the convergence criterion has been met. The cyclic coordinate descent method as discussed in the paper of Wu, Lange, et al. (2008) is a famous numerical optimization method for solving penalized functions.

The advantages of the coordinate descent method compared to LARS method are that the computation time is shorter and the estimation more robust. This method is often used for functions with a lasso type of penalization. The argument is that the lasso part in the optimization problem is convex but not differentiable. However, this problem can be split up, such that the global minimum can be found. Suppose that I can split the equation in two parts. The first part denoted as $L(\Pi)$ is the SSR and the second part denoted as $P(\Pi)$ the penalty term. I can define the formula as $L(\Pi) + P(\Pi)$. For the sake of convenience, I consider each variable m separately. $L_{m,t}(\Pi) = \sum_{t=1}^{T} (\mathbf{Y}_{m,t} - \sum_{m=1}^{k} \sum_{n=1}^{kP} \mathbf{\Pi}_{m,n}\mathbf{Q}_{n,t})^2$. The penalty term $P_m(\Pi)$ for variable m is denoted as $\lambda \sum_{n=1}^{kp} (\alpha\omega_{m,n}|\mathbf{\Pi}_{m,n}| + (1-\alpha)\mathbf{\Pi}_{m,n}^2)$. Since there exists no analytical solution for the lasso part of penalty term $P(\Pi)$, I apply a numerical optimization procedure, the so-called coordinate descent algorithm. The basic idea behind this algorithm is to conduct a scalar optimization. I can achieve this by optimizing each coefficient of the $\mathbf{\Pi}$ matrix, while holding all other coefficients constant. The paper of Tseng (2001) confirms that the global convergence can be reached by solving the partitioned subproblems within the framework of the coordinate descent method. Suppose that I want to optimize the VAR model by the cyclical coordinate descent method, I require some inputs. As the DGP is unknown to the researcher, he puts in an own indication of the maximum lag order. Before I can use the coordinate descent algorithm to update the coefficient matrix $\mathbf{\Pi}$, I need to provide the algorithm an initial estimation of the coefficients. This is in turn based on the specified maximum lag order. I initialized the $\mathbf{\Pi}$ matrix with only ones in the cells.

The first step is then to restrict the model by excluding the lagged series h $\mathbf{Q}_{n=h}$ and exclude the coefficient $\mathbf{\Pi}_{m,n=h}$. The next step is to calculate the residuals of this restricted model as $W_{m,n\neq h,t}$, also called the partial residual. Subsequently, I plot the residuals of the restricted model $W_{m,n\neq h,t}$ on the lagged series $\mathbf{Q}_{n=h}$ that has been excluded from the full model. The result is a preliminary estimation of $\mathbf{\Pi}_{m,n=h}$. Finally this preliminary estimator goes through the soft-threshold operator and scaled by the product of the ridge penalty term $\lambda(1-\alpha)$ and $\mathbf{Q}^2_{n=h}$. The latter part is the variance of $\mathbf{Q}_{n=h}$[2]. The idea of the soft-threshold operator is that the preliminary scalar coefficient estimation is shrunk to zero if the absolute value of the estimation is smaller than a pre-defined constant term $\gamma \times \omega_{m,n}$. In my case γ is defined as the product of $\lambda\alpha$. This is the effect of the lasso part of the elastic net. Secondly, the soft-threshold operator shrinks the absolute value of coefficients if their value exceeds that of $\gamma \times \omega_{m,n}$. By scaling the coefficient estimate after the soft-threshold operator by the variance of $\mathbf{Q}_{n=h}$, the more the preliminary estimate will be shrunk in a quadratic way. This is the effect of the ridge regression part of the elastic net. Appendix 8.1 will elaborate on the derivation of the elastic net model in a VAR framework in detail. The resulting estimator $\widehat{\mathbf{\Pi}}_{m,n=h}$ is given in equation 3.2.6.

[2]As $\mathbf{Q}_{n=h}$ has been standardized, I can mathematically conclude that the square $\mathbf{Q}^2_{n=h}$ is equal to the variance.

$$\widehat{\mathbf{\Pi}}_{m,n=h} = \frac{\mathcal{ST}(\sum_{t=1}^{T} W_{m,n\neq h,t}\, Q_{n=h,t}; \lambda\alpha\omega_{m,n})}{\lambda(1-\alpha\omega_{m,n})\sum_{t=1}^{T} Q_{n=h,t}^{2}}$$
(6)

I conduct a pathwise coordinate descent strategy on the estimation of the VAR elastic net model. This means that I calculate solutions based on a sequence of lambdas and alphas. Lambda can range in a sequence from λ_{max} until λ_{min}, where λ_{max} entails the relatively highest level of penalization in the L1-norm and λ_{min} relatively the least. Following Nicholson, Matteson, and Bien (2014), I use $\lambda_{max} = e^{log(max(QY))}$ and $\lambda_{min} = e^{log(max(QY))/R}$, where R is for the depth of the lambda grid and I put $R = 10^2$. Due to the required computation time, I only choose for a length of the lambda grid $M = 5$. For the alpha values I use $\alpha_{min} = 0.75$ and $\alpha_{max} = 0.95$ with a step size of 0.1 such that the length of alpha grid is 3.

As the coordinate descent method finds the solution by iterating, I therefore take a convergence rate of $\epsilon = 10^{-4}$ where ϵ is the scaled absolute maximum difference between the $\mathbf{\Pi}_{new}$ and $\mathbf{\Pi}_{old}$ following Nicholson, Matteson, and Bien (2014).

As I also have a L2-penalization norm and therefore have the α to balance between the lasso and ridge penalization, I also calculate different solutions based on different values of α. I can create the solution path by fixing α at a certain value and find solutions for different values of λ conditioned on the level of α.

3.2.7 Multi-step AEN algorithm

From the inspiration of the paper of Xiao and Xu (2015), I adjust and redefine the algorithm in the following stepwise way such that it fits my

VAR elastic net framework.

Step 1: Initialize the weight scalar $\omega_{m,n}$ with $m \in [1,....,k]$ and $n \in [1,....,kp]$

Step 2: In the iteration for r $\in$ [1,....,R]; Estimate the elastic net equation (5) for which in the initialization the coefficient estimation $\widehat{\Pi}_{m,n}$ is a function of the ridge or ordinary least square regression, and, the weight scalar $\omega_{m,n}$ is initially put to a constant value of 1. From the first iteration onward, $\Pi^{(r \geq 1)}(\lambda, \alpha)$ is a function of the tuning parameters. The weights can be updated by: $\omega^{(r)}_{m,n} = |\mathbf{\Pi}^{(r-1)}_{m,n} \times \lambda^{(r-1)}\alpha^{(r-1)}|^{-\delta}$.

It is important to note that after each iteration the amount of coefficients gets smaller implying that the amount of lags gets smaller as well until I reach the point of convergence where $p_{optimal} \subseteq p_{max}$. My aim is that the algorithm should be able to turn all abundant coefficients to the inactive set $\mathcal{S}_0$ and retain only the coefficients of the true lags and variables in the active set $\mathcal{S}_1$.

Regarding the convergence criterion of the multi-step adaptive elastic net iterations, I define it in the similar way as for the coordinate descent algorithm, also with $\epsilon = 10^{-4}$.

3.3 Specification of the maximum lag length p_{max}

Prior to applying the penalized methods on the estimation of the coefficients in the VAR model, it is important to first have an indication of what the maximum lag length, say, p_{max} might be. After the specification of p_{max}, the penalized methods are able to filter out the redundant variables which includes lags of own and other variables in the model, such that the model becomes sparser.

By inspection at the serial correlation plots of the variables at its own past values, an initial guess of p_{max} might be deduced. Also, considering the time-series frequency might give an indication of p_{max}.

In this paper I therefore consider the serial correlation as guidance for the maximum lag length p_{max}. However in the simulation study, I choose for a lag length bigger than what the serial correlation plots imply. That is because I want to investigate whether the multi-step adaptive elastic net is able to recognize the correct sparsity pattern by itself. This also implies finding the true lag length. This means that if I find a serial correlation of order 4 in quarterly data, then I take for p_{max} a multiple of 4, e.g. 8. Because if data is correlated with its previous year, then it might be correlated with two years back as well.

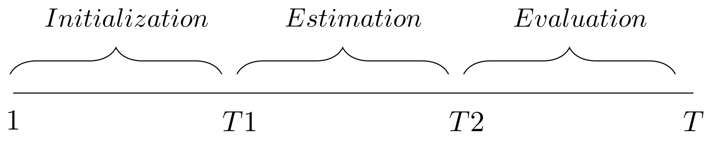
3.4 Specification of the tuning parameters

3.4.1 Specification of the penaly searching grid

I also need to specify the searching grid for the penalty terms λ and α. I let α range from 0.50 to 1 by incrementing with a step size of 0.1. The reason why I opt for a relatively higher α is that the variable selection results from the lasso penalization. Therefore, I give a large weight to the lasso penalization. The paper of Tabassum and Ollila (2017) affirms that the interval of $\alpha \in [0.5,0)$ is not that interesting as in that case the solutions will more likely move towards ridge regression solutions and as such will not result in clear variable selection. Following the papers of Nicholson, Matteson, and Bien (2014) and Friedman, Hastie, and Tibshirani (2010) I let λ decrement in a logarithmic-linear way. Where λ_{max} represents the value for which all coefficients $\Pi_{m,n}$ $\forall m, n$ are zero.

3.4.2 Data-based selection of penalty parameters λ, α

As e.g. used in Zou and Hastie (2005) for finding the tuning parameters α and λ, I apply the crossvalidation technique accordingly. However, due to the time dependence structure, the normal crossvalidation technique is not appropriate anymore. I opt for a rolling window estimation of the penalty parameter as explained in the paper of Song and Bickel (2011). This is because there is auto-correlation present in the data, which causes the data not to be independently identically distributed anymore. The underlying assumption of the traditional cross-validation technique is that the data should be independently identically distributed. For (V)AR models it is important to let the model be trained by past contiguous clustered data in order to take the sequential nature of time into account and to avoid the look-ahead bias that could occur if one would use future data as training input. I opt accordingly for a rolling crossvalidation technique as in line with the papers of Nicholson, Matteson, and Bien (2014) and Song and Bickel (2011). The idea is that the dataset is split up in three equivalent parts. The subperiods are calculated as $T_1 = \lfloor \frac{T}{3} \rfloor$ and $T_2 = \lfloor \frac{2T}{3} \rfloor$. The period from $t = 1$ until $t = T1 - 1$ is used to initialize the model, whereas the period from $t = T1$ until $t = T2 - 1$ is used for selecting the penalty parameters. The third period from $T2$ until T is used for the evaluation of the one step ahead forecasts.



I accordingly evaluate on the basis of one-step ahead forecasts. The proxy for the performance in terms of prediction is the MSFE.

$$MSFE(\lambda, \alpha) = \frac{1}{T_2 - 1 - T_1} \sum_{t=T_1}^{T_{2-1}} (\sum_{m=1}^{k} (\hat{Y}^{\lambda,\alpha}_{m,t+1} - Y_{m,t+1})^2 \tag{7}$$

3.5 Conventional information criterion on VAR models

The idea behind the information criterions AIC and BIC is that they tend to strike a balance between a good fit and the parsimony of a model. The practical difference between both information criterions is that the BIC penalizes model complexity heavier than the AIC does. Under the assumption that the errors are normally distributed, the information criterions can be defined in the following way as described in the paper of Luetkepohl (2009).

$$AIC = T\, ln \mid \tilde{\mathbf{\Sigma}} \mid + 2(pk^2 + k), \tag{8}$$

$$BIC = T\, ln \mid \tilde{\mathbf{\Sigma}} \mid + (ln\ T)(pk^2 + k), \tag{9}$$

where T is the amount of observations, $\tilde{\mathbf{\Sigma}_\nu}$ is the estimated covariance matrix of $\mathbf{\Sigma}_\nu$ which resembles the kxk covariance matrix of ϵ_t as introduced in equation 1. The covariance matrix is defined as

$$\mathbf{\Sigma}_\nu = \frac{1}{T} \sum_{t=1}^{T} \nu_t \nu_t^T, \tag{10}$$

whereby the analogy for the estimated covariance matrix holds.

The penalty term is for the AIC a constant factor of 2, whereas for the BIC model it is $ln\ T$. The factor after the penalty term, $pk^2 + k$, resembles the amount of parameters to estimate in the VAR(p) model. Altogether, this shows that the penalty terms of the information criterions exhibit a positive function with the dimension of the model, and, that the BIC criterion model penalizes the complexity of the model relatively heavier. Among others Kuha (2004) recommends the usage of both methods as complementary for the model selection procedure.

3.6 Simulation study

3.6.1 Setup

The aim is to simulate from several data generating processes following Kock and Callot (2015) with different true lag orders, different amount of variables, different levels of sparsity and different degree of stationarity. The particular focus is on large multivariate models, where the aim is to show how the models perform compared to small multivariate models. Different methods are evaluated in terms of sparsity and forecast performance.

As mentioned in the introduction, several literature studies show that the penalized estimation models are proper dimension reduction techniques for VAR models. The focus therefore lies on the comparison between these estimation techniques. Furman (2014) shows that the adaptive elastic net performs accurate forecasts within the VAR framework and the multi-step adaptive elastic net is an extension to it. Therefore I only focus on the comparison between the single-step elastic net with the multi-step elastic net. The techniques that are taken into consideration are elastic net (*enet*), adaptive elastic net with ridge weights (*aenetR*), adaptive elastic net with lasso weights (*aenetL*) and the multi-step adaptive elastic net (*maenet*).

For all DGPs, I use a time-invariant diagonal covariance matrix for the error term denoted as $\mathbf{\Sigma}_\nu$ with 0.10 as diagonal entries. The amount of simulation draws is $R = 100$.

It is important that the VAR model from which the DGP is simulated, is stationary. In order to establish stationarity, it is important that the condition $|\mathbf{I} - (\sum_{l=1}^{p} \lambda^l \mathbf{\Pi}_l)| = \mathbf{0}$ is met. Following Kock and Callot (2015), I use for DGP1 and DGP3 parameters generating stationary time series. However, for DGP2 I use parameters that generate data close to unit root.

In contrast to Kock and Callot (2015), I let the amount of variables only be in the limited set $k \in [5, 20]$ for DGP1 and DGP3, and the amount of time series observations be in the set $T \in [100, 1000]$ due to the required computation time, whereas they have a more expansive grid running from $k = 5$ to $k = 50$. Recall that the amount of parameters to be estimated grows quadratically in the amout of variables k. For DGP2, I use the set $k \in [5, 15]$ because of its complexity and that it is close to unit root requiring more computation power for estimating the coefficients.

DGP1 is based on a sparse $VAR_k(1)$ process where each variable is only dependent on its own previous lag with coefficient 0.5. This is a fairly simple case to consider, although very practical. The emphasis here is on the sparsity, since it is a diagonal VAR(1) coefficient matrix with zeroes on all the off-diagonal elements. One could consider this model as k separate AR(1) models collected in one VAR(1) coefficient matrix $\mathbf{\Pi}_1$. This type of models are interesting for variables that are collected on a monthly or quarterly basis and have a strong correlation with its own previous lag. Those are especially variables with a temporary behaviour. This might be interesting for financial variables where for instance a short-term momentum effect is present in the data.

DGP2 is based on a $VAR_k(4)$ process and has a diagonal block matrix structure with block size of *5x5*. This means that sets of variables exhibit a grouped structure, which is common in the field of macroeconomics. The entries in each block of $\mathbf{\Pi_1}$ consist of the value 0.15 and the entries in each block of $\mathbf{\Pi_4}$ -0.10. The coefficient matrices $\mathbf{\Pi_2}$ and $\mathbf{\Pi_3}$ are zero matrices. The largest root is 0.98, which implies a persistent behaviour in the time series and is close to unit root. This structure is in accordance with macroeconomic models based on quarterly data, where they exhibit a strong coherence with the past observations with a certain persistent behaviour. This is especially applicable in cases where for instance grouped macroeconomic variables are intertemporally related to each other but not to the variables outside the groups. Note that equation 3.6.1 represents a block matrix of $2x2$. This is only used for illustration purposes, since blocks of $5x5$ would take too much space to depict. Note that DGP2 is constructed for lag 1 and 4 as block diagonals of $5x5$. That means that the dimension of the matrix is a multiplication of 5.

DGP3 is based on a $VAR_k(1)$ process with $(-1)^{|m-n|}\psi^{|m-n|+1}$ and $\psi = 0.4$ on the entries of the $\mathbf{\Pi}_1$ matrix. The term $|m-n|$ displays the absolute distance with respect to the diagonal element of row m. This means that the diagonal elements are valued as 0.4 and the values on the off diagonal elements decrease exponentially in relation to the diagonal elements. In this situation the sparsity assumption is violated as none of the elements are zero. The further the off diagonal elements lie, the smaller the coefficients become. The aim for this DGP is to consider how the penalized methods perform in terms of sparsity and forecast performance despite the fact that the true DGP is not sparse. Equation 13 depicts the $\mathbf{\Pi}_1$ matrix for $k = 5$ variables. The practical relevance of such a VAR model is visible in cases where macroeconomic or financial variables are still intertemporally related but also sorted on the level of correlation they exhibit. That means that if variable k gets information from the previous lag of variable $k-1$ and $k+1$, it gets less information from $k-2$ or $k+2$ and even lesser from $k-3$ or $k+3$. That means that variables that are less coherent are placed further away from the diagonal and as such have a weaker intertemporal relation to the current variable k.

Considering that the true lag is unknown to the researcher, I use for

DGP1 and DGP3 the initialization guess for the potential maximum lag length as $p.max = 2$, since the data is generated on the basis of first order serial correlation in the data. That means that variables are related to their values in the previous quarter. For DGP2 the initial guess is set as $p.max = 8$, because this data is generated on the basis of the fourth order serial correlation. That means that the values of the current quarter are related to the values of the previous year, say, four quarters back. Given the fact that a researcher do not know the true lag, a guess could be made on the basis of the serial correlation. In this case, there is serial correlation at lag 4, thus lag 8 is a logical choice as well as data of eight quarters back might be of relevance as well. The aim is to show if the applied model is capable of selecting the right amount of lags by putting the redundant lags to zeroes and more importantly, finding the right sparsity pattern.

DGP 1: This dataset is generated by a sparse $VAR_k(1)$ process with a VAR coefficient matrix of

$$\mathbf{\Pi}_1^{DGP1} = \begin{pmatrix} 0.5 & & \\ & \ddots & \\ & & 0.5 \end{pmatrix} \tag{11}$$

DGP 2: This dataset is generated by a $VAR_k(4)$ process

$$\mathbf{\Pi}_1^{DGP2} = \left(\begin{array}{cc|cc} 0.15 & 0.15 & & \\ 0.15 & 0.15 & & \\ \hline & & 0.15 & 0.15 \\ & & 0.15 & 0.15 \end{array}\right)$$

$$\Pi_4^{DGP2} = \left(\begin{array}{cc|cc} -0.10 & -0.10 & & \\ -0.10 & -0.10 & & \\ \hline & & -0.10 & -0.10 \\ & & -0.10 & -0.10 \end{array}\right)$$

(12)

DGP 3: This dataset is generated by a non-sparse $VAR_k(1)$ process with a VAR coefficient matrix of

$$\mathbf{\Pi}_1^{DGP3} = \begin{pmatrix} \mathbf{0.40} & -0.16 & 0.06 & -0.03 & 0.01 \\ -0.16 & \mathbf{0.40} & -0.16 & 0.06 & -0.03 \\ 0.06 & -0.16 & \mathbf{0.40} & -0.16 & 0.06 \\ -0.03 & 0.06 & -0.16 & \mathbf{0.40} & -0.16 \\ 0.01 & -0.03 & 0.06 & -0.16 & \mathbf{0.40} \end{pmatrix} \tag{13}$$

In order to evaluate the performance of the *multi-step adaptive enet* algorithm, the following criteria are used which are partly based on the paper of Kock and Callot (2015), Tibshirani (1996), Zou and Hastie (2005) and the relatively more recent paper of Schnücker (2019).

Inclusion of true model

By this criterion I want to measure if the resulting coefficient matrix contains the true active coefficients, say $\mathcal{S}_1^{true} \subseteq \mathcal{S}_1^{model}$. This is proxied by an indicator variable where that takes value 1 if this criterion is met and 0 otherwise. Subsequently, the results are averaged over R simulations.

$$\frac{1}{R}\sum_{r=1}^{R} I(\mathcal{S}_1^{true} \subseteq \mathcal{S}_1^{model}). \tag{14}$$

The share of inclusion of the relevant variables

This proxy is used to measure the performance of the model by evaluating the accuracy of selecting relevant variables. In mathematical terms it can be defined

$$\frac{1}{R}\sum_{r=1}^{R}(\frac{1}{length(\mathcal{S}_1^{model})}\sum_{m=1}^{k}\sum_{n=1}^{kp}I(\widehat{\mathbf{\Pi}}_{m,n}=1,\mathbf{\Pi}_{m,n}=1)) \tag{15}$$

Evaluating sparsity

Interesting is to evaluate the sparsity of the model. This can be done by counting all the nonzero coefficients. Then the result can be averaged over all R simulations. In mathematical terms it can be defined as

$$\frac{1}{R}\sum_{r=1}^{R}(\sum_{m=1}^{k}\sum_{n=1}^{kp}I(\mathbf{\Pi}_{m,n}\neq 0)) \tag{16}$$

Mean squared error The mean squared error is the squared deviation of the estimated coefficient $\hat{\mathbf{\Pi}}_{m,n}$ compared to the the true coefficient $\mathbf{\Pi}_{m,n}$ averaged over all Monte Carlo replications.

$$MSE(\lambda,\alpha)=\frac{1}{R}\sum_{r=1}^{R}(\sum_{m=1}^{k}\sum_{n=1}^{kp}(\widehat{\mathbf{\Pi}}_{m,n}^{\lambda,\alpha}-\mathbf{\Pi}_{m,n})^2) \tag{17}$$

Mean squared forecast error The mean squared forecast error is the squared deviation of the estimated variable $\hat{Y}_m$ h-periods ahead compared to the the true value Y_m averaged over all Monte Carlo replications. Following the paper of Kock and Callot (2015) I apply an one-step ahead forecast window, thus $h=1$.

$$MSFE(\lambda,\alpha) = \frac{1}{R}\sum_{r=1}^{R}\Big(\frac{1}{T_2-h-T_1}\sum_{t=T_1}^{T_{2-h}}\Big(\sum_{m=1}^{k}(\hat{Y}^{\lambda,\alpha}_{m,t+h} - Y_{m,t+h})\Big)^2\Big) \quad (18)$$

4 Simulation results

As introduced in the methodology section, I use three different DGPs to show the performance of several models. The aim is to show the performance of the multi-step adaptive elastic net compared to the single step (adaptive) elastic net methods.

DGP1 is generated by a sparse VAR(1) process, whereas DGP2 is generated by a block-VAR(4) model and DGP3 by a non-sparse VAR(1) model. For DGP2, the largest amount of parameters should be estimated. That is $k^2 * p.max(= 8)$, which amounts for $k = 5$ to 200 parameters and for $k = 15$ to 1,800 parameters. For DGP1 and DGP3, however, it reduces to 50 respectively 800 parameters.

All results are based on the average of the amount of simulations. This is $R = 100$ for DGP1 and DGP3 and $R = 25$ for DGP2. Although I am aware that the amount of simulations for DGP2 is low for statistical conclusions, it nevertheless provides useful insights with consistent estimates. Because it still shows the performance of the *maenet* estimator compared to the others with results that converge in probability over the amount of simulation runs.

Overall the results show that the multi-step adaptive net performs better in terms of sparsity, share of relevant variables and difference between the estimated and true coefficient values. The forecast performance stays relatively the same, while the model has become much sparser using the multi-step adaptive elastic net. The model in general performs better in a VAR(1) setting compared to a VAR(4) model. When the sample increases, the multi-step adaptive elastic net in general provides better results than in a low time dimension. I can infer this from the improvement in the share of relevant variables, evaluating the level of sparsity and mean squared distance from true coefficients. Even in the high-dimensional case with $k = 20$ variables, the model is capable of getting a much sparser model while keeping the forecast performance more or less in line with the other competitive methods or even improves.

Table 1: DGP1 simulation results

DGP1		k = 5, T = 100	k = 5, T = 1000	k = 20, T = 100	k = 20, T = 1000
active variables	oracle	5.00	5.00	20.00	20.00
	OLS	50.00	50.00	800.00	800.00
	enet	42.23	42.03	656.09	668.27
	aenetL	6.68	6.42	74.24	43.00
	aenetR	17.08	14.98	233.79	201.12
	maenet	6.22	6.06	43.85	39.29
sparsity	oracle	0.00	0.00	0.00	0.00
	OLS	0.00	0.00	0.00	0.00
	enet	19.50	19.41	379.35	379.47
	aenetL	30.51	29.26	709.91	713.21
	aenetR	30.24	31.32	564.15	595.85
	maenet	29.30	28.83	723.47	711.82
share of relevant variables	oracle	1.00	1.00	1.00	1.00
	OLS	0.10	0.10	0.03	0.03
	enet	0.12	0.12	0.03	0.03
	aenetL	0.78	0.80	0.27	0.47
	aenetR	0.30	0.35	0.08	0.10
	maenet	0.82	0.84	0.45	0.52
inclusion of true model	oracle	1.00	1.00	1.00	1.00
	OLS	1.00	1.00	1.00	1.00
	enet	1.00	1.00	0.91	1.00
	aenetL	0.93	1.00	0.75	1.00
	aenetR	1.00	1.00	0.63	1.00
	maenet	0.94	1.00	0.63	1.00
MSE from true PI	oracle	0.000	0.000	0.000	0.000
	OLS	0.587	0.049	14.865	0.854
	enet	0.444	0.045	9.247	0.753
	aenetL	0.127	0.007	1.154	0.069
	aenetR	0.210	0.015	3.952	0.207
	maenet	0.125	0.007	1.356	0.069
MSFE	oracle				
	OLS				
	enet	0.874	0.771	1.124	0.724
	aenetL	0.821	0.764	0.809	0.749
	aenetR	0.820	0.767	0.912	0.737
	maenet	0.869	0.771	0.815	0.749
lag selection	AIC	1.01	1.00	1.16	1.00
	BIC	1.00	1.00	1.00	1.00
	enet	2.00	2.00	2.00	2.00
	aenetL	1.52	1.61	2.00	2.00
	aenetR	2.00	2.00	2.00	2.00
	maenet	1.47	1.47	2.00	2.00

This table presents the overview of the $R = 100$ simulation results of DGP1. The majority of the criterion is evaluated against different models: *oracle*, *OLS*, *enet*, *aenetL*, *aenetR* and *maenet* where *oracle* stands for knowing the true DGP matrix, *OLS* for an ordinary least squares estimation of the coefficient matrix, *enet* for elastic net, *aenet* for adaptive elastic net with weights based on initial lasso L or ridge R weights.

Table 2: DGP2 simulation results

DGP2

active variables

k = 5			*k = 15*		
	T = 100	*T = 1000*		*T = 100*	*T = 1000*
oracle	50.00	50.00	oracle	150.00	150.00
ols	200.00	200.00	ols	1380.00	1800.00
enet	148.00	170.46	enet	1042.29	1420.04
aenetL	26.54	72.23	aenetL	109.33	331.50
aenetR	49.46	75.39	aenetR	260.33	355.92
maenet	16.92	45.15	maenet	52.08	136.92

sparsity

k = 5			*k = 15*		
	T = 100	*T = 1000*		*T = 100*	*T = 1000*
oracle	0.00	0.00	oracle	0.00	0.00
ols	0.00	0.00	ols	0.00	0.00
enet	46.77	49.62	enet	798.88	749.63
aenetL	145.92	122.23	aenetL	1689.00	1459.71
aenetR	125.85	122.23	aenetR	1529.50	1437.21
maenet	154.85	127.46	maenet	1747.29	1552.75

share of relevant variables

k = 5			*k = 15*		
	T = 100	*T = 1000*		*T = 100*	*T = 1000*
oracle	1.00	1.00	oracle	1.00	1.00
ols	0.25	0.25	ols	0.11	0.08
enet	0.28	0.29	enet	0.11	0.11
aenetL	0.55	0.69	aenetL	0.36	0.44
aenetR	0.45	0.66	aenetR	0.21	0.42
maenet	0.60	0.96	maenet	0.46	0.89

inclusion of true model

k = 5			*k = 15*		
	T = 100	*T = 1000*		*T = 100*	*T = 1000*
oracle	1.00	1.00	oracle	1.00	1.00
OLS	1.00	1.00	OLS	0.00	1.00
enet	0.00	1.00	enet	0.00	0.71
aenetL	0.00	0.46	aenetL	0.00	0.00
aenetR	0.00	0.46	aenetR	0.00	0.00
maenet	0.00	0.00	maenet	0.00	0.00

MSE from true PI

k = 5			*k = 15*		
	T = 100	*T = 1000*		*T = 100*	*T = 1000*
oracle	0.000	0.000	oracle	0.000	0.000
OLS	4.149	0.225	OLS	NA	2.261
enet	1.613	0.219	enet	7.216	1.783
aenetL	1.166	0.149	aenetL	4.128	0.682
aenetR	1.161	0.153	aenetR	5.306	0.790
maenet	1.486	0.144	maenet	5.110	0.697

MSFE

k = 5			*k = 15*		
	T = 100	*T = 1000*		*T = 100*	*T = 1000*
oracle			oracle		
OLS			OLS		
enet	0.579	0.406	enet	0.774	0.341
aenetL	0.465	0.456	aenetL	0.378	0.372
aenetR	0.533	0.516	aenetR	0.765	0.374
maenet	0.567	0.405	maenet	0.755	0.393

lag selection

k = 5			*k = 15*		
	T = 100	*T = 1000*		*T = 100*	*T = 1000*
AIC	3.69	4.00	AIC	5.33	4.00
BIC	1.08	4.00	BIC	2.67	1.92
enet	8.00	8.00	enet	8.00	8.00
aenetL	6.85	7.85	aenetL	7.83	8.00
aenetR	8.00	8.00	aenetR	8.00	8.00
maenet	6.46	5.00	maenet	7.58	6.63

This table presents the overview of the $R = 25$ simulation results of DGP2. The majority of the criterion is evaluated against different models: *oracle*, *OLS*, *enet*, *aenetL*, *aenetR* and *maenet* where *oracle* stands for knowing the true DGP matrix, *OLS* for an ordinary least squares estimation of the coefficient matrix, *enet* for elastic net, *aenet* for adaptive elastic net with weights based on initial lasso L or ridge R weights.

Table 3: DGP3 simulation results

DGP3

active variables

$k = 5$			$k = 20$		
	$T = 100$	$T = 1000$		$T = 100$	$T = 1000$
oracle	25	25	oracle	400	400
ols	50	50	ols	800	800
enet	43.59	44.46	enet	645.815	671.472
aenetL	11.76	19.70	aenetL	87.426	109.792
aenetR	20.26	25.79	aenetR	232.167	229.528
maenet	10.23	18.38	maenet	55.741	96.547

sparsity

$k = 5$			$k = 20$		
	$T = 100$	$T = 1000$		$T = 100$	$T = 1000$
oracle	0	0	oracle	0	0
ols	0	0	ols	0	0
enet	0	0	enet	0	0
aenetL	0	0	aenetL	0	0
aenetR	0	0	aenetR	0	0
maenet	0	0	maenet	0	0

share of relevant variables

$k = 5$			$k = 20$		
	$T = 100$	$T = 1000$		$T = 100$	$T = 1000$
oracle	1.00	1.00	oracle	1	1
ols	0.50	0.50	ols	0.5	0.5
enet	0.53	0.54	enet	0.509	0.514
aenetL	0.85	0.89	aenetL	0.69	0.832
aenetR	0.72	0.79	aenetR	0.571	0.665
maenet	0.88	0.92	maenet	0.772	0.896

inclusion of true model

$k = 5$			$k = 20$		
	$T = 100$	$T = 1000$		$T = 100$	$T = 1000$
oracle	1.00	1.00	oracle	1	1
OLS	1.00	1.00	OLS	1	1
enet	0.11	0.38	enet	0	0
aenetL	0.00	0.00	aenetL	0	0
aenetR	0.00	0.00	aenetR	0	0
maenet	0.00	0.00	maenet	0	0

MSE from true PI

$k = 5$			$k = 20$		
	$T = 100$	$T = 1000$		$T = 100$	$T = 1000$
oracle	0.000	0	oracle	0	0
OLS	0.564	0.051	OLS	14.487	0.831
enet	0.430	0.047	enet	8.381	0.75
aenetL	0.333	0.033	aenetL	1.976	0.209
aenetR	0.320	0.031	aenetR	4.003	0.273
maenet	0.343	0.034	maenet	2.422	0.214

MSFE

$k = 5$			$k = 20$		
	$T = 100$	$T = 1000$		$T = 100$	$T = 1000$
oracle			oracle		
OLS			OLS		
enet	0.841	0.722	enet	1.230	0.633
aenetL	0.827	0.720	aenetL	0.847	0.610
aenetR	0.822	0.719	aenetR	1.060	0.615
maenet	0.837	0.720	maenet	0.874	NA

lag selection

$k = 5$			$k = 20$		
	$T = 100$	$T = 1000$		$T = 100$	$T = 1000$
AIC	1.00	1.00	AIC	1.037	1
BIC	1.00	1.00	BIC	1	1
enet	2.00	2.00	enet	2	2
aenetL	1.76	1.88	aenetL	2	2
aenetR	2.00	2.00	aenetR	2	2
maenet	1.70	1.82	maenet	2	2

This table presents the overview of the $R = 25$ simulation results of DGP3. The majority of the criterion is evaluated against different models: *oracle*, *OLS*, *enet*, *aenetL*, *aenetR* and *maenet* where *oracle* stands for knowing the true DGP matrix, *OLS* for an ordinary least squares estimation of the coefficient matrix, *enet* for elastic net, *aenet* for adaptive elastic net with weights based on initial lasso L or ridge R weights.

4.1 Parameter estimation

For evaluating the performance in terms of parameter estimation, I consider the bias and estimation errors as proxy.

In order to visualize the accuracy of parameter estimation by the different methods, I follow the paper of Kock and Callot (2015) by considering the first parameter in the $\mathbf{\Pi}$ matrix denoted as $\Pi[1,1]$. However, instead of depicting density graphs as they do, I opt for boxplots. The reason for this choice is that boxplots show the distribution of the data with the median as center gauge. As the amount of simulation $R = 100$ is not much and therefore the average value of $\Pi[1,1]$ might be biased, a more robust proxy for the center of $\Pi[1,1]$ estimates like the median is more appropriate. The median is less sensitive to the variability in the simulation set. Moreover, I only choose to discuss the boxplots of the high dimensional settings, as this is where the focus particularly on is.

The boxplots in figure 3 show that the true $\Pi[1,1]$ is 0.5 for DGP1, which is a sparse VAR(1) process. Furthermore, it seems that in a small sample the OLS, *enet* and *aenetL* are downward biased with average median values of 0.37, 0.36 and 0.38. Coefficient estimates by the *aenetL* and *maenet* seem to be closer to the true value of 0.5, where *maenet* provides an average median with the closest estimate of 0.44. Considering the boxplot below in figure 3, all methods perform approximately equal in terms of parameter estimation and converge towards 0.50 with a current median value of 0.49. Only the *maenet* estimates are correct with a median of 0.5. This is the same as the true value of $\Pi[1,1]$. Interesting as well is to note that as the sample gets larger, the spread in the estimation becomes smaller. In terms of statistics, this implies consistent estimators.

Table 1 of DGP1 shows that the deviation from the true $\mathbf{\Pi}$ is the largest for the *OLS* and the smallest for the *maenet* and thereafter for the *aenetL*. This pattern is visible in the small $k = 5$ as well as in the large dimension $k = 20$. This pattern is visible in the small sample $T = 100$ as well as in the large sample $T = 1000$. Interesting to see is that all methods learn from the data as the sample size increases resulting in lower estimation errors. The reason that the *OLS* and *enet* show the highest errors is that the *OLS* do not shrink the coefficients like the penalization methods do and as such estimates all coefficients including the redundant ones. For the *enet*, however, the penalties are not efficiently chosen as it is the case for the adaptive variants such as *aenetL* and *aenetR*. The reason that the *aenetL* shows lower deviation errors than *aenetR* is that *aenetL* uses the lasso estimates as initial weighting estimation matrix applying a stronger variable selection than using the ridge regression estimates as initial weighting matrix. The *maenet* improves this by iterating the estimation steps using updated weighting matrices in each iteration r until the solutions converge.

Table 2 of DGP2, which is a sparse VAR(4) process, shows slightly different results. Although *OLS* shows again the highest deviation from the true $\mathbf{\Pi}$, *maenet* does not show the best results anymore in terms of deviation from true $\mathbf{\Pi}$. It appears that *aenetL* performs on average better, except for $k = 5, T = 100$ where the *aenetR* seems to perform better on average instead. Given the low set of simulations, this last result might be a coincidence.

The boxplots of DGP2 in figure 4 show that all penalized methods except *enet* heavily penalizes the $\Pi[1, 1]$ to 0 in small samples ($T = 100$). The *OLS* estimate shows a high median biased estimate for $\Pi[1, 1]$ with a value of over 0.50 for $k = 15, T = 100$. It seems that the penalized methods have difficulties with correctly estimating the coefficients in small samples with high dimensional settings.

Table 3 shows the simulation results for DGP3, which is a non-sparse VAR(1) process. It is obvious that the *maenet* does not show the lowest deviation from the true $\Pi[1,1]$ anymore. This might be because this DGP is not sparse and as such no variables need to be selected, but only estimated. Due to the iteration procedure, *maenet* might unjustifiably turn coefficients to zero or shrink them too much. For $k = 5$, *aenetR* performs the best. This might be explained from the fact that the DGP is constructed such that the off-diagonal elements decrease exponentially from the diagonal. As $k = 5$ is a low dimensional setting, the off-diagonal elements are not that close to zero and a softer penalty is more appropriate. However for $k = 20$, the *aenetL* and *maenet* performs better than *aenetR*. This can be explained from the fact that the matrix for $k = 20$ with size 20x20 is much bigger than for $k = 5$ with 5x5 and therefore the distance of elements towards the diagonal increases. As the amount of elements with a growing distance to the diagonal increases and the coefficient is an exponential decreasing function from the diagonal element, the values on the outer off-diagonal might be close to zero for which a zero coefficient estimate (variable deselected) would be better instead of a biased nonzero value. Therefore, the methods that impose a stronger variable selection, seem to be preferred.

The boxplots of DGP3 in figure 5 show for $k = 20$ that the *maenet* estimates $\Pi[1,1]$ are the least biased estimates with a value of 0.34 in small samples. Moreover, it seems that *maenet* shows an unbiased estimate for $\Pi[1,1]$ with a value of 0.4 being the same as the true $\Pi[1,1]$ value. All estimators seem to be consistent, where the *maenet* estimator seems to be consistent and unbiased for $T = 1000$ (large sample). The consistency seems to be stronger for the adaptive elastic net variants compared to the normal elastic net and OLS by considering the spreads of the estimators for $T = 1000$.

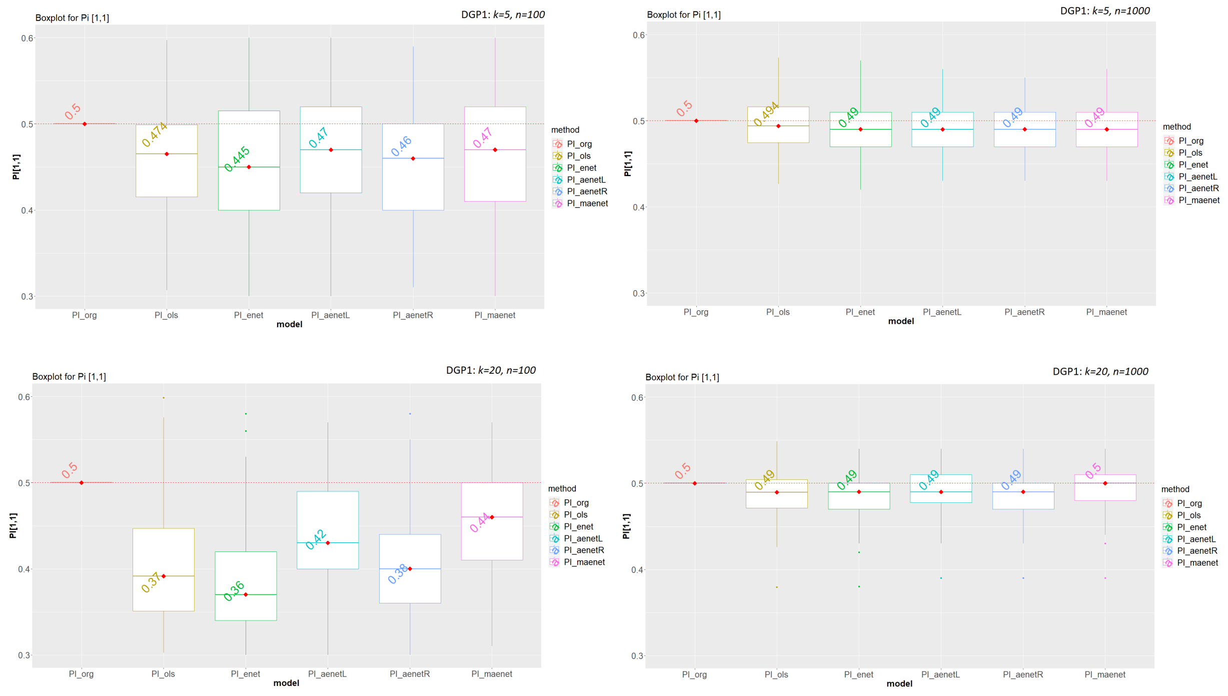


Fig. 3: Simulation plots for $\mathbf{\Pi}[1{,}1]$ with $k = 5$ and $k = 20$ (large dimension) and $T = 100$ (small sample) and $T = 1,000$ (large sample). $\mathbf{\Pi}$ estimated by *enet*, *aenetL*, *aenetR* and *maenet*.

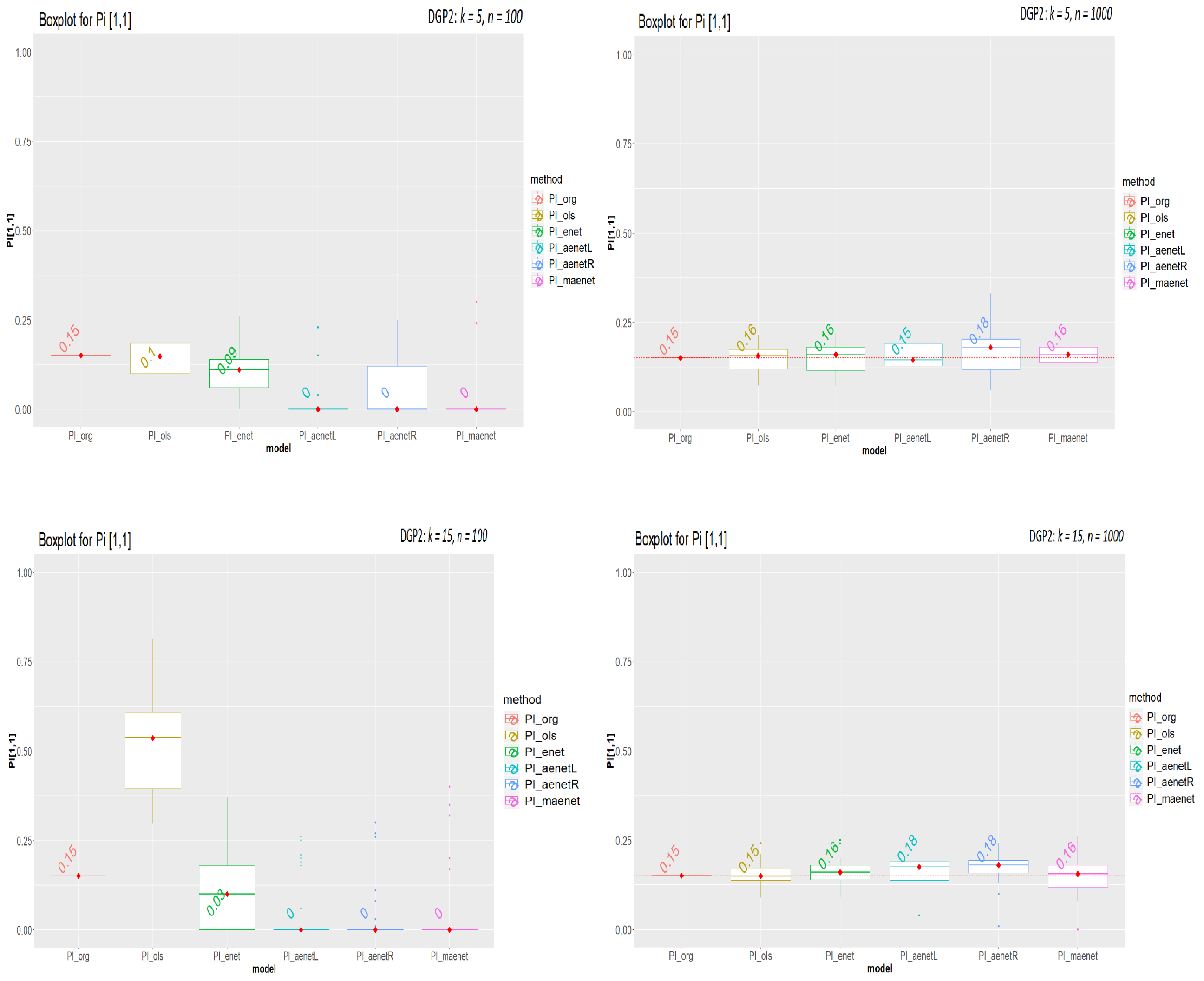


Fig. 4: Simulation plots for $\mathbf{\Pi}[1,1]$ with $k = 5$ (small dimension) and $k = 15$ (high dimension) for $T = 100$ (small sample) and $T = 1,000$ (large sample). $\mathbf{\Pi}$ estimated by *OLS, enet, aenetL, aenetR* and *maenet*.

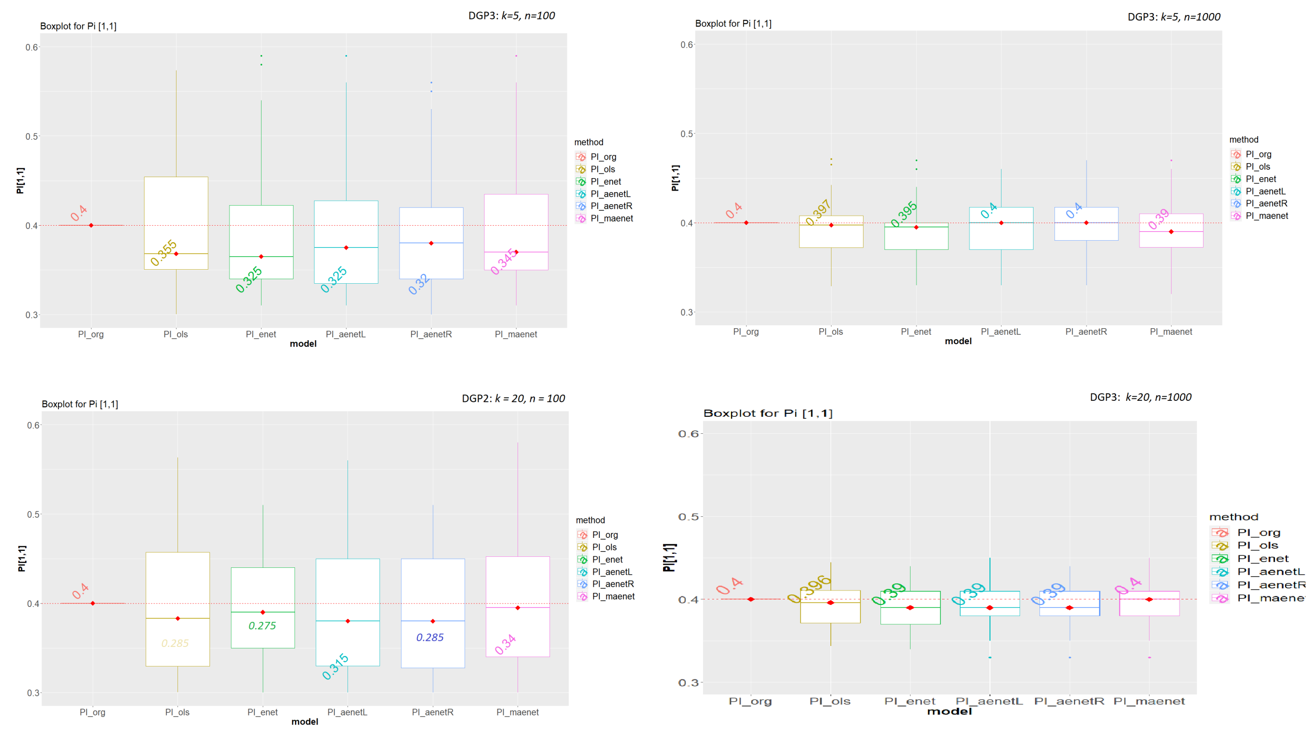


Fig. 5: Simulation plots for $\mathbf{\Pi}[1,1]$ with $k = 20$ (large dimension) and $T = 100$ (small sample) and $T = 1,000$ (large sample). $\mathbf{\Pi}$ estimated by *enet, aenetL, aenetR* and *maenet*.

4.2 Sparsity performance

In order to evaluate the performance in terms of sparsity, I consider the numerical results from table 1 of DGP1, table 2 of DGP2 and table 3 of DGP3 the *active variables*, *level of sparsity*, *share of relevant variables* and the *inclusion of true model.* In addition, I consider the graphical coefficient plots in figure 6 of DGP1, figure 7 of DGP2 and figure 8 of DGP3.

4.2.1 Active variables

For DGP1 it is visible that the *maenet* provides the amount of active variables that is the closest to the true amount of active variables in all settings where the dimension varies from $k = 5$ tot $k = 20$ and where the sample varies from $T = 100$ tot $T = 1000$. The good follow up is by the *aenetL* because of its lasso weights as input resulting in a stronger variable selection than *aenetR* does. It is also visible that *OLS* and *aenet* performs the worst, which is logical since the *OLS* does not perform variable selection and *enet* does not perform efficient variable selection because of the homogeneous weights used for all parameters. Interesting to note is that *maenet* strongly outperforms the other methods when $k = 20$. However, as the sample size increases from $T = 100$ to $T = 1000$, the performance of *maenet* and *aenetL* gets closer.

For DGP2 it is remarkable that for small samples, *maenet* estimates too many parameters as zero while they are actually nonzero. The adaptive single step elastic net seems to perform better in terms of estimating the correct amount of active variables. However, when the sample size increases from $T = 100$ to $T = 1000$, it seems that *maenet* is the most accurate.

For DGP3, on contrary, *maenet* underestimates the amount of variables in all cases. Here it seems that *aenetR* is the best among the estimators.

This can be explained from the fact that DGP3 is generated from a non-sparse VAR(1) process where all coefficients within lag 1 are actually active. The aim is only to put the coefficients on zero for lag 2, because we use $p.max = 2$, but none for lag 1 which is different from DGP1 and DGP2. That means that there is appearantly no need for a strong variable selection regime such as *aenetL* or more restrcivtive the *maenet*.

4.2.2 Sparsity plots

For this subsection I consider the sparsity plots of DGP1, DGP2 and DGP3 in figures 6, 7 respectively 8. In all cases it is visible that *maenet* presents the sparsest coefficient plot. It becomes particularly visible for small samples where the distinction between methods is more evident. The methods seem to get closer results as the sample size increases. This finding is mainly relevant for macroeconomic datasets with small sample sizes that are generated from a sparse model. The sparsity plots are supported by the numerical results under the row *sparsity* of the tables where it is visible that *maenet* is the sparsest model followed by *aenetL*. Exception is for DGP3 where for some strange reason no numerical results are available, however, the figures show that *maenet* provides the sparset model as well for DGP3.

In addition, *fraction of relevant variables* also show that *maenet* has the highest rate in all case, which implies that it performs a good variable selection. It is also consistent as it moves more towards 1 as the sample size increases.

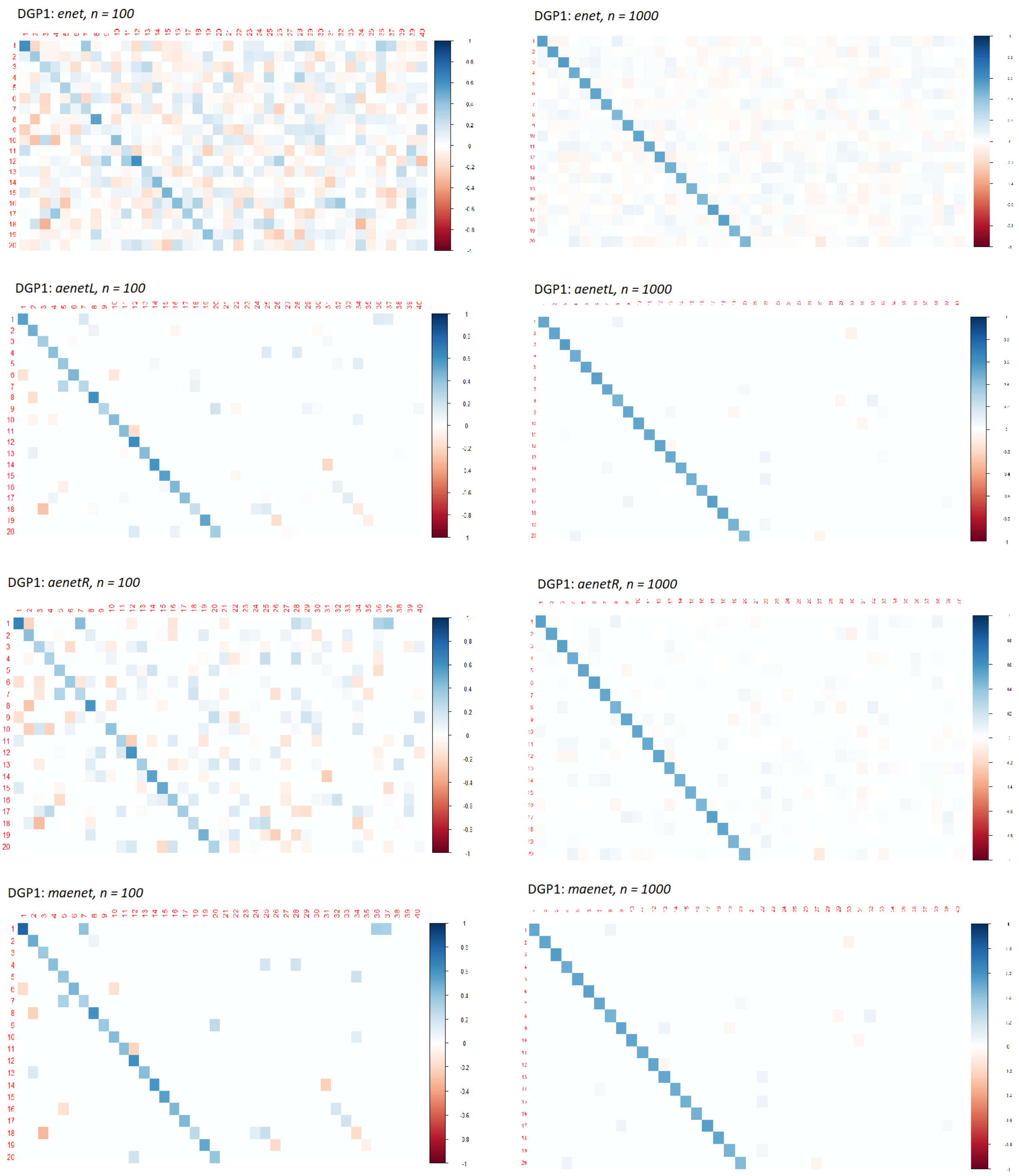


Fig. 6: Simulation plots for the $\mathbf{\Pi}$ coefficient matrix of DGP1 with $k = 20$ (large dimension) and $T = 100$ (small sample) and $T = 1,000$ (large sample). A representation of the coefficient plot estimated by *enet*, *aenetL*, *aenetR* and *maenet*.

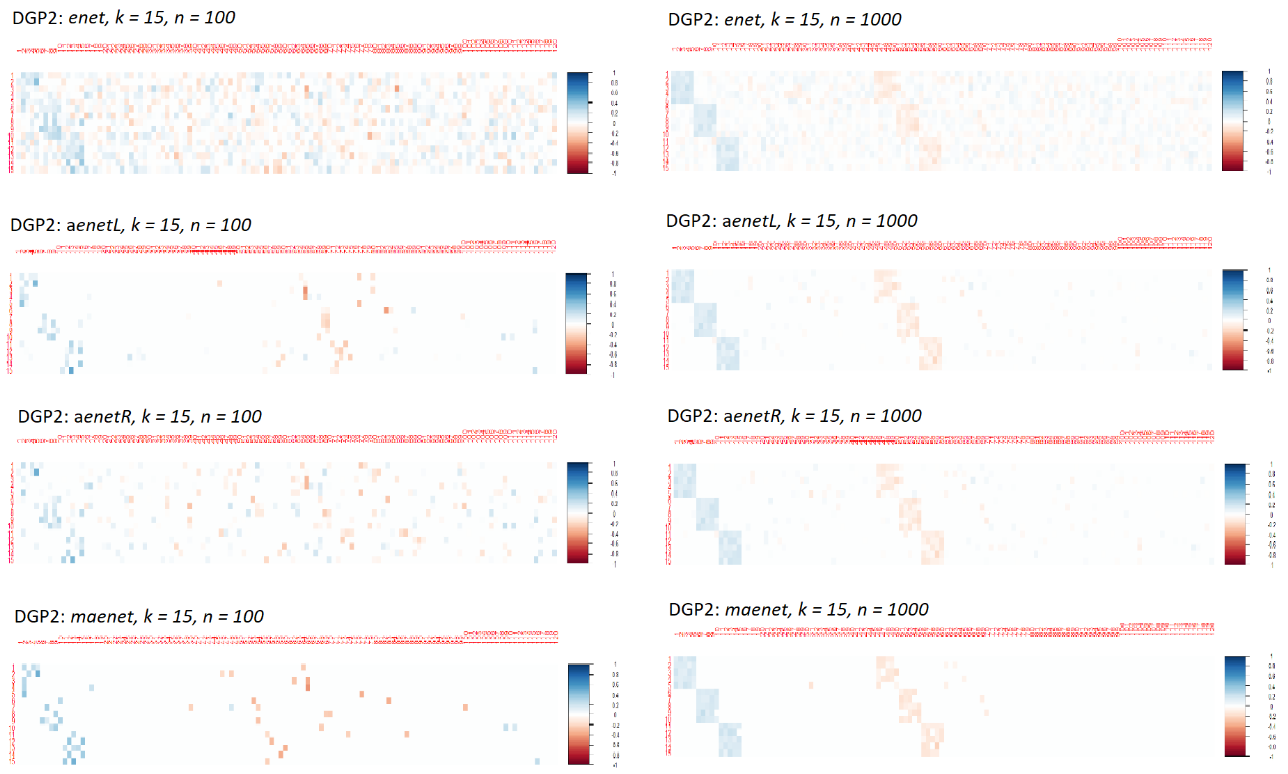


Fig. 7: Simulation plots for the $\mathbf{\Pi}$ coefficient matrix of DGP2 with $k = 20$ (large dimension) and $T = 100$ (small sample) and $T = 1,000$ (large sample). A representation of the coefficient plot estimated by *enet*, *aenetL*, *aenetR* and *maenet*.

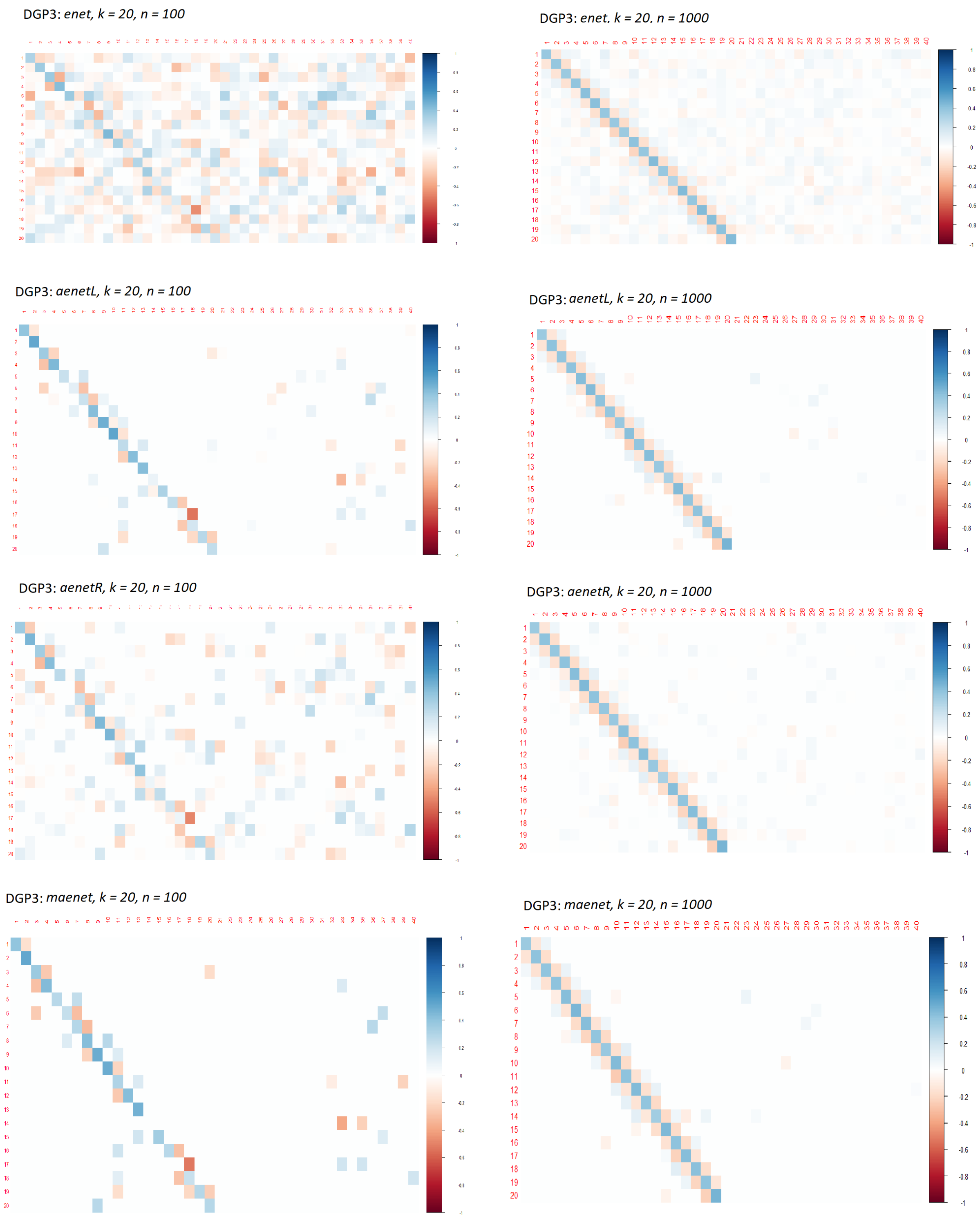


Fig. 8: Simulation plots for the $\mathbf{\Pi}$ coefficient matrix of DGP3 with $k = 20$ (large dimension) and $T = 100$ (small sample) and $T = 1,000$ (large sample). A representation of the coefficient plot estimated by *enet*, *aenetL*, *aenetR* and *maenet*.

4.2.3 Inclusion of true model

Considering the *share of relevant variables*, it is remarkable that all penalized models have a high rate of simulation runs that include the true model. This result increases along the increment of the sample size, but decreases along the increment of the dimension. All OLS runs contain the true model, which is logical as the OLS does not perform variable selection and the DGP is just a diagonal matrix with 0.5 on the entries.

However, it seems that alle estimation methods have difficulties finding the true model for DGP2 in the small sample, even the OLS. However, as the sample size increases, the methods have a higher probability of finding the true model, except the *maenet*. Apparently, *maenet* estimates a too sparse model as can be seen from the *active variables* as well. So it seems to oversimplify models resulting from DGPs constructed from correlated block matrices.

For DGP3, it is remarkable that none of the penalized methods but *enet* has a chance to find the true model for $k = 5$. However, when the dimension increases to $k = 20$, also *enet* is not able to find the true model anymore. Conform the expectations, OLS has a probability of 100% of finding the true model. Because of the non-sparse construction of DGP3 whereby none of the coefficients are zero, penalized methods are not well suited for finding the true model as they apply variable selection to a certain extent while they should not.

4.3 Lag selection performance

In this subsection I evaluate on the lag selection performance. There are two strands of lag selection methods. One strand consists of the conventional methods such as AIC and BIC. The second is based on the penalized estimation methods. I define the lag selection procedure in the latter one as follows. I divide the estimated $\mathbf{\Pi}$ in $p.max$ lags. Then I consider each of them as separate matrices. Subsequently, I start counting from $p.max$ to $p = 1$ meaning that if I start with $p.max$ then I look for matrix $\mathbf{\Pi}_{p.max}$ if all coefficients are estimated as zero. If yes, I consider the estimated amount of lags as $p.max - 1$ repeatedly until I end up at a nonzero matrix. The nonzero matrix belonging to that lag is considered as the estimated lag. Note that the regularized estimator already implies the lag selection where each variable might have a different lag where some or all lags in between might not be relevant at all. The reason that I still opt for setting whole matrices to zero by the aforementioned two-step procedure is that I assume the data to come from the same DGP with the same serial correlation over time.

The conventional lag selection criterions, such as the AIC and BIC, are overall the most accurate in selecting the right lag size. Although, for DGP2 it seems that BIC selects on average a smaller lag. For $k = 5$ the right lag is selected on average when the sample size increases to $T = 1000$ implying consistency. However, against all odds it does not happen when the dimension increases to $k = 15$.

Considering the penalized methods, it is interesting to see that they all overestimate the amount of lags for all DGPs. However, *maenet* shows the smallest amongst the overestimated lags, being closer to the true one.

It is interesting to note that for DGP2 where we take $p.max = 8$, being much higher than the true one, *maenet* is the only penalized method showing lag estimations that gets smaller as the sample size increases. However, for DGP1 and DGP3 where we take $p.max = 2$, it is remarkable that the amount of lags on average increases or remain the same as the sample size increases. It probably has difficulties with finding the exact lag. It might perform well in terms of finding a lower lag than the true one, however, when the *p.max* is so closely defined to the true lag then it does not find the exact one on average.

4.4 Forecast performance

In terms of forecast performance, it is notable that the forecast errors decrease as a function of sample size. That basically means that as T moves from 100 to 1000, the estimators have a bigger sample to learn from in order to make the predictions. In all cases it seems that the increase in sample size has a stronger impact on forecast improvement for high dimensional settings such as $k = 15$ and $k = 20$ compared to the situation of the low dimension such as $k = 5$.

Overall, *maenet* has more or less the same performance as the other penalization methods, especially for large samples $T = 1000$. For DGP2 it is remarkable that *aenetL* performs the best for the small sample $T = 100$. The *enet* performs in small samples the worst among the other penalized estimators because of the inefficient homogeneous penalty weights.

At least the *maenet* outperforms the *enet* for all DGPs with all different settings. Moreover, it is remarkable that for the high dimension $k = 20$(and $k = 15$) and small sample $T = 100$, *maenet* outperforms the other methods except for *aenetL*.

5 Empirical analysis

In this section I discuss the empirical results for applying an appropriate model to estimate the parameters and forecast. Looking at Table 4 it is visible that the *maenet* provides relatively the sparsest model with at last 93 active coefficients. The forecast performance is approximately the same as that for the *enet*, while gaining much in estimation efficiency. The *enet* estimation method has 1854 active coefficients, which is approximately 20 times as much as for the *maenet*. This is also visible in the plot of Figure 9.

For estimating the VAR model on this empirical dataset, I use as *p.max* = 5. The reason for this choice is that we deal with quarterly data and therefore I expect that the true lag amount is 4. If that is the case then the fifth lag would be penalized by the methods.

Considering the structure in the plot of Figure 9, it seems that the VAR(1) largely follows a diagonal structure. However, the structure of the remainder of the coefficient matrix looks noisy. Possibly a VAR(1) model could be sufficient for this empirical dataset. For discussing the economic interpretation of the coefficients, I use the *maenet* coefficient matrix.

Considering the four lags, it is remarkable that variables 4,5,7,8,9,10,12,14 and 15 follow a diagonal VAR(1) structure. Those are predominantly variables about *personal consumption of services and nondurable goods* (4,5), *residential and nonresidential fixed investment* (7,8,9,10) and *government receipts and spending* (14,15). All of them are negatively correlated to its previous lag, except for *government spending.*

In addition, *personal consumption* is negatively influenced by *government expenditures* of the previous quarter. This might be explained from the fact that if the government increases its expenses, inflation might rise resulting in lower personal consumption. Also, *GDP* (1) is positively influenced by *personal consumption of services* four quarters (one year) back. The economic reasoning for this is that if demand for services increases, people are getting educated over a period e.g. one year resulting in more appropriate employees and as such a higher GDP now.

Also, *the growth of government expenditures and investment* is negatively influenced by *the personal consumption* and positively influenced by *the residential fixed investment* (10) of the previous quarter. The economic reasoning for this is that as the personal consumption increases, employment opportunities increase and therefore the government gets more revenue based on more employees paying taxes over their salaries and tax revenues based on the products. Therefore, the government do not necessarily spend or invest that much to stimulate the economy anymore. For the second relation, one can note that when a country invests, there is some time and money needed before it can harvests from its investments. That is where the government might help by stimulating the investments.

Finally, *exports* are positively related to *investments* of four quarters back. The economic reasoning for this is that if a country invests in the development of products or services, it might improve its competitive position with other countries resulting in better exports over time.

Note that I do not discuss the significance of the coefficients as bootstraps are required to make statistical statements by reducing the probability of coincidence.

Table 4: Empirical results where the forecast performance and sparsity are depicted.

	MSFE	active coefs
enet	1.541	1854
aenetL	1.385	579
aenetR	1.292	817
maenet	1.457	93

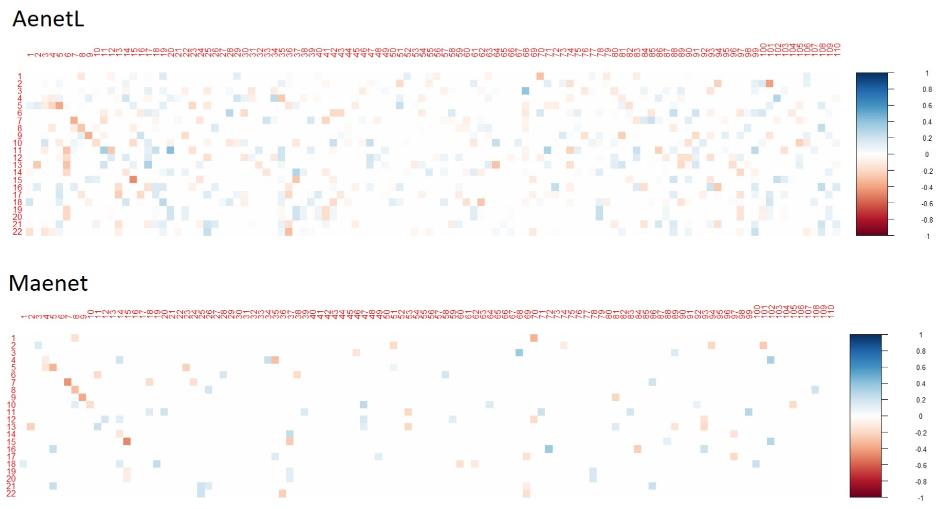


Fig. 9: Plots of the $\mathbf{\Pi}$ coefficient matrix of the empirical dataset with $k = 22$ (large dimension) and $T = 127$ (small sample). A representation of the coefficient plot estimated by the adaptive elastic net based on lasso weights and the multi-step adaptive elastic net.

6 Conclusion

The aim of this paper is to analyze if the multi-step adaptive elastic net (*maenet*) is able to provide the accurate VAR model compared to the other methods. I also consider its forecast performance and the gain in sparsity. A simulation study is conducted based on a VAR(1) sparse, VAR(4) a sparse block diagonal matrix and a VAR(1) non-sparse DGP.

The key finding in this paper is that the *maenet* method is capable of recognizing a sparse coefficient matrix being relatively close to the true coefficient matrix in a relative small sample. The simulation studies show that conditioned on a small sample, the other penalized methods do not show such a sparse model. This implies that this model provides consistent estimates at a higher rate. Because when the sample size increases, it converges faster to the true model compared to the others except for the situation where the DGP is based on a non-sparse VAR model with many redundant variables. In this case, *maenet* excludes these variables from the model and *aenetR* would be more appropriate because of its softer penalization algorithm.

This finding is incredibly useful for macroeconomic researchers. They oftentimes have to deal with a high dimensional dataset with relatively few datapoints as macroeconomic data is mostly on a quarterly basis resulting in a difficult quest for the true model. This requires a high consistent rate meaning that the algorithm should learn fast from relatively few data.

In terms of lag selection, *maenet* is in general able to be closer to the true lag compared to its single-step elastic net variants.

Much gain in efficiency can be achieved as the fraction of relevant variables is the highest with approximately the same or even better forecasts' performance as the other methods.

Altogether, this enlightens the current literature in this perspective, since there has been no study so far that focuses on the benefits by combining the multi-step adaptive elastic net with VAR model estimation.

7 Limitations and recommendations

One of the limitations of this research has been the simulation study. In the future, it would be interesting to test this method on a broader variety of simulated datasets with a much higher dimension, e.g. $k = 50$ or even $k = 100$ like the paper of Kock and Callot(2015) conducted. The reason that it could not be conducted so extensively in this thesis is due to my processing power of the computer. It has difficulties with handling such high dimensional settings.

Future research for optimizing the adaptive weights in the *maenet* could be interesting as well. One could think of selecting much more efficient input weights to reach a much faster convergence for the $\mathbf{\Pi}$ coefficient matrix estimation.

Another recommendation for future studies is to investigate the robustness of the model with respect to the selected guessed maximum lag length, say $p.max$. I initially considered this for my study of this model, however the calculation power is too demanding for my system.

Last point for future studies is to consider the consistency of this *maenet* method in the VAR context.

It would also be interesting to consider a two-step estimation method. In the first step the *maenet* estimation method can be used to find a sparser model. In the second step another method could be used for estimating the active coefficients in $\mathcal{S}_1$ found by the penalized estimation method *maenet*, in the most efficient way where the prediction performance can be improved. That is where the future research probably could focus on.

8 Appendix

8.1 Derivation of the adaptive elastic net for VAR models

This section discusses the derivation of the coordinate descent method for applying the elastic net model on VAR models. I got inspired by the paper of Nicholson, Matteson, and Bien (2014) where they did the derivation for a lasso-VAR model. I used the analogy of reasoning to derive the estimator for the elastic-net model.

First of all I denote the squared residuals of variable m of the VAR model as L_m. Suppose that I consider the residuals of a VAR model on variable g and suppose that I want to estimate $\mathbf{\Pi}_{g,h}$ which is the effect of the h-th row of the lagged variable matrix $\mathbf{Q}$ on the current value of variable g. The residuals can then be denoted as $L_{m=g,t} = (\mathbf{Y}_{m=g,t} - \sum_{n=1}^{kP} \mathbf{\Pi}_{m=g,n\neq h}\mathbf{Q}_{n\neq h,t} - \mathbf{\Pi}_{m=g,n=h}\mathbf{Q}_{n=h,t})^2$. Denote $W_{m=g,n\neq h,t}$ as the residual of the restricted VAR model for variable g, also called the partial residual in which the effect of the h-th row of the lagged matrix $\mathbf{Q}$ is excluded. Denote $P_{m,n}(\mathbf{\Pi})$ as the penalty part of the loss function for coefficient $\mathbf{\Pi}_{m,n}$. Now I can re-write the squared residual for variable g as $L_{m=g,t}$ in the following equation.

$$L_{g,t}(\mathbf{\Pi}) = (W_{g,n\neq h,t} - \mathbf{\Pi}_{g,h}\mathbf{Q}_{h,t})^2 \tag{19}$$

If I rewrite the quadratic term in equation 19, I get the following equation.

$$L_{g,t}(\mathbf{\Pi}) = W_{g,n\neq h,t}^2 + \mathbf{\Pi}_{g,h}^2\mathbf{Q}_{h,t}^2 - 2W_{g,n\neq h,t}\mathbf{\Pi}_{g,h}\mathbf{Q}_{h,t} \tag{20}$$

$$P_{g,h}(\mathbf{\Pi}|\lambda,\alpha,\omega) = \lambda(\alpha\omega_{g,h}|\mathbf{\Pi}_{g,h}| + (1-\alpha\omega_{g,h})\mathbf{\Pi}_{g,h}^2), \tag{21}$$

Now I aim to minimize function $f(\mathbf{\Pi}) = \frac{1}{2}L_{g,t}(\mathbf{\Pi}) + P_{g,h}(\mathbf{\Pi})$. Note that I added the factor of $\frac{1}{2}$ in front of $L_{g,t}$. I did this to make the derivation more convenient. In mathematical terms this does not change the location of the global minimum and therefore will not affect my outcome.

$$\delta f_{g,h}(\mathbf{\Pi}|\lambda, \alpha, \omega) = \underset{\mathbf{\Pi}_{g,h}}{min}[\frac{1}{2}\sum_{t=1}^{T} L_{g,t}(\mathbf{\Pi}) + P_{g,h}(\mathbf{\Pi}|\lambda, \alpha, \omega)] \tag{22}$$

and the subgradient of $L_{g,h}(\mathbf{\Pi}) + P_{g,h}(\mathbf{\Pi}|\lambda, \alpha, \omega)$ is denoted as follows.

$$P'_{g,h}(\mathbf{\Pi}|\lambda, \alpha, \omega) = \lambda\alpha\omega_{g,h}\psi(\mathbf{\Pi}_{g,h}) + 2\lambda(1 - \alpha\omega_{g,h})\mathbf{\Pi}_{g,h}, \tag{23}$$

$$L'_{g,t}(\mathbf{\Pi}) = 2\mathbf{\Pi}_{g,h}\mathbf{Q}^2_{h,t} - 2W_{g,n\neq h,t}\mathbf{Q}_{h,t}, (24)$$

whereby

$$\psi(\mathbf{\Pi}_{g,h}) = \begin{cases} sign(\mathbf{\Pi}_{g,h}) & \text{if } \mathbf{\Pi}_{g,h} \neq 0 \\ [-1, 1] & \text{if } \mathbf{\Pi}_{g,h} = 0 \end{cases}$$

In order to assure that I have a global minimum for $\mathbf{\Pi}_{g,h}$ in equation 22, the condition $0 \in \delta f_{g,h}$ should hold. As I take the first derivative of a quadratic loss function, this condition implies that the first derivative of $f_{g,h}$ is zero for a specific value of $\mathbf{\Pi}_{g,h}$. After some algebraic rewriting, I get the following result for the VAR adaptive elastic net estimator $\widehat{\mathbf{\Pi}}_{g,h}$.

$$\widehat{\mathbf{\Pi}}_{g,h} = \begin{cases} \frac{\sum_{t=1}^{T} W_{g,n\neq h,t}\mathbf{Q}_{h,t} - \lambda\alpha\omega_{g,h}\psi(\Pi_{g,h})}{2\lambda(1-\alpha\omega g,h) + \sum_{t=1}^{T}\mathbf{Q}^2_{h,t}} & \text{if } \widehat{\mathbf{\Pi}}_{g,h} > \lambda\alpha\omega_{g,h} \\ 0 & \text{if } |\widehat{\mathbf{\Pi}}_{g,h}| \leq \lambda\alpha\omega_{g,h} \\ \frac{\sum_{t=1}^{T} W_{g,n\neq h,t}\mathbf{Q}_{h,t} + \lambda\alpha\omega_{g,h}\psi(\Pi_{g,h})}{2\lambda(1-\alpha\omega g,h) + \sum_{t=1}^{T}\mathbf{Q}^2_{h,t}} & \text{if } \widehat{\mathbf{\Pi}}_{g,h} < \lambda\alpha\omega_{g,h} \end{cases}$$

(25)

I can translate the nonlinear estimator $\mathbf{\Pi}_{g,h}$ in equation 8.1 to the soft threshold $\mathcal{ST}$ operator, where $\mathcal{ST}(a,b) = sign(a) * max(|a| - b)$. For the estimator it boils down to the following equation.

$$\widehat{\mathbf{\Pi}}_{g,h} = \frac{\mathcal{ST}(W_{g,n \neq h,t}\mathbf{Q}_{h,t}, \lambda\alpha\omega_{g,h})}{2\lambda(1-\alpha\omega_{g,h}) + \sum_{t=1}^{T}\mathbf{Q}_{h,t}^2} \tag{26}$$

8.2 Ridge estimation within VAR framework

For the adaptive elastic net estimation, there are two possibilities for the weighting matrices. On the one hand I can use the $\mathbf{\Pi}$ matrix being estimated from the lasso method on the VAR model. That is when $\alpha = 1$ in equation 5. I use the coordinate descent algorithm to estimate this $\mathbf{\Pi}$ matrix. On the other hand by following the paper of Kock and Callot (2015), I can make use of the weighting matrix being estimated by the ridge regression technique where predictors are to a certain extent decorrelated. This is the case when $\alpha = 0$. The estimation is as follows. As this is an quadratic optimization, I decide to leave the proof behind as this derivation is quite similar as for the normal OLS.

$$\widehat{\mathbf{\Pi}} = \mathbf{YQ}'(\mathbf{Q}'\mathbf{Q} + \lambda\mathbf{I})^{-1} \tag{27}$$

9 Program codes in R

9.1 Functions

```r
############# FUNCTIONS ##############

# LOAD PACKAGES

library(tsDyn) # for sorting data
library(vars) # for estimating var model
library(rlist)
library(BigVAR) # for simulating VAR data
library(tidyverse)
library(lattice)
library(corrplot)
library(abind); library(magic)  # for block matrix

# -------------------- #

## Soft-thresholding algorithm for elastic net in VAR Models

soft_threshold <- function(z,gamma){
#coef is a scalar coefficient such as B_ij
#gamma is the hyperparameter, which is calculated as lambda*alpha

  if(z < -gamma) {z = z + gamma}          # shrink coef
  else if(z > gamma) {z = z - gamma}      # shrink coef
  else {z = 0}                            # eliminate coef

  return(z)}

# ------------------- #

# The coordinate descent algorithm applicable for standardized VAR
    data
## Got my inspiration from LASSO-VAR(p) algorithm of Nicholson (2014)

adap_enet_VAR <- function(input.data, p.max, initial = c("normal","
    adaptive"),
                            PI_ini, PI_weight,
                            weight_input = c("lasso", "ridge", "none"),
                                 M, R, eps,
                            alpha.min, alpha.max, lambda.grid, step.
                                size){


  # Setting default values for arguments

  if (missing(p.max)) {p.max = 8}
  if (missing(M)) {M = 5} #is needed for the length of the lambda
      grid
  if (missing(eps)) {eps = 10^-4}
  if (missing(alpha.min)) {alpha.min <- 0.75} #since we want to
      attach at least 75 percent to the lasso part for variable
      selection
  if (missing(alpha.max)) {alpha.max <- 0.95} # since we don't want
      that the elastic net solely depends on the lasso part.
  if (missing(step.size)) {step.size <- 0.1}
  if (missing(R)) {R <- 100} # is needed for the depth of the lambda.
      grid
  if (missing(initial)) {initial = "normal"}
  if (missing(weight_input)) {weight_input = "none"}



  # Initialize the weightings
  weight = 1
  delta <- 1 # chosen conform paper of Xiao (2015)


  ## Initialization of the data (with aid of vars package)

  var_check    <- VAR(input.data,p=p.max,type="none")
  datmat_check <- var_check$datamat
  Y_tilde      <- t(scale(datmat_check[,(1:ncol(input.data))]))
  Q_tilde      <- t(scale(datmat_check[,-(1:ncol(input.data))]))
  YQ_tilde     <- rbind(Y_tilde,Q_tilde)
  QY           <- Q_tilde %*% t(Y_tilde) #matrix product of Z_tilde
      and t(Y_tilde)
```

```
k              <- nrow(Y_tilde)
kp.max         <- nrow(Q_tilde); T <- ncol(Y_tilde)


if (missing(PI_ini)) {PI_ini <- matrix(rep(1,(k*kp.max)), nrow=k,
    ncol=kp.max)}
if (missing(PI_weight) && weight_input == "lasso")
    {PI_weight <- PI_ini
     PI_weight <- adap_enet_VAR2(input.data, p.max = p.max,
                                initial = initial,
                                alpha.min = 1, alpha.max = 1,
                                R = R)$PI_optimal # adaptive with
                                    lasso input
    }

if (missing(PI_weight) && weight_input == "ridge")
    {PI_weight <- PI_ini
    PI_weight <- adap_enet_VAR2(input.data, p.max = p.max,
                                initial = initial,
                                alpha.min = 0, alpha.max = 0,
                                R = R)$PI_optimal # adaptive with
                                    ridge input
    }


# penalty grid
alpha.grid <- sort(seq(alpha.min,alpha.max,step.size), decreasing =
     TRUE)
lambda.grid <- exp(seq(log(max(QY)),log(max(QY))/R,length.out = M))

# looping through sequence of penalty grids, where lambda is
    conditioned on alpha
PI_old  <- PI_new <- PI_ini
VAR_array    <- list(); counter = 1  # storage of PI_new and
    referring residual

ptm <- proc.time() #starts stopwatch


for (a in 1:length(alpha.grid)){
  alpha = alpha.grid[a]

  for (m in 1:M){        # looping through M iterations for lambda
      conditioned on specific value for alpha
    lambda <- lambda.grid[m]

    #### Run coordinate descent algorithm if alpha!=0
    Y_tilde.used = Y_tilde
    Q_tilde.used = Q_tilde

    #### Run coordinate descent algorithm when alpha != 0
    if(alpha != 0){PI_new <- coordesc_enet(PI_old = PI_old, Y_tilde
         = Y_tilde.used, Q_tilde = Q_tilde.used,
                              initial = initial, PI_weight = PI_
                                  weight, lambda = lambda, alpha =
                                  alpha, delta = delta)}

    #### Calculate analytically the result when alpha == 0; ridge
        regression
    if (alpha == 0) {PI_new = Y_tilde %*% t(Q_tilde) %*% solve((Q_
        tilde %*% t(Q_tilde)) + lambda*diag(1,nrow = kp.max))} #
        ridge solution


    error_new <- Y_tilde - PI_new %*% Q_tilde
    PI_old <- PI_new


    #store in results in a big list
    VAR_array[[counter]] <- list(PI.est = round(PI_new,2), lambda.
        est = round(lambda,2), alpha.est = alpha, delta = delta,
                                  residuals2 = round(sum(t(colSums(
                                      apply(error_new, 1, function(x
                                      ) x^2)))),2),
                                  iteration = counter)

    counter = counter + 1

  } # end M (lambda) loop

```

```
  } # end alpha loop

  # Create report based on optimization algorithm

  VAR.lag.df = as.data.frame(do.call(rbind, lapply(VAR_array, unlist)
      )) #convert the big list into a dataframe
  VAR.lag.df = unique(VAR.lag.df) #remove duplicates
  VAR.lag.df = dplyr::arrange(VAR.lag.df,residuals2,desc(alpha.est))

  # Return PI matrix of the lowest SSR
  PI_optimal         = matrix(as.numeric(unlist(VAR.lag.df[1,1:(k^2*p
      .max)])), nrow = k, ncol = kp.max)
  lambda_optimal     = as.numeric(unlist(VAR.lag.df[1,]$lambda.est))
  alpha_optimal      = as.numeric(unlist(VAR.lag.df[1,]$alpha.est))

  proc.time() - ptm # end of stopwatch

return(list(PI_optimal = PI_optimal,
            alpha_optimal = alpha_optimal, lambda_optimal = lambda_
                optimal))}

# ---------------------- #

# Mulit-step adaptive elastic net for VAR models algorithm

multi_step_enet_VAR <- function(input.data, R, p.max, alpha.min,
    alpha.max, eps){

  if (missing (eps)) {eps <- 10^-4}

  PI_weight <- PI_maenet_old <- PI_maenet_ini <- adap_enet_VAR(input.
      data = input.data, R = R, p.max = p.max, alpha.min = alpha.min,
       alpha.max = alpha.max, initial = "adaptive")$PI_optimal
  max.difference = 1
  counter.maenet = 1

  while (max.difference > eps){


    result.algorithm <- adap_enet_VAR(input.data = input.data,
                                       R = R, PI_weight = PI_weight,
                                       p.max = p.max, alpha.min = alpha.min, alpha.max = alpha.max,
                                       initial = "adaptive")

    PI_maenet_new <- result.algorithm$PI_optimal
    max.difference = max((abs(as.vector(PI_maenet_old) - as.vector(PI_maenet_new)))/(1+abs(as.vector(PI_maenet_old))))
    PI_maenet_old = PI_maenet_new
    PI_weight = PI_maenet_old
    counter.maenet = counter.maenet + 1

  }

  alpha_optimal      = result.algorithm$alpha_optimal
  lambda_optimal     = result.algorithm$lambda_optimal


  return(list(PI_optimal_maenet = PI_maenet_new, alpha_optimal = alpha_optimal,
              lambda_optimal = lambda_optimal))}


# ----------------------- #

## Coordinate descent algorithm for elastic net on VAR models

coordesc_enet <- function(PI_old, Y_tilde, Q_tilde, initial = c("normal", "adaptive"), PI_weight, lambda, alpha, delta){

# COORDINATE DESCENT UPDATE: update each coordinate of the PI matrix given lag, lambda, alpha, row i and column j

k <- nrow(Y_tilde);    kp.max <-  nrow(Q_tilde);    T <- ncol(Y_tilde)

weight = 1

```

```r
if (missing(PI_old)) {PI_old <- matrix(rep(1,(k*kp.max)), nrow=k, ncol=kp.max)}
PI_new <- PI_old

if (missing(PI_weight)) {PI_weight <- PI_old}

gamma <- lambda*alpha #hyperparameter
correction.enet <- 1+((lambda*(1-alpha))/(T*k)) # to correct for double shrinkage/bias correcting

R.vec   <- rep(NA,length.out=T) #Initiate a Tx1 vector for partial residuals
max.difference <- 1
eps <- 10^-4

while (max.difference > eps){

  for(i in 1:k){
    for (j in 1:kp.max){

      PI.Q_ex_j     <- PI_old[i,-j] %*% Q_tilde[-j,]
      R.vec         <- t(Y_tilde[i,] - PI.Q_ex_j) # vector of restricted (model) residuals
      Q_j_squared   <- t(Q_tilde[j,]) %*% (Q_tilde[j,])

      PI_ij_hat     <- (Q_tilde[j,] %*% R.vec)  # plot of restricted (model) residuals on j th row of original Z mat
      PI_ij_hat = bound(PI_ij_hat) # such that there exists no Inf or -Inf values

      ### weights for normal part
      if (initial == "normal") {weight = 1}

      if (initial == "adaptive") {weight = abs(PI_weight[i,j]*gamma)^(-delta)}

      if (is.finite(PI_weight[i,j]))
         {PI_new[i,j]    = correction.enet*((soft_threshold(PI_ij_hat, (gamma*weight)) / ((2*lambda*(1-(alpha*weight)))+Q_j_squared)))}
      if(!is.finite(PI_weight[i,j]))
         {PI_new[i,j] = 0}

      max.difference = max((abs(as.vector(PI_old) - as.vector(PI_new)
          ))/(1+abs(as.vector(PI_old))))

      PI_old[i,j]    = PI_new[i,j] #update the old PI matrix


    } #end j loop

    if (abs(sum(PI_new[i,])) > 1) {PI_new[i,] = normalize(PI_new[i,])
        }; PI_old = PI_new

  } #end k loop, thus whole PI matrix is updated

} #end while loop

return(PI_new)}

# ------------------------ #
#
# ------------------------ #

## Cross-validation for VAR models with Elastic-Net
# By means of this time-dependent cross validation we want to
    optimize the hyperparameters lambda and alpha of this model

enet_VAR_cval <- function(input.data, p.max,
                          initial = c("normal","adaptive"), weight_
                              input=c("lasso", "ridge"),
                          alpha.min, alpha.max,
                          R){


  # Create time intervals
  T_full  <- length(input.data[,1]) - p.max
  T1      <- floor(T_full/3)
  T2      <- floor(T_full/3*2)

```

```r
length.window = T1
length.eval   = T2 - T1
length.fcast  = T_full - T2


# Load and prepare dataset
var_check    <- VAR(input.data,p=p.max,type="none")
datmat_check <- var_check$datamat
Y_tilde_full <- t(scale(datmat_check[,(1:ncol(input.data))]))
Q_tilde_full <- t(scale(datmat_check[,-(1:ncol(input.data))]))

MSFE_matrix = matrix(NA, nrow = length.eval, ncol = 3)
colnames(MSFE_matrix) <- c("MSFE", "alpha", "lambda")


# train the parameters in block T1+1 to T2

for (t in 1:length.eval)
{

  algo_est_init             <- adap_enet_VAR(input.data = input.
      data[t:(T1+t-1+4),],
                                              initial = initial,
                                                  weight_input =
                                                  weight_input,
                                              R = R, p.max = p.max,
                                              alpha.min = alpha.min,
                                                   alpha.max = alpha
                                                  .max)
  eval_position             <- T1 + t

  Y_tilde_train             <- algo_est_init$Y_tilde
  Q_tilde_train             <- algo_est_init$Q_tilde

  PI_train                  <- algo_est_init$PI_optimal

  alpha_train               <- algo_est_init$alpha_optimal
  lambda_train              <- algo_est_init$lambda_optimal

```

```
    Y_train_fcast             <- PI_train %*% Q_tilde_full[,eval_
        position]
    fcast_train_err           <- Y_tilde_full[,eval_position] - Y_
        train_fcast

    MSFE_matrix[t,1]          <- sum(fcast_train_err^2)
    MSFE_matrix[t,2]          <- alpha_train
    MSFE_matrix[t,3]          <- lambda_train

  }

    for (parameters in 1:length.eval) {

      alpha_eval    = MSFE_matrix[parameters,2]
      lambda_eval   = MSFE_matrix[parameters,3]

      for (observation in 1:length.fcast) {

        fcast_position = T2 + observation
        Y_tilde_eval   = matrix(Y_tilde_full[,fcast_position], ncol
            =1)
        Q_tilde_eval   = matrix(Q_tilde_full[,fcast_position], ncol
            =1)

        begin.window = fcast_position - length.window
        end.window   = fcast_position

        if (alpha.min != 0 && alpha.max != 0)
        {PI_eval       = coordesc_enet(Y_tilde = Y_tilde_full[,begin.
            window:end.window],
                                       Q_tilde = Q_tilde_full[,begin.
                                           window:end.window],
                                       initial = initial,
                                       alpha   = alpha_eval,
                                       lambda  = lambda_eval,
                                       delta = 1)
        }

        if (alpha.min == 0 && alpha.max == 0)
        {
          PI_eval <- adap_enet_VAR2(input.data = input.data[begin.
              window:(end.window+p.max),], p.max = p.max,
                                    initial = initial,
                                    alpha.min = 0, alpha.max = 0,
                                    R = R)$PI_optimal
        }



        eval_error = Y_tilde_eval - PI_eval %*% Q_tilde_eval
        MSFE_eval_matrix[parameters,observation] = sum(eval_error^2)
      }

    }

    MSFE_eval_avg = cbind(rowMeans(MSFE_eval_matrix), MSFE_matrix
        [,2], MSFE_matrix[,3])
    colnames(MSFE_eval_avg) = c("MSFE", "alpha", "lambda")

    # find parameters with minimum MSFE
    optimal_parameters = MSFE_eval_avg[which.min(MSFE_eval_avg[,1])
        ,2:3]



  return(list(MSFE_matrix, MSFE_eval_avg, optimal_parameters))}

# ------------------------ #

bound <- function(x){

  # this function is meant for censoring the weights such that it can
      't take extreme values.
  if(!is.finite(x)) {x = 0}
  if(is.finite(x) && x == -Inf) {x = -10^100}
  if(is.finite(x) && x == Inf) {x = 10^100}

return(x)}


```

```
# ------------------------ #

## Forecast function

forecast <- function(input.data,
                     reg_type = c("ols", "penalized"),
                     initial = c("adaptive", "normal"),
                     weight_input = c("ridge", "lasso","none"), p.max
                         , R,
                     alpha, lambda,
                     iter) {


  # Create time intervals
  T_full  <- length(input.data[,1]) - p.max
  T1      <- floor(T_full/3)
  T2      <- floor(T_full/3*2)

  length.window = T2
  length.fcast  = T_full - length.window

  # Initialize dataset
  var_check    <- VAR(input.data,p=p.max,type="none")
  datmat_check <- var_check$datamat
  Y_tilde_full <- t(scale(datmat_check[,(1:ncol(input.data))]))
  Q_tilde_full <- t(scale(datmat_check[,-(1:ncol(input.data))]))


  SFE.error <- matrix(NA, nrow = length.fcast, ncol=nrow(Y_tilde_full
      ))

  for (observation in 1:length.fcast) {

    fcast_position  = T2 + observation
    Y_tilde_fcast   = matrix(Y_tilde_full[,fcast_position], ncol=1)
    Q_tilde_fcast   = matrix(Q_tilde_full[,fcast_position], ncol=1)

    begin.window = 1
    end.window   = length(input.data[,1])-length.fcast

```

```r
    input.eval = input.data[begin.window:end.window,]

    if(reg_type!="ols" || reg_type!="ols_oracle"){
     # PI_eval <- adap_enet_VAR(input.data = input.eval, p.max = p.
         max,
     #                         initial = initial, weight_input =
         weight_input,
     #                         alpha.min = alpha, alpha.max = alpha,
     #                         R = R)$PI_optimal}

     PI_eval = PI_input}

    if(reg_type=="ols"){
     PI_eval <- Bcoef(VAR(input.data = input.eval,p=p.max,type="none"
         ))

    }

    if(reg_type=="ols_oracle"){
      PI_eval = PI_org

    }

    fc_error = Y_tilde_fcast - PI_eval %*% Q_tilde_fcast
    SFE.error[observation,] = fc_error^2
  }

  MSFE <- mean(SFE.error)

  return(MSFE)}
# ------------------------- #

# normalize vector

normalize <- function(vector) {vector / sqrt(sum(vector^2))}

# ------------------------- #

compare_true_est_vars <- function(PI_org, PI_est)

```

```
{

  k = nrow(PI_org)
  p = ncol(PI_org)/k
  p.max = ncol(PI_est)/k
  kp.max = k*p.max

  PI_full_org <- matrix(0,nrow=k,ncol=kp.max)
  PI_full_org[,1:ncol(PI_org)] = PI_org


  PI_true_vec      <- as.vector(PI_full_org)
  PI_est_vec       <- as.vector(PI_est)

  true_active_index <- which(PI_true_vec!=0)
  est_active_index  <- which(PI_est_vec!=0)
  est_active        <- length(est_active_index)

  match_count       <- est_active_index %in% true_active_index

  frac_relevant_vars <- length(which(match_count==TRUE))/length(match
      _count)

  true_included     <- true_active_index %in% est_active_index
  true_included     = all(true_included == TRUE)

  diff_est_true    <- sum((PI_est_vec-PI_true_vec)^2)


  # correct sparsity

  zeroes_model = which(PI_est == 0)
  zeroes_DGP   = which(PI_org == 0)

  corr_spars   = length(which(zeroes_model != zeroes_DGP))


  summary_table <- matrix(NA, ncol=5)
  summary_table[,1] = est_active
  summary_table[,2] = frac_relevant_vars
  summary_table[,3] = true_included
  summary_table[,4] = diff_est_true
  summary_table[,5] = corr_spars

  colnames(summary_table) = c("est_act_vars", "frac_rel_vars", "true_
      included", "SE_PI", "corr_spars")

  return(summary_table)}

# ----------------------- #


# The coordinate descent algorithm applicable for standardized VAR
    data
## Got my inspiration from LASSO-VAR(p) algorithm of Nicholson (2014)

adap_enet_VAR2 <- function(input.data, p.max, initial = c("normal","
    adaptive"),
                          PI_ini, PI_weight,
                          weight_input = c("lasso", "ridge", "none"),
                               M, R, eps,
                          alpha.min, alpha.max, lambda.grid, step.
                              size){


  # Setting default values for arguments

  if (missing(p.max)) {p.max = 8}
  if (missing(M)) {M = 5} #is needed for the length of the lambda
      grid
  if (missing(eps)) {eps = 10^-4}
  if (missing(alpha.min)) {alpha.min <- 0.75} #since we want to
      attach at least 75 percent to the lasso part for variable
      selection
  if (missing(alpha.max)) {alpha.max <- 0.95} # since we don't want
      that the elastic net solely depends on the lasso part.
  if (missing(step.size)) {step.size <- 0.1}
  if (missing(R)) {R <- 100} # is needed for the depth of the lambda.
      grid
  if (missing(initial)) {initial = "normal"}
```

```r
if (missing(weight_input)) {weight_input = "none"}



# Initialize the weightings
weight = 1
delta <- 1 # chosen conform paper of Xiao (2015)


## Initialization of the data (with aid of vars package)

var_check    <- VAR(input.data,p=p.max,type="none")
datmat_check <- var_check$datamat
Y_tilde      <- t(scale(datmat_check[,(1:ncol(input.data))]))
Q_tilde      <- t(scale(datmat_check[,-(1:ncol(input.data))]))
YQ_tilde     <- rbind(Y_tilde,Q_tilde)
QY           <- Q_tilde %*% t(Y_tilde) #matrix product of Z_tilde
    and t(Y_tilde)
k            <- nrow(Y_tilde)
kp.max       <- nrow(Q_tilde); T <- ncol(Y_tilde)


if (missing(PI_ini)) {PI_ini <- matrix(rep(1,(k*kp.max)), nrow=k,
    ncol=kp.max)}
if (missing(PI_weight) && weight_input == "lasso")
{PI_weight <- PI_ini
PI_weight <- adap_enet_VAR2(input.data, p.max = p.max,
                            initial = initial,
                            alpha.min = 1, alpha.max = 1,
                            R = R)$PI_optimal # adaptive with lasso
                                 input
}

if (missing(PI_weight) && weight_input == "ridge")
{PI_weight <- PI_ini
PI_weight <- adap_enet_VAR2(input.data, p.max = p.max,
                            initial = initial,
                            alpha.min = 0, alpha.max = 0,
                            R = R)$PI_optimal # adaptive with ridge
                                 input
```

```
}



# penalty grid
alpha.grid <- sort(seq(alpha.min,alpha.max,step.size), decreasing = TRUE)
lambda.grid <- exp(seq(log(max(QY)),log(max(QY))/R,length.out = M))

# looping through sequence of penalty grids, where lambda is conditioned on alpha
PI_old  <- PI_new <- PI_ini
VAR_array    <- list(); counter = 1  # storage of PI_new and referring residual

ptm <- proc.time() #starts stopwatch


for (a in 1:length(alpha.grid)){
  alpha = alpha.grid[a]

  for (m in 1:M){        # looping through M iterations for lambda conditioned on specific value for alpha
    lambda <- lambda.grid[m]

    #### Run coordinate descent algorithm if alpha!=0
    Y_tilde.used = Y_tilde
    Q_tilde.used = Q_tilde

    #### Run coordinate descent algorithm when alpha != 0
    if(alpha != 0){PI_new <- coordesc_enet(PI_old = PI_old, Y_tilde = Y_tilde.used, Q_tilde = Q_tilde.used,
                                                   initial = initial, PI_weight = PI_weight, lambda = lambda, alpha = alpha, delta = delta)}

    #### Calculate analytically the result when alpha == 0; ridge regression
    if (alpha == 0) {PI_new = Y_tilde %*% t(Q_tilde) %*% solve((Q_tilde %*% t(Q_tilde)) + lambda*diag(1,nrow = kp.max))} # ridge solution


    error_new <- Y_tilde - PI_new %*% Q_tilde
    PI_old <- PI_new


    #store in results in a big list
    VAR_array[[counter]] <- list(PI.est = round(PI_new,2), lambda.est = round(lambda,2), alpha.est = alpha, delta = delta,
                                  residuals2 = round(sum(t(colSums(apply(error_new, 1, function(x) x^2)))),2),
                                  iteration = counter)

    counter = counter + 1

  } # end M (lambda) loop

} # end alpha loop

# Create report based on optimization algorithm

VAR.lag.df = as.data.frame(do.call(rbind, lapply(VAR_array, unlist))) #convert the big list into a dataframe
VAR.lag.df = unique(VAR.lag.df) #remove duplicates
VAR.lag.df = dplyr::arrange(VAR.lag.df,residuals2,desc(alpha.est))

# Return PI matrix of the lowest SSR
PI_optimal          = matrix(as.numeric(unlist(VAR.lag.df[1,1:(k^2*p.max)])), nrow = k, ncol = kp.max)
lambda_optimal      = as.numeric(unlist(VAR.lag.df[1,]$lambda.est))
alpha_optimal       = as.numeric(unlist(VAR.lag.df[1,]$alpha.est))

proc.time() - ptm # end of stopwatch

return(list(PI_optimal = PI_optimal,
                alpha_optimal = alpha_optimal, lambda_optimal = lambda_
                    optimal))}

############# END OF FUNCTIONS ################
```

./Codes_New/Nasser_functions_v2.R

Bibliography

Basu, S., G. Michailidis, et al. (2015). “Regularized estimation in sparse high-dimensional time series models”. *The Annals of Statistics* 43.4, pp. 1535–1567.

Davidson, J. (2009). “When is a time series I (0)”. *The Methodology and Practice of Econometrics: A Festschrift in Honour of David F. Hendry*, pp. 322–342.

Efron, B., T. Hastie, I. Johnstone, R. Tibshirani, et al. (2004). “Least angle regression”. *The Annals of Statistics* 32.2, pp. 407–499.

Engle, R. and C. Granger (1991). *Long-run economic relationships: Readings in cointegration.* Oxford University Press.

Friedman, J., T. Hastie, H. Höfling, R. Tibshirani, et al. (2007). “Pathwise coordinate optimization”. *The Annals of Applied Statistics* 1.2, pp. 302–332.

Friedman, J., T. Hastie, and R. Tibshirani (2010). “Regularization paths for generalized linear models via coordinate descent”. *Journal of statistical software* 33.1, p. 1.

Furman, Y. (2014). “VAR estimation with the adaptive elastic net”. *Available at SSRN 2456510.*

Kock, A. B. and L. Callot (2015). “Oracle inequalities for high dimensional vector autoregressions”. *Journal of Econometrics* 186.2, pp. 325–344.

Kuha, J. (2004). "AIC and BIC: Comparisons of assumptions and performance". *Sociological Methods & Research* 33.2, pp. 188–229.

Luetkepohl, H. (2009). *Econometric analysis with vector autoregressive models*. Wiley Online Library.

Nicholson, W. B., D. S. Matteson, and J. Bien (2014). "Structured regularization for large vector autoregressions". *Cornell University*.

Royston, P. (2004). "Multiple imputation of missing values". *The Stata Journal* 4.3, pp. 227–241.

Schnücker, A. (2019). *Penalized Estimation of Panel Vector Autoregressive Models*. Tech. rep.

Sharma, S. C. (1987). "The effects of autocorrelation among errors on the consistency property of OLS estimator". *Journal of Statistical Computation and Simulation* 28.1, pp. 43–52.

Sims, C. A. (1980). "Macroeconomics and reality". *Econometrica: journal of the Econometric Society*, pp. 1–48.

Song, S. and P. J. Bickel (2011). "Large vector auto regressions". *arXiv preprint arXiv:1106.3915*.

Stock, J. H. and M. W. Watson (2012). *Disentangling the Channels of the 2007-2009 Recession*. Tech. rep. National Bureau of Economic Research.

Tabassum, M. N. and E. Ollila (2017). "Pathwise least angle regression and a significance test for the elastic net". In: *2017 25th European Signal Processing Conference (EUSIPCO)*. IEEE, pp. 1309–1313.

Tibshirani, R. (1996). "Regression shrinkage and selection via the lasso". *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1, pp. 267–288.

Tseng, P. (2001). "Convergence of a block coordinate descent method for nondifferentiable minimization". *Journal of optimization theory and applications* 109.3, pp. 475–494.

Wu, T. T., K. Lange, et al. (2008). “Coordinate descent algorithms for lasso penalized regression”. *The Annals of Applied Statistics* 2.1, pp. 224–244.

Xiao, N. and Q.-S. Xu (2015). “Multi-step adaptive elastic-net: reducing false positives in high-dimensional variable selection”. *Journal of Statistical Computation and Simulation* 85.18, pp. 3755–3765.

Zou, H. (2006). “The adaptive lasso and its oracle properties”. *Journal of the American statistical association* 101.476, pp. 1418–1429.

Zou, H. and T. Hastie (2005). “Regularization and variable selection via the elastic net”. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 67.2, pp. 301–320.

Zou, H. and H. H. Zhang (2009). “On the adaptive elastic-net with a diverging number of parameters”. *Annals of statistics* 37.4, p. 1733.