Master's Thesis

Achieving smoothness in sparse spatio-temporal autoregressive modelling via the graph-guided fused LASSO

July 20, 2023

Student:
Jacco Broere
2749073

Supervisor:
dr. E.J.J. Wijler

Course code:
E_EORM_THSTR

Abstract

This study introduces a series of novel techniques to estimate spatial autoregressive models, combining the graph-guided fused LASSO methodology and penalized Yule-Walker estimation techniques. This approach, like the SPatial LAsso-type SHrinkage (SPLASH) estimator, derives spatial interactions directly from data, eliminating the need for predefined spatial interaction matrices. However, unlike SPLASH, we develop methods rooted in the graph-guided fused LASSO framework that enforce smooth parameter estimates along the edges of an underlying graph. This study offers a comprehensive overview of the spatio-temporal autoregressive setting, generalized Yule-Walker equations and various LASSO variants. It provides a detailed translation of this theory into a set of novel and specialized spatio-temporal models. These models are subsequently evaluated through a series of simulation studies with varying dimensionality and underlying interactions. The results reveal enhanced forecasting performance in several spatial settings and promising estimation speed for a subset of the developed methods. This presents intriguing opportunities for further research and optimization of these methods.

Keywords: Spatial Autoregressive Models, Spatio-Temporal Models, Penalized Yule-Walker Estimation, SPLASH, Generalized LASSO, Graph-guided Fused LASSO, Simulation Study.

Contents

Notation

The following list provides an overview of the mathematical notation and symbols used throughout this work.

$:=$ Defined to be equal to.

$\triangleq$ Equal by definition.

$\wedge$, $\vee$ The logical conjunction and disjunction operators, respectively.

$\exists z : A$ There exists a z such that A is true.

$z \in S$ The element of operator, i.e., z is an element of set S.

$|S|$ The cardinality of a set S, i.e., the number of elements in S.

S^2 The cartesian product of a set S with itself, i.e., $S^2 := S \times S$.

$\bigcup_{i=1}^{n} S_i$ The union of the collection of sets $\{S_1, \ldots, S_n\}$.

$\mathbb{E}(\cdot)$, $\mathrm{Var}(\cdot)$ The expectation and variance operator, respectively.

x_i The ith entry of vector $\boldsymbol{x} := (x_i)_{i=1}^{k} \in \mathbb{R}^k$.

m_{ij} The entry at the ith row and jth column of matrix $\boldsymbol{M} := \big(m_{ij}\big)_{i,j=1}^{n,k} \in \mathbb{R}^{n \times k}$.

$\boldsymbol{x}'$, $\boldsymbol{M}'$ The transpose of vector $\boldsymbol{x}$ and matrix $\boldsymbol{M}$, respectively.

$\|\boldsymbol{x}\|_p$ The L_p-norm of a vector $\boldsymbol{x} \in \mathbb{R}^k$, i.e., $\|\boldsymbol{x}\|_p := \left(\sum_{i=1}^{k} |x_i|^p\right)^{1/p}$.

$\|\boldsymbol{M}\|_p$ The L_p-norm of matrix $\boldsymbol{M} \in \mathbb{R}^{n \times k}$, i.e., $\|\boldsymbol{M}\|_p := \sup\left\{ \|\boldsymbol{M}\boldsymbol{x}\|_p \,/\, \|\boldsymbol{x}\|_p \,\middle|\, \boldsymbol{x} \in \mathbb{R}^k \right\}$.

$\|\boldsymbol{M}\|_1$ A special case of the L_p-norm, i.e., $\|\boldsymbol{M}\|_1 := \max_{1 \le j \le k} \sum_{i=1}^{n} |m_{ij}|$.

$\|\boldsymbol{M}\|_\infty$ A special case of the L_p-norm, i.e., $\|\boldsymbol{M}\|_\infty := \max_{1 \le i \le n} \sum_{j=1}^{k} |m_{ij}|$.

$\lambda_{\max}(\cdot)$ The largest eigenvalue operator.

$\|\boldsymbol{M}\|_2$ A special case of the L_p-norm, the spectral norm, i.e., $\|\boldsymbol{M}\|_2 := \sqrt{\lambda_{\max}(\boldsymbol{M'M})}$.

$\mathscr{B}_h(\boldsymbol{M})$ The h-banded counterpart of matrix $\boldsymbol{M}$, i.e., $\mathscr{B}_h(\boldsymbol{M}) = \left(m_{ij} \cdot \mathbb{1}_{\{|i-j| \le h\}}\right)_{i,j=1}^{n,k}$.

$\mathscr{E}(\boldsymbol{M})$ The set of entries of matrix $\boldsymbol{M}$, i.e., $\mathscr{E}(\boldsymbol{M}) = \left\{ m_{ij} \,\middle|\, m_{ij} \text{ is an entry of } \boldsymbol{M} \right\}$.

$\mathscr{D}_{\boldsymbol{M}}(k)$ The set of entries on the same diagonal, i.e., $\mathscr{D}_{\boldsymbol{M}}(k) = \left\{ m_{ij} \in \mathscr{E}(\boldsymbol{M}) \,\middle|\, i - j = k \right\}$.

1 Introduction

Spatio-temporal models have emerged as a powerful tool for modelling and analysing data exhibiting both spatial and temporal dependence. Their ability to capture intricate patterns in diverse fields such as climate modelling, epidemiology, and criminology, among many others, attests to their effectiveness and versatility (Aswi et al., 2019; Kakamu et al., 2008; Thorson, 2019).

The relevance of spatio-temporal models has recently been amplified in the face of contemporary global challenges. Climate change, one of the most pressing issues of our time, encompasses an extensive range of complex spatio-temporal problems (Vatsavai et al., 2012). To develop meaningful insights and predictions, it is critical to understand the interconnections between regions, and how climate change evolves over time. Spatio-temporal models provide a robust set of tools to capture these complexities, assisting researchers and decision-makers in tackling this profound issue. Similarly, the COVID-19 epidemic has highlighted the value of spatio-temporal modelling in health scenarios. The transmission dynamics of the virus exhibit patterns that vary both over location and time (Aswi et al., 2019; Hafner, 2020; Yu et al., 2021). Consequently, spatio-temporal models have proven instrumental in monitoring its progress, identifying high-risk areas, and orchestrating effective resource distribution for prevention and care.

On the one hand, we have the literature surrounding spatial modelling using predefined spatial weight matrices, such as in spatial autoregressive models (SAR) (Dou et al., 2016; Kelejian & Prucha, 1998; Qu & Lee, 2015). Building on this, there is a strand of literature concerning spatial dynamic panel data models (SDPD), which incorporate temporal dependence by extending SAR with lagged observations (Lee & Yu, 2010; Qu et al., 2017). In both SAR and SDPD models, the estimation and inference often focus on scalar parameters that multiply the predefined spatial weight matrices. These approaches manage to solve important problems inherent to spatio-temporal modelling, such as endogeneity and the large number of freely varying parameters. However, they also severely limit the model to find intricacies and relationships of the specific data at hand because the spatial weight matrix is completely based on the a priori expectation of the spatial interactions.

On the other hand, we have a more recent strand of literature exploring the possibilities of inferring the spatial relationships and intricacies from the data itself (Gao et al., 2019; Ma et al., 2023; Reuvers & Wijler, 2021). These approaches alleviate problems of endogeneity and identifiability by imposing other restrictions, such as banded coefficient matrices and employing generalized Yule-Walker equations in the estimation procedure. Reuvers and Wijler (2021) further exploit the spatial nature of the problem by developing the SPatial LAsso-type SHrinkage (SPLASH) estimator. This model exploits diagonally structured sparsity patterns that appear naturally in many spatial settings. To this end, they employ the sparse group LASSO to stimulate the model towards sparse diagonals while deriving all spatial interactions from the data. They report improved forecasting and estimation performance in simulated and empirical settings.

This thesis develops novel spatio-temporal estimation methods that are most closely related to the SPLASH estimator by Reuvers and Wijler (2021). Our methods also revolve

around the same principles of the generalized Yule-Walker equations, also seen in Gao et al. (2019) and Ma et al. (2023). However, our estimation procedures are rooted in the graph-guided fused LASSO framework, which stimulates smooth parameter estimates along the edges of a graph (Xin et al., 2016). The graph-guided fused LASSO proves most powerful in scenarios where we have a priori expectations about the relationships for the problem at hand. This can be especially useful in settings concerning spatial grids, where interactions between cells can be similar over the entire grid. In this way, our approach using the graph-guided fused LASSO strikes a balance between the two opposing strands of literature discussed above. Our approach incorporates a priori knowledge about the spatial setting, similar to SAR and SDPD. However, this approach still retains the freedom for the model to infer the spatial relationships from the data.

Given the expansive potential applications of spatio-temporal models, it is essential to direct our focus to ensure a good understanding of the specific context. Therefore, the scope of this thesis primarily concerns spatial autoregressive models over a spatial grid involving regular, two-dimensional arrangements. As such, the methods developed in this thesis are particularly tailored towards these settings. We aim to contribute to the broader literature by focusing on this specific spatial application while providing a framework for developing specialized spatio-temporal models using graphs to incorporate domain knowledge using the graph-guided fused LASSO. With this context in mind, our research question arises: *Can the integration of the graph-guided fused LASSO lead to improved forecasting performance and estimation accuracy of spatio-temporal estimators, specifically when compared to the SPLASH estimator?*

This thesis is structured as follows: section 2 provides a comprehensive review of the spatial autoregressive model, the SPLASH procedure, and theory on relevant LASSO variants. Section 3 develops the novel GF-SPLASH, F-SPLASH, and SSF-SPLASH estimators by integrating the graph-guided fused LASSO with generalized Yule-Walker estimation. Then, section 4 discusses the setup for the simulation study, whose results are discussed in section 5. Finally, section 6 and section 7 discuss our key findings and potential directions for future research.

2 Theoretical Framework

2.1 Spatio-temporal autoregressive model

Recent papers by Gao et al. (2019) and Reuvers and Wijler (2021) consider the following formulation of the spatio-temporal autoregressive model:

$$\boldsymbol{y}_t = \boldsymbol{A}\boldsymbol{y}_t + \boldsymbol{B}\boldsymbol{y}_{t-1} + \boldsymbol{\varepsilon}_t, \quad \text{for } t \in \{1, \dots, T\}, \tag{1}$$

where the vector $\boldsymbol{y}_t \in \mathbb{R}^p$ stacks p observations of spatial cells at time t, defined over a grid. The matrix $\boldsymbol{A} \in \mathbb{R}^{p \times p}$, having zeros on its main diagonal, controls the spatial dependence between the contemporaneous spatial cells in $\boldsymbol{y}_t$. The matrix $\boldsymbol{B} \in \mathbb{R}^{p \times p}$ controls the dependence on past observations through $\boldsymbol{y}_{t-1}$. Lastly the innovation vector is denoted by

$\boldsymbol{\varepsilon}_t \in \mathbb{R}^p$, satisfying the following conditions for all $t \in \{1, \ldots, T\}$

$$\mathbb{E}(\boldsymbol{\varepsilon}_t) = 0, \quad \text{Var}(\boldsymbol{\varepsilon}_t) = \boldsymbol{\Sigma}_\varepsilon, \quad \text{and} \quad \text{Cov}(\boldsymbol{y}_{t-j}, \boldsymbol{\varepsilon}_t) = 0, \quad \text{for } j \in \{1, \ldots, t-1\}, \tag{2}$$

with $\boldsymbol{\Sigma}_\varepsilon$ being a positive definite covariance matrix, similar to Gao et al. (2019). Furthermore, in line with the approach of Reuvers and Wijler (2021), we make the following assumptions to ensure stability of (1):

$$\max\left\{\|\boldsymbol{A}\|_1, \|\boldsymbol{A}\|_\infty\right\} \leq \delta_A \leq 1, \quad \max\left\{\|\boldsymbol{B}\|_1, \|\boldsymbol{B}\|_\infty\right\} \leq C_B < 1 - \delta_A. \tag{3}$$

These assumptions guarantee that $(\boldsymbol{I}_p - \boldsymbol{A})$ is invertible and thus allows for a transformation to the reduced form VAR(1) representation,

$$\boldsymbol{y}_t = \boldsymbol{\Phi}\boldsymbol{y}_{t-1} + \boldsymbol{u}_t \triangleq (\boldsymbol{I}_p - \boldsymbol{A})^{-1}\boldsymbol{B}\boldsymbol{y}_{t-1} + (\boldsymbol{I}_p - \boldsymbol{A})^{-1}\boldsymbol{\varepsilon}_t, \quad \text{for } t \in \{1, \ldots, T\}. \tag{4}$$

Naturally, the number of unknown parameters in $\boldsymbol{A}$ and $\boldsymbol{B}$ grows quadratically in p, being equal to $2p^2 - p$, since $a_{ii} = 0$ for $i \in \{1, \ldots, p\}$ by assumption. Reuvers and Wijler (2021) identify three complications regarding the model in equation (1), namely endogeneity of $\boldsymbol{y}_t$, the vast and rapidly growing dimensionality resulting in infeasibility of unregularized estimation procedures, and the identifiability of the parameters. They discuss early spatio-temporal modelling strategies involving a prespecified spatial matrix, leaving a lower number of parameters to estimate (e.g., Dou et al. (2016), Kelejian and Prucha (1998), and Qu and Lee (2015)). Gao et al. (2019) and Reuvers and Wijler (2021) take a different approach by making the assumption of bandedness in the matrices $\boldsymbol{A} = (a_{ij})_{i,j=1}^p$ and $\boldsymbol{B} = (b_{ij})_{i,j=1}^p$, i.e.,

$$a_{ij} = b_{ij} = 0, \quad \text{for all } (i,j) \in \left\{(i,j) \in \{1, \ldots, p\}^2 \,\middle|\, |i-j| > k_0\right\}. \tag{5}$$

Moreover, Reuvers and Wijler (2021) assume an upper bound for the bandwidth $k_0 \leq \lfloor (p-1)/4 \rfloor$ and make a similar assumption on the covariance matrix $\boldsymbol{\Sigma}_\varepsilon = (\sigma_{ij})_{i,j=1}^p$:

$$\sigma_{ij} = 0, \quad \text{for all } (i,j) \in \left\{(i,j) \in \{1, \ldots, p\}^2 \,\middle|\, |i-j| > l_0\right\}. \tag{6}$$

2.2 SPLASH estimator

Reuvers and Wijler (2021) address the three complications in section 2.1 by imposing a sparse structure on $\boldsymbol{A}$ and $\boldsymbol{B}$, and estimating the coefficients using Yule-Walker equations (Brockwell & Davis, 1991). They demonstrate that diagonally structured sparsity appears when the spatial cells of $\boldsymbol{y}_t$ are ordered in a structured way. More precisely, when the entries of $\boldsymbol{y}_t$ are enumerated row-wise as cells into a grid, the coefficients of the neighbouring cells corresponding to entries of $\boldsymbol{y}_t$ group together on the off-diagonals of $\boldsymbol{A}$ and $\boldsymbol{B}$. Naturally, spatial cells that are further away from each other do not (strongly) interact with each other. When cells do not interact, the corresponding coefficients will equal zero, resulting in the described sparsity.

2.2.1 Generalized Yule-Walker estimation

We will shortly review the notation and reasoning concerning the generalized Yule-Walker estimation procedure used by Reuvers and Wijler (2021), whose approach is inspired by the earlier work of Gao et al. (2019). Many of the same elements in this section will be reused throughout the remainder of this thesis, adhering to many of the same notational conventions.

Denote the contemporaneous autocovariance matrix and the autocovariance of the first lag by $\boldsymbol{\Sigma}_0 = \mathbb{E}(\boldsymbol{y}_t \boldsymbol{y}_t')$ and $\boldsymbol{\Sigma}_1 = \mathbb{E}(\boldsymbol{y}_t \boldsymbol{y}_{t-1}')$, respectively, and consider their sample counterparts $\hat{\boldsymbol{\Sigma}}_0 = \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{y}_t \boldsymbol{y}_t'$ and $\hat{\boldsymbol{\Sigma}}_1 = \frac{1}{T}\sum_{t=2}^{T} \boldsymbol{y}_t \boldsymbol{y}_{t-1}'$. Post-multiplication of equation (1) with $\boldsymbol{y}_{t-1}'$ yields the Yule-Walker equations

$$\boldsymbol{\Sigma}_1 = \boldsymbol{A}\boldsymbol{\Sigma}_1 + \boldsymbol{B}\boldsymbol{\Sigma}_0, \tag{7}$$

which can be rewritten as

$$\boldsymbol{\Sigma}_1' = \begin{bmatrix} \boldsymbol{\Sigma}_1' & \boldsymbol{\Sigma}_0 \end{bmatrix} \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \end{bmatrix}' =: \boldsymbol{V}\boldsymbol{C}'. \tag{8}$$

Note that all coefficients from the spatial VAR(1) model in equation (1) are now collected in $\boldsymbol{C}'$. The assumption in equation (5) ensures that the parameters of $\boldsymbol{A}$ and $\boldsymbol{B}$ are identified and results in a large part of the coefficients of $\boldsymbol{C}'$ being zero. In their estimation procedure, these coefficients are disregarded from the beginning, and all non-zero entries of $\boldsymbol{C}'$ are collected and stacked row-wise in the vector $\boldsymbol{c}$. For any $\boldsymbol{M} \in \mathbb{R}^{n \times k}$, let

$$\mathscr{B}_h(\boldsymbol{M}) := \left(m_{ij} \cdot \mathbb{1}_{\{|i-j| \leq h\}} \right)_{i,j=1}^{n,k}. \tag{9}$$

Then, they estimate the banded covariance matrices denoted by $\mathscr{B}_{h_0}(\hat{\boldsymbol{\Sigma}}_0)$ and $\mathscr{B}_{h_1}(\hat{\boldsymbol{\Sigma}}_1)$ and use them to construct

$$\hat{\boldsymbol{V}}_h = \begin{bmatrix} \mathscr{B}_{h_1}(\hat{\boldsymbol{\Sigma}}_1)' & \mathscr{B}_{h_0}(\hat{\boldsymbol{\Sigma}}_0) \end{bmatrix}, \tag{10}$$

abbreviating the collection $\{h_0, h_1\}$ as h. Subsequently, they take the columns from $\hat{\boldsymbol{V}}_h$ that share their column index with the column indices of the non-zero coefficients in the ith row $\boldsymbol{C}'$, then collect these columns in the matrix $\hat{\boldsymbol{V}}_{i,h}$. The non-zero coefficients from the ith row of $\boldsymbol{C}'$ make up the sub-vector $\boldsymbol{c}_i$, which in turn stack together into the vector $\boldsymbol{c} = (\boldsymbol{c}_1, \ldots, \boldsymbol{c}_p)$, this process is explained in more detail in section (3.1). The matrices $\hat{\boldsymbol{V}}_{i,h}$ for $i \in \{1, \ldots, p\}$ are collected in a sparse diagonal block matrix

$$\hat{\boldsymbol{V}}_h^{(d)} = \begin{bmatrix} \hat{\boldsymbol{V}}_{1,h} & \boldsymbol{0} & \ldots & \ldots & \boldsymbol{0} \\ \boldsymbol{0} & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \hat{\boldsymbol{V}}_{i,h} & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \boldsymbol{0} \\ \boldsymbol{0} & \ldots & \ldots & \boldsymbol{0} & \hat{\boldsymbol{V}}_{p,h} \end{bmatrix}, \tag{11}$$

and they define the vectorized banded sample covariance matrix

$$\hat{\boldsymbol{\sigma}}_h = \text{vec}(\mathscr{B}_{h_1}(\hat{\boldsymbol{\Sigma}}_1)'). \tag{12}$$

Moreover, denote $\mathscr{D}_{\boldsymbol{M}}(k) = \left\{ m_{ij} \in \mathscr{E}(\boldsymbol{M}) \,\middle|\, i - j = k \right\}$ as the set of entries on the same diagonal of a matrix $\boldsymbol{M}$, and define the sets of index sets corresponding to the entries lying on the same (pairs of) diagonals in either $\boldsymbol{A}$ or $\boldsymbol{B}$ as follows:

$$\mathscr{G}_{\boldsymbol{A}} := \left\{ g \subset \mathbb{N} \,\middle|\, \boldsymbol{c}_g = \mathscr{D}_{\boldsymbol{A}}(k) \wedge k \in \mathbb{Z} \wedge -k_0 \leq k \leq k_0 \wedge k \neq 0 \right\}, \tag{13}$$

$$\mathscr{G}_{\boldsymbol{B}} := \left\{ g \subset \mathbb{N} \,\middle|\, \boldsymbol{c}_g = \mathscr{D}_{\boldsymbol{B}}(k) \wedge k \in \mathbb{Z} \wedge -k_0 \leq k \leq k_0 \right\}. \tag{14}$$

Note that we adhere to a slightly different notation than Reuvers and Wijler (2021), namely for convenience of notation for the derivations in section 3. Finally, they collect all these elements to construct the SPLASH(α, λ) estimator, with the objective function

$$\mathcal{L}_{\alpha}^{\text{SPL}}(\boldsymbol{c}; \lambda) = \left\| \hat{\boldsymbol{\sigma}}_h - \hat{\boldsymbol{V}}_h^{(d)} \boldsymbol{c} \right\|_2^2 + \lambda \left((1-\alpha) \sum_{g \in \mathscr{G}} \sqrt{|g|} \left\| \boldsymbol{c}_g \right\|_2 + \alpha \|\boldsymbol{c}\|_1 \right), \tag{15}$$

where $\mathscr{G} := \mathscr{G}_{\boldsymbol{A}} \cup \mathscr{G}_{\boldsymbol{B}}$. This objective function has the form of the sparse group LASSO (SGL) (Simon et al., 2013) and represents a penalized generalized Yule-Walker estimation approach.

2.2.2 Bootstrapped banded autocovariance estimation

We will cover the bootstrapped banded autocovariance estimation by Guo et al. (2016), as employed in the SPLASH estimation methodology from section 2.2 (Reuvers & Wijler, 2021). More specifically, the bandwidths h_0 and h_1 used to construct the banded autocovariance estimators $\mathscr{B}_{h_1}(\hat{\boldsymbol{\Sigma}}_1)$ and $\mathscr{B}_{h_0}(\hat{\boldsymbol{\Sigma}}_0)$ are determined by through this bootstrap procedure.

The procedure starts with creating a bootstrap estimator for $\boldsymbol{\Sigma}_j := \mathbb{E}(\boldsymbol{y}_t \boldsymbol{y}_{t-j}')$ as follows:

$$\boldsymbol{\Sigma}_j^* = \frac{1}{T} \sum_{t=1}^{T-j} u_t (\boldsymbol{y}_t - \bar{\boldsymbol{y}})(\boldsymbol{y}_{t+j} - \bar{\boldsymbol{y}})', \tag{16}$$

using $\bar{\boldsymbol{y}} := \frac{1}{T} \sum_{t=1}^{T} \boldsymbol{y}_t$ and $\{u_t\}_{t=1}^{T}$ is a sequence of independent and identically distributed draws from a distribution with mean and variance equal to 1. The objective function for the bootstrap estimator for the autocovariance bandwidth is then defined as

$$H_j^*(h) = \frac{1}{K} \sum_{k=1}^{K} \left\| \mathscr{B}_h(\boldsymbol{\Sigma}_{j,k}^*) - \hat{\boldsymbol{\Sigma}}_j \right\|_1, \tag{17}$$

where $\{\boldsymbol{\Sigma}_{j,k}^*\}_{k=1}^{K}$ is the sequence of K bootstrap samples for $\boldsymbol{\Sigma}_j$, obtained through equation (16). Finally, the bootstrapped bandwidth is defined as $h_j = \arg\min_h H_j^*(h)$ and can be

used to construct the banded autocovariance estimators.

Reuvers and Wijler (2021) derive convergence rates for the banded sample autocovariance estimators $\mathscr{B}_{h_0}(\hat{\boldsymbol{\Sigma}}_0)$ and $\mathscr{B}_{h_1}(\hat{\boldsymbol{\Sigma}}_1)$ used in the SPLASH estimator, and show that the bootstrap procedure by Guo et al. (2016) described above increase the convergence rates of the SPLASH estimator.

2.3 Penalized vector autoregressive estimator

The vector autoregressive (VAR) model has been extensively employed in the field of multivariate time series, especially showing utility in forecasting applications (Hamilton & Hamilton, 1994). This basic model has been extended to a penalized VAR (PVAR) variant that introduces sparsity by means of an l_1 penalty (Nicholson et al., 2017b). The addition of this penalty is of particular benefit when dealing with high-dimensional scenarios that are often encountered in spatio-temporal contexts, the central theme of this thesis.

The PVAR model is not able to model the structural formulation of the spatio-temporal autoregressive model as stipulated in equation (1). Whereas SPLASH and the novel methods developed in section 3 are able to directly estimate the spatial coefficient matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, the PVAR model cannot. However, the PVAR model can be applied to the reduced form VAR(1) representation in equation (4) to estimate the matrix $\boldsymbol{\Phi} = (\boldsymbol{I}_p - \boldsymbol{A})^{-1}\boldsymbol{B}$. To reiterate, write

$$\boldsymbol{y}_t = \boldsymbol{\Phi}\boldsymbol{y}_{t-1} + \boldsymbol{u}_t, \quad \text{for } t \in \{1, \dots T\}. \tag{18}$$

The PVAR(λ) estimator for this VAR(1) model can then be written as

$$\hat{\boldsymbol{\Phi}}(\lambda) = \arg\min_{\boldsymbol{\Phi}} \sum_{t=2}^{T} \|\boldsymbol{y}_t - \boldsymbol{\Phi}\boldsymbol{y}_{t-1}\|_2^2 + \lambda \sum_{i=1}^{p}\sum_{j=1}^{p} \big|\phi_{ij}\big|, \tag{19}$$

with $\boldsymbol{\Phi} = (\phi_{ij})_{i,j=1}^{p}$.

Nicholson et al. (2017b) have devised a comprehensive framework that efficiently computes the PVAR(λ) model. Their work encompasses numerous other model specifications and penalty structures, which go beyond the scope of this thesis. These methods have been implemented in the *BigVAR* package for the statistical software *R* (Nicholson et al., 2017a). This package serves as a powerful tool for large-scale VAR estimation and is readily available on the Comprehensive R Archive Network (CRAN).[1]

2.4 LASSO variants

This section reviews various variants of the Least Absolute Shrinkage and Selection Operator (LASSO), a method originally introduced by Tibshirani (1996). These LASSO variants have been covered extensively in the literature, and we will adhere to standard nomenclature for these models throughout this section. Subsequently, in section 3, we will adapt these methods and adjust the notation to our specific use case.

[1] The *R* package *BigVAR* is available on *CRAN* (https://cran.r-project.org/).

The LASSO is a method of regression analysis that performs both variable selection and regularization, enhancing both the prediction accuracy and the interpretability of the resulting statistical model, particularly in scenarios with high-dimensional data (Tibshirani, 1996). In the context of LASSO, define the target variable $\boldsymbol{y} \in \mathbb{R}^n$, and the matrix of explanatory variables or features $\boldsymbol{X} \in \mathbb{R}^{n \times p}$. Moreover, define the vector of coefficients, $\boldsymbol{\beta} \in \mathbb{R}^p$, and the regularization parameter, $\lambda \in [0, \infty)$. With these definitions, the standard LASSO estimator can be expressed as

$$\hat{\boldsymbol{\beta}}_{\text{LASSO}} := \arg\min_{\boldsymbol{\beta}} \frac{1}{2} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_1. \tag{20}$$

For sufficiently large values of λ, some coefficients in the estimated coefficient vector $\hat{\boldsymbol{\beta}}_{\text{LASSO}}$ will be exactly zero. This feature of LASSO is beneficial in many modelling contexts as it enables automatic variable selection and results in simpler, more interpretable models.

2.4.1 Generalized LASSO

The generalized LASSO broadens the scope of the conventional LASSO by incorporating a penalty matrix, denoted by $\boldsymbol{D}$, within the regularization component. Using this penalty matrix provides a degree of flexibility in tailoring the structure of the coefficient estimates (She, 2010; Tibshirani & Taylor, 2011). The regular LASSO model, through the implementation of the l_1 penalty, promotes a sparse solution, implying that a considerable number of coefficients are exactly zero. This attribute of sparsity is preserved in the generalized LASSO model while simultaneously permitting the inclusion of additional structure within the non-zero coefficients. The generalized LASSO can be written as

$$\hat{\boldsymbol{\beta}}_{\text{GLASSO}} := \arg\min_{\boldsymbol{\beta}} \frac{1}{2} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda \|\boldsymbol{D}\boldsymbol{\beta}\|_1, \tag{21}$$

the penalty matrix, represented by $\boldsymbol{D} \in \mathbb{R}^{m \times p}$ in equation (21), encapsulates the structure intended to be imposed on the coefficients, through penalizing linear combinations of the coefficients in $\boldsymbol{\beta}$. The penalization structure encoded in $\boldsymbol{D}$ empowers the generalized LASSO to handle a diverse range of problem settings. The generalized LASSO encompasses many specific variants of the LASSO, such as the Sparse Group LASSO used in SPLASH by Reuvers and Wijler (2021) or the 1D fused LASSO by Tibshirani et al. (2005), which ensures smoothly varying adjacent coefficients. The flexibility and customizability inherent to the generalized LASSO make it an appealing choice for numerous scenarios where an additional or custom structure is either expected or desired while remaining in a well-established and robust framework.

Tibshirani and Taylor (2011) showed that the generalized LASSO in equation (21) can be reduced to the regular LASSO form in equation (20) in two different scenarios. The first and simpler scenario occurs when the matrix $\boldsymbol{D}$ is $p \times p$ and invertible. By introducing

$\boldsymbol{\theta} = \boldsymbol{D}\boldsymbol{\beta}$ we can write (21) as

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \frac{1}{2}\left\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{D}^{-1}\boldsymbol{\theta}\right\|_2^2 + \lambda\|\boldsymbol{\theta}\|_1, \tag{22}$$

which can be solved by regular LASSO estimators, using $\boldsymbol{X}\boldsymbol{D}^{-1}$ instead of $\boldsymbol{X}$ in (20) and reconstructing the generalized LASSO estimate with $\hat{\boldsymbol{\beta}}_{\text{GLASSO}} = \boldsymbol{D}^{-1}\hat{\boldsymbol{\theta}}$.

In the scenario that $\boldsymbol{D}$ is a $m \times p$ matrix with $m < p$ and $\text{rank}(\boldsymbol{D}) = m$, Tibshirani and Taylor (2011) show that the matrix $\boldsymbol{D}$ can be extended by collecting the vectors that span a basis for the right nullspace of $\boldsymbol{D}$, denoted by $\text{null}(\boldsymbol{D})$. Let $\text{null}(\boldsymbol{D}) = \text{span}\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{p-m}\}$ and construct

$$\boldsymbol{N}^{(D)} = \begin{bmatrix} \boldsymbol{v}_1' \\ \vdots \\ \boldsymbol{v}_{p-m}' \end{bmatrix} \quad \text{and} \quad \tilde{\boldsymbol{D}} = \begin{bmatrix} \boldsymbol{D} \\ \boldsymbol{N}^{(D)} \end{bmatrix}. \tag{23}$$

Now, with $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) = \tilde{\boldsymbol{D}}\boldsymbol{\beta}$, which is partitioned such that $\boldsymbol{\theta}_1 = \boldsymbol{D}\boldsymbol{\beta}$ and $\boldsymbol{\theta}_2 = \boldsymbol{N}^{(D)}\boldsymbol{\beta}$, we can rewrite equation (21) as

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \frac{1}{2}\left\|\boldsymbol{y} - \boldsymbol{X}\tilde{\boldsymbol{D}}^{-1}\boldsymbol{\theta}\right\|_2^2 + \lambda\|\boldsymbol{\theta}_1\|_1. \tag{24}$$

Note that there is a difference between the coefficient vector $\boldsymbol{\theta}$ in the first term and $\boldsymbol{\theta}_1$ in the penalty term. To transform this into a regular LASSO problem, we partition the matrix product

$$\boldsymbol{X}\tilde{\boldsymbol{D}}^{-1} = \begin{bmatrix} \tilde{\boldsymbol{X}}_1 & \tilde{\boldsymbol{X}}_2 \end{bmatrix}, \tag{25}$$

where $\tilde{\boldsymbol{X}}_1 : p \times m$ and $\tilde{\boldsymbol{X}}_2 : p \times (p-m)$. Now, equation (24) can be viewed as an Ordinary Least Squares problem with respect to $\boldsymbol{\theta}_2$, which is clear when the objective in equation (24) is rewritten as

$$\min_{\boldsymbol{\theta}_2} \frac{1}{2}\left\|\left(\boldsymbol{y} - \tilde{\boldsymbol{X}}_1\boldsymbol{\theta}_1\right) - \tilde{\boldsymbol{X}}_2\boldsymbol{\theta}_2\right\|_2^2 + \lambda\|\boldsymbol{\theta}_1\|_1, \tag{26}$$

with solution $\hat{\boldsymbol{\theta}}_2 = (\tilde{\boldsymbol{X}}_2'\tilde{\boldsymbol{X}}_2)^{-1}\tilde{\boldsymbol{X}}_2'(\boldsymbol{y} - \tilde{\boldsymbol{X}}_1\boldsymbol{\theta}_1)$. Substitute $\hat{\boldsymbol{\theta}}_2$ back in equation (26) to obtain a problem that can be minimized with respect to $\boldsymbol{\theta}_1$

$$\min_{\boldsymbol{\theta}_1} \frac{1}{2}\left\|\left(\boldsymbol{y} - \tilde{\boldsymbol{X}}_1\boldsymbol{\theta}_1\right) - \tilde{\boldsymbol{X}}_2(\tilde{\boldsymbol{X}}_2'\tilde{\boldsymbol{X}}_2)^{-1}\tilde{\boldsymbol{X}}_2'(\boldsymbol{y} - \tilde{\boldsymbol{X}}_1\boldsymbol{\theta}_1)\right\|_2^2 + \lambda\|\boldsymbol{\theta}_1\|_1. \tag{27}$$

We can reorganize equation (27) by introducing the *annihilator* matrix

$$\boldsymbol{M} := \boldsymbol{I}_p - \tilde{\boldsymbol{X}}_2(\tilde{\boldsymbol{X}}_2'\tilde{\boldsymbol{X}}_2)^{-1}\tilde{\boldsymbol{X}}_2', \tag{28}$$

and obtain the estimator for $\boldsymbol{\theta}_1$, which is exactly formulated as a regular LASSO problem

$$\hat{\boldsymbol{\theta}}_1 = \arg\min_{\boldsymbol{\theta}_1} \frac{1}{2}\left\|\boldsymbol{M}\boldsymbol{y} - \boldsymbol{M}\tilde{\boldsymbol{X}}_1\boldsymbol{\theta}_1\right\|_2^2 + \lambda\|\boldsymbol{\theta}_1\|_1. \tag{29}$$

Finally, we reconstruct $\hat{\boldsymbol{\theta}} = (\hat{\boldsymbol{\theta}}_1, \hat{\boldsymbol{\theta}}_2)$, which in turn can be used to obtain $\hat{\boldsymbol{\beta}}_{\text{GLASSO}} = \tilde{\boldsymbol{D}}^{-1}\hat{\boldsymbol{\theta}}$, the solution to our original problem in equation (21).

Duan et al. (2016) show that the resulting estimator does not depend on the exact choice of $\boldsymbol{N}^{(D)}$ in equation (23), provided that the rows of $\boldsymbol{N}^{(D)}$ are orthogonal to the rows in $\boldsymbol{D}$.

2.4.2 Graph-guided fused LASSO

In the previous section, we shortly touched on a specific case of the generalized LASSO, more specifically, the 1D fused LASSO, which ensures smooth differences of the adjacent coefficients over time (Tibshirani et al., 2005). The 1D fused LASSO is limited in that it can only impose smoothness on the coefficients that are directly adjacent in the $\boldsymbol{\beta}$ vector. To allow for a more general approach in the pairs of coefficients that are enforced to have smooth differences, we consider the graph-guided fused LASSO (GF-LASSO) (Xin et al., 2016). Introduce the graph $G = (V, E)$, where V and E denote the sets vertices and edges of this graph, respectively. The edges $(u, v) \in E$ are (ordered) pairs of vertices that are connected in G. Each vertex $v \in V$ corresponds to the index of a coefficient in $\boldsymbol{\beta}$, such that each edge connects two coefficients. Now, we can define the GF-LASSO estimator as follows:

$$\hat{\boldsymbol{\beta}}_{\text{GF-LASSO}} := \arg\min_{\boldsymbol{\beta}} \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda \sum_{(i,j)\in E} \left|\beta_i - \beta_j\right|. \tag{30}$$

The graph G can be defined in any way appropriate for the specific problem at hand by connecting the coefficients such that the differences are penalized at the points where we expect or want to enforce smoothness among them.

The GF-LASSO can be rewritten to the generalized LASSO by considering the oriented incidence matrix of G (e.g., Gross and Yellen (2005)). The oriented incidence matrix is the incidence matrix of any orientation of the graph. An orientation of an undirected graph is obtained by assigning a direction to each of the edges, turning it into a directed graph.[2] The elements of the oriented incidence matrix of graph G can be defined as[3]

$$d_{ij}^{(G)} = \begin{cases} 1 & \text{if vertex } j \text{ is the head of edge } i \\ -1 & \text{if vertex } j \text{ is the tail of edge } i, \\ 0 & \text{otherwise.} \end{cases} \tag{31}$$

[2] Any orientation of the undirected graph G will lead to the same model because the penalty is defined as the sum of absolute values of the resulting differences.

[3] Technically, we consider the transpose of the oriented incidence matrix, which is more useful for our application.

Let G have p vertices and m edges, then

$$\boldsymbol{D}^{(G)} = \left(d_{ij}^{(G)}\right)_{i,j=1}^{m,p}. \tag{32}$$

For example, in the case of the 1D fused LASSO, G is the chain graph and $\boldsymbol{D}^{(G)}$ has the following structure:

$$\boldsymbol{D}^{(G)} = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \end{bmatrix}, \tag{33}$$

in the case of $\boldsymbol{\beta} : 5 \times 1$, such that $\boldsymbol{D}^{(G)} : (5-1) \times 5$. In essence, the oriented incidence matrix serves as an alternative representation of G.

Using $\boldsymbol{D}^{(G)}$, an equivalent formulation of the GF-LASSO estimator in equation (30) is

$$\hat{\boldsymbol{\beta}}_{\text{GF-LASSO}} = \arg\min_{\boldsymbol{\beta}} \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda \left\| \boldsymbol{D}^{(G)}\boldsymbol{\beta} \right\|_1. \tag{34}$$

This formulation complies with the form of the generalized LASSO in equation (21), and the algorithms to solve the generalized LASSO can directly be applied to any case of the GF-LASSO (Arnold & Tibshirani, 2014; Chen et al., 2010; Zhu, 2017).

The GF-LASSO is a specific instance of generalized LASSO that leverages a given graph structure to impose certain constraints on the model's coefficients. In GF-LASSO, the graph represents the interrelationships among the variables, and the edges of the graph encode the expectation that connected coefficients will have similar estimates. It is particularly powerful in scenarios where we have some a priori knowledge of relationships between predictors, which we can represent through the graph. This approach allows us to incorporate domain knowledge into the model and promote interpretable solutions.

2.4.3 Algorithms

Several algorithms to address the computational challenges of the LASSO and generalized LASSO have been proposed and employed over the years. Here, we briefly describe and compare three notable approaches that are relevant to the algorithms we develop in section 3: Cyclical Coordinate Descent (Friedman et al., 2010), Augmented Alternating Direction Method of Multipliers (Zhu, 2017), and the generalized LASSO Dual Path algorithm (Tibshirani & Taylor, 2011).

The Cyclical Coordinate Descent (CCD) algorithm optimizes the variables of the objective function one at a time, with each iteration cyclically passing through each variable (Wright, 2015). The *glmnet* package, developed by Friedman et al. (2010) and available in the statistical software R, employs the CCD algorithm for the LASSO problem, among others. The CCD algorithm, while simple in concept, can be highly efficient for high-dimensional problems with the ability to exploit potential sparsity in the objective.

The Augmented Alternating Direction Method of Multipliers (AugADMM) algorithm, developed by Zhu (2017), provides a fast and stable algorithm for solving the generalized LASSO problem in the form of equation (21). The AugADMM algorithm improves upon the regular ADMM algorithm (e.g., Boyd et al. (2011)) by reducing the computational cost at each iteration while retaining similar convergence speeds. This reduction is achieved by augmenting the variable in the updating step such that the matrix multiplications in the updating step simplify significantly.

Tibshirani and Taylor (2011) developed a Dual Path algorithm for solving the generalized LASSO in equation (21). Similar to the popular Least Angle Regression (LARS) algorithm for solving regular LASSO problems, it traces the entire solution path, which can be particularly advantageous when there is a need to choose the regularization parameter through cross-validation, for example (Efron et al., 2004). Arnold and Tibshirani (2014) provide efficient implementations of the Dual Path algorithm for the generalized LASSO in the R package *genlasso*, exploiting the piecewise linearity of the solution to the dual problem.[4] Moreover, they provide specialized implementations for specific cases of the generalized LASSO, such as the graph-guided fused LASSO in equation (30).

The AugADMM algorithm by Zhu (2017) and Dual Path algorithm by Tibshirani and Taylor (2011) are relatively efficient solutions to the generalized LASSO problem. However, these algorithms still do not attain the speed of the fastest algorithms available for solving the regular LASSO problem, such as the CCD algorithm by Friedman et al. (2010). Therefore, it can be preferred to centre modelling efforts for very high-dimensional problems around the regular LASSO formulation in equation (20), especially when time constraints are in play.

3 Methodology

In this section, we will develop custom estimators designed to estimate the spatio-temporal autoregressive model in the form of equation 1. Our first step is to create the Generalized Fused SPLASH (GF-SPLASH) estimator in section 3.1. This estimator is the most comprehensive of the three we will present. It offers a broad range of customizability but at the cost of computational efficiency. Using the GF-SPLASH estimator as a foundation, we continue to design two additional estimators: Fused SPLASH (F-SPLASH) in section 3.2 and Semi-Sparse Fused SPLASH (SSF-SPLASH) in section 3.3. These estimators are more specialized versions of GF-SPLASH and offer improved efficiency at the cost of flexibility.

Throughout this process, we heavily rely on the principles found in the generalized Yule-Walker estimation procedure, as developed by Reuvers and Wijler (2021). This procedure has proven effective in high-dimensional time-series models, and its principles are invaluable to the estimators developed here. While we preserve the fundamental principles inherent to the SPLASH estimators in this work, our main contribution lies in developing alternative penalty structures within the generalized LASSO framework and exploiting specific characteristics of this framework.

[4]The R packages *glmnet* and *genlasso* are open-source and available on *CRAN* (https://cran.r-project.org/)

The implementations of all three algorithms, along with the other code to generate and reproduce all content of this thesis, are available on the author's GitHub page.[5]

As explained in section 2.4.2, the graph-guided fused LASSO becomes particularly powerful in scenarios where there exists some a priori expectation about the relationships between the coefficients or variables in the problem at hand. This knowledge allows us to define the underlying graph representing the fusion penalty, which points the model in a specific direction while still retaining the freedom for the model to infer relationships from the data.

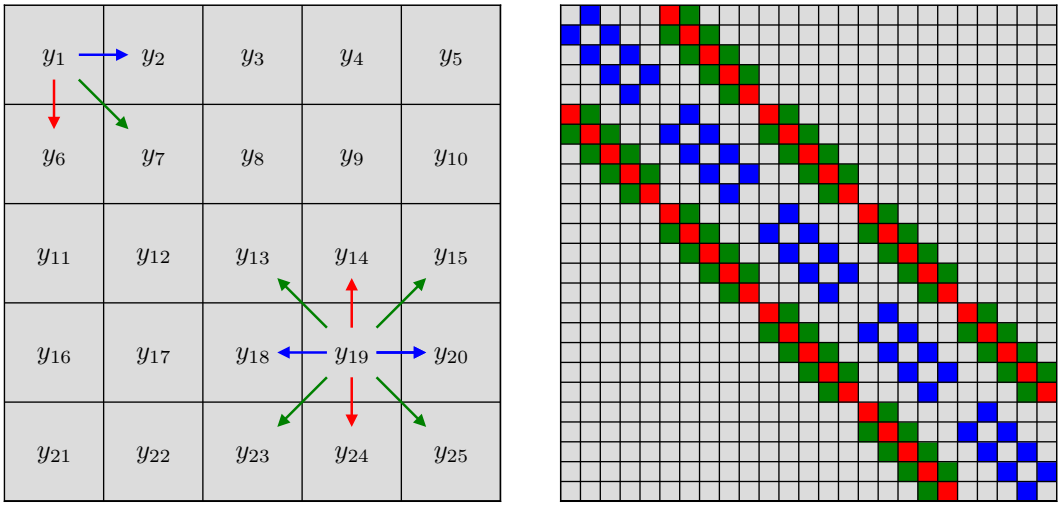

Figure 1: The left figure shows a 5×5 spatial grid with horizontal (blue), vertical (red), and diagonal (green) neighbouring interactions. The right figure represents the spatial coefficient matrix with coloured cells representing the coefficients corresponding to the neighbouring interactions.

This thesis is primarily concerned with the problem setting of spatial grids. Borrowing the graphical representation from Reuvers and Wijler (2021), Figure 1 represents a 5×5 grid of observations in a regular, two-dimensional arrangement. In this illustration, every spatial cell has interactions with its horizontal (blue), vertical (red), and diagonal (green) neighbours. The right plot represents the 25×25 spatial coefficient matrix, corresponding to $\boldsymbol{A}$ in equation (1). Reuvers and Wijler (2021) found that the coefficients corresponding to these neighbouring interactions collect on the diagonals, as visualized in the right plot of Figure 1.

In a spatial setting, especially for regularly arranged grids, it could be natural to expect that the interaction between horizontal neighbours y_1 and y_2 is similar to the horizontal interaction between y_2 and y_3, for example. The fusion penalty in the graph-guided fused LASSO can penalize the difference between the corresponding coefficients, i.e., $|a_{12} - a_{23}|$, by creating an edge between these coefficients. As such, we can create a graph connecting the coefficients that correspond to the same type of neighbouring interaction. In Figure 1, this would mean that edges are created between the coefficients on the diagonals of the

[5] All code pertaining to this thesis is available on the author's GitHub page (https://github.com/jaccobroere/vu-msc-thesis)

coloured cells. If we penalize the coefficients diagonally downwards in a pairwise fashion, it results in a penalization structure that emphasizes smoothness of coefficients pertaining to similar interactions.

3.1 GF-SPLASH estimator

The primary goal of the GF-SPLASH estimator is to provide estimates for the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, which form the core of the spatio-temporal autoregressive model as given in equation (1). The construction of this estimator draws heavily from the generalized Yule-Walker estimation procedure, which has been applied to estimate $\boldsymbol{A}$ and $\boldsymbol{B}$ in previous studies by Reuvers and Wijler (2021) and Gao et al. (2019). Borrowing directly from the SPLASH procedure by Reuvers and Wijler (2021), as discussed in section 2.2, we construct banded autocovariance $\mathscr{B}_{h_1}(\hat{\boldsymbol{\Sigma}}_1)$ and $\mathscr{B}_{h_0}(\hat{\boldsymbol{\Sigma}}_0)$ as per Guo et al. (2016). Subsequently, these banded estimators serve as the foundation to build the target variable $\hat{\boldsymbol{\sigma}}_h$ and the data matrix $\hat{\boldsymbol{V}}_h^{(d)}$, based on equation (12) and (11), respectively. We decide to slightly deviate from the notation of the SPLASH procedure, namely denoting the coefficient vector by $\boldsymbol{\phi}$ instead of $\boldsymbol{c}$, to avoid ambiguity of notation at later stages.

We continue to use these elements to construct an objective function following the structure of the generalized LASSO as per equation (21). The foundational form of the GF-SPLASH objective function is as follows:

$$\mathcal{L}(\boldsymbol{\phi}; \lambda) = \left\| \hat{\boldsymbol{\sigma}}_h - \hat{\boldsymbol{V}}_h^{(d)} \boldsymbol{\phi} \right\|_2^2 + \lambda \|\boldsymbol{D}\boldsymbol{\phi}\|_1. \tag{35}$$

The resemblance to the SPLASH estimator, by Reuvers and Wijler (2021) in equation (15) is clear. The l_2 error term based on the generalized Yule-Walker equations remains unchanged. The main difference lies in the penalty term, which we alter through the introduction of the penalty matrix $\boldsymbol{D}$. This matrix governs the structure of the penalty imposed on $\boldsymbol{\phi}$ by creating linear combinations of coefficients that should be penalized jointly.

To construct the penalty matrix $\boldsymbol{D}$, we need to elaborate on the concepts introduced in sections 2.2 and 2.4. The construction of $\boldsymbol{D}$ is a structured process, resulting in a tailored penalty that exploits the spatial nature of the problem outlined in Figure 1.

First, note that the coefficient vector $\boldsymbol{\phi}$ is constructed such that only the coefficients from $\boldsymbol{A}$ and $\boldsymbol{B}$ that are allowed to be non-zero, are included, as discussed in section 2.2.1. Namely, all entries outside the bandwidth in the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are set to zero directly. We adopt the same bandedness assumption as Reuvers and Wijler (2021), described in equation (5), which means that all entries $a_{ij} \in \boldsymbol{A}$ and $b_{ij} \in \boldsymbol{B}$ are equal to zero for $|i-j| > k_0 = \lfloor (p-1)/4 \rfloor$, along with a zero diagonal in $\boldsymbol{A}$. Those entries are excluded from the estimation procedure by leaving those entries out of the coefficient vector $\boldsymbol{\phi}$. Figure 2 visualizes the entries that are put in $\boldsymbol{\phi}$ in shades of red, along with their corresponding index. The excluded entries are coloured blue for $p = 9$.

We create a mapping from the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ to the coefficient vector $\boldsymbol{\phi}$. We can construct this mapping iteratively, starting at the first coefficient of $\boldsymbol{\phi}$ and iterate through the entries of $\boldsymbol{C} = \begin{bmatrix} A & B \end{bmatrix}$ row by row, if the ijth entry of $\boldsymbol{C}$ falls within the bandwidth,

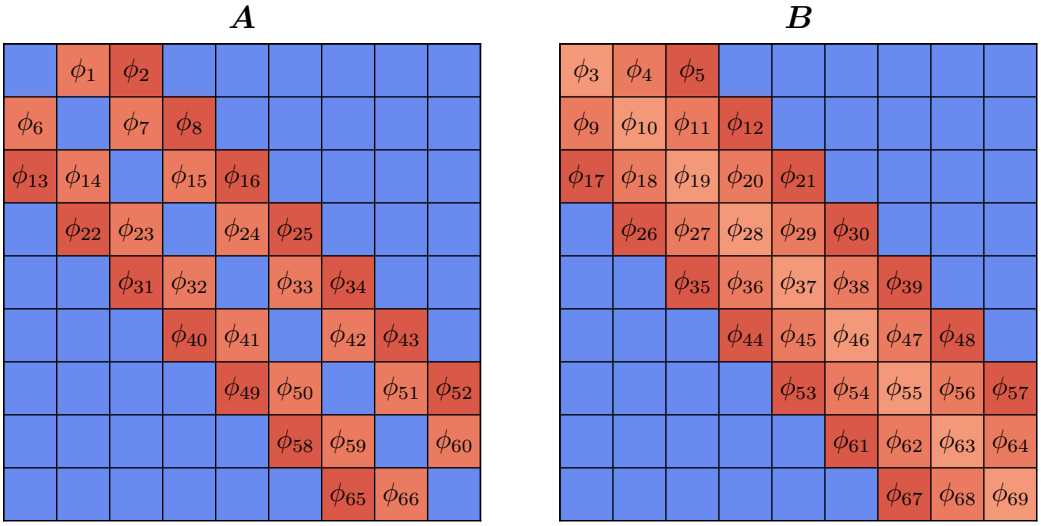

Figure 2: Coefficients of $\boldsymbol{\phi}$ with corresponding position in $\boldsymbol{A}$ (left) and $\boldsymbol{B}$ (right) for $p = 9$

we assign it to the next index of $\boldsymbol{\phi}$.

More formally, we first introduce the bandwidth index sets as

$$k_{\boldsymbol{A}} = \{i \in \mathbb{Z} \,|\, -k_0 \leq i \leq k_0 \wedge i \neq 0\} \quad \text{and} \quad k_{\boldsymbol{B}} = \{i \in \mathbb{Z} \,|\, -k_0 \leq i \leq k_0\}, \tag{36}$$

using the maximum bandwidth $k_0 = \lfloor (p-1)/4 \rfloor$. Next, define the sets of entries on particular non-zero diagonals of $\boldsymbol{A}$ and $\boldsymbol{B}$ as

$$\mathscr{D}_{\boldsymbol{A}}(k) = \left\{a_{ij} \in \mathscr{E}(\boldsymbol{A}) \,\middle|\, i - j = k\right\} \quad \text{and} \quad \mathscr{D}_{\boldsymbol{B}}(k) = \left\{b_{ij} \in \mathscr{E}(\boldsymbol{B}) \,\middle|\, i - j = k\right\}, \tag{37}$$

denoting $\mathscr{E}(\boldsymbol{M})$ as the set of entries of any matrix $\boldsymbol{M}$. Now, write the set of all non-zero entries in $\boldsymbol{A}$ and $\boldsymbol{B}$ as

$$\mathscr{D}_{\boldsymbol{C}} := \mathscr{D}_{\boldsymbol{A}} \cup \mathscr{D}_{\boldsymbol{B}} \triangleq \left(\bigcup_{k \in k_{\boldsymbol{A}}} \mathscr{D}_{\boldsymbol{A}}(k) \right) \cup \left(\bigcup_{k \in k_{\boldsymbol{B}}} \mathscr{D}_{\boldsymbol{B}}(k) \right). \tag{38}$$

Since the non-zero entries of $\boldsymbol{C}$ should be enumerated row by row into $\boldsymbol{\phi}$, we can define the indices of the entry to be added in the nth iteration as

$$i(n) = \min \left\{ i \in \mathbb{N} \,\middle|\, \exists j : c_{ij} \in \mathscr{M}_n^{\boldsymbol{\phi}} \right\}, \tag{39}$$

$$j(n) = \min \left\{ j \in \mathbb{N} \,\middle|\, i = i(n) \wedge c_{ij} \in \mathscr{M}_n^{\boldsymbol{\phi}} \right\}. \tag{40}$$

Here, we define the set of entries that have yet to be added to $\boldsymbol{\phi}$ on the nth iteration as

$$\mathscr{M}_n^{\boldsymbol{\phi}} = \mathscr{D}_{\boldsymbol{C}} \backslash \{\phi_i\}_{i=1}^{n-1}. \tag{41}$$

Now, the vector of coefficients can be defined as $\boldsymbol{\phi} = (\phi_n)_{n=1}^{k}$, where

$$\phi_n = c_{ij}, \quad \text{with } (i,j) = \big(i(n), j(n)\big)\,. \tag{42}$$

In constructing the penalty matrix $\boldsymbol{D}$, we draw upon the principles of the graph-guided fused LASSO and construct an underlying graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, as described in section 2.4.2. The mapping between $\boldsymbol{C}$ and $\boldsymbol{\phi}$ is used to create the underlying graph. The vertices are defined to be the set of coefficients in $\boldsymbol{\phi}$, i.e., $\mathcal{V} := \mathscr{E}(\boldsymbol{\phi})$. The number of vertices in $\mathcal{V}$ can be calculated as

$$k := |\mathcal{V}| \triangleq \big|\mathscr{E}(\boldsymbol{\phi})\big| = p(4k_0 + 1) - 2(k_0^2 + k_0). \tag{43}$$

Refer to Appendix A for a derivation of the result in equation (43). Using the mapping, we create an edge between two vertices if their respective positions are diagonally connected in $\boldsymbol{\phi}$, as the same type of neighbouring interactions are collected in the diagonals. Figure 3 visualizes the pairs of coefficients in $\boldsymbol{A}$ and $\boldsymbol{B}$ that make up the edges by connecting them with white arrows. More formally, the set of edges can then be defined using the indices

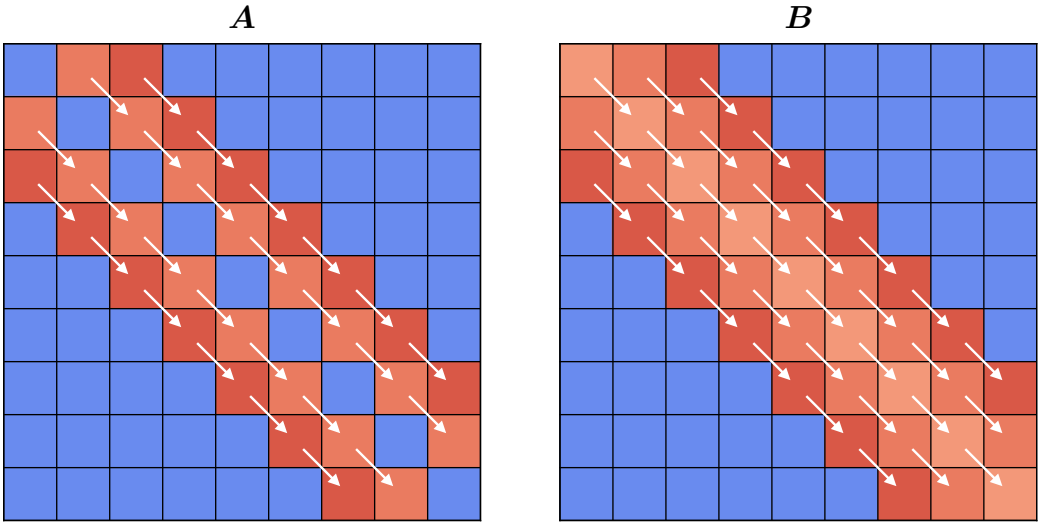

Figure 3: Pairs to construct edges for diagonally connected coefficients in $\boldsymbol{A}$ (left) and $\boldsymbol{B}$ (right) for $p = 9$

of the entries in the original matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, write

$$\begin{aligned}\mathcal{E} := &\Big\{(a_{ij}, a_{gh}) \in \mathscr{D}^2_{\boldsymbol{A}} \,\Big|\, g = i+1 \wedge h = j+1\Big\} \\ &\cup \Big\{(b_{ij}, b_{gh}) \in \mathscr{D}^2_{\boldsymbol{B}} \,\Big|\, g = i+1 \wedge h = j+1\Big\}\,.\end{aligned} \tag{44}$$

With $\mathcal{G}$ constructed, define $\boldsymbol{D}^{(\mathcal{G})}$ to be the oriented incidence matrix of graph $\mathcal{G}$, as per equation (31). By setting $\boldsymbol{D} = \boldsymbol{D}^{(\mathcal{G})}$ in equation (35), we obtain an estimator which has a pure fusion penalty, where only the pairwise differences of coefficients on the diagonals

are penalized

$$\begin{aligned}\hat{\boldsymbol{\phi}}(\lambda) &= \arg\min_{\boldsymbol{\phi}} \left\| \hat{\boldsymbol{\sigma}}_h - \hat{\boldsymbol{V}}_h^{(d)} \boldsymbol{\phi} \right\|_2^2 + \lambda \left\| \boldsymbol{D}^{(\mathcal{G})} \boldsymbol{\phi} \right\|_1, \\ &= \arg\min_{\boldsymbol{\phi}} \left\| \hat{\boldsymbol{\sigma}}_h - \hat{\boldsymbol{V}}_h^{(d)} \boldsymbol{\phi} \right\|_2^2 + \lambda \sum_{(\phi_i, \phi_j) \in \mathcal{E}} \left| \phi_i - \phi_j \right|.\end{aligned} \tag{45}$$

The second line reminds of the SPLASH$(0, \lambda)$ estimator, which only penalizes complete diagonals without any stimulation towards sparsity within these diagonals. Similar to SPLASH(α, λ) with $\alpha > 0$, we can add an l_1 penalty term on the individual coefficients of $\boldsymbol{\phi}$ to equation (45) and introduce the hyperparameter $\alpha \in [0, 1]$ as follows:

$$\hat{\boldsymbol{\phi}}(\alpha, \lambda) = \arg\min_{\boldsymbol{\phi}} \left\| \hat{\boldsymbol{\sigma}}_h - \hat{\boldsymbol{V}}_h^{(d)} \boldsymbol{\phi} \right\|_2^2 + \lambda \left((1-\alpha) \sum_{(\phi_i, \phi_j) \in \mathcal{E}} \left| \phi_i - \phi_j \right| + \alpha \|\boldsymbol{\phi}\|_1 \right), \tag{46}$$

which is better known in the literature as the sparse fused LASSO, as it enforces a fusion penalty and promotes sparsity among the individual coefficients. However, this formulation does not comply with the generalized LASSO framework; reformulating back to the generalized LASSO requires adjusting the penalty matrix to accompany the l_1 penalty. Similarly to Tibshirani and Taylor (2011), define

$$\boldsymbol{D}_\alpha = \begin{bmatrix} (1-\alpha)\boldsymbol{D}^{(\mathcal{G})} \\ \alpha \boldsymbol{I}_k \end{bmatrix}, \tag{47}$$

and rewrite the estimator in equation (46) to

$$\hat{\boldsymbol{\phi}}(\alpha, \lambda) = \arg\min_{\boldsymbol{\phi}} \left\| \hat{\boldsymbol{\sigma}}_h - \hat{\boldsymbol{V}}_h^{(d)} \boldsymbol{\phi} \right\|_2^2 + \lambda \|\boldsymbol{D}_\alpha \boldsymbol{\phi}\|_1, \tag{48}$$

which complies with the formulation of the generalized LASSO again.

We can further extend the estimator in (48), by extending $\boldsymbol{D}_\alpha$ with another set of penalties. Intuitively, in spatial settings, the effects of two spatial cells could be expected to be symmetric, i.e., the effect of spatial cell y_i on y_j could be expected to be similar or equal to the effect of y_j on y_i. To accompany this expectation, we can introduce a fusion penalty between the coefficients that are positioned across each other on pairs of diagonals. Figure (4) illustrates this by highlighting the non-zero entries of $\boldsymbol{A}$ and $\boldsymbol{B}$ in shades of red and drawing arrows between the entries that engage in the fusion penalty.

Formally, create a second graph $\mathcal{G}_s = (\mathcal{V}_s, \mathcal{E}_s)$, where the vertices again correspond to the coefficients in $\boldsymbol{\phi}$. Then we use the same mapping from $\boldsymbol{A}$ and $\boldsymbol{B}$ to the coefficients of $\boldsymbol{\phi}$ as we did for the diagonally connected coefficients, and create the edges as follows:

$$\mathcal{E}_s = \left\{ (a_{ij}, a_{ji}) \in \mathscr{D}^2_{\boldsymbol{A}} \,\middle|\, i < j \right\} \cup \left\{ (b_{ij}, b_{ji}) \in \mathscr{D}^2_{\boldsymbol{B}} \,\middle|\, i < j \right\}. \tag{49}$$

With $\mathcal{G}_s$ constructed, calculate the corresponding oriented incidence matrix, and denote it by $\boldsymbol{D}^{(\mathcal{G}_s)}$. Introduce another hyperparameter $s \in \{0, 1\}$ and use $\boldsymbol{D}^{(\mathcal{G}_s)}$ to extend the

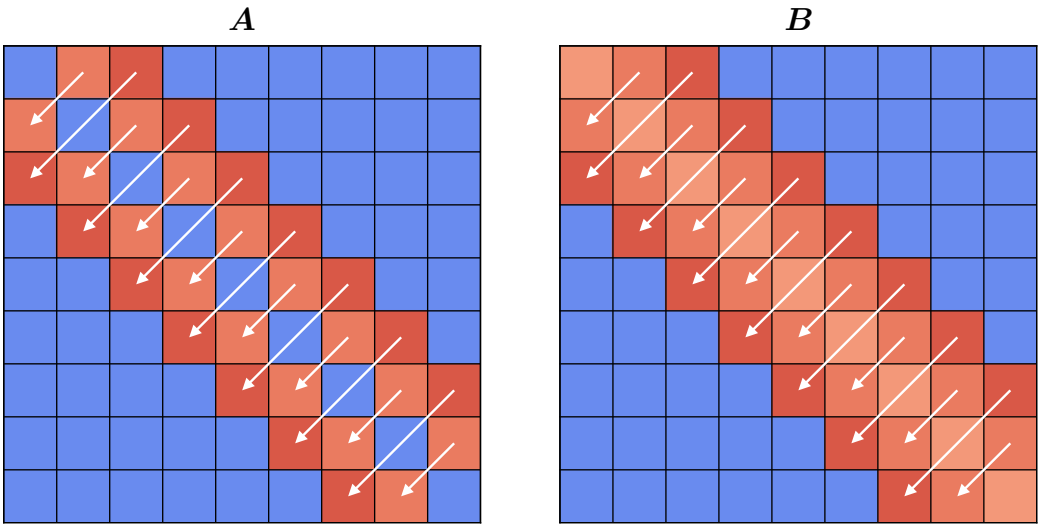

Figure 4: Pairs to construct a penalty for symmetric diagonals in $\boldsymbol{A}$ (left) and $\boldsymbol{B}$ (right) for $p = 9$

penalty matrix if $s = 1$

$$\tilde{\boldsymbol{D}}_{\alpha,1} = \begin{bmatrix} (1-\alpha)\boldsymbol{D}^{(\mathcal{G})} \\ \alpha \boldsymbol{I}_k \\ s\boldsymbol{D}^{(\mathcal{G}_s)} \end{bmatrix}, \tag{50}$$

in the case of $s = 0$ we have $\tilde{\boldsymbol{D}}_{\alpha,0} := \boldsymbol{D}_\alpha$ from equation (47). Note that the penalty matrix $\boldsymbol{D}^{(\mathcal{G}_s)}$ purely extends the existing penalty matrix, controlled by $s \in \{0, 1\}$, it does not change the existing penalties on the diagonally connected coefficients using $\boldsymbol{D}^{(\mathcal{G})}$ or the individual coefficients using $\boldsymbol{I}_k$.

Finally, we can write the GF-SPLASH(α, λ, s) estimator as

$$\hat{\boldsymbol{\phi}}(\alpha, \lambda, s) = \arg\min_{\boldsymbol{\phi}} \left\| \hat{\boldsymbol{\sigma}}_h - \hat{\boldsymbol{V}}_h^{(d)} \boldsymbol{\phi} \right\|_2^2 + \lambda \left\| \tilde{\boldsymbol{D}}_{\alpha,s} \boldsymbol{\phi} \right\|_1. \tag{51}$$

To summarize, GF-SPLASH(α, λ, s) has three parameters; $\alpha \in [0, 1]$ governs the balance between the fusion penalization of the diagonally connected coefficients and the sparsity of the individual coefficients, $\lambda \in [0, \infty)$ controls the general strength of the regularization, and $s \in \{0, 1\}$ determines whether the symmetry penalty is included. Moreover, we conjecture that the effectiveness of the model can be increased by letting s vary continuously, giving it finer control of the balance between the symmetric and diagonal penalty. However, we decided to leave this outside the scope of this study.

In section 2.4.1, we discussed the transformation of the generalized LASSO to a regular LASSO. This is possible whenever the penalty matrix $\boldsymbol{D}$ has equal or more columns than rows and is of full row rank. Then, we can invert the penalty matrix directly or extend the penalty matrix to be invertible and transform the problem to a regular LASSO. Consider the penalty matrix $\tilde{\boldsymbol{D}}_{\alpha,s}$ in the GF-SPLASH estimator from equation (51). By definition,

the number of columns is equal to k as per equation (43). Moreover, denote the number of rows of $\tilde{\boldsymbol{D}}_{\alpha,s}$ by m. The number of rows m can be calculated by considering the dimension of the submatrices that make up $\tilde{\boldsymbol{D}}_{\alpha,s}$ as per equation (50). The dimension of the middle term $\alpha \boldsymbol{I}_k$ is straightforward, being a scaled $k \times k$ identity matrix. The number of rows in the matrices $\boldsymbol{D}^{(\mathcal{G})}$ and $\boldsymbol{D}^{(\mathcal{G}_s)}$ are determined by the number of edges in their corresponding graphs. First, consider the number of edges in $\mathcal{G}$. As Figure 3 illustrates, the number of edges is exactly one less than the number of vertices for each diagonal, thus we have

$$|\mathcal{E}| = k - (4k_0 + 1) = p(4k_0 + 1) - 2(k_0^2 + k_0) - (4k_0 + 1). \tag{52}$$

The number of edges in $\mathcal{G}_s$ is equal to half the number of vertices that correspond to off-diagonal entries, as illustrated in Figure 4. We have

$$|\mathcal{E}_s| = 2\sum_{i=1}^{k_0}(p - i) = 2pk_0 - k_0(k_0 + 1), \tag{53}$$

and total number of rows in $\tilde{\boldsymbol{D}}_{\alpha,s}$ can now easily be calculated as

$$m = k + \mathbb{1}_{\{\alpha>0\}}|\mathcal{E}| + s|\mathcal{E}_s|. \tag{54}$$

Clearly, if $\alpha > 0$, we have that $m > k$ for both $s = 0$ and $s = 1$, which means that the GF-SPLASH estimator in equation (51) cannot be transformed to a regular LASSO problem. In section 3.2 and 3.3, we will explore the case that $\alpha = 0$ and construct more restricted estimators that exploit the advantages of transforming the problem to the regular LASSO.

Algorithm 1 in Appendix C provides the *pseudo-algorithm* to fit the general form of the GF-SPLASH(α, s, λ) estimator in equation 51. The algorithm selects the parameters α and λ using time-series cross-validation, which is explained in more detail in section 3.4. Furthermore, the GF-SPLASH(α, s, λ) in equation (51) is formulated as a generalized LASSO problem, which fits the requirements to use the AugADMM algorithm, as discussed in section 2.4.3 (Zhu, 2017).

3.2 F-SPLASH estimator

This section is dedicated to the construction of a specialized variant of the GF-SPLASH estimator, namely the Fused SPLASH (F-SPLASH) estimator. In section 2.4.1, it is shown that a generalized LASSO problem can be reduced to a LASSO problem under the requirement that $\boldsymbol{D} \in \mathbb{R}^{m\times p}$ has rank$(\boldsymbol{D}) = m$ with $m \leq p$ (Tibshirani & Taylor, 2011). The complete GF-SPLASH(α, λ, s) estimator in equation (51) does not satisfy this condition if $\alpha \neq 0$ or $s \neq 0$.

However, in the case of $\alpha = 0$ and $s = 0$, we are left with the estimator as per equation (45). Then, we have that the penalty matrix is equal to $\boldsymbol{D}^{(\mathcal{G})} \in \mathbb{R}^{m\times k}$, where m and k are defined in equations (52) and (43), respectively. Notice that we have $m < k$ for any bandwidth k_0 and cross-sectional dimension p, which means that the problem can be reduced to a regular lasso using equations (23)-(29) from section 2.4.1.

Tibshirani and Taylor (2011) show that the right nullspace of the oriented incidence

matrix $\boldsymbol{D}^{(G)}$ of any graph G with p vertices, is spanned by indicator vectors. Let $C = \{C_1, \ldots, C_n\}$ be the set of connected components of G. A connected component of G is a set of vertices that are connected by the edges in G (e.g., Gross and Yellen (2005)). The indicator vector for the ith connected component can then be defined element-wise with its jth element equal to

$$(\mathbf{1}_{C_i})_j := \begin{cases} 1 & \text{if vertex } j \in C_i \\ 0 & \text{otherwise.} \end{cases} \tag{55}$$

Tibshirani and Taylor (2011) show that the right nullspace of G is spanned by these indicator vectors, i.e.

$$\text{null}\left(\boldsymbol{D}^{(G)}\right) = \text{span}\left\{\mathbf{1}_{C_1}, \ldots, \mathbf{1}_{C_n}\right\}. \tag{56}$$

Consider $\mathcal{G}$, the graph of the diagonally connected coefficients in $\boldsymbol{\phi}$, as developed in section 3.1. Clearly, the set of connected components of graph $\mathcal{G}$, denoted by $\mathcal{C}$, is equal to the sets of coefficients that are positioned on the same diagonal, as illustrated in Figure 3. This means that there are $4k_0 + 1$ connected components, which is equal to $k - m$, the number of vertices minus the number of edges of $\mathcal{G}$, as per equations (43) and (52). The right nullspace of $\boldsymbol{D}^{(\mathcal{G})} \in \mathbb{R}^{m \times k}$ is then spanned by $\{\mathbf{1}_{\mathcal{C}_1}, \ldots, \mathbf{1}_{\mathcal{C}_{k-m}}\}$. Now construct

$$\boldsymbol{N}^{(\mathcal{G})} = \begin{bmatrix} \mathbf{1}'_{\mathcal{C}_1} \\ \vdots \\ \mathbf{1}'_{\mathcal{C}_{k-m}} \end{bmatrix} \quad \text{and} \quad \tilde{\boldsymbol{D}} = \begin{bmatrix} \boldsymbol{D}^{(\mathcal{G})} \\ \boldsymbol{N}^{(\mathcal{G})} \end{bmatrix}, \tag{57}$$

which we can use to transform the problem to a regular LASSO, as per equations (24)-(29). A visualization of the structure of the penalty matrix $\tilde{\boldsymbol{D}}$ can be found in Figure 7 in Appendix B. Note that the penalty matrix extension serves solely as a tool to transform the problem to a regular LASSO, making the estimation process computationally efficient. This extension explicitly does not impose any additional penalties or change the existing penalties on the estimator. More specifically, the resulting F-SPLASH(λ) estimator is equivalent to GF-SPLASH($0, 0, \lambda$).

More specifically, we continue to apply the procedure of equations (24)-(29), starting from the GF-SPLASH($0, 0, \lambda$) estimator

$$\hat{\boldsymbol{\phi}}(\lambda) = \arg\min_{\boldsymbol{\phi}} \left\| \hat{\boldsymbol{\sigma}}_h - \hat{\boldsymbol{V}}_h^{(d)} \boldsymbol{\phi} \right\|_2^2 + \lambda \left\| \boldsymbol{D}^{(\mathcal{G})} \boldsymbol{\phi} \right\|_1, \tag{58}$$

and using $\tilde{\boldsymbol{D}}$ as in equation (57) to define

$$\boldsymbol{\theta} = \begin{bmatrix} \boldsymbol{\theta}_1 \\ \boldsymbol{\theta}_2 \end{bmatrix} := \begin{bmatrix} \boldsymbol{D}^{(\mathcal{G})} \boldsymbol{\phi} \\ \boldsymbol{N}^{(\mathcal{G})} \boldsymbol{\phi} \end{bmatrix} \triangleq \tilde{\boldsymbol{D}} \boldsymbol{\phi}. \tag{59}$$

Now, we rewrite the GF-SPLASH$(0, 0, \lambda)$ estimator as follows:

$$\hat{\boldsymbol{\theta}}(\lambda) = \arg\min_{\boldsymbol{\theta}} \left\| \hat{\boldsymbol{\sigma}}_h - \hat{\boldsymbol{V}}_h^{(d)} \tilde{\boldsymbol{D}}^{-1} \boldsymbol{\theta} \right\|_2^2 + \lambda \|\boldsymbol{\theta}_1\|_1. \tag{60}$$

Next, as in section 3.1, we calculate

$$\hat{\boldsymbol{V}}_h^{(d)} \tilde{\boldsymbol{D}}^{-1} = \begin{bmatrix} \tilde{\boldsymbol{X}}_1 & \tilde{\boldsymbol{X}}_2 \end{bmatrix} \quad \text{and} \quad \hat{\boldsymbol{\theta}}_2 = (\tilde{\boldsymbol{X}}_2' \tilde{\boldsymbol{X}}_2)^{-1} \tilde{\boldsymbol{X}}_2' (\hat{\boldsymbol{\sigma}}_h - \tilde{\boldsymbol{X}}_1 \hat{\boldsymbol{\theta}}_1). \tag{61}$$

Then, using the *annihilator* matrix $\boldsymbol{M} := \boldsymbol{I}_p - \tilde{\boldsymbol{X}}_2 (\tilde{\boldsymbol{X}}_2' \tilde{\boldsymbol{X}}_2)^{-1} \tilde{\boldsymbol{X}}_2'$, we substitute $\hat{\boldsymbol{\theta}}_2$ in equation (60) and rewrite the estimator to a LASSO problem

$$\hat{\boldsymbol{\theta}}_1(\lambda) = \arg\min_{\boldsymbol{\theta}_1} = \frac{1}{2} \left\| \boldsymbol{M} \hat{\boldsymbol{\sigma}}_h - \boldsymbol{M} \tilde{\boldsymbol{X}}_1 \boldsymbol{\theta}_1 \right\|_2^2 + \lambda \|\boldsymbol{\theta}_1\|_1. \tag{62}$$

Refer to equations (24)-(29) in section 3.1 for the intermediate steps of this transformation. Finally, reconstruct $\hat{\boldsymbol{\theta}} = (\hat{\boldsymbol{\theta}}_1, \hat{\boldsymbol{\theta}}_2)$ to obtain the F-SPLASH(λ) estimator

$$\hat{\boldsymbol{\phi}}(\lambda) = \tilde{\boldsymbol{D}}^{-1} \hat{\boldsymbol{\theta}}. \tag{63}$$

Algorithm 2 in Appendix C provides the *pseudo-algorithm* for computing the F-SPLASH estimator in equation 63. In contrast to the GF-SPLASH estimator using the AugADMM by Zhu (2017), the F-SPLASH estimator can leverage the computational efficiency of the CCD algorithm to solve the main step of the algorithm in equation (62) (Friedman et al., 2010). However, there are other parts of the F-SPLASH that can also be computationally expensive, such as the inversion of $\tilde{\boldsymbol{D}}$, whose dimension grows quadratically in p. The matrix $\tilde{\boldsymbol{D}}$, however, depends only on the dimensionality of the problem and not the data itself. Therefore, especially in use cases where the same model is fitted on different or updated data, the inverse of $\tilde{\boldsymbol{D}}$ has to be calculated only once and can be saved for future use. Moreover, $\tilde{\boldsymbol{D}}$ is a highly sparse matrix, allowing the use of more efficient sparse matrix algorithms to compute the inverse, such as the sparse LU factorization (Davis & Duff, 1997).

3.3 SSF-SPLASH estimator

This section develops the Semi-Sparse Fused SPLASH (SSF-SPLASH) estimator, serving as an alternative specialized variant of the GF-SPLASH estimator in section 3.1. The previous section introduced the F-SPLASH estimator, where we transform GF-SPLASH from a generalized LASSO to a regular LASSO problem. In this transformation we noted that $\boldsymbol{D}^{(\mathcal{G})} \in \mathbb{R}^{m \times k}$, has fewer rows than columns, i.e., $m < k$. Such, that $\boldsymbol{D}^{(\mathcal{G})}$ should be extended by an orthogonal set of rows to make the matrix square and invertible, allowing it to be rewritten to a regular LASSO problem (Duan et al., 2016).

However, this is not the only possible approach to extending the matrix $\boldsymbol{D}^{(\mathcal{G})}$ to be invertible; any set of $k - m$ linearly independent rows would suffice. The approach taken for the F-SPLASH(λ) estimator only ensures that it results in the same solution as GF-SPLASH$(0, 0, \lambda)$ would find, though much more efficiently. This provokes a hypothesis; can

we use another set of linearly independent rows that not only extend the penalty matrix to be invertible but also provides additional penalization to induce better solutions?

This section explores this hypothesis by extending the penalty matrix to induce sparsity of complete diagonals. To prevent violating the condition that requires the penalty matrix to have fewer rows than columns, the extension of the penalty matrix is limited to $k-m = 4k_0+1$ additional rows, as stipulated by equations (52) and (43). The difference, $k-m$, is exactly equal to the number of diagonals that are allowed to be non-zero, which means that we can extend the penalty matrix with one penalty corresponding to each diagonal. To indirectly promote sparse diagonals, we include an l_1 penalty on a single coefficient within each diagonal. More specifically, we select the coefficients in $\boldsymbol{\phi}$ from the first row of $\boldsymbol{C}$ where all non-zero diagonals are represented. For $p = 9$, as in Figure 2, this would correspond to the third row. Formally, let the penalty vectors $\boldsymbol{u}_i \in \mathbb{R}^k$ for $i \in \{1, \dots, k-m\}$, be defined element-wise with their jth element equal to

$$(\boldsymbol{u}_i)_j := \begin{cases} \sqrt{\left|\mathscr{D}_{\boldsymbol{A}}(i-k_0-1)\right|} & \text{if } i \leq k_0 \wedge j = 3k_0^2 + i \\ \sqrt{\left|\mathscr{D}_{\boldsymbol{A}}(i-k_0)\right|} & \text{if } k_0 < i \leq 2k_0 \wedge j = 3k_0^2 + i \\ \sqrt{\left|\mathscr{D}_{\boldsymbol{B}}(i-3k_0-1)\right|} & \text{if } i > 2k_0 \wedge j = 3k_0^2 + i \\ 0 & \text{otherwise.} \end{cases} \tag{64}$$

Note that there are $3k_0^2$ coefficients in the rows before the row where all coefficients are represented, as per equation (76) in Appendix A. Here we adopt a scaling reminiscent of the sparse group LASSO algorithm by Simon et al. (2013), namely by scaling the penalties for each diagonal set by the square root of their cardinality. The scaling is intended to equalize the importance of the fusion penalty and the promotion of sparse diagonals. Since there are only $4k_0 + 1$ penalties pertaining to the latter, opposing to the large number of fusion penalties, as per equation (52).

We continue to extend the penalty matrix $\boldsymbol{D}^{(\mathcal{G})}$ with the penalty vectors in a manner similar to the extension for F-SPLASH in equation (57). Namely, collect the penalty vectors in a matrix as follows:

$$\boldsymbol{D}^{(u)} = \begin{bmatrix} \boldsymbol{u}_1' \\ \vdots \\ \boldsymbol{u}_{k-m}' \end{bmatrix} \tag{65}$$

and write the complete SSF-SPLASH penalty matrix as follows:

$$\tilde{\boldsymbol{D}}_\alpha = \begin{bmatrix} (1-\alpha)\boldsymbol{D}^{(\mathcal{G})} \\ \alpha \boldsymbol{D}^{(u)} \end{bmatrix}. \tag{66}$$

The SSF-SPLASH penalty matrix is visualized in Figure 7 in Appendix B. Note that the conversion process from a generalized LASSO to a regular LASSO differs between SSF-SPLASH and the previously discussed F-SPLASH. For SSF-SPLASH, we extend the penalty matrix with additional penalties we want to impose on the model, resulting in a

penalty matrix that is directly invertible, which means that we can use the more straightforward method of converting to a regular LASSO, as described in equation (22) in section 2.4.1. In contrast to F-SPLASH, where we extended the penalty matrix solely to transform the existing problem to a regular LASSO, which demanded additional logic to ensure that the penalty matrix extension did not affect the final estimator.

Next, the SSF-SPLASH estimator is defined as a LASSO estimator, using $\boldsymbol{\theta} = \tilde{\boldsymbol{D}}_\alpha \boldsymbol{\phi}$

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} = \frac{1}{2}\left\| \hat{\boldsymbol{\sigma}}_h - \hat{\boldsymbol{V}}_h^{(d)} \tilde{\boldsymbol{D}}_\alpha^{-1} \boldsymbol{\theta} \right\|_2^2 + \lambda \|\boldsymbol{\theta}\|_1. \tag{67}$$

Finally, transform this back to obtain the SSF-SPLASH(α, λ) estimator

$$\hat{\boldsymbol{\phi}}(\lambda) = \tilde{\boldsymbol{D}}_\alpha^{-1} \hat{\boldsymbol{\theta}}. \tag{68}$$

The SSF-SPLASH(α, λ) estimator has the same computational advantages as the F-SPLASH estimator, along with the advantage that there are fewer costly transformations and matrix multiplications necessary to transform to a LASSO problem. For computational details, refer to the *pseudo-algorithm* of the SSF-SPLASH estimator in Appendix C.

3.4 Time-series cross-validation

Time-series are characterized by their temporal nature, with observations being dependent on their preceding values. Traditional cross-validation techniques are not directly applicable to time-series data due to this inherent temporal dependency (Bergmeir & Benítez, 2012). For example, standard k-fold cross-validation randomly splits the dataset into k folds, which is appropriate for independent and identically distributed data. However, applying this method to time-series violates the assumption of independence, as it disregards the temporal dependency of observations and can cause so-called *leakage* of future information into the validation process. This may result in overly optimistic performance estimates on the validation data that are not representative of true out-of-sample forecasting accuracy.

To address this issue, we can employ a cross-validation scheme that takes the temporal dependence into account. Multiple approaches exist; however, we opt to implement *rolling-window* cross-validation (Bergmeir & Benítez, 2012). This method preserves temporal order by training on a window of consecutive observations and evaluating on subsequent ones. On the next fold, the window moves forward by removing the earliest (set of) observations in the current window and adding the next (set of) observations in the sequence. Then it repeats the training and evaluation process until a predefined number of folds is reached or the entire sequence of observations is covered. Figure 5 illustrates this concept with a total of 15 observations and a window size of 10 (blue). The performance in each fold is evaluated on the single observation in red, known as the validation data.

We employ this cross-validation scheme for the selection of the parameters α, s and λ in the GF-SPLASH(α, s, λ), F-SPLASH(λ), and SSF-SPLASH(α, λ) models. For a detailed explanation and the steps of this cross-validation scheme applied to our models, refer to the *pseudo-algorithms* in Appendix C.

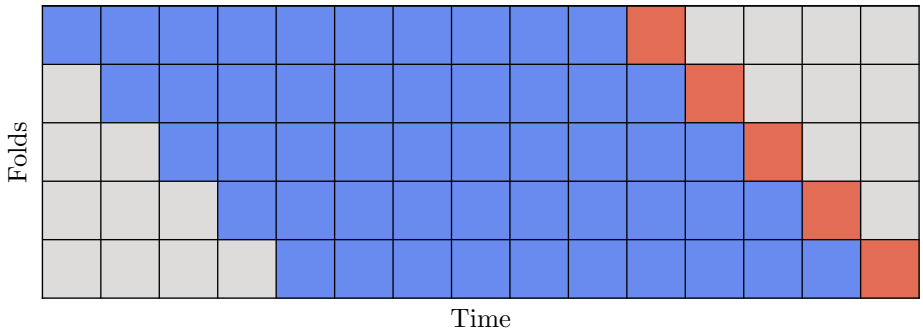

Figure 5: Schematic illustration for five folds of time-series cross-validation using a sliding window. The training data (blue) consists of 10 observations in each fold, and the validation data (red) consists of a single observation in each fold.

4 Simulation study

4.1 Data generating process

In this section, we set up three Monte Carlo simulation experiments that replicate particular spatial scenarios. The intention is to evaluate the performance of the developed models in section 3 and gauge their capabilities in different settings where they could be of practical interest. All simulations are repeated for $N_{\text{sim}} = 250$ iterations. Moreover, we consider three different lengths of time-series, namely $T \in \{500, 1000, 2000\}$, and two cross-sectional dimensions $p \in \{25, 36\}$. The data is simulated according to the spatio-temporal VAR model in equation (1). Where we draw all innovation vectors according to

$$\boldsymbol{\varepsilon}_t \overset{\text{i.i.d.}}{\sim} N(0, \boldsymbol{I}_p), \quad \text{for } t \in \{1, \dots, T+1\}. \tag{69}$$

This leaves us with only $\boldsymbol{A}$ and $\boldsymbol{B}$ to specify in the simulation designs. Afterwards, we can use $\Phi := (\boldsymbol{I}_p - \boldsymbol{A})^{-1}\boldsymbol{B}$ to iteratively generate the entire set of observations using the reduced form VAR(1) representation in equation (4).

4.1.1 Design 1: Horizontal and vertical neighbours

Design 1 concerns a spatial setting on a grid, also considered by Reuvers and Wijler (2021). We consider this design for the cross-sectional dimensions $p \in \{25, 36\}$, which correspond to 5×5 and 6×6 spatial grids, respectively. More specifically, we define the contemporaneous spatial interaction matrix $\boldsymbol{A}$ to contain interactions between first-order horizontal and vertical neighbours. The magnitude of the interactions in $\boldsymbol{A}$ are equal to 0.2. Furthermore, the matrix $\boldsymbol{B}$, governing the dependence on the previous observation $\boldsymbol{y}_{t-1}$, is defined as a diagonal matrix with entries equal to 0.25 and 0.23 for $p = 25$ and $p = 36$, respectively. This design results in a reduced form matrix $\boldsymbol{\Phi}$ respectively having a maximum eigenvalue of 0.814 and 0.824 for $p = 25$ and $p = 36$, ensuring that the reduced form VAR(1) process is stable.

4.1.2 Design 2: Asymmetric neighbours

Design 2 is a slight modification to the spatial setting that Design 1 stipulates; we consider the same spatial setting with horizontal and vertical neighbours. However, to test the effect of imposing symmetry in the model, we take the spatial interaction matrix $\boldsymbol{A}$ and scale the entries in the lower diagonal part of the matrix. More specifically, denote the spatial interaction matrix for Design 1 and 2 with $\boldsymbol{A} = (a_{ij})_{i,j=1}^{p}$ and $\boldsymbol{A}^* = (a_{ij}^*)_{i,j=1}^{p}$, then we write the entries of $\boldsymbol{A}^*$ as

$$a_{ij}^* = \begin{cases} \tau a_{ij} & \text{for } i > j, \\ a_{ij} & \text{otherwise,} \end{cases} \tag{70}$$

and let $\tau = 0.75$. Moreover, the matrix $\boldsymbol{B}$ remains a diagonal matrix; however, its diagonal entries are respectively set to 0.30 and 0.28 for $p = 25$ and $p = 36$ in this design. This setup results in a reduced form matrix $\boldsymbol{\Phi} = (\boldsymbol{I}_p - \boldsymbol{A}^*)^{-1}\boldsymbol{B}$ which has a maximum eigenvalue of 0.750 and 0.745 for $p = 25$ and $p = 36$, respectively.

4.1.3 Design 3: Extension to diagonal neighbours

Design 3 also concerns a spatial setting on a grid and serves as a natural extension of Design 1. First-order diagonal neighbouring interactions are added along with the horizontal and vertical interactions. The magnitude of the horizontal and vertical interactions is now 0.15, while the diagonal interactions have a magnitude of 0.1, considering that the average distance is larger to points in the cell of a diagonal neighbour compared to the horizontal and vertical neighbours. This results in a matrix $\boldsymbol{A}$, having the structure of the matrix presented in the right plot of Figure 1. The matrix $\boldsymbol{B}$ is again a diagonal matrix, now with entries respectively equal to 0.15 and 0.11 for $p = 25$ and $p = 36$. This design results in a reduced form matrix $\boldsymbol{\Phi}$ that respectively has a maximum eigenvalue of 0.832 and 0.817 for $p = 25$ and $p = 36$.

4.2 Models and parameter selection

In section 2 we discussed the SPLASH(α, λ) estimator, as developed by Reuvers and Wijler (2021), and the PVAR(λ) estimator (Nicholson et al., 2017b). Furthermore, in section 3, we introduced three novel estimators, GF-SPLASH(α, s, λ), F-SPLASH(λ), and SSF-SPLASH(α, λ). To gauge the performance of these models in spatio-temporal contexts, we employ a set of model specifications on each simulation design discussed in section 4.1. We limit the simulation study to model variants with $\alpha \in \{0, 0.5\}$ and $s \in \{0, 1\}$ to reduce the number of possible configurations and computational effort required. The case of $\alpha = 1$ is not considered in this study because GF-SPLASH and SPLASH become identical and Reuvers and Wijler (2021) reported worse performance of this specification, compared to SPLASH with $\alpha \in \{0, 0.5\}$. The first set of configurations we consider in this simulation study corresponds to $\alpha = 0$, namely F-SPLASH(λ), GF-SPLASH$(0, 1, \lambda)$, and SPLASH$(0, \lambda)$. These configurations stimulate only the spatial structures, namely the fusion penalties over a graph in F-SPLASH and GF-SPLASH and the group penalty in the

SPLASH model. The configurations with $\alpha = 0.5$ represent the second set. They strike a balance between the spatially structured penalties and the unstructured LASSO penalty on individual coefficients.

Furthermore, the parameter λ, governing the severity of penalization, is selected separately for each model configuration and Monte Carlo iteration through a time-series cross-validation scheme, as described in section 3.4. Specifically, we employ a three-fold cross-validation. In each fold, we define an 80/20 split between training and validation data. For each Monte Carlo iteration, cross-validation fold, and model configuration, we determine the minimum value for λ that produces a solution with all coefficients equal to zero, denoted by λ_0.

In the F-SPLASH and SSF-SPLASH, the objective function is transformed to a regular LASSO, for which we have

$$\lambda_0^{\text{F}} = \frac{1}{p^2}\left\|\left(\boldsymbol{M}\tilde{\boldsymbol{X}}_1\right)'\boldsymbol{M}\hat{\boldsymbol{\sigma}}_h\right\|_\infty \quad \text{and} \quad \lambda_0^{\text{SSF}} = \frac{1}{p^2}\left\|\left(\hat{\boldsymbol{V}}_h^{(d)}\tilde{\boldsymbol{D}}_\alpha^{-1}\right)'\hat{\boldsymbol{\sigma}}_h\right\|_\infty, \tag{71}$$

where the columns of $\boldsymbol{M}\tilde{\boldsymbol{X}}_1$ and $\hat{\boldsymbol{V}}_h^{(d)}\tilde{\boldsymbol{D}}_\alpha^{-1}$ have first been standardized (Friedman et al., 2010). The fraction $1/p^2$ is an artefact from the fact that the number of rows of $(\boldsymbol{M}\tilde{\boldsymbol{X}}_1)$ and $\hat{\boldsymbol{V}}_h^{(d)}\tilde{\boldsymbol{D}}_\alpha^{-1}$ are equal to p^2. For the GF-SPLASH model, we use the same approach as Zhu (2017). Namely, λ_0 can be calculated by estimating equation (51) using only the first iteration of the *genlasso* dual path algorithm for the generalized LASSO, developed by Tibshirani and Taylor (2011).[6] Note that λ_0 depends on the training data that is used to calculate it. Hence, each cross-validation fold has a different value for λ_0.

Using λ_0 at each fold, we construct a decreasing sequence of $M = 20$ values, denoted by $\Omega_\lambda = \{\lambda_0, \ldots, \lambda_{M-1}\}$, where we have

$$\log_{10}(\lambda_i) = \log_{10}(\lambda_0) - \frac{i}{M-1}\log_{10}(r) \quad \text{for } i \in \{0, \ldots, M-1\}. \tag{72}$$

Here, $r = 10^{-4}$ is the multiplier used to define the minimum value for λ in the sequence, such that $\lambda_{M-1} = r\lambda_0$. This approach results in a grid of parameter values that are equally spaced on a logarithmic scale. We determine the optimal value by first identifying the grid index i, yielding the lowest average mean squared forecast error (MSFE) across all folds. Afterwards, we determine λ_0 and Ω_λ using all the training data and select the optimal value as λ_i at the previously identified optimal index.

4.3 Evaluation setup

Having described the simulation designs, models, and parameter selection, we shift our focus to the evaluation strategy and adopt a similar approach as Reuvers and Wijler (2021). A simulation study has the advantage that the models can directly be evaluated in terms of the estimated coefficients, as we are aware of the ground truth underlying each data generating process. Therefore, at each Monte Carlo iteration, we assess the estimation

[6] The R package *genlasso* is available on *CRAN* (https://cran.r-project.org/)

error (EE) for both estimated matrices $\hat{\boldsymbol{A}}$ and $\hat{\boldsymbol{B}}$ and the one-step ahead relative mean squared forecast error (RMSFE) of each model.

The RMSFE is calculated as follows. For each Monte Carlo iteration, we calculate the one-step ahead forecast $\hat{\boldsymbol{y}}_{T+1}^q$ using the estimated model, where T is the last time period of the training data and $q \in \{1, \ldots, N_{\text{sim}}\}$ denotes the current Monte Carlo iteration. The one-step ahead RMSFE for the estimated model over all Monte Carlo simulations is then calculated as

$$\textbf{RMSFE} = \frac{\sum_{q=1}^{N_{\text{sim}}} \left\| \hat{\boldsymbol{y}}_{T+1}^q - \boldsymbol{y}_{T+1}^q \right\|_2^2}{\sum_{q=1}^{N_{\text{sim}}} \left\| \boldsymbol{\Phi} \boldsymbol{y}_T^q - \boldsymbol{y}_{T+1}^q \right\|_2^2}, \tag{73}$$

where $\boldsymbol{y}_t$ and $\hat{\boldsymbol{y}}_t$ respectively denote the true observations and predictions for $t \in \{1, \ldots, T+1\}$. Moreover, $\boldsymbol{\Phi} := (\boldsymbol{I}_p - \boldsymbol{A})^{-1}\boldsymbol{B}$ is the true coefficient matrix governing the reduced form VAR(1) representation as stipulated in equation (4).

The estimation error (EE) for the estimated coefficients matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ can be defined as follows. First, collect the estimated matrices of $\boldsymbol{A}$ and $\boldsymbol{B}$ of the qth Monte Carlo iterations in

$$\hat{\boldsymbol{C}}^q := \begin{bmatrix} \hat{\boldsymbol{A}}^q & \hat{\boldsymbol{B}}^q \end{bmatrix}, \quad \text{for } q \in \{1, \ldots, N_{\text{sim}}\}, \tag{74}$$

and write the estimation error as

$$\textbf{EE} = \frac{1}{N_{\text{sim}}} \sum_{q=1}^{N_{\text{sim}}} \left\| \hat{\boldsymbol{C}}^q - \boldsymbol{C} \right\|_2. \tag{75}$$

Note that this metric is not available for the PVAR model, as it directly estimates $\boldsymbol{\Phi}$.

These measures allow us to assess both the forecasting and estimation performance of the models under each of the simulation designs. By evaluating these metrics across different lengths of time-series and cross-sectional dimensions, we can gain insights into how the models perform under varying data characteristics, which is critical for a good assessment of their practical utility.

4.4 Runtime analysis

In addition to assessing the performance of our estimators, it is crucial to examine their computational efficiency. The runtime of these models is paramount in applications where computational resources are limited or when the dimensionality of the problem is large. Therefore, this section presents a runtime analysis of the models for different problem dimensionalities.

The objective of this analysis is to identify how the computational time required to obtain a solution with an algorithm changes as the cross-sectional dimensionality of the problem, p, increases. This provides an empirical benchmark of the algorithm's efficiency, a critical aspect, especially when it comes to handling large-scale spatio-temporal data in practical scenarios. As the dimensionality grows, computational time could become

impractical or prohibitive if not managed effectively.

To assess the computational efficiency, we compute the solution for a grid of ten λ values on a validation set and then use the optimal value for λ to estimate the model using all training data. We perform these calculations for varying cross-sectional dimensionalities p while keeping the time dimension fixed at $T = 1000$. The value of T does not have a large impact on the runtime of each model, as it is only involved in the calculation of the sample covariance matrices and the validation of the models for parameter selection. Furthermore, we exclude the bootstrap procedure from the runtime estimates, as the procedure is the same for each algorithm and would distract from the assessment of model efficiency. The data for this experiment is generated using the instructions from Design B in section 4.1. This process is repeated ten times for each model, and we calculate the average runtime over these ten iterations to obtain a robust estimate of the computational efficiency of each model.

The computation for this runtime analysis is carried out on a computer equipped with an Apple M1 Pro processor (8 cores).[7] It is worth noting that, while every effort was made to ensure optimal conditions for these tests, actual runtimes may vary due to the system load at the time of the tests. Hence, the runtimes reported in this analysis should be interpreted as indicative rather than definitive.

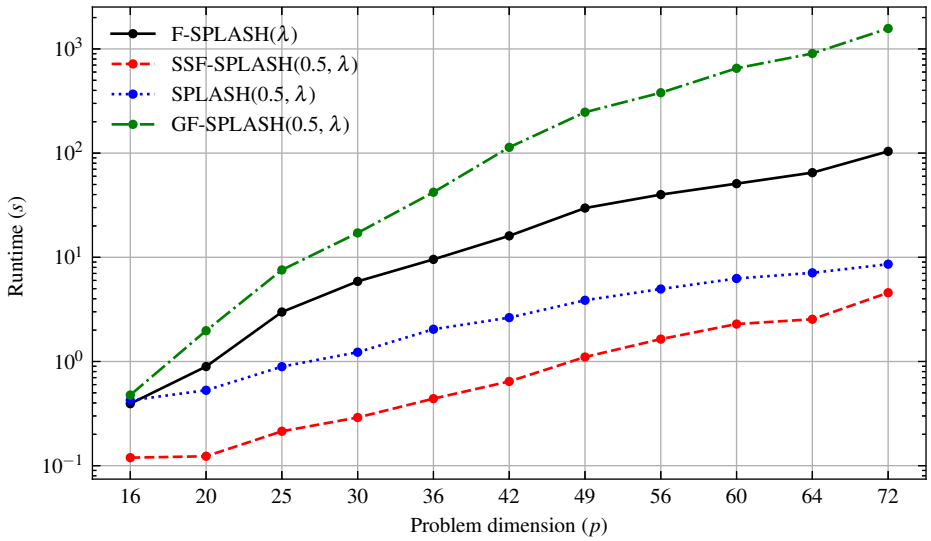

Figure 6: Average runtime of different models across ten iterations for varying problem dimensionalities p.

Figure 6 shows the runtimes of the different models, with the cross-sectional problem dimension p on the x-axis and the runtime in seconds on the y-axis. Note that the y-axis uses a logarithmic scale to accommodate the large discrepancy in runtimes for higher dimensionalities.

The first thing that stands out in Figure 6 is the very fast runtimes of SSF-SPLASH.

[7] A comprehensive benchmark and full specifications of the Apple M1 Pro (8 cores) processor can be found here: https://www.notebookcheck.net/apple-m1-pro-processor-benchmarks-and-specs.579915.0.html

However, as discussed in more detail in section 5, we find that SSF-SPLASH often provides near zero predictions, resulting in very fast convergence towards these severely sparse solutions. Instead, SPLASH provides a solid baseline performance, to compare our methods with. Reuvers and Wijler (2021) provide a specialized and optimized algorithm in C++, and we find that SPLASH achieves stable and fast runtimes under increasing dimensionality. The GF-SPLASH model, our most general model, employs the Augmented ADMM algorithm to solve the generalized LASSO problem, as detailed in section 3.1 (Zhu, 2017). We find that the runtimes of the GF-SPLASH model are quickly becoming infeasible under rising dimensionality, crossing the 1000 seconds mark for a cross-sectional dimensionality of $p = 72$. In contrast, F-SPLASH exploits the more efficient CCD algorithm for solving the regular LASSO, which is reflected by the lower and more manageable runtimes.

It is essential to note that the computational efficiency is significantly impacted by the programmatic implementation of the algorithms. All algorithms have been implemented using a combination of R and C++. Despite the diligent effort to optimize the algorithms, we still employed a lot of generic and possibly slow functions. As such, we conjecture that the implementations are suboptimal and that the efficiency of the F-SPLASH, GF-SPLASH, and SSF-SPLASH estimators could see substantial improvement by creating a specialized low-level implementation of these algorithms.

5 Results

The simulation results for Design 1, 2, and 3 are displayed in Tables 1, 2, and 3, respectively. Each table contains two rubrics, namely the Relative Mean Squared Forecast Error (RMSFE) in the first and the Estimation Error (EE) for the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ in the second rubric. Note that we have abbreviated names of the novel estimators (F-SPLASH, SSF-SPLASH, and GF-SPLASH), replacing SPLASH with SPL to make the tables more compact and readable.

Starting with Design 1 in Table 1, concerning the spatial setting with first-order horizontal and vertical neighbours. Firstly, we observe a general decrease in the forecasting performance, as indicated by the RMSFE, as the sample size T increases. Anomalies to this trend appear in the F-SPLASH(λ) and GF-SPLASH(0, 1, λ) models. Specifically, these models showed a marginally worse forecasting performance for $T = 1000$ compared to $T = 500$. The F-SPLASH model emerges as the best in terms of forecasting performance. However, it was marginally outperformed twice by two other models. Both these models, GF-SPLASH(0, 1, λ) and SPLASH(0, λ), have a purely spatial penalty structure. These purely spatial model configurations report notably better performance than the balanced configurations with $\alpha = 0.5$. Next, we observe that SPLASH$(0, \lambda)$ shows a steady increase in performance as T increases, while the F-SPLASH and GF-SPLASH models report good performance already at $T = 500$.

Furthermore, we find that the forecasting performance of the SSF-SPLASH model is remarkably worse than the other models. A closer look at the SSF-SPLASH estimates reveals an interesting pattern. It appears that the model often estimates the spatial interaction matrix $\boldsymbol{A}$ to be completely zero. This suggests that the model might be disregarding

any contemporaneous spatial interaction. Lastly, we generally see that the SPLASH models are quite strongly outperformed by the F-SPLASH and GF-SPLASH(0, 1, λ) models for $T \in \{500, 1000\}$. However, the performance gap decreases, and the margins become especially close when $T = 2000$. This might be an indication of faster convergence rates for the fused LASSO-based models compared to the SPLASH models.

In terms of the estimation error for Design 1 in Table 1, we find that the SPLASH(0, λ) model outperforms the other methods, next the SSF-SPLASH(0.5, λ) and SPLASH(0.5, λ) show good performance. This could be attributed to the fact that Design 1 has many completely sparse diagonals. The SPLASH estimator can effectively exploit these with its group LASSO. Similarly, the SSF-SPLASH estimator has a mechanism that promotes sparse diagonals indirectly. On the other hand, the GF-SPLASH models and F-SPLASH models do not promote completely sparse diagonals but smooth diagonals, which are not necessarily promoted to zero. We find that this decrease in estimation performance does not necessarily mean that the forecasting performance is also harmed. Again, we note that the estimation performance generally becomes better as T increases.

Now, we examine the results for Design 2 in Table 2, where the neighbouring interactions from Design 1 are made asymmetric. The most surprising result here is that the GF-SPLASH estimator with $s = 1$ performs best in terms of forecasting performance for most scenarios. This is unexpected as these estimators promote equal opposing diagonals, which is explicitly not the case in this simulation design. This might be caused by the symmetry penalties inducing better sparsity patterns throughout the entire problem, yielding better forecasting performance overall. By inspection, we found that GF-SPLASH(0, 1, λ) did indeed yield an estimated matrix $\hat{\boldsymbol{A}}$ with equal opposing diagonals. In terms of estimation accuracy, we find very similar results as in Design 1, where the SPLASH models are able to more accurately recover the coefficient matrices.

Lastly, we evaluate the results for Design 3 in Table 3. Additionally, to Design 1 and 2, diagonal neighbouring interactions are introduced here, making for the most complete and arguably realistic simulation setting covered in this thesis. We find that the most performant model in terms of forecasting is GF-SPLASH(0, 1, λ), performing best in four out of six settings while the margins are minimal, especially for $T = 2000$. F-SPLASH topping the GF-SPLASH model only by a very small margin in the other two cases. In contrast to Design 1 and 2, we observe a steady decrease in RMSFE for increasing T.

We shift our focus to the estimation accuracy in Design 3 and find that the GF-SPLASH(0, 1, λ) performs best in five out of six configurations. This is in stark contrast with Design 1 and 2, where SPLASH edged out the other models quite considerably in terms of estimation accuracy. We conjecture that this might indicate that the earlier problem of the fused LASSO-based models not having a strong mechanism of promoting completely sparse diagonals is less pronounced in this case. Namely, due to Design 3 having a more dense spatial interaction matrix $\boldsymbol{A}$, because diagonally neighbouring interactions being included doubles the amount of non-zero diagonals in $\boldsymbol{A}$, as can be seen in Figure 1.

Table 1: Simulation results for Design 1

p	T	F-SPL(λ)	SSF-SPL($0.5, \lambda$)	GF-SPL($0.5, 0, \lambda$)	GF-SPL($0, 1, \lambda$)	GF-SPL($0.5, 1, \lambda$)	SPLASH($0, \lambda$)	SPLASH($0.5, \lambda$)	PVAR(λ)
RMSFE									
25	500	1.003	1.076	1.012	**1.002**	1.009	1.012	1.013	1.026
25	1000	**1.005**	1.082	1.01	1.006	1.007	1.006	1.007	1.016
25	2000	**1.001**	1.056	1.003	1.002	1.002	1.002	1.002	1.011
36	500	**1.004**	1.073	1.014	1.004	1.01	1.014	1.016	1.036
36	1000	**1.006**	1.07	1.01	1.006	1.009	1.010	1.012	1.023
36	2000	1.005	1.061	1.005	1.004	1.004	**1.003**	1.004	1.012
EE									
25	500	0.682	0.535	0.797	0.674	0.792	**0.409**	0.455	-
25	1000	0.612	0.424	0.784	0.605	0.778	**0.328**	0.369	-
25	2000	0.489	0.354	0.727	0.464	0.716	**0.238**	0.271	-
36	500	0.673	0.58	0.834	0.676	0.823	**0.560**	0.593	-
36	1000	0.62	0.463	0.837	0.626	0.828	**0.456**	0.502	-
36	2000	0.59	0.378	0.785	0.595	0.78	**0.341**	0.383	-

Note: Numbers in **bold** inidicate best performance for each combination of p and T

Note: Simulation results are based on $N_{\text{sim}} = 250$ Monte Carlo simulations

Table 2: Simulation results for Design 2

p	T	F-SPL(λ)	SSF-SPL($0.5, \lambda$)	GF-SPL($0.5, 0, \lambda$)	GF-SPL($0, 1, \lambda$)	GF-SPL($0.5, 1, \lambda$)	SPLASH($0, \lambda$)	SPLASH($0.5, \lambda$)	PVAR(λ)
RMSFE									
25	500	1.006	1.072	1.011	**1.004**	1.008	1.009	1.009	1.019
25	1000	1.006	1.049	1.006	1.005	**1.004**	1.006	1.007	1.013
25	2000	1.004	1.056	1.003	1.002	1.002	1.002	**1.002**	1.008
36	500	1.004	1.063	1.013	**1.002**	1.009	1.01	1.013	1.026
36	1000	1.008	1.054	1.009	**1.007**	1.007	1.008	1.009	1.016
36	2000	1.006	1.062	1.008	**1.005**	1.007	1.006	1.007	1.013
EE									
25	500	0.635	0.608	0.682	0.612	0.681	**0.381**	0.417	-
25	1000	0.558	0.585	0.695	0.533	0.694	**0.293**	0.331	-
25	2000	0.507	0.537	0.697	0.47	0.702	**0.220**	0.249	-
36	500	0.628	0.637	0.705	0.617	0.703	**0.502**	0.528	-
36	1000	0.584	0.627	0.716	0.569	0.716	**0.403**	0.426	-
36	2000	0.506	0.589	0.729	0.494	0.727	**0.304**	0.334	-

Note: Numbers in **bold** inidicate best performance for each combination of p and T

Note: Simulation results are based on $N_{\text{sim}} = 250$ Monte Carlo simulations

Table 3: Simulation results for Design 3

p	T	F-SPL(λ)	SSF-SPL($0.5, \lambda$)	GF-SPL($0.5, 0, \lambda$)	GF-SPL($0, 1, \lambda$)	GF-SPL($0.5, 1, \lambda$)	SPLASH($0, \lambda$)	SPLASH($0.5, \lambda$)	PVAR(λ)
RMSFE									
25	500	1.009	1.12	1.018	**1.009**	1.019	1.016	1.016	1.03
25	1000	**1.003**	1.133	1.008	1.004	1.009	1.01	1.011	1.024
25	2000	**1.002**	1.147	1.004	1.003	1.002	1.005	1.005	1.009
36	500	1.009	1.08	1.018	**1.007**	1.014	1.015	1.018	1.035
36	1000	1.006	1.076	1.009	**1.006**	1.008	1.007	1.008	1.023
36	2000	1.005	1.076	1.007	**1.005**	1.006	1.006	1.007	1.014
EE									
25	500	0.452	0.528	0.815	**0.425**	0.78	0.516	0.564	-
25	1000	0.451	0.626	0.682	**0.423**	0.621	0.465	0.51	-
25	2000	0.486	0.687	0.552	0.456	0.471	**0.419**	0.453	-
36	500	0.476	0.503	0.934	**0.462**	0.911	0.542	0.577	-
36	1000	0.433	0.455	0.815	**0.395**	0.792	0.474	0.513	-
36	2000	0.44	0.525	0.66	**0.386**	0.59	0.409	0.448	-

Note: Numbers in **bold** inidicate best performance for each combination of p and T

Note: Simulation results are based on $N_{\text{sim}} = 250$ Monte Carlo simulations

6 Discussion

In the examination of the three simulation designs, we observed various noteworthy outcomes and patterns which contribute to the ongoing discourse on spatio-temporal modelling, particularly with respect to the SPatial LAsso-type SHrinkage (SPLASH) estimator (Reuvers & Wijler, 2021). A comparison of our novel F-SPLASH, SSF-SPLASH, and GF-SPLASH models with the SPLASH estimator revealed slightly superior forecasting performance for the F-SPLASH and GF-SPLASH(0, 1, λ) models, particularly for smaller sample sizes. These findings suggest potentially improved convergence rates for the aforementioned estimators compared to SPLASH. The results also demonstrated that estimators employing a solely spatial penalty structure generally surpassed those with penalties on individual coefficients. Except for the final simulation, the SPLASH models documented superior estimation accuracy over our novel methods.

A crucial observation is that while the SPLASH models often provided better estimation accuracy, they often fell short in terms of forecasting performance compared to their F-SPLASH and GF-SPLASH counterparts, particularly in scenarios with smaller sample sizes. However, SPLASH(0, λ) showed a consistent increase in forecasting performance as T increased. This indicates that SPLASH may prove more effective in larger datasets due to the enhanced estimation accuracy accompanied by improved forecasting performance that emerges as the sample size expands.

One particular shortcoming of our proposed fused LASSO-based models is the lack of explicit encouragement for complete diagonal sparsity, a feature inherent in SPLASH via the group LASSO. Instead, the novel methods provide smooth estimates of the coefficients along the diagonal, however, the resulting estimates are not sparse. This causes the model to fill the entire allowed bandwidth with non-zero values, although their magnitude can become very small. We attempted to address this by creating the SSF-SPLASH estimator, which stimulates sparse diagonals indirectly. However, the performance of SSF-SPLASH in the simulation experiments was underwhelming, and the coefficient estimates showed to be quite unstable. Subsequently, we performed some tests to combine the group and fused LASSO in a single estimator in an extension that is similar to SSF-SPLASH. Unfortunately, the preliminary results of this estimator showed similar shortcomings as the SSF-SPLASH estimator. However, we believe this concept has merit and warrants further exploration. An effective combination or balance of these two principles could potentially yield excellent results, where the great estimation accuracy seen in the SPLASH estimators is unified with the apparent advantages of our novel fused LASSO-based methods.

We recognize the limitations in the scope of the GF-SPLASH estimator, specifically situations that do not align with the strict spatial assumptions made in this thesis. Nevertheless, our work demonstrates how specific domain knowledge can be incorporated into the penalty structure using a graph-guided approach to impose a domain-specific penalty structure. This can result in tailored estimators that yield improved performance in specialized use cases. While the specific implementation in this thesis may not always be the optimal solution, the improved forecasting performance of the fused LASSO-based model in this spatial setting shows that our structured approach can serve as an effective framework

to build upon.

7 Conclusion

The purpose of this thesis was to examine the potential of graph-guided fused LASSO-based methods for enhancing the methodology laid out in the SPatial LAsso-type SHrinkage (SPLASH) estimator for spatio-temporal modelling (Reuvers & Wijler, 2021). The study built upon the foundation of SPLASH and corresponding generalized Yule-Walker estimation. We proposed three novel spatio-temporal estimators: GF-SPLASH, F-SPLASH, and SSF-SPLASH. These novel models sought to evaluate the potential of stimulating smoothness of spatially related coefficients, possibly enhancing estimation and forecasting performance.

Our methodology involved the integration of the graph-guided fused LASSO framework with the generalized Yule-Walker estimation procedure as used in SPLASH. We constructed graphs that capture the assumed spatial relationships among the coefficients and introduce hyperparameters that control how strongly the spatial relationships are stimulated. These graphs were used to construct the general form of our spatio-temporal estimator in GF-SPLASH. Next, we derived simplifications for specific cases of the GF-SPLASH estimator that could be estimated more efficiently, resulting in the F-SPLASH and SSF-SPLASH estimators.

Our findings revealed that the purely spatial model configurations F-SPLASH(λ) and GF-SPLASH(0, 1, λ) demonstrated superior forecasting performance compared to the original SPLASH estimator, especially in scenarios with smaller sample sizes. This suggests a faster convergence rate for these novel fused LASSO-based models, which can be highly beneficial in practical situations where the availability of larger datasets might be a constraint. However, it is essential to note that the original SPLASH models still provided superior estimation accuracy in most cases and that the forecasting performance converged to a similar level of F-SPLASH and GF-SPLASH for larger sample sizes.

This thesis contributes valuable insights to the field of spatio-temporal modelling by introducing and evaluating the performance of novel fused LASSO-based methods. Our results highlight the potential benefits of these methods, particularly in scenarios with smaller sample sizes, while also emphasizing the value of developing specialized models that incorporate domain knowledge using the graph-guided fused LASSO. The novel methods proposed in this study can serve as a basis for further research and refinement, with the potential to significantly enhance the capabilities of spatio-temporal modelling in various practical applications.

References

Arnold, T., & Tibshirani, R. (2014, November 3). Efficient implementations of the generalized lasso dual path algorithm. Retrieved March 26, 2023, from http://arxiv.org/abs/1405.3222

Aswi, A., Cramb, S. M., Moraga, P., & Mengersen, K. (2019). Bayesian spatial and spatio-temporal approaches to modelling dengue fever: A systematic review. *Epidemiology & Infection*, *147*, e33. https://doi.org/10.1017/S0950268818002807

Bergmeir, C., & Benítez, J. M. (2012). On the use of cross-validation for time series predictor evaluation. *Information Sciences*, *191*, 192–213. https://doi.org/10.1016/j.ins.2011.12.028

Boyd, S., Parikh, N., Chu, E., Peleato, B., & Eckstein, J. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, *3*(1), 1–122. https://doi.org/10.1561/2200000016

Brockwell, P. J., & Davis, R. A. (1991). *Time series: Theory and methods*. Springer. https://doi.org/10.1007/978-1-4419-0320-4

Chen, X., Kim, S., Lin, Q., Carbonell, J. G., & Xing, E. P. (2010, May 19). Graph-structured multi-task regression and an efficient optimization method for general fused lasso. https://doi.org/10.48550/arXiv.1005.3579

Davis, T. A., & Duff, I. S. (1997). An unsymmetric-pattern multifrontal method for sparse LU factorization. *SIAM Journal on Matrix Analysis and Applications*, *18*(1), 140–158. https://doi.org/10.1137/S0895479894246905

Dou, B., Parrella, M. L., & Yao, Q. (2016). Generalized yule–walker estimation for spatio-temporal models with unknown diagonal coefficients. *Journal of Econometrics*, *194*(2), 369–382. https://doi.org/10.1016/j.jeconom.2016.05.014

Duan, J., Soussen, C., Brie, D., Idier, J., Wan, M., & Wang, Y.-P. (2016). Generalized LASSO with under-determined regularization matrices. *Signal processing*, *127*, 239–246. https://doi.org/10.1016/j.sigpro.2016.03.001

Efron, B., Hastie, T., Johnstone, I., & Tibshirani, R. (2004). Least angle regression. *The Annals of Statistics*, *32*(2), 407–499. https://doi.org/10.1214/009053604000000067

Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, *33*(1), 1–22. Retrieved May 31, 2023, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2929880/

Gao, Z., Ma, Y., Wang, H., & Yao, Q. (2019). Banded spatio-temporal autoregressions. *Journal of Econometrics*, *208*(1), 211–230. https://doi.org/10.1016/j.jeconom.2018.09.012

Gross, J. L., & Yellen, J. (2005). *Graph theory and its applications*. CRC press.

Guo, S., Wang, Y., & Yao, Q. (2016). High-dimensional and banded vector autoregressions. *Biometrika*, *103*(4), 889–903. https://doi.org/10.1093/biomet/asw046

Hafner, C. M. (2020). The spread of the covid-19 pandemic in time and space. *International Journal of Environmental Research and Public Health*, *17*(11), 3827. https://doi.org/10.3390/ijerph17113827

Hamilton, J. D. (D., & Hamilton, J. D. (1994). *Time series analysis*. Princeton University Press. Retrieved June 9, 2023, from http://catdir.loc.gov/catdir/description/prin021/93004958.html

Kakamu, K., Polasek, W., & Wago, H. (2008). Spatial interaction of crime incidents in japan. *Mathematics and Computers in Simulation*, *78*(2), 276–282. https://doi.org/10.1016/j.matcom.2008.01.019

Kelejian, H. H., & Prucha, I. R. (1998). A generalized spatial two-stage least squares procedure for estimating a spatial autoregressive model with autoregressive disturbances. *The Journal of Real Estate Finance and Economics*, *17*(1), 99–121. https://doi.org/10.1023/A:1007707430416

Lee, L.-f., & Yu, J. (2010). A SPATIAL DYNAMIC PANEL DATA MODEL WITH BOTH TIME AND INDIVIDUAL FIXED EFFECTS. *Econometric Theory*, *26*(2), 564–597. https://doi.org/10.1017/S0266466609100099

Ma, Y., Guo, S., & Wang, H. (2023). Sparse spatio-temporal autoregressions by profiling and bagging. *Journal of Econometrics*, *232*(1), 132–147. https://doi.org/10.1016/j.jeconom.2020.10.010

Nicholson, W., Matteson, D., & Bien, J. (2017a, February 22). BigVAR: Tools for modeling sparse high-dimensional multivariate time series. https://doi.org/10.48550/arXiv.1702.07094

Nicholson, W. B., Matteson, D. S., & Bien, J. (2017b). VARX-l: Structured regularization for large vector autoregressions with exogenous variables. *International Journal of Forecasting*, *33*(3), 627–651. https://doi.org/10.1016/j.ijforecast.2017.01.003

Qu, X., & Lee, L.-f. (2015). Estimating a spatial autoregressive model with an endogenous spatial weight matrix. *Journal of Econometrics*, *184*(2), 209–232. https://doi.org/10.1016/j.jeconom.2014.08.008

Qu, X., Lee, L.-f., & Yu, J. (2017). QML estimation of spatial dynamic panel data models with endogenous time varying spatial weights matrices. *Journal of Econometrics*, *197*(2), 173–201. https://doi.org/10.1016/j.jeconom.2016.11.004

Reuvers, H., & Wijler, E. (2021, December 30). Sparse generalized yule-walker estimation for large spatio-temporal autoregressions with an application to NO2 satellite data. https://doi.org/10.48550/arXiv.2108.02864

She, Y. (2010). Sparse regression with exact clustering. *Electronic Journal of Statistics*, *4*. https://doi.org/10.1214/10-EJS578

Simon, N., Friedman, J., Hastie, T., & Tibshirani, R. (2013). A sparse-group lasso. *Journal of Computational and Graphical Statistics*, *22*(2), 231–245. Retrieved March 16, 2023, from https://www.jstor.org/stable/43304828

Thorson, J. T. (2019). Guidance for decisions using the vector autoregressive spatio-temporal (VAST) package in stock, ecosystem, habitat and climate assessments. *Fisheries Research*, *210*, 143–161. https://doi.org/10.1016/j.fishres.2018.10.013

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, *58*(1), 267–288. https://doi.org/10.1111/j.2517-6161.1996.tb02080.x

Tibshirani, R., Saunders, M., Rosset, S., Zhu, J., & Knight, K. (2005). Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *67*(1), 91–108. https://doi.org/10.1111/j.1467-9868.2005.00490.x

Tibshirani, R. J., & Taylor, J. (2011). The solution path of the generalized lasso. *The Annals of Statistics*, *39*(3), 1335–1371. https://doi.org/10.1214/11-AOS878

Vatsavai, R. R., Ganguly, A., Chandola, V., Stefanidis, A., Klasky, S., & Shekhar, S. (2012). Spatiotemporal data mining in the era of big spatial data: Algorithms and applications. *Proceedings of the 1st ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data*, 1–10. https://doi.org/10.1145/2447481.2447482

Wright, S. J. (2015, February 16). Coordinate descent algorithms. https://doi.org/10.48550/arXiv.1502.04759

Xin, B., Kawahara, Y., Wang, Y., Hu, L., & Gao, W. (2016). Efficient generalized fused lasso and its applications. *ACM Transactions on Intelligent Systems and Technology*, *7*(4), 60:1–60:22. https://doi.org/10.1145/2847421

Yu, H., Li, J., Bardin, S., Gu, H., & Fan, C. (2021). Spatiotemporal dynamic of COVID-19 diffusion in china: A dynamic spatial autoregressive model analysis. *ISPRS International Journal of Geo-Information*, *10*(8), 510. https://doi.org/10.3390/ijgi10080510

Zhu, Y. (2017). An augmented ADMM algorithm with application to the generalized lasso problem. *Journal of Computational and Graphical Statistics*, *26*(1), 195–204. https://doi.org/10.1080/10618600.2015.1114491

Appendices

A Derivations

Number of elements in $\boldsymbol{\phi}$. In equation (43), we state the formula for the number of elements in $\boldsymbol{\phi}$, which is determined by the dimension of the problem p, and the bandwidth k_0. To derive the result, we sum the number non-zero elements that $\boldsymbol{A}$ and $\boldsymbol{B}$ can have per row, starting at the first. There are a total of $p - 2k_0$ rows containing an entry from each of the $4k_0 + 1$ non-zero diagonals, totalling $(p - 2k_0)(4k_0 + 1)$ elements. The top and bottom k_0 rows contain fewer rows, as some diagonals are cut off at that point. The top k_0 rows contain

$$\sum_{i=0}^{k_0-1} 2k_0 + 1 + 2i = k_0(2k_0 + 1) + 2 \cdot \frac{(k_0 - 1)k_0}{2} = 3k_0^2 \tag{76}$$

elements. Since the top and bottom k_0 contain equal amount of elements, the total number of non-zero elements is

$$k = (p - 2k_0)(4k_0 + 1) + 6k_0^2 = p(4k_0 + 1) - 2(k_0^2 + k_0). \tag{77}$$

B Figures

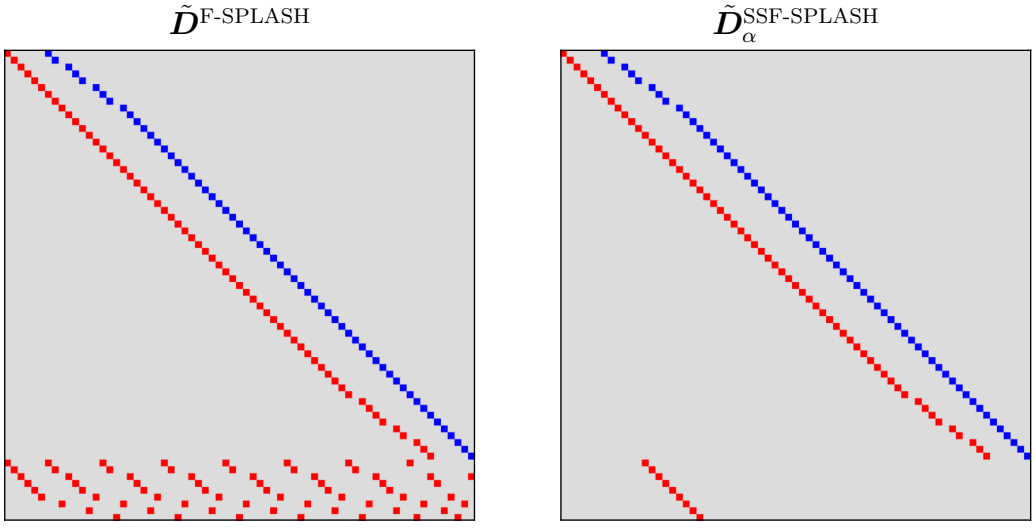

Figure 7: Penalty matrices for the estimators F-SPLASH(λ) (left) and SSF-SPLASH(α, λ) (right), pertaining to a cross-sectional dimension $p = 9$. Red and blue cells correspond to positive and negative values, respectively. The lower part of both matrices are the extensions used to ensure invertibility, as per equations (57) and (66).

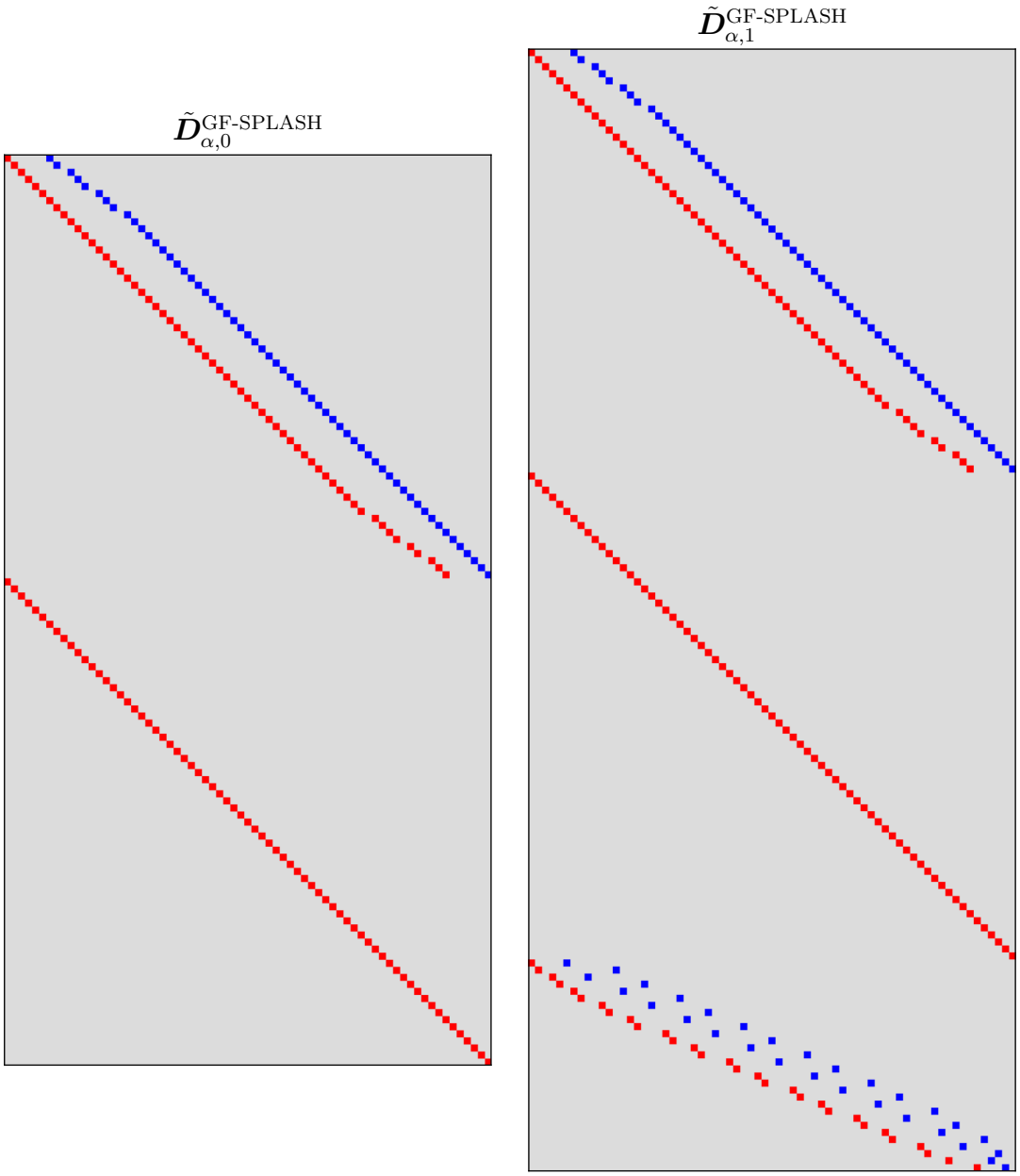

Figure 8: Penalty matrices for the GF-SPLASH(α, s, λ) estimator for a cross-sectional dimension $p = 9$. The left matrix displays the penalty matrix without a symmetric penalty $(s = 0)$ and the right matrix displays the penalty matrix with the symmetric penalty $(s = 1)$. Red and blue cells correspond to positive and negative values, respectively. Note that the elements of the matrices would be scaled accordingly for varying α, as per equation (50).

C Pseudo-algorithms

On the following pages, the *pseudo-algorithms* are given for the novel estimators developed in this thesis. Namely, the GF-SPLASH(α, s, λ), F-SPLASH(λ), and SSF-SPLASH(α, λ). These algorithms include steps for performing optional hyperparameter optimization using cross-validation. However, these steps can be discarded and replaced by alternative and

possibly better hyperparameter optimization techniques.

Algorithm 1 GF-SPLASH(α, λ, s)

Compute $\hat{\boldsymbol{\Sigma}}_0 = \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{y}_t \boldsymbol{y}_t'$ and $\hat{\boldsymbol{\Sigma}}_1 = \frac{1}{T}\sum_{t=2}^{T} \boldsymbol{y}_t \boldsymbol{y}_{t-1}'$.
if bandwidths h = $\{h_0, h_1\}$ are unknown **then**
    Compute bandwidth using $h_j = \arg\min_h H_j^*(h)$ for $j \in \{0, 1\}$ as per equation (17).
end if
Define graph $\mathcal{G}$ and corresponding oriented incidence matrix $\boldsymbol{D}^{(\mathcal{G})}$ as per section 3.1.
if s = 1 **then**
    Define graph $\mathcal{G}_s$ and corresponding oriented incidence matrix $\boldsymbol{D}^{(\mathcal{G}_s)}$ as per section 3.1.
end if
if $\hat{\alpha}$ or $\hat{\lambda}$ are unknown **then**
    if $\hat{\alpha}$ is unknown **then**
        Define parameter grid Ω_α, e.g., $\Omega_\alpha = \{0, 0.1, \ldots, 1\}$.
        Construct $\boldsymbol{D}_{\alpha,s}$ for each $\alpha \in \Omega_\alpha$ as per equation (50).
    else
        Set $\Omega_\alpha = \{\hat{\alpha}\}$.
    end if
    Select a forecast horizon q to evaluate the performance on.
    Generate K time-series cross-validation folds as described in section 3.4.
    for $k \in \{1, \ldots, K\}$ **do**
        Compute $\hat{\boldsymbol{V}}_h^{(d)}$ and $\hat{\boldsymbol{\sigma}}_h$ as in section (2.2.1), using the training data in the kth fold.
        if $\hat{\lambda}$ is unknown **then**
            Calculate one iteration of *genlasso* estimating $\hat{\boldsymbol{\phi}}(\alpha, \lambda, s)$ for all $\alpha \in \Omega_\alpha$.
            Save λ_0^α for all $\alpha \in \Omega_\alpha$ from the first iteration of *genlasso*.
            Define parameter grid $\Omega_{\lambda,\alpha} = \{\lambda_0^\alpha, \ldots, \lambda_{M-1}^\alpha\}$ for all $\alpha \in \Omega_\alpha$, with

$$\log_{10}(\lambda_i^\alpha) = \log_{10}(\lambda_0^\alpha) - \frac{i}{M-1}\log_{10}(r) \quad \text{for all } i \in \{0, \ldots, M-1\},$$

            where r is a small multiplier, e.g., $r = 10^{-4}$.
        else
            Set $\Omega_{\lambda,\alpha} = \{\hat{\lambda}\}$.
        end if
        for each $\lambda_i^\alpha \in \Omega_{\lambda,\alpha}$ and each $\alpha_j \in \Omega_\alpha$ **do**
            Compute $\hat{\boldsymbol{\phi}}(\alpha_j, \lambda_i^\alpha, s)$ using the Augmented ADMM algorithm (Zhu, 2017).
            Compute $\hat{\boldsymbol{y}}_{T+q}$, the q-step ahead predictions, using $\hat{\boldsymbol{\phi}}(\alpha_j, \lambda_i^\alpha, s)$.
            Compute and save $\text{MSFE}_{i,j,k}^q = \left\|\boldsymbol{y}_{T+q} - \hat{\boldsymbol{y}}_{T+q}\right\|_2^2$.
        end for
    end for
    Select $(\hat{\alpha}, \hat{\lambda}) = (\alpha_j, \lambda_i^\alpha)$ as the parameters yielding the lowest $\frac{1}{K}\sum_{k=1}^{K} \text{MSFE}_{i,j,k}^q$.
end if
Construct $\boldsymbol{D}_{\hat{\alpha},s}$ as per equation (50).
Compute $\hat{\boldsymbol{V}}_h^{(d)}$ and $\hat{\boldsymbol{\sigma}}_h$ as in section 2.2.1, using all training data.
Compute $\hat{\boldsymbol{\phi}}(\hat{\alpha}, \hat{\lambda}, s)$ using the Augmented ADMM algorithm (Zhu, 2017).

Algorithm 2 F-SPLASH(λ)

Compute $\hat{\boldsymbol{\Sigma}}_0 = \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{y}_t \boldsymbol{y}_t'$ and $\hat{\boldsymbol{\Sigma}}_1 = \frac{1}{T}\sum_{t=2}^{T} \boldsymbol{y}_t \boldsymbol{y}_{t-1}'$.
if bandwidths h $= \{h_0, h_1\}$ are unknown **then**
    Compute bandwidth using $h_j = \arg\min_h H_j^*(h)$ for $j \in \{0, 1\}$ as per equation (17).
end if
Define graph $\mathcal{G}$ and corresponding oriented incidence matrix $\boldsymbol{D}^{(\mathcal{G})}$ as per section 3.1.
Construct $\tilde{\boldsymbol{D}}$ as per equation (57), and calculate its inverse $\tilde{\boldsymbol{D}}^{-1}$.
if $\hat{\lambda}$ is unknown **then**
    Select a forecast horizon q to evaluate the performance on.
    Generate K time-series cross-validation folds as described in section 3.4
    for $k \in \{1, \ldots, K\}$ **do**
        Compute $\hat{\boldsymbol{V}}_h^{(d)}$ and $\hat{\boldsymbol{\sigma}}_h$ as in section (2.2.1), using the training data in the kth fold.
        Compute the matrix partitions $\tilde{\boldsymbol{X}}_1$ and $\tilde{\boldsymbol{X}}_2$ from $\hat{\boldsymbol{V}}_h^{(d)}\tilde{\boldsymbol{D}}^{-1}$ as per equation (61).
        As per equation (71), calculate

$$\lambda_0 = \frac{1}{p^2}\left\| \left(\boldsymbol{M}\tilde{\boldsymbol{X}}_1\right)' \boldsymbol{M}\hat{\boldsymbol{\sigma}}_h \right\|_\infty .$$

        Define parameter grid $\Omega_\lambda = \{\lambda_0, \ldots, \lambda_{M-1}\}$, with

$$\log_{10}(\lambda_i) = \log_{10}(\lambda_0) - \frac{i}{M-1}\log_{10}(r) \quad \text{for all } i \in \{0, \ldots, M-1\},$$

        where r is a small multiplier, e.g., $r = 10^{-4}$.
        for each $\lambda_i \in \Omega_\lambda$ **do**
            Compute $\hat{\boldsymbol{\theta}}_1$ using the CCD algorithm (Friedman et al., 2010) as per equation (62).
            Compute $\hat{\boldsymbol{\theta}}_2$ with

$$\hat{\boldsymbol{\theta}}_2 = (\tilde{\boldsymbol{X}}_2'\tilde{\boldsymbol{X}}_2)^{-1}\tilde{\boldsymbol{X}}_2'(\hat{\boldsymbol{\sigma}}_h - \tilde{\boldsymbol{X}}_1\hat{\boldsymbol{\theta}}_1).$$

            Define $\hat{\boldsymbol{\theta}} = (\hat{\boldsymbol{\theta}}_1, \hat{\boldsymbol{\theta}}_1)$ and compute $\hat{\boldsymbol{\phi}}(\lambda_i) = \tilde{\boldsymbol{D}}^{-1}\hat{\boldsymbol{\theta}}$.
            Compute $\hat{\boldsymbol{y}}_{T+q}$, the q-step ahead predictions, using $\hat{\boldsymbol{\phi}}(\lambda_i)$.
            Compute and save $\text{MSFE}_{i,k}^q = \left\|\boldsymbol{y}_{T+q} - \hat{\boldsymbol{y}}_{T+q}\right\|_2^2$.
        end for
    end for
    Select $\hat{\lambda} = \lambda_i$ with $i = \arg\min_i \frac{1}{K}\sum_{k=1}^{K} \text{MSFE}_{i,k}^q$.
end if
Compute $\hat{\boldsymbol{V}}_h^{(d)}$ and $\hat{\boldsymbol{\sigma}}_h$ as in section 2.2.1, using all training data.
Compute the matrix partitions $\tilde{\boldsymbol{X}}_1$ and $\tilde{\boldsymbol{X}}_2$ from $\hat{\boldsymbol{V}}_h^{(d)}\tilde{\boldsymbol{D}}^{-1}$ as per equation (61).
Compute $\hat{\boldsymbol{\phi}}(\hat{\lambda})$ by executing steps (18)-(20) using $\hat{\lambda}$ in place of λ_i.

Algorithm 3 SSF-SPLASH(α, λ)

Compute $\hat{\boldsymbol{\Sigma}}_0 = \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{y}_t \boldsymbol{y}_t'$ and $\hat{\boldsymbol{\Sigma}}_1 = \frac{1}{T}\sum_{t=2}^{T} \boldsymbol{y}_t \boldsymbol{y}_{t-1}'$.
if bandwidths h $= \{h_0, h_1\}$ are unknown **then**
    Compute bandwidth using $h_j = \arg\min_h H_j^*(h)$ for $j \in \{0, 1\}$ as per equation (17).
end if
Define graph $\mathcal{G}$ and corresponding oriented incidence matrix $\boldsymbol{D}^{(\mathcal{G})}$ as per section 3.1.
if $\hat{\alpha}$ or $\hat{\lambda}$ is unknown **then**
    if $\hat{\alpha}$ is unknown **then**
        Define parameter grid Ω_α, e.g., $\Omega_\alpha = \{0, 0.1, \ldots, 1\}$.
        Construct $\tilde{\boldsymbol{D}}_\alpha$ for each $\alpha \in \Omega_\alpha$ as per equation (66) and calculate $\tilde{\boldsymbol{D}}_\alpha^{-1}$.
    else
        Set $\Omega_\alpha = \{\hat{\alpha}\}$.
    end if
    Select a forecast horizon q to evaluate the performance on.
    Generate K time-series cross-validation folds as described in section 3.4.
    for $k \in \{1, \ldots, K\}$ **do**
        Compute $\hat{\boldsymbol{V}}_h^{(d)}$ and $\hat{\boldsymbol{\sigma}}_h$ as in section (2.2.1), using the training data in the kth fold.
        if $\hat{\lambda}$ is unknown **then**
            Calculate

$$\lambda_0 = \frac{1}{p^2} \left\| \left(\hat{\boldsymbol{V}}_h^{(d)} \tilde{\boldsymbol{D}}_\alpha^{-1} \right)' \hat{\boldsymbol{\sigma}}_h \right\|_\infty .$$

            Define parameter grid $\Omega_{\lambda,\alpha} = \{\lambda_0^\alpha, \ldots, \lambda_{M-1}^\alpha\}$ for all $\alpha \in \Omega_\alpha$, with

$$\log_{10}(\lambda_i^\alpha) = \log_{10}(\lambda_0^\alpha) - \frac{i}{M-1} \log_{10}(r) \quad \text{for all } i \in \{0, \ldots, M-1\},$$

            where r is a small multiplier, e.g., $r = 10^{-4}$.
        else
            Set $\Omega_{\lambda,\alpha} = \{\hat{\lambda}\}$.
        end if
        for each $\lambda_i^\alpha \in \Omega_{\lambda,\alpha}$ and each $\alpha_j \in \Omega_\alpha$ **do**
            Compute $\hat{\boldsymbol{\theta}}(\alpha_j, \lambda_i^\alpha)$ using the CCD algorithm (Friedman et al., 2010).
            Compute $\hat{\boldsymbol{\phi}}(\alpha_j, \lambda_i^\alpha) = \tilde{\boldsymbol{D}}_{\alpha_j}^{-1} \hat{\boldsymbol{\theta}}(\alpha_j, \lambda_i^\alpha)$.
            Compute $\hat{\boldsymbol{y}}_{T+q}$, the q-step ahead predictions, using $\hat{\boldsymbol{\phi}}(\alpha_j, \lambda_i^\alpha)$.
            Compute and save $\text{MSFE}_{i,j,k}^q = \left\| \boldsymbol{y}_{T+q} - \hat{\boldsymbol{y}}_{T+q} \right\|_2^2$.
        end for
    end for
    Select $(\hat{\alpha}, \hat{\lambda}) = (\alpha_j, \lambda_i^\alpha)$ as the parameters yielding the lowest $\frac{1}{K}\sum_{k=1}^{K} \text{MSFE}_{i,j,k}^q$.
end if
Compute $\hat{\boldsymbol{V}}_h^{(d)}$ and $\hat{\boldsymbol{\sigma}}_h$ as in section 2.2.1, using all training data.
Compute $\hat{\boldsymbol{\phi}}(\hat{\alpha}, \hat{\lambda}, s)$ by executing steps (26) and (27) for $\hat{\alpha}$ and $\hat{\lambda}$.