Operator-valued Kernel-based Vector Autoregressive Models for Network Inference

Néhémy Lim
IBISC EA 4526, Université d'Évry-Val d'Essonne,
23 Bd de France, 91000,Évry, France,
and CEA, LIST, 91191 Gif-sur-Yvette CEDEX, France,

Florence d'Alché-Buc
INRIA-Saclay, LRI umr CNRS 8623, Université Paris Sud, France
and IBISC EA 4526, Université d'Évry-Val d'Essonne

Cédric Auliac
CEA, LIST, 91191 Gif-sur-Yvette CEDEX, France,

George Michailidis
Department of Statistics, University of Michigan,
Ann Arbor, MI 48109-1107

February 14, 2014

Abstract

Reverse-modeling of dynamical systems from time-course data still remains a challenging and canonical problem in knowledge discovery. For this learning task, a number of approaches primarily based on sparse linear models or Granger causality have been proposed in the literature. However when the dynamics are nonlinear, there does not exist a systematic answer that takes into account the nature of the underlying system. We introduce a novel family of vector autoregressive models based on different operator-valued kernels to identify the dynamical system and retrieve the target network. As in the linear case, a key issue is to control the model's sparsity. This control is performed through the joint learning of the structure of the kernel and the basis vectors. To solve this learning task, we propose an alternating optimization algorithm based on proximal gradient procedures that learn both the structure of the kernel and the basis vectors. Results on the DREAM3 competition gene regulatory benchmark networks of size 10 and 100 show the new model outperforms existing methods.

1 Introduction

In many scientific problems, *high dimensional data* with *network structure* play a key role in knowledge discovery (Kolaczyk, 2009). For example, recent advances in high throughput technologies have facilitated the simultaneous study of components of complex biological systems. Hence, molecular biologists are able to measure the expression levels of the entire genome and a good portion of the proteome and metabolome under different conditions and thus gain insight on how organisms respond to their environment. For this reason, reconstruction of gene regulatory networks from expression data has become a canonical problem in computational systems biology (Lawrence et al, 2010). Similar data structures emerge in other scientific domains. For instance, political scientists have focused on the analysis of roll call data of legislative bodies, since they allow them to study party cohesion and coalition formation through the underlying network reconstruction (Morton and Williams, 2010), while economists have focused on understanding companies creditworthiness or contagion (Gilchrist et al, 2009). Understanding climate changes implies to be able to predict the behavior of climate variables and their dependence relationship (Parry et al, 2007; Liu et al, 2010). Two classes of network inference problems have emerged simultaneously from all these fields: the inference of association networks that represent coupling between variables of interest (Meinshausen and Bühlmann, 2006; Kramer et al, 2009) and the inference of "causal" networks that describe how variables influence other ones (Murphy, 1998; Perrin et al, 2003; Auliac et al, 2008; Zou and Feng, 2009; Shojaie and Michailidis, 2010; Maathuis et al, 2010; Bolstad et al, 2011; Dondelinger et al, 2013; Chatterjee et al, 2012).

Over the last decade, a number of statistical techniques have been introduced for estimating networks from high-dimensional data in both cases. They divide into model-free and model-driven approaches. Model-free approaches for association networks directly estimate information-theoretic measures, such as mutual information to detect edges in the network (Hartemink, 2005; Margolin et al, 2006). Among model-driven approaches, graphical models have emerged as a powerful class of models and a lot of algorithmic and theoretical advances have occured for *static* (independent and identically distributed) data under the assumption of *sparsity*. For instance, Gaussian graphical models have been thoroughly studied (see Bühlmann and van de Geer (2011) and references therein) under different regularization schemes to reinforce sparsity in linear models in an unstructured or a structured way. In order to infer causal relationship networks, Bayesian networks (Friedman, 2004; Lèbre, 2009) have been developed either from static data or time-series within the framework of dynamical Bayesian networks. In the case of continuous variables, linear multivariate autoregressive modeling (Michailidis and d'Alché Buc, 2013) has been developed with again an important focus on sparse models. In this latter framework, Granger causality models have attracted an increasing interest to capture causal relationships.

However, little work has focused on network inference for continuous variables in the presence of nonlinear dynamics despite the fact that regulatory mechanisms involve such dynamics. Of special interest are approaches based on

parametric ordinary differential equations (Chou and Voit, 2009) that alternatively learn the structure of the model and its parameters. The most successful approaches decompose into Bayesian Learning (Mazur et al, 2009; Aijo and Lahdesmaki, 2009) that allows to deal with stochasticity of the biological data, while easily incorporating prior knowledge and genetic programming (Iba, 2008) that provides a population-based algorithm for a stochastic search in the structure space. In this study, we start from a regularization theory perspective and introduce a general framework for nonlinear multivariate modeling and network inference. Our aim is to extend the framework of sparse linear modeling to that of *sparse nonlinear modeling*. In the machine learning community, a powerful tool to extend linear models to nonlinear ones is based on kernels. The famous kernel trick allows to deal with nonlinear learning problems by working implicitly in a new feature space, where inner products can be computed using a symmetric positive semi-definite function of two variables, called a kernel. In particular, a given kernel allows to build a unique Reproducing Kernel Hilbert Space (RKHS), e.g. a functional space where regularized models can be defined from data using representer theorems. The RKHS theory provides a unified framework for many kernel-based models and a principled way to build new (nonlinear) models. Since multivariate time-series modeling requires defining vector-valued models, we propose to build on operator-valued kernels and their associated reproducing kernel Hilbert space theory (Senkene and Tempel'man, 1973) that were recently introduced in machine learning by Micchelli and Pontil (2005) for the multi-task learning problem with vector-valued functions. Among different ongoing research on the subject (Alvarez et al, 2011), new applications concern vector field regression (Baldassarre et al, 2010), structured classification (Dinuzzo and Fukumizu, 2011), functional regression (Kadri et al, 2011) and link prediction (Brouard et al, 2011). However, their use in the context of time series is novel.

Building upon our previous work (Lim et al, 2013) that focused on a specific model, we define a whole family of nonlinear vector autoregressive models based on various operator-valued kernels. Once an operator-valued kernel-based model is learnt, we compute an empirical estimate of its Jacobian, providing a generic and simple way to extract dependence relationship among variables. We discuss how a chosen operator-valued kernel can produce not only a good approximation of the systems dynamics, but also a flexible and controllable Jacobian estimate. To obtain sparse networks and get sparse Jacobian estimates, we extend the sparsity constraint applied to the design matrix regularly employed in linear modeling. To control smoothness of the model, the definition of the loss function involves an ℓ_2-norm penalty and additionally, may include two different kinds of penalties, depending on the nature of the estimation problem: an ℓ_1 penalty that imposes sparsity to the whole matrix of parameter vectors, suitable when the main goal is to ensure sparsity of the Jacobian, and a mixed ℓ_1/ℓ_2-norm that allows one to deal with an unfavorable ratio between the network size and the length of observed time-series. To optimize a loss function that contains these non-differentiable terms, we develop a general proximal gradient algorithm.

Note that selected operator-valued kernels involve a positive semi-definite

matrix as a hyperparameter. The background knowledge required for its definition is in general not available, especially in a network inference task. To address this kernel design task together with the other parameters, we introduce an efficient strategy that alternatively learns the parameter vectors and the positive semi-definite matrix that characterizes the kernel. This matrix plays an important role regarding the Jacobian sparsity; the estimation procedure for the matrix parameter also involves an ℓ_1 penalty and a positive semi-definiteness constraint. We show that without prior knowledge on the relationship between variables, the proposed algorithm is able to retrieve the network structure of a given underlying dynamical system from the observation of its behavior through time.

The structure of the paper is as follows: in Section 2, we present the general network inference scheme. In Section 3, we recall elements of RKHS theory devoted to vector-valued functions and introduce operator-valued kernel-based autoregressive models. Section 4 presents the learning algorithm that estimates both the parameters of the model and the parameters of the kernel. Section 5 illustrates the performance of the model and the algorithm through extensive numerical work based on both synthetic and real data, and comparison with state-of-the-art methods.

2 Network inference from nonlinear vector autoregressive models

Let $\mathbf{x}_t \in \mathbb{R}^d$ denote the *observed* state of a dynamical system comprising of d state variables. We are interested in inferring direct influences of a state variable j on other variables $i \neq j$, $(i,j) \in \{1, \cdots, d\}^2$. The set of influences among state variables is encoded by a *network* matrix $A = (a_{ij})$ of size $d \times d$ for which a coefficient $a_{ij} = 1$ if the state variable j influences the state variable i, 0 otherwise. Further, we assume that a first-order stationary model is adequate to capture the temporal evolution of the system under study, which can exhibit nonlinear dynamics captured by a function $h : \mathbb{R}^d \to \mathbb{R}^d$:

$$\mathbf{x}_{t+1} = h(\mathbf{x}_t) + \mathbf{u}_t \tag{1}$$

where $\mathbf{u}_t$ is a noise term.

Some models such as linear models $h(\mathbf{x}_t) = B\mathbf{x}_t$ or parametric models, explicitly involve a matrix that can be interpreted as a *network* matrix and its estimation (possibly sparse) can be directly accomplished. However, for nonlinear models this is a more involved task. Our strategy is to first learn h from the data and subsequently estimate A using the values of the instantaneous Jacobian matrix of model h, measured at each time point. The underlying idea is that partial derivatives $\frac{\partial h(\mathbf{x}_t)^i}{\partial x_t^j}$ reflects the influence of explanatory variable j at time t on the value of the i-th model's output $h(\mathbf{x}_t)^i$. We interpret that if $\frac{\partial h(\mathbf{x}_t)^i}{\partial x_t^j}$ is high in absolute value, then variable j influences variable i. Several

estimators of A can be built from those partial derivatives. We propose to use the empirical mean of the instantaneous Jacobian matrix of h. Specifically, denote by $\mathbf{x}_1, \ldots, \mathbf{x}_{N+1}$ the observed time series of the network state. Then, $\forall (i,j) \in \{1, \ldots, d\}^2$, an estimate $\hat{J}(h)$ of the Jacobian matrix $\nabla h = J(h)$ is given by:

$$\hat{J}(h)_{ij} = \frac{1}{N} \sum_{t=1}^{N} \frac{\partial h(\mathbf{x}_t)^i}{\partial x_t^j} \tag{2}$$

In the remainder of the paper, we note $\hat{J}(h)_{ij}$ the (i,j) coefficient of the empirical mean of Jacobian of h and $\hat{J}(h)_{ij}(t)$ its value at a given time t. Each coefficient $\hat{J}(h)_{ij}$ in absolute value gives a score to the potential influence of variable j on variable i. To provide a final estimate of A, these coefficients are sorted in increasing order and a rank matrix R is built according to the following rule: let $r(i,j)$ be the rank of the coefficient $\hat{J}(h)_{ij}$ among the $\frac{p(p+1)}{2}$ sorted coefficients, then we define R, the rank matrix as: $R_{ij} = r(i,j)$.
To get an estimate of A, coefficients of R are thresholded given a positive threshold θ: $\hat{A}_{ij} = 1$, if $(R_{ij} > \theta)$, and 0, otherwise.

Note that to obtain a high quality estimate of the network, we need a class of functions h whose Jacobian matrices can be controlled during learning in such a way that they could provide good continuous approximators of A. In this work, we propose a new class of nonparametric vector autoregressive models that exhibit such properties. Specifically, we introduce Operator-valued Kernel-based Vector AutoRegressive (OKVAR) models, that constitute a rich class as discussed in the next section.

3 Operator-valued kernels and vector autoregressive models

3.1 From scalar-valued kernel to operator-valued kernel models of autoregresssion

In order to solve the vector autoregression problem set in Eq. (1) with a nonlinear model, one option is to decompose it into d tasks and use, for each task i, a scalar-valued model h_i such as a kernel-based model. Dataset $\mathcal{D}_N$ now reduces into d datasets of the form $\mathcal{D}_N^i = \{(\mathbf{x}_\ell, x_{\ell+1}^i), \ell = 1, \ldots, N\}$. Each task i is now a regression task that we may solve by estimating a nonparametric model. For instance, kernel-based regression models are good candidates for those tasks. A unique feature of these approaches is that they can be derived from Reproducing Kernel Hilbert Space theory that offers a rigorous background for regularization. For instance, kernel-ridge regression and Support Vector Regression provide consistent estimator as soon as the chosen kernel is universal. Then, for each $i = 1, \ldots, d$, a model based on a positive definite kernel $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$

writes as:

$$h_i(\mathbf{x}_t) = \sum_{\ell=1}^{N} w_\ell^i k(\mathbf{x}_\ell, \mathbf{x}_t), \tag{3}$$

where $\mathbf{w}^i$ is the parameter vector attached to model i. Although this decomposition is well justified when the covariance matrix of noise $\mathbf{u}_t$ is diagonal, in the general case, the d regression tasks are not independent. The purpose of this work is therefore to extend such approaches to the vector autoregression problem in order to provide (i), a general family of nonparametric models and (ii), suitable models for network inference by Jacobian estimation. We now aim to predict the state vector of a dynamical system $\mathbf{x}_{t+1}$ at time $t+1$, given its state $\mathbf{x}_t$ at time t using kernel-based models appropriate for vectors. As a vector autoregressive model is a vector-valued function, the RKHS theory based on scalar-valued kernel does not apply. However, if the kernel is chosen to be operator-valued e.g. matrix-valued in our setting, then RKHS theory devoted to operator-valued kernel provides a similar framework to build models and to justify their use. In the following, we introduce the basic building blocks of operator-valued kernel-based theory and notations to extend (3) into models of the following form:

$$h(\mathbf{x}_t) = \sum_{\ell=1}^{N} K(\mathbf{x}_t, \mathbf{x}_\ell)\mathbf{w}_\ell, \tag{4}$$

where K is an operator-valued kernel to be defined in next section and $\mathbf{w}_\ell, \ell = 1, \ldots, N$ are parameter vectors of dimension d.

3.2 RKHS theory for vector-valued functions

In RKHS theory with operator-valued kernels, we consider functions with input in some set $\mathcal{X}$ and with vector values in some given Hilbert space $\mathcal{F}_y$. For completeness, we first describe the general framework and then come back to the case of interest, namely $\mathcal{X} = \mathcal{F}_y = \mathbb{R}^d$. Denote by $L(\mathcal{F}_y)$, the set of all bounded linear operators from $\mathcal{F}_y$ to itself. Given $A \in L(\mathcal{F}_y)$, A^* denotes its adjoint. Then, an operator-valued kernel K is defined as follows:

Definition 1 (Operator-valued kernel) *(Senkene and Tempel'man, 1973; Caponnetto et al, 2008)*
Let $\mathcal{X}$ be a set and $\mathcal{F}_y$ a Hilbert space. Then, $K : \mathcal{X} \times \mathcal{X} \to L(\mathcal{F}_y)$ is a kernel if:

- $\forall \ \ (\mathbf{x}, \mathbf{z}) \in \mathcal{X} \times \mathcal{X}$, $K(\mathbf{x}, \mathbf{z}) = K(\mathbf{z}, \mathbf{x})^*$
- $\forall m \in \mathbb{N}$, $\forall \{(\mathbf{x}_i, \mathbf{y}_i), i = 1, \ldots, m\} \subseteq \mathcal{X} \times \mathcal{F}_y$, $\sum_{i,j=1}^{m} \langle \mathbf{y}_i, K(\mathbf{x}_i, \mathbf{x}_j)\mathbf{y}_j \rangle_{\mathcal{F}_y} \geq 0$

When $\mathcal{F}_y = \mathbb{R}^d$, each kernel function is matrix-valued, which means that two inputs $\mathbf{x}$ and $\mathbf{z}$ can be compared in more details. Moreover, Senkene and Tempel'man (1973); Micchelli and Pontil (2005) established that one can build a

unique RKHS $\mathcal{H}_K$ from a given operator-valued kernel K. The RKHS $\mathcal{H}_K$ is built by taking the closure of $span\{K(\cdot,\mathbf{x})\mathbf{y}|\mathbf{x}\in\mathcal{X},\mathbf{y}\in\mathcal{F}_y\}$ endowed with the scalar product $\langle f,g\rangle_{\mathcal{H}_K}=\sum_{i,j}\langle\mathbf{u}_i,K(\mathbf{r}_i,\mathbf{s}_j)\mathbf{v}_j\rangle_{\mathcal{F}_y}$ with $f(\cdot)=\sum_i K(\cdot,\mathbf{r}_i)\mathbf{u}_i$ and $g(\cdot)=\sum_j K(\cdot,\mathbf{s}_j)\mathbf{v}_j$. The corresponding norm $\|\cdot\|_{\mathcal{H}_K}$ is defined by $\|f\|^2_{\mathcal{H}_K}=\langle f,f\rangle_{\mathcal{H}_K}$.

For the sake of notational simplicity, we omit K and use $\mathcal{H}=\mathcal{H}_K$ in the remainder of the paper. As in the scalar case, one of the most appealing features of RKHS is to provide a theoretical framework for regularization, e.g. representer theorems. Let us consider the case of regression with a convex loss function V. We denote by $\mathcal{D}_N=\{(\mathbf{x}_\ell,\mathbf{y}_\ell),\ell=1,\ldots,N\}\subseteq\mathcal{X}\times\mathcal{F}_y$ the data set under consideration.

Theorem 1 ((Micchelli and Pontil, 2005)) *Let V be some prescribed loss function, and $\lambda>0$ the regularization parameter. Then, any minimizer of the following optimization problem:*

$$\operatorname*{argmin}_{h\in\mathcal{H}}\mathcal{L}(h)=\sum_{\ell=1}^{N}V(h(\mathbf{x}_\ell),\mathbf{y}_\ell)+\lambda\|h\|^2_{\mathcal{H}}\ ,$$

admits an expansion:

$$\hat{h}(\cdot)=\sum_{\ell=1}^{N}K(\cdot,\mathbf{x}_\ell)\mathbf{c}_\ell\ , \tag{5}$$

where the coefficients $\mathbf{c}_\ell,\ell=\{1,\cdots,N\}$ are vectors in the Hilbert space $\mathcal{F}_y$.

Such a result justifies a new family of models of the form (5) for vector regression in $\mathbb{R}^d$. Then, the operator-valued kernel (OVK) becomes a matrix-valued one. In case this matrix is diagonal, the model reduces to d *independent* models with scalar outputs and there is no need for a matrix-valued kernel. In other cases, when we assume that the different components of the vector-valued function are not independent and may share some underlying structure, a non-diagonal matrix-valued kernel allows to take into consideration similarities between the components of the input vectors. Initial applications of matrix-valued kernels deal with structured output regression tasks, such as multi-task learning and structured classification. In the following, we propose to apply this framework to autoregression. We examine different matrix-valued kernels as well as different loss functions and discuss about their relevance for network inference.

3.3 The OKVAR family

Let now fix $\mathcal{X}=\mathcal{F}_y=\mathbb{R}^d$. Recall that the objective is to estimate an vector autoregressive model. Given the observed $d-$dimensional time series $\mathbf{x}_1,\ldots,\mathbf{x}_{N+1}$ that we use as the training dataset $\mathcal{D}_N=\{(\mathbf{x}_1,\mathbf{x}_2),\ldots,(\mathbf{x}_N,\mathbf{x}_{N+1})\}$, the non-parametric model h is defined as

$$h(\mathbf{x}_t;\mathcal{D}_N)=\sum_{\ell=1}^{N}K(\mathbf{x}_t,\mathbf{x}_\ell)\mathbf{c}_\ell \tag{6}$$

where $K(\cdot,\cdot)$ is a matrix-valued kernel and each $\mathbf{c}_\ell$ ($\ell \in \{1,\ldots,N\}$) is a vector of dimension d. In the following, we denote by $C \in \mathcal{M}^{N,d}$, the matrix composed of the N row vectors $\mathbf{c}_\ell^T$ of dimension d. We call Operator-valued Kernel Vector Autoregression (OKVAR), the vector autoregressive models of the form given by Eq. (6). In this study, we focus on nonlinear kernels by considering three kernel candidates, that fulfill the properties of an operator-valued kernel, one of them presenting the property to be universal.

Let us recall the definition of the scalar-valued Gaussian kernel $k_{Gauss} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$: $k_{Gauss}(\mathbf{x},\mathbf{z}) = \exp(-\gamma||\mathbf{x}-\mathbf{z}||^2)$. Please notice that in the special case $d = 1$, $k_{Gauss}(x,z)$ reduces to $\exp(-\gamma(x-z)^2)$.

As a baseline, we first consider the Gaussian transformable kernel which extends the standard Gaussian kernel to the matrix-valued case. If $\mathbf{x}$ is a vector, we denote x^m its m^{th} coordinate. Then the Gaussian transformable kernel is defined as follows:

Definition 2 (Gaussian (transformable) kernel)

$$\forall(\mathbf{x},\mathbf{z}) \in \mathbb{R}^d \times \mathbb{R}^d, \forall(i,j) \in \{1,\ldots,d\}^2, K_{Gauss}(\mathbf{x},\mathbf{z})_{ij} = k_{Gauss}(x^i, z^j) \quad (7)$$

Interestingly, each (i,j)-coefficient of the kernel K_{Gauss} compares the i^{th} coordinate of $\mathbf{x}$ to the j^{th} coordinate of $\mathbf{z}$, allowing a richer comparison between $\mathbf{x}$ and $\mathbf{z}$. For sake of simplicity, we will call this kernel the Gaussian kernel in the remainder of the paper. Note that the Gaussian kernel depends on only one single hyperparameter γ. It gives rise to the following Gaussian OKVAR model.

Definition 3 (Gaussian OKVAR)

$$h_{Gauss}(\mathbf{x}_t) = \sum_{\ell=1}^{N} K_{Gauss}(\mathbf{x}_t, \mathbf{x}_\ell)\mathbf{c}_\ell \quad (8)$$

An interesting feature of the Gaussian kernel-based OKVAR model is that each coordinate i of the vector model $h_{Gauss}(\mathbf{x}_t)^i$ can be expressed as a linear combination of nonlinear functions of variables $j = 1,\ldots,d$: $h_{Gauss}(\mathbf{x}_t)^i = \sum_\ell \sum_j \exp(-\gamma(x_t^i - x_\ell^j)^2)c_\ell^j$.

Decomposable kernels are another class of kernels that have been first defined by Micchelli and Pontil (2005) in order to address multi-task regression problems and structured classification. When based on Gaussian kernels, they are defined as follows:

Definition 4 (Decomposable (Gaussian) kernel) *Let B is a positive semi-definite matrix of size $d \times d$.*

$$\forall(\mathbf{x},\mathbf{z}) \in \mathbb{R}^d \times \mathbb{R}^d, K_{dec}(\mathbf{x},\mathbf{z}) = k_{Gauss}(\mathbf{x},\mathbf{z})B \quad (9)$$

In this kernel, B is related to the structure underlying outputs: B imposes that some outputs are dependent. This kernel has been proved to be universal by Caponnetto et al (2008), e.g. the induced RKHS is a family of universal approximators. The decomposable Gaussian OKVAR model is then defined as follows:

Definition 5 (Decomposable Gaussian OKVAR)

$$h_{dec}(\mathbf{x}_t) = \sum_{\ell=1}^{N} \exp(-\gamma||\mathbf{x}_t - \mathbf{x}_\ell||^2) B\mathbf{c}_\ell \tag{10}$$

Let now K_{dec} be a decomposable Gaussian kernel with scalar parameter γ_1 and matrix parameter B and K_{Gauss} be a Gaussian kernel with scalar parameter γ_2. As proposed in Lim et al (2013), we combine the Gaussian kernel and the decomposable kernel with the Hadamard product to get a kernel that involves nonlinear functions of single coordinates of the input vectors while imposing some structure to the kernel through a positive semi-definite matrix B. The resulting kernel is called the Hadamard kernel.

Definition 6 (Hadamard kernel)

$$\forall \ \ (\mathbf{x}, \mathbf{z}) \in \mathbb{R}^d \times \mathbb{R}^d, K_{Hadamard}(\mathbf{x}, \mathbf{z}) = K_{dec}(\mathbf{x}, \mathbf{z}) \circ K_{Gauss}(\mathbf{x}, \mathbf{z}) \tag{11}$$

where $\circ$ denotes the Hadamard product for matrices.

The resulting kernel $K_{Hadamard}$ possesses the kernel property, i.e:

Proposition 1 *The kernel defined by (11) is a matrix-valued kernel.*

Proof: A Hadamard product of two matrix-valued kernels is a matrix-valued kernel (proposition 4 in Caponnetto et al (2008)).

The Hadamard OKVAR model has the following form:

Definition 7 (Hadamard OKVAR)

$$h_{Hadamard}(\mathbf{x}_t) = \sum_{\ell=1}^{N} \exp(-\gamma_1||\mathbf{x}_t - \mathbf{x}_\ell||^2) B \circ K_{Gauss}(\mathbf{x}_t, \mathbf{x}_\ell)\mathbf{c}_\ell \tag{12}$$

3.4 Jacobians of the OKVAR models

As mentioned in the introduction, the network structure will be inferred by the empirical mean of the instantaneous Jacobian matrices $\hat{J}(h)(t)$ of h over observed time-points. At any given time point t, for a given target state variable i and a matrix-valued kernel-based model h, we have:

$$\forall j \in \{1, \ldots, d\}, \hat{J}(h)_{ij}(t) = \sum_{\ell=1}^{N} \frac{\partial (K(\mathbf{x}_t, \mathbf{x}_\ell)\mathbf{c}_\ell)^i}{\partial x_t^j} \tag{13}$$

Hence, each component of h should be a function of the state variables in such a way that the coefficients of the Jacobian reflect the dependence of the output component on some of the state variables. Due to our assumption of nonlinear dynamics of the underlying system, the kernel should contain nonlinear functions of the state variables. Moreover, a relevant matrix-valued kernel-based model should allow the sparsity of the Jacobian to be controlled through the values of its parameters. The kernels proposed previously in Section 3.3 give rise to the following expressions for instantaneous Jacobian.

Gaussian OKVAR. The (i,j)-th entry of the Jacobian at time t for the Gaussian-OKVAR model (8) writes as:

$$\hat{J}(h_{Gauss})_{ij}(t) = 2\gamma(x_t^i - x_t^j)\exp\left(-\gamma(x_t^i - x_t^j)^2\right)c_t^j,$$

which implies that the c_t^j's have the same impact, no matter what the target variable i is. As a consequence, it becomes impossible to control those parameters for network inference purposes.

Decomposable OKVAR. If we now consider the decomposable model, h_{dec}, defined in (10), the corresponding (i,j)-th term of the Jacobian is given by:

$$\hat{J}(h_{dec})_{ij}(t) = \sum_{\ell=1}^{N} \frac{\partial \exp(-\gamma||\mathbf{x}_t - \mathbf{x}_\ell||^2)}{\partial x_t^j}(B\mathbf{c}_\ell)^i \tag{14}$$

which implies that the nonlinear term involved in the matrix-valued kernel does not differ from one pair (i,j) to another. Then, it is impossible to control specific values of the Jacobian matrix using B or the $\mathbf{c}_\ell$'s.

Hadamard OKVAR. Finally, we obtain a richer class of Jacobians, more suitable for the inference task at hand, if we use Hadamard OKVAR defined in (12) for which the entries of its Jacobian at time t $\hat{J}(h_{Hadamard})_{ij}(t) = \hat{J}_{ij}(t)$ are given by:

$\hat{J}_{ij}(t) = \sum_{\ell=1}^{N} \frac{\partial \exp(-\gamma_1||\mathbf{x}_t - \mathbf{x}_\ell||^2)}{\partial x_t^j} B \circ K_{Gauss}(\mathbf{x}_t, \mathbf{x}_\ell)\mathbf{c}_\ell + \exp(-\gamma_1||\mathbf{x}_t - \mathbf{x}_\ell||^2)\frac{\partial B \circ K_{Gauss}(\mathbf{x}_t, \mathbf{x}_\ell)\mathbf{c}_\ell}{\partial x_t^j}$

which after some calculations reduces to:

$$\begin{aligned}\hat{J}_{ij}(t) \quad = \quad & 2\gamma_2 b_{ij}(x_t^i - x_t^j)\exp\left(-\gamma_2(x_t^i - x_t^j)^2\right)c_t^j \\ & -2\gamma_1 \sum_{\ell \neq t} \exp(-\gamma_1||\mathbf{x}_t - \mathbf{x}_\ell||^2)(x_t^j - x_\ell^j)\sum_{p=1}^{d} b_{ip}\exp\left(-\gamma_2(x_t^i - x_\ell^p)^2\right)c_\ell^p\end{aligned} \tag{15}$$

The obtained expression exhibits some interesting characteristics: if we choose γ_1 very close to 0 (for $\gamma_1 = 0$, k_{Gauss} is no more a kernel), then $k_{Gauss}(\mathbf{x}_t, \mathbf{x}_\ell)$ is close to one for any pair $(\mathbf{x}_t, \mathbf{x}_\ell)$ and the second term in (15) is approximatively 0. Then, the value of the Jacobian for variables (i,j) is controlled by the value of b_{ij}: hence, B is capable of imposing structure in the model. For example, for a given pair of variables (i,j), if $b_{ij} = 0$, then irrespective of the values of c_ℓ^j, $\ell = 1, \cdots, N$, the corresponding Jacobian coefficient will be zero as well; i.e variable j does not influence variable i. Conversely, a non-zero coefficient b_{ij} does not reflect influence from j to i since the $\mathbf{c}_\ell$ parameters can still set the corresponding coefficient in the Jacobian to zero. Thus, the parameter B captures some of the structure of the underlying network, *together* with C. Note that the vectors $\mathbf{c}_\ell$'s and the cross-difference between coordinates in equation (15) allow us to have non-symmetric Jacobian matrices, suitable for reconstructing directed graphs.

4 Learning OKVAR with proximal gradient algorithms

4.1 Learning C for fixed kernel

In some applications, kernel K may be already specified. For instance, the transformable Gaussian kernel depends on a parameter γ that might be preset. For a decomposable or a Hadamard kernel, the matrix B may be provided a priori. Thus, learning the resulting OKVAR model boils down to learning the matrix of model parameters C. We denote the model h_C to highlight that dependence. To estimate C, we employ the following general regularized loss function :

$$\mathcal{L}(C) = \sum_{t=1}^{N} ||h_C(\mathbf{x}_t) - \mathbf{x}_{t+1}||^2 + \lambda_h ||h_C||^2_{\mathcal{H}} + \Omega(C) \tag{16}$$

The squared norm $||h_C||^2_{\mathcal{H}} = \sum_{i,j=1}^{N} \mathbf{c}_i^T K(\mathbf{x}_i, \mathbf{x}_j)\mathbf{c}_j$ plays the role of a weighted ℓ_2 norm on C. When $\Omega(C) = 0$, minimizing (16) turns out to solving the kernel ridge regression problem. In this case C can be computed in closed-form solution :

$$\mathbf{c} = (\mathbf{K} + \lambda_h Id)^{-1} \mathbf{x}_{2:N+1} \tag{17}$$

where $\mathbf{c}$ is the vectorized form of matrix C, $\mathbf{K} = (K(\mathbf{x}_\ell, \mathbf{x}_t))_{\ell,t} \in \mathcal{M}^{Nd}$ is the block-Gram matrix computed on pairs of data $(\mathbf{x}_\ell, \mathbf{x}_t)$, $\ell, t = 1 \dots N$ and $\mathbf{x}_{2:N+1} \in \mathbb{R}^{Nd}$ is the concatenated vector of data $\mathbf{x}_2, \dots, \mathbf{x}_{N+1}$. However this solution is usually not sparse. In order to control the sparsity of the model, necessary for obtaining a sparse Jacobian, one may introduce an ℓ_1-norm constraint on C, that is $\Omega_1(C) = \lambda_C \|C\|_1$ where $\|\cdot\|_1$ denotes *both* the ℓ_1 norm of a vector and that of the vectorized form of a matrix. Then the loss function becomes analogous to the one used in elastic-net type regularized models in the scalar case.

In nonparametric approaches, one key issue is to control model complexity. A way to perform it is to make use of few parameters $\mathbf{c}_\ell$, which implies that only a few data are involved in the model. By referring to Support Vector Machine, we denote by *Support Vectors* data corresponding to a null vector $\mathbf{c}_\ell$. Regularizing by the ℓ_1-norm does not allow to set a whole vector $\mathbf{c}_\ell$ to zero, e.g. to exhibit support vectors. To achieve that, a *structured sparsity* strategy is more appropriate by considering the columns of C, i.e. vectors $\mathbf{c}_\ell$'s, as a partition of the matrix coefficients. Such a constraint Ω_{struct} may take the following form:

$$\Omega_{struct}(C) = \lambda_C \sum_{\ell=1}^{N} w_\ell \|\mathbf{c}_\ell\|_2 \tag{18}$$

As it is defined in (18), Ω_{struct} is the so-called mixed ℓ_1/ℓ_2-norm. First used in Yuan and Lin (2006) for the group Lasso, this norm has interesting features : it behaves like an ℓ_1-norm on each vector $\mathbf{c}_\ell$ while within each vector $\mathbf{c}_\ell$, the

coefficients are subject to an ℓ_2-norm constraint. w_ℓ's are positive weights whose values depend on the application. For instance, in our case, as the observed time-course data are the response of a dynamical system to some given initial condition, w_ℓ should increase with ℓ, meaning that we put emphasis on the first time-points.

We can thus note that (16) is a convex loss function that is a sum of two terms: f_C which is differentiable with respect to C and g_C which is non-smooth, but nevertheless convex and subdifferentiable with respect to C:

$$\mathcal{L}(C) = \underbrace{\sum_{t=1}^{N} \left\| \sum_{\ell=1}^{N} K(\mathbf{x}_t, \mathbf{x}_\ell)\mathbf{c}_\ell - \mathbf{x}_{t+1} \right\|^2 + \lambda_h \sum_{t,\ell=1}^{N} \mathbf{c}_t^T K(\mathbf{x}_t, \mathbf{x}_\ell)\mathbf{c}_\ell}_{f_C(C)} + \underbrace{\Omega(C)}_{g_C(C)}$$

This leads us to employ a proximal gradient algorithm, which is appropriate for solving this problem. Its steps are outlined in Algorithm 1:

Algorithm 1 Solve Problem (16)

Inputs : $C_0 \in \mathcal{M}^{Nd}$;M;$\epsilon_{\mathbf{c}}$;L_C
Initialize : $m = 0; \mathbf{y}^{(1)} = \mathbf{c}^{(0)}; t^{(1)} = 1$; STOP=false
while $m < M$ and STOP=false **do**
  Step 0: $m \leftarrow m + 1$
  Step 1: $\mathbf{c}^{(m)} = \text{prox}_{\frac{1}{L_C}}(g_C)\left(\mathbf{y}^{(m)} - \frac{1}{L_C}\nabla_{\mathbf{y}^{(m)}} f_C(\mathbf{y}^{(m)})\right)$
  if $||\mathbf{c}^{(m)} - \mathbf{c}^{(m-1)}|| \leq \epsilon_{\mathbf{c}}$ **then**
    STOP:=true
  else
    Step 2: $t^{(m+1)} = \frac{1+\sqrt{1+4t^{(m)2}}}{2}$
    Step 3: $\mathbf{y}^{(m)} = \mathbf{c}^{(m)} + \frac{t^{(m)}-1}{t^{(m+1)}}\left(\mathbf{c}^{(m)} - \mathbf{c}^{(m-1)}\right)$
  end if
end while

The algorithm relies on the following:

- L_C is a Lipschitz constant of $\nabla_C f_C$ the derivative of f_C for variable C
- For $s > 0$, the proximal operator of a function g applied to some $\mathbf{v} \in \mathbb{R}^{Nd}$ is given by: $\text{prox}_s(g)(\mathbf{v}) = \text{argmin}_{\mathbf{u}}\left\{g(\mathbf{u}) + \frac{1}{2s}||\mathbf{u} - \mathbf{v}||^2\right\}$
- Intermediary variables $t^{(m)}$ and $\mathbf{y}^{(m)}$ in Step 2 and Step 3 respectively are introduced to accelerate the proximal gradient method (Beck and Teboulle, 2010).

The proximal operator of Ω_1 or Ω_{struct} is the elementwise shrinkage or soft-thresholding operator $\mathcal{T}_s : \mathbb{R}^{Nd} \to \mathbb{R}^{Nd}$:

Let $\mathcal{G}$ be a partition of the indices of $\mathbf{v}$, for a given subset of indices $I \in \mathcal{G}$,

$$\mathcal{T}_s(\mathbf{v})_I \quad = \quad \left(1 - \frac{s}{\|\mathbf{v}^I\|_2}\right)_+ \mathbf{v}^I$$

where $\mathbf{v}^I \in \mathbb{R}^{|I|}$ denotes the coefficients of $\mathbf{v}$ indexed by I. Then, the proximal gradient term in the m^{th} iteration is given by:

$$\text{prox}_{\frac{1}{L_C}}(g_C)\left(\mathbf{y}^{(m)} - \frac{1}{L_C}\nabla_{\mathbf{y}^{(m)}} f_C(\mathbf{y}^{(m)})\right)_{I_\ell} \quad = \quad \mathcal{T}_{s_\ell}\left(\mathbf{y}^{(m)} - \frac{1}{L_C}\nabla_{\mathbf{y}^{(m)}} f_C(\mathbf{y}^{(m)})\right)_{I_\ell}$$

For $g_C = \Omega_{struct}$, $s_\ell = \frac{\lambda_C w_\ell}{L_C}$ and I_ℓ, $\ell = 1, \ldots, N$ is the subset of indices corresponding to the ℓ-th column of C, while for $g_C = \Omega_1$, $s_\ell = \frac{\lambda_C}{L_C}$ and I_ℓ, $\ell = 1, \ldots, Nd$ is a singleton corresponding to a single entry of C.

We also need to calculate L_C a Lipschitz constant of $\nabla_C f$. We can notice that $f_C(C)$ can be rewritten as:

$$f_C(C) = \|\mathbf{Kc} - \mathbf{x}_{2:N+1}\|^2 + \lambda_h \mathbf{c}^T \mathbf{Kc}$$

Hence,

$$\nabla_C f_C(C) = 2\mathbf{K}([\mathbf{K} + \lambda_h Id]\mathbf{c} - \mathbf{x}_{2:N+1})$$

Using some algebra, we can establish that:
For $\mathbf{c}_1, \mathbf{c}_2 \in \mathbb{R}^{Nd}$,

$$||\nabla_C f_C(\mathbf{c}_1) - \nabla_C f_C(\mathbf{c}_2)|| \leq \underbrace{2\rho\left(\mathbf{K}^2 + \lambda_h \mathbf{K}\right)}_{L_C} \|\mathbf{c}_1 - \mathbf{c}_2\|$$

where $\rho(\mathbf{K}^2 + \lambda_h \mathbf{K})$, is the largest eigenvalue of $\mathbf{K}^2 + \lambda_h \mathbf{K}$.
Remark: It is of interest to notice that Algorithm 1 is very general and may be used as long as the loss function can be *split* into two convex terms, one of which is differentiable.

4.2 Learning C and the kernel

Other applications require the learning of kernel K. For instance, in order to tackle the network inference task, one may choose a Gaussian decomposable kernel K_{dec} or a Gaussian transformable kernel $K_{Hadamard}$. When bandwidth parameters γ are preset, the positive semi-definite matrix B underlying these kernels imposes structure in the model and has to be learnt. This leads to the more involved task of simultaneously learning the matrix of the model parameters C as well as B. Thus, we aim to minimize the following loss function for the two models h_{dec} and $h_{Hadamard}$:

$$\mathcal{L}(B, C) = \sum_{t=1}^{N} ||h_{B,C}(\mathbf{x}_t) - \mathbf{x}_{t+1}||^2 + \lambda_h ||h_{B,C}||^2_{\mathcal{H}} + \Omega(C) + \Omega(B) \qquad (19)$$

with $\Omega(C)$ a sparsity-inducing norm (Ω_1 or Ω_{struct}), $\Omega(B) = \lambda_B||B||_1$ and subject to the constraint that $B \in \mathcal{S}_d^+$ where $\mathcal{S}_d^+$ denotes the cone of positive semi-definite matrices of size $d \times d$. For fixed C, the squared norm of $h_{B,\hat{C}}$ imposes a smoothing constraint on B, while the ℓ_1 norm of B aids in controlling the sparsity of the model and its Jacobian.

Further, for fixed B, the loss function $\mathcal{L}(B_{fixed}, C)$ is convex in C and conversely, for fixed C, $\mathcal{L}(B, C_{fixed})$ is convex in B. We propose an alternating optimization scheme to minimize the overall loss $\mathcal{L}(B, C)$. Since both loss functions $\mathcal{L}(B_{fixed}, C)$ and $\mathcal{L}(B, C_{fixed})$ involve a sum of two terms, one being differentiable and the other being sub-differentiable, we employ proximal gradient algorithms to achieve the minimization.

For fixed $\hat{B}$, the loss function becomes:

$$\mathcal{L}(\hat{B}, C) = \sum_{t=1}^{N} ||h_{\hat{B},C}(\mathbf{x}_t) - \mathbf{x}_{t+1}||^2 + \lambda_h ||h_{\hat{B},C}||_{\mathcal{H}}^2 + \Omega(C), \tag{20}$$

while for given $\hat{C}$, it is given by:

$$\mathcal{L}(B, \hat{C}) = \sum_{t=1}^{N} ||h_{B,\hat{C}}(\mathbf{x}_t) - \mathbf{x}_{t+1}||^2 + \lambda_h ||h_{B,\hat{C}}||_{\mathcal{H}}^2 + \lambda_B ||B||_1 \tag{21}$$

In summary, the general form of the algorithm is given in Algorithm 2.

Algorithm 2 Solve Problem (19)

Inputs : $B_0 \in \mathcal{S}_d^+; \epsilon_B; \epsilon_C$
Initialize : $m = 0$; STOP=false
while STOP=false **do**
  Step 1: Given B_m, minimize the loss function (20) and obtain C_m
  Step 2: Given C_m, minimize the loss function (21) and obtain B_{m+1}
  if $m > 0$ **then**
    STOP:=$||B_m - B_{m-1}|| \leq \epsilon_B$ and $||C_m - C_{m-1}|| \leq \epsilon_C$
  end if
  Step 3: $m \leftarrow m + 1$
end while

At iteration m, B_m is fixed, kernel K is thus defined. Hence, estimation of C_m in Step 1 boils down to applying Algorithm 1 to minimize (20).

4.2.1 Learning the matrix B for fixed C

For given parameter matrix C, the loss function $\mathcal{L}(B, \hat{C})$ is minimized subject to the constraint that B is positive semi-definite.

$$\mathcal{L}(B, \hat{C}) = \underbrace{\sum_{t=1}^{N} ||h_{B,\hat{C}}(\mathbf{x}_t) - \mathbf{x}_{t+1}||^2 + \lambda_h ||h_{B,\hat{C}}||_{\mathcal{H}}^2}_{f_B(B)} + \underbrace{\lambda_B ||B||_1}_{g_{1,B}(B)} + \underbrace{1_{\mathcal{S}_d^+}(B)}_{g_{2,B}(B)} \tag{22}$$

where $1_{\mathcal{S}_d^+}$ denotes the indicator function : $1_{\mathcal{S}_d^+}(B) = 0$ if $B \in \mathcal{S}_d^+$, $+\infty$ otherwise.

Note that f_B is differentiable with respect to B, while both $g_{1,B}$ and $g_{2,B}$ are non-smooth, but convex and sub-differentiable with respect to B. When there is more than one non-smooth function involved in the loss function to minimize, we cannot use the same proximal gradient algorithm as delineated in Algorithm 1. We decide to adopt a strategy proposed by Raguet et al (2011), where the authors generalize the classic forward-backward splitting algorithm to the case of an arbitrary number of non-smooth functions. The method has recently proven successful for the estimation of matrices with sparsity and low rank constraints (Richard et al, 2012). Our algorithm is presented below.

Algorithm 3 Solve problem (21)

Inputs : $M; \epsilon; \epsilon_{\mathcal{L}}; \epsilon_B; Z_1^{(0)}, Z_2^{(0)} \in \mathcal{S}_d^+; \alpha \in]0,1[; \forall m \in \mathbb{N}, \eta_m \in]0, 2/L_B[; \forall m \in \mathbb{N}, \mu_m \in I_\mu$
Initialize : $m = 0; B_0 = \alpha Z_1^{(0)} + (1-\alpha) Z_2^{(0)}$; STOP=false
while $m < M$ and STOP=false **do**
  Step 1.1: $Z_1^{(m+1)} = Z_1^{(m)} + \mu_m \left(\text{prox}_{\frac{\eta_m}{\alpha}}(g_{B,1}) \left(2B_m - Z_1^{(m)} - \eta_m \nabla_B f_B(B_m) \right) - B_m \right)$
  Step 1.2: $Z_2^{(m+1)} = Z_2^{(m)} + \mu_m \left(\text{prox}(g_{B,2}) \left(2B_m - Z_2^{(m)} - \eta_m \nabla_B f_B(B_m) \right) - B_m \right)$
  Step 2: $B_{m+1} = \alpha Z_1^{(m+1)} + (1-\alpha) Z_2^{(m+1)}$
  if $\mathcal{L}(B_{m+1}, \hat{C}) \leq \epsilon$ and $|\mathcal{L}(B_{m+1}, \hat{C}) - \mathcal{L}(B_m, \hat{C})| \leq \epsilon_{\mathcal{L}}$ and $||B_{m+1} - B_m||_{Fro} \leq \epsilon_B$ **then**
    STOP:=true
  end if
  Step 3: $m \leftarrow m + 1$
end while

Two proximal operators need to be computed. The proximal operator of $g_{1,B}$ is the soft-threshold operator while the proximal operator corresponding to the indicator function $1_{\mathcal{S}_d^+}$ is the projection onto the cone of positive semidefinite matrices : for $Q \in \mathcal{S}_d, \text{prox}(g_{B,2})(Q) = \Pi_{\mathcal{S}_d^+}(Q) = \text{argmin}_{B \in \mathcal{S}_d^+} ||B - Q||_{Fro}$. Sequence (B_m) is guaranteed to convergence under the following assumptions (Theorem 2.1 in Raguet et al (2011)):
Set $\overline{\lim} \eta_m = \overline{\eta}$,

(A) (i) $0 < \underline{\lim} \mu_m \leq \overline{\lim} \mu_m < \min\left(\frac{3}{2}, \frac{1+2/(L_B \overline{\eta})}{2}\right)$
(ii) $\sum_{m=0}^{+\infty} \|u_{2,m}\| < +\infty$, and for $i \in \{1,2\}$, $\sum_{m=0}^{+\infty} \|u_{1,m,i}\| < +\infty$

(B) (i) $0 < \underline{\lim} \eta_m \leq \overline{\eta} < \frac{2}{L_B}$
(ii) $I_\mu =]0, 1]$

where for $i \in \{1,2\}$, $u_{1,m,i}$ denotes the error at iteration m when computing the proximal operator $\text{prox}(g_{B,i})$ and $u_{2,m}$ is the error when applying $\nabla_B f_B$ to its argument.

This algorithm also makes use of the following gradient computations. Here we present the computations for kernel $K_{Hadamard}$, but all of them still hold for a decomposable kernel by setting $K_{Gauss} = 1$.

Specifically, we get:

- for $\frac{\partial}{\partial b_{ij}} \left\| h_{B,\hat{C}}(\mathbf{x}_t) - \mathbf{x}_{t+1} \right\|^2$

$$\begin{aligned}
\frac{\partial}{\partial b_{ij}} \left\| h_{B,\hat{C}}(\mathbf{x}_t) - \mathbf{x}_{t+1} \right\|^2 &= \frac{\partial}{\partial b_{ij}} \sum_{p=1}^{d} \left(h_{B,\hat{C}}(\mathbf{x}_t)^p - x_{t+1}^p \right)^2 \\
&= 2 \left(\sum_{\ell=1}^{N} k_{Gauss}(\mathbf{x}_t, \mathbf{x}_\ell) K_{Gauss}(\mathbf{x}_t, \mathbf{x}_\ell)_{ij} c_\ell^j \right) \left(h_{B,\hat{C}}(\mathbf{x}_t)^i - x_{t+1}^i \right) \\
&\quad +2 \left(\sum_{\ell=1}^{N} k_{Gauss}(\mathbf{x}_t, \mathbf{x}_\ell) K_{Gauss}(\mathbf{x}_t, \mathbf{x}_\ell)_{ji} c_\ell^i \right) \left(h_{B,\hat{C}}(\mathbf{x}_t)^j - x_{t+1}^j \right)
\end{aligned}$$

- for $\frac{\partial}{\partial b_{ij}} ||h_{B,\hat{C}}||^2_{\mathcal{H}}$

$$\begin{aligned}
\frac{\partial}{\partial b_{ij}} ||h_{B,\hat{C}}||^2_{\mathcal{H}} &= \frac{\partial}{\partial b_{ij}} \sum_{t=1}^{N} \sum_{\ell=1}^{N} \mathbf{c}_t^T K_{Hadamard}(\mathbf{x}_t, \mathbf{x}_\ell) \mathbf{c}_\ell \\
&= \sum_{t=1}^{N} \sum_{\ell=1}^{N} c_t^i k_{Gauss}(\mathbf{x}_t, \mathbf{x}_\ell) K_{Gauss}(\mathbf{x}_t, \mathbf{x}_\ell)_{ij} c_\ell^j + c_t^j k_{Gauss}(\mathbf{x}_t, \mathbf{x}_\ell) K_{Gauss}(\mathbf{x}_t, \mathbf{x}_\ell)_{ji} c_\ell^i
\end{aligned}$$

We again need to compute a Lipschitz constant L_B for $\nabla_B f_B(B)$. After some calculations, one can show the following inequality :
For $B_1, B_2 \in \mathcal{S}_d^+$,

$$\|\nabla_B f_B(B_1) - \nabla_B f_B(B_2)\|^2_{Fro} \leq L_B^2 \|B_1 - B_2\|^2_{Fro}$$

with

$$\begin{aligned}
L_B^2 &= 4 \sum_{i,j=1}^{d} \left(\sum_{t=1}^{N} \left(\sum_{\ell=1}^{N} k_{Gauss}(\mathbf{x}_t, \mathbf{x}_\ell) K_{Gauss}(\mathbf{x}_t, \mathbf{x}_\ell)_{ij} c_\ell^j \right) \left(\sum_{\ell=1}^{N} \sum_{q=1}^{d} k_{Gauss}(\mathbf{x}_t, \mathbf{x}_\ell) K_{Gauss}(\mathbf{x}_t, \mathbf{x}_\ell)_{iq} c_\ell^q \right) \right. \\
&\quad \left. + \sum_{t=1}^{N} \left(\sum_{\ell=1}^{N} k_{Gauss}(\mathbf{x}_t, \mathbf{x}_\ell) K_{Gauss}(\mathbf{x}_t, \mathbf{x}_\ell)_{ji} c_\ell^i \right) \left(\sum_{\ell=1}^{N} \sum_{q=1}^{d} k_{Gauss}(\mathbf{x}_t, \mathbf{x}_\ell) K_{Gauss}(\mathbf{x}_t, \mathbf{x}_\ell)_{jq} c_\ell^q \right) \right)^2
\end{aligned}$$

5 Results

5.1 Implementation

The performance of the developed OKVAR model family and the proposed optimization algorithms were assessed on simulated data from a biological system

(DREAM3 challenge data set). These algorithms include a number of tuning parameters. Specifically, in Algorithm 3, we set $Z_1^{(0)} = Z_2^{(0)} = B_0 \in \mathcal{S}_d^+$, $\alpha = 0.5$ and $\mu_m = 1$. Parameters of the kernels were also fixed a priori : parameter γ was set to 0.2 for a transformable Gaussian kernel and to 0.1 for a decomposable Gaussian kernel. In the case of a Hadamard kernel, two parameters need to be chosen : parameter γ_2 of the transformable Gaussian kernel remains unchanged ($\gamma_2 = 0.2$). On the other hand, as discussed in Section 3.4, parameter γ_1 of the decomposable Gaussian kernel is fixed to a low value ($\gamma_1 = 10^{-5}$) since it does not play a key role in the network inference task.

5.2 DREAM3 dataset

We start our investigation by considering data sets obtained from the DREAM3 challenge (Prill et al, 2010). DREAM stands for Dialogue for Reverse Engineering Assessments and Methods (`http://wiki.c2b2.columbia.edu/dream/index.php/The_DREAM_Project`) and is a scientific consortium that organizes challenges in computational biology and especially for gene regulatory network inference. In a gene regulatory network, a gene i is said to regulate another gene j if the expression of gene i at time t influences the expression of gene j at time $t + 1$. The DREAM3 project provides realistic simulated data for several networks corresponding to different organisms (e.g. *E. coli*, Yeast, etc.) of different sizes and topological complexity. We focus here on size-10 and size-100 networks generated for the DREAM3 in-silico challenges. Each of these networks corresponds to a subgraph of the currently accepted *E. coli* and *S. cerevisiae* gene regulatory networks and exhibits varying patterns of sparsity and topological structure. They are referred to as E1, E2, Y1, Y2 and Y3 with an indication of their size. The data were generated by imbuing the networks with dynamics from a thermodynamic model of gene expression and Gaussian noise. Specifically, 4 and 46 time series consisting of 21 points were available respectively for size-10 and size-100 networks.

In all the experiments conducted, we assess the performance of our model using the area under the ROC curve (AUROC) and under the Precision-Recall curve (AUPR) for regulation ignoring the sign (positive vs negative influence). For the DREAM3 data sets we also show the best results obtained from other competing teams using **only** time course data. The challenge made available other data sets, including ones obtained from perturbation (knock-out/knock-down) experiments, as well as observing the organism in steady state, but these were not considered in the results shown in the ensuing tables.

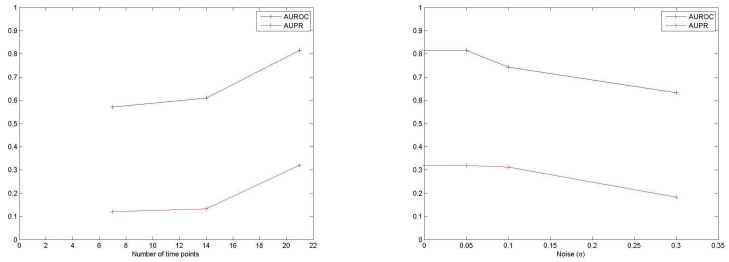
Further, several time series may also be available, because of multiple related initial conditions and/or technical replicates. In this case, the procedure is repeated accordingly. Hence, the predictions of each run on each time series are combined to build a *consensus* network. Specifically, for each run, we compute a rank matrix as described in Section 2. Then the coefficients of these matrices are averaged and eventually thresholded to obtain a final estimate of the adjacency matrix.

5.2.1 Effects of hyperparameters, noise and sample size

Next, we study the impact of different parameters including hyperparameters λ_C and λ_B, the sample size of the dataset (number of time points) and the noise level. Results for the DREAM3 size-10 E1 network are given in Table 1 and Figure 1, respectively.

Table 1: Consensus AUROC and AUPR (given in %) for the DREAM3 size-10 E1 network using different hyperparameters. $\lambda_h = 1$

		λ_B		
		10^{-2}	10^{-1}	1
λ_C	10^{-2}	79.1/31.5	81.5/32.1	73.5/21.2
	10^{-1}	79.7/36.5	76.9/21.6	66.7/25.9
	1	78.1/25.9	71.0/20.7	61.4/16.0



(a) (b)

Figure 1: Consensus AUROC (blue lines) and AUPR (red lines) for the DREAM3 size-10 E1 network (a) using $N = 7, 14$ and 21 time points (b) adding zero-mean Gaussian noise with standard deviations $\sigma = 0, 0.05, 0.1$ and 0.3. $\lambda_h = 1, \lambda_C = 10^{-2}, \lambda_B = 10^{-1}$.

We notice that both AUROC and AUPR values are not that sensitive to small changes in the hyperparameters (Table 1), which strongly indicates that it is sufficient to pick them within a reasonable range of values. Furthe, as expected, performance deteriorates with increasing levels of noise and decreasing number of time points (Figure 1).

5.2.2 Comparison between OKVAR models

Next, we propose to compare the OKVAR models outlined in Table 2 and investigate how relevant these models are regarding the network inference task.

Table 3 shows that the transformable Gaussian and the decomposable Gaussian kernels can achieve good performance on selected DREAM3 size-10 datasets,

Table 2: Synthetic table of studied OKVAR models.

	OKVAR models						
	h^{Ridge}_{Gauss}	$h^{\ell_1}_{Gauss}$	$h^{\ell_1/\ell_2}_{Gauss}$	$h^{\ell_1}_{dec}$	h^{ℓ_1/ℓ_2}_{dec}	$h^{\ell_1}_{Hadamard}$	$h^{\ell_1/\ell_2}_{Hadamard}$
Kernel	Transformable Gaussian			Decomposable Gaussian		Hadamard	
Loss	Eq. (16)			Eq. (19)			
$\Omega(C)$	0	Ω_1	Ω_{struct}	Ω_1	Ω_{struct}	Ω_1	Ω_{struct}

Table 3: Consensus AUROC and AUPR (given in %) for the DREAM3 size-10 networks. Studied OKVAR models refer to Table 2. The numbers in **boldface** are the maximum values of each column.

	AUROC					AUPR				
OKVAR models	E1	E2	Y1	Y2	Y3	E1	E2	Y1	Y2	Y3
h^{Ridge}_{Gauss}	68.8	37.7	62.1	68.6	66.7	15.6	11.2	15.5	46.9	32.9
$h^{\ell_1}_{Gauss}$	69.3	38.0	61.9	69.3	66.7	15.7	11.3	15.2	47.4	32.8
$h^{\ell_1/\ell_2}_{Gauss}$	68.7	37.1	62.4	68.6	66.7	15.5	11.1	15.6	47.5	32.6
$h^{\ell_1}_{dec}$	67.0	68.5	38.2	45.4	38.3	23.6	20.8	7.4	21.1	16.8
h^{ℓ_1/ℓ_2}_{dec}	65.9	47.8	45.3	56.6	38.5	23.1	14.0	8.3	28.5	16.8
$h^{\ell_1}_{Hadamard}$	81.2	46.2	47.7	**76.2**	70.5	23.5	12.7	8.7	**50.1**	39.5
$h^{\ell_1/\ell_2}_{Hadamard}$	**81.5**	**78.7**	**76.5**	70.3	**75.1**	**32.1**	**50.1**	**35.4**	37.4	**39.7**

although none of these two kernels alone can faithfully recover all of the networks, no matter the types of sparsity-inducing norms employed. On the other hand, the Hadamard kernel-based model learnt with a mixed ℓ_1/ℓ_2-norm regularization consistently outranks other OKVAR models. On the whole, these results tend to corroborate the discussion presented in Section 3.4. In the remainder of the paper, we focus on Hadamard kernel-based models that will be referred to as *OKVAR-Prox*.

5.2.3 Comparison with state-of-the-art methods

The performance of the OKVAR approach for prediction of the network structure is assessed on two types of tasks: DREAM3 size-10 datasets whose ratio between the number of measurements and the network size is of 21/10 and size-100 datasets which come with a much more unfavorable ratio of 21/100.

Results are presented in Tables 4 and 5 for size-10 and size-100 data sets, respectively. The entries of these tables correspond to the following methods: (i) OKVAR + *True B* corresponds to an OKVAR model with the true adjacency matrix given, and projected onto the positive semidefinite cone. (ii) OKVAR-Prox was learnt using a mixed ℓ_1/ℓ_2-norm constraint on model parameters for size-10 datasets and an ℓ_1-norm for size-100 datasets. (iii) The LASSO algo-

rithm corresponds to an ℓ_1-regularized linear least squares regression model of the form $x_{t+1,i} = \mathbf{x}_t\beta$, applied to each dimension (gene i). An edge is assigned for each nonzero β coefficient. The LASSO algorithm employed all the available time series and a final consensus network is built in the same fashion as delineated above. (iv) G1DBN is an algorithm that performs Dynamic Bayesian Network inference using first-order conditional dependencies (Lèbre, 2009). (v) GPODE is a structure inference method based on non-parametric Gaussian process modeling and parameter estimation of ordinary differential equations (Aijo and Lahdesmaki, 2009). (vi,vii) Finally, the two last rows present the results from two competing teams that exhibited a very good performance based only on similar time-series data. Although there is no information on the structure of Team 236's algorithm, its authors responded to the post-competition DREAM3 survey stating that their method employs Bayesian models with an in-degree constraint (Prill et al, 2010). Team 190 (Table 4) reported in the same survey that their method is also Bayesian with a focus on nonlinear dynamics and local optimization.

Table 4: Consensus AUROC and AUPR (given in %) for OKVAR-Prox, LASSO, GPODE, G1DBN, Team 236 and Team 190 (DREAM3 competing teams) run on DREAM3 size-10 networks. The numbers in **boldface** are the maximum values of each column.

	AUROC					AUPR				
Size-10	E1	E2	Y1	Y2	Y3	E1	E2	Y1	Y2	Y3
OKVAR + *True B*	96.2	86.9	89.2	75.6	86.6	75.2	67.7	47.3	52.3	58.6
OKVAR-Prox	**81.5**	**78.7**	**76.5**	**70.3**	**75.1**	**32.1**	**50.1**	**35.4**	37.4	**39.7**
LASSO	69.5	57.2	46.6	62.0	54.5	17.0	16.9	8.5	32.9	23.2
GPODE	60.7	51.6	49.4	61.3	57.1	18.0	14.6	8.9	**37.7**	34.1
G1DBN	63.4	77.4	60.9	50.3	62.4	16.5	36.4	11.6	23.2	26.3
Team 236	62.1	65.0	64.6	43.8	48.8	19.7	37.8	19.4	23.6	23.9
Team 190	57.3	51.5	63.1	57.7	60.3	15.2	18.1	16.7	37.1	37.3

The AUROC and AUPR values obtained for size-10 networks (Table 4) strongly indicate that OKVAR-Prox outperforms state-of-the-art models and the teams that exclusively used the same set of time series data in the DREAM3 competition, except for size-10 Y2 (nearly equivalent AUPR). In particular, we note that the OKVAR consensus runs exhibited excellent AUPR values compared to those obtained by other approaches.

A comparison of competing algorithms for size-100 networks (Table 5) shows that the OKVAR method again achieves superior AUROC results compared to Team 236, although it only lags behind by a slight margin for size-100 Y1 and Y3 in terms of AUPR. Team 236 was the only team that exclusively used time series data for the size-100 network challenge, since Team 190 did not submit any results. No results are provided for the GPODE method on size-100 networks

either since the algorithm requires a full combinatorial search when no prior knowledge is available, which is computationally intractable for large networks. The OKVAR method is outranked by G1DBN for size-100 E2 in terms of AUPR and for size-100 Y2 with quite comparable AUROC values. It is noticeable that the AUPR values in all rows are rather small (lower than 10%) compared to their size-10 counterparts. Such a decrease suggests that the AUPR values can be impacted more strongly by the lower density of the size-100 networks, where the *non-edges* class severely outnumbers the *edges* class, rather than the choice of algorithm. Such difficult tasks require much more available time-series to achieve better results in terms of AUROC and AUPR. Therefore, for size-100 datasets, we applied a pure ℓ_1-norm constraint on model parameters, allowing any C coefficients to be set to 0 rather than a mixed ℓ_1/ℓ_2-norm regularization that would have been too stringent in terms of data parsimony.

Finally, it is worth noting that OKVAR-Prox would have ranked in the top five and ten, respectively for size-10 and size-100 challenges, while the best results employed knock-out/knock-down data in addition to time-series data, the latter being rich in information content (Michailidis, 2012).

Table 5: Consensus AUROC and AUPR (given in %) for OKVAR-Prox, LASSO, G1DBN, Team 236 (DREAM3 competing team) run on DREAM3 size-100 networks. The numbers in **boldface** are the maximum values of each column.

	AUROC					AUPR				
Size-100	E1	E2	Y1	Y2	Y3	E1	E2	Y1	Y2	Y3
OKVAR + *True B*	96.2	97.1	95.8	90.6	89.7	43.2	51.6	27.9	40.7	36.4
OKVAR-Prox	**65.4**	**64.0**	**54.9**	56.8	**53.5**	**4.6**	2.6	2.3	**5.0**	6.3
LASSO	52.2	55.0	53.2	52.4	52.3	1.4	1.3	1.8	4.3	6.1
G1DBN	53.4	55.8	47.0	**58.1**	43.4	1.6	**6.3**	2.2	4.6	4.4
Team 236	52.7	54.6	53.2	50.8	50.8	1.9	4.2	**3.5**	4.6	**6.5**

5.2.4 Comparison with OKVAR-Boost

In previous work (Lim et al, 2013), a boosting algorithm (OKVAR-Boost) combining features from L_2-boosting and randomization-based algorithms was designed. At each boosting iteration, a Hadamard kernel-based OKVAR model was learnt on a random subspace. One main difference between OKVAR-Boost and OKVAR-Prox concerns the learning strategy. While B and C are learnt jointly for the latter, the learning of B and C is decoupled in OKVAR-Boost, meaning that B is firstly estimated by means of a statistical independence test and then C is learnt using an elastic-net regularized loss. A comparison of the two related algorithms is given in Table 6.

OKVAR-Prox achieves better AUROC values than OKVAR-Boost for size-10 networks, except for the E1 network, while there is no clear winner in terms of

Table 6: Consensus AUROC and AUPR (given in %) for OKVAR-Prox and OKVAR-Boost run on DREAM3 size-10 and size-100 networks. The numbers in **boldface** are the maximum values of each column.

	AUROC					AUPR				
Size-10	E1	E2	Y1	Y2	Y3	E1	E2	Y1	Y2	Y3
OKVAR-Prox	81.5	**78.7**	**76.5**	**70.3**	**75.1**	32.1	50.1	**35.4**	**37.4**	39.7
OKVAR-Boost	**85.3**	74.9	68.9	65.3	69.5	**58.3**	**53.6**	28.3	26.8	**44.3**
Size-100	E1	E2	Y1	Y2	Y3	E1	E2	Y1	Y2	Y3
OKVAR-Prox	65.4	64.0	54.9	56.8	53.5	**4.6**	2.6	2.3	5.0	6.3
OKVAR-Boost	**71.8**	**77.2**	**72.9**	**65.0**	**64.3**	3.6	**10.7**	**4.2**	**7.3**	**6.9**

AUPR. On size-100 inference tasks, OKVAR-Boost which benefits from projections on random subspaces patently outperforms OKVAR-Prox which operates directly in the 100-dimensional space with a limited amount of time points.

6 Conclusion

Network inference from multivariate time-series represents a key problem in numerous scientific domains. In this paper, we addressed it by introducing and learning a new family of nonlinear vector autoregressive models based on operator-valued kernels. The new models generalize linear vector autoregressive models and benefit from the framework of regularization. To obtain a sparse network estimate, we define appropriate non-smooth penalties on the model parameters and a proximal gradient algorithm to deal with them. Some of the chosen operator-valued kernels are characterized by a positive semi-definite matrix that also plays a role in the network estimation. In this case, an alternating optimization scheme based on two proximal gradient procedures is proposed to learn both kinds of parameters. Results obtained from benchmark size-10 and size-100 biological networks show very good performance of the OKVAR model. Future extensions include applications of OKVAR to other scientific fields, the use of the model in ensemble methods, and the application of proximal gradient algorithms to other structured output prediction tasks.

Acknowledgements FA-B's work was supported in part by ANR (call SYSCOM-2009, ODESSA project). GM's work was supported in part by NSF grants DMS-1161838 and DMS-1228164 and NIH grant 1-R21-GM-101719-01-A1

References

Aijo T, Lahdesmaki H (2009) Learning gene regulatory networks from gene expression measurements using non-parametric molecular kinetics. Bioinfor-

matics 25:22:2937–2944

Alvarez MA, Rosasco L, D LN (2011) Kernels for vector-valued functions: a review. Tech. rep., MIT_CSAIL-TR-2011-033

Auliac C, Frouin V, Gidrol X, d'Alché Buc F (2008) Evolutionary approaches for the reverse-engineering of gene regulatory networks: a study on a biologically realistic dataset. BMC Bioinformatics 9(1):91

Baldassarre L, Rosasco L, Barla A, Verri A (2010) Vector field learning via spectral filtering. In: Balczar J, Bonchi F, Gionis A, Sebag M (eds) Machine Learning and Knowledge Discovery in Databases, Lecture Notes in Computer Science, vol 6321, Springer Berlin / Heidelberg, pp 56–71

Beck A, Teboulle M (2010) Gradient-based algorithms with applications to signal recovery problems. In: Palomar D, Eldar Y (eds) Convex Optimization in Signal Processing and Communications, Cambridge press, pp 42–88

Bolstad A, Van Veen B, Nowak R (2011) Causal network inference via group sparsity regularization. IEEE Trans Signal Process 59(6):2628–2641

Brouard C, d'Alché Buc F, Szafranski M (2011) Semi-supervised penalized output kernel regression for link prediction. In: ICML-2011, pp 593–600

Bühlmann P, van de Geer S (2011) Statistics for High-Dimensional Data: Methods, Theory and Applications. Springer

Caponnetto A, CA, Pontil M, Ying Y (2008) Universal multitask kernels. J Mach Learn Res 9

Chatterjee S, Steinhaeuser K, Banerjee A, Chatterjee S, Ganguly AR (2012) Sparse group lasso: Consistency and climate applications. In: SDM, SIAM / Omnipress, pp 47–58

Chou I, Voit EO (2009) Recent developments in parameter estimation and structure identification of biochemical and genomic systems. Math Biosci 219(2):57–83

Dinuzzo F, Fukumizu K (2011) Learning low-rank output kernels. In: Proceedings of the 3rd Asian Conference on Machine Learning, JMLR: Workshop and Conference Proceedings, vol 20

Dondelinger F, Lèbre S, Husmeier D (2013) Non-homogeneous dynamic bayesian networks with bayesian regularization for inferring gene regulatory networks with gradually time-varying structure. Machine Learning Journal 90(2):191–230

Friedman N (2004) Inferring cellular networks using probabilistic graphical models. Science 303(5659):799–805

Gilchrist S, Yankov V, Zakrajšek E (2009) Credit market shocks and economic fluctuations: Evidence from corporate bond and stock markets. Journal of Monetary Economics 56(4):471–493

Hartemink A (2005) Reverse engineering gene regulatory networks. Nat Biotechnol 23(5):554–555

Iba H (2008) Inference of differential equation models by genetic programming. Inf Sci 178(23):4453–4468

Kadri H, Rabaoui A, Preux P, Duflos E, Rakotomamonjy A (2011) Functional regularized least squares classication with operator-valued kernels. In: ICML-2011, pp 993–1000

Kolaczyk ED (2009) Statistical Analysis of Network Data: Methods and Models. Springer: series in Statistics

Kramer MA, Eden UT, Cash SS, Kolaczyk ED (2009) Network inference with confidence from multivariate time series. Physical Review E 79(6):061,916+

Lawrence N, Girolami M, Rattray M, Sanguinetti G (eds) (2010) Learning and Inference in Computational Systems Biology. MIT Press

Lèbre S (2009) Inferring dynamic genetic networks with low order independencies. Statistical applications in genetics and molecular biology 8(1):1–38

Lim N, Senbabaoglu Y, Michailidis G, d'Alché Buc F (2013) Okvar-boost: a novel boosting algorithm to infer nonlinear dynamics and interactions in gene regulatory networks. Bioinformatics 29(11):1416–1423

Liu Y, Niculescu-Mizil A, Lozano A (2010) Learning temporal causal graphs for relational time-series analysis. In: Fürnkranz J, Joachims T (eds) ICML-2010

Maathuis M, Colombo D, Kalish M, Bühlmann P (2010) Predicting causal effects in large-scale systems from observational data. Nature Methods 7:247–248

Margolin I Aand Nemenman, Basso K, Wiggins C, Stolovitzky G, Favera R, Califano A (2006) Aracne: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. BMC Bioinf 7(Suppl 1):S7

Mazur J, Ritter D, Reinelt G, Kaderali L (2009) Reconstructing nonlinear dynamic models of gene regulation using stochastic sampling. BMC Bioinformatics 10(1):448

Meinshausen N, Bühlmann P (2006) High dimensional graphs and variable selection with the lasso. Annals of Statistics 34:1436–1462

Micchelli CA, Pontil MA (2005) On learning vector-valued functions. Neural Computation 17:177–204

Michailidis G (2012) Statistical challenges in biological networks. Journal of Computational and Graphical Statistics 21(4):840–855

Michailidis G, d'Alché Buc F (2013) Autoregressive models for gene regulatory network inference: Sparsity, stability and causality issues. Math Biosci 246(2):326 – 334

Morton R, Williams KC (2010) Experimental Political Science and the Study of Causality. Cambridge University Press

Murphy KP (1998) Dynamic bayesian networks: representation, inference and learning. PhD thesis, Computer Science, University of Berkeley, CA, USA

Parry M, Canziani O, Palutikof J, van der Linden P, Hanson C, et al (2007) Climate change 2007: impacts, adaptation and vulnerability. Intergovernmental Panel on Climate Change

Perrin BE, Ralaivola L, A M, S B, Mallet J, d'Alché Buc F (2003) Gene networks inference using dynamic bayesian networks. Bioinformatics 19(S2):38–48

Prill R, Marbach D, Saez-Rodriguez J, Sorger P, Alexopoulos L, Xue X, Clarke N, Altan-Bonnet G, Stolovitzky G (2010) Towards a rigorous assessment of systems biology models: The dream3 challenges. PLoS ONE 5(2):e9202

Raguet H, Fadili J, Peyré G (2011) Generalized forward-backward splitting. arXiv preprint arXiv:11084404

Richard E, Savalle PA, Vayatis N (2012) Estimation of simultaneously sparse and low rank matrices. In: Langford J, Pineau J (eds) ICML-2012, Omnipress, New York, NY, USA, pp 1351–1358

Senkene E, Tempel'man A (1973) Hilbert spaces of operator-valued functions. Lithuanian Mathematical Journal 13(4)

Shojaie A, Michailidis G (2010) Discovering graphical granger causality using a truncating lasso penalty. Bioinformatics 26(18):i517–i523

Yuan M, Lin Y (2006) Model selection and estimation in regression with grouped variables. Journal of the Royal Statistical Society: Series B 68(1):49–67

Zou C, Feng J (2009) Granger causality vs. dynamic bayesian network inference: a comparative study. BMC Bioinformatics 10(1):122