Functional Directed Graphical Models and Applications in Root-Cause Analysis and Diagnosis

Abstract

Directed graphical models aim to represent the probabilistic relationships between variables in a system. Learning a directed graphical model from data includes parameter learning and structure learning. Several methods have been developed for directed graphical models with scalar variables. However, the case in which the variables are infinite-dimensional has not been studied thoroughly. Nowadays, in many applications, the variables are infinite-dimensional signals that need to be treated as functional random variables. This paper proposes a novel method to learn directed graphical models in the functional setting. When the structure of the graph is known, function-to-function linear regression is used to estimate the parameters of the graph. When the goal is to learn the structure, a penalized least square loss function with a group LASSO penalty, for variable selection, and an L_2 penalty, to handle group selection of nodes, is defined. Cyclic coordinate accelerated proximal gradient descent algorithm is employed to minimize the loss function and learn the structure of the directed graph. Through simulations and a case study, the advantage of the proposed method is proven.

Keywords: directed graphical models, functional random variables, parameter learning, structure learning, function-to-function regression, penalized loss function

1 Introduction

Complex biological, physical, and social systems are often comprised of various entities and variables that exhibit intricate relationships and interactions. Directed graphical models (DGMs) have been widely used to provide a probabilistic representation of these complex systems. For example, in reliability modeling, DGMs are used to compute the overall reliability of a system, given the reliability of the individual components and how they interact, Langseth and Portinale (2007). In manufacturing processes, they are used to learn the causal relationship among process variables and quality measures, and to facilitate process control, Li and Shi (2007). Additional applications

to medical diagnosis, clinical decision support, crime factor analysis, sensor validation, information retrieval, credit-rating, risk management, and robotics are found in Pourret et al. (2008).

In the graphical representation of a system, the nodes represent random variables, and the directed arcs express the probabilistic relationships between the variables. The graph captures the way in which the joint distribution over all the random variables can be decomposed into a product of factors, each depending only on a subset of the variables, Bishop (2006). Therefore, the problem of estimating a joint distribution is simplified by the problem of estimating a few low-dimensional conditional distributions.

Most of the existing DGMs assume that the random variables associated with the nodes are scalars. However, in practice, it is common to encounter systems where the variables have a functional form, over time or space. In such cases, it makes sense to think of the variables as functional variables. For example, the exhaust after-treatment process of an internal combustion engine can be monitored by several sensors, measuring, for example, the vehicle velocity, the engine rotational speed, and the intercooler pressure, which provide a large number of waveform signals or functional data, as shown in Figure 1. Faults in the exhaust after-treatment process translate into higher fuel consumption and uncontrolled emissions of pollutants, such as nitrogen oxides (NO_x), into the environment (Pacella, 2018). Faults are easily detected by an air-to-fuel ratio (λ-upstream) falling bellow an acceptability threshold. However, the root-causes behind this behavior may change and are not clearly defined. By modeling the exhaust after-treatment process as a functional DGM, the root-causes behind fault events can be identified, and control actions can be taken. Understanding the causal relationships of this complex system can significantly improve the overall performance of the engine and the vehicle's environmental impact.

The main challenge to learn functional DGMs is preserving the functional information in each node. Existing probabilistic graphical models, developed for scalars, cannot be used or easily extended to deal with functional data. If each point in the functional data is considered as a scalar, the functional structure is lost. Recently, some methodologies have been developed to learn the structure of functional undirected graphical models. Zhu, Strawn and, Dunson (2016) proposed

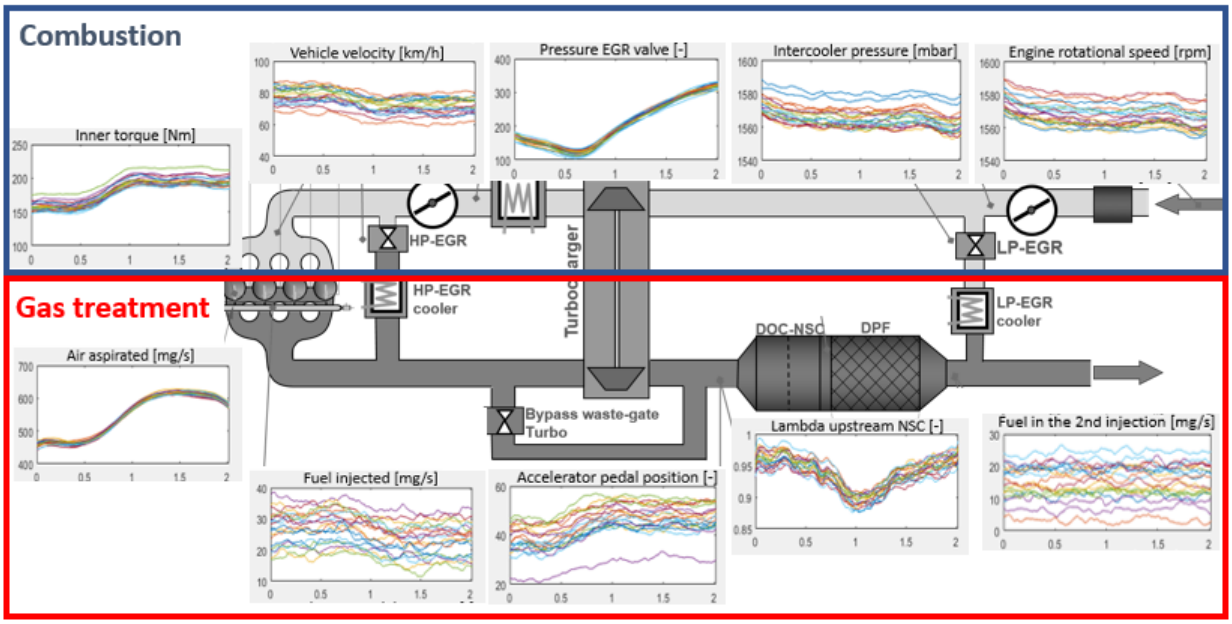


Figure 1: Exhaust after-treatment process of an internal combustion engine

decomposable graphical models for multivariate functional data from a Bayesian perspective. Qiao et al. (2018) extended the graphical LASSO of Yuan and Lin (2007) to the functional setting. Li and Solea (2018), extended the previous work, to a non-parametrical setting. On the other hand, methodologies aiming to learn functional DGM have been developed in a very specific context: brain connectivity. Lindquist (2012) proposed an algorithm to learn the parameters of a graph with one functional node, two scalar nodes, and a known structure. The goal of the study was to find brain regions whose activity acted as a potential mediator of the relationship between a treatment variable and an outcome variable. Cao et al. (2019) proposed a causal dynamic network to estimate the parameters of differential equations' models, representing latent neuronal states, from fMRI data. Under their methodology, the parameters linking two functional nodes are fixed scalars and represent brain activations and connections. The authors do not learn the parameters of the conditional distributions of the graphical model. General methodologies aiming to learn the functional parameters and the structure of a DGM with functional variables are scarce. To the best of our knowledge, there is only one paper on this topic. Sun, Huang, and Jin (2017) proposed a modeling strategy for functional DGMs in manufacturing processes. However, they assumed that time t of

a functional node is only affected by time t of its functional parents. Their approach ignores the temporal cross-correlation between the variables.

The main goal of this paper is to develop a methodology for learning functional DGMs while preserving the existing cross-correlation between the variables in the system. For example, we would like to learn how the variables in the internal combustion engine interact to identify the root-causes behind a system fault event.

Throughout the article, we have two main assumptions. (A1) The functional variables jointly follow from a multivariate Gaussian process (MGP). This is a common assumption in learning graphical models that has been shown to hold in practice (Qiao et al. (2019); Sun et al. (2017); Zhu et al. (2016)). (A2) The functional variables can be represented as a directed acyclic graph (DAG). Therefore, there exists an ordering of the variables, called topological ordering, where only the variables with lower orders can be parents (variable i is a parent of variable j if there is an arc from i to j) for the variables with higher orders. A practical example of a system where the variables have such an ordering is a sequential manufacturing process where upstream process variables can affect the downstream process variables, but the reverse cannot happen. For the internal combustion engine example, thanks to domain knowledge, it is possible to establish a topological ordering. For example, we know that the vehicle velocity depends on the accelerator pedal position and on the gear ratio. Therefore, in the model, these two variables have a lower order than the vehicle velocity.

First, we assume that the structure of the graph is known, and use function-to-function regression, Reisi Gahrooei et al. (2020), to perform parameter learning of the graph. Then, inspired by Meinshausen and Buhlmann's (2006) neighborhood selection method, we present a new approach to learn the structure of functional DGMs. In order to learn the structure of a system, we define a penalized least square loss function with two penalties: a group LASSO penalty, for variable selection, and an L_2 penalty, to handle grouped selection of nodes. We recursively employ cyclic coordinate accelerated proximal gradient descent as an algorithm to minimize the loss function and learn the structure of the directed graph. Finally, we adapt the modified cross-validation for penalized high-dimensional linear regression models, Yu and Feng (2014), to the functional setting, to

tune the parameters of the penalty terms. The main contributions of the paper are: (1) we establish a methodology to estimate the parameters of functional DGMs, given the structure of the graph, that outperforms existing methods, and (2) we develop a novel structure learning algorithm that identifies the parents of the functional variables in a graph in a neighborhood selection fashion. The proposed methodology can be used for diagnosis and root-cause analysis, as well as system control when the variables involved have a functional form.

The rest of the paper is organized as follows. In section 2, we provide a methodology overview. In section 3, we first present the parameter learning method for functional DGMs when the structure is known. Then, we present the methodology to learn the structure of functional DGMs. Simulation studies and real data analysis are conducted in sections 4 and 5, respectively. Finally, we conclude the paper in section 6.

2 Methodology Overview

A general framework of the proposed methodology is shown in Figure 2. If the structure of the functional DGM is known, we use the measured functional observations to fit function-to-function linear regressions and learn the parameters of the conditional distributions governing the DGM. The first step is to transform the infinite-dimensional regression problem into a finite-dimensional one by using a functional basis expansion. We propose to use either a data-driven basis or a domain knowledge basis to reduce the dimensionality of the problem. The use of a basis set transforms the function-to-function linear regression into a multilinear regression problem with a closed-form solution. The solution to this problem allows for the estimation of the parameters of the functional DGM. On the other hand, if the structure of the graph is unknown, we propose to add a penalty term to the loss function of the function-to-function linear regressions. In this scenario, we use a domain knowledge basis to reduce the dimensionality of the problem. A closed-form solution no longer exists because of the penalty term. We employ cyclic coordinate accelerated proximal gradient descent to solve the regression problems and estimate the structure and the parameters of the functional DGM. The detailed analysis of each step will be elaborated in subsequent sections.

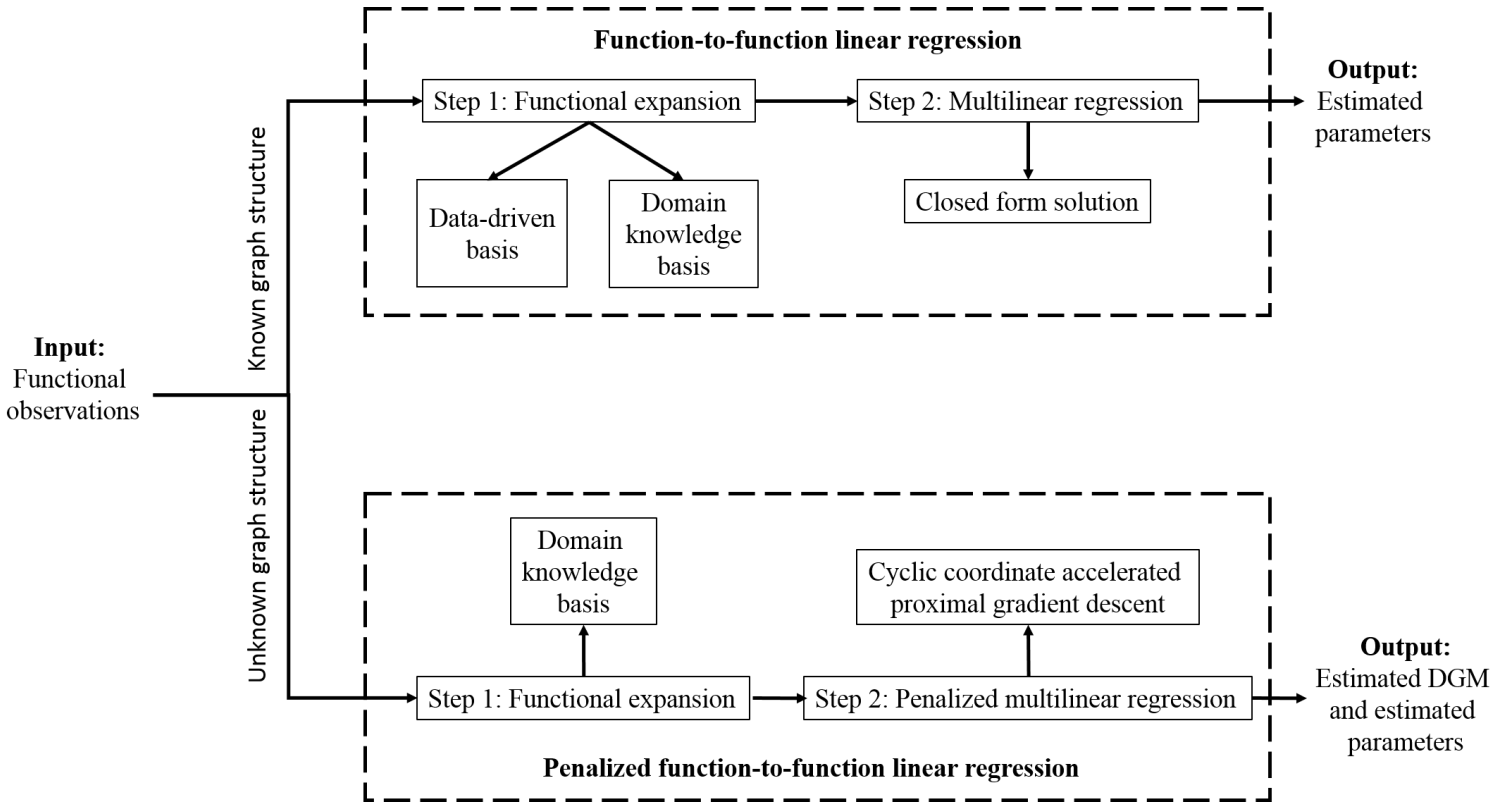


Figure 2: Flow diagram of the proposed methodology

3 Proposed Functional Directed Graphical Models

As our motivating example, we use a dataset collected during fault events of the exhaust after-treatment process of an internal combustion engine (Pacella, 2018). Variables, such as vehicle velocity, intercooler pressure, and air-to-fuel ratio (λ-upstream), are measured by D sensors mounted on the engine (Figure 1). There is information on M fault events. Let $x_{ki}(t), k = 1, \cdots, D, i = 1, \cdots, M$, be the ith observation measured by the kth sensor at time t. Then, we have M samples of independent and identically distributed multivariate functional random variables $\{\boldsymbol{x}_i(t) : t \in \Gamma, i = 1, \cdots, M\}$ where $\boldsymbol{x}_i(t) = (x_{1i}(t), x_{2i}(t), \cdots, x_{Di}(t))'$ and Γ is a compact set. Without loss of generality, we assume $\Gamma = [0, 1]$. The detailed information about this case will be given in Section 5. The main goal of this article is to develop a methodology to learn the relationship between the variables measured by the different sensors. Understanding the root-causes behind a fault event is critical to improve the performance of the engine and to control the vehicle's environmental impact. Learning

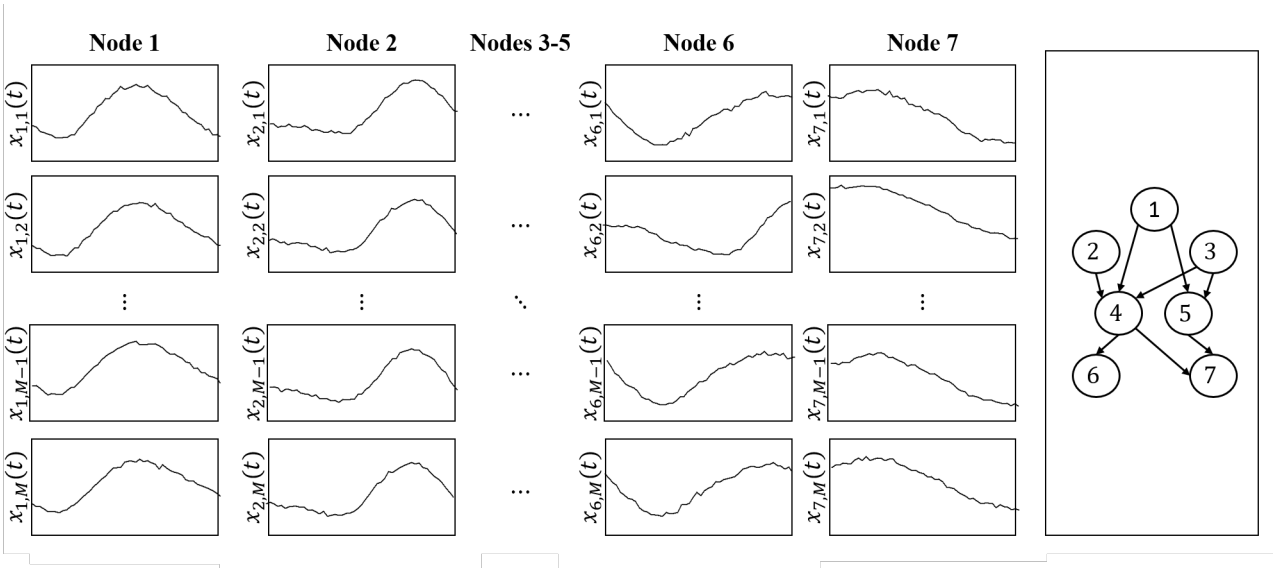


Figure 3: Illustrative example. Left: the data, $x_{ki}(t)$ for $k = 1, \cdots, 7$ nodes and $i = 1, \cdots, M$ observations. Right: The true underlying graph structure.

the relationship between the variables is equivalent to learning the underlying DGM. We begin by assuming that the structure of the graph is known, and explain how to estimate the parameters of a functional DGM. Then, we present a learning structure algorithm.

3.1 Known Graph Structure

Throughout the article, we assume that (A1) the functional variables $x_{1i}(t), x_{2i}(t), \cdots, x_{Di}(t)$ jointly follow from a D-dimensional multivariate Gaussian process, $\mathcal{G}(t)$, independently and identically, and that (A2) they can be represented as a DAG, $G = (N, A)$, with node set $N = \{1, \cdots, D\}$ and arc set A. If $(j, k) \in A$, we have that there is an arc going from node j to node k, which is equivalent to say that node j belongs to the parents' set of node k ($j \in \text{pa}_k$). In this section, we assume that A is known. Therefore, we can write the joint distribution $\mathcal{G}(t)$ as the product of the conditional distributions of each node, given the variables corresponding to its parents. That is

$$\mathcal{G}(t) = p(x_1(t), \cdots, x_D(t)) = \prod_{k=1}^{D} p(x_k(t)|\text{pa}_k). \tag{1}$$

This equation expresses the factorization properties of the joint distribution for a DGM, Bishop

(2006). The estimation of the joint distribution, $\mathcal{G}(t)$, can be simplified by estimating the parameters of the low-dimensional conditional distributions, $p(x_k(t)|\text{pa}_k)$ for $k = 1, \cdots, D$.

Figure 3 provides an illustrative example with $D = 7$. The left panel presents the data, functions $x_{ki}(t)$, where $k = 1, \cdots, 7$ and $i = 1, \cdots, M$. The right panel illustrates the conditional dependence structure of these functions, that is, the DGM. By exploring the graph structure, we see that, for example, nodes 1 and 3 are parents of nodes 4 and 5. Additionally, we have that

$$\begin{aligned} p\left(x_1(t), \cdots, x_7(t)\right) =& p\left(x_1(t)\right) \times p\left(x_2(t)\right) \times p\left(x_3(t)\right) \\ & \times p\left(x_4(t)|x_1, x_2, x_3)\right) \times p\left(x_5(t)|x_1, x_3\right) \\ & \times p\left(x_6(t)|x_4\right) \times p\left(x_7(t)|x_4, x_5\right) \end{aligned} \tag{2}$$

We can conclude that the variables 4 and 5 are conditionally independent, given the states of the variables 1 and 3. Our goal, in this section, is to take the observed functions in the left panel and estimate the parameters of the DGM in the right panel.

Beyond our motivating example, learning a functional DGM can be of interest in many other contexts. Consider a sequential manufacturing system where M samples of D process variables are measured over time. $x_{ki}(t)$ represents the performance of process variable k when producing part i at time t. This example can be modeled as a DAG since the upstream process variables can be potential parents for the downstream process variables, but the reverse cannot happen. Another example, arises in reliability systems, with D components, where $x_{ki}(t)$ represents the reliability of component k for system i at time t. The distribution of the components in the system determines the structure of the DAG.

Since the functional variables $x_{1i}(t), x_{2i}(t), \cdots, x_{Di}(t)$ jointly follow from a D-dimensional multivariate Gaussian process (MGP), $\mathcal{G}(t)$, independently and identically, and since $\mathcal{G}(t)$ can be decomposed as a product of conditional distributions following the known graph structure, we have

that the conditional distribution of $x_{ki}(t)$ can be written as

$$x_{ki}(t) \sim \mathcal{N}\left(\sum_{j \in \text{pa}_k} \int_0^1 \beta_{kj}(t,s) x_{ji}(s) \mathrm{d}s, \sigma_k^2\right) \tag{3}$$

where $\beta_{kj}(t,s)$, for $k = 1, \cdots, D$ and $j \in \text{pa}_k$, are functional parameters governing the mean and σ_k^2 is the variance of the conditional distribution for $x_{ki}(t)$. Given the conditional distribution of $x_{ki}(t)$, we can write $x_{ki}(t)$ as a function of its parents using function-to-function linear regression,

$$x_{ki}(t) = \sum_{j \in \text{pa}_k} \int_0^1 \beta_{kj}(t,s) x_{ji}(s) \mathrm{d}s + \sigma_k \epsilon_{ki}(t) \tag{4}$$

were $\epsilon_{ki}(t)$ is a standard Gaussian random variable.

Learning the parameters of the functional DGM is equivalent to estimating the parameters $\beta_{kj}(t,s), t, s \in [0,1]$, by using the M samples of the D-functional random variables, $x_{ki}(t), k = 1, \cdots, D$ and $i = 1, \cdots, M$. In practice, the function $x_{ki}(t)$ is observed over a grid of size n_k. Thus, we estimate $\beta_{kj}(t,s)$ with $\{x_{ki}(t_1), x_{ki}(t_2), ..., x_{ki}(t_{n_k})\}_{k=1,i=1}^{D,M}$. The main challenge is that the functional parameters are continuous functions with infinite-dimension. To address this issue, following Horváth and Kokoszka (2012), we assume that the parameters have a functional expansion, defined by

$$\beta_{kj}(t,s) = \sum_{p=1}^{P_k} \sum_{q=1}^{P_j} b_{kjpq} \theta_{kp}(t) \theta_{jq}(s) \tag{5}$$

where $\{\theta_{kp}(t) : p = 1, 2, \cdots, P_k \ll n_k\}$ and $\{\theta_{jq}(s) : q = 1, 2, \cdots, P_j \ll n_j\}$ are small sets of basis functions suitable for expanding $x_{ki}(t)$ and $x_{ji}(s)$, respectively. Given the basis functions, this expansion transforms the functional parameters, with infinite dimension, to a set of finite parameters b_{kjpq} that can be estimated using the training data. There are two approaches for choosing appropriate basis functions. The first one is to use data-driven basis functions, such as eigenbasis obtained by functional principal components analysis (FPCA), and the second one is to use a set of pre-specified basis functions such as splines, Fourier, or wavelets, based on the domain knowledge about the system. One of the advantages of using a functional expansion is that the functional

variables do not need to be observed with the same frequency (i.e., we can have different values for n_k, for $k = 1, \cdots, D$). We discuss both of the functional expansion approaches next.

3.1.1 FPCA Basis Functions

FPCA has been widely used for reducing the dimensionality of functional data to a small set of finite features preserving the majority of the data variability, Horváth and Kokoszka (2012). Using the eigen-decomposition of the covariance function of x_k, $\text{cov}(x_k(t), x_k(t'))$, $x_{ki}(t)$ can be written as:

$$x_{ki}(t) = \sum_{p=1}^{\infty} \xi_{kip}\theta_{kp}(t) \tag{6}$$

where $\{\theta_{kp}(t)\}$ are the eigen-functions, $\{\xi_{kip} = \langle x_{ki}(t), \theta_{kp}(t)\rangle\}$ are the FPC scores, $k = 1, \cdots, D$, and $i = 1, \cdots, M$.

If both $x_{ki}(t)$ and $x_{ji}(s)$ are expanded as in (6), by plugging (5) into (4), we have

$$\sum_{p=1}^{\infty} \xi_{kip}\theta_{kp}(t) = \sum_{j \in \text{pa}_k} \int_0^1 \sum_{p=1}^{\infty}\sum_{q=1}^{\infty} b_{kjpq}\theta_{kp}(t)\theta_{jq}(s) \sum_{q=1}^{\infty} \xi_{jiq}\theta_{jq}(s)ds + \sigma_k\epsilon_{ki}(t). \tag{7}$$

Using the orthonormality of the θ_{jq}'s, equation (7) can be reduced to

$$\sum_{p=1}^{\infty} \xi_{kip}\theta_{kp}(t) = \sum_{j \in \text{pa}_k} \sum_{p=1}^{\infty}\sum_{q=1}^{\infty} b_{kjpq}\xi_{jq}\theta_{kp}(t) + \sigma_k\epsilon_{ki}(t). \tag{8}$$

Multiplying this equation by $\theta_{kp}(t)$ and integrating over t, would result in

$$\xi_{kip} = \sum_{j \in \text{pa}_k} \sum_{q=1}^{\infty} b_{kjpq}\xi_{jiq} + \sigma_k\varepsilon_{ki} \tag{9}$$

where $\varepsilon_{ki} = \int_0^1 \theta_{kp}(t)\epsilon_{ki}(t)dt$.

The goal is to estimate the parameters b_{kjpq}. Since the FPC scores are descendingly ordered, the first few FPC scores can capture the most important information of the data and provide good

approximates. Therefore, we set $P_k \ll n_k$ and $P_j \ll n_j$ and approximate $x_{ki}(t)$ and $x_{ji}(s)$ by

$$\hat{x}_{ki}(t) = \sum_{p=1}^{P_k} \hat{\xi}_{kip}\hat{\theta}_{kp}(t) \tag{10}$$

$$\hat{x}_{ji}(s) = \sum_{q=1}^{P_j} \hat{\xi}_{jiq}\hat{\theta}_{jiq}(s). \tag{11}$$

Let $\boldsymbol{\xi}_k = [\hat{\xi}_{kip}] \in \mathbb{R}^{M\times P_k}$, $\boldsymbol{\xi}_j = [\hat{\xi}_{jiq}] \in \mathbb{R}^{M\times P_j}$, $\boldsymbol{b}_{kj} = [b_{kjpq}] \in \mathbb{R}^{P_j\times P_k}$, and $\boldsymbol{\varepsilon}_k = [\varepsilon_{kip}] \in \mathbb{R}^{M\times P_k}$, $i = 1, \cdots, M$, $p = 1, \cdots, P_k$, $q = 1, \cdots, P_j$, equation (9) can be approximated by the following multilinear regression problem,

$$\boldsymbol{\xi}_k = \sum_{j\in \text{pa}_k} \boldsymbol{\xi}_j \boldsymbol{b}_{kj} + \sigma_k \boldsymbol{\varepsilon}_k \tag{12}$$

Let $\boldsymbol{\Xi}_k = [\boldsymbol{\xi}_j]$, $j \in \text{pa}_k$, be the M by $Q_k = \sum_{j\in \text{pa}_k} P_j$ design matrix, and $\boldsymbol{B}_k = [\boldsymbol{b}_{kj}]^\top$, $j \in \text{pa}_k$, be the Q_k by P_k matrix of coefficients that should be estimated. Consequently, the matrix form of equation (12) is

$$\boldsymbol{\xi}_k = \boldsymbol{\Xi}_k \boldsymbol{B}_k + \sigma_k \boldsymbol{\varepsilon}_k \tag{13}$$

To estimate $\boldsymbol{B}_k$, we minimize the least square loss function,

$$L(\boldsymbol{B}_k) = \frac{1}{2}||\boldsymbol{\xi}_k - \boldsymbol{\Xi}_k \boldsymbol{B}_k||_2^2 \tag{14}$$

which results in a closed-form solution in the form of

$$\hat{\boldsymbol{B}}_k = (\boldsymbol{\Xi}_k^\top \boldsymbol{\Xi}_k)^{-1} \boldsymbol{\Xi}_k^\top \boldsymbol{\xi}_k. \tag{15}$$

When the truncation parameters P_k and P_j go to infinity with the sample size M, our proposed estimates $\hat{\beta}_{kj}(t, s)$ are consistent, that is

$$\lim_{M\to\infty} \int_0^1 \int_0^1 [\beta_{kj}(t, s) - \hat{\beta}_{kj}(t, s)]^2 \text{d}s\text{d}t = 0 \text{ in probabiliy}, \tag{16}$$

under some assumptions on the unknown functional parameters, $\beta_{kj}(t, s)$. This property implies

that, when the sample size is large, the parameters estimated using FPCA basis functions are close to the true parameters governing the functional DGM. The consistency of the estimators follows the proof in Yao et al. (2005).

Learning the parameters of the functional DGM, using FPCA basis functions, involves two key steps. First, for each variable k, $k = 1, \cdots, D$, we estimate the FPC scores, the computational cost associated with this step is $\mathcal{O}(Mn_k^2 + n_k^3)$. The second step is estimating the parameters $\hat{\boldsymbol{B}}_k$, for $k = 1, \cdots, D$, with the closed-form solution presented in (15). The computational cost for this step is $\mathcal{O}(MQ_k^2 + Q_k^3 + MP_kQ_k)$.

3.1.2 Pre-specified Basis Functions

In practice, sometimes, basis functions can be chosen based on the domain knowledge about the system and on the type and shape of the functional data. Examples of such basis functions include polynomials, splines, wavelets, and Fourier. Define $\{\theta_{kp}(t) : p = 1, 2, \cdots, P_k\}$ and $\{\theta_{jq}(s) : q = 1, 2, \cdots, P_j\}$ as the pre-specified basis for the functional variables $x_k(t)$ and $x_j(s)$. Let $\boldsymbol{B}_{kj} = [b_{kjpq}], 1 \leq p \leq P_k, 1 \leq q \leq P_j$ represent the P_j by P_k matrix of coefficients. The regression problem defining the functional DGM in (4) becomes

$$\boldsymbol{x}_k(t) = \sum_{j \in \text{pa}_k} \int_0^1 \boldsymbol{x}_j(s)\boldsymbol{\theta}_j^\top(s)\boldsymbol{B}_{kj}\boldsymbol{\theta}_k(t)\text{d}s + \sigma_k\boldsymbol{\epsilon}_k(t) \tag{17}$$

where $\boldsymbol{x}_k(t) = [x_{k1}(t), \cdots, x_{kM}(t)]^\top$ and $\boldsymbol{x}_j(s) = [x_{j1}(s), \cdots, x_{jM}(s)]^\top$. In order to further simplify the notation, we define $\boldsymbol{Z}_j := \int_0^1 \boldsymbol{x}_j(s)\boldsymbol{\theta}_j^\top(s)\text{d}s \in \mathbb{R}^{M \times P_j}$, $\boldsymbol{X}_k := [\boldsymbol{x}_k(t_1), \cdots, \boldsymbol{x}_k(t_{n_k})] \in \mathbb{R}^{M \times n_k}$, where $\{\boldsymbol{x}_k(t_1), \cdots, \boldsymbol{x}_k(t_{n_k})\}$ correspond to the grid of observed values for the function $x_k(t)$, $\boldsymbol{\Theta}_k := [\boldsymbol{\theta}_k(t_1), \cdots, \boldsymbol{\theta}_k(t_{n_k})] \in \mathbb{R}^{P_k \times n_k}$, and $\boldsymbol{E}_k := [\boldsymbol{\epsilon}_k(t_1), \cdots, \boldsymbol{\epsilon}_k(t_{n_k})] \in \mathbb{R}^{M \times n_k}$, which simplifies the above equation to

$$\boldsymbol{X}_k = \sum_{j \in \text{pa}_k} \boldsymbol{Z}_j\boldsymbol{B}_{kj}\boldsymbol{\Theta}_k + \sigma_k\boldsymbol{E}_k. \tag{18}$$

This can be further simplified to

$$\boldsymbol{x_k} = \sum_{j \in \text{pa}_k} \boldsymbol{\Xi}_{kj} \boldsymbol{b}_{kj} + \sigma_k \boldsymbol{e}_k, \tag{19}$$

where $\boldsymbol{x}_k = vec(\boldsymbol{X}_k^\top)$, $\boldsymbol{\Xi}_{kj} = \boldsymbol{Z}_j \otimes \boldsymbol{\Theta}_k^\top$, $\boldsymbol{b}_{kj} = vec(\boldsymbol{B}_{kj}^\top)$, and $\boldsymbol{e_k} = vec(\boldsymbol{E}_k^\top)$. Our goal is to estimate $\boldsymbol{b}_{kj}$, for all j in pa_k. To this end, we minimize the least square loss function given by

$$L(\boldsymbol{b}_{kj} | j \in \text{pa}_k) = \frac{1}{2} ||\boldsymbol{x}_k - \sum_{j \in \text{pa}_k} \boldsymbol{\Xi}_{kj} \boldsymbol{b}_{kj}||_2^2. \tag{20}$$

The problem has the closed-form solution,

$$\hat{\boldsymbol{B}}_k = (\boldsymbol{\Xi_k}^\top \boldsymbol{\Xi_k})^{-1} \boldsymbol{\Xi_k}^\top \boldsymbol{x}_k \tag{21}$$

where $\boldsymbol{\Xi}_k = [\boldsymbol{\Xi}_{kj}]$ and $\boldsymbol{B}_k = [\boldsymbol{b}_{kj}]^\top$, $j \in \text{pa}_k$. Consistency properties of this approach are not fully understood, Horváth and Kokoszka (2012), however, it gives useful estimates, which can be computed analogous to the univariate case.

Learning the parameters of the functional DGM, using pre-specified basis, involves three main steps. For each variable k, $k = 1, \cdots, D$, first, we compute the matrix $\boldsymbol{Z}_k$, the associated cost is $\mathcal{O}(Mn_kP_k)$. Second, we build the matrix $\boldsymbol{\Xi}_k$, the computational cost is $\mathcal{O}(Mn_kP_kQ_k)$, where $Q_k = \sum_{j \in \text{pa}_k} P_j$. Finally, we estimate the parameters $\hat{\boldsymbol{B}}_k$ with the closed-form solution presented in (21), the associated cost is $\mathcal{O}(Mn_kP_k^2Q_k^2 + P_k^3Q_k^3 + Mn_kP_kQ_k)$. The number of operations required can be reduced by using a functional expansion for $\boldsymbol{x}_k(t)$.

In section 4.1, we learn the parameters of a functional DGM using the functional expansion with FPCA bases and the functional expansion with pre-specified bases. We compare the two approaches by considering prediction accuracy and computational time.

3.2 Structure Learning

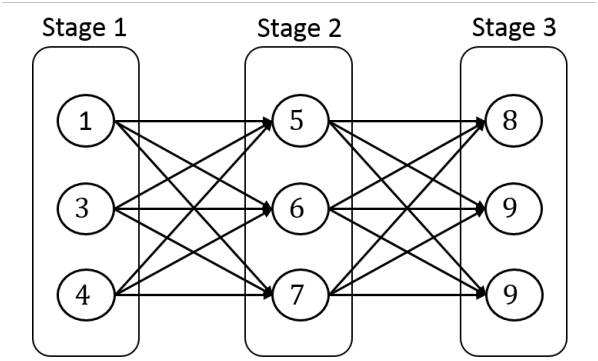


Figure 4: Illustrative example of a sequential manufacturing process.

In this section, we assume that the structure of the functional DGM is unknown. Our goal is to take the observed functions in the left panel of Figure 3 and learn the structure depicted in the right panel. The first step is to define the set of candidate parents cpa_k for each variable k, $k = 1, \cdots, D$. This step is critical as we need to guarantee that the learned DGM is, in fact, a DAG.

To avoid cycles in the DGM it is necessary to order the variables in such a way that only the variables with lower orders can be candidate parents for the variables with higher orders (i.e., for variable k, $\text{cpa}_k \subseteq \{1, \cdots, k-1\}$, $k = 1, \cdots, D$). If the system satisfies assumption (A2), this ordering exists and its definition is possible thanks to domain knowledge on the system.

We illustrate the definition of cpa_k, for $k = 1, \cdots, D$, with an example of a sequential manufacturing process. Suppose the process has three stages and three process variables per stage, as seen in Figure 4. The arcs represent the set of potential relationships between the variables in the system. Since variables 1, 2, and 3 are on the same stage and do not have any predecessor, we can conclude that they are independent. Therefore, the set of candidate parents for these variables is empty ($\text{cpa}_k = \emptyset$, for $k = 1, 2, 3$). We see that variables in stage 2 are conditionally independent from each other given the state of the variables in stage 1, thus $\text{cpa}_k = \{1, 2, 3\}$ for $k = 4, 5, 6$. Finally, we have that the variables in stage 3 are conditionally independent from each other and from the variables in stage 1, given the state of the variables in stage 2. Therefore, $\text{cpa}_k = \{4, 5, 6\}$ for $k = 7, 8, 9$. This example shows how previous domain knowledge on the system is crucial to define the set of candidate parents for each variable to guarantee that the learned DGM is a DAG.

Once the set of candidate parents for each variable in the system is defined, inspired by the neighborhood selection method, introduced by Meinshausen and Buhlmann (2006), we specify our model as a penalized function-to-function linear regression. Using the notation of Section 3.1.2, for every variable k, $k = 1, \cdots, D$, we minimize the loss function:

$$L(\boldsymbol{b}_{kj}|j \in \text{cpa}_k) = \frac{1}{2}||\boldsymbol{x}_k - \sum_{j \in \text{cpa}_k} \Xi_{kj}\boldsymbol{b}_{kj}||_2^2 + \gamma \sum_{j \in \text{cpa}_k} \sqrt{q_{kj}}||\boldsymbol{b}_{kj}||_2 + \frac{\lambda}{2} \sum_{j \in \text{cpa}_k} ||\boldsymbol{b}_{kj}||_2^2. \quad (22)$$

The first term corresponds to the least square loss function presented in Section 3.1.2. The second term is a group LASSO penalty that encourages sparsity in the model by performing variable selection (Hastie et al. (2015)), where q_{kj} is a weight representing the size of vector $\boldsymbol{b}_{kj}$. The third term is an L_2 norm penalty as used in the elastic net regularization problem (Zou and Hastie (2005)). If there is a group of highly correlated variables, the group LASSO penalty tends to select one variable and ignore the others, adding the L_2 norm penalty overcomes this limitation. For example, consider a graph with three nodes, assume that node 1 is a parent of nodes 2 and 3, and that node 2 is a parent of node 3. It is clear that nodes 1 and 2 are highly correlated. When minimizing the loss function for node 3, if the L_2 norm penalty is not considered, the group LASSO penalty will enforce sparsity and select only one of the two nodes as a parent. Finally, γ and λ are tuning parameters.

The goal of learning the structure of the functional DGM reduces to estimating $\boldsymbol{b}_{kj}$, for every node k, and $j \in \text{cpa}_k$, $k = 1, \cdots, D$. If all elements of $\boldsymbol{b}_{kj}$ are shrunk to zero, for some $j \in \text{cpa}_k$, we conclude that node j is not a parent of node k. Thanks to the assumption (A2) and to the corresponding definition of the set of candidate parents for each variable, we conclude that if variable j belongs to the estimated set of parents of variable k, then variable j causes variable k.

To learn the structure of the graphical model, we minimize the loss function, $L(\boldsymbol{b}_{kj}|j \in \text{cpa}_k)$, for every node k in the system. We adopt a cyclic coordinate accelerated proximal gradient descent algorithm, Hastie et al. (2015). First, we notice that the loss function is block coordinate separable. That is, given the group of parameters $\boldsymbol{b}_{kl}$, for all l in cpa_k, $l \neq j$, the loss function in (22) can be

reduced to

$$L(\boldsymbol{b}_{kj}) = \frac{1}{2}||\boldsymbol{r}_{kj} - \boldsymbol{\Xi}_{kj}\boldsymbol{b}_{kj}||_2^2 + \gamma\sqrt{q_{kj}}||\boldsymbol{b}_{kj}||_2 + \frac{\lambda}{2}||\boldsymbol{b}_{kj}||_2^2 + C \tag{23}$$

where $\boldsymbol{r}_{kj} = \boldsymbol{x}_k - \sum_{l\neq j}\boldsymbol{\Xi}_{kl}\boldsymbol{b}_{kl}$ is the l^{th} partial residual, and $C = \sum_{l\neq j}\sqrt{q_{kj}}||\boldsymbol{b}_{kl}||_2 + \sum_{l\neq j}||\boldsymbol{b}_{kl}||_2^2$ is a constant independent of $\boldsymbol{b}_{kj}$. To minimize the loss function, $L(\boldsymbol{b}_{kj}|j \in \text{cpa}_k)$, we repeatedly cycle through the candidate parents of node k. At the j^{th} step, we update the coefficients $\boldsymbol{b}_{kj}$ by minimizing $L(\boldsymbol{b}_{kj})$, while holding $\boldsymbol{b}_{kl}, l \neq j$, fixed at their current values.

The next step is to find an optimization algorithm to minimize the loss function for each coordinate. The proximal gradient descent (PGD) method is an optimization algorithm, focusing on minimizing the summation of a group of convex functions, some of which are not differentiable. As $L(\boldsymbol{b}_{kj})$ is the sum of $f(\boldsymbol{b}_{kj}) = \frac{1}{2}||\boldsymbol{r}_{kj} - \boldsymbol{\Xi}_{kj}\boldsymbol{b}_{kj}||_2^2 + \frac{\lambda}{2}||\boldsymbol{b}_{kj}||_2^2 + C$, which is convex and differentiable, and $g(\boldsymbol{b}_{kj}) = \gamma\sqrt{q_j}||\boldsymbol{b}_{kj}||_2$, which is convex and non-differentiable, PGD can be used to find the optimal solution through an iterative algorithm given by

$$\boldsymbol{b}_{kj}^{(t+1)} = \arg\min_{\boldsymbol{b}_{kj}}\left\{f(\boldsymbol{b}_{kj}^{(t)}) + \left\langle\nabla f(\boldsymbol{b}_{kj}^{(t)}), \boldsymbol{b}_{kj} - \boldsymbol{b}_{kj}^{(t)}\right\rangle + \frac{1}{2s^{(t)}}||\boldsymbol{b}_{kj} - \boldsymbol{b}_{kj}^{(t)}||_2^2 + g(\boldsymbol{b}_{kj})\right\} \tag{24}$$

where the super-indices (t) and $(t+1)$ denote iteration numbers, and $s^{(t)} > 0$ is a step-size parameter. At iteration t, PGD has a closed form solution as stated in the following proposition.

Proposition 1. *The proximal gradient descent algorithm, with step-size $s^{(t)}$, at iteration t, has a closed form solution in the form of a soft-tresholding function, given by (Proof in Appendix A):*

$$\begin{aligned}\boldsymbol{z}^{(t+1)} &\leftarrow \boldsymbol{b}_{kj}^{(t)} + s^{(t)}\left(\boldsymbol{\Xi}_{kj}^{\top}(\boldsymbol{r}_{kj} - \boldsymbol{\Xi}_{kj}\boldsymbol{b}_{kj}^{(t)}) - \lambda\boldsymbol{b}_{kj}^{(t)}\right)\\ \boldsymbol{b}_{kj}^{(t+1)} &\leftarrow \left(1 - \frac{s^{(t)}\gamma\sqrt{q_{kj}}}{||\boldsymbol{z}^{(t+1)}||_2}\right)_+ \boldsymbol{z}^{(t+1)}\end{aligned} \tag{25}$$

To increase the convergence speed of the optimization algorithm, we use Nesterov's accelerated PGD method, which uses weighted combinations of the current and previous gradient directions. The accelerated gradient method involves a pair of sequences $\{\boldsymbol{b}_{kj}^{(t)}\}_{t=0}^{\infty}$ and $\{\boldsymbol{\eta}_{kj}^{(t)}\}_{t=0}^{\infty}$, and some initialization $\boldsymbol{b}_{kj}^{(0)} = \boldsymbol{\eta}_{kj}^{(0)}$. For iterations $t = 1, 2, \ldots$ the solution is then updated according to the

following recursive equations:

$$\begin{aligned} \boldsymbol{z}^{(t+1)} &\leftarrow \boldsymbol{\eta}_{kj}^{(t)} + s^{(t)}\left(\boldsymbol{\Xi}_{kj}^{\top}(\boldsymbol{r}_{kj} - \boldsymbol{\Xi}_{kj}\boldsymbol{\eta}_{kj}^{(t)}) - \lambda\boldsymbol{\eta}_{kj}^{(t)}\right) \\ \boldsymbol{b}_{kj}^{(t+1)} &\leftarrow \left(1 - \frac{s^{(t)}\gamma\sqrt{q_{kj}}}{||\boldsymbol{z}^{t+1}||_2}\right)_{+} \boldsymbol{z}^{(t+1)} \\ \boldsymbol{\eta}_{kj}^{(t+1)} &\leftarrow \boldsymbol{b}_{kj}^{(t+1)} + \frac{t}{t+3}(\boldsymbol{b}_{kj}^{(t+1)} - \boldsymbol{b}_{kj}^{(t)}) \end{aligned} \tag{26}$$

Algorithm 1 summarizes the estimation procedure. This algorithm has a convergence guarantee, if the component f is continuously differentiable with a Lipschitz gradient. It is clear that $f(\boldsymbol{b}_{kj})$ is continuously differentiable, and by Proposition 2 we have that $f(\boldsymbol{b}_{kj})$ has a Lipschitz gradient, Therefore, we can conclude that the algorithm converges. Furthermore, the computational complexity for each step of the PGD method is $\mathcal{O}(Mn_kP_kQ_k)$.

Proposition 2. *$f(\boldsymbol{b}_{kj}) = \frac{1}{2}||\boldsymbol{r}_{kj} - \boldsymbol{\Xi}_{kj}\boldsymbol{b}_{kj}||_2^2 + \frac{\lambda}{2}||\boldsymbol{b}_{kj}||_2^2 + C$ has a Lipschitz gradient (i.e. there exist a constant L such that for every α, β, $||\nabla f(\alpha) - \nabla f(\beta)||_2 \leq L||\alpha - \beta||_2$). (Proof in Appendix B)*

Algorithm 1: Structure learning algorithm for functional DGM

Set a convergence threshold $\epsilon > 0$
for $k = 1$ **to** D **do**
  Initialize $\boldsymbol{b}_{kj}$ and $\boldsymbol{\eta}_{kj}$ for all j in cpa_k; set $i = 1$; let $L_1 - L_0 = \inf$ and $L_0 = \inf$
  while $|L_i - L_{i-1}| > \epsilon$ **do**
    for $j \in cpa_k$ **do**
      $\boldsymbol{r}_{kj} = \boldsymbol{x}_k - \sum_{l \in \text{cpa}_k, l \neq j} \boldsymbol{\Xi}_{kl}\boldsymbol{b}_{kl}$
      $||\boldsymbol{b}_{kj}^1 - \boldsymbol{b}_{kj}^0||_2^2 = \inf$
      $t = 1$
      while $||\boldsymbol{b}_{kj}^{(t)} - \boldsymbol{b}_{kj}^{(t-1)}||_2^2 > \epsilon$ **do**
        $\boldsymbol{z}^{(t)} = \boldsymbol{\eta}_{kj}^{(t)} + s^{(t)}(\boldsymbol{\Xi}_{kj}^{\top}(\boldsymbol{r}_{kj} - \boldsymbol{\Xi}_{kj}\boldsymbol{\eta}_{kj}^{(t)}) - \lambda\boldsymbol{\eta}_{kj}^{(t)})$
        $\boldsymbol{b}_{kj}^{(t)} = \left(1 - \frac{s^{(t)}\gamma\sqrt{q_{kj}}}{||\boldsymbol{z}^{(t)}||_2}\right)_{+} \boldsymbol{z}^{(t)}$
        $\boldsymbol{\eta}_{kj}^{(t)} = \boldsymbol{b}_{kj}^{(t)} + \frac{t}{t+3}\left(\boldsymbol{b}_{kj}^{(t)} - \boldsymbol{b}_{kj}^{(t-1)}\right)$
        $t = t + 1$
    $L_i = L(\boldsymbol{b}_{kj}^i | j \in \text{cpa}_k)$
    $i = i + 1$

The performance of the proposed method depends on the choice and number of basis functions considered. In the simulations and case study, we use B-splines as the basis functions. The selection of the basis functions depends on the functional forms of the nodes and should be done based on domain knowledge, or initial analysis. In order to learn the structure of the graph, we used a pre-specified basis, since the parameter estimation is done in an analogous way to the univariate case. However, the methodology can be extended to use a data-driven basis, as presented in section 3.1.1.

The choice of tuning parameters γ and λ can be made based on information-type criteria methods (e.g. AIC, BIC, GCV, C_p), or using cross-validation, which is a data-driven approach. In this paper, we will follow the methods proposed in Yu and Feng (2014), as they have a good performance for penalized high-dimensional linear regression models. In addition, their proposed criterion to select the optimal tuning parameters can be easily adapted to the functional setting. Specifically, we randomly split the dataset into training dataset designated by c, with size m_c, and validation dataset designated by v, with size m_v ($m_c + m_v = M$), b times. For every node k in the graph, we minimize the loss function $L(\boldsymbol{b}_{kj}|j \in \text{cpa}_k)$, for each training dataset and each combination of penalty parameters γ and λ, and obtain a model denoted by $M_{c,k}^{(\gamma,\lambda)}$ with $\tilde{\beta}_{kj}^{(\gamma,\lambda)}(t,s)$ as the least square estimates. For each split, we use the corresponding validation data set to calculate the values of the criterion function and average them across the b replicates. The modified cross-validation criterion function, adapted to the functional setting, is given by

$$L_k(\gamma,\lambda) = \frac{1}{m_v}\sum_{i\in m_v}\frac{1}{n_k}\sum_{t=1}^{n_k}(x_{ki}^{(v)}(t) - \tilde{x}_{ki}^{(c)}(t))^2 \tag{27}$$

where $\tilde{x}_{ki}^{(c)}(t) = \int_0^1 \tilde{\beta}_{kj}(t,s)x_{ji}(s)\mathrm{d}s$. This criterion is designed to remove the systematic bias introduced by the shrinkage. We find the optimal $\hat{\gamma}_k$ and $\hat{\lambda}_k$ as the penalty parameters that result in the smallest average criterion value. Finally, we fit a function-to-function regression for the model $M_k^{(\hat{\gamma}_k,\hat{\lambda}_k)}$. After cycling through all the nodes in the graph, we obtain the structure for the functional directed graphical model.

Factors	Description	Levels
Number of nodes	Number of nodes in the graph	10, 20
Density of arcs	Proportion of arcs included in the graph compared with a fully connected graph	For 10 nodes: 0.2, 0.4 For 20 nodes: 0.1, 0.2
Signal-to-noise ratio (SNR)	Ratio of variance of the response and noise term	20, 200

Table 1: Simulation factor settings

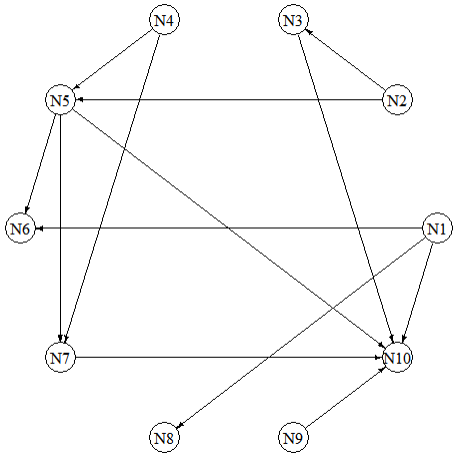
With simulations (Section 4.2) and a case study (Section 5), we test the performance of the proposed methodology to learn the structure of functional DGM.

4 Performance Evaluation via Simulations

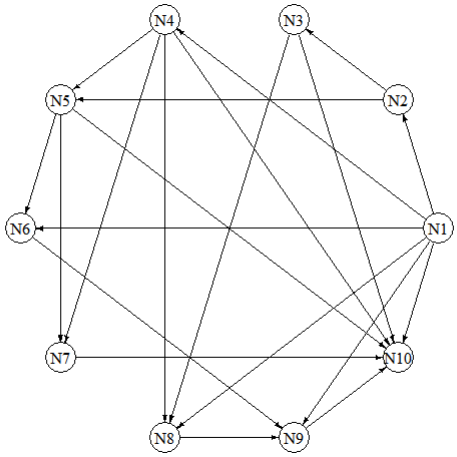
In this section, we conduct two simulation studies to evaluate the performance of the proposed methods. In the first study, we assume that the causal graph structure is known. We compare the graph parameters obtained using FPCA basis and pre-specified basis with the existing benchmark. In the second study, we evaluate the performance of our learning structure methodology when the graph structure is unknown.

4.1 Known Graph Structure

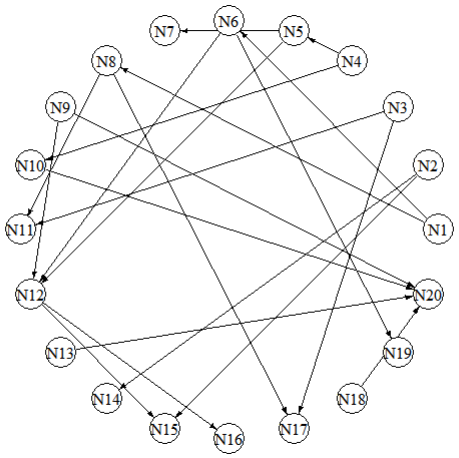
To evaluate the performance of the proposed methodology, we simulate eight different scenarios, in which we vary three different factors, as shown in Table 1. To build the different simulation scenarios, the first step is to randomly create the DAGs. The four structures used in this section are presented in Figure 5. The second step is to generate the functional random variables. The curves are produced by following the simulation study in Luo and Qi (2017). The root nodes are sampled from a Gaussian process with covariance function $\Sigma_1(t,t') = e^{-10(t-t')^2}$. Noises are added in two different ways. First, as a Gaussian process with covariance function $\Sigma_2(t,t') = e^{-0.1(t-t')^2}$, and then as white noise, using a Gaussian process with covariance function $\Sigma_3(t,t') = \sigma^2 \boldsymbol{I}$, where σ^2 is defined using the SNR established for each scenario. For each node k, that is not a root node, and for all $j \in \text{pa}_k$, we randomly generate $\beta_{kj}(s,t) = \sum_{l=1}^{3} \gamma_{lkj}(t)\phi_{lkj}(s)$ where $\gamma_{lkj}(t)$ and $\phi_{lkj}(s)$,



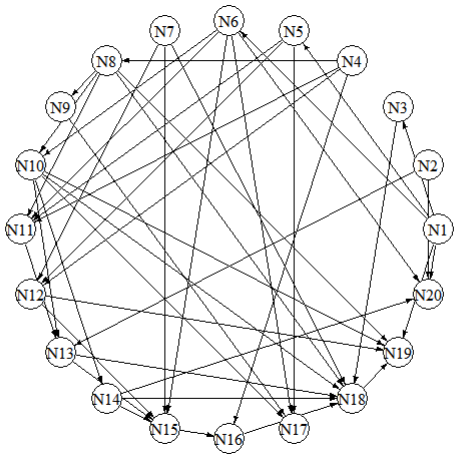
(a) 10 nodes, density 0.2



(b) 10 nodes, density 0.4



(c) 20 nodes, density 0.1



(d) 20 nodes, density 0.2

Figure 5: DAGs generated for each scenario

$l = 1, 2, 3$, are Gaussian processes with covariance function Σ_1. Finally, we generate the response curves as

$$x_k(t) = \sum_{j \in \text{pa}_k} \int \beta_{kj}(s,t) x_j(s) \mathrm{d}s + \sigma_k \epsilon_k(t) \tag{28}$$

where $\epsilon_k(t)$ is generated from a standard normal distribution, and σ_k^2 is defined by the SNR. We generate all the curves with $C_k = [0, 1]$, $k = 1, \cdots, D$, and take samples over an equidistant grid of size $n_k = 50$.

To evaluate the performance of each parameter estimation method, we generate a set of $M =$

100 samples and randomly divide the data into a training set of size 80 and a test set of size 20. We calculate the mean square prediction error (MSPE), for each node, using the testing data as follows:

$$MSPE_k = \frac{1}{M_{test}} \sum_{i=1}^{M_{test}} \frac{1}{50} \sum_{t=1}^{50} (x_{ki}(t) - \hat{x}_{ki}(t))^2. \tag{29}$$

To have an overall performance metric, we compute the average across all nodes,

$$MSPE = \frac{1}{D} \sum_{k=1}^{D} MSPE_k. \tag{30}$$

We compare the results obtained using FPCA basis (labeled as FPCR) and pre-specified basis (labeled as FDGM) with the existing benchmark proposed in Sun et al. (2017) (labeled as FGM). The loss function in Sun et al. (2017) is given by ,

$$\begin{aligned} L(b_{kj}(t)|j \in \text{pa}_k, t \in C_k) \quad = \quad & \sum_{i=1}^{M} \sum_{t \in C_k} \left(x_{ki}(t) - \sum_{j \in \text{pa}_k} x_{ji}(t) b_{kj}(t) \right)^2 \\ & + \lambda \sum_{t \in C_k} ||b_{kj}(t) - \frac{1}{n_k} \sum_{t \in C_k} b_{kj}(t)||_2^2 \end{aligned} \tag{31}$$

Each simulation scenario is replicated a thousand times, the average MSPE values and the standard errors for the proposed methods as well, as for the benchmark, are reported in Table 2. As can be seen from the table, both FPCR and FDGM outperform the benchmark FGM, consistently. The MSPE is much higher for FGM. This occurs because the benchmark method does not consider the correlation structure between different points in time of the functional nodes. Additionally, it is important to notice that FGM method requires all the functional nodes to be observed on the same grid, as $x_{ki}(t)$ depends only on $x_{ji}(t)$, for all $j \in \text{pa}_k$. This is a disadvantage when compared to our proposed methods. Another disadvantage is that FGM has a tuning parameter, and, given the nature of the penalty term, no closed-form solution exists. Therefore, the computational time of this method is considerably larger than ours, as reported in Table 3. Both FPCR and FDGM have a closed-form solution, with no tuning parameter. Furthermore, Table 3 shows that the fastest method is FDGM.

In Table 4, the three methods are compared, using the MSPE for every node, for the scenario with 10 nodes, density 0.4 (refer to Figure 5(b)), and a SNR of 200. It can be seen that, for the

Nodes	Density	SNR	FGM	FPCR	FDGM
10	0.2	200	2.1941 (0.0162)	0.0821 (0.0009)	0.0206 (0.0001)
		20	2.4270 (0.0181)	0.2527 (0.0017)	0.2031 (0.0001)
	0.4	200	4.5831 (0.0263)	0.0602 (0.0006)	0.0275 (0.0003)
		20	4.9397 (0.0295)	0.2991 (0.0029)	0.2717 (0.0027)
20	0.1	200	0.5883 (0.0034)	0.0554 (0.0004)	0.0067 (0.0000)
		20	0.6598 (0.0037)	0.1132 (0.0007)	0.0669 (0.0005)
	0.2	200	9.4871 (0.0527)	0.3361 (0.0033)	0.0700 (0.0005)
		20	9.3164 (0.0525)	0.8314 (0.0061)	0.7077 (0.0049)

Table 2: Comparison between methods using $MSPE$, 1000 simulations. Results are reported in the form of mean (standard error).

Nodes	Density	FGM	FPCR	FDGM
10	0.2	5.8875 (0.0036)	0.4214 (0.0012)	0.1408 (0.0004)
	0.4	28.9699 (0.0331)	0.4198 (0.0007)	0.2543 (0.0003)
20	0.1	13.8311 (0.0121)	0.8421 (0.0012)	0.1438 (0.0003)
	0.2	17.7029 (0.0162)	0.8381 (0.0010)	0.5594 (0.0005)

Table 3: Comparison between methods in terms of computational time in seconds, 100 simulations. Results are reported in the form of mean (standard error).

down-stream variables (i.e. higher-numbered nodes), the corresponding $MSPE_k$ increases. This is due to the error propagation from the up-stream predictions to the down-stream predictions. From Tables 2, 3, and 4, it is clear that FDGM is consistently the best method.

	Node 2	Node 3	Node 4	Node 5	Node 6
FGM	0.3906 (0.0015)	0.0457 (0.0002)	0.0866 (0.0005)	0.1352 (0.0007)	1.7813 (0.0094)
FPCR	0.0082 (0.0001)	0.0011 (0.0000)	0.0089 (0.0001)	0.0036 (0.0000)	0.0129 (0.0001)
FDGM	0.0019 (0.0000)	0.0009 (0.0000)	0.0004 (0.0000)	0.0022 (0.0000)	0.0054 (0.0001)
	Node 7	Node 8	Node 9	Node 10	
FGM	0.0207 (0.0001)	0.5306 (0.0030)	8.4657 (0.0369)	29.791 (0.1840)	
FPCR	0.0026 (0.0003)	0.1785 (0.0017)	0.1721 (0.0014)	0.1542 (0.0016)	
FDGM	0.0002 (0.0000)	0.0035 (0.0000)	0.1269 (0.0013)	0.1059 (0.0012)	

Table 4: Comparison between methods using $MSPE_k$, for the scenario with 10 nodes, density 0.4, and signal-to-noise-ratio 200, 1000 simulations. Results are reported in the form of mean (standard error).

4.2 Structure Learning

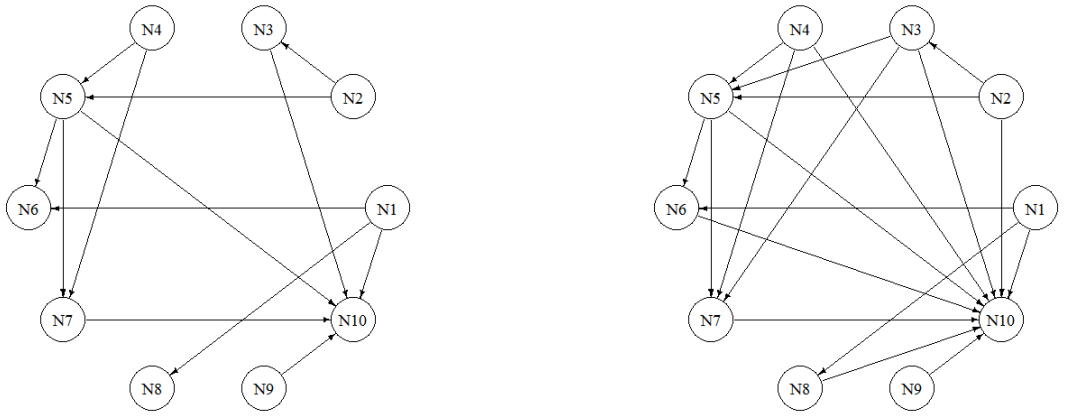
In this simulation study, we evaluate the proposed structure learning method using a graphical model with 10 nodes and a density of 0.2 (Figure 6 (a)). We consider two different SNR: 200 and 20. The signals are generated as in the previous section. To learn the structure, we use cubic B-splines with 8 knots as basis functions. We select γ from $\{0, 0.2, 0.4, 0.6, 0.8, 1, 3, 5, 7, 9, 11, 13, 15\}$ and λ from $\{10^{-10}, 10^{-8}, 10^{-6}, 10^{-4}, 10^{-2}\}$. We use $m_v = M^{0.7}$ and $b = 5$. As mentioned earlier, we assume that we have a directed acyclic graph with ordered nodes such that each child node has a higher number than its parents. Therefore, we consider 1 as the root node, and learn the structure in an orderly fashion for nodes 2 through 10. For node k, the set cpa_k is equal to $\{1, \ldots, k-1\}$.

The true model and the learned structure over a thousand simulation experiments are given in Figure 6. For the two different simulation scenarios with different SNRs (i.e. 200, 20), the learned structures coincide. This implies that the noise of the signals does not affect the structure learned. However, the mean square prediction errors and their standard errors are different, as observed in Table 5. As expected, when the noise increases, the SNR decreases, and, therefore, the prediction task is harder.

To evaluate the recall and precision of the method, the confusion matrix is computed in Table 6. We observe that the proposed method has a true positive rate of 100%. However, the learned structure has additional arcs. This is because the group LASSO penalty tends to select more variables. As an overall performance measure, we use the F_β-score, which is a harmonic mean of precision and recall. We have:

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}} \tag{32}$$

where precision $= TP/(TP + FP)$, recall $= TP/(TP + FN)$, and β is a coefficient that determines the weight assigned to precision and recall. TP, FP, and FN stand for true positive, false positive, and false negative, respectively. If we assign the same weight to precision and recall, the F_1-score is 81%. However, in many applications, having a false negative is worse than a false positive. In health prediction, for example, it is preferable to order further analysis for a healthy



(a) True structure

(b) Learned structure

Figure 6: True structure and structure learned for a functional DGM with 10 nodes, density 0.2, and signal-to-noise ratio 200/20.

	MSPE	St. Error
SNR 200	0.13017	0.0049
SNR 20	0.25385	0.0053

Table 5: Mean square prediction error and standard error for functional DGM with 10 nodes, density 0.2, and SNR 200/20

patient that is believed to have a disease (false positive), than to send a sick patient home with no treatment (false negative). Similarly, in a manufacturing plant, it is preferable to have a false alarm (false positive) when predicting the quality of a product, than to mistakenly sell defective items (false negatives). In a scenario in which recall is twice as important as precision, the F_2-score is 92%.

The computational time to learn the parents of each node is presented in Table 7. We can see that the computational time highly depends on the number of candidate parents for each node. Furthermore, it is important to notice that the set of parents for each node can be estimated in parallel which speeds-up the process of learning the functional DGM.

		Predicted Model	
		True	False
True Model	True	13	0
	False	6	26

Table 6: Confusion matrix for a DGM with 10 nodes, density 0.2, and signal-to-noise-ratio 200/20.

	Average Time (s.)	Standard Error (s.)
Node 2	131.61	1.82
Node 3	545.06	6.85
Node 4	540.78	7.15
Node 5	1,219.67	15.30
Node 6	1,823.94	22.85
Node 7	2,501.29	32.43
Node 8	2,878.99	38.19
Node 9	4,810.76	71.78
Node 10	3,798.18	50.04

Table 7: Computational time to establish the parents of each node in the graph for a functional DGM with 10 nodes and density 0.2.

5 Case Study

In this section, we illustrate how our method for learning a functional graphical model can be applied to real data. We focus on root-cause analysis for the internal combustion engine case study described earlier. An internal combustion engine produces gas with polluting substances such as nitrogen oxides (NO_x). Gas emission control regulations have been set up to protect the environment and are becoming increasingly restrictive. To fulfill legislation requirements, a higher number of on-board sensors is needed to monitor the performance of the combustion and the exhaust gas after-treatment process. The compliance of combustion engines with emission regulations demands more efficient and reliable emission control systems.

The NO_x Storage Catalyst (NSC) is an exhaust after-treatment system by which the exhaust gas is treated after the combustion process in two alternating phases: adsorption (molecules of NO_x in the exhaust gas are captured by an adsorber), and regeneration (the stored NO_x is reduced in a catalytic process). The regeneration phase starts when the NO_x adsorber is saturated. During the regeneration phase, of duration ranging between 30 and 90 seconds, the engine control unit is

programmed to maintain the combustion process in a rich air-to-fuel status. This status is related to the amount of oxygen present during the combustion process. The relative air-to-fuel ratio normalized by stoichiometry (λ-upstream), which is measured upstream of the NSC, is the indicator of a correct regeneration phase. During regeneration, λ-upstream should assume values in the interval [0.92,0.95]. However, faults occur, they are detected by a λ-upstream value falling bellow an acceptability threshold of 0.9. This kind of fault, which is called λ-undershoot, worsens the NSC performance during the regeneration phase. Although a λ-undershoot fault can be easily detected by monitoring the λ-upstream sensor, the root-causes behind this behavior may change and are not clearly defined.

Pacella (2018) developed a methodology of unsupervised classification for the profile data obtained, under real driving conditions, by on-board sensors (channels) during λ-undershoot fault events. Based on the clusters found, field experts analyze cluster patterns to further understand the root-cause behind a fault event.

In this section, we learn the structure of the network of on-board sensors, for different clusters, to identify the cause of λ-undershoot. The signals of 12 on-board sensors (Table 8) are available for two clusters, cluster A has 20 fault events, and cluster B has 5 fault events. All signals are measured over a two-second interval with a sample rate of 100Hz. The signals for each channel in clusters A and B can be seen in Figure 7. From the signals in cluster A, the experts noted that the velocity is constant (Ch12) and that there is no change in the accelerator pedal position (Ch02) or in the gear index (Ch11). Furthermore, throttle valve (Ch10) is constantly opened for air intake, and Exhausted Gas Recirculated (EGR) valve (Ch03) is actuated to bring a portion of the exhausted gas back to the cylinders to reduce the temperature. For experts, this clearly represents a driving condition in which no additional power and torque are required to the engine during the regeneration phase. On the other hand, for cluster B, experts concluded that the fault occurred during an acceleration phase of the vehicle, recognized by sudden changes in the throttle valve (Ch10). As a consequence of the acceleration phase, the fluctuations of the intake air mass (Ch01) during the NSC regeneration phase are higher and the EGR valve (Ch03) is not active. Additionally, the λ-upstream value

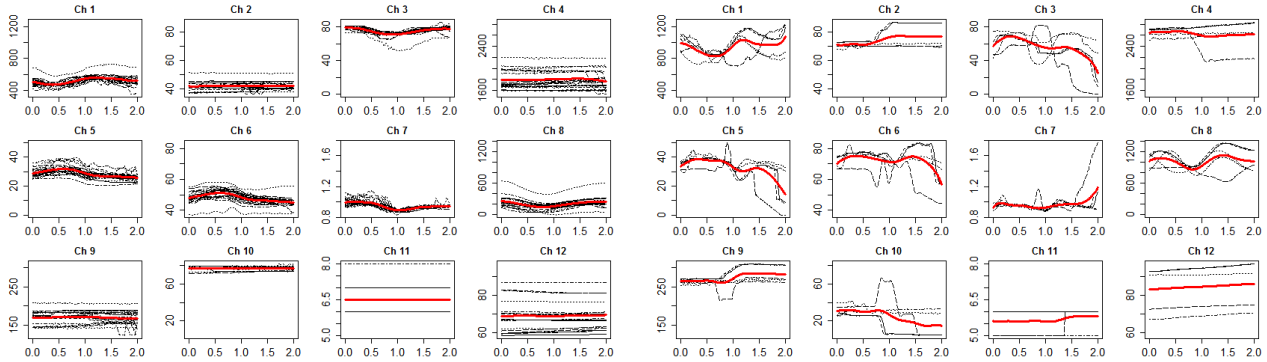
(Ch07) presents an increasing trend in the final part of the window. This indicates the end of the NSC regeneration phase, which is due to changes in the engine operation conditions, such as a reduction of the amount of injected fuel charge in the second pre-injection (Ch05) and higher engine rotational speed (Ch04) and downstream intercooler pressure (Ch08), Pacella (2018).

Using our proposed method, we learned the structure for the two available clusters. The structures obtained, after 100 replications of the learning process, can be observed in Figure 8. For the case study, the set of candidate parents for each node was proposed by a group of field experts. We used cubic B-splines with 8 knots as basis functions, and selected λ from $\{10^{-12}, 10^{-8}, 10^{-4}\}$ and γ from $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\}$. We used $m_v = M^{0.7}$ and $b = 5$. We can observe that the structures obtained support the observations of the experts. For cluster A, the conclusion is that the λ-upstream sensor (Ch07) has no parents. This follows from the fact that most of the channels in this cluster have a constant behavior, as cluster A refers to a stationary mode of the Diesel engine without acceleration. In cluster B, the λ-upstream sensor (Ch07) has four parents: air aspirated per cylinder (Ch01), engine rotational speed (Ch04), amount of injected fuel charge in the second pre-injection (Ch05), and downstream intercooler pressure (Ch08). The structure learned agrees with the observations made by the experts, the fault is generated by changes produced in the parent nodes of the sensor due to an acceleration during the ending of the NSC regeneration phase. In addition, most of the arcs are the same for both clusters. This makes sense as both structures represent the relationship of different sensors in a car. However, the difference in λ-undershoot root-cause is detected as the parents for the λ-sensor (Ch07) change. The structures learned detect that, for cluster A, the fault is due to a poor performance of the NSC controller or sensor, while, in the case of cluster B, it is due to the dynamics of the engine operation, rather than to the NSC efficiency.

To further study the performance of the proposed method, for each one of the 100 replications, we randomly divided the data into a training set, with 80% of the signals, and a test set, with 20% of the signals. We compute the $MSPE_k$ for each variable k, $k = 1, \cdots, 12$, for each cluster. The results are reported in Table 9. We observe that the prediction is fairly accurate in both cases. However, we obtain better results for Cluster A as we have more observations in this cluster, and,

#	Description	Label	Unit
1	air aspirated per cylinder	Ch01	[mg/s]
2	accelerator pedal position	Ch02	-
3	low pressure EGR valve	Ch03	-
4	engine rotational speed	Ch04	[rpm]
5	fuel in the 2nd pre-injection	Ch05	[mg/s]
6	total quantity of fuel injected	Ch06	[mg/s]
7	lambda upstream NSC	Ch07	-
8	down-stream intercooler pressure	Ch08	[mbar]
9	inner torque	Ch09	[Nm]
10	aperture ratio of inlet valve	Ch10	-
11	gear index	Ch11	-
12	vehicle velocity	Ch12	[km/h]

Table 8: List of on-board sensors (channels)



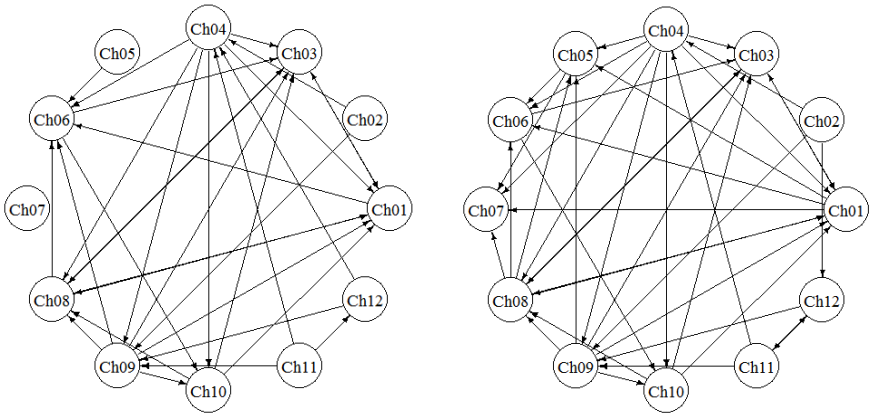
(a) Cluster A (b) Cluster B

Figure 7: 12 channels for clusters A and B. Each panel depicts the signals (black), and the mean signal (red)

therefore, more information.

6 Conclusion

This paper proposed a novel method to learn functional DGM, while taking into account the correlation structure between functional variables over time. First, we presented two methods to learn the parameters of a functional DGM when the structure is known. Both are based on function-to-function linear regression. In order to evaluate their performances, we conducted a simulation study with eight different scenarios. We compared the performance of the proposed methods with a



(a) Cluster A (b) Cluster B

Figure 8: Structure learned for clusters A and B

Label	Description	Cluster A	Cluster B
Ch01	air aspirated per cylinder	0.0014 (0.0002)	0.0067 (0.0008)
Ch03	low pressure EGR valve	0.0014 (0.0002)	0.0387 (0.0026)
Ch04	engine rotational speed	0.0019 (0.0003)	0.0046 (0.0003)
Ch05	fuel in the 2nd pre-injection	0.0057 (0.0003)	0.0263 (0.0013)
Ch06	total quantity of fuel injected	0.0004 (0.0000)	0.0057 (0.0002)
Ch07	lambda upstream NSC	0.0005 (0.0000)	0.0025 (0.0002)
Ch08	downstream intercooler pressure	0.0018 (0.0001)	0.0164 (0.0008)
Ch09	inner torque	0.0005 (0.0000)	0.0006 (0.0000)
Ch10	aperture ratio of inlet valve	0.0004 (0.0000)	0.0131 (0.0014)
Ch11	gear index	0.0056 (0.0003)	0.0033 (0.0003)
Ch12	vehicle velocity	0.0083 (0.0005)	0.0015 (0.0000)

Table 9: $MSPE_k$, $k = 1, \cdots, 12$, for clusters A and B, over 100 replicates. Results are reported in the form of mean (standard error).

method presented in Sun et al. (2017). The mean square prediction errors for the proposed methods were consistently smaller. Another advantage of the proposed methods is that they have a closed-form solution, therefore, they are computationally more efficient than the benchmark. In a second stage, we extended the methodology to the case when the structure of the graph is unknown. A learning structure algorithm was presented. The parents of every node in the graph are selected from a set of candidate parents, by iteratively fitting penalized function-to-function linear regressions. The loss function used includes a group LASSO penalty, for variable selection, and an L_2 penalty to handle group selection of correlated nodes. The cyclic coordinate accelerated proximal gradient descent algorithm was employed to find the optimal model, and to learn the parameters. In the simulation study, we saw that our method is able to learn a structure with a recall of 100%. We obtained an F_1-score of 81%. If we consider that in many real-life situations, a false negative is worse than a false positive, the method performance improves, as the F_2-score is 92%. In the case study, we proved that the proposed method is able to detect different root-causes of λ-undershoot.

In this paper, we assumed that (A2) the functional variables can be represented as a DAG. This assumption limits the size of the system of study. To learn the structure of the graph, it is necessary to order the variables in the system in such a way that the lower ordered variables can be candidate parents for the higher-ordered variables, but the reverse cannot happen. This ordering is based on domain knowledge of the system and could be difficult to achieve when the number of variables is large. If previous domain knowledge on the system is unavailable to create a topological ordering of the variables, the task of learning a DGM becomes a challenging problem. It is possible to define $\text{cpa}_k = N \setminus \{k\}$, for $k = 1, \cdots, D$, where N is the set of nodes. However, this strategy is computationally expensive and allows for undirected edges and cycles in the model. Therefore, the learned structure is not guaranteed to be a DAG, and the causality relationship between variables could be lost. Solutions to deal with undirected edges remain to be investigated.

References

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg.

Cao, X., Sandstede, B., and Luo, X. (2019). A functional data method for causal dynamic network modeling of task-related fmri. *Frontiers in Neuroscience*, 13(127):1–19.

Hastie, T., Tibshirani, R., and Wainwright, M. (2015). *Statistical Learning with Sparsity: The Lasso and Generalizations*. Chapman & Hall/CRC.

Horváth, L. and Kokoszka, P. (2012). *Inference for functional data with applications*, volume 200. Springer Science & Business Media.

Langseth, H. and Portinale, L. (2007). Bayesian networks in reliability. *Reliability Engineering & System Safety*, 92(1):92 – 108.

Li, B. and Solea, E. (2018). A nonparametric graphical model for functional data with application to brain networks based on fmri. *Journal of the American Statistical Association*, 113(524):1637–1655.

Li, J. and Shi, J. (2007). Knowledge discovery from observational data for process control using causal bayesian networks. *IIE Transactions*, 39(6):681–690.

Lindquist, M. (2012). Functional causal mediation analysis with an application to brain connectivity. *Journal of the American Statistical Association*, 107(500):1297–1309.

Luo, R. and Qi, X. (2017). Function-on-function linear regression by signal compression. *Journal of the American Statistical Association*, 112(518):690–705.

Meinshausen, N. and Buhlmann, P. (2006). High-dimensional graphs and variable selection with the lasso. *The Annals of Statistics*, 34(3):1436–1462.

Pacella, M. (2018). Unsupervised classification of multichannel profile data using pca: An application to an emission control system. *Computers and Industrial Engineering*, 122:161 – 169.

Pourret, O., Naim, P., and Marcot, B. (2008). *Bayesian Networks. A Practical Guide to Applications*. Wiley.

Qiao, X., Guo, S., and James, G. M. (2019). Functional graphical models. *Journal of the American Statistical Association*, 114(525):211–222.

Reisi Gahrooei, M., Paynabar, K., Pacella, M., and Shi, J. (2020). Process modeling and prediction with large number of high-dimensional variables using functional regression. *IEEE Transactions on Automation Science and Engineering*, 17(2):684–696.

Sun, H., Huang, S., and Jin, R. (2017). Functional graphical models for manufacturing process modeling. *IEEE Transactions on Automation Science and Engineering*, 14(4):1612–1621.

Yao, F., Muller, H. G., and Wang, J. L. (2005). Functional linear regression analysis for longitudinal data. *The Annals of Statistics*, 33(6):2873–2903.

Yu, Y. and Feng, Y. (2014). Modified cross-validation for penalized high-dimensional linear regression models. *Journal of Computational and Graphical Statistics*, 23(4):1009–1027.

Zhu, H., Strawn, N., and Dunson, D. B. (2016). Bayesian graphical models for multivariate functional data. *Journal of Machine Learning Research*, 17(204):1–27.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B*, 67:301–320.

Appendix A Proof of Proposition 2: Proximal Gradient Descent Closed Form Solution

Proposition 2. *The proximal gradient descent, with step-size $s^{(t)}$, at iteration t, presented in section 3.2, consists of the following two steps:*

$$\begin{aligned} \boldsymbol{z}^{(t+1)} &\leftarrow \boldsymbol{b}_{kj}^{(t)} + s^{(t)} \left(\boldsymbol{\Xi}_{kj}^{\top} (\boldsymbol{r}_{kj} - \boldsymbol{\Xi}_{kj} \boldsymbol{b}_{kj}^{(t)}) - \lambda \boldsymbol{b}_{kj}^{(t)} \right) \\ \boldsymbol{b}_{kj}^{(t+1)} &\leftarrow \left(1 - \frac{s^{(t)} \gamma \sqrt{q_{kj}}}{||\boldsymbol{z}^{(t+1)}||_2} \right)_{+} \boldsymbol{z}^{(t+1)} \end{aligned}$$

Proof. We want to solve:

$$\boldsymbol{b}_{kj}^{(t+1)} = \arg\min_{\boldsymbol{b}_{kj}} \left\{ f(\boldsymbol{b}_{kj}^{(t)}) + \left\langle \nabla f(\boldsymbol{b}_{kj}^{(t)}), \boldsymbol{b}_{kj} - \boldsymbol{b}_{kj}^{(t)} \right\rangle + \frac{1}{2s^{(t)}} ||\boldsymbol{b}_{kj} - \boldsymbol{b}_{kj}^{(t)}||_2^2 + g(\boldsymbol{b}_{kj}) \right\}$$

We define the proximal map of a convex function g, as:

$$\text{prox}_g(z) := \arg\min_{\theta} \left\{ \frac{1}{2} ||z - \theta||_2^2 + g(\theta) \right\}$$

We will show that the update has the equivalent representation:

$$\boldsymbol{b}_{kj}^{(t+1)} = \text{prox}_{s^{(t)} g} \left(\boldsymbol{b}_{kj}^{(t)} - s^{(t)} \nabla f(\boldsymbol{b}_{kj}^{(t)}) \right)$$

We have:

$$\begin{aligned} \boldsymbol{b}_{kj}^{(t+1)} &= \text{prox}_{s^{(t)} g} \left(\boldsymbol{b}_{kj}^{(t)} - s^{(t)} \nabla f(\boldsymbol{b}_{kj}^{(t)}) \right) \\ &= \arg\min_{\boldsymbol{b}_{kj}} \left\{ \tfrac{1}{2} ||\boldsymbol{b}_{kj}^{(t)} - s^{(t)} \nabla f(\boldsymbol{b}_{kj}^{(t)}) - \boldsymbol{b}_{kj}||_2^2 + s^{(t)} g(\boldsymbol{b}_{kj}) \right\} \\ &= \arg\min_{\boldsymbol{b}_{kj}} \left\{ \tfrac{1}{2s^t} ||\boldsymbol{b}_{kj}^{(t)} - s^{(t)} \nabla f(\boldsymbol{b}_{kj}^{(t)}) - \boldsymbol{b}_{kj}||_2^2 + g(\boldsymbol{b}_{kj}) \right\} \\ &= \arg\min_{\boldsymbol{b}_{kj}} \left\{ \nabla f(\boldsymbol{b}_{kj}^{(t)})^{\top} \nabla f(\boldsymbol{b}_{kj}^{(t)}) + \left\langle \nabla f(\boldsymbol{b}_{kj}^{(t)}), \boldsymbol{b}_{kj} - \boldsymbol{b}_{kj}^{(t)} \right\rangle + \tfrac{1}{2s^{(t)}} ||\boldsymbol{b}_{kj} - \boldsymbol{b}_{kj}^{(t)}||_2^2 + g(\boldsymbol{b}_{kj}) \right\} \\ &\Leftrightarrow \arg\min_{\boldsymbol{b}_{kj}} \left\{ f(\boldsymbol{b}_{kj}^{(t)}) + \left\langle \nabla f(\boldsymbol{b}_{kj}^{(t)}), \boldsymbol{b}_{kj} - \boldsymbol{b}_{kj}^{(t)} \right\rangle + \tfrac{1}{2s^{(t)}} ||\boldsymbol{b}_{kj} - \boldsymbol{b}_{kj}^{(t)}||_2^2 + g(\boldsymbol{b}_{kj}) \right\} \end{aligned}$$

Using the proximal gradient descent method, first, we take a gradient step:

$$\boldsymbol{z}^{(t)} = \boldsymbol{b}_{kj}^{(t)} - s^{(t)} \nabla f(\boldsymbol{b}_{kj}^{(t)}) = \boldsymbol{b}_{kj}^{(t)} - s^{(t)} \left(-\boldsymbol{\Xi}_{kj}^{T} (\boldsymbol{r}_{kj} - \Xi_{kj} \boldsymbol{b}_{kj}^{(t)}) + \lambda \boldsymbol{b}_{kj}^{(t)} \right)$$

Second, we update the parameters:

$$\begin{aligned} \boldsymbol{b}_{kj}^{(t+1)} &= \text{prox}_{s^{(t)} g}(\boldsymbol{z}^{(t+1)}) \\ &= \arg\min_{\boldsymbol{b}_{kj}} \tfrac{1}{2s^{(t)}} ||\boldsymbol{z}^{(\boldsymbol{t+1})} - \boldsymbol{b}_{kj}||_2^2 + g(\boldsymbol{b}_{kj}) \\ &= \arg\min_{\boldsymbol{b}_{kj}} \tfrac{1}{2s^{(t)}} ||\boldsymbol{z}^{(\boldsymbol{t+1})} - \boldsymbol{b}_{kj}||_2^2 + \gamma \sqrt{q_{kj}} ||\boldsymbol{b}_{kj}||_2 \end{aligned}$$

We need to solve the previous optimization problem. Let:

$$h(\boldsymbol{b}_{kj}) = \frac{1}{2s^{(t)}} ||\boldsymbol{z}^{(t+1)} - \boldsymbol{b}_{kj}||_2^2 + \gamma \sqrt{q_{kj}} ||\boldsymbol{b}_{kj}||_2$$

Since, h is not differentiable at $\boldsymbol{b}_{kj} = 0$, we need to use sub-gradients. We have that:

$$\partial h(\boldsymbol{b}_{kj}) = \frac{1}{s^{(t)}} (\boldsymbol{b}_{kj} - \boldsymbol{z}^{(t)+1}) + \gamma \sqrt{q_{kj}} \partial ||\boldsymbol{b}_{kj}||_2$$

To minimize h, we need to solve:

$$0 \in \partial h(\boldsymbol{b}_{kj}) \Leftrightarrow \frac{\boldsymbol{z}^{(t+1)} - \boldsymbol{b}_{kj}}{s^{(t)} \gamma \sqrt{q_{kj}}} \in \partial ||\boldsymbol{b}_{kj}||_2$$

We know that $\partial ||\boldsymbol{b}_{kj}||_2 = \boldsymbol{b}_{kj} / ||\boldsymbol{b}_{kj}||_2$ if $\boldsymbol{b}_{kj} \neq 0$ and $||\boldsymbol{b}_{kj}||_2 \leq 1$ if $\boldsymbol{b}_{kj} = 0$. From this, we need to consider two cases:

- $\boldsymbol{b}_{kj} \neq 0 \Rightarrow \partial ||\boldsymbol{b}_{kj}||_2 = \boldsymbol{b}_{kj} / ||\boldsymbol{b}_{kj}||_2$

  We need to solve:

$$\frac{\boldsymbol{z}^{(t+1)} - \boldsymbol{b}_{kj}}{s^{(t)} \gamma \sqrt{q_{kj}}} = \frac{\boldsymbol{b}_{kj}}{||\boldsymbol{b}_{kj}||_2} \Leftrightarrow \boldsymbol{b}_{kj} = \frac{||\boldsymbol{b}_{kj}||_2}{||\boldsymbol{b}_{kj}||_2 + s^{(t)} \gamma \sqrt{q_{kj}}} \boldsymbol{z}^{(t+1)}$$

Since $||\boldsymbol{b}_{kj}||_2 > 0$ and $s^{(t)}\gamma\sqrt{q_{kj}} \geq 0$, we have that $\boldsymbol{b}_{kj} = a\boldsymbol{z}^{(t+1)}$, where a is a positive constant. We have that:

$$a = \frac{a||\boldsymbol{z}^{(t+1)}||_2}{a||\boldsymbol{z}^{(t+1)}||_2 + s^{(t)}\gamma\sqrt{q_{kj}}} \Rightarrow a = 1 - \frac{s^{(t)}\gamma\sqrt{q_{kj}}}{||\boldsymbol{z}^{(t+1)}||_2}$$

Since $a > 0$, we must have $||\boldsymbol{z}^{(t+1)}||_2 > s^{(t)}\gamma\sqrt{q_{kj}}$.

- $\boldsymbol{b}_{kj} = 0$

If $\boldsymbol{b}_{kj} = 0$ then $\boldsymbol{z}^{(t+1)}/(s^{(t)}\gamma\sqrt{q_{kj}}) \in \partial||\boldsymbol{b}_{kj}||_2$. Therefore, $||\boldsymbol{z}^{(t+1)}/(s^{(t)}\gamma\sqrt{q_{kj}})||_2 \leq 1$. This implies that $||\boldsymbol{z}^{(t+1)}||_2 \leq s^{(t)}\gamma\sqrt{q_{kj}}$.

We can conclude that the optimal solution to the problem is:

$$\boldsymbol{b}_{kj}^{(t+1)} = \left(1 - \frac{s^{(t)}\gamma\sqrt{q_{kj}}}{||\boldsymbol{z}^{(t+1)}||_2}\right)_+ \boldsymbol{z}^{(t+1)}$$

Therefore, the updates for the proximal gradient descent are:

$$\begin{aligned}
\boldsymbol{z}^{(t+1)} &\leftarrow \boldsymbol{b}_{kj}^{(t)} + s^{(t)}\left(\boldsymbol{\Xi}_{kj}^{\top}(\boldsymbol{r}_{kj} - \boldsymbol{\Xi}_{kj}\boldsymbol{b}_{kj}^{(t)}) - \lambda\boldsymbol{b}_{kj}^{(t)}\right) \\
\boldsymbol{b}_{kj}^{(t+1)} &\leftarrow \left(1 - \frac{s^{(t)}\gamma\sqrt{q_{kj}}}{||\boldsymbol{z}^{(t+1)}||_2}\right)_+ \boldsymbol{z}^{(t+1)}
\end{aligned}$$

□

Appendix B Proof of Proposition 3: Convergence of Proximal Gradient Descent Algorithm

Proposition 3. *$f(\boldsymbol{b}_{kj}) = \frac{1}{2}||\boldsymbol{r}_{kj} - \boldsymbol{\Xi}_{kj}\boldsymbol{b}_{kj}||_2^2 + \frac{\lambda}{2}||\boldsymbol{b}_{kj}||_2^2 + C$ has a Lipschitz gradient.*

Proof. Let $\alpha,\ \beta \in \mathbb{R}^{q_{kj}}$,

$$\begin{aligned}
||\nabla f(\alpha) - \nabla f(\beta)||_2 &= || - \boldsymbol{\Xi}_{kj}^{\top}(\boldsymbol{r}_{kj} - \boldsymbol{\Xi}_{kj}\alpha) + \lambda\alpha + \boldsymbol{\Xi}_{kj}^{\top}(\boldsymbol{r}_{kj} - \boldsymbol{\Xi}_{kj}\beta) - \lambda\beta)||_2 \\
&= ||(\boldsymbol{\Xi}_{kj}^{\top}\boldsymbol{\Xi}_{kj} + \lambda I)(\alpha - \beta)||_2 \\
&\leq \theta_{\max}(\boldsymbol{\Xi}_{kj}^{\top}\boldsymbol{\Xi}_{kj} + \lambda I)||\alpha - \beta||_2 \\
&\leq L||\alpha - \beta||_2
\end{aligned}$$

where I is the identity matrix with same dimensions as $\boldsymbol{\Xi}_{kj}^{\top}\boldsymbol{\Xi}_{kj}$, and $\theta_{\max}(\boldsymbol{\Xi}_{kj}^{\top}\boldsymbol{\Xi}_{kj} + \lambda I)$ is the maximum eigenvalue of $\boldsymbol{\Xi}_{kj}^{\top}\boldsymbol{\Xi}_{kj} + \lambda I$. Therefore, f has a Lipschitz gradient with L the maximum eigenvalue of $\boldsymbol{\Xi}_{kj}^{\top}\boldsymbol{\Xi}_{kj} + \lambda I$. □