MODELING, MONITORING, AND DIAGNOSIS OF COMPLEX SYSTEMS WITH HIGH-DIMENSIONAL STREAMING DATA

A Dissertation
Presented to
The Academic Faculty

By

Ana María Estrada Gómez

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Industrial and Systems Engineering

Georgia Institute of Technology

August 2021

MODELING, MONITORING, AND DIAGNOSIS OF COMPLEX SYSTEMS WITH HIGH-DIMENSIONAL STREAMING DATA

Date approved: July 14, 2021

So we beat on, boats against the current, borne back ceaselessly into the past.

F. Scott Fitzgerald

For my grandfather, Hugo, who gave me wings.

ACKNOWLEDGMENTS

First and foremost, I would like to express my deep gratitude to my advisor Dr. Kamran Paynabar for his invaluable guidance and unconditional support during my Ph.D. journey. In five years, he taught me to be an independent researcher, an encouraging mentor, and an engaging teacher. The lessons learned will travel with me wherever life takes me.

I sincerely thank Dr. Jianjun Shi, Dr. Brani Vidakovic, Dr. Yajun Mei, and Dr. Babak Mahmoudi for serving on my thesis committee. Their constructive comments and insightful advice helped immensely in improving this dissertation. I would also like to thank Dr. Massimo Pacella, Dr. Pinar Keskinocak, Dr. Joel Sokol, Dr. Edmond Chow, Lucas Erlandson, and Dan Li for collaborating with me and helping me grow as a researcher.

Special thanks go to Dr. María Elsa Correal, Dr. Adolfo Quiroz, and Dr. Carlos Valencia for their guidance when I was a student at the Universidad de los Andes. Without them and their support, I would have never considered a life in academia. My Ph.D. journey started with them.

Next, I would like to thank my academic family, Samaneh Ebrahimi, Xiaolei Fang, Jinhyeun Kim, Dan Li, Anjolaoluwa Popoola, Mostafa Reisi, Qian Wang, Wei Yang , and Zhen Zhong, for allowing me to be a part of their journey. I learned many things from them, their feedback constantly improved the quality of my work.

I would also like to express my gratitude to my dear friends who were there to celebrate my wins but also to support me through difficult times. Thank you Juan David Barbosa, Beste Basciftci, Alessia Benevento, Jana Boerger, Mariana De Almeida, Akane Fujimoto, Paola Guevara, Kirthana Hampapur, Daniela Hurtado, Arvind Krishna, Adriana Lozoya, Diana Montañes, Germán Ramirez, Camilo Riveros, and Catalina Villabona.

Last but not least, I would like to express my most profound appreciation to my mom, Claudia, my sister, Carolina, my grandparents, Helena and Hugo, and the rest of my family for their unconditional love, encouragement, and patience. I could not have traveled this path without them. Finally, but most importantly, I would like to thank my husband, Camilo, who has held my hand every step of the way. I feel fortunate to share my life journey with him.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

ADMM Alternating Direction Method of Multipliers

ARL Average Run Length

CP CANDECOMP/PARAFAC

CP-WOPT CP-Weighted Optimization

CUSUM Cumulative Sum

DAG Directed Acyclic Graph

DGMs Directed Graphical Models

DSSGM Dynamic Sparse Spectral Graphical Model

DSSL Dynamic Sparse Subspace Learning

EEG electroencephalography

EGR Exhausted Gas Recirculated

EWMA Exponentially Weighted Moving Average

FC Functional Connectivity

FPCA Functional Principal Components Analysis

HD High-dimensional

LD Low-dimensional

MGP Multivariate Gaussian Process

MSPE Mean Square Prediction Error

NSC NO_x Storage Catalyst

PGD Proximal Gradient Descent

RCP-WOPT Recursive CP-Weighted Optimization

RD Recursive Diagnosis

RMSE Root Mean Square Error

RTR Recursive Tensor Recovery

SDM Spectral Density Matrix

SNR Signal-to-noise ratio

ST-SSD Spatio-temporal Smooth Sparse Decomposition

TSS Tensor Sequential Sampling

SUMMARY

With the development of technology, sensing systems became ubiquitous. As a result, a wide variety of complex systems are continuously monitored by hundreds of sensors collecting large volumes of rich data. Learning the structure of complex systems, from sensing data, provides unique opportunities for real-time process monitoring and for accurate fault diagnosis in a wide range of applications. This dissertation presents new methodologies to analyze the high-dimensional data collected by sensors to learn the interactions between different entities in complex systems for system monitoring and diagnosis. Chapter 1 presents the research background, motivation, and challenges, and briefly introduces the methodologies developed.

Chapters 2 and 3 aim at representing complex systems as probabilistic graphical models to capture the relationships between the variables in the system. Chapter 2 presents a methodology to learn directed graphical models when the sensing data has a functional form. The goal is to use the direction of the edges in the graph to identify the root-causes behind system failures. Learning a directed graphical model from data includes parameter learning and structure learning. 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. The cyclic coordinate accelerated proximal gradient descent algorithm is employed to minimize the loss function and learn the structure of the directed graph. To illustrate the advantages of the proposed methodology, multivariate sensor data from an internal combustion engine is used.

Chapter 3 aims at monitoring the structural evolution of complex systems by sequentially estimating undirected graphical models from high-dimensional streaming data. The main idea is to exploit the spectral information contained in the data to learn the system's

structure over time. For this purpose, the streaming data is divided into windows, and the graphical LASSO for time series data is used to learn the conditional independence relationships of the system's variables. The structural change between windows is allowed but regularized to allow for change-point detection. The proposed monitoring strategy is efficiently implemented by applying the Alternating Direction Method of Multipliers and can be used for real-time monitoring. The effectiveness of the methodology is demonstrated in two case studies. First, we track human motion, and later we monitor for changes in the brain's functional connectivity.

Chapter 4 leaves aside probabilistic graphical models to deal with a challenge commonly encountered when analyzing high-dimensional sensing data: incomplete information. In many applications, the sensing system that collects online data can only provide partial information from the process under study due to resource constraints. In such cases, an adaptive sampling strategy is needed to decide where to collect data while maximizing the change detection capability. This chapter proposes an adaptive sampling strategy for online monitoring and diagnosis with partially observed data. The proposed methodology integrates two novel ideas: (i) the recursive projection of the high-dimensional streaming data onto a low-dimensional subspace to capture the spatio-temporal structure of the data while performing missing data imputation; and (ii) the development of an adaptive sampling scheme, balancing exploration and exploitation, to decide where to collect data at each acquisition time. Through simulations and two case studies, the proposed framework's performance is evaluated and compared with benchmark methods.

Lastly, Chapter 5 concludes the dissertation and outlines topics for future research.

CHAPTER 1
INTRODUCTION

1.1 Background and Motivation

Nowadays, most complex systems are continuously monitored by hundreds of sensors. These sensors provide a variety of HD streaming data with rich information about the system's performance. Analyzing the HD data provides unique opportunities for system modeling, monitoring, and diagnosis in a wide range of applications. For example, in the automotive industry, sensors mounted on car engines are used to identify the root-causes behind engines' faults. In precision agriculture, sensors located in the fields are used to continuously monitor the quality of the crops. In neuroscience, intracranial electroencephalography (EEG) sensors are used to detect seizures in epileptic patients and to identify the brain regions responsible for the attacks. While real-time system monitoring and diagnosis are crucial in many applications, the complex characteristics of the data collected pose significant analytical and computational challenges yet to be addressed. These challenges are presented in more detail in section 1.2. The main goal of this dissertation is to develop new methodologies for the analysis of HD sensing data to translate data-rich environments into smart decision-making by addressing existing challenges. In section 1.3 we present an overview of the methodologies developed.

1.2 Data Characteristics and Challenges

Analyzing HD sensing data for modeling, monitoring, and diagnosing complex systems is an important yet challenging task. In this section, we discuss the common characteristics and modeling challenges associated with analyzing HD data.

High-dimensionality. Sensors monitoring complex systems are continuously taking measurements over time and generating various types of HD data. In some applications, sensors produce one observation at each sampling time, which results in time series data with hundreds or thousands of observations over time. In other instances, sensors sample profiles or functional data, where a single observation represents hundreds of data measurements. Other types of sensors capture images for system monitoring. In these cases, the dimensionality is even higher. In addition to the large number of pixels in each image, the number of images linearly grows over time as new images are recorded. The high-dimensionality of each sensor signal, coupled with the large number of sensors, produces prohibitive volumes of data, which raises significant **scalability** challenges for modeling, monitoring, and diagnosing complex systems.

High-velocity. With the development of technology, many sensing devices generate data at a fast rate. For example, a commercially available ultrasonic sensor can easily record data at the rate of 1KHz, and a high-speed industrial camera is capable of scanning a product surface with the rate of 80 million pixels per second or faster. These high collection rates pose significant **computational** challenges for real time monitoring and diagnosis, and require new methodologies to be computationally efficient.

Complex correlation structure. HD streaming data has an intrinsic complex spatio-temporal correlation structure. Neighbouring sensors exhibit high spatial correlation, while measurements across time are temporally correlated. When analyzing HD data, the auto-correlation of each signal over time needs to be considered. Additionally, the cross- correlation among different signals needs to be modeled. The complex correlation structure pose significant **analytical** challenges. The correlation patterns need to be modeled carefully to achieve high monitoring and diagnosis performances.

Incomplete data. In some cases, it is possible to have resources constraints on the system, that limit the amount of data available for modeling, monitoring, and diagnosis. For example, in environmental monitoring, sensors have a limited battery lifetime, and the cost of replacing the battery is often high. Therefore, monitoring needs to be done with a limited number of operational sensors, at any given time. Partial observations can also arise when the number of available sensors is small compared to the number of variables for monitoring. Finally, transmission and processing constraints can limit the amount of available data. In all of these settings, a **robust** sampling strategy needs to be implemented to decide where to collect data in order to maximize the change detectability.

This dissertation focuses on developing new methodologies that address the aforementioned scalability, computational, analytical, and robustness challenges to analyze HD sensing data with the goal of learning the interactions between different entities in complex systems for system monitoring and diagnosis.

1.3 Overview of the Dissertation

The dissertation is organized as follows. In the first part (Chapters 2 and 3), we use probabilistic graphical models to capture the complex correlation structure of the system under study. In the second part (Chapter 4) we leave aside probabilistic graphical models, to focus on the incomplete data challenge. Next, we introduce each of the methodologies developed.

1.3.1 Functional Directed Graphical Models for Root-Cause Analysis and Diagnosis

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 [1]. In manufacturing processes, they are used to learn the causal relationship among process variables and quality measures, and to facilitate process control [2].

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 [3]. 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. The main challenge to learn functional DGMs is preserving the functional information in each node.

The main goal of this chapter is to develop a methodology for learning functional DGMs while preserving the existing cross-correlation between the variables in the system. For this purpose, we first assume that the structure of the graph is known, and use function-to-function regression [4], to perform parameter learning of the graph. Then, inspired by Meinshausen and Buhlmann's (2006) neighborhood selection method [5], we present a new approach to learn the structure of functional DGMs. The main contributions of the chapter are: (i) we establish a methodology to estimate the parameters of functional DGMs, given the structure of the graph, that outperforms existing methods, and (ii) 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.

1.3.2 Online Structural Change-point Detection via Sparse Spectral Graphical Models

Complex systems are continuously monitored by sensors collecting high-dimensional (HD) streaming data. The sensing data provides unique opportunities to learn the system state by exploiting the cross-correlation structure between the system entities. For example, in neuroscience, EEG sensors record the brain's spontaneous electrical activity to identify different emotional states based on functional connectivity patterns [6]. In biomechanics, Kinect sensors capture body poses by tracking body joints. The data is used to recognize different human gestures based on the skeletal body part movements [7]. As another example, in the automotive industry, sensors mounted on smart cars allow to identify different driving events such as driving straight, turning, and stopping [8].

Furthermore, complex systems are constantly evolving over time. For example, the brain can transition from a happy state to a sad state, the human body can perform different activities one after the other, and a car can go from driving straight to turning. As a system evolves, the cross-correlation structure between its entities changes. Therefore, using the HD streaming data collected by sensors to detect structural changes in the system is critical to monitor the system evolution.

The main goal of this chapter is to develop a new online structural change-point detection method able to estimate the cross-correlation structure of the HD streaming data collected by sensors. For this purpose, we sequentially estimate the cross-correlation structure for streaming windows of data. For each window, we assume that the time series data is stationary, and exploit the underlying spectral information as it is known that zeros at all frequencies in the inverse spectral density matrices characterize the conditional independence between the entities in the system [9]. Therefore, for each window, we estimate a probabilistic sparse spectral graphical model capturing the cross-correlation of the data. To control the change from window to window, we regularize it by using a group LASSO penalty [10]. Furthermore, to detect a structural-change point, we use an Exponentially Weighted Moving Average (EWMA) control chart that monitors the performance of the sparse spectral graphical model.

The main contributions of this chapter are: (i) we develop a new methodology to detect, in real-time, the structural changes in a system via sparse spectral graphical models, (ii) we estimate the cross-correlation structure of the system as a graph at each point in time.

1.3.3 Adaptive Sampling Strategy for Online Monitoring and Diagnosis of HD Streaming Data

Monitoring HD streaming data, in real-time, is critical to detect anomalies and system failures. For example, in power distribution systems, smart sensors located across the distribution grid are used for the detection and isolation of outages [11, 12]. In manufacturing, a large number of process variables are measured during the production for online quality insurance [13, 14, 15].

In this chapter we address two main challenges associated with the analysis of HD streaming data for system monitoring and diagnosis. On the one hand, we take into account that the system has resource constraints and only a subset of the variables can be observed at any point in time. Hence, a sequential sampling strategy needs to be implemented to decide where to collect data in order to maximize the change detectability. On the other hand, we consider the complex spatio-temporal structure of the data, as we know that neighboring measurements exhibit high spatial correlation, while measurements across time are temporally correlated with a non-stationary behavior.

The main goal of this chapter is to develop an adaptive sampling strategy, and an online process monitoring and diagnosis approach for incomplete and non-stationary HD streaming data, by incorporating its spatial and temporal information. To develop our adaptive sampling strategy, we propose to consider the streaming data as an incomplete tensor. We develop a Recursive Tensor Recovery (RTR) model that decomposes the streaming tensor into three components: a low-rank component, a sparse component, and a noise component. The low-rank component captures the spatio-temporal correlation of the data, which is estimated through tensor factorization [16, 17, 18], while the sparse component captures the variables suspicious of change. We use the information contained in the sparse component for monitoring and diagnosis. To define where to sample at each acquisition time, we propose a Tensor Sequential Sampling (TSS) scheme that balances exploration and exploitation by combining two probability functions. When the process is in-control, our sampling strategy behaves like random sampling, and when the process is out-of-control, it behaves like greedy sampling to locate where the changes have occurred.

The main contributions of this chapter are: (i) we develop an adaptive sequential sampling strategy for online monitoring of HD streaming data that outperforms existing methods by capturing their spatio-temporal correlation, (ii) we estimate the values of the unobserved variables at each acquisition time, and (iii) we develop a Recursive Diagnosis (RD) algorithm able to detect the abnormal variables quickly by incorporating the spatial correlation.

CHAPTER 2
FUNCTIONAL DIRECTED GRAPHICAL MODELS AND APPLICATIONS IN ROOT-CAUSE ANALYSIS AND DIAGNOSIS

2.1 Introduction

Complex biological, physical, and social systems are often comprised of various entities and variables that exhibit intricate relationships and interactions. 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 [1]. In manufacturing processes, they are used to learn the causal relationship among process variables and quality measures, and to facilitate process control [2]. Additional applications to medical diagnosis, clinical decision support, crime factor analysis, sensor validation, credit-rating, risk management, and robotics are found in [19].

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 [3]. 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,

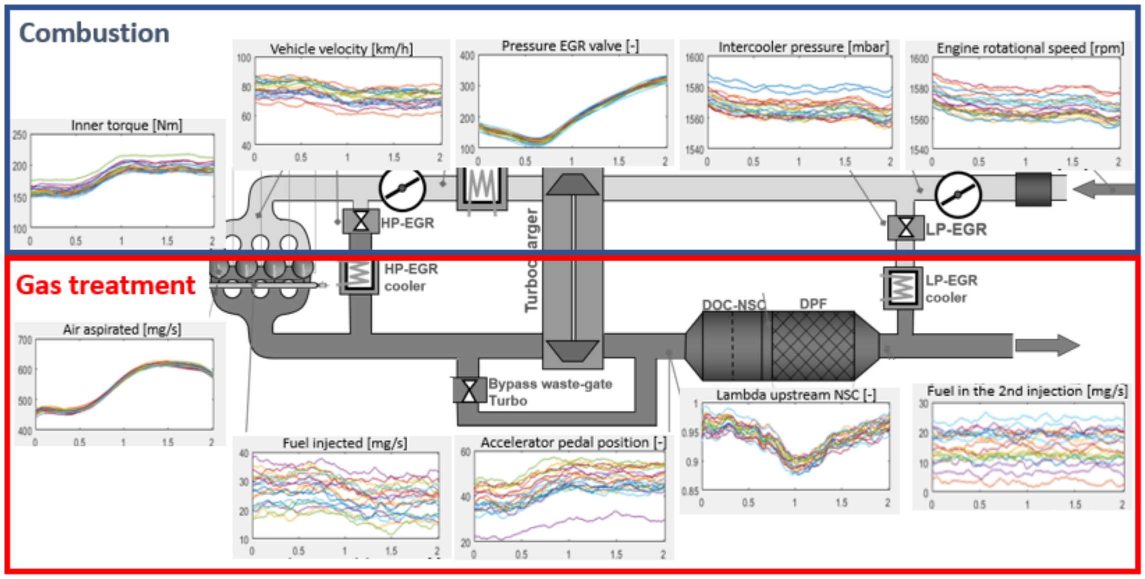


Figure 2.1: Exhaust after-treatment process of an internal combustion engine

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 2.1. 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.

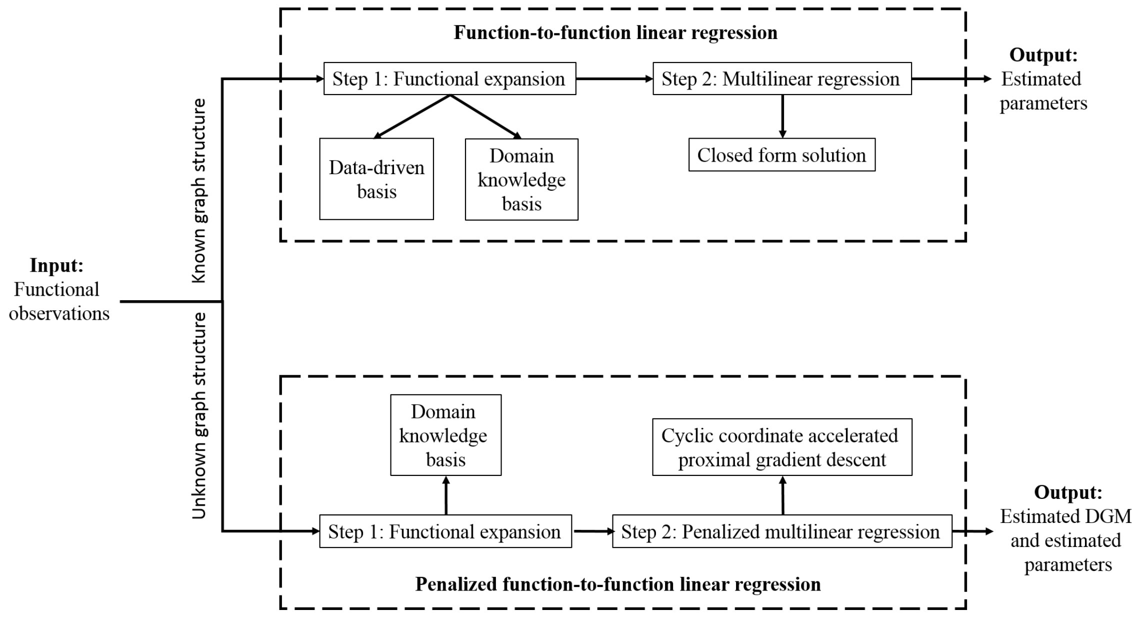
The main challenge to learn functional DGMs is preserving the functional information in each node. Existing probabilistic graphical models, developed for scalars, cannot be used or easily extended to deal with functional data. If each point in the functional data is considered as a scalar, the functional structure is lost. Recently, some methodologies have been developed to learn the structure of functional undirected graphical models. Zhu, Strawn and, Dunson (2016) proposed 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 chapter 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 chapter, 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 [22, 26, 21]. (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 [4], 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 [27], to the functional setting, to tune the parameters of the penalty terms. The main contributions of the chapter 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 chapter is organized as follows. In section 2.2, we provide a methodology overview. In section 2.3, we first present the parameter learning method for functional DGMs when the structure is known. Then, we present the methodology to learn the



Function-to-function linear regression
Step 1: Functional expansion
Step 2: Multilinear regression
Data-driven basis
Domain knowledge basis
Closed form solution
Output: Estimated parameters
Known graph structure
Input: Functional observations
Unknown graph structure
Domain knowledge basis
Cyclic coordinate accelerated proximal gradient descent
Step 1: Functional expansion
Step 2: Penalized multilinear regression
Output: Estimated DGM and estimated parameters
Penalized function-to-function linear regression

Figure 2.2: Flow diagram of the proposed methodology

structure of functional DGMs. Simulation studies and real data analysis are conducted in section 2.4 and section 2.5, respectively. Finally, we conclude the chapter in section 2.6.

2.2 Methodology Overview

A general framework of the proposed methodology is shown in Figure 2.2. 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.

2.3 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 [20]. Variables, such as vehicle velocity, intercooler pressure, and air-to-fuel ratio (λ-upstream), are measured by D sensors mounted on the engine (Figure 2.1). 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 2.5. The main goal of this chapter 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 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.

2.3.1 Known Graph Structure

Throughout the chapter, we assume that the functional variables $x_{1i}(t), x_{2i}(t), \cdots, x_{Di}(t)$ jointly follow from a D-dimensional MGP, $\mathcal{G}(t)$, independently and identically, and that 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

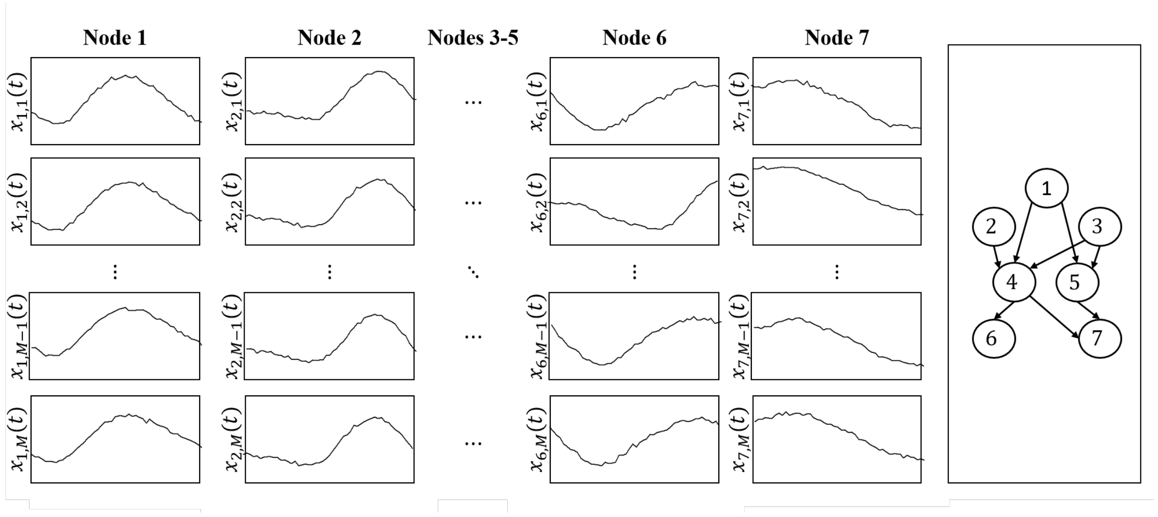


Figure 2.3: Illustrative example. Left: the data, $x_{ki}(t)$ for $k = 1, \cdots, 7$ nodes and $i = 1, \cdots, M$ observations. Right: the true underlying graph structure.

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

$$\mathcal{G}(t) = p(x_1(t), \cdots, x_D(t)) = \prod_{k=1}^{D} p(x_k(t)|\text{pa}_k). \tag{2.1}$$

This equation expresses the factorization properties of the joint distribution for a DGM [3]. 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 2.3 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

$$\begin{aligned} p\left(x_1(t), \cdots, x_7(t)\right) =& p\left(x_1(t)\right) \times p\left(x_2(t)\right) \times p\left(x_3(t)\right) \\ & \times p\left(x_4(t)|x_1, x_2, x_3)\right) \times p\left(x_5(t)|x_1, x_3\right) \\ & \times p\left(x_6(t)|x_4\right) \times p\left(x_7(t)|x_4, x_5\right) \end{aligned} \tag{2.2}$$

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 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

$$x_{ki}(t)|\text{pa}_k \sim \mathcal{N}\left(\sum_{j\in\text{pa}_k}\int_0^1 \beta_{kj}(t,s)x_{ji}(s)\text{d}s, \sigma_k^2\right) \tag{2.3}$$

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,

$$x_{ki}(t) = \sum_{j \in \text{pa}_k} \int_0^1 \beta_{kj}(t,s) x_{ji}(s) \mathrm{d}s + \sigma_k \epsilon_{ki}(t) \tag{2.4}$$

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 [28], we assume that the parameters have a functional expansion, defined by

$$\beta_{kj}(t,s) = \sum_{p=1}^{P_k} \sum_{q=1}^{P_j} b_{kjpq} \theta_{kp}(t) \theta_{jq}(s) \tag{2.5}$$

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.

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 [28]. Using the eigen-decomposition of the covariance function of x_k, cov$(x_k(t), x_k(t'))$, $x_{ki}(t)$ can be written as:

$$x_{ki}(t) = \sum_{p=1}^{\infty} \xi_{kip}\theta_{kp}(t) \tag{2.6}$$

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 Equation 2.6, by plugging Equation 2.5 into Equation 2.4, we have

$$\sum_{p=1}^{\infty} \xi_{kip}\theta_{kp}(t) = \sum_{j\in \text{pa}_k} \int_0^1 \sum_{p=1}^{\infty}\sum_{q=1}^{\infty} b_{kjpq}\theta_{kp}(t)\theta_{jq}(s) \sum_{q=1}^{\infty} \xi_{jiq}\theta_{jq}(s)ds + \sigma_k\epsilon_{ki}(t). \tag{2.7}$$

Using the orthonormality of the θ_{jq}'s, Equation 2.7 can be reduced to

$$\sum_{p=1}^{\infty} \xi_{kip}\theta_{kp}(t) = \sum_{j\in \text{pa}_k} \sum_{p=1}^{\infty}\sum_{q=1}^{\infty} b_{kjpq}\xi_{jq}\theta_{kp}(t) + \sigma_k\epsilon_{ki}(t). \tag{2.8}$$

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

$$\xi_{kip} = \sum_{j\in \text{pa}_k} \sum_{q=1}^{\infty} b_{kjpq}\xi_{jiq} + \sigma_k\varepsilon_{ki} \tag{2.9}$$

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

$$\hat{x}_{ki}(t) = \sum_{p=1}^{P_k} \hat{\xi}_{kip}\hat{\theta}_{kp}(t) \tag{2.10}$$

$$\hat{x}_{ji}(s) = \sum_{q=1}^{P_j} \hat{\xi}_{jiq}\hat{\theta}_{jiq}(s). \tag{2.11}$$

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 2.9 can be approximated by the following multilinear regression problem,

$$\boldsymbol{\xi}_k = \sum_{j\in \text{pa}_k} \boldsymbol{\xi}_j \boldsymbol{b}_{kj} + \sigma_k \boldsymbol{\varepsilon}_k \tag{2.12}$$

Let $\Xi_k = [\boldsymbol{\xi}_j]$, $j \in \text{pa}_k$, 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 2.12 is

$$\boldsymbol{\xi}_k = \Xi_k \boldsymbol{B}_k + \sigma_k \boldsymbol{\varepsilon}_k \tag{2.13}$$

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

$$L(\boldsymbol{B}_k) = \frac{1}{2}||\boldsymbol{\xi}_k - \Xi_k \boldsymbol{B}_k||_2^2 \tag{2.14}$$

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

$$\hat{\boldsymbol{B}}_k = (\Xi_k^\top \Xi_k)^{-1} \Xi_k^\top \boldsymbol{\xi}_k. \tag{2.15}$$

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

$$\lim_{M\to\infty}\int_0^1\int_0^1[\beta_{kj}(t,s)-\hat{\beta}_{kj}(t,s)]^2\mathrm{d}s\mathrm{d}t=0 \text{ in probabiliy,} \tag{2.16}$$

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 [29].

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 Equation 2.15. The computational cost for this step is $\mathcal{O}(MQ_k^2+Q_k^3+MP_kQ_k)$.

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 Equation 2.4 becomes

$$\boldsymbol{x}_k(t)=\sum_{j\in\mathrm{pa}_k}\int_0^1\boldsymbol{x}_j(s)\boldsymbol{\theta}_j^\top(s)\boldsymbol{B}_{kj}\boldsymbol{\theta}_k(t)\mathrm{d}s+\sigma_k\boldsymbol{\epsilon}_k(t) \tag{2.17}$$

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

$$\boldsymbol{Z}_j := \int_0^1 \boldsymbol{x}_j(s)\boldsymbol{\theta}_j^\top(s)\mathrm{d}s \in \mathbb{R}^{M\times P_j}$$

and

$$\boldsymbol{X}_k := [\boldsymbol{x}_k(t_1), \cdots, \boldsymbol{x}_k(t_{n_k})] \in \mathbb{R}^{M\times n_k},$$

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

$$\boldsymbol{X}_k = \sum_{j\in \mathrm{pa}_k} \boldsymbol{Z}_j\boldsymbol{B}_{kj}\boldsymbol{\Theta}_k + \sigma_k\boldsymbol{E}_k. \tag{2.18}$$

This can be further simplified to

$$\boldsymbol{x_k} = \sum_{j\in \mathrm{pa}_k} \Xi_{kj}\boldsymbol{b}_{kj} + \sigma_k\boldsymbol{e}_k, \tag{2.19}$$

where $\boldsymbol{x}_k = vec(\boldsymbol{X}_k^\top)$, $\Xi_{kj} = \boldsymbol{Z}_j \otimes \boldsymbol{\Theta}_k^\top$, $\boldsymbol{b}_{kj} = vec(\boldsymbol{B}_{kj}^\top)$, 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

$$L(\boldsymbol{b}_{kj}|j \in \mathrm{pa}_k) = \frac{1}{2}||\boldsymbol{x}_k - \sum_{j\in \mathrm{pa}_k} \Xi_{kj}\boldsymbol{b}_{kj}||_2^2. \tag{2.20}$$

The problem has the closed-form solution,

$$\hat{\boldsymbol{B}}_k = (\boldsymbol{\Xi_k}^\top\boldsymbol{\Xi_k})^{-1}\boldsymbol{\Xi_k}^\top\boldsymbol{x}_k \tag{2.21}$$

where $\Xi_k = [\Xi_{kj}]$ and $\boldsymbol{B}_k = [\boldsymbol{b}_{kj}]^\top$, $j \in \mathrm{pa}_k$. Consistency properties of this approach are not fully understood [28], 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 Ξ_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 Equation 2.21, the associated cost is $\mathcal{O}(Mn_kP_k^2Q_k^2 + P_k^3Q_k^3 + Mn_kP_kQ_k)$.

2.3.2 Structure Learning

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 2.3 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 2.4. 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

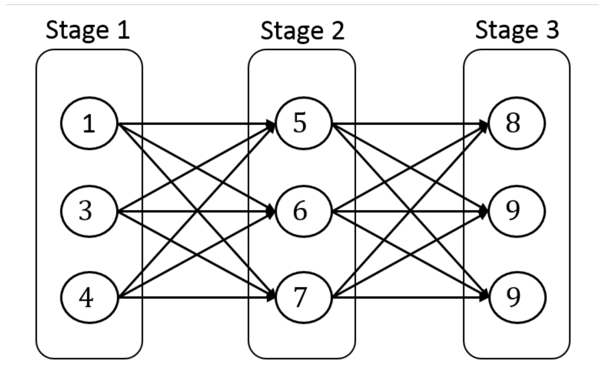


Figure 2.4: Illustrative example of a sequential manufacturing process

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 [5], we specify our model as a penalized function-to-function linear regression. For every variable k, $k = 1, \cdots, D$, we minimize the loss function:

$$L(\boldsymbol{b}_{kj}|j \in \text{cpa}_k) = \frac{1}{2}||\boldsymbol{x}_k - \sum_{j \in \text{cpa}_k} \boldsymbol{\Xi}_{kj}\boldsymbol{b}_{kj}||_2^2 + \gamma \sum_{j \in \text{cpa}_k} \sqrt{q_{kj}}||\boldsymbol{b}_{kj}||_2 + \frac{\lambda}{2} \sum_{j \in \text{cpa}_k} ||\boldsymbol{b}_{kj}||_2^2. \quad (2.22)$$

The first term corresponds to the least square loss function presented in Equation 2.20. The second term is a group LASSO penalty that encourages sparsity in the model by performing variable selection [10], 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 [30]. 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 [10]. 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 Equation 2.22 can be reduced to

$$L(\boldsymbol{b}_{kj}) = \frac{1}{2}||\boldsymbol{r}_{kj} - \Xi_{kj}\boldsymbol{b}_{kj}||_2^2 + \gamma\sqrt{q_{kj}}||\boldsymbol{b}_{kj}||_2 + \frac{\lambda}{2}||\boldsymbol{b}_{kj}||_2^2 + C \tag{2.23}$$

where $\boldsymbol{r}_{kj} = \boldsymbol{x}_k - \sum_{l\neq j} \Xi_{kl}\boldsymbol{b}_{kl}$ 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 $f(\boldsymbol{b}_{kj}) = \frac{1}{2}||\boldsymbol{r}_{kj} - \Xi_{kj}\boldsymbol{b}_{kj}||_2^2 + \frac{\lambda}{2}||\boldsymbol{b}_{kj}||_2^2 + C$, 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

$$\boldsymbol{b}_{kj}^{(t+1)} = \arg\min_{\boldsymbol{b}_{kj}} \left\{ f(\boldsymbol{b}_{kj}^{(t)}) + \left\langle \nabla f(\boldsymbol{b}_{kj}^{(t)}), \boldsymbol{b}_{kj} - \boldsymbol{b}_{kj}^{(t)} \right\rangle + \frac{1}{2s^{(t)}} ||\boldsymbol{b}_{kj} - \boldsymbol{b}_{kj}^{(t)}||_2^2 + g(\boldsymbol{b}_{kj}) \right\} \tag{2.24}$$

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, section A.1):*

$$\begin{aligned} \boldsymbol{z}^{(t+1)} &\leftarrow \boldsymbol{b}_{kj}^{(t)} + s^{(t)} \left(\Xi_{kj}^{\top} (\boldsymbol{r}_{kj} - \Xi_{kj} \boldsymbol{b}_{kj}^{(t)}) - \lambda \boldsymbol{b}_{kj}^{(t)} \right) \\ \boldsymbol{b}_{kj}^{(t+1)} &\leftarrow \left(1 - \frac{s^{(t)} \gamma \sqrt{q_{kj}}}{||\boldsymbol{z}^{(t+1)}||_2} \right)_+ \boldsymbol{z}^{(t+1)} \end{aligned} \tag{2.25}$$

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:

$$\begin{aligned} \boldsymbol{z}^{(t+1)} &\leftarrow \boldsymbol{\eta}_{kj}^{(t)} + s^{(t)} \left(\Xi_{kj}^{\top} (\boldsymbol{r}_{kj} - \Xi_{kj} \boldsymbol{\eta}_{kj}^{(t)}) - \lambda \boldsymbol{\eta}_{kj}^{(t)} \right) \\ \boldsymbol{b}_{kj}^{(t+1)} &\leftarrow \left(1 - \frac{s^{(t)} \gamma \sqrt{q_{kj}}}{||\boldsymbol{z}^{t+1}||_2} \right)_+ \boldsymbol{z}^{(t+1)} \\ \boldsymbol{\eta}_{kj}^{(t+1)} &\leftarrow \boldsymbol{b}_{kj}^{(t+1)} + \frac{t}{t+3} (\boldsymbol{b}_{kj}^{(t+1)} - \boldsymbol{b}_{kj}^{(t)}) \end{aligned} \tag{2.26}$$

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 $\mathcal{O}(Mn_kP_kQ_k)$.

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
            $\boldsymbol{r}_{kj} = \boldsymbol{x}_k - \sum_{l \in \text{cpa}_k, l \neq j} \Xi_{kl}\boldsymbol{b}_{kl}$
            $||\boldsymbol{b}^1_{kj} - \boldsymbol{b}^0_{kj}||^2_2 = \text{inf}$
            $t = 1$
            while $||\boldsymbol{b}^{(t)}_{kj} - \boldsymbol{b}^{(t-1)}_{kj}||^2_2 > \epsilon$ do
                $\boldsymbol{z}^{(t)} = \boldsymbol{\eta}^{(t)}_{kj} + s^{(t)}(\Xi^\top_{kj}(\boldsymbol{r}_{kj} - \Xi_{kj}\boldsymbol{\eta}^{(t)}_{kj}) - \lambda\boldsymbol{\eta}^{(t)}_{kj})$
                $\boldsymbol{b}^{(t)}_{kj} = \left(1 - \frac{s^{(t)}\gamma\sqrt{q_{kj}}}{||\boldsymbol{z}^{(t)}||_2}\right)_+ \boldsymbol{z}^{(t)}$
                $\boldsymbol{\eta}^{(t)}_{kj} = \boldsymbol{b}^{(t)}_{kj} + \frac{t}{t+3}\left(\boldsymbol{b}^{(t)}_{kj} - \boldsymbol{b}^{(t-1)}_{kj}\right)$
                $t = t + 1$
        $L_i = L(\boldsymbol{b}^i_{kj} | j \in \text{cpa}_k)$
        $i = i + 1$
```

Proposition 2. $f(\boldsymbol{b}_{kj}) = \frac{1}{2}||\boldsymbol{r}_{kj} - \Xi_{kj}\boldsymbol{b}_{kj}||^2_2 + \frac{\lambda}{2}||\boldsymbol{b}_{kj}||^2_2 + C$ *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, section A.2)*

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.

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 chapter, we will follow the methods proposed in [27], 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

$$L_k(\gamma, \lambda) = \frac{1}{m_v} \sum_{i \in m_v} \frac{1}{n_k} \sum_{t=1}^{n_k} (x_{ki}^{(v)}(t) - \tilde{x}_{ki}^{(c)}(t))^2 \tag{2.27}$$

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.

2.4 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.

Table 2.1: Simulation factor settings

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
SNR	Ratio of variance of the response and noise term	20, 200

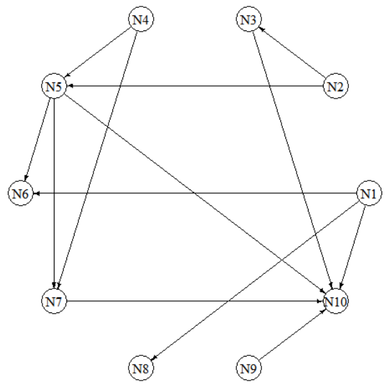
2.4.1 Known Graph Structure

To evaluate the performance of the proposed methodology, we simulate eight different scenarios, in which we vary three different factors, as shown in Table 2.1. 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 2.5. The second step is to generate the functional random variables. The curves are produced by following the simulation study in [31]. 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 $\phi_{lkj}(s)$, $l = 1,2,3$, are Gaussian processes with covariance function Σ_1. Finally, we generate the response curves as

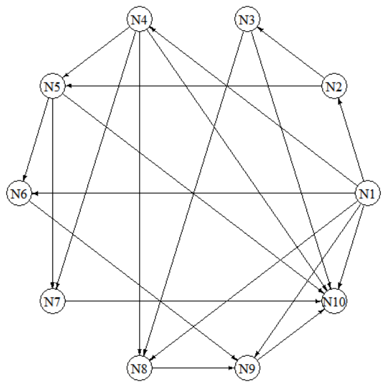
$$x_k(t) = \sum_{j\in\text{pa}_k} \int \beta_{kj}(s,t)x_j(s)\mathrm{d}s + \sigma_k\epsilon_k(t) \tag{2.28}$$

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 $n_k = 50$.

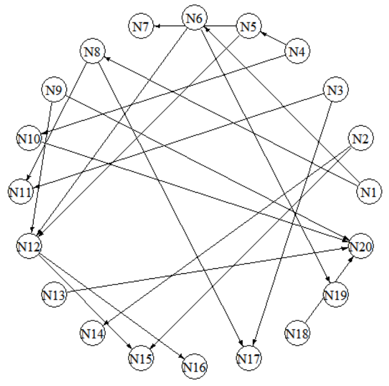
To evaluate the performance of each parameter estimation method, we generate a set of $M = 100$ samples, and randomly divide the data into a training set of size 80 and a test set



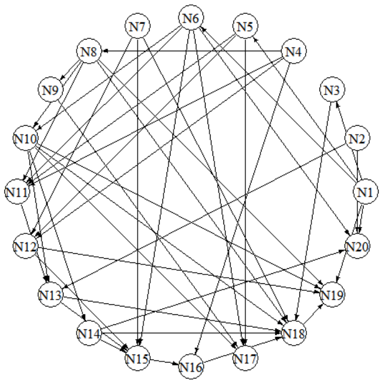
(a) 10 nodes, density 0.2



(b) 10 nodes, density 0.4



(c) 20 nodes, density 0.1



(d) 20 nodes, density 0.2

Figure 2.5: DAGs generated for each simulation scenario

of size 20. We calculate the Mean Square Prediction Error (MSPE), for each node, using the testing data as follows:

$$MSPE_k = \frac{1}{M_{test}} \sum_{i=1}^{M_{test}} \frac{1}{50} \sum_{t=1}^{50} (x_{ki}(t) - \hat{x}_{ki}(t))^2. \tag{2.29}$$

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

$$MSPE = \frac{1}{D} \sum_{k=1}^{D} MSPE_k. \tag{2.30}$$

We compare the results obtained using FPCA basis (labeled as FPCR) and pre-specified basis (labeled as FDGM) with the existing benchmark proposed in [26] (labeled as FGM). The loss function in [26] is given by,

$$\begin{aligned} L(b_{kj}(t)|j \in \text{pa}_k, t \in C_k) \;&=\; \textstyle\sum_{i=1}^{M} \sum_{t \in C_k} \left(x_{ki}(t) - \sum_{j \in \text{pa}_k} x_{ji}(t) b_{kj}(t) \right)^2 \\ &\quad + \lambda \textstyle\sum_{t \in C_k} ||b_{kj}(t) - \dfrac{1}{n_k} \sum_{t \in C_k} b_{kj}(t)||_2^2 \end{aligned} \tag{2.31}$$

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 2.2. 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 2.3. Both FPCR and FDGM have a closed-form solution, with no tuning parameter. Furthermore, Table 2.3 shows that

Table 2.2: Comparison between methods using MSPE, 1000 simulations. Results are reported in the form of mean (standard error).

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 2.3: Comparison between methods in terms of computational time in seconds, 100 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)

the fastest method is FDGM.

In Table 2.4, the three methods are compared, using the MSPE for every node, for the scenario with 10 nodes, density 0.4 (refer to Figure 2.5 (b)), and a SNR of 200. It can be seen that, for the 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 Table 2.2, Table 2.3, and Table 2.4, it is clear that FDGM is consistently the best method.

2.4.2 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 2.6 (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 $\{40.2, 0.4, 0.6, 0.8, 1, 3, 5, 7, 9, 11, 13, 15\}$ and λ from $\{10^{-10}, 10^{-8}, 10^{-6}, 10^{-4}, 10^{-2}\}$. We

Table 2.4: Comparison between methods using $MSPE_k$, for the scenario with 10 nodes, density 0.4, and SNR 200, 1000 simulations. Results are reported in the form of mean (standard error).

	Node 2	Node 3	Node 4	Node 5	Node 6
FGM	0.391(0.002)	0.046(0.000)	0.087(0.001)	0.135(0.001)	1.781(0.009)
FPCR	0.008(0.000)	0.001(0.000)	0.009(0.000)	0.004(0.000)	0.013(0.000)
FDGM	0.002(0.000)	0.001(0.000)	0.000(0.000)	0.002(0.000)	0.005(0.000)

	Node 7	Node 8	Node 9	Node 10
FGM	0.021(0.000)	0.531(0.003)	8.466(0.037)	29.791(0.184)
FPCR	0.003(0.000)	0.179(0.002)	0.172(0.001)	0.154(0.002)
FDGM	0.000(0.000)	0.004(0.000)	0.127(0.001)	0.106(0.001)

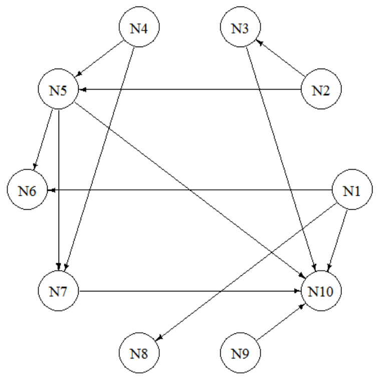
use $m_v = M^{0.7}$ and $b = 5$. As mentioned earlier, we assume that we have a DAG with ordered nodes such that each child node has a higher number than its parents. Therefore, we consider 1 as the root node, and learn the structure in an orderly fashion for nodes 2 through 10. For node k, the set cpa_k is equal to $\{1, \ldots, k-1\}$.

The true model and the learned structure over a thousand simulation experiments are presented in Figure 2.6. 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 2.5. As expected, when the noise increases, the SNR decreases, and, therefore, the prediction task is harder.

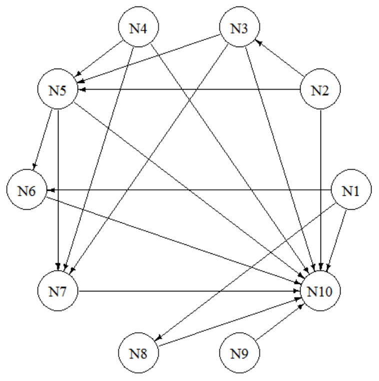
To evaluate the recall and precision of the method, the confusion matrix is computed in Table 2.6. We observe that the proposed method has a true positive rate of 100%. However, the learned structure has additional arcs. This is because the group LASSO penalty tends to select more variables. As an overall performance measure, we use the F_β-score, which is a harmonic mean of precision and recall. We have:

$$F_\beta = (1+\beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}} \tag{2.32}$$

where precision $= TP/(TP+FP)$, recall $= TP/(TP+FN)$, and β is a coefficient that



(a) True structure



(b) Learned structure

Figure 2.6: True structure and structure learned for a functional DGM with 10 nodes, density 0.2, and SNR ratio 200/20

Table 2.5: MSPE and standard error for a functional DGM with 10 nodes, density 0.2, and SNR 200/20

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

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 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 2.7. 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.

Table 2.6: Confusion matrix for a functional DGM with 10 nodes, density 0.2, and SNR 200/20

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

Table 2.7: Computational time to establish the parents of each node in the graph for a functional DGM with 10 nodes and density 0.2

	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

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.

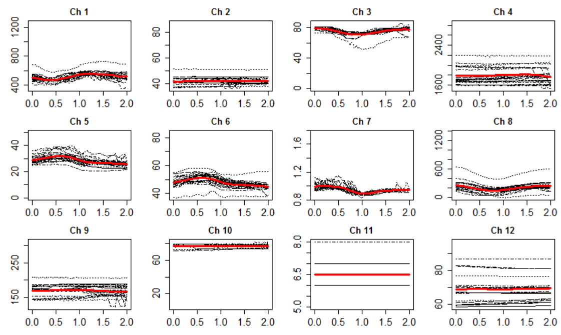
The NO_x Storage Catalyst (NSC) is an exhaust after-treatment system by which the exhaust gas is treated after the combustion process in two alternating phases: adsorption (molecules of NO_x in the exhaust gas are captured by an adsorber), and regeneration (the stored NO_x is reduced in a catalytic process). 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 analyzed the cluster patterns to further understand the root-causes behind fault events.

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 2.8) 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 100 Hz. The signals for each channel in clusters A and B can be seen in Figure 2.7. 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, the throttle valve (Ch10) is constantly opened for air intake, and the 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 re-

Table 2.8: List of on-board sensors (channels)

#	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]


(a) Cluster A

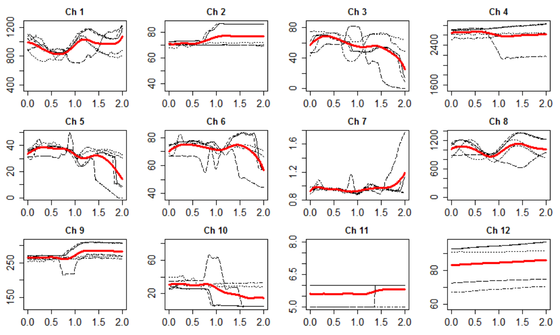
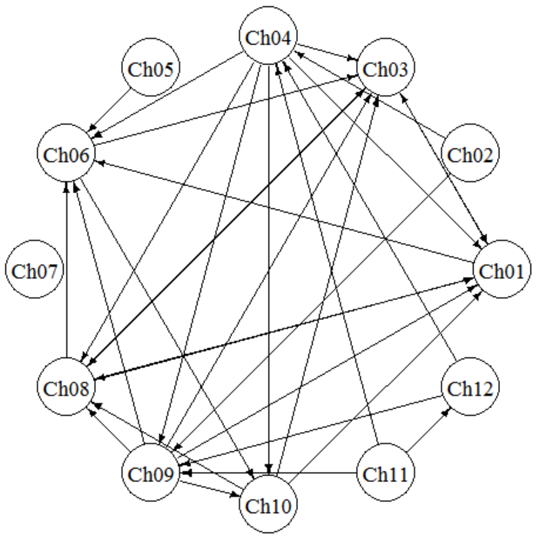

(b) Cluster B

Figure 2.7: 12 channels for clusters A and B. Each panel depicts the signals (black), and the mean signal (red).

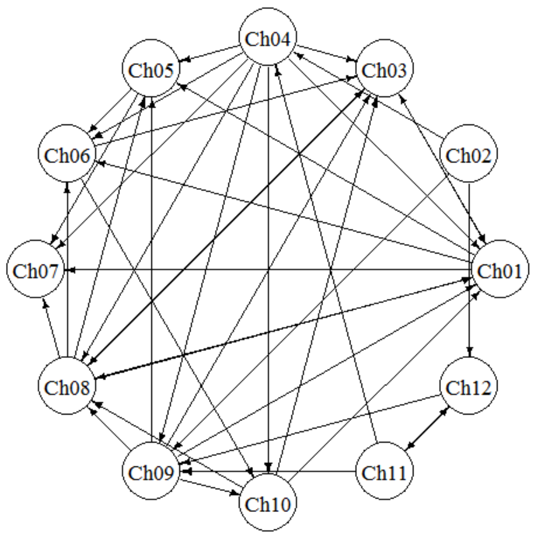
generation phase. On the other hand, for cluster B, experts concluded that the fault occurred during an acceleration phase of the vehicle, recognized by sudden changes in the throttle valve (Ch10). As a consequence of the acceleration phase, the fluctuations of the intake air mass (Ch01) during the NSC 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) [20].

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 2.8. 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 computed the $MSPE_k$ for each variable k, $k = 1, \cdots, 12$, for each cluster. The results are reported in Table 2.9. 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, therefore, more information.



(a) Cluster A



(b) Cluster B

Figure 2.8: Structure learned for clusters A and B

Table 2.9: $MSPE_k$, $k = 1, \cdots, 12$, for clusters A and B, over 100 replicates. Results are reported in the form of mean (standard error).

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)

2.6 Conclusion

This chapter 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 method presented in [26]. 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 chapter, 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.

CHAPTER 3
ONLINE STRUCTURAL CHANGE-POINT DETECTION OF HIGH-DIMENSIONAL STREAMING DATA VIA SPARSE SPECTRAL GRAPHICAL MODELS

3.1 Introduction

Complex systems are continuously monitored by sensors collecting high-dimensional (HD) streaming data. The sensing data provides unique opportunities to learn the system state by exploiting the cross-correlation structure between the system entities. For example, in neuroscience, EEG sensors record the brain's spontaneous electrical activity to identify different emotional states based on functional connectivity patterns [6]. In biomechanics, Kinect sensors capture body poses by tracking body joints. The data is used to recognize different human gestures based on the skeletal body part movements [7]. As another example, in the automotive industry, sensors mounted on smart cars allow to identify different driving events such as driving straight, turning, and stopping [8].

Furthermore, complex systems are constantly evolving over time. For example, the brain can transition from a happy state to a sad state, the human body can perform different activities one after the other, and a car can go from driving straight to turning. As a system evolves, the cross-correlation structure between its entities changes. Therefore, using the HD streaming data collected by sensors to detect structural changes in the system is critical to monitor the system evolution.

Assume the system is being monitored by p sensors collecting data at a grid of time points $u_1, u_2, \cdots, u_T$, and that M samples are available. One possible approach to learn the dynamic cross-correlation between the system entities is to estimate T probabilistic graphical models, one per time point. In the graphical representation, the nodes act as the entities in the system, and the edges express the partial correlations between the entities. In particular, if there is no edge between node j and node l, we can conclude that these two variables are conditionally independent given all the other variables in the system. To learn each graph at time $t = 1, \cdots, T$, the graphical LASSO [32] can be used. This method assumes that the variables jointly follow from a multivariate Gaussian distribution and estimates the precision matrix $\mathbf{\Theta}(t)$, with entries $\mathbf{\Theta}(t)_{jl}$, $j, l = 1, \cdots, p$, $t = 1, \cdots, T$. The precision matrix encodes the conditional independence structure of the system, if $\mathbf{\Theta}(t)_{jl} = 0$, then node j and node l are conditionally independent given all other variables at time t. However, this approach allows for too much variability as a different structure can be learned at each point in time.

In order to generate temporally consistent graphs, methodologies to jointly learn T graphical models that share certain characteristics are developed [33, 34, 35]. Specifically, in [35], Hallac et al. propose jointly learning T graphical models by regularizing the change for consecutive time points. They suggest selecting the regularization penalty based on domain knowledge about the system, where different evolutionary patterns are considered (few edges change at a time, global restructuring, block-wise restructuring,...). Other strategies to learn T temporally consistent graphs are based on smoothed sample covariance matrix estimators [36, 37, 38]. In particular, in [38], Qiu et al. propose a kernel based method to jointly estimate the graphical models. Nonetheless, these methodologies are not computationally scalable in cases where p and T are large. Additionally, they still give too much flexibility for the graphs dynamics. Even for some time intervals where no cross-correlation changes at all, these models generate fictitious dynamics, leading to the biased estimation of the cross-correlation change-points [39].

Recently, two new methods have been developed to overcome these limitations. On the one hand, in [40], Qiao et al. propose to estimate doubly graphical models. First, they apply a non-parametric approach to smooth p covariance functions and represent each curve using the first functional principal components. Then, the finite-dimensional representations of the curves lead to the functional estimate for the $p \times p$ covariance matrix. Finally, the structure of the system is estimated by computing the functional sparse precision matrix on a gird of points. On the other hand, in [39], Zhang et al. assume different functions come from different subspaces, in the sense that only functions of the same subspace have non-zero cross-correlation with each other. The subspace relationship is learned as a sparse self-expressive linear regression. To describe the cross-correlation dynamics, the regression coefficients are allowed to change over time, but the change is regularized with a fused LASSO penalty. Later on, the functions are clustered into different subspaces. Although, these methodologies are able to learn the dynamic cross-correlation structure of a system using functional data, they do it in an offline fashion. They cannot sequentially estimate the cross-correlation structure and detect, in real-time, the structural change-points. Additionally, in order to learn the dynamic cross-correlation structure, they need the number of samples M to be greater than one.

To the best of our knowledge, there is only one paper dealing with online structural change-point detection. In [41], Xu et al. propose a dynamic sparse subspace learning methodology inspired by [39], where the subspace relationship is still learned as a self-expressive linear regression. However, the authors assume that the parameters of the regression are not time-dependent for segments with fixed correlation structures. Therefore, they replace the fused LASSO penalty with one regularizing the number of change-points. With these modifications, the proposed approach is computationally efficient to perform online structural change-point detection. Nonetheless, this methodology has some drawbacks. First, by the way the problem is formulated, the underlying assumption is that inside each segment, the observations collected by the sensors are independent and identically distributed. In practice, this might not be true as in HD streaming data there is temporal correlation between consecutive measurements that needs to be properly modeled. Second, the methodology does not provide a graphical representation of the system in real-time, which is critical to monitor the state of the system.

The main goal of this chapter is to develop a new online structural change-point detection method able to estimate the cross-correlation structure of the HD streaming data collected by sensors. For this purpose, we sequentially estimate the cross-correlation structure for streaming windows of data. For each window, we assume that the time series data is stationary, and exploit the underlying spectral information as it is known that zeros at all frequencies in the inverse spectral density matrices characterize the conditional independence between the entities in the system [9]. Therefore, for each window, we estimate a probabilistic sparse spectral graphical model capturing the cross-correlation of the data. To control the change from window to window, we regularize it by using a group LASSO penalty [10]. Furthermore, to detect a structural-change point, we use an EWMA control chart that monitors the performance of the sparse spectral graphical model. The proposed monitoring strategy is efficiently implemented by applying the Alternating Direction Method of Multipliers (ADMM) and can be used for real-time monitoring.

The main contributions of this chapter are: (i) we develop a new methodology to detect, in real-time, the structural changes in a system via sparse spectral graphical models, (ii) we estimate the cross-correlation structure of the system as a graph at each point in time.

The rest of the chapter is organized as follows. In section 3.2, we introduce probabilistic graphical models for stationary time series data. In section 3.3, we present the proposed online structural change-point detection methodology. Simulation studies and two real-data analysis are conducted in section 3.4 and section 3.5, respectively. Finally, we conclude the chapter in section 3.6.

3.2 Probabilistic Graphical Models for Stationary Time Series

3.2.1 Gaussian Graphical Models

A graphical model is used to depict the conditional dependence structure among p random variables, $\boldsymbol{X} = (X_1, \cdots, X_p)^\top$. Such a network consists of p nodes, one for each variable, and a number of edges connecting a subset of the nodes. The edges describe the conditional dependence structure of the p variables, that is, nodes j and l are connected by an edge if and only if X_j and X_l are dependent, conditional on the other $p-2$ variables.

For Gaussian data, where $\boldsymbol{X}$ follows a multivariate Gaussian distribution, i.e., $\boldsymbol{X} \sim N(\mathbf{0}, \boldsymbol{\Sigma})$, one can show that estimating the edge set, E, is equivalent to identifying the locations of the nonzero elements in the precision matrix, $\boldsymbol{\Theta}(= \boldsymbol{\Sigma}^{-1})$ [32]. Therefore, $E = \{(j,l) : \boldsymbol{\Theta}_{jl} \neq 0, (j,l) \in V \times V, j \neq l\}$.

In practice, $\boldsymbol{\Theta}$ must be estimated based on a set of M observed p-dimensional realizations, $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_M$, of the random vector $\boldsymbol{X}$. The graphical LASSO [32] considers a regularized estimator for $\boldsymbol{\Theta}$ by adding an l_1 penalty on the off-diagonal entries of the precision matrix to the Gaussian log-likelihood:

$$\hat{\boldsymbol{\Theta}} = \arg\min_{\boldsymbol{\Theta}} \left\{ -\log\det \boldsymbol{\Theta} + \operatorname{trace}\{\hat{\boldsymbol{\Sigma}}\boldsymbol{\Theta}\} + \lambda \sum_{j \neq l} |\boldsymbol{\Theta}_{jl}| \right\} \tag{3.1}$$

where $\boldsymbol{\Theta} \in \mathbb{R}^{p \times p}$ is symmetric positive definite, $\hat{\boldsymbol{\Sigma}}$ is the sample covariance matrix of $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_M$, and λ is a non-negative tuning parameter. The l_1 penalty regularizes the estimate and ensures that $\hat{\boldsymbol{\Theta}}$ is sparse. Once $\hat{\boldsymbol{\Theta}}$ is computed, it is straight forward to estimate the edge set E.

3.2.2 Graphical Models for Stationary Time Series

Let $\boldsymbol{X}(t) = (X_1(t), \cdots, X_p(t))^\top$, for $t \in \mathbb{Z}$, be a stationary multivariate Gaussian time series. We have that $\mathbb{E}[\boldsymbol{X}(t)] = \mathbf{0}$ and $\mathbb{E}[\boldsymbol{X}(t), \boldsymbol{X}(t+h)] = \boldsymbol{\Gamma}(h)$, for all $t \in \mathbb{Z}$, where $\boldsymbol{\Gamma}(h)$ is the autocovariance function of the time series. For all $h \in \mathbb{Z}$, $\boldsymbol{\Gamma}(h)$ is a $p \times p$ symmetric positive definite matrix. Furthermore, assume that $\sum_{h=-\infty}^{\infty} ||\boldsymbol{\Gamma}(h)||_2 < \infty$ where $|| \cdot ||_2$ is the spectral norm. We can define the Spectral Density Matrix (SDM) of the time series, $\boldsymbol{S}(\omega)$, $\omega \in [0, 2\pi]$, via the discrete Fourier transform of the autocovariance function:

$$\boldsymbol{S}(\omega) = \frac{1}{2\pi} \sum_{h=-\infty}^{\infty} \boldsymbol{\Gamma}(h) e^{-ih\omega}. \tag{3.2}$$

For each ω, $\boldsymbol{S}(\omega)$ is a $p \times p$ Hermitian positive definite matrix. In addition, the function $\omega \to \boldsymbol{S}(\omega)$ is 2π periodic, and, for real-valued random variables, symmetric. Therefore, we only need to consider the SDM for $\omega \in [0, \pi]$.

The idea behind Gaussian graphical models can be naturally extended to Gaussian stationary time series. Estimating the edge set, E, is equivalent to identifying the location of the nonzero elements in the inverse SDM $\boldsymbol{\Theta}(\omega)(= \boldsymbol{S}(\omega))^{-1})$, $\omega \in [0, \pi]$ [9]. Specifically, if $\boldsymbol{\Theta}(\omega)_{jl} = 0$ for all $\omega \in [0, \pi]$, then the components X_j and X_l are conditionally independent given the remaining $p - 2$ series. Therefore, $E = \{(j, l) : \exists \omega \in [0, \pi] \text{ s.t. } \boldsymbol{\Theta}(\omega)_{jl} \neq 0, (j, l) \in V \times V, j \neq l\}$.

In practice, $\boldsymbol{\Theta}(\omega)$, $\omega \in [0, \pi]$, must be estimated based on a realization $\boldsymbol{x}(t)$, for $t = 1, \cdots, T$, of the time series $\boldsymbol{X}(t)$. The graphical LASSO for time series [42] considers a regularized estimator for $\boldsymbol{\Theta}(\omega)$. In the frequency domain, the log-likelihood of the time series can be efficiently computed using the Whittle approximation [43, 44]:

$$\log p(\boldsymbol{x}(1), \cdots, \boldsymbol{x}(T)) \approx \frac{1}{2} \sum_{k=0}^{T-1} \Big\{ \log \det \boldsymbol{\Theta}(\omega_k) - \operatorname{trace}\{\hat{\boldsymbol{S}}(\omega_k)\boldsymbol{\Theta}(\omega_k)\} \Big\} - \frac{Tp}{2} \log 2\pi, \tag{3.3}$$

where $\hat{\boldsymbol{S}}(\omega_k)$ is the SDM estimate at frequency $\omega_k = 2\pi k/T$. A sample based estimate of the SDM is given by the periodogram,

$$\boldsymbol{I}(\omega_k) = \frac{1}{2\pi} \boldsymbol{d}(k)\boldsymbol{d}(k)^H \tag{3.4}$$

where $\boldsymbol{d}(k)$, $k = 0, \cdots, T-1$ is the discrete Fourier transform of $\boldsymbol{x}(t)$, and $\boldsymbol{d}(k)^H$ is its Hermitian transpose. However, the periodogram is not a consistent estimator of the SDM. For consistency, it is common to smooth the periodogram and set

$$\hat{\boldsymbol{S}}(\omega_k) = \sum_{h=-\infty}^{\infty} w(h)\boldsymbol{I}(\omega_{k+h}) \tag{3.5}$$

for $w(h)$ a smoothing window that is symmetric and sums to one [45, 46].

The graphical LASSO for time series [42, 46] considers a regularized estimator for $\boldsymbol{\Theta}(\omega_k)$, $k = 0, \cdots, (T-1)/2$, by adding a group LASSO penalty to the Whittle approximation of the log-likelihood of the time series:

$$\hat{\boldsymbol{\Theta}}(\cdot) = \arg\min_{\boldsymbol{\Theta}(\cdot)} \frac{2}{T} \sum_{k=0}^{(T-1)/2} \Big\{ -\log\det \boldsymbol{\Theta}(\omega_k) + \mathrm{trace}\{\hat{\boldsymbol{S}}(\omega_k)\boldsymbol{\Theta}(\omega_k)\} \Big\} + \lambda \sum_{j \neq l} \sqrt{\frac{2}{T} \sum_{k=0}^{(T-1)/2} |\boldsymbol{\Theta}(\omega_k)_{jl}|^2} \quad (3.6)$$

where $\boldsymbol{\Theta}(\cdot) = (\boldsymbol{\Theta}(\omega_0), \cdots, \boldsymbol{\Theta}(\omega_{(T-1)/2}))$ is a sequence of Hermitian positive definite matrices such that $\boldsymbol{\Theta}(\omega_k) \in \mathbb{C}^{p \times p}$ for $k = 0, \cdots, (T-1)/2$, and λ is a non-negative tuning parameter. The group LASSO penalty regularizes the estimate, ensures that $\hat{\boldsymbol{\Theta}}(\cdot)$ is sparse, and that the zero paterns are shared across all frequencies. Once $\hat{\boldsymbol{\Theta}}(\cdot)$ is computed, it is straight forward to estimate the edge set E.

3.3 Structural Change-point Detection via Sparse Spectral Graphical Models

3.3.1 Problem Definition

Let $\boldsymbol{X}(t) = (X_1(t), \cdots, X_p(t))^\top$, for $t \in \mathbb{Z}$, be a multivariate time series, and let $\boldsymbol{x}(t) = (x_1(t), \cdots, x_p(t))^\top$ be a realization of the time series. Assume that the cross-correlation structure between the series remains constant for a certain period of time, and then changes to another constant state as the system evolves, which is a common assumption in practice [35, 39, 41]. In other words, the cross-correlation structure has only step-wise changes at certain time points, as illustrated in Figure 3.1 [35].

The main goal of this chapter is to detect the structural change-points, in real-time, while learning the different cross-correlation structures. For this purpose, we need to answer the following questions: (i) how to model the evolving cross-correlation structure between the time series and (ii) how to use the proposed model to quickly detect a system change while maintaining a pre-specified false alarm rate and an in-control ARL. To an-

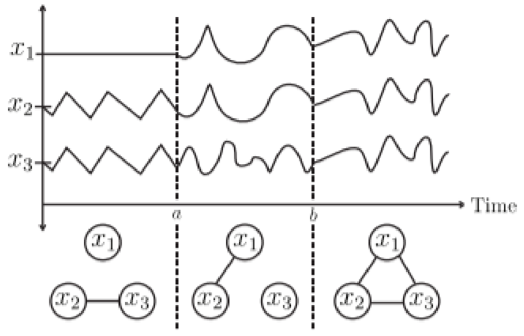


Figure 3.1: Three sensors with associated time series readings, and three different cross-correlation structures [35]

swer these questions, we make the following assumption. (A1) For each segment ι where the cross-correlation structure is fixed, the underlying time series is stationary with mean zero and autocovariance function $\mathbf{\Gamma}_\iota(h)$, $h \in \mathbb{Z}$, such that $\sum_{h=-\infty}^{\infty} ||\mathbf{\Gamma}_\iota(h)||_2 < \infty$.

3.3.2 Sparse Spectral Graphical Models for Non-stationary Time Series

For now, let's assume that the realization of the time series is fully observed. We have $\boldsymbol{x}(t) = (x_1(t), \cdots, x_p(t))^\top$, for $t = 1, \cdots, T$, where T is a fixed finite number. We are interested in learning the system's dynamic cross-correlation structure. For this purpose, we divide the data into overlapping windows of size W, where the overlap is O, as illustrated in Figure 3.2. In total, we have $N = (T - O)/(W - O)$ windows. A naive approach to learn the dynamic cross-correlation structure is to learn a graphical model for each window $n = 1, \cdots, N$. Based on assumption (A1), we suppose that the time series in each window are stationary and use the graphical LASSO for time series to estimate the inverse SDM, $\hat{\mathbf{\Theta}}_n(\omega_k)$, $k = 0, \cdots, (W-1)/2$, for each window $n = 1, \cdots, N$. However, this approach allows for too much variability as different structures can be learned for each window.

To generate temporally consistent graphs, we propose to jointly learn the N graphical models, ensuring that the sparsity pattern is shared across windows. The estimated inverse

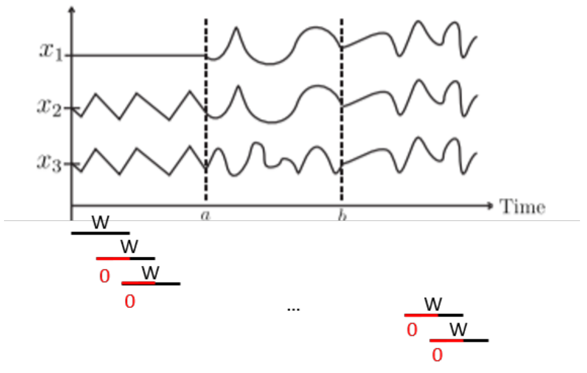


Figure 3.2: Overlapping windows to learn the dynamic cross-correlation structure of a system with three sensors. W is the window size and O is the overlap.

SDM are found by solving the following optimization problem:

$$\min_{\substack{\boldsymbol{\Theta}_n(\cdot)\\ n=1,\cdots,N}} \frac{2}{NW}\sum_{n=1}^{N}\sum_{k=0}^{(W-1)/2}\Big\{-\log\det\boldsymbol{\Theta}_n(\omega_k)+\text{trace}\{\hat{\boldsymbol{S}}_n(\omega_k)\boldsymbol{\Theta}_n(\omega_k)\}\Big\} \\ +\lambda\sum_{j\neq l}\sqrt{\frac{2}{NW}\sum_{n=1}^{N}\sum_{k=0}^{(W-1)/2}|\boldsymbol{\Theta}_n(\omega_k)_{jl}|^2} \quad (3.7)$$

where $\boldsymbol{\Theta}_n(\cdot)=(\boldsymbol{\Theta}_n(\omega_0),\cdots,\boldsymbol{\Theta}_n(\omega_{(W-1)/2}))$, $n=1,\cdots,N$, is a sequence of Hermitian positive definite matrices such that $\boldsymbol{\Theta}_n(\omega_k)\in\mathbb{C}^{p\times p}$, $\hat{\boldsymbol{S}}_n(\omega_k)$ is the estimate of the SDM at frequency ω_k, $k=0,\cdots,(W-1)/2$, and λ is a non-negative tuning parameter. The last term is a group LASSO penalty which ensures that the structure learned for each window is sparse by enforcing zeros across frequencies. Additionally, the penalty term regularizes the pattern of sparsity across windows to ensure the estimation of temporally consistent graphs. By estimating the inverse SDM for each window, we are able to estimate the corresponding edge set, as we have seen that there is a one-to-one correspondence between the two, and, thus, the system's dynamic cross-correlation structure.

Even though the idea of jointly estimating graphical models is not new [33, 34, 35], the proposed methodology has two main advantages. First, it is able to learn the system's dynamic cross-correlation structure with only one sample of observations at each point in time. Existing approaches estimate one graph per sampling time $t=1,\cdots,T$, which requires the number of samples M to be greater than one. Second, the proposed method models the temporal cross-correlation of the data while existing approaches ignore it by assuming that observations across time are independent. Modeling the temporal cross-correlation is critical to avoid learning fictitious dynamics over time.

So far, we have assumed that the realization of the time series is fully observed. However, in practice, for streaming data, at each point in time, we receive a new observation. We need to be able to update, in real-time, the cross-correlation structure of the series as new data is received. Additionally, we are interested in detecting a structural change. Next, we extend the proposed approach to be able to handle streaming time series.

3.3.3 Sparse Spectral Graphical Models for HD Streaming Data

Assume that we observe a realization of the time series $\boldsymbol{x}(t) = (x_1(t), \cdots, x_p(t))^\top$, for $t = 1, 2, \cdots$. Since our goal is to detect a structural change-point we do not need to consider all historical data. Hence, we learn the structure for the most recent N windows. The optimization problem that we want to solve is:

$$\min_{\substack{\boldsymbol{\Theta}_n(\cdot)\\ n=1,\cdots,N}} \frac{2}{NW} \sum_{n=1}^{N} \sum_{k=0}^{(W-1)/2} \Big\{ -\log\det \boldsymbol{\Theta}_n(\omega_k) + \operatorname{trace}\{\hat{\boldsymbol{S}}_n(\omega_k)\boldsymbol{\Theta}_n(\omega_k)\} \Big\} + \lambda \sum_{j \neq l} \sqrt{\frac{2}{NW} \sum_{n=1}^{N} \sum_{k=0}^{(W-1)/2} |\boldsymbol{\Theta}_n(\omega_k)_{jl}|^2} \tag{3.8}$$

where $\boldsymbol{\Theta}_n(\cdot) = (\boldsymbol{\Theta}_n(\omega_0), \cdots, \boldsymbol{\Theta}_n(\omega_{(W-1)/2}))$, $n = 1, \cdots, N$, is a sequence of Hermitian positive definite matrices such that $\boldsymbol{\Theta}_n(\omega_k) \in \mathbb{C}^{p\times p}$, $\hat{\boldsymbol{S}}_n(\omega_k)$ is the estimate of the SDM at frequency ω_k, $k = 0, \cdots, (W-1)/2$, and λ is a non-negative tuning parameter. The last term is a group LASSO penalty which ensures that the structure learned for each window is sparse by enforcing zeros across frequencies. Additionally, the penalty term regularizes the pattern of sparsity across windows to ensure the estimation of temporally consistent graphs. By estimating the inverse SDM for each window, we are able to estimate the corresponding edge set, as we have seen that there is a one-to-one correspondence between the two, and, thus, the system's dynamic cross-correlation structure.

We solve the problem in Equation 3.8 using the ADMM algorithm [47]. To split the problem, we introduce consensus variables $\boldsymbol{Z}_n(\omega_k)$, $k = 0, \cdots, (W-1)/2$ and $n = 1, \cdots, N$. With this we can rewrite the problem as:

$$\begin{aligned}
\min_{\substack{\boldsymbol{\Theta}_n(\cdot), \boldsymbol{Z}_n(\cdot) \\ n=1,\cdots,N}} \quad & \frac{2}{NW} \sum_{n=1}^{N} \sum_{k=0}^{(W-1)/2} \Big\{ -\log\det \boldsymbol{\Theta}_n(\omega_k) + \operatorname{trace}\{\hat{\boldsymbol{S}}_n(\omega_k)\boldsymbol{\Theta}_n(\omega_k)\} \Big\} \\
& + \lambda \sum_{j \neq l} \sqrt{\frac{2}{NW} \sum_{n=1}^{N} \sum_{k=0}^{(W-1)/2} |\boldsymbol{Z}_n(\omega_k)_{jl}|^2} \\
\text{s.t.} \quad & \boldsymbol{\Theta}_n(\cdot) = \boldsymbol{Z}_n(\cdot) \quad \text{for} \quad n = 1, \cdots, N
\end{aligned} \tag{3.9}$$

where $\boldsymbol{\Theta}_n(\cdot) = (\boldsymbol{\Theta}_n(\omega_0), \cdots, \boldsymbol{\Theta}_n(\omega_{(W-1)/2}))$ and $\boldsymbol{Z}_n(\cdot) = (\boldsymbol{Z}_n(\omega_0), \cdots, \boldsymbol{Z}_n(\omega_{(W-1)/2}))$, $n = 1, \cdots, N$.

The corresponding augmented Lagrangian becomes

$$\begin{aligned}
\mathcal{L}_\rho(\boldsymbol{\Theta}, \boldsymbol{Z}, \boldsymbol{U}) =& \frac{2}{NW} \sum_{n=1}^{N} \sum_{k=0}^{(W-1)/2} \Big\{ -\log\det \boldsymbol{\Theta}_n(\omega_k) + \operatorname{trace}\{\hat{\boldsymbol{S}}_n(\omega_k)\boldsymbol{\Theta}_n(\omega_k)\} \Big\} \\
& + \lambda \sum_{j \neq l} \sqrt{\frac{2}{NW} \sum_{n=1}^{N} \sum_{k=0}^{(W-1)/2} |\boldsymbol{Z}_n(\omega_k)_{jl}|^2} \\
& + \frac{\rho}{NW} \sum_{n=1}^{N} \sum_{k=0}^{(W-1)/2} \{ ||\boldsymbol{\Theta}_n(\omega_k) - \boldsymbol{Z}_n(\omega_k) + \boldsymbol{U}_n(\omega_k)||_F^2 - ||\boldsymbol{U}_n(\omega_k)||_F^2 \}
\end{aligned} \tag{3.10}$$

where $\boldsymbol{U}_n(\cdot) = (\boldsymbol{U}_n(\omega_0), \cdots, \boldsymbol{U}_n(\omega_{(W-1)/2}))$, $n = 1, \cdots, N$ are the scaled dual variables and $\rho > 0$ is the ADMM penalty parameter [47]. The ADMM consists of the following updates where s denotes the iteration number:

- $\boldsymbol{\Theta}^{s+1} = \arg\min_{\substack{\boldsymbol{\Theta}_n(\cdot) \\ n=1,\cdots,N}} \mathcal{L}_\rho(\boldsymbol{\Theta}, \boldsymbol{Z}^s, \boldsymbol{U}^s)$
- $\boldsymbol{Z}^{s+1} = \arg\min_{\substack{\boldsymbol{Z}_n(\cdot) \\ n=1,\cdots,N}} \mathcal{L}_\rho(\boldsymbol{\Theta}^{s+1}, \boldsymbol{Z}, \boldsymbol{U}^s)$

- $\boldsymbol{U}^{s+1} = \boldsymbol{U}^{s} + \boldsymbol{\Theta}^{s+1} - \boldsymbol{Z}^{s+1}$

By separating the problem in Equation 3.8 into two blocks of variables, $\boldsymbol{\Theta}$ and $\boldsymbol{Z}$, the proposed ADMM approach is guaranteed to converge to the global optimum [47]. Our iterative algorithm uses a stopping criterion based on the primal and dual residual values being below specified thresholds as recommended in [47].

$\boldsymbol{\Theta}$ Update

The $\boldsymbol{\Theta}$ step can be split into separate updates for each $\boldsymbol{\Theta}_n(\omega_k)$, $n = 1, \cdots, N$, $k = 0, \cdots, (W-1)/2$, which can then be solved in parallel:

$$\boldsymbol{\Theta}_n^{s+1}(\omega_k) = \arg \min_{\boldsymbol{\Theta}_n(\omega_k)} -\log\det \boldsymbol{\Theta}_n(\omega_k) + \operatorname{trace}\{\hat{\boldsymbol{S}}_n(\omega_k)\boldsymbol{\Theta}_n(\omega_k)\} + \frac{\rho}{2}||\boldsymbol{\Theta}_n(\omega_k) - \boldsymbol{Z}_n^s(\omega_k) + \boldsymbol{U}_n^s(\omega_k)||_F^2 \quad (3.11)$$

Let $\boldsymbol{Q}\boldsymbol{D}\boldsymbol{Q}^H$ denote the eigen-decomposition of $\hat{\boldsymbol{S}}_n(\omega_k) - \rho\left(\boldsymbol{Z}_n^s(\omega_k) - \boldsymbol{U}_n^s(\omega_k)\right)$. The solution to the problem presented in Equation 3.11 is given by $\boldsymbol{Q}\tilde{\boldsymbol{D}}\boldsymbol{Q}^H$, where $\tilde{D}$ is a diagonal matrix with jth diagonal element $(-D_{jj} + \sqrt{D_{jj}^2 + 4\rho})/(2\rho)$ [48]. Then, the $\boldsymbol{\Theta}$ update is

$$\boldsymbol{\Theta}_n^{s+1}(\omega_k) = \boldsymbol{Q}\tilde{\boldsymbol{D}}\boldsymbol{Q}^H \quad (3.12)$$

for $n = 1, \cdots, N$ and $k = 0, \cdots, (W-1)/2$.

$\boldsymbol{Z}$ Update

For the $\boldsymbol{Z}$ step, we need to solve:

$$\boldsymbol{Z}^{s+1} = \arg \min_{\substack{\boldsymbol{Z}_n(\cdot) \\ n=1,\cdots,N}} \lambda \sum_{j \neq l} \sqrt{\frac{2}{NW} \sum_{n=1}^{N} \sum_{k=0}^{(W-1)/2} |\boldsymbol{Z}_n(\omega_k)_{jl}|^2} + \frac{\rho}{NW} \sum_{n=1}^{N} \sum_{k=0}^{(W-1)/2} ||\boldsymbol{\Theta}_n^{s+1}(\omega_k) - \boldsymbol{Z}_n(\omega_k) + \boldsymbol{U}_n^s(\omega_k)||_F^2 \quad (3.13)$$

Let $\boldsymbol{A}_n(\omega_k) = \boldsymbol{\Theta}_n^{s+1}(\omega_k) + \boldsymbol{U}_n^s(\omega_k)$. Since the minimization problem in Equation 3.13 has the form of a group LASSO problem, its solution is

$$\hat{\boldsymbol{Z}}_n^{s+1}(\omega_k)_{jl} = \left(1 - \frac{\lambda}{\rho\sqrt{\frac{2}{NW}\sum_{n'=1}^{N}\sum_{k'=0}^{(W-1)/2}|A_{n'}(\omega_{k'})_{jl}|}}\right)_+ \boldsymbol{A}_n(\omega_k)_{jl}, \qquad (3.14)$$

for $n = 1, \cdots, N$, $k = 0, \cdots, (W-1)/2$, and $j, l = 1, \cdots, p$ [42, 47], where $(\cdot)_+ = \max(0, \cdot)$.

By iteratively solving the problem in Equation 3.8, using the ADMM, for streaming windows of HD data, we can sequentially estimate the system's dynamic cross-correlation structure. One thing left to discuss is how to tune the different model parameters. The proposed method has five parameters: the window size W, the overlap between windows O, the number of windows considered at each point in time N, the penalty parameter λ, and the ADMM paramter ρ. Next, we provide guidelines for selecting these parameters.

- Window size W: This parameter needs to be sufficiently large to capture the spectral information contained in the data, it also needs to be sufficiently small to satisfy the assumption of the time series being stationary inside the window. We recommend using the Nyquist-Shannon sampling theorem [49] in combination with domain knowledge about the system to select W.

- Overlap between windows O: This parameter controls the number of new observations considered in each iteration of the proposed methodology. If it is too small it will hinder the detection capability, but if it is too large it will unnecessarily increase the computational time. This parameter should be chosen based on domain knowledge about the system.

- Number of windows considered N: This parameter needs to be sufficiently large to guaranty the time consistency of the sparse spectral graphical models. However, if it is too large, the computational time would become prohibitive for online change-point detection. In practice, we recommend choosing this parameter based on the frequency at which data is being collected to ensure online monitoring is feasible.

- Penalty parameter λ: This parameter controls the level of sparseness of the graphs, as well as the level of similarity between neighboring windows. If there is domain knowledge about the level of sparsity in the system, it can be used to tune this parameter. If not, AIC or BIC methods that control the model complexity can be used.
- ADMM paramter ρ: This parameter controls the convergence speed of the ADMM. In [47], Boyd et al. provide guidelines for selecting it. In practice, the selection of ρ does not have a considerable impact on the computational time of the proposed method. For the simulations and case studies of this chapter, we used $\rho = 1$.

3.3.4 Online Structural Change-point Detection

With the proposed sparse spectral graphical model for streaming time series, we are able to learn the dynamic cross-correlation structure of the data. However, we still need to explain how to monitor the learning process to be able to detect a structural change.

At every iteration ι, the performance of the proposed method can be assessed by studying the resulting approximate log-likelihood:

$$ll_\iota = \frac{2}{NW} \sum_{n=1}^{N} \sum_{k=0}^{(W-1)/2} \left\{ \log\det \mathbf{\Theta}_n^{(\iota)}(\omega_k) - \mathrm{trace}\{\hat{\boldsymbol{S}}_n^{(\iota)}(\omega_k)\mathbf{\Theta}_n^{(\iota)}(\omega_k)\} \right\}. \tag{3.15}$$

A higher value of ll_ι implies a better performance. When there is a structural change, the performance, and thus the value of ll_ι, will decrease as the group LASSO penalty in Equation 3.8 enforces the similarity across the N windows under study, even when there is a change. Therefore, a structural change can be detected by a sudden decrease in the value of the approximate log-likelihood.

We use an EWMA control chart. The EWMA monitoring statistic, for all $\iota > 1$, is given by

$$z_\iota = \gamma ll_\iota + (1-\gamma) z_{\iota-1}, \tag{3.16}$$

where $\gamma \in (0,1)$ is a weight parameter that controls the importance of historical data, and $z_1 = ll_1$. In the simulations and case studies, we use $\gamma = 0.5$. The stopping time of the monitoring procedure can be determined by:

$$T(h) = \inf\{\iota | \; z_\iota \geq h\}, \tag{3.17}$$

where h is the control limit estimated based on the empirical in-control distribution of z_ι [50]. The value of h is related to the pre-specified desired in-control ARL of the monitoring scheme, when no change occurs in the system.

3.4 Performance Evaluation via Simulations

In this section, we evaluate the performance of the proposed method using simulations. First, we will illustrate the detection capability of the proposed method. Then, we will compare the proposed method with the state-of-art method. Finally, we will show the performance of the learning algorithm.

In our simulations, the time series has T sampling time points, and a structural change-point at time τ ($0 < \tau < T$). Therefore, we have two time segments where the structure remains unchanged. For the first time segment, $t = 1, \cdots, \tau - 1$, we construct a vector autoregressive process VAR(1), with variables $\boldsymbol{X}(t) = (X_1(t), \cdots, X_p(t))^\top$ such that

$$\boldsymbol{X}(t) = \boldsymbol{A}\boldsymbol{X}(t-1) + \boldsymbol{\epsilon}(t) \tag{3.18}$$

where $\boldsymbol{\epsilon}(t) \sim N(0, \sigma \boldsymbol{I}_p)$, $\boldsymbol{A} \in \mathbb{R}^{p\times p}$ is the process transition matrix, and $\boldsymbol{I}_p$ is the $p \times p$ identity matrix. To construct $\boldsymbol{A}$, we first set the diagonal entries to be 0.2. Then, in each row of $\boldsymbol{A}$, we randomly select one entry and set it to -0.5 with probability 0.5, or to 0.5

with probability 0.5. Lastly, we divide $\boldsymbol{A}$ by its maximum eigenvalue in order to make the process stable. We set $\sigma = ||\boldsymbol{A}||_2/2$.

The structure of the first time segment, can be determined as follows. If $\boldsymbol{A}_{jl} \neq 0$, we know that node l influences node j by construction of the VAR(1) process. Let $S_l^{(A)} = \{j : \boldsymbol{A}_{jl} \neq 0, j \in V\}$, for $l \in V$. We have that node j is conditionally independent of node l given the remaining $p-2$ variables, if $S_j^{(A)} \cap S_l^{(A)} = \emptyset$. Therefore, $E^{(A)} = \{(j,l) : S_j^{(A)} \cap S_l^{(A)} \neq \emptyset : (j,l) \in V \times V, j \neq l\}$.

To generate the second segment, $t = \tau, \cdots, T$, we construct a VAR(1) process such that

$$\boldsymbol{X}(t) = \boldsymbol{B}\boldsymbol{X}(t-1) + \boldsymbol{\epsilon}(t) \tag{3.19}$$

where $\boldsymbol{\epsilon}(t) \sim N(0, \sigma \boldsymbol{I}_p)$ and $\boldsymbol{B} \in \mathbb{R}^{p \times p}$ is the process transition matrix. To construct $\boldsymbol{B}$, we first set $\boldsymbol{B} = \boldsymbol{A}$. Then, we randomly sample c of the non-diagonal, non-zero entries of $\boldsymbol{A}$ and c of the zero entries, and switch their values. If we define $S_l^{(B)} = \{j : \boldsymbol{B}_{jl} \neq 0, j \in V\}$, the structure of the second segment is given by $E^{(B)} = \{(j,l) : S_j^{(B)} \cap S_l^{(B)} \neq \emptyset : (j,l) \in V \times V, j \neq l\}$.

In the simulations, we use $p = 10$, $T = 2,000$ and $\tau = 1,000$. Therefore, $\boldsymbol{A}$ has 10 non-diagonal, non-zero entries. We set $c = \lfloor 10 \cdot SNR \rfloor$, where $\lfloor 10 \cdot SNR \rfloor$ is the greatest integer smaller than or equal to $10 \cdot SNR$, and the SNR is defined as the percentage of the non-diagonal, non-zero entries modified. We considered three different levels of SNR: 0.25, 0.5, and 1.00.

For the proposed method, we set the window size $W = 200$, the overlap $O = 190$, the number of windows $N = 3$, and the ADMM parameter $\rho = 1$. The penalty parameter λ is tuned by fixing the sparsity level at 10. To test the detection power of the proposed method, we fix the in-control ARL0 to be 200 and compare the out-of-control ARL1 under the different SNR. The out-of-control ARLs obtained from 100 simulation replicates are shown in Figure 3.3. As expected, as the SNR increases, the structural change is easier to detect, i.e., smaller ARL1 values. The next step is to compare our method, denoted

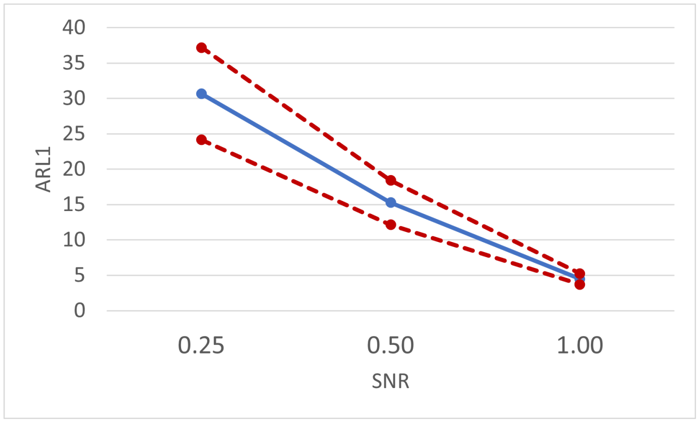


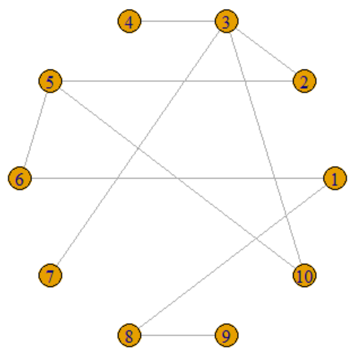
Figure 3.3: Detection power of the proposed method. Solid blue curve is the ARL1, pointed red curves represent the confidence interval

Table 3.1: Comparison of detection power for SNR=1.00

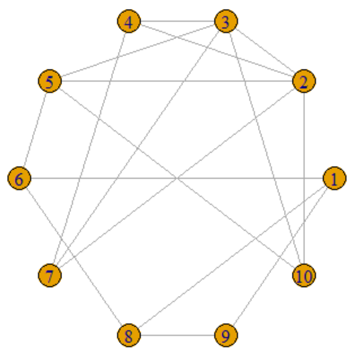
	DSSGM	DSSL
ARL1	4.48	15.60
St. error	0.39	2.76

as Dynamic Sparse Spectral Graphical Model (DSSGM), with the state-of-art method, denoted Dynamic Sparse Subspace Learning (DSSL). The DSSL method is based on dynamic sparse subspace learning [41]. We compare both methods using the simulation setting where SNR=1.00. For the DSSL method, we used the best parameters we could find by following the authors recommendations on parameter selection. The results are presented in Table 3.1. We observe that the detection capability of our method is better. We are able to detect a change faster because we incorporate the temporal cross-correlation of the data in our model.

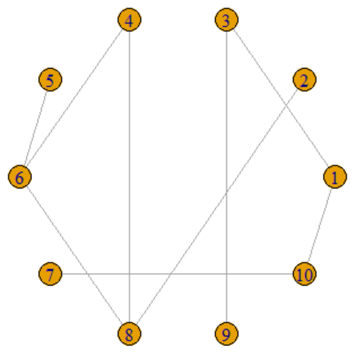
An additional advantage of the DSSGM is that it is able to learn the structure of the data in real-time, while the DSSL method cannot. In Figure 3.4, we present the true structures for segments 1 and 2, as well as the learned structures, for one simulation replicate of the setting SNR=1.00. We observe that existing edges are correctly estimated, however, the method estimates additional edges. To evaluate the overall performance of the learning method, we compute the following three criteria for each segment: (i) precision, defined as the proportion of learned edges that are true edges, (ii) recall, defined as the proportion of learned edges that are correctly identified, and (iii) F-score, a single criterion that combines precision and recall by calculating their harmonic mean. A boxplot for each metric and each segment is presented in Figure 3.5 for the setting SNR=1.00. Similar results are obtained for other simulation settings, they are presented in Appendix B.



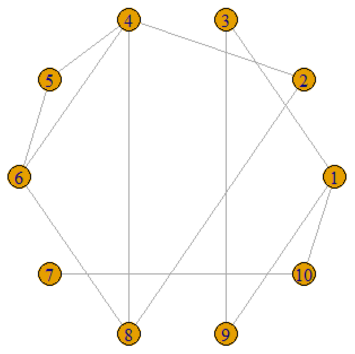
(a) True Structure 1



(b) Estimated structure 1

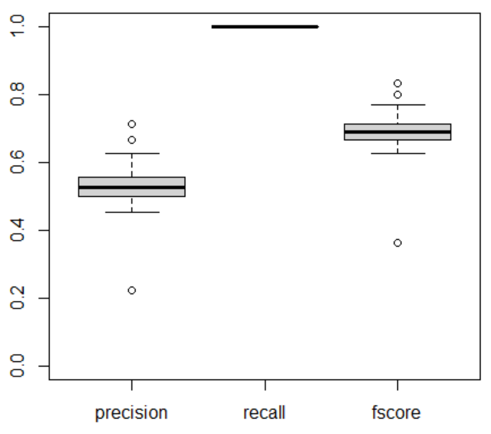


(c) True structure 2

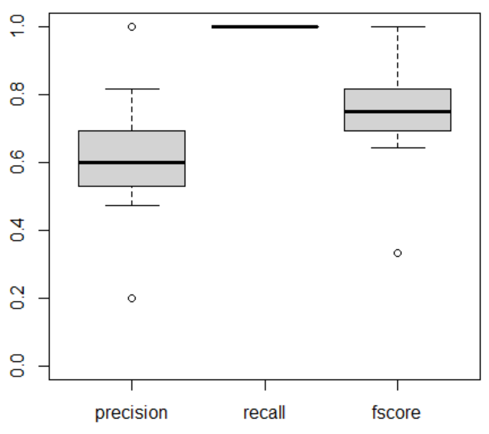


(d) Estimated structure 2

Figure 3.4: Example of true and estimated structures for one simulation run, for SNR=1.00



(a) Structure 1



(b) Structure 2

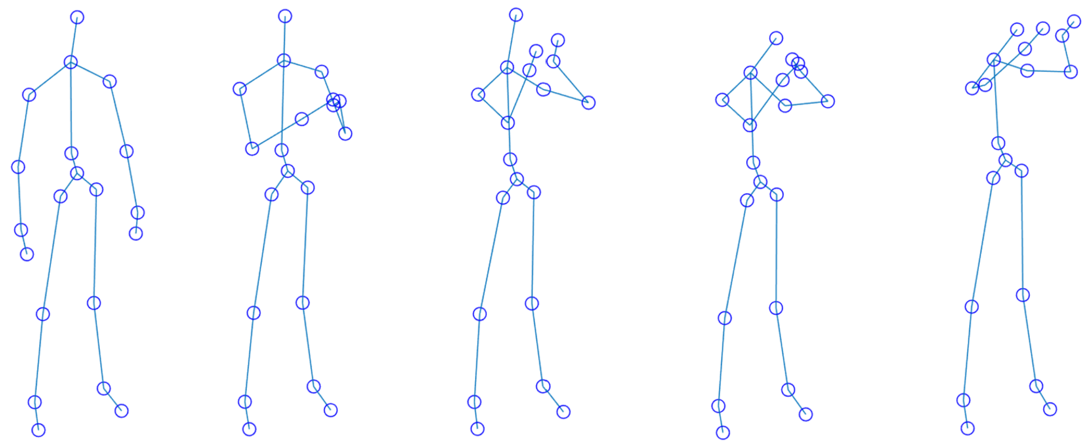
Figure 3.5: Structure learning performance for SNR=1.00, in terms of precision, recall,and F-score, over 100 simulation replicates

3.5 Case Studies

3.5.1 Human Gesture Tracking

One application where the proposed methodology can be used is motion segmentation, which is a critical step for human gesture recognition. Kinect sensors track the movements of human subjects by capturing the location of 18 body joints. Their data acquisition rate is 30 Hz with 2 cm accuracy in the joint positions. Every joint is recorded as a point in a three-dimensional Cartesian coordinate system. The MSRC-12 gesture dataset [7] consists of sequences of human gestures performed by 30 subjects and tracked by Kinect sensors. The position of the joints and some snapshots of the *shoot* and *throw* gestures are shown in Figure 3.6.

To evaluate the performance of the proposed methodology in motion segmentation, we combine the sequences of two gestures, *shoot* and *throw*, performed by the same subject, which leads to a time series with 239 observations. In the first segment, the subject stretches his arms out in front of him, holding a pistol, makes a recoil movement and then returns


(a) Five snapshots of the shoot gesture

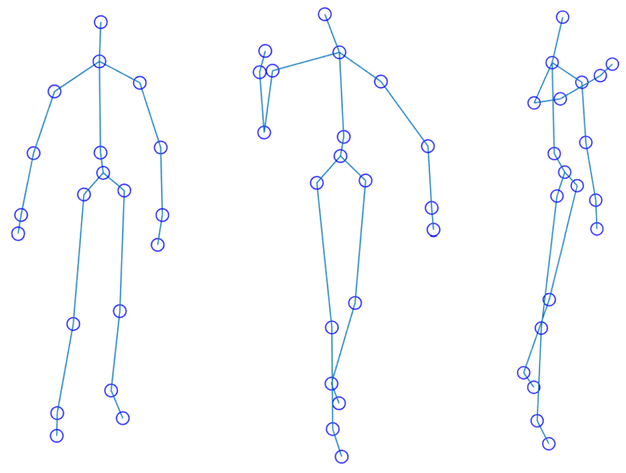

(b) Five snapshots of the throw gesture

Figure 3.6: Snapshots of gestures being tracked by a Kinect sensor

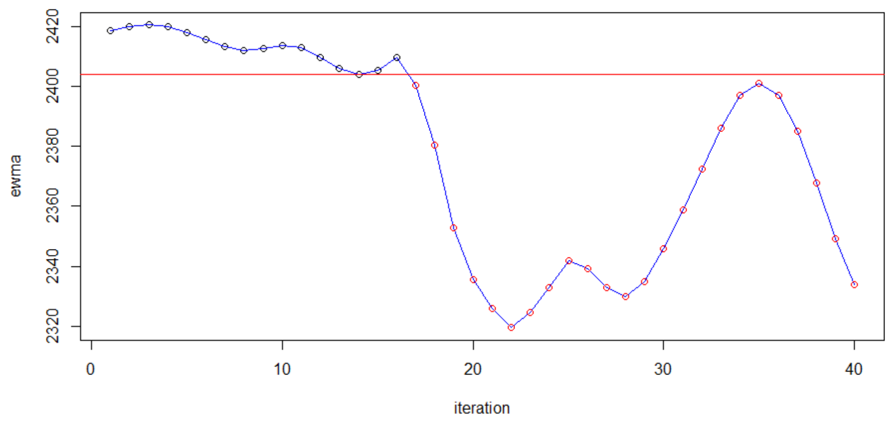
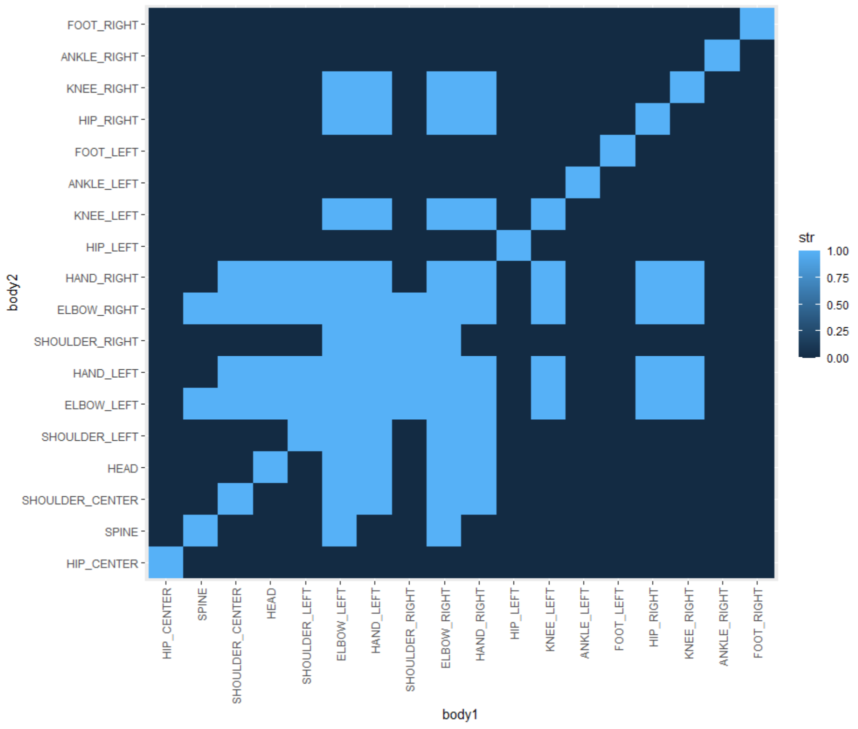


Figure 3.7: Control chart for motion segmentation

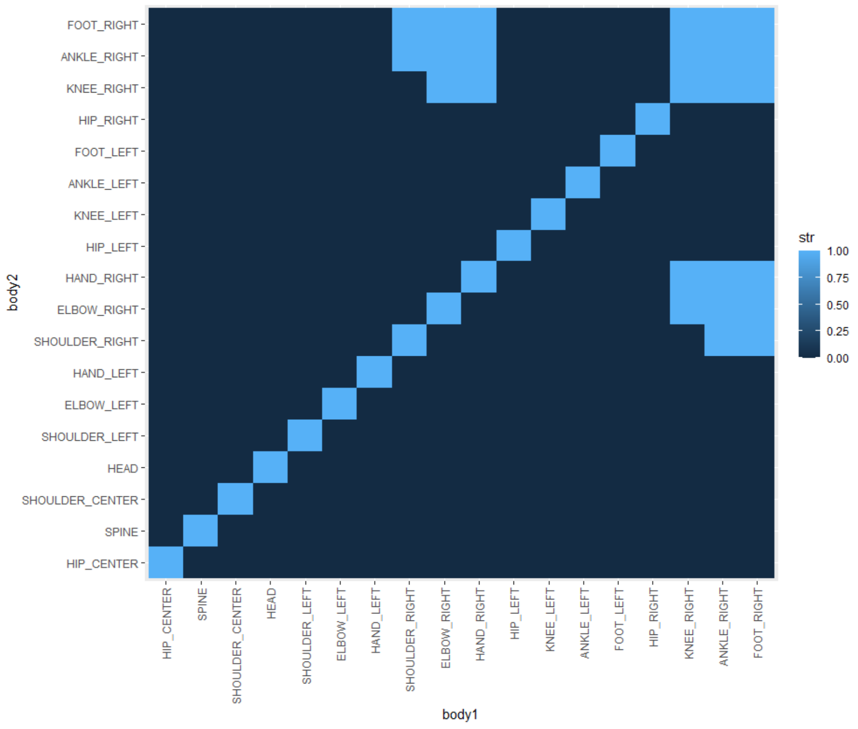
to the initial position. In the second segment, the subject uses his right arm to make an overarm throwing movement, and then returns to the initial position. The gesture change occurs at $t = 119$.

To favor the interpretability of the graphs learned, we convert the Cartesian coordinates of each joint into a single distance variable, representing the distance of the joint to a reference point established by the Kinect sensor. Our goal is to detect when the gesture changes. For this purpose, we set $W = 30$, $O = 25$, and $N = 3$. The monitoring control chart is presented in Figure 3.7. We detect the change at iteration 17, which corresponds to $t = 130$, which leads to a detection delay of 11 time observations.

The structures learned for each gesture are presented in Figure 3.8. We see that for the *shoot* gesture, the hands, elbows, and shoulders from both arms are connected. The structure is consistent with the gesture, as both arms are performing similar movements to shoot. The graph learned for the *throw* gesture is sparser. In this case, the right side of the body is connected (foot, ankle, knee, hand, elbow, shoulder). This is consistent with the fact that the subject involves the right side of the body in the throwing movement.



(a) Body structure for shooting



(b) Body structure for throwing

Figure 3.8: Body structures learned for each gesture

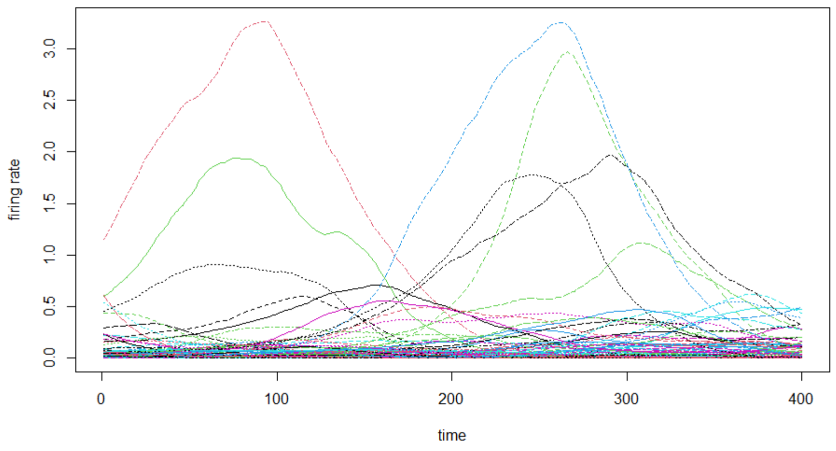


Figure 3.9: Simulated firing rates of 80 brain regions over 40 seconds

3.5.2 Monitoring Dynamic Functional Brain Connectivity

The proposed methodology can also be used to model the dynamic Functioanal Connectivity (FC) of the brain. FC is the statistical dependency between distinct brain regions and has been an important tool in understanding brain processes [51]. Recently, FC has been shown to fluctuate over time, implying that the cross-correlation structure between different neuron populations changes over time [52]. Thus, assessing and characterizing the dynamic FC is critical to understand how our brain works.

To illustrate how the proposed methodology can learn the dynamic FC, we simulate whole-brain dynamics using an example dataset from *neurolib*, a computational framework for simulating coupled neural mass models [53]. We consider the firing rates of 80 ecxitatory populations of neurons (brain regions) during 40 seconds at a rate of 10Hz, which leads to 400 time observations. The signals can be observed in Figure 3.9. We set $W = 200$, $O = 190$, and $N = 3$, and use the DSSGM to learn the dynamic FC.

The monitoring control chart is presented in Figure 3.10. We detect a change in the FC at iteration 12. The cross-correlations structures learned for each segment can be seen in Figure 3.11. For the first cross-correlation structure, we observe that regions 14, 57, and

64 are highly active as they are conditionally dependent on many other regions. For the second structure, we observed that the number of active regions increases. Regions 45, 49, 57, 64, and 73 are conditionally dependent on other regions. Additionally, we can see that the number of connections between regions 60 to 80 increases. So far, given the way the example dataset is simulated by *neurolib* [53] we ignore the ground truth, and do not have a reliable way to validate our results. In the near future, we will explore how to generate the simulations by defining the structural connectivity matrices in order to be able to validate our results.

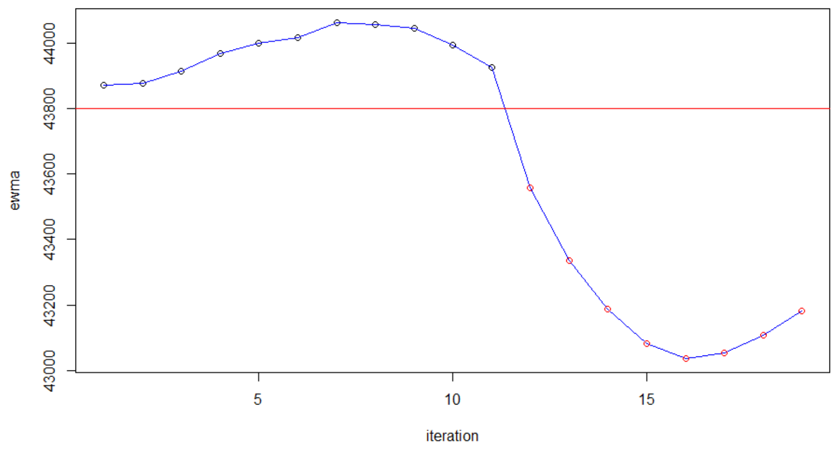
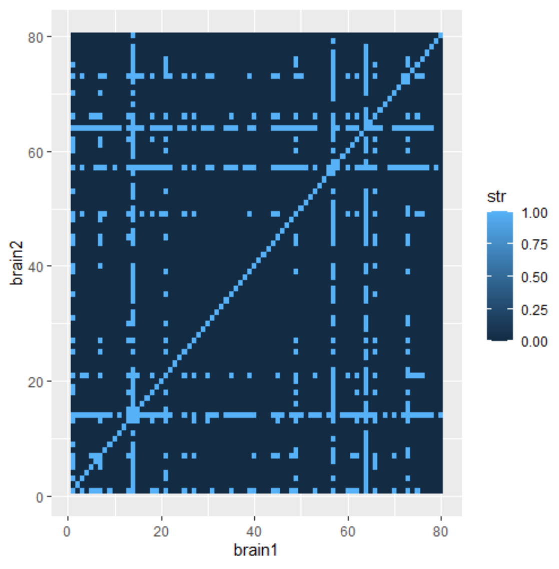
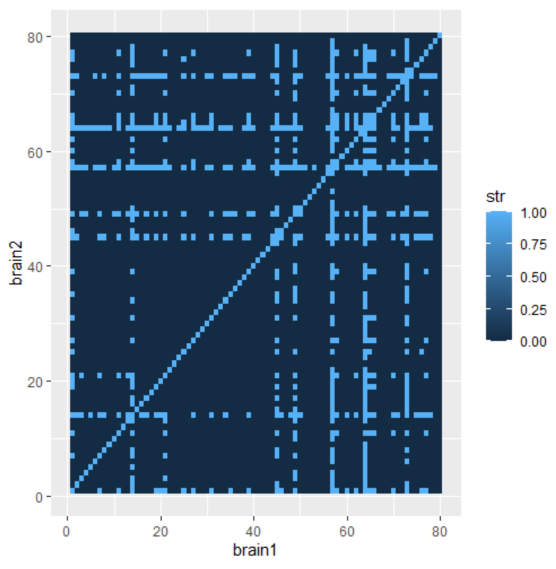


Figure 3.10: Control chart for dynamic FC



(a) FC 1



(b) FC 2

Figure 3.11: FC for two identified segments

3.6 Conclusion

This chapter proposed a novel method to learn the evolving cross-correlation structure of systems collecting HD streaming data. At each point in time, the method is able to represent the system as a graph, where the nodes represent the system's entities and the edges represent the system's conditional dependence structure. Specifically, if there is an edge between node j and node l, we can conclude that these entities are conditionally dependent given all other variables in the system. The proposed method is not only capable of learning the cross-correlation structure, it can also detect, in real-time, structural change-points in the system.

To learn the system's evolving cross-correlation structure, we partition the streaming data into streaming windows. Then, we assume that, inside each window, the time series is stationary and use sparse spectral graphical models to estimate the structure of the system. By exploiting the spectral information contained in the data, we are able to model the temporal cross-correlation, which represents an advantage against existing methods. Additionally, to ensure that we obtain temporally consistent graphs from one window to the next, we regularize the change using a group LASSO penalty. Finally, to detect a structural change-point, we use an EWMA control chart that monitors the performance of the sparse spectral graphical model. The proposed method is efficiently implemented by applying the ADMM and can be used in real-time in many applications.

With the simulation experiments and the two case studies, we showed that the proposed method is able to learn the evolving cross-correlation structure of a system. Furthermore, we showed that it is also capable of timely identifying the structural change-points. The proposed method presents two main advantages against existing methods. First, it incorporates the temporal cross-correlation of the data improving its detection capability. Second, it is able to detect structural changes in the system while simultaneously learning the structure.

In the near future, we want to expand our simulation experiments. We want to compare with the benchmark method in different scenarios to test the detection capability of our method. Additionally, we would like to perform a sensitivity analysis to explore how each parameter affects the performance of the DSSGM. Furthermore, we want to further explore the simulation capabilities of *neurolib* to be able to validate the results obtained with the proposed method. We also plan to use our methodology in real neurological datasets, collected by EEG sensors, to investigate how the structure of our brain changes over time. We are also interested in expanding our methodology for network classification. Consider the human gesture tracking example. If we have enough historical data recorded when a subject performs different tasks, we can train a gesture classification model using graphical representations of each gesture. Then, with the DSSGM we can learn the current structure, in real-time, and classify it.

CHAPTER 4
AN ADAPTIVE SAMPLING STRATEGY FOR ONLINE MONITORING AND DIAGNOSIS OF HIGH-DIMENSIONAL STREAMING DATA

4.1 Introduction

Nowadays, most complex systems are continuously monitored by hundreds of sensors that provide a variety of spatio-temporal streaming data with rich information about the system's performance. Monitoring such HD streaming data, in real-time, is critical to detect anomalies and system failures. For example, in power distribution systems, smart sensors located across the distribution grid are used for the detection and isolation of outages [11, 12]. In manufacturing, a large number of process variables are measured during the production for online quality insurance [13, 14, 15]. In healthcare, intracranial EEG sensors are used for early detection of seizures in epileptic patients, and to identify the brain regions responsible for the seizures [54].

While online process monitoring and diagnosis are crucial in many applications, the complex characteristics of the HD streaming data pose some analytical challenges yet to be addressed. One of these challenges arises when only partial observations are available for monitoring due to the system's resource constraints. These constraints are common in practice. For example, in environmental monitoring, sensors have a limited battery lifetime, and the cost of replacing the battery is often high. Therefore, monitoring micro-climate for agriculture [55], forest fires, [56], and volcanic earthquakes [57], needs to be done with a limited number of operational sensors, at any given time. Partial observations can also arise when the number of available sensors is small compared to the number of variables for monitoring. For example, water quality monitoring requires wireless water sensor technology together with unmanned surface vehicles [58], an expensive pair of equipment. Finally, transmission and processing constraints can limit the amount of available data. When analyzing streaming images for detecting process changes [59], the high resolution and high acquisition rate of images hinder the monitoring process due to the limited bandwidth and processing capacity. Hence, only a subset of the pixels can be analyzed in real-time. In all of these settings, a sequential sampling strategy needs to be implemented to decide where to collect data in order to maximize the change detectability. Furthermore, in HD streaming data, neighboring measurements exhibit high spatial correlation, while measurements across time are temporally correlated with a non-stationary behavior. This spatio-temporal structure of the HD data poses another challenge for online monitoring and diagnosis.

In the literature, a series of papers has focused on sequential sampling for online monitoring of HD streaming data. Liu et al. (2015) proposed an adaptive sampling algorithm based on local Cumulative Sum (CUSUM) statistics. Under the assumption of regional shifts, Wang et al. (2018) proposed a spatial-adaptive sampling strategy where a wide search strategy is combined with a deep search strategy to speed up the detection of abnormal regions. The drawback of these methods is that they assume that the observations from all the data streams are independent and follow a normal distribution. To overcome this limitation, Xian et al. (2018) developed a non-parametric adaptive sampling strategy. The authors adopted a rank-based, non-parametric CUSUM procedure as a baseline to construct the monitoring statistic. To deal with the unobserved variables, they proposed a methodology to correct the anti-rank statistics. However, this adaptive sampling strategy still assumes that the data streams are independent and identically distributed with an in-control joint distribution, which may not be valid for spatio-temporal streaming data. To the best of our knowledge, none of the existing sequential sampling methods considers the spatio-temporal correlation of the HD streaming data.

In the absence of missing data due to resource constraints, monitoring methods that incorporate the spatio-temporal structure of the HD streaming data have been developed. These methods reduce the dimensionality of the problem by projecting the HD data streams onto Low-dimensional (LD) subspaces. The extracted features and residuals are used for monitoring and diagnosis [63, 64, 65]. The main drawback of these methods is that they cannot be directly used for non-stationary data streams. Recently, a Spatio-temporal Smooth Sparse Decomposition (ST-SSD) method was proposed to monitor non-stationary data streams [15]. However, the ST-SSD method cannot perform monitoring in a setting where the available data is incomplete due to resource constraints, and cannot be easily adapted.

The main goal of this chapter is to develop an adaptive sampling strategy, and an online process monitoring and diagnosis approach for incomplete and non-stationary HD streaming data, by incorporating its spatial and temporal information. To develop our adaptive sampling strategy, we propose to consider the streaming data as an incomplete tensor. We develop a RTR model that decomposes the streaming tensor into three components: a low-rank component, a sparse component, and a noise component. The low-rank component captures the spatio-temporal correlation of the data, which is estimated through tensor factorization [16, 17, 18], while the sparse component captures the variables suspicious of change. We use the information contained in the sparse component for monitoring and diagnosis. To define where to sample at each acquisition time, we propose a TSS scheme that balances exploration and exploitation by combining two probability functions. When the process is in-control, our sampling strategy behaves like random sampling, and when the process is out-of-control, it behaves like greedy sampling to locate where the changes have occurred.

The main contributions of this chapter are: (i) we develop an adaptive sequential sampling strategy for online monitoring of HD streaming data that outperforms existing methods by capturing their spatio-temporal correlation, (ii) we estimate the values of the unobserved variables at each acquisition time, and (iii) we develop a RD algorithm able to detect the abnormal variables quickly by incorporating the spatial correlation.

The rest of the chapter is organized as follows. In section 4.2, we provide a methodology overview. In section 4.3, we give a review on multilinear algebra and tensor completion algorithms. In section 4.4, we present the recursive tensor recovery model. In section 4.5 we present the adaptive sampling strategy, while, in section 4.6, we develop the monitoring and diagnosis methods. Simulation studies and two real data analysis are conducted in section 4.7 and section 4.8, respectively. Finally, we conclude in section 4.9.

4.2 Overview of the Proposed Methodology

4.2.1 Problem Formulation

Suppose there are $I \times J$ variables of interest distributed over a rectangular grid. Let $x_{i,j,t}$, $i = 1, \cdots, I$, $j = 1, \cdots, J$, $t = 1, 2, \cdots$, denote the measurement values of these variables at location (i, j), and at acquisition time t. They can be represented as an $I \times J$ matrix, denoted by $\boldsymbol{X}_t$. For example, each pixel of an image corresponds to an element of this matrix. We are interested in detecting a possible change in the components of $\boldsymbol{X}_t$. However, due to resource constraints, only q ($q \leq I \times J$) measurements can be observed at each acquisition time. Let $w_{i,j,t}$ be a binary variable, equal to 1 when $x_{i,j,t}$ is observed. Consequently, $\sum_{i=1}^{I} \sum_{j=1}^{J} w_{i,j,t} = q$, for all time t.

To develop an effective monitoring and diagnosis framework for partially observed data streams, we need to answer the following questions: (i) how to adaptively choose $w_{i,j,t}$ for each acquisition time t, (ii) how to use the available data to quickly detect a system change while maintaining a pre-specified false alarm rate and an in-control ARL, and (iii) how to determine the location(s) of the change or the set of variables responsible for the change. To answer these questions, we make four assumptions. (A1) The in-control data streams have a spatial structure that may gradually change over time, and (A2) can be represented as a low-rank tensor. These two assumptions are satisfied in several practical applications [66, 67]. (A3) At some finite time t_a, there are changes in the means of a small number of spatially correlated data streams. The change point t_a, the number of abnormal streams, and the magnitude and direction of the post-change means are unknown. The assumption of sparse changes is common when monitoring high-dimensional streaming data as an assignable cause likely impacts only a subset of sensors or small local regions. For example, in a steel rolling process, seam defects occur in small sections of the rolling bar [15]. As another example, in environmental monitoring, abrupt changes can be observed in specific regions when monitoring climate data [62]. (A4) The sampling strategy can be timely implemented without any cost within each sampling interval. This assumption is common in adaptive sampling strategies for online monitoring of sensing systems [60, 61, 62].

4.2.2 Methodology Overview

The general framework of the proposed methodology is shown in Figure 4.1. At each acquisition time, the sensor readings are organized into a tensor, and a RTR algorithm is proposed to estimate two key components from the partially observed data stream: a low-rank component, that captures the spatio-temporal structure of the data and contains estimates for the unobserved variables, and a sparse component, that captures the suspicious variables that may indicate an out-of-control state. To detect statistically significant changes using the sparse component, we construct an exponentially weighted moving average (EWMA) control chart. If an alarm is raised, we propose a RD algorithm to find the abnormal regions. Otherwise, based on the location and pattern of the suspicious observations in the sparse component, we decide where to sample next. Each step of the methodology will be elaborated in subsequent sections.

4.3 Multilinear Algebra and Tensor Completion

4.3.1 Tensor Notation and Multilinear Algebra

A tensor is a multidimensional array. A first-order tensor is a vector, a second-order tensor is a matrix, and tensors of order three or higher are called higher-order tensors. The order of a tensor is the number of dimensions, also known as ways or modes. Throughout the chapter, tensors of order $N \geq 3$ are denoted by Euler script letters, e.g., $\boldsymbol{\mathcal{X}}$, matrices are

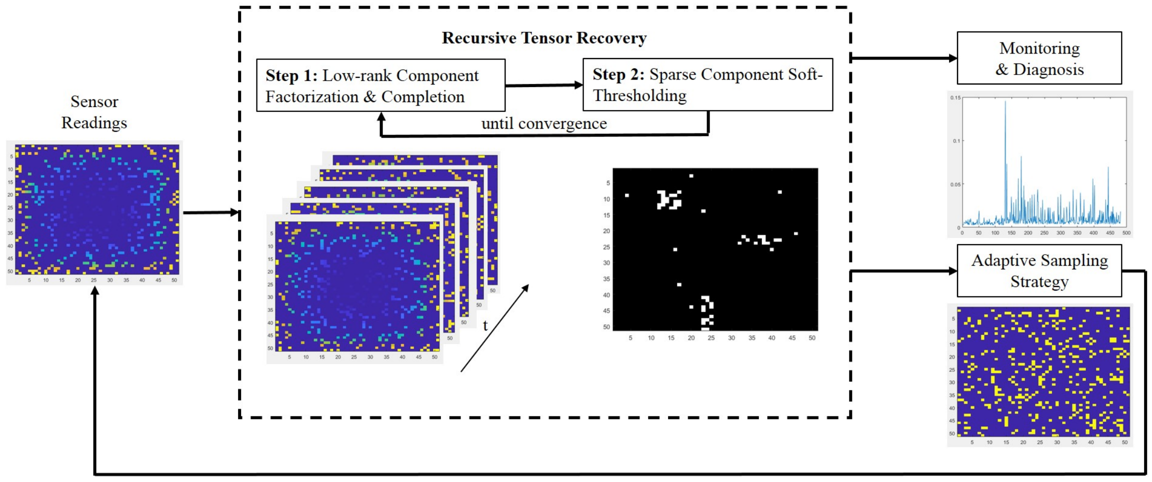

Recursive Tensor Recovery
Step 1: Low-rank Component Factorization & Completion
Step 2: Sparse Component Soft-Thresholding
until convergence
Sensor Readings
t
Monitoring & Diagnosis
Adaptive Sampling Strategy

Figure 4.1: Flow diagram of the proposed Tensor Sequential Sampling (TSS) methodology

denoted by boldface capital letters, e.g., $\boldsymbol{X}$, vectors are denoted by boldface lowercase letters, e.g., $\boldsymbol{x}$, and scalars are denoted by lower case letters, e.g., x. Entries of a matrix or a tensor are denoted by lower case letters with subscripts, e.g., the $(i_1, i_2, \cdots, i_N)$ entry of an N-way tensor $\mathcal{X}$ is denoted by $x_{i_1,i_2,\cdots,i_N}$.

Fibers are the higher-order analogue of matrix rows and columns. A fiber is defined by fixing every index but one. A matrix column is a mode-1 fiber and a matrix row is a mode-2 fiber. Third-order tensors have column, row, and tube fibers, denoted by $x_{:jk}$, $x_{i:k}$, and $x_{ij:}$, respectively.

An N-way tensor can be unfolded into a matrix, this is also known as matricization or flattening. The mode-n matricization of a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is denoted by $\boldsymbol{X}_{(n)}$ and arranges the mode-n one dimensional fibers to be the columns of the resulting matrix.

The Hadamard product of two tensors $\boldsymbol{\mathcal{X}}$, $\boldsymbol{\mathcal{Y}}$ of equal size, $I_1 \times I_2 \times \cdots \times I_N$, is denoted by $\boldsymbol{\mathcal{X}} * \boldsymbol{\mathcal{Y}}$ and defined as

$$(\boldsymbol{\mathcal{X}} * \boldsymbol{\mathcal{Y}})_{i_1,i_2,\cdots,i_N} = x_{i_1,i_2,\cdots,i_N} y_{i_1,i_2,\cdots,i_N}.$$

The inner product of $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{\mathcal{Y}}$ is the sum of the product of their entries,

$$\langle \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}} \rangle = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_N=1}^{I_N} x_{i_1,i_2,\cdots,i_N} y_{i_1,i_2,\cdots,i_N}.$$

The norm of tensor $\boldsymbol{\mathcal{X}}$ is defined as $||\boldsymbol{\mathcal{X}}|| = \sqrt{\langle \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{X}} \rangle}$. For matrices, $||\cdot||$ refers to the analogous Frobenius norm, and for vectors, $||\cdot||$ refers to the analogous L_2 norm.

The CANDECOMP/PARAFAC (CP) [68] tensor decomposition factorizes a tensor into a sum of rank-one tensors. For example, given a third-order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I \times J \times K}$, its CP-decomposition is $\boldsymbol{\mathcal{X}} \approx \sum_{r=1}^{R} \boldsymbol{a}_r \circ \boldsymbol{b}_r \circ \boldsymbol{c}_r$, where $\circ$ represents the vector outer product, R is a positive integer approximating the rank of the tensor, and $\boldsymbol{a}_r \in \mathbb{R}^I$, $\boldsymbol{b}_r \in \mathbb{R}^J$ and $\boldsymbol{c}_r \in \mathbb{R}^K$ for $r = 1 \cdots, R$. We define the factor matrices as the combination of the vectors from the rank-one components: $\boldsymbol{A} = [\boldsymbol{a}_1\ \boldsymbol{a}_2\ \cdots\ \boldsymbol{a}_R]$, $\boldsymbol{B} = [\boldsymbol{b}_1\ \boldsymbol{b}_2\ \cdots\ \boldsymbol{b}_R]$, and $\boldsymbol{C} = [\boldsymbol{c}_1\ \boldsymbol{c}_2\ \cdots\ \boldsymbol{c}_R]$, and write $\boldsymbol{\mathcal{X}} \approx [\![\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!]$. The rank of a tensor $\boldsymbol{\mathcal{X}}$ is defined as the smallest number of rank-one tensors sum of which is exactly equal to $\boldsymbol{\mathcal{X}}$. See [69] for a comprehensive overview of higher-order tensor decomposition and their applications.

4.3.2 Tensor Completion Methodologies

Several methods have been proposed to complete tensors with missing entries. They can be divided into three groups. The first one aims to minimize the estimated tensor's rank while requiring that the known entries remain unchanged. Since finding the rank of a tensor is an NP-hard problem [70], the tensor's rank is approximated by flattening the tensor and using matrix completion techniques [66, 67, 71]. The problem with this approach is that it does not preserve the multi-way structure of the data. The second group of tensor completion methods uses CP tensor decomposition to extract the underlying factors in each dimension of the tensor and to perform missing data imputation. Several optimization methods have been proposed to solve the decomposition problem, including expected minimization alternating least squares [16], second-order methods [17], and first-order methods

[18]. Although these methods are able to extract the latent factors and impute the missing data, while preserving the spatio-temporal structure, they are highly sensitive to outliers and anomalies in the data [72]. In an attempt to make tensor completion methods robust to corrupted data, robust low-rank tensor recovery methodologies have been developed [72, 73]. However, these methodologies have three main drawbacks: they rely on flattening the tensor to minimize the rank, hence losing the multi-way structure, they assume that the outliers and anomalies are fully observed, and they are mainly developed for offline diagnosis and are not scalable to online monitoring.

In the next section, using CP tensor decomposition, we develop a new recursive tensor recovery model that preserves the spatio-temporal structure of the streaming data, is robust to corrupted data, and can be used for online monitoring.

4.4 Recursive Tensor Recovery Model

As seen in the problem formulation section, the streaming data can be represented as a third-order tensor, $\boldsymbol{\mathcal{X}}$, with the first two modes representing spatial locations across the $I \times J$ grid, and the third mode representing time. Since our goal is online monitoring, we do not need to consider all historical data. Hence, we use a sliding window of size T, defined by the user. We discuss how to choose a value for T at the end of the section. Let the tensor $\boldsymbol{\mathcal{X}}^{(t)} \in \mathbb{R}^{I\times J\times T}$ represent the measurements $x_{i,j,k}$ for $i = 1, \cdots, I$, $j = 1, \cdots, J$, and $k = t - T + 1, \cdots, t$. Similarly, the indicator tensor for missing data is defined as $\boldsymbol{\mathcal{W}}^{(t)} \in \mathbb{R}^{I\times J\times T}$, where

$$w_{i,j,k} = \begin{cases} 1 & \text{if } x_{i,j,k} \text{ is observed} \\ 0 & \text{if } x_{i,j,k} \text{ is missing} \end{cases} .$$

We define $\boldsymbol{X}_k^{(t)}$ as the kth temporal slice (matrix) of tensor $\boldsymbol{\mathcal{X}}^{(t)}$, $k = 1, \cdots, T$, $\boldsymbol{x}_{i,:,k}^{(t)}$ as the ith row of this matrix, and $\boldsymbol{x}_{:,j,k}^{(t)}$ as the jth column.

One of the questions that we need to answer is how to quickly and accurately detect anomalies in the streaming data at each sampling time t, where anomalies are defined as mean changes. To answer this question, we start by decomposing the streaming tensor $\boldsymbol{\mathcal{X}}^{(t)}$, $t > T$, into three components, a low-rank component $\boldsymbol{\mathcal{L}}^{(t)}$, a sparse component $\boldsymbol{\mathcal{S}}^{(t)}$, and a noise component $\boldsymbol{\mathcal{E}}^{(t)}$, as $\boldsymbol{\mathcal{X}}^{(t)} = \boldsymbol{\mathcal{L}}^{(t)} + \boldsymbol{\mathcal{S}}^{(t)} + \boldsymbol{\mathcal{E}}^{(t)}$. The low-rank component, $\boldsymbol{\mathcal{L}}^{(t)}$, captures the spatio-temporal structure of the streaming data, while the sparse component, $\boldsymbol{\mathcal{S}}^{(t)}$, captures the variables suspicious of change. The entries of the noise component, $\boldsymbol{\mathcal{E}}^{(t)}$, are assumed to be uncorrelated with constant variance σ^2. To estimate $\boldsymbol{\mathcal{L}}^{(t)}$ and $\boldsymbol{\mathcal{S}}^{(t)}$, we propose the following model:

$$\begin{aligned} \min_{\boldsymbol{\mathcal{L}}^{(t)},\boldsymbol{\mathcal{S}}^{(t)}} \quad & \frac{1}{2}||\boldsymbol{\mathcal{W}}^{(t)} * \boldsymbol{\mathcal{E}}^{(t)}||^2 + \lambda \sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=t-T}^{t} w_{i,j,k}|s_{i,j,k}| \\ \text{s.t.} \quad & \boldsymbol{\mathcal{W}}^{(t)} * \boldsymbol{\mathcal{X}}^{(t)} = \boldsymbol{\mathcal{W}}^{(t)} * (\boldsymbol{\mathcal{L}}^{(t)} + \boldsymbol{\mathcal{S}}^{(t)} + \boldsymbol{\mathcal{E}}^{(t)}), \end{aligned} \tag{4.1}$$

where the first term in the objective function is a quadratic loss, the second term is an L_1 penalty term that encourages sparsity in $\boldsymbol{\mathcal{S}}^{(t)}$, and $\lambda > 0$ is a tuning parameter to be determined, its choice will be discusses at the end of the section.

Due to the high-dimensionality of the data, it would be impossible to directly solve this optimization problem. Therefore, we use the CP decomposition to capture the low-rank spatio-temporal structure of the streaming data. We set $\boldsymbol{\mathcal{L}}^{(t)} \approx [\![\boldsymbol{A}^{(t)}, \boldsymbol{B}^{(t)}, \boldsymbol{C}^{(t)}]\!]$, where $\boldsymbol{A}^{(t)} \in \mathbb{R}^{I\times R}, \boldsymbol{B}^{(t)} \in \mathbb{R}^{J\times R}$, and $\boldsymbol{C}^{(t)} \in \mathbb{R}^{T\times R}$ are the factor matrices of the low-rank component $\boldsymbol{\mathcal{L}}^{(t)}$, and R is the approximate rank of $\boldsymbol{\mathcal{L}}^{(t)}$. In practice, we estimate R using historical in-control data by trying different values and penalizing model complexity using a BIC criterion. By incorporating the CP decomposition into Equation 4.1, the optimization model can be rewritten as:

$$\begin{aligned} \min_{\boldsymbol{A}^{(t)},\boldsymbol{B}^{(t)},\boldsymbol{C}^{(t)},\boldsymbol{\mathcal{S}}^{(t)}} \quad & \frac{1}{2}||\boldsymbol{\mathcal{W}}^{(t)} * \boldsymbol{\mathcal{E}}^{(t)}||^2 + \lambda \sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=t-T}^{t} w_{i,j,k}|s_{i,j,k}| \\ \text{s.t.} \quad & \boldsymbol{\mathcal{W}}^{(t)} * \boldsymbol{\mathcal{X}}^{(t)} = \boldsymbol{\mathcal{W}}^{(t)} * \left([\![\boldsymbol{A}^{(t)}, \boldsymbol{B}^{(t)}, \boldsymbol{C}^{(t)}]\!] + \boldsymbol{\mathcal{S}}^{(t)} + \boldsymbol{\mathcal{E}}^{(t)}\right). \end{aligned} \tag{4.2}$$

By decomposing the tensor $\boldsymbol{\mathcal{X}}^{(t)}$ as the sum of a low-rank component, a sparse component, and a noise component, we guarantee that the CP decomposition of the low-rank component $\boldsymbol{\mathcal{L}}^{(t)}$ is not affected by the presence of anomalies in the streaming data. The factor matrices, $\boldsymbol{A}^{(t)}$, $\boldsymbol{B}^{(t)}$, and $\boldsymbol{C}^{(t)}$, estimate the missing entries of the low-rank tensor $\boldsymbol{\mathcal{L}}^{(t)}$. Additionally, the sparse component $\boldsymbol{\mathcal{S}}^{(t)}$ captures the variables suspicious of change at each point in time. This information is critical for the development of our adaptive sequential sampling strategy, as we demonstrate in section 4.5.

The problem presented in Equation 4.2 is a large-scale decomposable optimization problem that can be efficiently solved by alternating minimization as shown in algorithm 3. For each acquisition time $t > T$, in step 1, we fix $\boldsymbol{\mathcal{S}}^{(t)}$ and find the optimal factor matrices $\boldsymbol{A}^{(t)}, \boldsymbol{B}^{(t)}$, and $\boldsymbol{C}^{(t)}$. Then, in step 2, given that we have estimated the suspicious locations for previous acquisition times (i.e., $\boldsymbol{S}_k$ for $k < t$) and by fixing the low-rank component, $\boldsymbol{\mathcal{L}}^{(t)} = [\![\boldsymbol{A}^{(t)}, \boldsymbol{B}^{(t)}, \boldsymbol{C}^{(t)}]\!]$, we optimize $\boldsymbol{S}_T^{(t)}$. We iterate steps 1 and 2 until convergence, given a tolerance parameter ϵ. In our numerical experiments, we observed that the iterative process converges very fast (often in less than 5 iterations). The details of the optimization algorithm for each step are discussed in the following subsections.

4.4.1 Step 1: Recursive Tensor Factorization

Since the problem in Equation 4.2 is a large-scale decomposable optimization problem, we use alternating minimization to solve it. In step 1, we assume that $\boldsymbol{\mathcal{S}}^{(t)}$ is known and find the optimal factor matrices defining the low-rank component, $\boldsymbol{\mathcal{L}}^{(t)}$. Consequently, the problem reduces to:

$$\min_{\boldsymbol{A}^{(t)}, \boldsymbol{B}^{(t)}, \boldsymbol{C}^{(t)}} \quad \frac{1}{2} || \boldsymbol{\mathcal{W}}^{(t)} * (\boldsymbol{\mathcal{X}}^{(t)} - [\![\boldsymbol{A}^{(t)}, \boldsymbol{B}^{(t)}, \boldsymbol{C}^{(t)}]\!] - \boldsymbol{\mathcal{S}}^{(t)})||^2. \tag{4.3}$$

Equation 4.3 is a tensor factorization problem with missing data. To solve it, we use the CP-Weighted Optimization (CP-WOPT) algorithm [18]. This algorithm uses nonlinear conjugate gradient descent [74] to solve the optimization problem and is faster than other optimization methods available in the literature [16, 17].

Although the CP-WOPT is a rather fast algorithm, it may not scale to online monitoring applications, which require solving problem in Equation 4.3 for each acquisition time $t > T$. Therefore, we propose a new recursive CP weighted optimization algorithm that accelerates the tensor completion process. Recursive tensor factorization methods have been previously studied in the literature [75, 76]. These methods estimate the underlying factor matrices recursively so that the estimation error decreases at each iteration, and thus cannot be used for online monitoring. For monitoring streaming data, for each $t > T$, we need to find the factor matrices $\hat{\boldsymbol{A}}^{(t)}, \hat{\boldsymbol{B}}^{(t)}$, and $\hat{\boldsymbol{C}}^{(t)}$ that minimize the problem in Equation 4.3. For this reason, we develop a new recursive tensor factorization method, named Recursive CP-Weighted Optimization (RCP-WOPT).

At time t, we observe the new temporal slice (matrix) of the tensor, i.e. $\boldsymbol{X}_t = \boldsymbol{X}_T^{(t)}$. With the proposed decomposition method, we have that $\boldsymbol{X}_T^{(t)} = \boldsymbol{L}_T^{(t)} + \boldsymbol{S}_T^{(t)} + \boldsymbol{E}_T^{(t)}$. Assuming that $\boldsymbol{S}_T^{(t)}$ is known, we need to estimate $\boldsymbol{L}_T^{(t)}$. We have that $\boldsymbol{L}_T^{(t)} = \boldsymbol{A}^{(t)} diag(\boldsymbol{c}_{T,:}^{(t)})(\boldsymbol{B}^{(t)})^\top$, where $diag(\boldsymbol{c}_{T,:}^{(t)})$ is a square diagonal matrix with the elements of the vector $\boldsymbol{c}_{T,:}^{(t)}$ on the main diagonal. Since we are assuming that the spatial structure gradually changes over time (Assumption (A1)), we start by estimating $\boldsymbol{c}_{T,:}^{(t)}$, using $\hat{\boldsymbol{A}}^{(t-1)}$ and $\hat{\boldsymbol{B}}^{(t-1)}$. For this purpose, we solve the following optimization problem:

$$\min_{\boldsymbol{c}_{T,:}^{(t)} \in \mathbb{R}^R} \frac{1}{2} \sum_{i=1}^{I} \sum_{j=1}^{J} \left(w_{i,j,t} \left(x_{i,j,t} - \left(\boldsymbol{a}_{i,:}^{(t-1)} * \boldsymbol{b}_{j,:}^{(t-1)} \right)^\top \boldsymbol{c}_{T,:}^{(t)} - s_{i,j,t} \right) \right)^2 . \tag{4.4}$$

This problem is derived from Equation 4.3, by considering only the last temporal slice of the tensors. It is easy to see that this problem has a closed-form solution given by

$$\begin{aligned} \hat{\boldsymbol{c}}_{T,:}^{(t)} &= \left(\sum_{i=1}^{I} \sum_{j=1}^{J} w_{i,j,t} \left(\boldsymbol{a}_{i,:}^{(t-1)} * \boldsymbol{b}_{j,:}^{(t-1)} \right) \left(\boldsymbol{a}_{i,:}^{(t-1)} * \boldsymbol{b}_{j,:}^{(t-1)} \right)^\top \right)^{-1} \\ &\quad \times \left(\sum_{i=1}^{I} \sum_{j=1}^{J} w_{i,j,t} (x_{i,j,t} - s_{i,j,t}) \left(\boldsymbol{a}_{i,:}^{(t-1)} * \boldsymbol{b}_{j,:}^{(t-1)} \right) \right) . \end{aligned} \tag{4.5}$$

With an initial estimate for $\hat{\boldsymbol{c}}_{T,:}^{(t)}$, we proceed to run CP-WOPT with a warm start, using the matrices $\hat{\boldsymbol{A}}^{(t-1)}$, $\hat{\boldsymbol{B}}^{(t-1)}$, and $\hat{\boldsymbol{C}}^{(t)}$, where $\hat{\boldsymbol{c}}_{k,:}^{(t)} = \hat{\boldsymbol{c}}_{k+1,:}^{(t-1)}$, for $k = 1, \cdots, T-1$, and $\hat{\boldsymbol{c}}_{T,:}^{(t)}$ was estimated using Equation 4.5. With the warm start the CP-WOPT converges in very few iterations, due to the spatial structure that gradually changes over time. The RCP-WOPT algorithm is summarized in algorithm 2.

Algorithm 2: RCP-WOPT Algorithm

```
For a fixed $t > T$
**Input:** $\boldsymbol{\mathcal{W}}^{(t)}, \boldsymbol{\mathcal{X}}^{(t)}, \boldsymbol{\mathcal{S}}^{(t)}, \hat{\boldsymbol{A}}^{(t-1)}, \hat{\boldsymbol{B}}^{(t-1)}, \hat{\boldsymbol{C}}^{(t-1)}$
**Step 1:** Set $\hat{\boldsymbol{c}}_{k,:}^{(t)} = \hat{\boldsymbol{c}}_{k+1,:}^{(t-1)}$ for $k = 1, \cdots, T-1$ and compute $\hat{\boldsymbol{c}}_{T,:}^{(t)}$ using Equation 4.5
**Step 2:** Run CP-WOPT with $\hat{\boldsymbol{A}}^{(t-1)}, \hat{\boldsymbol{B}}^{(t-1)}$ and $\hat{\boldsymbol{C}}^{(t)}$ as a warm start
**Output:** $\hat{\boldsymbol{\mathcal{L}}}^{(t)} = [\![\hat{\boldsymbol{A}}^{(t)}, \hat{\boldsymbol{B}}^{(t)}, \hat{\boldsymbol{C}}^{(t)}]\!]$
```

4.4.2 Step 2: Soft-Thresholding

In this step, we try to estimate the location of the suspicious variables given the updated low-rank structure. As these locations have already been estimated for previous acquisition times, i.e., $\boldsymbol{S}_k$, for $k < t$, we only need to estimate $\boldsymbol{S}_t = \boldsymbol{S}_T^{(t)}$, given $\hat{\boldsymbol{\mathcal{L}}}^{(t)} = [\![\hat{\boldsymbol{A}}^{(t)}, \hat{\boldsymbol{B}}^{(t)}, \hat{\boldsymbol{C}}^{(t)}]\!]$, obtained in Step 1. Therefore, the problem presented in Equation 4.2 reduces to:

$$\begin{aligned} \min_{\boldsymbol{s_{i,j,t}}} \quad & \frac{1}{2}\sum_{i=1}^{I}\sum_{j=1}^{J} w_{i,j,t} e_{i,j,t}^2 + \lambda \sum_{i=1}^{I}\sum_{j=1}^{J} w_{i,j,t}|s_{i,j,t}| \\ \text{s.t.} \quad & w_{i,j,t}x_{i,j,t} = w_{i,j,t}\left(\sum_{r=1}^{R} \hat{a}_{ir}^{(t)}\hat{b}_{jr}^{(t)}\hat{c}_{Tr}^{(t)} + s_{i,j,t} + e_{i,j,t}\right), \end{aligned} \tag{4.6}$$

for all $i = 1, \cdots, I$ and $j = 1, \cdots, J$. The solution to this optimization problem has a closed-form computed by the soft-thresholding function as shown in Proposition 3.

Proposition 3. *Given $\boldsymbol{S}_k$, for $k < t$, and $\hat{\boldsymbol{\mathcal{L}}}^{(t)} = [\![\hat{\boldsymbol{A}}^{(t)}, \hat{\boldsymbol{B}}^{(t)}, \hat{\boldsymbol{C}}^{(t)}]\!]$, the solution to the* *problem presented in Equation 4.6 is obtained by the soft-thresholding function:*

$$\hat{s}_{i,j,t} = S_\lambda \left(w_{i,j,t} \left(x_{i,j,t} - \sum_{r=1}^{R} \hat{a}_{ir}^{(t)} \hat{b}_{jr}^{(t)} \hat{c}_{Tr}^{(t)} \right) \right), \tag{4.7}$$

where $S_\lambda(\cdot) = sgn(\cdot)\,(|\cdot| - \lambda)_+$, $sgn(\cdot)$ *is the sign function, and* $(\cdot)_+ = \max(\cdot, 0)$. *(Proof in Appendix, section C.1)*

The soft-thresholding function translates the value of $w_{i,j,t}\left(x_{i,j,t} - \sum_{r=1}^{R} \hat{a}_{ir}^{(t)} \hat{b}_{jr}^{(t)} \hat{c}_{Tr}^{(t)}\right)$ toward zero by the amount λ, and sets it to zero if $\left|w_{i,j,t}\left(x_{i,j,t} - \sum_{r=1}^{R} \hat{a}_{ir}^{(t)} \hat{b}_{jr}^{(t)} \hat{c}_{Tr}^{(t)}\right)\right| \leq \lambda$. Therefore, if $\hat{s}_{i,j,t} \neq 0$, we conclude that the variable corresponding to the location (i, j) has a suspicious behaviour at time t, not captured by the low-rank component, and that this location is a feasible candidate for sampling.

The proposed RTR methodology, summarized as algorithm 3, has two main advantages: (i) RTR is robust to abnormal changes in the data as it decomposes the streaming tensor into a low-rank component, $\boldsymbol{\mathcal{L}}^{(t)}$, and a sparse component, $\boldsymbol{\mathcal{S}}^{(t)}$, and performs the CP decomposition on only the low-rank component; and (ii) if the system is in-control, at each acquisition time, RTR is able to recover the value of the unobserved variables in the system, while other sequential sampling methods [60, 61, 62] are not capable of missing data imputation.

One question that remains to be answered is how to choose the window size T and the regularization parameter λ for the RTR algorithm. Next, we discuss how to set values for these two parameters.

The window size T cannot be too large or too small. If T is too large, running the RTR algorithm is computationally too expensive for online monitoring and diagnosis. On the other hand, if T is too small, we loose information on the temporal structure of the data and the detection capability of the proposed method worsens. In practice, the value for T can be determined with in-control historical data by fixing a desired Root Mean Square Error (RMSE).

Algorithm 3: Recursive Tensor Recovery (RTR) Algorithm

```
For a fixed $t > T$ and a tolerance parameter $\epsilon > 0$,
**Input:** $\boldsymbol{\mathcal{W}}^{(t)}, \boldsymbol{\mathcal{X}}^{(t)}, \hat{\boldsymbol{A}}^{(t-1)}, \hat{\boldsymbol{B}}^{(t-1)}, \hat{\boldsymbol{C}}^{(t-1)}, \hat{\boldsymbol{\mathcal{S}}}^{(t-1)}$
Let $\iota = 0$
Estimate $\hat{\boldsymbol{c}}_{T,:}^{(t)}$ (subsection 4.4.1, Equation 4.5)
Set $\hat{\boldsymbol{\mathcal{L}}}^{(t,\iota)} = [\![\hat{\boldsymbol{A}}^{(t-1)}, \hat{\boldsymbol{B}}^{(t-1)}, \hat{\boldsymbol{C}}^{(t,\iota)}]\!]$ where $\hat{\boldsymbol{c}}_{k,:}^{(t)} = \hat{\boldsymbol{c}}_{k+1,:}^{(t-1)}$ for $k = 1, \cdots, T-1$
Set $\hat{\boldsymbol{S}}_k^{(t,\iota)} = \hat{\boldsymbol{S}}_{k+1}^{(t-1)}$ for $k = 1, \cdots, T-1$ and $\hat{\boldsymbol{S}}_T^{(t,\iota)} = \boldsymbol{0}$
**repeat**
    $\iota = \iota + 1$
    **Step 1:** Solve the recursive tensor factorization problem (subsection 4.4.1) to
      get $\hat{\boldsymbol{\mathcal{L}}}^{(t,\iota)} = [\![\hat{\boldsymbol{A}}^{(t,\iota)}, \hat{\boldsymbol{B}}^{(t,\iota)}, \hat{\boldsymbol{C}}^{(t,\iota)}]\!]$
    **Step 2:** Solve the soft-thresholding problem (subsection 4.4.2) to estimate
      $\hat{\boldsymbol{S}}_T^{(t,\iota)}$
**until** *convergence, i.e.* $||\hat{\boldsymbol{\mathcal{L}}}^{(t,\iota)} + \hat{\boldsymbol{\mathcal{S}}}^{(t,\iota)} - \hat{\boldsymbol{\mathcal{L}}}^{(t,\iota-1)} - \hat{\boldsymbol{\mathcal{S}}}^{(t,\iota-1)}||^2 \le \epsilon$;
**Output:** $\hat{\boldsymbol{\mathcal{L}}}^{(t,\iota)} = [\![\hat{\boldsymbol{A}}^{(t,\iota)}, \hat{\boldsymbol{B}}^{(t,\iota)}, \hat{\boldsymbol{C}}^{(t,\iota)}]\!], \hat{\boldsymbol{\mathcal{S}}}^{(t,\iota)}$
```

Under the assumption that the entries of the noise component are independent normal random variables with mean zero and variance σ^2, a reasonable choice for the regularization parameter is $\lambda = \hat{\sigma}\sqrt{2\log(q)}$, where q is the number of variables observed, and $\hat{\sigma}$ is estimated using in-control data. This is because the expected value of the maximum of n independent and identically distributed normal random variables with mean zero and variance σ^2 cannot exceed $\sigma\sqrt{2\log(n)}$, when n is large [77]. However, this threshold may not be appropriate for the mean shifts smaller than $\sigma\sqrt{2\log(q)}$. For such cases, following the recommendations in [78], we could use the BIC criterion to tune the penalty parameter at each iteration, denoted by λ_{BIC}. If there is no domain knowledge on the magnitude of the shift, we suggest using the following regularization parameter:

$$\lambda^* = \min(\lambda_{BIC}, \hat{\sigma}\sqrt{2\log(q)}). \tag{4.8}$$

Finally, it is important to highlight the differences between the proposed RTR algorithm and the ST-SSD method [15] which decomposes the data into a functional mean component, an abnormal component, and a noise component, and performs online monitoring and diagnosis of non-stationary streaming data. First, ST-SSD assumes that the data has a smooth spatial structure and uses pre-specified spatial and temporal bases to reduce the dimensionality of the components to be estimated. The definition of the bases relies on domain knowledge about the process and the anomalies. In contrast, RTR uses the CP decomposition to estimate the low-rank component of the data which is a data-driven approach and does not need to assume a smooth spatial structure. Second, ST-SSD was not designed for the case where there is missing data due to resource constraints. On the one hand, it cannot perform missing data imputation, and on the other the assumptions on the monitoring test statistic will not hold.

4.5 Adaptive Sampling Strategy

In this section, we define an adaptive sampling strategy that balances between exploring the space and sampling around the locations that are suspicious of change. At each acquisition time $t > T$, we sample q out of the $I \times J$ variables, where q depends on the resource constraints, from the following probability mass function:

$$p_t(i,j) = \beta r_t(i,j) + (1-\beta) g_t(i,j); i = 1, \cdots, I; j = 1, \cdots, J, \tag{4.9}$$

where r_t is a probability mass function that encourages exploration by random sampling over the space, g_t is a probability mass function that encourages greedy sampling around suspicious areas of the grid, and $\beta \in (0,1)$ is a tuning parameter that balances the exploration and exploitation. With the proposed adaptive sampling strategy, it is possible to sample a variable multiple times. If this is the case, we continue sampling until we select q different variables. The q chosen variables are observed at time t and used for online monitoring and diagnosis. The exploration probability mass function is defined as:

$$r_t(i,j) = \frac{1}{Z}\left(1 - \frac{1}{t-1}\sum_{k=1}^{t-1} w_{i,j,k}\right), \tag{4.10}$$

where $\sum_{k=1}^{t-1} w_{i,j,k}$ is the number of times a variable has been observed in the past and Z is a normalizing constant. According to this mass function, the more frequently a variable has been visited, the smaller the chances that the variable will be visited again. Hence, it encourages exploration over the grid. The exploitation probability mass function, g_t, is defined by fitting an empirical kernel density function over the sparse component, $\hat{\boldsymbol{S}}_{t-1}$, obtained by the RTR algorithm. We define the set of all variables as $N = \{(i,j)|\ i = 1,\cdots,I,\ j = 1,\cdots,J\}$ and the set of suspicious variables, at time $t-1$, as $U_{t-1} = \{(i,j)|\ \hat{s}_{i,j,t-1} \neq 0,\ i = 1,\cdots,I,\ j = 1,\cdots,J\}$. For all $\boldsymbol{n} \in N$, the empirica kernel density function is computed as:

$$g_t(\boldsymbol{n}) = \frac{1}{\#(U_{t-1})} \sum_{i=1}^{\#(U_{t-1})} K_\theta(\boldsymbol{n} - \boldsymbol{u}_i), \tag{4.11}$$

where $\#(\cdot)$ is the cardinality of a set, $K_\theta(\cdot)$ is the kernel density, θ is a bandwidth parameter, and $\boldsymbol{u}_i \in U_{t-1}$, for $i = 1,\cdots,\#(U_{t-1})$. In practice, any kernel can be used, and its choice may depend on the domain knowledge about the anomalies. In this chapter, we use a Gaussian kernel with an optimal bandwidth obtained by using the L-BFGS-B algorithm [79]. If there are no suspicious variables, i.e. $\hat{\boldsymbol{S}}_{t-1} = \boldsymbol{0}$, we set g_t as a uniform mass function. When the system is out-of-control, sampling according to this mass function guarantees a fast discovery of the defective regions, as it greedily samples around the suspicious locations.

The proposed sequential sampling strategy ensures the fast detection of a wide range of possible shifts, thanks to the exploration probability mass function (Proposition 4), and identifies the shifted variables or anomalous regions, thanks to the exploitation probability mass function (Proposition 5).

Proposition 4. *Let V_1 denote the set of variables $\boldsymbol{n} \in N$ which will never be observed after a finite time t_0. If the proposed sequential sampling procedure is followed, with probability 1 we observe all the variables after a finite time t_0. That is, $P(V_1 = \emptyset) = 1$, where $\emptyset$ represents the empty set. (Proof in Appendix, section C.2)*

Observing all the variables after a finite time is an important property as it ensures that if a change appears at a particular location, it will not be missed due to incomplete sampling, and it will be detected after a finite number of sampling iterations. Note that the effectiveness of the sampling strategy is highly dependent on the number of available sensors, q. As q increases, the detection capability of the sampling strategy also increases.

Proposition 5. *Assume that the entries of the noise component are normally and independently distributed with mean zero and variance σ^2. Let V_2 denote the set of variables $\boldsymbol{n} \in N$, which have a mean shift at time t_a. If the mean shift is larger than $\sigma\sqrt{2\log(q)}$, for any finite time $t \geq t_a$, once a variable $\boldsymbol{v}_2 \in V_2$ is observed, there is a non-zero probability that the variable will be observed at each time after t. (Proof in Appendix, section C.3)*

Proposition 5 indicates that, when the system is out-of-control, once an abnormal variable is observed (which is surely observed according to Proposition 4), it will be observed indefinitely with probability greater than zero. The foregoing propositions guarantee that if $q \geq \#(V_2)$, all variables in V_2 will eventually be observed, and an accurate diagnosis can be achieved.

It remains to discuss how to set a value for the parameter β. If β is close to one, the sampling algorithm will favor the exploration of the space, and, if it is close to zero, it will exploit the information contained in the sparse component. If there is no domain knowledge on the number of clustered anomalies or on the magnitude of their change, we recommend choosing $\beta = 0.5$, giving equal weight to the exploration and exploitation probability mass functions. If it is possible to observe more than one clustered anomaly, choosing $\beta \geq 0.5$ is recommended. Even if an abnormal cluster is detected, exploring the space is needed to find the remaining abnormal areas. Furthermore, domain knowledge on the magnitude of the change can influence the choice of β. For small changes (i.e., a SNR smaller than 3) we recommend giving more weight to the exploitation probability mass function, and choosing $\beta \leq 0.5$.

4.6 Online Monitoring and Diagnosis

In this section, we propose a monitoring procedure that integrates the proposed RTR method with an EWMA control chart. We also discuss how to perform diagnosis after a change is detected by the monitoring scheme.

4.6.1 Monitoring Statistic and Stopping Time

The estimated sparse component, $\hat{\boldsymbol{S}}_t$, at time $t > T$, contains information about the potential changes in the system. If the system is out-of-control, the elements of $\hat{\mathcal{S}}_t$ are expected to be large. Therefore, to detect a change, we propose to monitor the sum of the absolute values of its entries, denoted as $y_t = \sum_{i=1}^{I} \sum_{j=1}^{J} |\hat{s}_{i,j,t}|$, using an EWMA control chart. The EWMA monitoring statistic, for all $t > T + 1$, is given by

$$z_t = \gamma y_t + (1 - \gamma) z_{t-1}, \tag{4.12}$$

where $\gamma \in (0, 1)$ is the weight parameter that controls the importance of historical data, and $z_{T+1} = y_{T+1}$. In the simulations and case studies, we use $\gamma = 0.2$. The stopping time of the monitoring procedure can be determined by:

$$T_{TSS}(h) = \inf\{t > T |\ z_t \geq h\}, \tag{4.13}$$

where h is the control limit estimated based on the empirical in-control distribution of z_t. The practitioner can determine the h value from a sufficiently large in-control data or via Monte Carlo Simulation or bootstrapping [50]. The value of h is related to the pre-specified desired in-control ARL of the monitoring scheme, when no change occurs in the system. In the Appendix, section C.4, a simulation procedure for obtaining the h value is given.

4.6.2 Diagnosis of the Detected Changes

After the proposed monitoring approach triggers an alarm, the next step is to diagnose the detected change and find the variables (regions) affected by the change. Our diagnosis method provides an advantage over existing adaptive monitoring methodologies, which lack diagnosability.

Suppose that the control chart triggers an alarm at time τ. A naïve approach for diagnosis is to identify the location (i, j) that has a non-zero $\hat{s}_{i,j,\tau}$ value as an out-of-control variable. However, this approach ignores the fact that there might be unobserved out-of-control variables because of the sparse sampling. Consequently, it would not be able to detect such variables. In contrast, our diagnosis method relies on multiple samples that are selected according to the proposed adaptive sampling strategy.

At each acquisition time after the alarm is raised, i.e., $t \geq \tau$, we use the average of the entries of the sparse component, $\bar{s}_{i,j,t} = \dfrac{1}{t-\tau}\sum_{k=\tau}^{t} \hat{s}_{i,j,t}$, to identify the anomalous variables or regions. Furthermore, since there is a spatial correlation between the variables, to have a smooth estimate of the anomalous regions, we convolve a smoothing filter with $\bar{\boldsymbol{S}}_t$. For this purpose, different filters including the mean filter, the weighted average filter, and the Gaussian filter can be used. The resulting smoothed estimate of the anomaly, denoted by $\tilde{\boldsymbol{S}}_t$, is further denoised by thresholding. That is,

$$\tilde{\tilde{s}}_{i,j,t} = S_{\lambda_D}(\tilde{s}_{i,j,t}), \tag{4.14}$$

for all $i = 1, \cdots, I$ and $j = 1, \cdots, J$, where the threshold, λ_D, is a tuning parameter. In practice, we use the BIC criterion to select the optimal value of λ_D at each iteration. We conclude that the location (i, j) is abnormal if $\tilde{\tilde{s}}_{i,j,t}$ is non-zero. When the set of abnormal variables does not change significantly from one acquisition time to the next, we stop the diagnosis procedure, which is summarized in algorithm 4.

```
Algorithm 4: Recursive Diagnosis (RD) Algorithm
  For a fixed $t \geq \tau$
  Input: $\hat{\boldsymbol{S}}_k$, $k = \tau, \cdots, t$
  repeat
      Step 1: Compute the average of the abnormal component $\bar{\boldsymbol{S}}_t$
      Step 2: Use a filter to smooth the average sparse component and get $\tilde{\boldsymbol{S}}_t$
      Step 3: Use soft-thresholding to estimate $\tilde{\tilde{\boldsymbol{S}}}_t$
  until the set of abnormal variables does not change significantly;
  Output: Set with abnormal variables, at time $t$,
    $ABN_t = \{(i,j) | \, \tilde{\tilde{s}}_{i,j,t} \neq 0, \ i = 1, \cdots, I, \ j = 1, \cdots, J\}$
```

At this point, we have discussed all the steps in our proposed adaptive sampling strategy for online monitoring and diagnosis of HD streaming data. algorithm 5 summarizes the whole proposed TSS methodology. At each acquisition step, we use our adaptive sampling strategy to decide where to sample, then we run the RTR algorithm to estimate the low-rank and the sparse components, and use an EWMA control-chart for monitoring. Once an alarm is raised, we use the RD algorithm for diagnosis. It is important to note that since all of the algorithms are recursive (i.e., they exploit the estimations of previous acquisition times), our proposed methodology is efficient in terms of computational time and memory usage. Next, we evaluate the performance of the TSS algorithm with simulation experiments and two case studies.

```
Algorithm 5: Tensor Sequential Sampling (TSS) Algorithm
  For a fixed $t > T$
  Input: $\boldsymbol{\mathcal{W}}^{(t-1)}, \hat{\boldsymbol{\mathcal{L}}}^{(t-1)} = [\![\hat{\boldsymbol{A}}^{(t-1)}, \hat{\boldsymbol{B}}^{(t-1)}, \hat{\boldsymbol{C}}^{(t-1)}]\!], \hat{\boldsymbol{\mathcal{S}}}^{(t-1)}$
  Step 1: Estimate the exploration probability mass function $r_t$
  Step 2: Estimate the exploitation probability mass function $g_t$
  Step 3: Select the $q$ variables to be observed, by sampling from the probability
    mass function $p_t$
  Step 4: Run the RTR algorithm to estimate $\hat{\boldsymbol{\mathcal{X}}}^{(t)}$, $\hat{\boldsymbol{\mathcal{L}}}^{(t)} = [\![\hat{\boldsymbol{A}}^{(t)}, \hat{\boldsymbol{B}}^{(t)}, \hat{\boldsymbol{C}}^{(t)}]\!]$ and $\hat{\boldsymbol{\mathcal{S}}}^{(t)}$
  Step 5: Compute the monitoring statistic $z_t$. If $z_t > h$ raise an alarm and proceed
    to the RD algorithm. Otherwise $t = t + 1$.
  Output: $\boldsymbol{\mathcal{W}}^{(t)}$ and $\hat{\boldsymbol{\mathcal{X}}}^{(t)} = [\![\hat{\boldsymbol{A}}^{(t)}, \hat{\boldsymbol{B}}^{(t)}, \hat{\boldsymbol{C}}^{(t)}]\!] + \hat{\boldsymbol{\mathcal{S}}}^{(t)}$
```

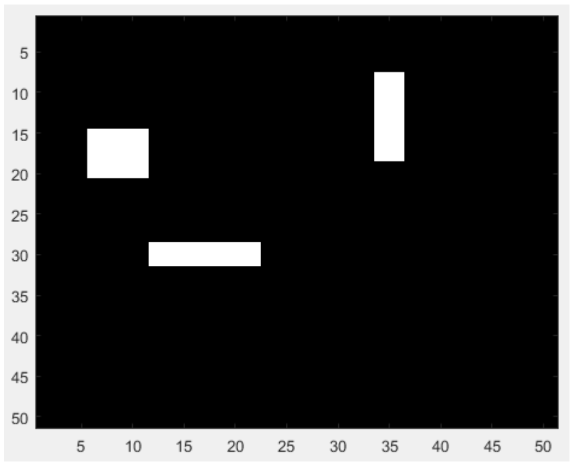
4.7 Performance Evaluation via Simulations

In this section, we evaluate the performance of the proposed methodology using simulations. We generate a stream of images, assuming that not all pixel data is accessible at each epoch due to data transmission constraints. To simulate a functional mean with a spatial structure that gradually changes over time, we generate data from a heat transfer process. The functional mean $H(x, y, t)$ is generated according to the following heat transfer equation [80]:

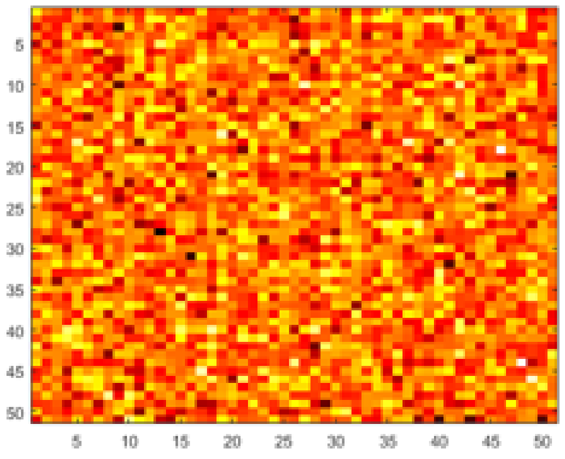
$$\frac{\partial H}{\partial t} = \alpha \left(\frac{\partial^2 H}{\partial x^2} + \frac{\partial^2 H}{\partial y^2} \right) \tag{4.15}$$

where $x, y \in [0, 0.05]$ denote pixel locations in the image, $\alpha = 5 \times 10^{-5}$ is a thermal diffusivity coefficient, describing how fast the material can conduct thermal energy, and t is the time frame. The initial condition is set as $H|_{t=0} = 0$ and the boundary conditions are set as $H|_{x=0} = H|_{y=0} = H|_{x=0.05} = H|_{y=0.05} = 1$. At each time t, the functional mean $H(x, y, t)$ is recorded at locations $x = \frac{i}{I+1}, y = \frac{j}{I+1}$, $i, j = 1, \cdots, I$, which results in an $I \times I$ matrix denoted by $\boldsymbol{H}_t$. For the simulation study we set $I = 51$ and $t = 1, \cdots, 401$, which leads to 401 images of size 51×51, that can be represented as a $51 \times 51 \times 401$ tensor. Independent and identically distributed normal random noises with standard deviation $\sigma = 0.01$ are added to each pixel. To generate out-of-control streams, for $t \geq 50$, we randomly generate three clustered anomalous regions, two 10×2 rectangles and one 5×5 square, by adding a constant value δ whose intensity is defined by a pre-specified Signal-to-Noise Ratio (SNR). Therefore, 2.5% of the pixels have an abnormal behaviour, for $t \geq 50$. An example of the simulated anomalies, the random noise, and the functional mean is shown in Figure 4.2.

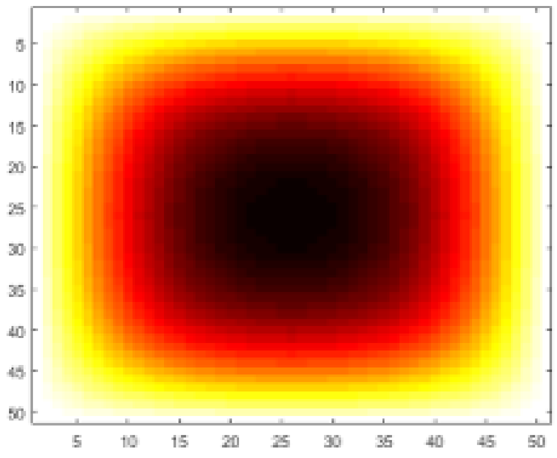
We begin by evaluating the effectiveness of the TSS algorithm in imputing missing data when there are no abnormal regions and no noise. For this purpose, we generate a $51 \times 51 \times 401$ tensor using only the heat transfer equation (Equation 4.15). Using the TSS algorithm, we complete the streaming tensor ($51 \times 51 \times T$, where T is the window



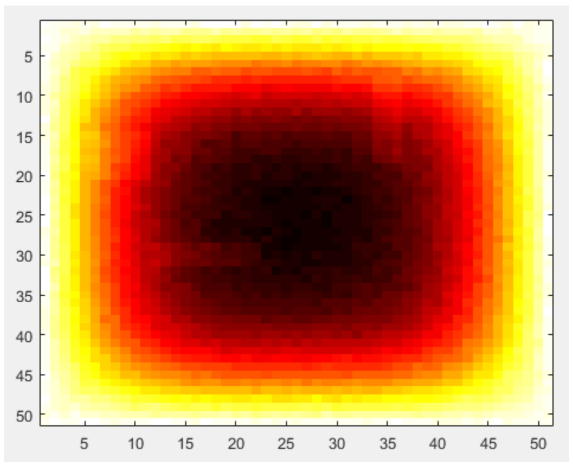
(a) Clustered anomalies



(b) Normal i.i.d. noise

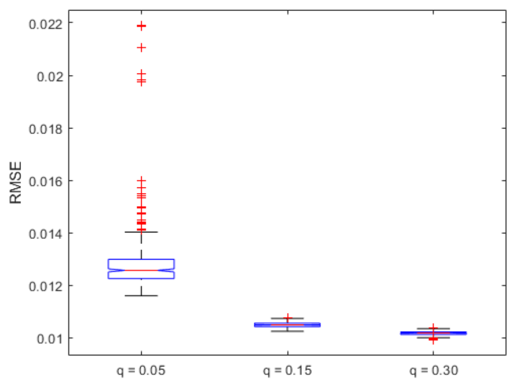


(c) Functional mean

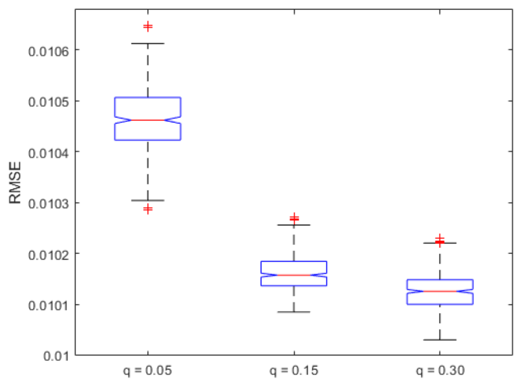


(d) Simulated image

Figure 4.2: Simulated image with functional mean, clustered anomalies and noise at time $t = 50$



(a) $T = 5$



(b) $T = 20$

Figure 4.3: RMSE distribution for in-control data

size) at each acquisition time and compute the RMSE. The results are presented for two window sizes, $T = 5$ and $T = 20$, and three sampling percentages, $q = 5\%$, $q = 15\%$, $q = 30\%$, in Figure 4.3. As expected, we observe that a larger tensor window and larger sampling percentages have smaller completion errors. The fact that the TSS algorithm is able to estimate the value of unobserved pixels, reasonably well, is a considerable advantage over the existing sequential sampling monitoring methods, which are not capable of data imputation.

Next, we study the effect of the parameters on the monitoring performance of the TSS algorithm. The window size of the streaming tensor (T), the percentage of pixels observed in each image (q), and the parameter β, balancing exploration and exploitation in the proposed sampling scheme, can impact the performance of the TSS algorithm. Specifically, we consider two window sizes, $T = 5$, $T = 20$, three sampling percentages, $q = 5\%$, $q = 15\%$, $q = 30\%$, and three balancing parameters, $\beta = 0.2$, $\beta = 0.5$, $\beta = 0.8$. We fix the in-control ARL_0 for all scenarios to be 200 and compare the out-of-control ARL_1 under different SNRs. The out-of-control $ARLs$ obtained from 500 simulation replicates are shown in Table 4.1. As expected, in all scenarios, as the SNR increases, the anomalies are easier to detect, i.e., smaller ARL_1 values. In terms of the parameters T, q, and β, we observe the following:

Table 4.1: Sensitivity analysis results for ARL_1, for different parameter combinations over 500 simulation replicates. Results are in the form of mean(standard deviation).

		β = 0.8			β = 0.5			β = 0.2		
		q = 0.05	q = 0.15	q = 0.30	q = 0.05	q = 0.15	q = 0.30	q = 0.05	q = 0.15	q = 0.30
T=5	SNR = 1.5	141.42(0.25)	171.40(0.24)	160.41(0.24)	181.38(0.26)	152.78(0.22)	148.90(0.22)	152.02(0.25)	157.88(0.22)	152.31(0.23)
	SNR = 2.0	111.39(0.20)	111.73(0.20)	102.56(0.19)	118.29(0.23)	94.09(0.18)	91.88(0.17)	117.79(0.22)	93.46(0.17)	89.60(0.16)
	SNR = 2.5	66.59(0.15)	56.82(0.13)	50.00(0.10)	69.18(0.16)	47.00(0.09)	46.84(0.10)	67.37(0.15)	50.23(0.11)	44.16(0.10)
	SNR = 3.0	32.19(0.08)	21.03(0.04)	25.48(0.05)	32.67(0.08)	20.91(0.06)	22.15(0.05)	34.87(0.09)	26.48(0.06)	22.01(0.05)
	SNR = 3.5	13.08(0.03)	10.01(0.02)	12.33(0.02)	16.80(0.05)	9.27(0.02)	13.17(0.03)	17.65(0.05)	10.98(0.03)	12.63(0.03)
	SNR = 4.0	5.59(0.01)	4.80(0.01)	8.63(0.02)	8.33(0.02)	4.76(0.01)	8.93(0.02)	10.94(0.03)	5.29(0.01)	9.15(0.02)
	SNR = 4.5	3.22(0.00)	2.51(0.00)	4.89(0.01)	3.98(0.01)	3.03(0.01)	5.42(0.01)	5.45(0.01)	3.12(0.01)	4.71(0.01)
	SNR = 5.0	2.08(0.00)	1.38(0.00)	2.00(0.00)	2.70(0.00)	1.46(0.00)	2.08(0.00)	3.64(0.01)	1.76(0.00)	2.56(0.00)
T=20	SNR = 1.5	169.06(0.27)	130.06(0.23)	138.39(0.23)	166.98(0.26)	164.48(0.24)	156.22(0.24)	195.03(0.27)	155.30(0.24)	135.93(0.23)
	SNR = 2.0	122.88(0.24)	73.37(0.16)	85.65(0.18)	114.32(0.23)	92.44(0.19)	79.59(0.17)	121.27(0.23)	87.46(0.18)	77.91(0.17)
	SNR = 2.5	46.92(0.14)	29.17(0.09)	31.71(0.10)	44.95(0.13)	44.61(0.12)	35.30(0.09)	65.05(0.18)	36.11(0.10)	35.52(0.10)
	SNR = 3.0	17.08(0.06)	11.37(0.03)	15.80(0.04)	16.87(0.07)	17.80(0.05)	16.96(0.05)	23.94(0.08)	15.84(0.04)	16.12(0.04)
	SNR = 3.5	4.57(0.02)	4.35(0.01)	9.09(0.02)	5.97(0.03)	4.91(0.01)	9.19(0.02)	8.85(0.04)	6.83(0.02)	9.46(0.02)
	SNR = 4.0	1.98(0.00)	2.46(0.00)	5.06(0.01)	2.83(0.00)	3.22(0.00)	5.43(0.01)	3.08(0.00)	2.95(0.00)	6.64(0.02)
	SNR = 4.5	1.44(0.00)	1.42(0.00)	2.61(0.00)	1.50(0.00)	1.60(0.00)	2.80(0.00)	2.03(0.00)	1.75(0.00)	2.99(0.01)
	SNR = 5.0	1.16(0.00)	1.10(0.00)	1.28(0.00)	1.28(0.00)	1.08(0.00)	1.40(0.00)	1.49(0.00)	1.34(0.00)	1.66(0.00)

- A larger window size T results in shorter out-of-control $ARLs$. The reason for this is that more temporal information is captured by larger window sizes, which translates into a higher detectability.

- For 45 out of the 48 combinations of T, β, and SNR, using the sampling percentages $q = 15\%$ and $q = 30\%$ is preferred. In general, when the SNR is smaller than 3, $q = 30\%$ is preferred, and, when the SNR is larger than 3, $q = 15\%$ achieves better results. This means that when the shift is small, more sensors are needed to detect it, but when the shift is large enough, a small number of sensors is sufficient. Given that in the simulation study only 2.5$\%$ of the pixels have a mean shift, sampling 15$\%$ of the sensors is enough to detect the change when the SNR is sufficiently large. However, it is important to remember that, in practice, this parameter is chosen based on resource constraints and sensing capabilities.

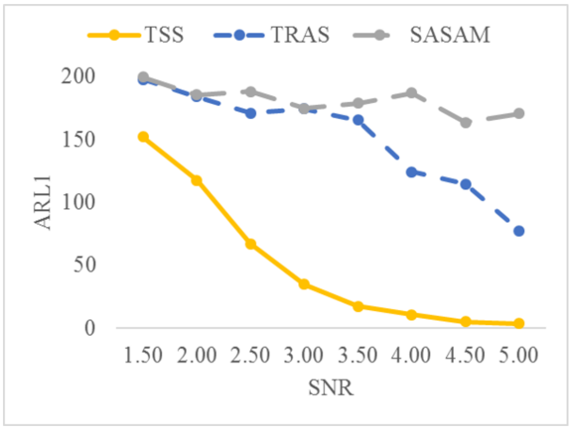
- For 43 out of the 48 combinations of T, q, and SNR, a parameter β favoring exploration is preferred, i.e. $\beta \geq 0.5$. This aligns with the intuition that more exploration increases the chances of detecting an out-of-control signal faster.

- The combination $T = 20$, $\beta = 0.8$, and $q = 15\%$ achieves the best results for all but two SNR values.

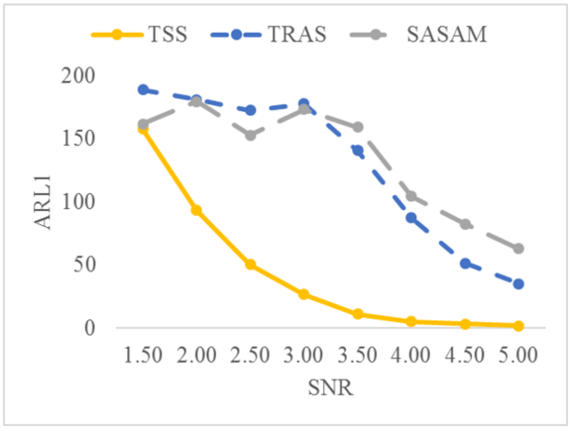
We proceed to compare the monitoring performance of our method with two benchmarks. Specifically, we compare the TSS algorithm with the sequential sampling method based on local CUSUM statistics [60] (designated as TRAS) and the spatial-adaptive sampling method [61] (designated as SASAM). For the TSS algorithm, we use the parameters with the worst performance, identified in Table 4.1 ($T = 5$ and $\beta = 0.2$). For the TRAS and the SASAM algorithms, we use the best parameters we could find by following the authors' recommendations on parameter selection. Additionally, it should be noted that as none of the benchmarks is designed for non-stationary data, to have a fair comparison, we take the difference between consecutive images to remove the background. For all methods, we fix the in-control ARL_0 to be 200 and compare the out-of-control ARL_1 under different SNR. The average time for computing the monitoring statistic for each acquisition time is presented in Table 4.2, and the out-of-control ARL curves obtained from 500 simulations replicates are shown in Figure 4.4. As can be seen from the figure, in all cases, the TSS algorithm has a better detection performance than the benchmark methods. The reason for this is that the TRAS and the SASAM algorithms lack the ability to model both the spatial and temporal structures of the streaming data. When the sampling percentage is 5%, the SASAM algorithm is not able to detect a change, the out-of-control ARL stays close to 200 for all levels of SNR. The performance of the TRAS algorithm is slightly better. However, for a SNR of 5, its ARL_1 is 15 times higher than the ARL_1 of the TSS algorithm. By using the low-rank structure of the data, and preserving the spatio-temporal information, we are able to detect changes more quickly with fewer observations. When the sampling percentage increases, the performance of the TRAS and the SASAM algorithms also improves. Nonetheless, these methods are unable to detect small changes. For $q = 30\%$, and $SNR = 3$, the ARL_1 for the TRAS algorithm is 5 times larger than that of the TSS algorithm. Even though the computational time of TSS is larger than that of TRAS and SASAM, it is still small enough to be used for online monitoring applications. In short, the simulation studies attest to the superior performance of the TSS algorithm over the benchmarks and validate its capability in capturing the spatio-temporal structure of the streaming data and effectively imputing missing data.

Table 4.2: Computational time of the TSS algorithm and other benchmark methods

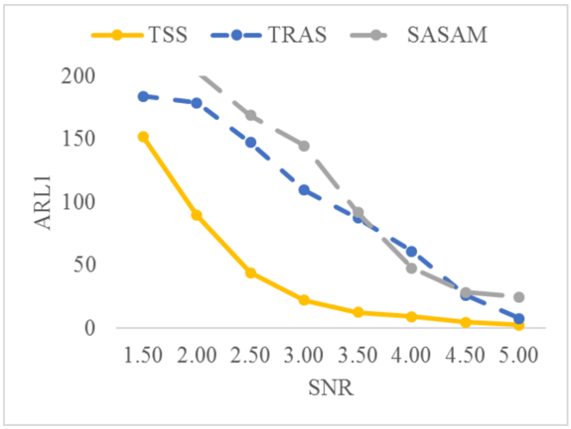
	TSS	TRAS	SASAM
Time (s.)	2.3366	0.0158	0.0692



(a) Sampling percentage 5%



(b) Sampling percentage 15%



(c) Sampling percentage 30%

Figure 4.4: Detection power comparison based on ARL_1

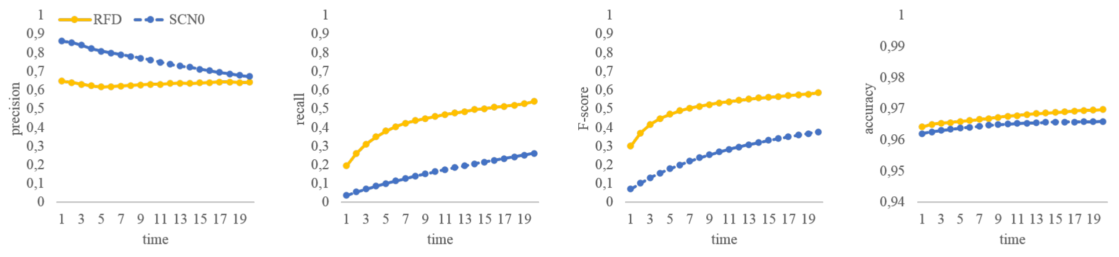
As discussed earlier, another advantage of the TSS algorithm is its diagnosability once an alarm is raised. Next, we evaluate the performance of the diagnosis method. Since none of the previous sequential sampling methods in the literature have proposed an explicit diagnosis solution, we cannot use them for comparison. Instead, we compare the results of the recursive diagnosis algorithm (designated as RD), with an ad-hoc approach that considers the non-zero entries in the sparse component as faulty signals (designated as SCN0). For this purpose, we compute the following four criteria after a shift is detected: (i) precision, defined as the proportion of detected anomalies that are true anomalies, (ii) recall, defined as the proportion of detected anomalies that are correctly identified, (iii) F-score, a single criterion that combines precision and recall by calculating their harmonic mean, and (iv) accuracy, defined as the proportion of signals that are correctly classified. We evaluate how these criteria evolve as the number of samples after detecting the change increases. For the RD algorithm, we use the following blurring filter for convolution:

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

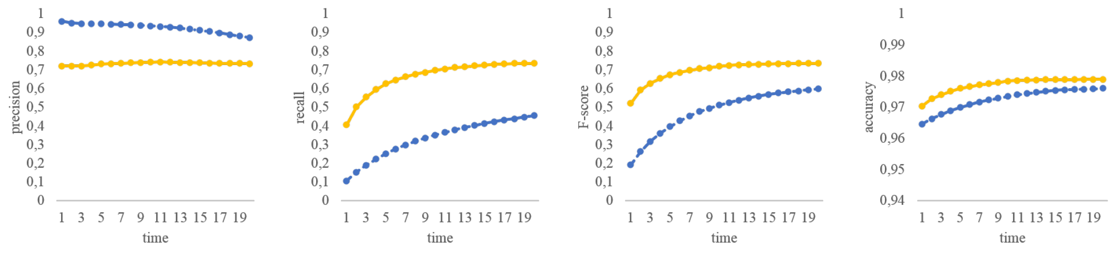
We fix $T = 20$ and $q = 15\%$, and study the behaviour of the two methods for different values of SNR ($SNR = 3$, $SNR = 5$) and β ($\beta = 0.2$, $\beta = 0.8$). The results for 500 simulation replicates, can be found in Figure 4.5 and Figure 4.6. We observe that, overall, the RD algorithm is better than the SCN0 algorithm in terms of recall, F-score, and accuracy. As time increases, the performance of the SCN0 algorithm converges to the performance of the RD algorithm. Exploiting the spatial structure of the data allows identifying the abnormal regions faster. Therefore, the RD algorithm should be preferred over the SCN0 algorithm.

Moreover, we observe that the RD algorithm results are better for $SNR = 5$, as a larger SNR is easier to detect. It is also interesting to see that the value of recall (proportion of correctly detected anomalies) increases as a function of time. This observation aligns with the fact that after an anomaly is detected, as time progresses, we have more samples, and thus more information, which allows us to identify the majority of the abnormal streams more effectively. After a finite number of acquisition times, it seems that the value of recall converges to a constant value. For $SNR = 3$, this number is around $t = 10$, and, for $SNR = 5$, it is close to 5. These results imply that when the shift magnitude is small, more samples are needed to detect the location of the faulty signals. On the other hand, we see that the value of precision (proportion of detected anomalies that are true anomalies) seems to be constant across time. As time progresses, with the collection of more samples, we are able to correctly identify more abnormal signals, however, given the thresholding scheme used by the RD algorithm, the number of false positives increases at the same rate as the number of true positives. Finally, it is important to notice that the performance is better for $\beta = 0.8$. It can be the case that the alarm is raised because variables from one abnormal region were sampled, it is important to keep exploring the space to detect other abnormal regions. In our simulation, since we have three abnormal regions, a larger β, favoring exploration, is preferred.

Additional simulation results for different sampling percentages q, presented in the Appendix, section C.5, indicate a similar behavior. An example of detected anomalies is presented in Figure 4.7, for $t = \tau + 6$, $SNR = 5$, $T = 20$, $q = 30\%$ and $\beta = 0.8$.

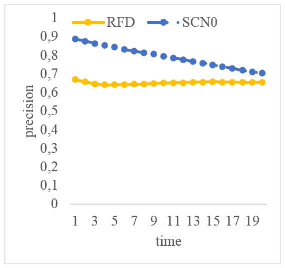


(a) Precision, SNR = 3 (b) Recall, SNR = 3 (c) F-score, SNR = 3 (d) Accuracy, SNR = 3

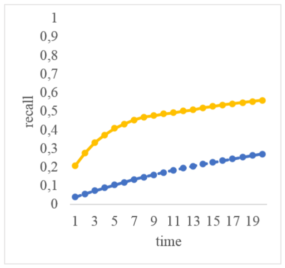


(e) Precision, SNR = 5 (f) Recall, SNR = 5 (g) F-score, SNR = 5 (h) Accuracy, SNR = 5

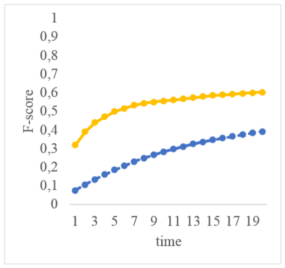
Figure 4.5: Diagnosis comparison for $T = 20$, $q = 15\%$, and $\beta = 0.2$ in terms of precision, recall, F-score, and accuracy, over 500 simulation replicates



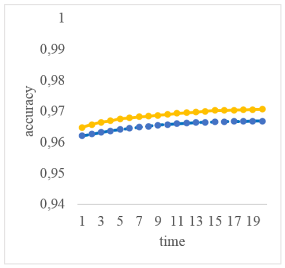
(a) Precision, SNR = 3



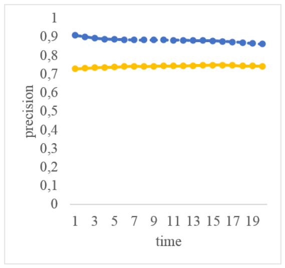
(b) Recall, SNR = 3



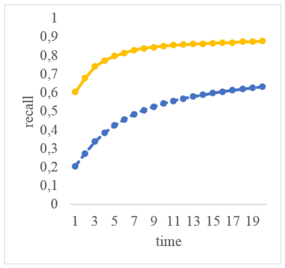
(c) F-score, SNR = 3



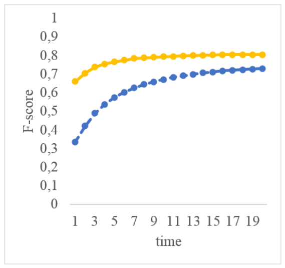
(d) Accuracy, SNR = 3



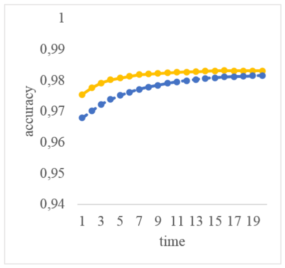
(e) Precision, SNR = 5



(f) Recall, SNR = 5

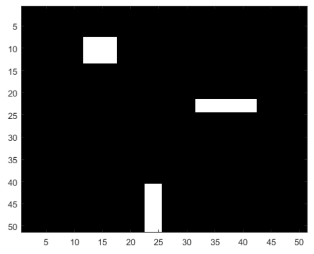


(g) F-score, SNR = 5

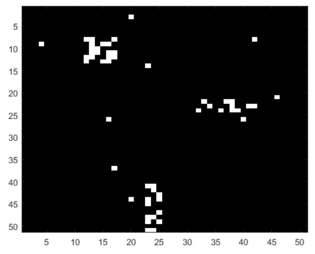


(h) Accuracy, SNR = 5

Figure 4.6: Diagnosis comparison for $T = 20$, $q = 15\%$, and $\beta = 0.8$ in terms of precision, recall, F-score, and accuracy, over 500 simulation replicates



(a) True anomaly



(b) SCN0

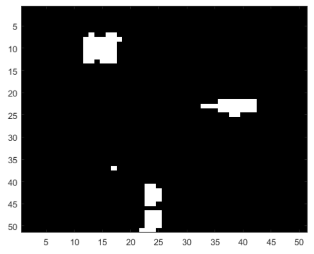


Figure 4.7: Detected anomalies with two methods at $\tau + 6$, for $SNR = 5$, $T = 20$, $q = 30\%$, and $\beta = 0.8$

4.8 Case Studies

In this section, the proposed TSS method for monitoring and diagnosis is applied to two real datasets. The first dataset contains data collected from a solar data observatory in the form of a stream of images. This dataset has been widely used in the literature [15, 60, 61, 81]. The second dataset provides information on soil temperature for a dryland agricultural field in the form of multi-stream scalars [82].

4.8.1 Online Monitoring of Solar Activity

As a first case study, we use a stream of solar images, collected from a solar data observatory. The goal is to monitor the solar activity to detect solar flares. A solar flare is defined as a sudden, transient, and intense variation in brightness observed over the Sun's surface [83]. A solar flare emits a large number of energetic charged particles, which may potentially cause failures of power grids or radio communications. Thus, the quick detection of solar flares is a critical task. However, due to the large amount of data generated by satellites, traditional methodologies for image change detection [84] can exceed the transmission and processing capabilities, and thus are incapable of detecting solar flares in real-time.

The dataset used in this study is publicly available online at http:/nislab.ee.duke.edu/MOUSSE/index.html. It contains a sequence of 300 images of size 232×292 pixels. This dataset was used to illustrate the performance of TRAS [60] and SASAM [61]. Note that as these sequential sampling methods assume that the data is independent and identically normally distributed, they first used the Multiscale Online Union of SubSpaces Estimation (MOUSSE) algorithm [81] to remove the background information, and then monitor the residuals which are approximately normally and independently distributed.

As in [60], we assume that due to limited transmission and processing capabilities, only 2000 (2.95%) out of the 67,774 pixels are available at each data frame, and can be sent back to the fusion center for analysis. The first 100 frames are considered as the in-control sample. To detect the solar flare in real-time, the proposed TSS algorithm is used with $T = 20$ and $\beta = 0.5$. The monitoring control chart is plotted in Figure 4.8. Two solar flares are detected at $[190, 201]$ and $[216, 258]$. This is compatible with the results reported in [60] and [61]. The TRAS algorithm detects two solar flares, at $t = 190$ and $t = 221$, while the SASAM algorithm only reports the detection of the first solar flare, at $t = 190$. The three methods detect the first solar flare at the same time. However, for the second solar flare, the detection delay of the TSS algorithm is much shorter than the detection

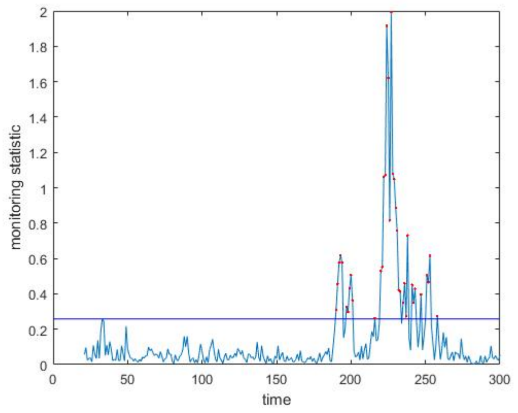
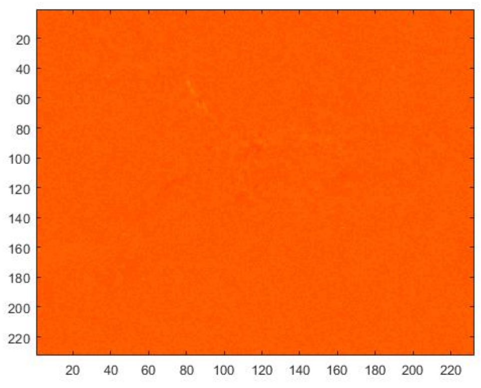


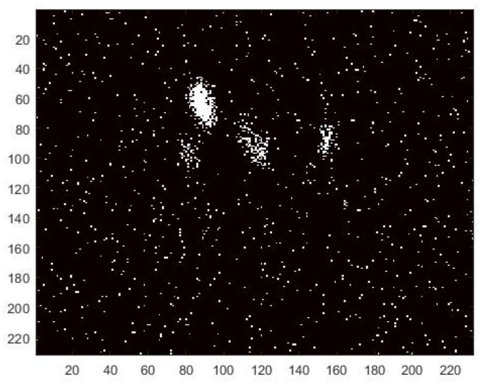
Figure 4.8: Control chart for solar flare monitoring (no background information)

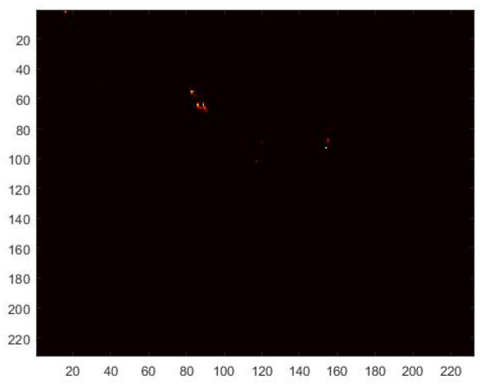
delay of the TRAS algorithm. Furthermore, the TSS algorithm is able to detect the second solar flare as fast as the methods presented in [15, 81], which use the full information of the 67,744 pixels in each data frame for monitoring.

To find the location of the solar flares in the out-of-control images, the proposed recursive diagnosis is used. Figure 4.9 and Figure 4.10 show the observed pixels and the identified anomalous regions at the time when the alarms were raised, as well as the identified anomalies after 10 iterations of the RD algorithm. The results indicate that the proposed method is not only able to detect changes in the process, but can also identify the location of solar flares in different time frames.

Previous results are obtained by removing the background information from the video and using the residuals for monitoring and diagnosis. This step is needed for TRAS and SASAM as they assume that the data streams are independent and identically normally distributed. However, this pre-processing step cannot be done in real-time and limits the applicability of the benchmarks. To further test the capabilities of our proposed method, we perform online monitoring of the raw images captured by the satellites. In order to capture the spatio-temporal structure of the data, we need to sample more pixels at each acquisition time. Since, unlike the previous scenario, the data stream contains spatial information, we increase the number of sampled pixels to 6,774 (10%). The monitoring control chart of


(a) Solar flare ($t = 190$)


(b) Samples ($t = 190$)


(c) Anomalies ($t = 190$)

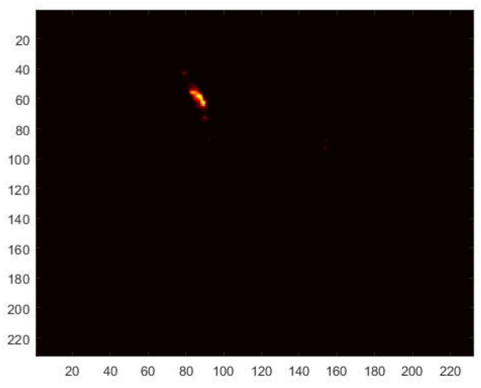
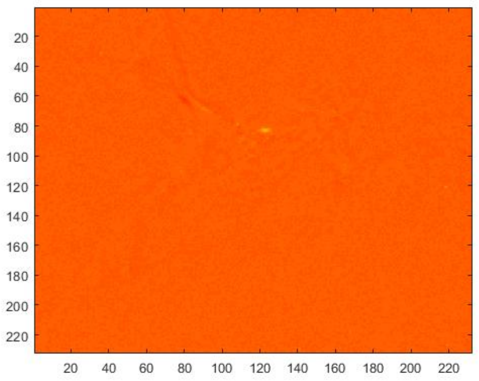
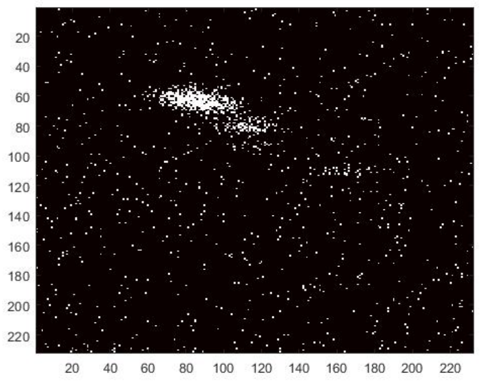

(d) Diagnosis ($t = 200$)

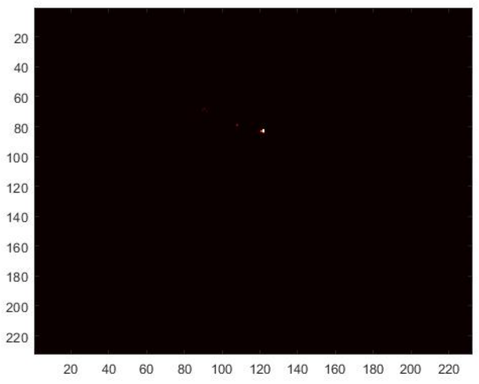
Figure 4.9: Detection results for solar flares at time $t = 190$ (no background information)



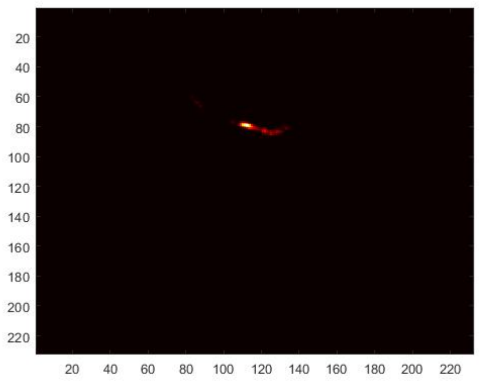
(a) Solar flare ($t = 216$)



(b) Samples ($t = 216$)



(c) Anomalies ($t = 216$)



(d) Diagnosis ($t = 226$)

Figure 4.10: Detection results for solar flares at time $t = 216$ (no background information)

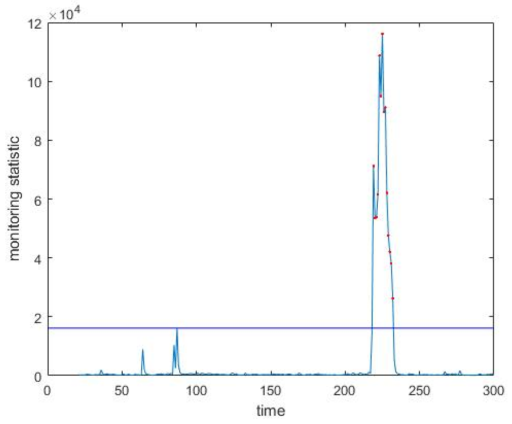


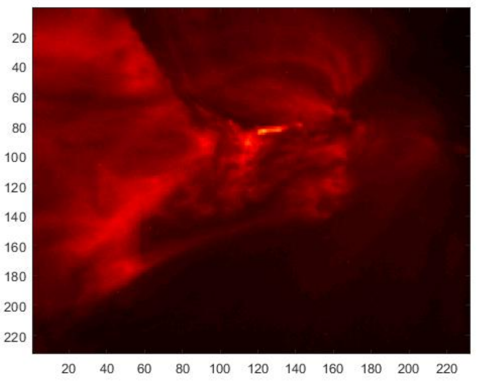
Figure 4.11: Control chart for solar flare monitoring (original data captured by satellites)

the TSS algorithm is given in Figure 4.11. In this case, we only detect the second solar flare at $t = 219$. Figure 4.12 shows the diagnosis results obtained once the monitoring algorithm detects the solar flare. With only 10% of the pixels used at each acquisition time, we are able to detect and locate the second solar flare in the video, without applying any pre-processing step to the original data. Note that the first solar flare could be detected by increasing the number of pixels observed with the cost of increased computational time.

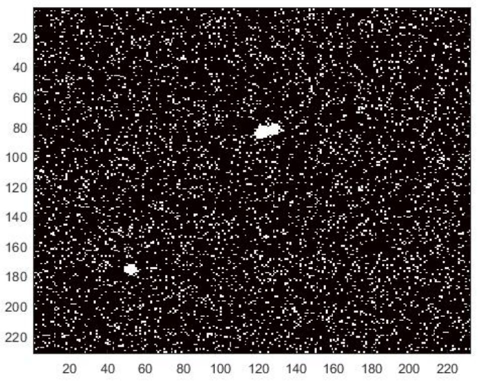
4.8.2 Online Monitoring of Water Temperature in a Dryland Agricultural Field

The second case study illustrates how to use the TSS algorithm for environmental monitoring under resource constraints. Understanding soil water dynamics is critical for agriculture, especially in dryland annual croplands that depend on stored soil water [85]. The stored water temperature highly affects the rate of plant development, and extreme temperatures have a negative impact on production [86]. In this section, we use the TSS algorithm to monitor water temperature at the R.J. Cook Agronomy Farm (CAF) [82], a long-term agro-ecosystem research site operated by Washington State University.

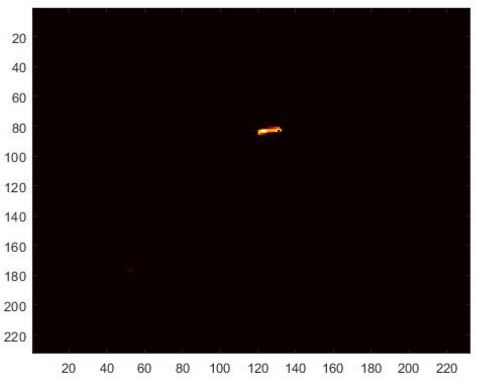
The CAF is a working farm, and farming operations have posed some challenges to the sensor network design and maintenance [85]. One of the challenges is the presence of missing data due to the battery death of the sensors. To overcome this limitation and



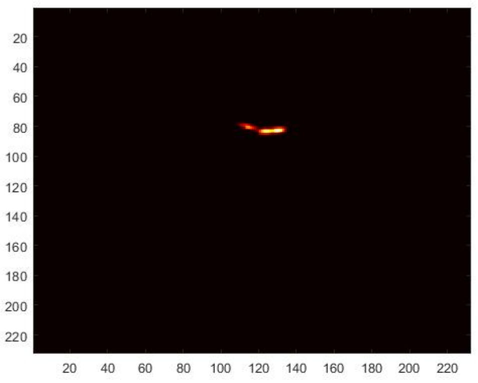
(a) Solar flare ($t = 219$)



(b) Samples ($t = 219$)



(c) Anomalies ($t = 219$)



(d) Diagnosis ($t = 239$)

Figure 4.12: Detection results for solar flare at time $t = 219$ (original data captured by satellites)

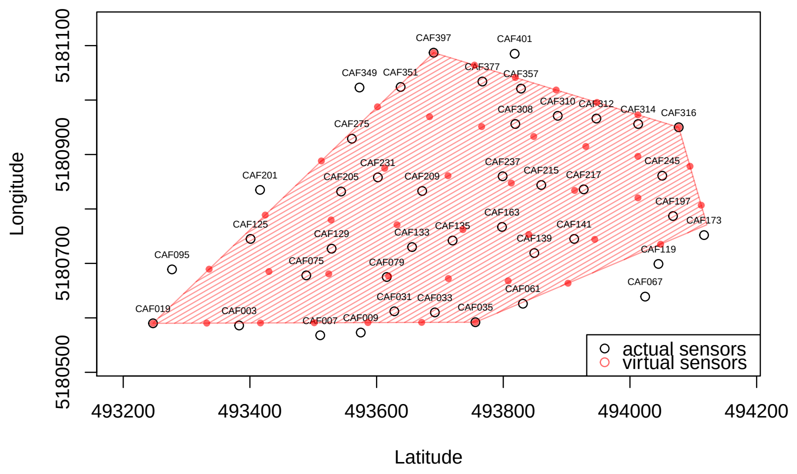


Figure 4.13: Sensor network measuring water temperature at the CAF

reduce maintenance costs, it is desirable to have only some of the sensors operational at any point in time. Next, we illustrate how to monitor water temperature to detect extreme temperatures that hinder the crops' development by deciding which sensors to activate at each acquisition time.

At the CAF, water temperature measurements are collected by 42 sensors, installed in a non-aligned systematic grid with a depth of 30cm. The sensors' location can be observed in Figure 4.13. We monitor the daily average temperature from January 1, 2013 to September 27, 2015.

First, we align the measurements to create a data tensor. For this purpose, we generate grid coordinates by uniformly distributing virtual measurement locations on the approximated pentagonal area of the agricultural field (shaded area shown in Figure 4.13). After generating the location coordinates of the virtual measurements, the virtual water temperature measurements are estimated using a Gaussian kernel:

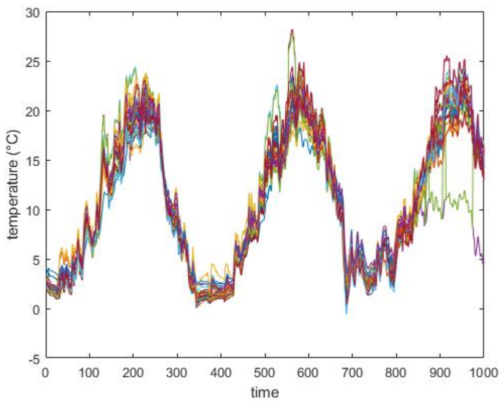
$$\hat{Temp}(x, y) = \frac{\sum_{k=1}^{42} Temp_k K(x_k, y_k)}{\sum_{k=1}^{42} K(x_k, y_k)}, \tag{4.16}$$

where x and y are the coordinates of the virtual sensor location, x_k and y_k are the coordinates of sensor k, $Temp_k$ is the temperature measurement of sensor k, and $K(x_k, y_k) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{(x_k-x)^2+(y_k-y)^2}{2\sigma^2}\right)$ with $\sigma = 50$. We obtain a $6 \times 7 \times 1000$ tensor to monitor water temperature. We set a window size T equal to 7 days, a sampling percentage q of 25% (at each acquisition time we have 11 operational sensors), and a balancing parameter β of 0.9.

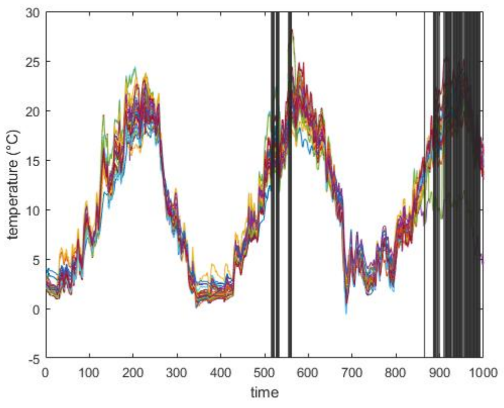
The monitoring results are in Figure 4.14. Three main abnormal periods are detected. The first one goes from 06/07/2014 to 06/24/2014, the second one starts on 07/16/2014 and ends on 07/23/2014, and the third one starts on 05/22/2015 and continues until the end of the dataset. We observe that the first two periods correspond with high temperatures, while the last abnormal period indicates low temperatures. This implies that the TSS algorithm is able to detect temperature changes in both directions. Next, we use our diagnosis algorithm to identify the regions in the farm where the stored soil water temperature is abnormal. The abnormal regions, for each period, are depicted in Figure 4.15. We verified that these regions correspond to the abnormal series observed in Figure 4.14. It is interesting to see that the water temperatures on the bottom right corner are always abnormal, which might be related to some physical and environmental properties of the reservoirs in this area. Further investigation, however, is required to verify this.

4.9 Conclusion

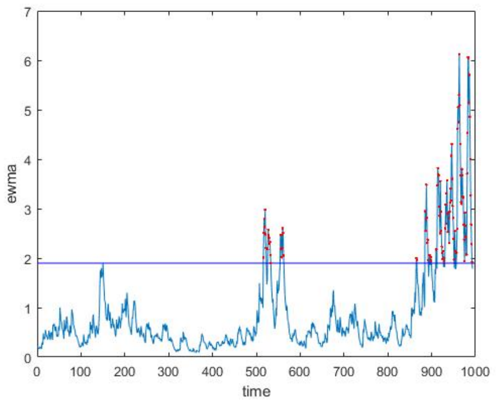
Online monitoring of HD streaming data under resource constraints is critical in various applications. In this chapter, we proposed a new sequential sampling methodology for real-time monitoring and diagnosis of sensing systems that generate incomplete data streams. First, we developed the RTR algorithm, a recursive tensor recovery algorithm that effectively decomposes the streaming tensor into a low-rank component and a sparse component. By estimating the low-rank component, we were able to approximate the value of the unobserved variables at each acquisition time. This represents an advantage over other adaptive sampling methods in the literature. By monitoring the sparse component, using an EWMA control chart, we were able to detect abrupt changes in the process. We also



(a) Temperature time series

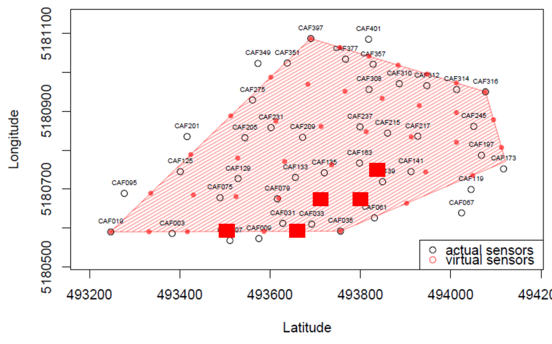


(b) Identified abnormal periods

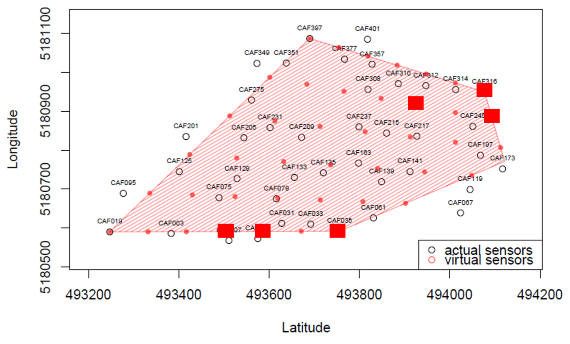


(c) Monitoring control chart

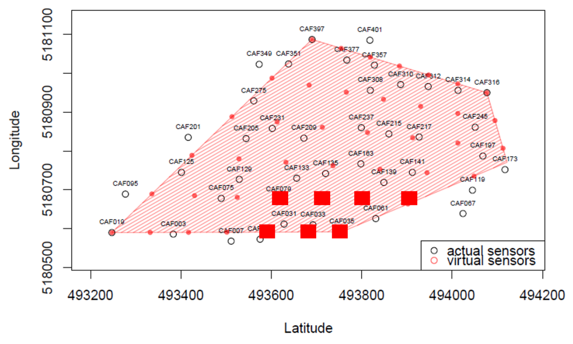
Figure 4.14: Temperature (°C) readings for the 42 sensors at the CAF with out-of-control periods and monitoring control chart



(a) Abnormal region 06/14/2014



(b) Abnormal region 07/23/2014



(c) Abnormal region 05/29/2015

Figure 4.15: Detection of abnormal regions (depicted as squares) of water temperature at the CAF

developed a recursive diagnosis algorithm that exploits the spatial correlation of the data to locate all the abnormal streams quickly once an alarm is raised. Finally, we developed an adaptive tensor sequential sampling algorithm, that defines the variables to observe at each acquisition time. The TSS algorithm balances exploration, to detect a wide variety of shifts, and exploitation, to quickly locate abnormal streams. Since all of our algorithms have a recursive form, they are computationally efficient and allow for real-time monitoring and diagnosis. In the simulation study, we showed that the proposed method outperforms existing sequential sampling methods, under resource constraints. This happens because we incorporate both the spatial and temporal information contained in the data. The results from the case studies demonstrated the capability of the proposed method in identifying not only the time of process changes but also the location of the detected anomalies.

The proposed methodology was developed for the monitoring and diagnosis of sparse persistent changes. Proposition 4 and Proposition 5 only hold if the changes in the data streams are persistent in time. When sparse transient changes (outliers) appear, one possibility is that no alarm is raised by our methodology. This will be the case if we do not sample the streams during the short period when they present an abnormal behavior. The second possibility is that the abnormal streams are sampled. In this case, the abnormal behavior will be captured by the sparse component, and, depending on the magnitude of the change, an alarm will be raised. Since we monitor the sum of the absolute values of the sparse component, we could potentially differentiate between transient changes and persistent changes by visually inspecting the behavior of the control chart after the alarm is raised. However, the differentiation between outliers and sparse changes needs to be further investigated.

In this chapter, we assumed that at each acquisition time, we can decide to observe any of the variables. However, in practice, it is possible to have additional restrictions. For example, in a grid of sensors, it is possible to have malfunctioning sensors that cannot be used. In the future, we plan to extend our method to be applicable under such constraints.

CHAPTER 5
CONCLUSIONS AND FUTURE RESEARCH

5.1 Conclusions

This thesis has presented new methodologies to analyze high-dimensional data collected by sensors to learn the complex spatio-temporal structure of a variety of systems for process monitoring and diagnosis. The main research results and new contributions of this dissertation are summarized as follows.

In Chapter 2, a new methodology was developed to learn directed graphical models from functional data for root-cause analysis and diagnosis. If the structure of the functional DGM was known, functional observations were used to fit function-to-function linear regressions and learn the parameters of the conditional distributions governing the model. To transform the infinite-dimensional regression problems into finite-dimensional ones, we used data-driven or domain knowledge functional basis expansions. To evaluate the performance of the parameter learning methodology, we conducted a simulation study and compared the performance of the proposed methodology with a benchmark method. The mean square prediction errors for the developed methodology were consistently smaller. On the other hand, if the structure of the DGM was unknown, we proposed to add a penalty term to the loss function of the function-to-function linear regressions to learn the set of parents for each node. To reduce the dimensionality of the problem, we used a domain knowledge basis and employed cyclic coordinate accelerated proximal gradient descent to solve the regression problems. With simulations, we showed that the structure learning methodology is able to learn a structure with a recall of 100% and with an F_1-score of 81%. In the case study, we proved that the proposed framework is able to detect different root-causes of λ-undershoot in an internal combustion engine.

In Chapter 3, a new online monitoring strategy was proposed to detect structural changes in a system collecting time-dependent high-dimensional streaming data. The main contributions of this Chapter are (i) learning in real-time the complex cross-correlation structure between the system's variables and (ii) detecting structural changes due to the system evolution. The first step in the proposed methodology was to divide the streaming data into windows. Then, sequentially, we estimated an undirected graphical model for each window. To learn the cross-correlation structure, we exploited the spectral information contained in the data and used graphical LASSO for time series data. To detect structural changes, we allowed the graphical model to evolve over time while regularizing the change flexibility. An efficient framework based on the Alternating Direction Method of Multipliers was proposed for online optimization and change-point detection. Numerical studies together with two case studies demonstrated the efficiency and applicability of the proposed methodology. Using dynamic spectral functional graphical models, we were able to perform human motion tracking and to monitor functional brain connectivity.

In Chapter 4, a novel adaptive sequential sampling strategy for real-time monitoring and diagnosis of sensing systems that generate incomplete data streams due to resources constraints was developed. At each acquisition time, the sensor readings were organized into a tensor. Then, a recursive tensor recovery algorithm was proposed to estimate two key components from the partially observed data stream: a low-rank component, that captured the spatio-temporal structure of the data and contained estimates for the unobserved variables, and a sparse component, that captured the suspicious variables that may indicate an out-of-control state. To detect statistically significant changes using the sparse component, we constructed an EWMA control chart. If an alarm was raised, we proposed a recursive diagnosis algorithm to find the abnormal regions. Otherwise, based on the location and pattern of the suspicious observations in the sparse component, we decided where to sample next. Since all of our algorithms have a recursive form, they are computationally efficient and allow for real-time monitoring and diagnosis. In the simulation study, we showed that the proposed method outperforms existing sequential sampling methods for online monitoring under resource constraints. This happens because we incorporate both the spatial and temporal information contained in the data. Furthermore, the results from the case studies demonstrated the capability of the proposed method in identifying not only the time of process changes but also the location of the detected anomalies.

5.2 Future Research

Modeling and analyzing high-dimensional sensing data for system monitoring and diagnosis is an important yet challenging research problem. In this dissertation, we have made efforts to tackle some of the challenges. However, there are still many interesting research opportunities. Some of which are summarized in what follows.

It would be exciting to extend the methodologies presented in Chapters 2 and 3 and learn probabilistic graphical models for systems where the sensing data comes in different forms (functions, images, videos, point-clouds). Nowadays, with technological advancements, sensors are able to collect heterogeneous types of data. For example, in additive manufacturing, the layering process is monitored by cameras, while different sensors capture important process variables such as temperature and speed. In healthcare, for example, ambient assisted smart living homes collect data from cameras, motion sensors, and sensors carried by the user. It is imperative to learn the relationships between the variables in these systems for real-time system monitoring and control, and for accurate fault diagnosis. This is a challenging task, as the correlation between heterogeneous forms of data needs to be modeled. To approach this task, new statistical learning techniques need to be developed. A potential avenue is to use tensors or manifolds to represent the data collected and learn heterogeneous probabilistic graphical models.

Potential extensions for Chapter 4 can also be considered. First, the proposed methodology was developed for monitoring and diagnosis of sparse persistent changes. However, it can also detect sparse transient changes (outliers). Currently, the differentiation between outliers and sparse persistent changes is not possible. It would be interesting to develop an adaptive sequential sampling methodology robust to outliers in the data. Second, in Chapter 4, we assumed that at each acquisition time, we could decide to observe any of the variables. However, in practice, it is possible to have additional restrictions due to malfunctioning sensors, for example. In the future, it would be great to extend the proposed method to incorporate such constraints.

Another fruitful research avenue revolves around federated learning. Federated learning involves training statistical models over remote devices or siloed data centers while keeping data localized. Mobile phones, wearable devices, and autonomous vehicles are just a few of the modern distributed networks generating a wealth of data each day. Due to the growing computational power of these devices, coupled with concerns over transmitting private information, it is increasingly attractive to store data locally and push computations to the edge. A global statistical model is then learned based on the information shared by a small number of devices. One question that needs to be answered is how to select the devices that will participate in each round of training. Currently, the decision is based on the devices' specific computation and communication capabilities, assuming these are static over time. It would be interesting to develop an adaptive sampling strategy to sample a set of small but sufficiently representative devices based on the underlying statistical structure. In this setting, it is critical to ensure that the proposed methodology does not induce bias in the model. Another interesting challenge arises from the fact that the data collected from different devices is heterogeneous (user-specific distribution). Methods that are robust and scalable to train models within the federated setting need to be developed.

Appendices

APPENDIX A
SUPPLEMENTARY MATERIALS OF CHAPTER 2

A.1 Proof of Proposition 1: Proximal Gradient Descent Closed Form Solution

Proposition 1. *The proximal gradient descent, with step-size $s^{(t)}$, at iteration t, presented in Equation 2.25, consists of the following two steps:*

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

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:

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

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 $\boldsymbol{b}_{kj} = 0$. 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:

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

$\square$

A.2 Proof of Proposition 2: Convergence of the PGD Algorithm

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.*

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

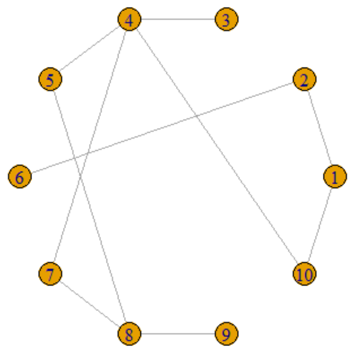
$$\begin{aligned}
||\nabla f(\alpha) - \nabla f(\beta)||_2 &= ||-\boldsymbol{\Xi}_{kj}^\top(\boldsymbol{r}_{kj} - \boldsymbol{\Xi}_{kj}\alpha) + \lambda\alpha + \boldsymbol{\Xi}_{kj}^\top(\boldsymbol{r}_{kj} - \boldsymbol{\Xi}_{kj}\beta) - \lambda\beta)||_2 \\
&= ||(\boldsymbol{\Xi}_{kj}^\top\boldsymbol{\Xi}_{kj}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

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 $\boldsymbol{\Xi}_{kj}^\top\boldsymbol{\Xi}_{kj} + \lambda I$. □

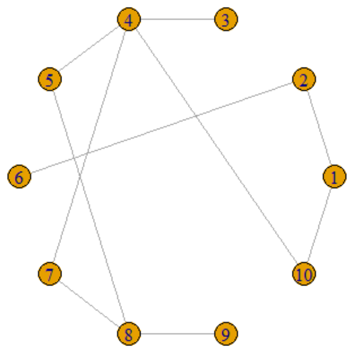
APPENDIX B

SUPPLEMENTARY MATERIALS OF CHAPTER 3

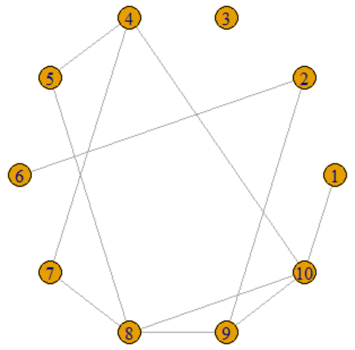
B.1 Structure Learning: Additional Simulation Results



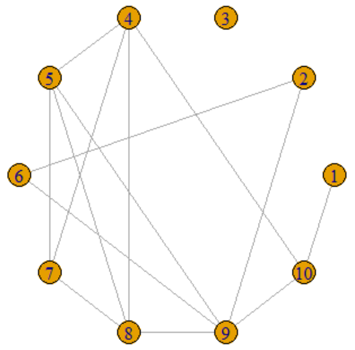
(a) True Structure 1



(b) Estimated structure 1

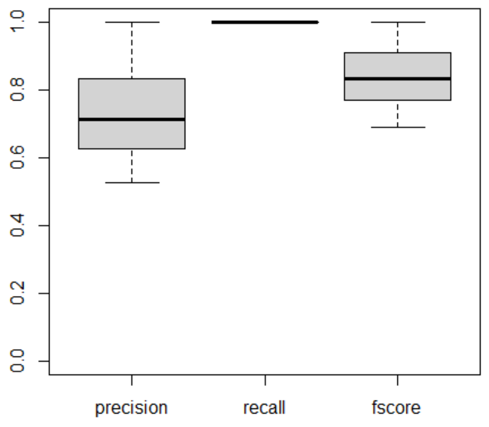


(c) True structure 2

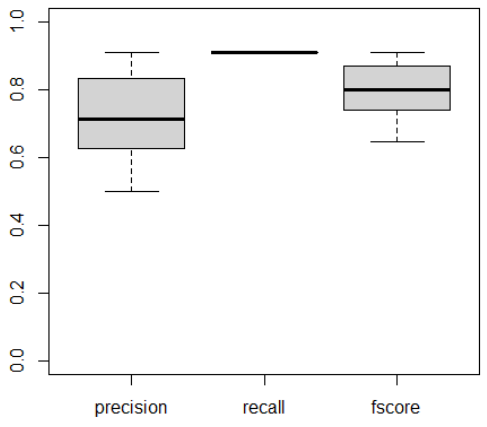


(d) Estimated structure 2

Figure B.1: Example of true and estimated structures for one simulation run for SNR=0.25



(a) Structure 1



(b) Structure 2

Figure B.2: Structure learning performance for SNR=0.25, in terms of precision, recall,and F-score, over 100 simulation replicates

APPENDIX C
SUPPLEMENTARY MATERIALS OF CHAPTER 4

C.1 Soft-thresholding: Proof of Proposition 3

Proposition 3. *Given $\boldsymbol{S}_k$, for $k < t$, and $\hat{\boldsymbol{\mathcal{L}}}^{(t)} = [\![\hat{\boldsymbol{A}}^{(t)}, \hat{\boldsymbol{B}}^{(t)}, \hat{\boldsymbol{C}}^{(t)}]\!]$, the solution to the optimization problem in Equation 4.6 is obtained by the soft-thresholding function:*

$$\hat{s}_{i,j,t} = S_\lambda\left(w_{i,j,t}\left(x_{i,j,t} - \sum_{r=1}^{R} \hat{a}_{ir}^{(t)}\hat{b}_{jr}^{(t)}\hat{c}_{Tr}^{(t)}\right)\right),$$

where $S_\lambda(\cdot) = sgn(\cdot)\left(|\cdot| - \lambda\right)_+$, $sgn(\cdot)$ is the sign function, and $(\cdot)_+ = \max(\cdot, 0)$.

Proof. The optimization problem in Equation 4.6 can be written as:

$$\min_{s_{i,j,t}} \quad \frac{1}{2}w_{i,j,t}\left(x_{i,j,t} - \sum_{r=1}^{R} \hat{a}_{ir}^{(t)}\hat{b}_{jr}^{(t)}\hat{c}_{Tr}^{(t)} - s_{i,j,t}\right)^2 + \lambda\sum_{i=1}^{I}\sum_{j=1}^{J} w_{i,j,t}|s_{i,j,t}|.$$

We need to consider two cases. If $w_{i,j,t} = 0$, $s_{i,j,t}$ can take any value, in this case we set $s_{i,j,t} = 0$. If $w_{i,j,t} = 1$, we need to solve:

$$\min_{s_{i,j,t}} \quad \frac{1}{2}\left(x_{i,j,t} - \sum_{r=1}^{R} \hat{a}_{ir}^{(t)}\hat{b}_{jr}^{(t)}\hat{c}_{Tr}^{(t)} - s_{i,j,t}\right)^2 + \lambda\sum_{i=1}^{I}\sum_{j=1}^{J} |s_{i,j,t}|.$$

The sub-gradient equation for this problem is:

$$-\left(x_{i,j,t} - \sum_{r=1}^{R} \hat{a}_{ir}^{(t)}\hat{b}_{jr}^{(t)}\hat{c}_{Tr}^{(t)}\right) + s_{i,j,t} + \lambda\partial|s_{i,j,t}| = 0$$

where $\partial|s_{i,j,t}| = \text{sgn}(s_{i,j,t})$, if $s_{i,j,t} \neq 0$, and $\partial|s_{i,j,t}| \in [-1, 1]$, if $s_{i,j,t} = 0$. With this we have that:

$$\hat{s}_{i,j,t} = \begin{cases} \left(x_{i,j,t} - \sum_{r=1}^{R} \hat{a}_{ir}^{(t)} \hat{b}_{jr}^{(t)} \hat{c}_{Tr}^{(t)}\right) - \lambda & \text{if} \quad \left(x_{i,j,t} - \sum_{r=1}^{R} \hat{a}_{ir}^{(t)} \hat{b}_{jr}^{(t)} \hat{c}_{Tr}^{(t)}\right) > \lambda \\ 0 & \text{if} \quad \left(x_{i,j,t} - \sum_{r=1}^{R} \hat{a}_{ir}^{(t)} \hat{b}_{jr}^{(t)} \hat{c}_{Tr}^{(t)}\right) = \lambda \\ \left(x_{i,j,t} - \sum_{r=1}^{R} \hat{a}_{ir}^{(t)} \hat{b}_{jr}^{(t)} \hat{c}_{Tr}^{(t)}\right) + \lambda & \text{if} \quad \left(x_{i,j,t} - \sum_{r=1}^{R} \hat{a}_{ir}^{(t)} \hat{b}_{jr}^{(t)} \hat{c}_{Tr}^{(t)}\right) < -\lambda \end{cases},$$

which can be written succinctly as:

$$\hat{s}_{i,j,t} = S_\lambda \left(x_{i,j,t} - \sum_{r=1}^{R} \hat{a}_{ir}^{(t)} \hat{b}_{jr}^{(t)} \hat{c}_{Tr}^{(t)} \right).$$

□

C.2 Exploration: Proof of Proposition 4

Proposition 4. *Let V_1 denote the set of variables $\boldsymbol{n} \in N$ which will never be observed after a finite time t_0. If the proposed sequential sampling procedure is followed, with probability 1 we observe all the variables after a finite time t_0. That is, $P(V_1 = \emptyset) = 1$, where $\emptyset$ represents the empty set.*

Proof. Let $\boldsymbol{v}_1 \in V_1$. For all $t \geq t_0$, we must have that

$$p_t(\boldsymbol{v}_1) = \beta r_t(\boldsymbol{v}_1) + (1-\beta) g_t(\boldsymbol{v}_1) = 0$$

This means that $r_t(\boldsymbol{v}_1) = 0$ and $g_t(\boldsymbol{v}_1) = 0$. We have that

$$r_t(\boldsymbol{v}_1) = 0 \Leftrightarrow 1 - \frac{1}{t-1} \sum_{k=1}^{t-1} w_{\boldsymbol{v}_1,k} = 0 \Leftrightarrow \sum_{k=1}^{t-1} w_{\boldsymbol{v}_1,k} = t-1.$$

This means that variable $\boldsymbol{v}_1$ has been observed at every acquisition time. This contracticts the fact that $\boldsymbol{v}_1 \in V_1$. Therefore, we conclude that $P(V_1 = \emptyset) = 1$. □

C.3 Exploitation: Proof of Proposition 5

Proposition 5. *Assume that the entries of the noise component are normally and independently distributed with mean zero and variance σ^2. Let V_2 denote the set of variables $\boldsymbol{n} \in N$, which have a mean shift at time t_a. If the mean shift is larger than $\sigma\sqrt{2\log(q)}$, for any finite time $t \geq t_a$, once a variable $\boldsymbol{v}_2 \in V_2$ is observed, there is a non-zero probability that the variable will be observed at each time after t.*

Proof. Let $\boldsymbol{v}_2 \in V_2$. Suppose $\boldsymbol{v}_2$ is observed at time $t_0 \geq t_a$. We want to show that

$$P(w_{\boldsymbol{v}_2,t_0} = 1, w_{\boldsymbol{v}_2,t_0+1} = 1, \cdots) > 0.$$

We are going to prove this using induction.

- **Base case:** Since at time t_0 we observed $\boldsymbol{v}_2$, we have that

$$P(w_{\boldsymbol{v}_2,t_0} = 1) = 1 > 0.$$

- **Induction step:** Suppose that for $\eta > 0$ we have that

$$P(w_{\boldsymbol{v}_2,t_0} = 1, \cdots, w_{\boldsymbol{v}_2,t_0+\eta} = 1) > 0.$$

  We want to show that

$$P(w_{\boldsymbol{v}_2,t_0} = 1, \cdots, w_{\boldsymbol{v}_2,t_0+\eta} = 1, w_{\boldsymbol{v}_2,t_0+(\eta+1)} = 1) > 0.$$

  Using conditional probabilities, we have that

$$\begin{aligned}&P(w_{\boldsymbol{v}_2,t_0} = 1, \cdots, w_{\boldsymbol{v}_2,t_0+(\eta+1)} = 1)\\&= P(w_{\boldsymbol{v}_2,t_0+(\eta+1)} = 1 | w_{\boldsymbol{v}_2,t_0} = 1, \cdots, w_{\boldsymbol{v}_2,t_0+\eta} = 1) P(w_{\boldsymbol{v}_2,t_0} = 1, \cdots, w_{\boldsymbol{v}_2,t_0+\eta} = 1)\end{aligned}$$

The induction hypothesis is that $P(w_{\boldsymbol{v}_2,t_0} = 1, \cdots, w_{\boldsymbol{v}_2,t_0+\eta} = 1) > 0$, it remains to show that $P(w_{\boldsymbol{v}_2,t_0+(\eta+1)} = 1 | w_{\boldsymbol{v}_2,t_0} = 1, \cdots, w_{\boldsymbol{v}_2,t_0+\eta} = 1) > 0$. The latter is true if and only if, given that $w_{\boldsymbol{v}_2,t_0} = 1, \cdots, w_{\boldsymbol{v}_2,t_0+\eta} = 1$,

$$p_{t_0+(\eta+1)}(\boldsymbol{v}_2) = \beta r_{t_0+(\eta+1)}(\boldsymbol{v}_2) + (1-\beta) g_{t_0+(\eta+1)}(\boldsymbol{v}_2) > 0.$$

Next, we show that $g_{t_0+(\eta+1)}(\boldsymbol{v}_2) > 0$, which concludes the proof of Proposition 5.

We have that

$$g_{t_0+(\eta+1)}(\boldsymbol{v}_2) = \frac{1}{\#(U_{t_0+\eta})} \sum_{i=1}^{\#(U_{t_0+\eta})} K_\theta(\boldsymbol{v}_2 - \boldsymbol{u}_i),$$

where $U_{t_0+\eta} = \{(i,j) |\ \hat{s}_{i,j,t_0+\eta} \neq 0,\ i = 1, \cdots, I,\ j = 1, \cdots, J\}$ and $\boldsymbol{u}_i \in U_{t_0+\eta}$, for $i = 1, \cdots, \#(U_{t_0+\eta})$.

If $\boldsymbol{v}_2 \in U_{t_0+\eta}$, by definition of kernel density function, we have that $K_\theta(\boldsymbol{v}_2 - \boldsymbol{v}_2) > 0$, and, therefore, $g_{t_0+(\eta+1)}(\boldsymbol{v}_2) > 0$.

We have that $\boldsymbol{v}_2 \in U_{t_0+\eta}$ if and only if

$$\hat{s}_{\boldsymbol{v}_2,t+\eta} = S_{\lambda^*}\left(w_{\boldsymbol{v}_2,t+\eta}\left(x_{\boldsymbol{v}_2,t+\eta} - \hat{l}_{\boldsymbol{v}_2,t+\eta}\right)\right) \neq 0.$$

Since $w_{\boldsymbol{v}_2,t+\eta} = 1$, we have that $\hat{s}_{\boldsymbol{v}_2,t+\eta} \neq 0$ if and only if $\left| x_{\boldsymbol{v}_2,t+\eta} - \hat{l}_{\boldsymbol{v}_2,t+\eta} \right| > \lambda^*$, where $\lambda^* = \min(\lambda_{BIC}, \hat{\sigma}\sqrt{2\log(q)})$. One of the assumptions of this proposition is that the mean shift of $\boldsymbol{v}_2$ is larger than $\sigma\sqrt{2\log(q)})$, therefore, we have that

$$\left| x_{\boldsymbol{v}_2,t+\eta} - \hat{l}_{\boldsymbol{v}_2,t+\eta} \right| > \hat{\sigma}\sqrt{2\log(q)} \geq \min(\lambda_{BIC}, \hat{\sigma}\sqrt{2\log(q)}).$$

We conclude that $\boldsymbol{v}_2 \in U_{t_0+\eta}$.

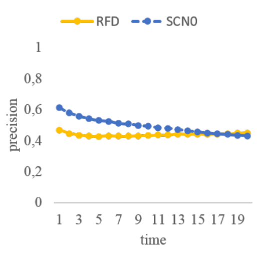
$\square$

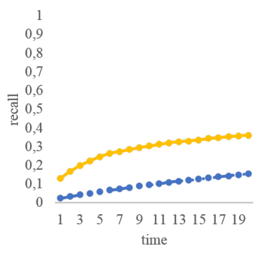
C.4 Monitoring: Control Limit

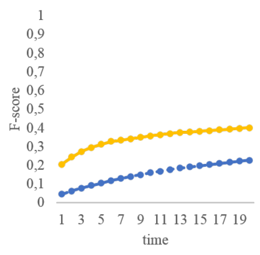
This appendix describes the detailed steps to estimate the control limit h given a prescribed in-control ARL. In the simulation experiments, we used an in-control ARL of 200.

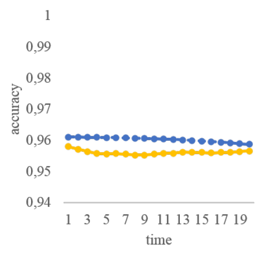
1. Set $h_{\min}$ and $h_{\max}$ a small and a large values as the initial lower and upper bounds of h, respectively. Let $h = \frac{h_{\min}+h_{\max}}{2}$.
2. Generate an in-control sample of HD data streams. For the simulation experiments, we generated 401 frames from a heat transfer process, as described in Section 7.
3. Implement the TSS algorithm and record the index of the first out-of-control sample, RL.
4. Repeat steps 2 and 3 for $M(= 500)$ times. Calculate the average of RL, $\bar{RL}$.
5. If the $\bar{RL}$ value is larger than the prescribed in-control ARL, let $h_{\max} = h$. Otherwise, let $h_{\min} = h$. Then, update $h = \frac{h_{\min}+h_{\max}}{2}$.
6. Repeat steps 2-5 until there is no difference between $\bar{RL}$ and the prescribed in-control ARL.

C.5 Diagnosis: Additional Simulations

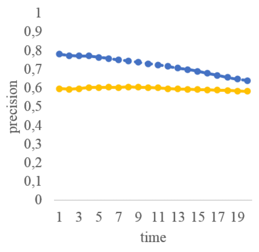


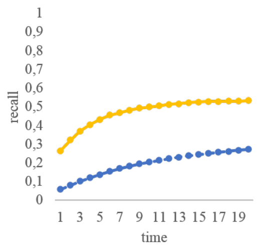
(a) Precision, SNR = 3


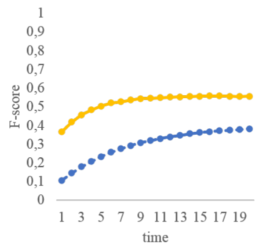
(b) Recall, SNR = 3


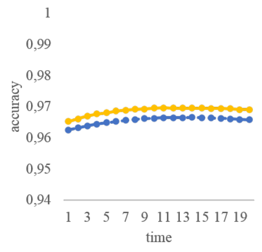
(c) F-score, SNR = 3


(d) Accuracy, SNR = 3



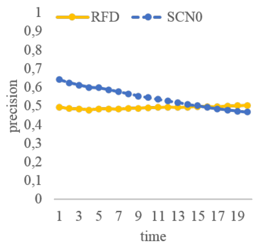
(e) Precision, SNR = 5


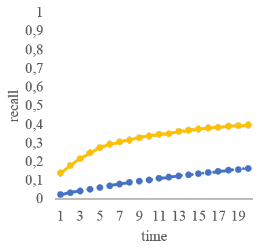
(f) Recall, SNR = 5


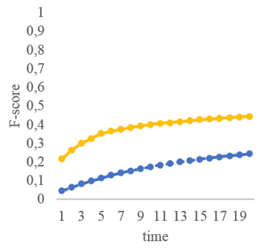
(g) F-score, SNR = 5


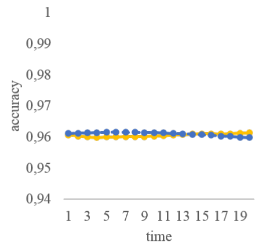
(h) Accuracy, SNR = 5

Figure C.1: Diagnosis comparison for $T = 20$, $q = 5\%$, and $\beta = 0.2$ in terms of precision, recall, F-score, and accuracy, over 500 simulation replicates

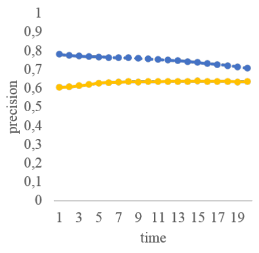


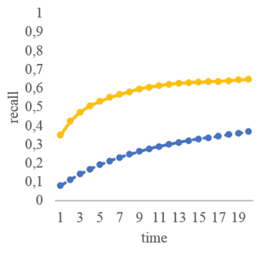
(a) Precision, SNR = 3


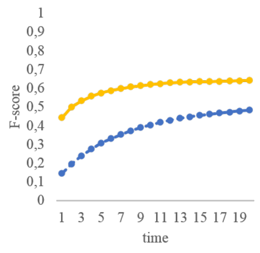
(b) Recall, SNR = 3


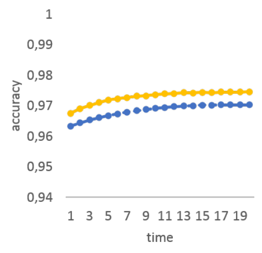
(c) F-score, SNR = 3


(d) Accuracy, SNR = 3



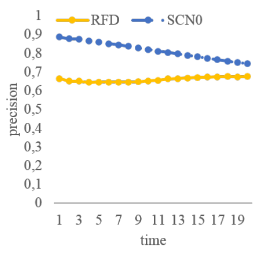
(e) Precision, SNR = 5


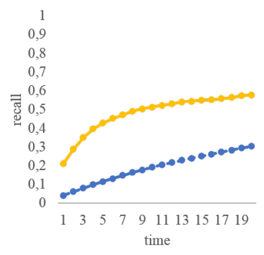
(f) Recall, SNR = 5


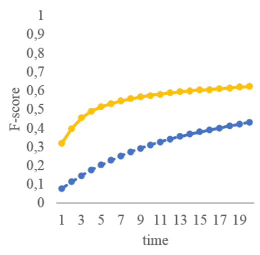
(g) F-score, SNR = 5


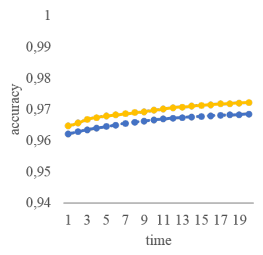
(h) Accuracy, SNR = 5

Figure C.2: Diagnosis comparison for $T = 20$, $q = 5\%$, and $\beta = 0.8$ in terms of precision, recall, F-score, and accuracy, over 500 simulation replicates

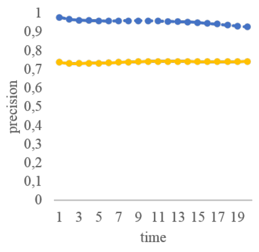


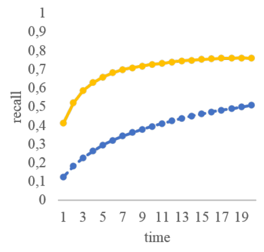
(a) Precision, SNR = 3


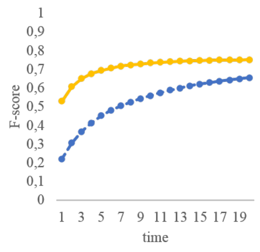
(b) Recall, SNR = 3


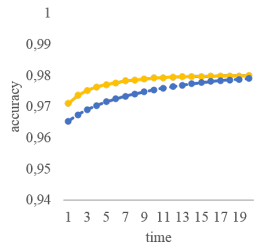
(c) F-score, SNR = 3


(d) Accuracy, SNR = 3



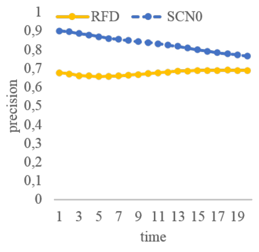
(e) Precision, SNR = 5


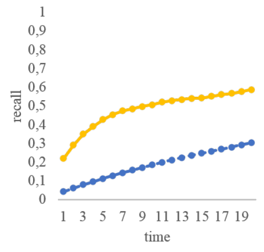
(f) Recall, SNR = 5


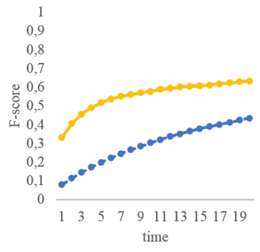
(g) F-score, SNR = 5


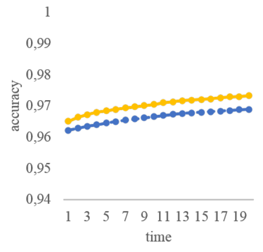
(h) Accuracy, SNR = 5

Figure C.3: Diagnosis comparison for $T = 20$, $q = 30\%$, and $\beta = 0.2$ in terms of precision, recall, F-score, and accuracy, over 500 simulation replicates

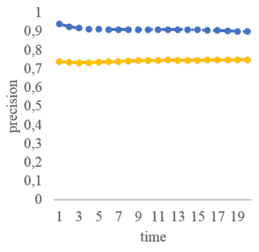


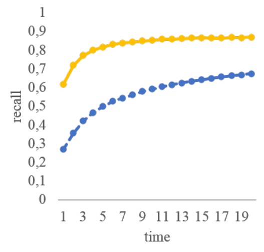
(a) Precision, SNR = 3


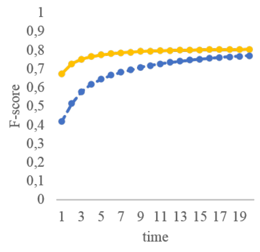
(b) Recall, SNR = 3


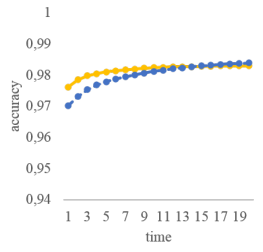
(c) F-score, SNR = 3


(d) Accuracy, SNR = 3



(e) Precision, SNR = 5


(f) Recall, SNR = 5


(g) F-score, SNR = 5


(h) Accuracy, SNR = 5

Figure C.4: Diagnosis comparison for $T = 20$, $q = 30\%$, and $\beta = 0.8$ in terms of precision, recall, F-score, and accuracy, over 500 simulation replicates

REFERENCES

[1] H. Langseth and L. Portinale, “Bayesian networks in reliability,” *Reliability Engineering & System Safety*, vol. 92, no. 1, pp. 92–108, 2007.

[2] J. Li and J. Shi, “Knowledge discovery from observational data for process control using causal bayesian networks,” *IIE Transactions*, vol. 39, no. 6, pp. 681–690, 2007.

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

[4] M. Reisi Gahrooei, K. Paynabar, M. Pacella, and J. Shi, “Process modeling and prediction with large number of high-dimensional variables using functional regression,” *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 2, pp. 684–696, 2020.

[5] N. Meinshausen and P. Buhlmann, “High-dimensional graphs and variable selection with the lasso,” *The Annals of Statistics*, vol. 34, no. 3, pp. 1436–1462, 2006.

[6] Y.-Y. Lee and S. Hsieh, “Classifying different emotional states by means of eeg-based functional connectivity patterns,” *PloS one*, vol. 9, no. 4, e95415, 2014.

[7] S. Fothergill, H. Mentis, P. Kohli, and S. Nowozin, “Instructing people for training gestural interactive systems,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI ’12, Austin, Texas, USA: Association for Computing Machinery, 2012, pp. 1737–1746.

[8] C. Zhang, M. Patel, S. Buthpitiya, K. Lyons, B. Harrison, and G. D. Abowd, “Driver classification based on driving behaviors,” in *Proceedings of the 21st International Conference on Intelligent User Interfaces*, ser. IUI ’16, Sonoma, California, USA: Association for Computing Machinery, 2016, pp. 80–84.

[9] R. Dahlhaus, “Graphical interaction models for multivariate time series1,” *Metrika*, vol. 51, pp. 157–172, 2000.

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

[11] X. Fang, S. Misra, G. Xue, and D. Yang, “Smart grid, the new and improved power grid: A survey,” *IEEE Communications Surveys Tutorials*, vol. 14, no. 4, pp. 944–980, 2012.

[12] P. Wang and M. Govindarasu, "Cyber-physical anomaly detection for power grid with machine learning," in *Industrial Control Systems Security and Resiliency: Practice and Theory*. Cham: Springer International Publishing, 2019, pp. 31–49.

[13] C. Zou, Z. Wang, X. Zi, and W. Jiang, "An efficient online monitoring method for high-dimensional data streams," *Technometrics*, vol. 57, no. 3, pp. 374–387, 2015.

[14] D. Qi, Z. Li, and Z. Wang, "On-line monitoring data quality of high-dimensional data streams," *Journal of Statistical Computation and Simulation*, vol. 86, no. 11, pp. 2204–2216, 2016.

[15] H. Yan, K. Paynabar, and J. Shi, "Real-time monitoring of high-dimensional functional data streams via spatio-temporal smooth sparse decomposition," *Technometrics*, vol. 60, no. 2, pp. 181–197, 2018.

[16] B. Walczak and D. Massart, "Dealing with missing data: Part i," *Chemometrics and Intelligent Laboratory Systems*, vol. 58, no. 1, pp. 15–27, 2001.

[17] G. Tomasi and R. Bro, "Parafac and missing values," *Chemometrics and Intelligent Laboratory Systems*, vol. 75, no. 2, pp. 163–180, 2005.

[18] E. Acar, D. M. Dunlavy, T. G. Kolda, and M. Mørup, "Scalable tensor factorizations for incomplete data," *Chemometrics and Intelligent Laboratory Systems*, vol. 106, no. 1, pp. 41–56, 2011.

[19] O. Pourret, P. Naim, and B. Marcot, *Bayesian Networks. A Practical Guide to Applications*. Wiley, 2008.

[20] M. Pacella, "Unsupervised classification of multichannel profile data using pca: An application to an emission control system," *Computers and Industrial Engineering*, vol. 122, pp. 161–169, 2018.

[21] H. Zhu, N. Strawn, and D. B. Dunson, "Bayesian graphical models for multivariate functional data," *Journal of Machine Learning Research*, vol. 17, no. 204, pp. 1–27, 2016.

[22] X. Qiao, S. Guo, and G. M. James, "Functional graphical models," *Journal of the American Statistical Association*, vol. 114, no. 525, pp. 211–222, 2019.

[23] B. Li and E. Solea, "A nonparametric graphical model for functional data with application to brain networks based on fmri," *Journal of the American Statistical Association*, vol. 113, no. 524, pp. 1637–1655, 2018.

[24] M. Lindquist, “Functional causal mediation analysis with an application to brain connectivity,” *Journal of the American Statistical Association*, vol. 107, no. 500, pp. 1297–1309, 2012.

[25] X. Cao, B. Sandstede, and X. Luo, “A functional data method for causal dynamic network modeling of task-related fmri,” *Frontiers in Neuroscience*, vol. 13, no. 127, pp. 1–19, 2019.

[26] H. Sun, S. Huang, and R. Jin, “Functional graphical models for manufacturing process modeling,” *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 4, pp. 1612–1621, 2017.

[27] Y. Yu and Y. Feng, “Modified cross-validation for penalized high-dimensional linear regression models,” *Journal of Computational and Graphical Statistics*, vol. 23, no. 4, pp. 1009–1027, 2014.

[28] L. Horváth and P. Kokoszka, *Inference for functional data with applications*. Springer Science & Business Media, 2012.

[29] F. Yao, H. G. Muller, and J. L. Wang, “Functional linear regression analysis for longitudinal data,” *The Annals of Statistics*, vol. 33, no. 6, pp. 2873–2903, 2005.

[30] H. Zou and T. Hastie, “Regularization and variable selection via the elastic net,” *Journal of the Royal Statistical Society, Series B*, vol. 67, pp. 301–320, 2005.

[31] R. Luo and X. Qi, “Function-on-function linear regression by signal compression,” *Journal of the American Statistical Association*, vol. 112, no. 518, pp. 690–705, 2017.

[32] J. Friedman, T. Hastie, and R. Tibshirani, “Sparse inverse covariance estimation with the graphical lasso,” *Biostatistics*, vol. 9, no. 3, pp. 432–441, 2007.

[33] P. Danaher, P. Wang, and D. M. Witten, “The joint graphical lasso for inverse covariance estimation across multiple classes,” *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, vol. 76, no. 2, pp. 373–397, 2014.

[34] T. T. Cai, H. Li, W. Liu, and J. Xie, “Joint estimation of multiple high-dimensional precision matrices,” *Statistica Sinica*, vol. 26, no. 2, pp. 445–464, 2016.

[35] D. Hallac, Y. Park, S. Boyd, and J. Leskovec, “Network inference via the time-varying graphical lasso,” in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD ’17, Halifax, NS, Canada: Association for Computing Machinery, 2017, pp. 205–213.

[36] S. Zhou, J. Lafferty, and L. Wasserman, “Time varying undirected graphs,” *Machine Learning*, vol. 80, pp. 295–319, 2010.

[37] M. Kolar, L. Song, A. Ahmed, and E. P. Xing, “Estimating time-varying networks,” *The Annals of Applied Statistics*, vol. 4, no. 1, pp. 94–123, 2010.

[38] H. Qiu, F. Han, H. Liu, and B. Caffo, “Joint estimation of multiple graphical models from high dimensional time series,” *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 78, no. 2, pp. 487–504, 2016.

[39] C. Zhang, H. Yan, S. Lee, and J. Shi, “Dynamic multivariate functional data modeling via sparse subspace learning,” *Technometrics*, vol. 0, no. 0, pp. 1–14, 2020.

[40] X. Qiao, C. Qian, G. M. James, and S. Guo, “Doubly functional graphical models in high dimensions,” *Biometrika*, vol. 107, no. 2, pp. 415–431, Feb. 2020.

[41] R. Xu, J. Wu, X. Yue, and Y. Li, *Online structural change-point detection of high-dimensional streaming data via dynamic sparse subspace learning*, 2020. arXiv: 2009.11713 [stat.ML].

[42] A. Jung, G. Hannak, and N. Goertz, “Graphical lasso based model selection for time series,” *IEEE Signal Processing Letters*, vol. 22, no. 10, pp. 1781–1785, 2015.

[43] P. Whittle, “Estimation and information in stationary time series,” *Arkiv för Matematik*, vol. 2, no. 5, pp. 423–434, 1953.

[44] P. Stoica and R. Moses, *Introduction to Spectral Analysis*. Prentice Hall, 1997, ISBN: 9780132584197.

[45] F. Bach and M. Jordan, “Learning graphical models for stationary time series,” *IEEE Transactions on Signal Processing*, vol. 52, no. 8, pp. 2189–2199, 2004.

[46] N. J. Foti, R. Nadkarni, A. Lee, and E. B. Fox, “Sparse plus low-rank graphical models of time series for functional connectivity in meg,” in *2nd KDD Workshop on Mining and Learning from Time Series*, 2016.

[47] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, “Distributed optimization and statistical learning via the alternating direction method of multipliers,” *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.

[48] D. M. Witten and R. Tibshirani, “Covariance-regularized regression and classification for high dimensional problems,” *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 71, no. 3, pp. 615–636, 2009.

[49] C. Shannon, "Communication in the presence of noise," *Proceedings of the IRE*, vol. 37, no. 1, pp. 10–21, 1949.

[50] B. Efron and R. J. Tibshirani, *An Introduction to the Bootstrap*, ser. Monographs on Statistics and Applied Probability 57. Boca Raton, Florida, USA: Chapman & Hall/CRC, 1993.

[51] A. G. Mahyari, D. M. Zoltowski, E. M. Bernat, and S. Aviyente, "A tensor decomposition-based approach for detecting dynamic network states from eeg," *IEEE Transactions on Biomedical Engineering*, vol. 64, no. 1, pp. 225–237, 2017.

[52] M. G. Preti, T. A. Bolton, and D. Van De Ville, "The dynamic functional connectome: State-of-the-art and perspectives," *NeuroImage*, vol. 160, pp. 41–54, 2017, Functional Architecture of the Brain.

[53] C. Cakan and K. Obermayer, "Biophysically grounded mean-field models of neural populations under electrical stimulation," *PLoS computational biology*, vol. 16, no. 4, e1007822, 2020.

[54] J. Liang, R. Lu, C. Zhang, and F. Wang, "Predicting seizures from electroencephalography recordings: A knowledge transfer strategy," in *2016 IEEE International Conference on Healthcare Informatics (ICHI)*, 2016, pp. 184–191.

[55] N. Watthanawisuth, A. Tuantranont, and T. Kerdcharoen, "Microclimate real-time monitoring based on zigbee sensor network," in *SENSORS, IEEE*, 2009, pp. 1814–1818.

[56] K. Bouabdellah, H. Noureddine, and S. Larbi, "Using wireless sensor networks for reliable forest fires detection," *Procedia Computer Science*, vol. 19, pp. 794–801, 2013.

[57] R. Tan, G. Xing, J. Chen, W.-Z. Song, and R. Huang, "Fusion-based volcanic earthquake detection and timing in wireless sensor networks," *ACM Trans. Sen. Netw.*, vol. 9, no. 2, 17:1–17:25, 2013.

[58] S. Siyang and T. Kerdcharoen, "Development of unmanned surface vehicle for smart water quality inspector," in *2016 13th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, 2016, pp. 1–5.

[59] N. Jin, S. Zhou, and T. Chang, "Identification of impacting factors of surface defects in hot rolling processes using multi-level regression analysis," *Transactions of NAMRI/SME*, vol. 32, pp. 557–564, 2004.

[60] K. Liu, Y. Mei, and J. Shi, “An adaptive sampling strategy for online high-dimensional process monitoring,” *Technometrics*, vol. 57, no. 3, pp. 305–319, 2015.

[61] A. Wang, X. Xian, F. Tsung, and K. Liu, “A spatial-adaptive sampling procedure for online monitoring of big data streams,” *Journal of Quality Technology*, vol. 50, no. 4, pp. 329–343, 2018.

[62] X. Xian, A. Wang, and K. Liu, “A nonparametric adaptive sampling strategy for online monitoring of big data streams,” *Technometrics*, vol. 60, no. 1, pp. 14–25, 2018.

[63] R. Y. Liu, “Control charts for multivariate processes,” *Journal of the American Statistical Association*, vol. 90, no. 432, pp. 1380–1387, 1995.

[64] H. Yan, K. Paynabar, and J. Shi, “Image-based process monitoring using low-rank tensor decomposition,” *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 1, pp. 216–227, 2015.

[65] K. Paynabar, C. Zou, and P. Qiu, “A change-point approach for phase-i analysis in multivariate profile monitoring and diagnosis,” *Technometrics*, vol. 58, no. 2, pp. 191–204, 2016.

[66] J. Liu, P. Musialski, P. Wonka, and J. Ye, “Tensor completion for estimating missing values in visual data,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 208–220, 2013.

[67] K. Xie, L. Wang, X. Wang, G. Xie, J. Wen, G. Zhang, J. Cao, and D. Zhang, “Accurate recovery of internet traffic data: A sequential tensor completion approach,” *IEEE/ACM Transactions on Networking*, vol. 26, no. 2, pp. 793–806, 2018.

[68] H. A. L. Kiers, “Towards a standardized notation and terminology in multiway analysis,” *Journal of Chemometrics*, vol. 14, no. 3, pp. 105–122, 2000.

[69] T. G. Kolda and B. W. Bader, “Tensor decompositions and applications,” *SIAM REVIEW*, vol. 51, no. 3, pp. 455–500, 2009.

[70] J. Hastad, “Tensor rank is np-complete,” *Journal of Algorithms*, vol. 11, no. 4, pp. 644–654, 1990.

[71] H. Tan, Y. Wu, B. Shen, P. J. Jin, and B. Ran, “Short-term traffic prediction based on dynamic tensor completion,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 8, pp. 2123–2133, 2016.

[72] D. Goldfarb and Z. (Qin, "Robust low-rank tensor recovery: Models and algorithms," *SIAM Journal on Matrix Analysis and Applications*, vol. 35, no. 1, pp. 225–253, 2014.

[73] K. Xie, X. Li, X. Wang, G. Xie, J. Wen, J. Cao, and D. Zhang, "Fast tensor factorization for accurate internet anomaly detection," *IEEE/ACM Transactions on Networking*, vol. 25, no. 6, pp. 3794–3807, 2017.

[74] J. Nocedal and S. J. Wright, *Numerical Optimization*, second. New York, NY, USA: Springer, 2006.

[75] M. Mardani, G. Mateos, and G. B. Giannakis, "Subspace learning and imputation for streaming big data matrices and tensors," *IEEE Transactions on Signal Processing*, vol. 63, no. 10, pp. 2663–2677, 2015.

[76] H. Kasai, "Online low-rank tensor subspace tracking from incomplete data by cp decomposition using recursive least squares," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 2519–2523.

[77] J. Fan, "Test of significance based on wavelet thresholding and neyman's truncation," *Journal of the American Statistical Association*, vol. 91, no. 434, pp. 674–688, 1996.

[78] H. Zou, T. Hastie, and R. Tibshirani, "On the degrees of freedom of the lasso," *Ann. Statist.*, vol. 35, no. 5, pp. 2173–2192, 2007.

[79] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, "Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization," *ACM Trans. Math. Softw.*, vol. 23, no. 4, pp. 550–560, 1997.

[80] S. V. Patankar, *Numerical heat transfer and fluid flow*, ser. Series on Computational Methods in Mechanics and Thermal Science. Hemisphere Publishing Corporation (CRC Press, Taylor & Francis Group), 1980.

[81] Y. Xie, J. Huang, and R. Willet, *Multiscale online tracking of manifolds.* Paper presented at the 2012 IEEE Statistical Signal Processing Workshop (SSP), Ann Arbor, MI, August 4-7, 2012.

[82] C. Gasch and D. Brown, *Data from: A field-scale sensor network data set for monitoring and modeling the spatial and temporal variation of soil moisture in a dryland agricultural field. ag data commons. https://doi.org/10.15482/usda.adc/1349683. accessed 2020-03-07*, 2017.

[83] C. R. A. Augusto, A. C. Fauth, C. E. Navia, H. Shigeouka, and K. H. Tsui, “Connection among spacecrafts and ground level observations of small solar transient events,” *Exp Astron*, vol. 31, p. 177, 2011.

[84] R. J. Radke, S. Andra, O. Al-Kofahi, and B. Roysam, “Image change detection algorithms: A systematic survey,” *IEEE Transactions on Image Processing*, vol. 14, no. 3, pp. 294–307, 2005.

[85] C. K. Gasch, D. J. Brown, C. S. Campbell, D. R. Cobos, E. S. Brooks, M. Chahal, and M. Poggio, “A field-scale sensor network data set for monitoring and modeling the spatial and temporal variation of soil water content in a dryland agricultural field,” *Water Resources Research*, vol. 53, no. 12, pp. 10 878–10 887, 2017.

[86] J. L. Hatfield and J. H. Prueger, “Temperature extremes: Effect on plant growth and development,” *Weather and Climate Extremes*, vol. 10, pp. 4–10, 2015.

VITA

Ana María Estrada Gómez's research interests lie in developing novel methodologies and efficient algorithms for the analysis of high-dimensional, incomplete, and heterogeneous data for data-driven decision making using statistical machine learning. Ana María is originally from Bogotá, Colombia, where she did her undergraduate studies and started her graduate studies. She received B.Sc. in Industrial Engineering and Mathematics from la Universidad de los Andes in 2013 and 2015, respectively, and an M.Sc. in Industrial Engineering from the same university in 2015. At the Georgia Institute of Technology, she received an M.Sc. in Statistics in 2018 and her Ph.D. in Industrial Engineering in 2021. Ana María is the recipient of the SPES+Q&P Best Student Paper Award from ASA, the QSR Best Poster Award from INFORMS, and the IISE Doctoral Colloquium Best Poster Award. At the Georgia Institute of Technology, she has been recognized with the Graduate Teaching Fellowship, granted by the Center for Teaching and Learning, and with Stewart Fellowship, awarded by the School of Industrial and Systems Engineering. She has also been appointed as a Latina Trailblazer in Engineering Fellow by Purdue's College of Engineering.