ESTIMATION OF TIME VARYING GRAPH SIGNALS WITH GRAPH ARMA PROCESSES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

EYLEM TUĞÇE GÜNEYİ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

SEPTEMBER 2021

Approval of the thesis:

ESTIMATION OF TIME VARYING GRAPH SIGNALS WITH GRAPH ARMA PROCESSES

submitted by **EYLEM TUĞÇE GÜNEYİ** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. İlkay Ulusoy
Head of Department, **Electrical and Electronics Engineering** _____

Assoc. Prof. Dr. Elif Vural
Supervisor, **Electrical and Electronics Engineering, METU** _____

Examining Committee Members:

Prof. Dr. Abdullah Aydın Alatan
Electrical and Electronics Engineering, METU _____

Assoc. Prof. Dr. Elif Vural
Electrical and Electronics Engineering, METU _____

Prof. Dr. Gözde Bozdağı Akar
Electrical and Electronics Engineering, METU _____

Assoc. Prof. Dr. Emre Özkan
Electrical and Electronics Engineering, METU _____

Assist. Prof. Dr. Aykut Koç
Electrical and Electronics Engineering, Bilkent University _____

Date: 08.09.2021

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Eylem Tuğçe Güneyi

Signature :

ABSTRACT

ESTIMATION OF TIME VARYING GRAPH SIGNALS WITH GRAPH ARMA PROCESSES

Güneyi, Eylem Tuğçe
M.S., Department of Electrical and Electronics Engineering
Supervisor: Assoc. Prof. Dr. Elif Vural

September 2021, 54 pages

Graph models provide efficient tools for analyzing data defined over irregular domains such as social networks, sensor networks, and transportation networks. Real-world graph signals are usually time-varying signals. The characterization of the joint behavior of time-varying graph signals in the time and the vertex domains has recently arisen as an interesting research problem, contrasted to the independent processing of graph signals acquired at different time instants. The concept of wide sense stationarity, which facilitates the analysis of random time processes in statistical signal processing, has been extended to graph domains for the joint time-vertex analysis of time-varying graph random processes. In this thesis, we study the problem of learning parametric joint wide sense stationary models to analyze and estimate time-varying graph signals. Since parametric models with few parameters typically require less training data than nonparametric models, they are expected to perform better in case of incomplete observations. We model time-varying graph signals as autoregressive moving average (ARMA) processes in this study. The graph ARMA process parameters are learnt from a prior coarse estimation of the joint power spectral density (JPSD) of the process that models a given set of time-varying graph signals

with missing observations. The JPSD estimation is then refined and improved based on the learnt graph ARMA process model. The estimated JPSD is finally used to recover the missing observations of the given time-varying graph signals. Experiments performed on synthetic and real data show that using ARMA graph process models to analyze time-varying graph signals yields promising results.

Keywords: Graph Signal Processing, Stationary Time-Vertex Signal Processing, Spectral Estimation, Parametric Estimation, Joint Power Spectral Density

ÖZ

GRAF ARMA SÜREÇLERİ İLE ZAMANDA DEĞİŞEN GRAF SİNYALLERİNİN TAHMİNİ

Güneyi, Eylem Tuğçe
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. Elif Vural

Eylül 2021 , 54 sayfa

Graf modelleri sosyal ağlar, sensör ve ulaşım ağları gibi düzensiz topolojilerdeki verileri analiz etmek için etkili yöntemler sunar. Gerçek graf sinyalleri genellikle zamanda değişim göstermektedir. Bu nedenle, farklı zamanlardaki graf sinyallerinin bağımsız olarak ele alınması yerine graf sinyallerinin graf-zaman bileşik uzayında analiz edilmesi son yıllarda ilgi çeken bir araştırma konusu olmuştur. Geçtiğimiz yıllarda rastgele süreçler, geniş anlamda durağanlık gibi kavramları graflara uyarlayan çalışmalar gerçekleştirilmiştir. Bu tez çalışmasında zamanda değişen graf sinyallerinin analiz ve tahmini için parametrik bileşik geniş anlamda durağan modellerin öğrenilmesi incelenmiştir. Az sayıda parametre içeren modeller, parametrik olmayan modellere göre genellikle daha az eğitim verisi gerektirdiğinden, gözlemlerin kısıtlı olduğu durumlarda daha iyi performans gösterebilmektedir. Bu çalışmada zamanda değişen graf sinyalleri otoregresif hareketli ortalama (ARMA) süreçleri ile modellenmiştir. Graf ARMA süreç parametreleri graf sinyallerinin eksik gözlemlerinden hesaplanan görece yüksek hatalı bir bileşik güç spektral yoğunluğu (JPSD) kestiriminden öğrenilmiş, ardından JPSD kestirimi öğrenilen süreç modeline dayalı olarak

iyileştirilmiştir. Bu şekilde elde edilen JPSD kestirimi graf sinyallerinin eksik gözlemlerinin hesaplanmasında kullanılmıştır. Sentetik ve gerçek verilerle gerçekleştirilen deneyler, zamanda değişen graf sinyallerinin analizinde parametrik graf süreç modellerinin kullanılmasının umut verici sonuçlar verdiğini göstermektedir.

Anahtar Kelimeler: Graf Sinyal İşleme, Durağan Zaman-Nod Sinyal İşleme, Spektral Kestirim, Parametrik Kestirim, Bileşik Spektral Güç Yoğunluğu

To my family

ACKNOWLEDGMENTS

This thesis would have been impossible without the support of many people. Before everything, I want to express how I am grateful to work with my thesis advisor Assoc. Prof. Dr. Elif Vural. I would like thank her from the bottom of my heart for her guidance, understanding, and patience.

I would like to thank Yusuf Yiğit Pilavcı. Our undergraduate and graduate studies on graph signal processing have contributed a lot to me.

I would also like to thank Abdullah Canbolat. The experiments part could not have been completed without his support.

I want to say thanks to my colleagues at ESEN for their understanding over the past few months. Thanks to their support, I was able to balance my work life and education life.

Last in order but not of importance, I would like to thank my family and Batuhan Beşcan for their unconditional love and support. I could never have completed this thesis without their help.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

AR	Autoregressive
ARMA	Autoregressive Moving Average
CNN	Convolutional Neural Networks
DFT	Discrete Fourier Transform
FIR	Finite Impulse Response Filters
GFT	Graph Fourier Transform
GSP	Graph Signal Processing
GNN	Graph Neural Networks
G-VAR	Graph Vector Autoregressive Recursions
GP-VAR	Graph Polynomial Vector Autoregressive Recursions
IGFT	Inverse Graph Fourier Transform
IIR	Infinite Impulse Response Filters
IGFT	Inverse Graph Fourier Transform
JPSD	Joint Power Spectral Density
JFT	Joint Fourier Transform
JWSS	Joint Time-Vertex Wide Sense Stationary
k-NN	k nearest neighbors
LMMSE	Linear Minimum Mean Square Error
NMSE	Normalized Mean Square Error
PSD	Power Spectral Density
SNR	Signal-to-Noise Ratio
VAR	Vector Autoregressive Process Model

CHAPTER 1

INTRODUCTION

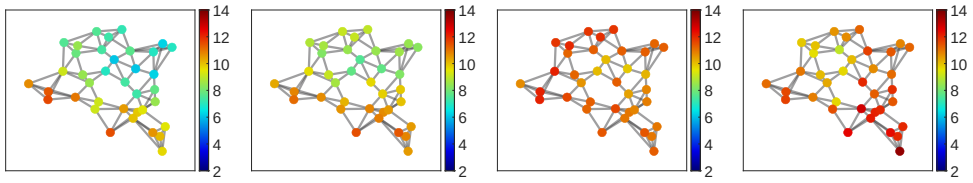
1.1 Motivation

Thanks to the recent advances in technology, a tremendous amount of data is recorded everyday. The fact that these data generally have an irregular structure makes them difficult to model. On the other hand, there is an underlying graph structure in many irregular domains such as social networks, sensor networks, and transportation networks. By courtesy of this, graphs can effectively model such complex structures and their interactions. For example, if we consider data from a sensor network, sensors can be modeled as nodes of a graph, and sensor measurements can be considered as graph signals.

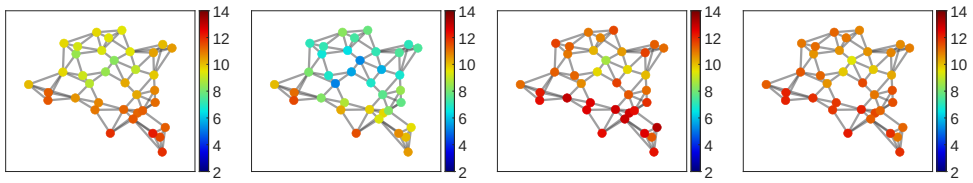
Due to the irregular structures of graphs, the application of classical signal processing concepts such as the Fourier transform, signal filtering, and frequency analysis to graph signals is not straightforward. Graph Signal Processing (GSP) aims to develop methods for analyzing graph signals by extending classical signal processing methods to graphs [2, 3]. GSP methods can be applied to different types of problems, such as customer behavior prediction [4], compression [4], classification [4], smoothing [5], denoising [6] and interpolation [6].

Data measurements may vary over time; for instance, the measurements on a sensor network or users' tendencies on a social network may have time-dependent characteristics. These time-varying measurements can be modeled as time-varying graph signals. For example, hourly temperature measurements from different weather stations in a meteorological measurement network can be modeled as time-varying graph signals. A simple illustration of this data is given in Figure 1.1. The colors of nodes

represent the temperature measurements. If we look at each graph signal separately, we can observe that the temperature signal is smoothly varying on the graph. On the other hand, if we consider the signal at a fixed node, each node at different time instants as a time series, they are smooth over time. Using the information in both the time and the vertex domains can provide better models for the time-varying graph signals. The prevalence of such real-world data sets has prompted researchers to seek ways to jointly analyze these signals in both time and vertex domains. Time vertex signal processing [7] aims to analyze time-varying graph signals jointly.



(a) Day 1, 00:00 (b) Day 1, 06:00 (c) Day 1, 12:00 (d) Day 1, 18:00

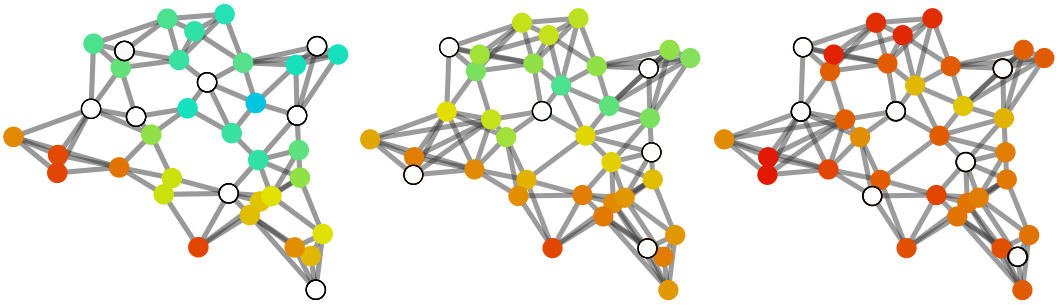


(e) Day 2, 00:00 (f) Day 2, 06:00 (g) Day 2, 12:00 (h) Day 2, 18:00

Figure 1.1: Temperature measurements from different weather stations at different times.

The inference of data on networks is a problem of interest in many applications. As mentioned above, data collections on irregular topologies such as sensor networks or social networks can typically be modeled as time-varying graph signals. Meanwhile, in many data acquisition scenarios over networks, measurements are only partially observed in the vertex domain and the time domain due to, e.g., sensor failures, partial availability of user information, such that measurements may be missing at arbitrary graph nodes at arbitrary time instants. A simple illustration of missing observations is given in Figure 1.2. In Figure 1.2, node colors represent the value of the signal. There are no measurements at the white nodes. As mentioned above, missing entries are arbitrarily distributed. The inference of the unobserved measurements from the observed ones is a problem relevant to many applications such as inpainting and

forecasting.



(a) 00:00 (b) 06:00 (c) 12:00

Figure 1.2: Time-varying graph signals with missing observations. Missing observations may occur due to sensor failure, partial availability of measurements, etc.

In this thesis, we consider the problem of estimating time-varying graph signals from partial observations without any assumptions on the observation pattern, i.e., allowing the missing observations to occur at any graph nodes and any time instants like in Figure 1.2. Our general approach to the problem consists of estimating the missing observations based on the covariance patterns of the time-vertex signal, which we obtain by analyzing the joint (time-vertex) spectrum of the process via its joint power spectral density (JPSD). We assume that data come from a parametric ARMA graph process model. We consider a joint time-vertex stationary ARMA graph process model and derive the JPSD of the process in terms of the ARMA model parameters. We formulate an optimization problem where the ARMA model parameters are optimized to fit an initial rough estimate of the JPSD obtained through a numerical estimation of the covariance matrix of the process. The problem of learning parametric graph process models often leads to non-convex optimization problems [8], which is also the case in our initial problem formulation. To put it into a more tractable form, we develop a convex relaxation for our optimization problem, which leads to an algorithm that can effectively learn ARMA graph process models from incomplete realizations. Once the ARMA graph process models are computed, the estimates of the JPSD and the covariance matrix are refined, and the missing observations are computed via linear minimum mean square error (LMMSE) estimation. We note that the proposed approach for learning ARMA graph processes based on convex relaxations largely differs from common techniques for learning ARMA processes in classical

signal processing theory, such as approximations using Durbin's method, or PSD factorization solutions via modified Yule-Walker equations [9]. The extension of such classical algorithms, typically employing classical Fourier transform or Z-transform techniques, to graph domains is not straightforward, since the computation of ARMA graph models that are coherent with the joint spectrum of graph processes involves more elaborate concepts such as the joint (time-vertex) Fourier transform.

1.2 Contributions

Our work proposes the following methodological contributions over previous literature:

- We learn ARMA graph process models based on the time-vertex joint spectrum of the process, which has not been explored in a previous work to the best of our knowledge.
- We propose an algorithm which can employ past observations $(x_{t-1}, x_{t-2}, \dots)$ of the process as well as future ones $(x_{t+1}, x_{t+2}, \dots)$ when inferring the value x_t of the process at a certain time instant t, unlike many state-of-the-art algorithms which need to treat the observations sequentially in time.
- We assume that the missing observations may occur at arbitrary time instants and arbitrary graph nodes. This feature contrasts with many state-of-the-art algorithms requiring a "training" phase where model parameters are to be learnt from observations completely available on the whole graph during a sufficiently long time interval.

1.3 Thesis Outline

In Chapter 2, first the fundamental concepts of graph signal processing are explained. Then, joint time-vertex graph signal processing is introduced. Our study aims to estimate time-varying graph signals from partial observations using ARMA graph process models. We introduce related concepts and different possible approaches to

graph signal estimation problems in Chapter 2. Finally, different approaches in the literature that can be used to estimate missing samples of graph signals are summarized.

We present our proposed method for estimating time-varying graph signals by learning graph ARMA process models in Chapter 3. Details of definitions of stationarity for time-varying graph signals and ARMA graph process model are given. These definitions provide the preliminary concepts underlying our proposed method. Then, the proposed method for estimating time-varying graph signals is described.

Experimental results evaluating the performance of the proposed method on synthetic and real time-varying graph signal data sets are presented in Chapter 4.

Finally in the concluding Chapter 5, the main implications are summarized, and notes on future work are provided.

CHAPTER 2

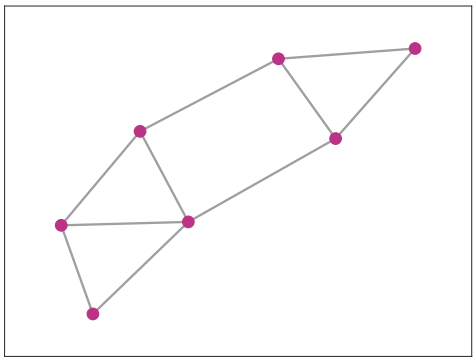
RELATED WORK

We first provide a brief overview of graph signal processing in Section 2.1. After that, joint time-vertex signal processing concepts are discussed in Section 2.2. Finally, several approaches used to estimate graph signals are reviewed in Section 2.3.

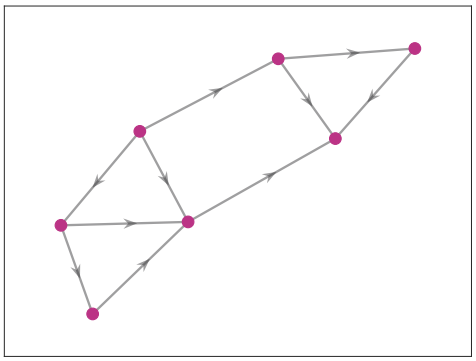
2.1 Introduction to Graph Signal Processing

Graphs represent pairwise relationships between data samples. A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consists of a set of nodes (vertices) $\mathcal{V} = \{v_1, v_2, \cdots, v_N\}$, and a set of edges $\mathcal{E}$. Edges are the connections between the nodes, and they can be either directed or undirected. Graphs with directed edges are called directed graphs. Similarly, the ones with undirected edges are called undirected graphs. A simple illustration of different types of graphs is given in Figure 2.1. While the graphs in Figure 2.1a and Figure 2.1c are undirected graphs, those in Figure 2.1b and Figure 2.1d are examples of directed graphs. A graph with edge weights is referred to as a weighted graph. The graphs in Figure 2.1a and Figure 2.1b show unweighted graphs. Figure 2.1c and Figure 2.1d are weighted graphs. Undirected weighted graphs are studied in this thesis, such as in Figure 2.1c. From now on, we consider weighted undirected graphs when we use the term "graph".

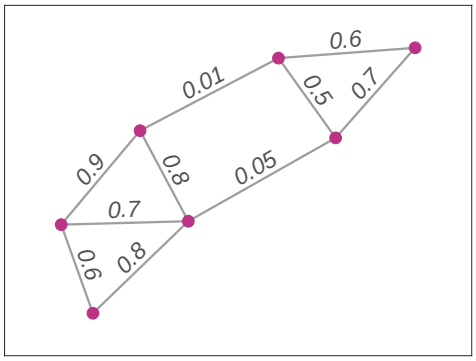
Edge weights represent the relationships between nodes in weighted graphs. A weight matrix denoted by W represents the pairwise relationships between the nodes. The weight W_{ij} of the edge which connects v_i to v_j represents how strong node v_i and node v_j are related. If there is an edge between v_i and v_j, the edge weight W_{ij} is larger than zero. Otherwise, the weight is equal to 0.



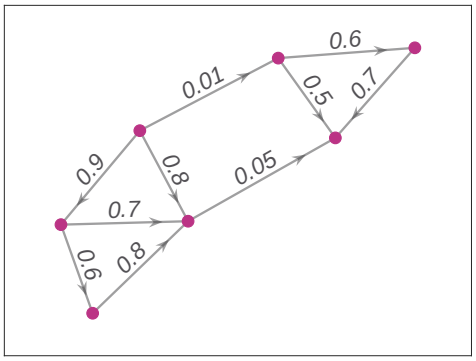
(a) Unweighted undirected graph



(b) Unweighted directed graph



(c) Weighted undirected graph



(d) Weighted directed graph

Figure 2.1: Graph Types

The choice of how the graph will be constructed can vary based on the application. A typical way to build an undirected weighted graph is to determine the edges of a node from its k nearest neighbors(k-NN) and set the edge weights using a Gaussian weighting function

$$W_{ij} = \exp(-dist(i,j)^2)/2\sigma^2, \tag{2.1}$$

where $dist(i,j)$ is the pairwise distance between node v_i, and node v_j, and σ is the kernel scale parameter.

A graph signal x is defined as a mapping from the node set to real numbers, $x : \mathcal{V} \to \mathbb{R}$. A graph signal on a graph with N nodes can be written as $x = \begin{bmatrix} x(v_1) \cdots x(v_N) \end{bmatrix}^\intercal \in \mathbb{R}^N$, and it is an N-dimensional vector. $x(v_i)$ shows the value of the graph signal on node v_i for $i = 1, 2, \cdots, N$. Let us consider a graph and a graph signal as shown in Figure 2.2, where each node represents a cat image or a dog image. Blue nodes repre-

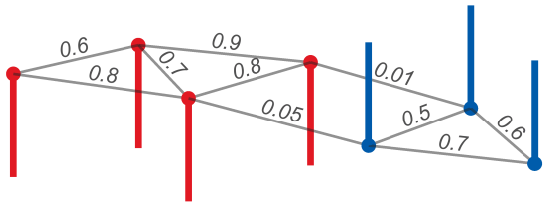


Figure 2.2: Graph Signal

sent dog images and red ones represent cat images. An example of a graph signal can be the label signal which is 1 if the node represents a cat image and -1 otherwise. If we look at the weights in the graph, we also realize that the edges between the nodes which have the same label have larger weights. Otherwise, the edge weights are smaller.

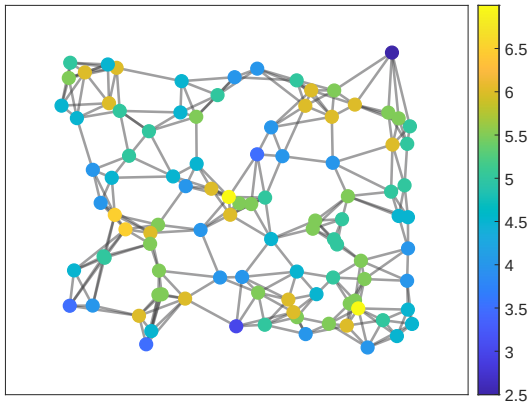


Figure 2.3: Degrees of Nodes

The degree matrix D is a diagonal matrix with diagonals $d_i = \sum_{j=1}^{N} W_{ij}$. The degree of a node d_i can be considered as a measure of the interaction of that node with its neighbors. For example, degrees of isolated or weakly connected nodes in a graph tend to have smaller values. Let us consider the sensor graph consisting of 100 nodes depicted in Figure 2.3, where the locations of the sensors are selected randomly from the unit square. A k-NN graph with $k = 5$ is constructed in this example. Note that each sensor can only be connected to another sensor if the distance between them is less than a threshold. 100 nodes 5-NN graph shown in Figure 2.3. Edge weights are set by applying a Gaussian kernel on the distance between nodes. Therefore, in this example, nodes which are closer tend to have higher weights. The colors of the nodes encode their degree values. As can be seen from Figure 2.3, weakly connected

sensors which have less interaction with other sensors tend to have smaller degree values. On the other hand, if a node has a relatively large number of edges, it tends to have a relatively large degree value. Note that the degree of a node depends not only on the number of its edges, but also on the edge weights.

The definition of the combinatorial graph Laplacian is $\mathcal{L}_\mathcal{G} = D - W$. Since the combinatorial graph Laplacian matrix is symmetric and real, it has orthonormal eigenvectors $\mathcal{L}_\mathcal{G} u = \lambda_n u_n$. The eigenvectors of the combinatorial graph Laplacian form a complete set. There is also a normalized version of the graph Laplacian, which is defined as follows:

$$\mathcal{L}_\mathcal{G} = D^{-1/2}(D - W)D^{-1/2} \tag{2.2}$$

Similar to the combinatorial graph Laplacian, the normalized Laplacian matrix is a symmetric and real matrix as well. It also has a complete set of eigenvectors. The eigenvalues of the normalized graph Laplacian are bounded as $0 = \lambda_0 \leq \lambda_1 \leq ... \leq \lambda_N \leq 2$. Depending on the application, different types of graph Laplacians can be selected. Eigenvalues of the graph Laplacian can be interpreted as analogous to the frequency concept in Fourier analysis. An eigenvector with a small eigenvalue does not change fast. Lower eigenvalues define slowly changing parts of the signal. On the contrary, eigenvalues with higher values represent fast-changing parts.

In classical signal processing, complex exponentials $e^{j\omega t}$ are the Fourier basis functions. Moreover, they are eigenfunctions of the one-dimensional Laplace operator. The graph Fourier transform (GFT) is then defined using this analogy. The eigenvectors of a graph Laplacian operator are considered as graph Fourier basis vectors and define the GFT. The eigenvalue λ_n corresponding to an eigenvector u_n can be considered as its frequency [2]. GFT of a graph signal x is defined as

$$\hat{x}(\lambda_n) = \langle x, u_n \rangle = \sum_{i=1}^{N} x(i) u_n^*(i), \tag{2.3}$$

where u_n^* is the complex conjugate of the k^{th} eigenvector, and $\hat{x}$ denotes the GFT of x. Note that, $x(v_i)$ for $i = 1, 2, \cdots, N$ will be denoted by $x(i)$ in short. $u_n(i)$ represents the i^{th} element of u_n. The GFT can be rewritten as a matrix multiplication

$$\hat{x} = {U_\mathcal{G}}^H x, \tag{2.4}$$

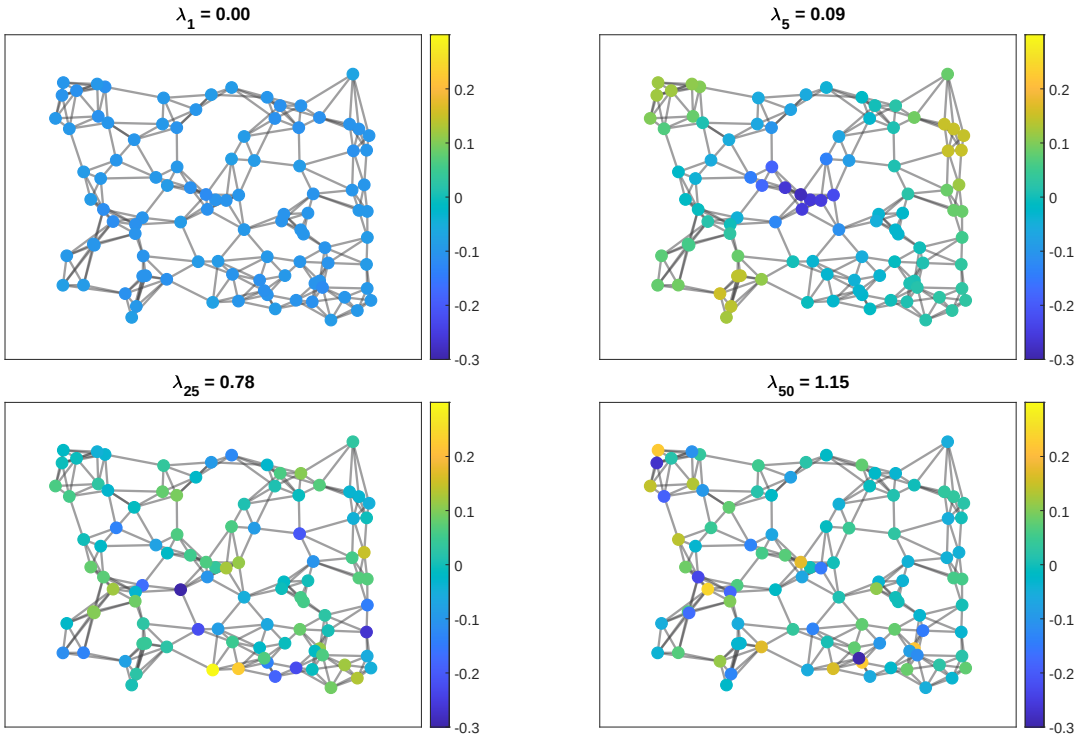


Figure 2.4: Eigenvectors of the Sensor Graph

where $U_{\mathcal{G}} = [u_1\ u_2 \cdots u_N] \in \mathbb{C}^{N \times N}$ is the matrix consisting of the graph Fourier basis vectors and $(\cdot)^H$ denotes the Hermitian (conjugate transpose) of a matrix. The definition of the inverse graph Fourier transform (IGFT) is given below [2]:

$$x(i) = \sum_{n=1}^{N} u_n(i)\hat{x}(\lambda_n) = U_{\mathcal{G}}\hat{x}. \tag{2.5}$$

In classical signal processing, filtering a signal can be defined with the convolution of the filter and the input signal in the time domain. It is also possible to filter a signal in the frequency domain, which is performed by the multiplication of the frequency responses of the filter and the input signal. Since graphs are irregular structures, it is not easy to define signal filtering in the vertex domain. Therefore, it is defined in the spectral domain as the multiplication of the graph Fourier transforms of the input graph signal and the filter as in classical signal processing. Graph filters $g(\lambda)$ are continuous kernels $g : \mathbb{R}_+ \to \mathbb{R}$, so that the value $g(\lambda)$ of the kernel at the graph frequency λ determines how much that the graph frequency is attenuated or amplified by the graph filter. The graph filtering in the spectral domain can be written as

$$\hat{y}(n) = g(\lambda_n)\ \hat{x}(n), \tag{2.6}$$

where $\hat{y}$ is the GFT of the output signal. Equivalently

$$\hat{y} = g(\Lambda)\ \hat{x} = g(\Lambda){U_{\mathcal{G}}}^{H} x, \tag{2.7}$$

where $g(\Lambda)$ is a diagonal matrix with diagonals $g(\lambda_n)$. The output signal in the vertex domain can be found by taking the inverse GFT of the $\hat{y}$.

$$y = U_{\mathcal{G}}\hat{y} \tag{2.8}$$

Consequently, filtering a graph signal in the vertex domain can be defined as follows

$$y = U_{\mathcal{G}} g(\Lambda){U_{\mathcal{G}}}^{H} x. \tag{2.9}$$

The graph filter $g(\mathcal{L}_{\mathcal{G}})$ in the vertex domain can be given [2]

$$g(\mathcal{L}_{\mathcal{G}}) = U_{\mathcal{G}} g(\Lambda){U_{\mathcal{G}}}^{H}. \tag{2.10}$$

The graph filter in vertex domain is a function of the graph Laplacian matrix. Then, filtering a graph signal in the vertex domain is equivalent to multiplying the input signal with a matrix which is a function of the graph Laplacian.

$$y = g(\mathcal{L}_{\mathcal{G}})x \tag{2.11}$$

Graph kernel can simply be selected as a polynomial function of the graph Laplacian. Then, the polynomial graph filters can be written as follows

$$g(\mathcal{L}_{\mathcal{G}}) = \sum_{k=0}^{K} a_k \mathcal{L}_{\mathcal{G}}^{k}, \tag{2.12}$$

where a_k is polynomial graph filter coefficients and $\mathcal{L}_{\mathcal{G}}^{k}$ is the k^{th} power of the graph Laplacian.

2.2 Joint Time-Vertex Signal Processing

In various settings, graph signals may have a time-varying nature; for instance, temperature measurements acquired in different weather stations in a network at different time instants is a time-varying graph signal. Such time-vertex graph signals can be represented as $X = \begin{bmatrix} x_1 & x_2 & \cdots & x_T \end{bmatrix} \in \mathbb{R}^{N \times T}$. Here each column $x_t \in \mathbb{R}^N$ of the X matrix is a graph signal, while each row of X is a time signal.

The joint time-vertex frequency behavior of time-varying graph signals can be analyzed using the joint Fourier transform [1]. If the matrix X is regarded as a collection of N time signals lying in its rows, the discrete Fourier transform (DFT) of these signals can be obtained as

$$DFT\{X\} = XU_T{}^*, \tag{2.13}$$

where U_T is the normalized DFT matrix given by

$$U_T(t,\tau) = \frac{e^{j\omega_\tau t}}{\sqrt{T}}, \quad \omega_\tau = \frac{2\pi(\tau-1)}{T} \text{ for } t,\tau = 1,\cdots,T \tag{2.14}$$

and $(\cdot)^*$ denotes the complex conjugate of a matrix. On the other hand, if X is regarded as a collection of T graph signals lying in its columns, the GFT of these signals can be computed as

$$GFT\{X\} = U_{\mathcal{G}}{}^H X. \tag{2.15}$$

Consider the 2D Fourier transform in classical signal processing, which takes the Fourier transform first along one dimension and then along the other one. The joint Fourier transform for time-vertex signals is defined similarly to this.The joint Fourier transform (JFT) $\widehat{X}$ of a time-vertex signal X takes the GFT along the node dimension and the DFT along the time dimension [1]

$$\widehat{X} = JFT\{X\} = U_{\mathcal{G}}{}^H X U_T^*. \tag{2.16}$$

Denoting the vectorized form of a time-vertex signal $X \in \mathbb{R}^{N\times T}$ as $\bar{x} \in \mathbb{R}^{NT}$, the joint Fourier transform can also be expressed as [1]

$$\hat{\bar{x}} = U_J{}^H \bar{x}, \tag{2.17}$$

where $U_J = U_T \otimes U_{\mathcal{G}}$ is the Kronecker product of U_T and $U_{\mathcal{G}}$. The filtering of time-vertex signals can similarly be defined in the joint spectral domain through the use of the joint Laplacian operator $\mathcal{L}_J = \mathcal{L}_T \otimes I_N + I_T \otimes \mathcal{L}_J$, where $\mathcal{L}_T$ represents the Laplacian of a cyclic graph. Its eigenvector matrix is represented as U_T [1], and $I_N \in \mathbb{R}^{N\times N}$ and $I_T \in \mathbb{R}^{T\times T}$ are identity matrices. Then for a joint graph filter $g(\mathcal{L}_J)$, the relation between the input and output time-vertex signals X and Y can be represented in terms of their vectorized forms as

$$\bar{y} = g(\mathcal{L}_J)\bar{x} = U_J\, g(\Lambda,\Omega)\, U_J{}^H \bar{x}, \tag{2.18}$$

where $\Omega \in \mathbb{R}^{T \times T}$ is a diagonal matrix with diagonals ω_τ for $\tau = 1, 2, \cdots, T$. Here, $g(\Lambda, \Omega)$ is also a diagonal matrix containing on its diagonals the vectorized form of the matrix [1]

$$\begin{bmatrix} g(\lambda_1, \omega_1) & \cdots & g(\lambda_1, \omega_T) \\ \vdots & \ddots & \vdots \\ g(\lambda_N, \omega_1) & \cdots & g(\lambda_N, \omega_T) \end{bmatrix}, \tag{2.19}$$

where $g(\lambda_n, \omega_\tau)$ is the joint filter kernel that represents the desired filter response at graph frequency λ_n and time frequency ω_τ.

2.3 Estimation of Missing Samples of Graph Signals

In recent years, a major effort in graph signal processing has been the extension of classical signal processing concepts to graphs in order to analyze and estimate graph signals [2, 3, 10]. Inference problems for graph signals have been studied in various previous works. Estimation of graph signals has been studied in the context of semi-supervised learning [11, 12, 13, 14, 15].

Machine learning problems can be traditionally categorized as supervised, semi-supervised, and unsupervised problems [16]. If training data consist of both input vectors and their corresponding output values, it is a supervised learning problem. Supervised learning algorithms take the input vector and aim to learn a model that computes the output value that best fits the input vector. If the output values can only belong to a finite set of discrete values, the problem is called a classification problem. On the other hand, if the purpose of the problem is to assign the input vector to a continuous value, it is called a regression problem. Determining whether an image is a dog image or a cat image can be an example of a classification problem. An example of a regression problem is to estimate missing sensor measurements due to a temporary failure in the sensor. Machine learning problems where the training data consists of only a set of input vectors without output values are called unsupervised learning problems. Their aim is to learn the underlying structure of the input vectors. For instance, clustering and density estimation can be considered as examples of unsupervised learning. Semi-supervised learning uses a small amount of labeled data and a large amount of unlabeled data to utilize the advantages of unsupervised and

supervised learning methods.

If each vertex of the graph is considered as a data point in a semi-supervised learning problem, the outputs of some of the vertices are available, and the others are not available. Graph-based semi-supervised learning algorithms generally construct a graph from labeled and unlabeled samples. Then, information in the labeled samples is tried to be propagated over the graph. The most common assumption of graph-based semi-supervised learning algorithms to propagate information in labeled data is that the graph signal is smooth over the graph. The smoothness of a graph signal can be defined using the graph Laplacian quadratic form

$$s(x) = \sum_{i,j \in \mathcal{E}} W_{ij}(x(j) - x(i))^2 = x^\intercal \mathcal{L}_\mathcal{G} x, \tag{2.20}$$

where x is the graph signal, and $\mathcal{L}_\mathcal{G}$ is the graph Laplacian matrix [17]. This is also known as the total variation of a graph signal [2]. If a graph signal x is smooth over the graph, in other words, if the signal does not change rapidly between neighbors, its total variation on the graph is expected to be low. The Gaussian random fields and harmonic function methods [12] fix the values of the labeled data and try to find other labels such that they are smooth. The optimization problem of this method can be considered as follows:

$$\min_x \mu \sum_{i \in S} (x(i) - y(i)) + x^\intercal \mathcal{L}_\mathcal{G} x, \tag{2.21}$$

where S is a set which consists of indices of labeled data and $y(i)$ for $i \in S$ represents available labeled samples of the graph signal x. Here, $\mu \to \infty$ space. The optimization problem in Tikhonov regularization algorithm [11] is defined as follows:

$$\min_x \sum_{i \in S} (x(i) - y(i))^2 + \gamma x^\intercal \mathcal{L}_\mathcal{G} x, \tag{2.22}$$

where γ is a positive constant. Here, the Tikhonov regularization algorithm aims to predict the signal that is close to the labeled data and smooth over the graph, rather than directly fixing the values of the labeled data.

In classical signal processing, it is common to model signals to be produced by filtering other signals. Finite impulse response filters (FIR) and infinite impulse response

filters (IIR) are the two types of filters. If the impulse response of a filter has infinite time spread, it is an IIR filter. Otherwise, it is an FIR filter. Similar to classical signal processing, filters can be used to model a graph signal in applications such as signal interference and estimation. For this purpose, both finite impulse response filters (FIR) [4, 18, 19] and infinite impulse response filters [20] (IIR) in classical signal processing have been extended to analyze graph signals. If graph filters are used to model and analyze time-varying graph signals, they have to analyze graph signals at different time instants separately. Therefore, interactions in the time dimension cannot be modeled. This makes graph filters inadequate for modeling time-varying signals.

If a signal is band-limited, it can be fully recovered from its samples in classical signal processing. In recent years, sampling theory has been extended to graphs to recover band-limited graph signals from their samples, which is called as graph sampling theory. Graph sampling theory is first introduced in [21]. A graph signal x is band-limited if its spectral support is between $[0, \omega]$. Pesenson shows that graph signals which contain low frequencies can be perfectly reconstructed from their samples. On the other hand, high-frequency components can be uniquely determined only by their values at all vertices [21]. Narang et al. use graph sampling theory to estimate missing values of band-limited graph signals [22]. They aim to estimate band-limited graph signals that can be obtained by reconstructing using available samples [23, 24, 25].

Sparse representations are efficient tools to represent signals [26]. A signal $x \in \mathbb{R}^M$ is sparse if it only k nonzero elements where $k \ll M$ [26]. Sparse signal models aim to represent a given signal x as a linear combination of a few signals in a set of signals called a dictionary $\mathcal{D} \in \mathbb{R}^{N \times M}$. The elements $d_l \in \mathbb{R}^N$ of the dictionary are called dictionary atoms where $l = 1, \ldots, M$. In order to obtain sparse representations, overcomplete dictionaries (i.e., $M > N$) are used. Dictionaries can be selected as prespecified transform matrices such as wavelet transform [27] or can be learned from data [28, 29, 30, 31]. These dictionaries can be usedd to model and analyze signals. Therefore, both transform-based methods and dictionary learning methods have also been studied in graph signal processing to model graph signals. Hammond et al. proposed spectral graph wavelet transform for weighted graphs [32]. This graph wavelet transform can be use to model the graph signals.

Neural networks gained massive attention in the past few years. Similarly to the above studies, neural networks have also extensions to graph domains to model the graph signals. Gori et al. introduced graph neural networks (GNN) in [33], and then detailed them in [34]. Firstly in GNNs, a graph and the initial representations of nodes are chosen. Then, GNNs aim to learn new representations for each node which also encodes the neighborhood information. However, the learning process in GNNs is computationally expensive. Therefore, the studies in [35, 36] focused on overcoming this problem. The success of convolutional neural networks (CNN) inspired researchers to work on convolutional graph neural networks. Convolutional graph neural networks can be divided into two groups: the spectral-based approaches [37, 38, 39, 40] and the spatial-based approaches [41, 42, 43]. The learned representations with graph neural networks can be used to model and analyze the graph signals.

The stationarity of a random process in the classical signal processing means that its statistical properties do not change over time. Although the signal itself changes over time, the way it changes is constant over time. Stationary processes are important because they can be modeled and analyzed more easily. Due to the irregular structure of graphs, the classical definition of stationarity does not apply to graphs. The irregular structure of graphs makes it difficult to define stationarity. Various studies have been conducted on the generalization of wide sense stationary processes to graphs in the literature [8, 1, 44, 45]. Girault et al. describe wide stationary processes via the isometric shift operator [44, 46]. Perraudin et al. use the graph localization operator to define wide sense stationarity in graph processes [45]. Marques et al. have defined wide stationary graph processes over graph shift operators [8]. Although these methods define stationarity in graph processes, they ignore the time dimension of the signals. Joint (time-vertex) stationarity has been defined for statistical analysis of time-varying graph signals in the following studies [1, 47].

Marques et al. use autoregressive moving average graph filters to model graph signals [8]. They firstly calculate a rough power spectral density and correct this power spectral density using the ARMA model assumption. They assume that the observations come from a wide sense stationary graph process. This thesis is based on motivation that joint (time-vertex) wide sense stationary process models should be able to

model most graph signals of time-varying nature more accurately than graph (vertex) wide sense stationary assumption. Therefore, in this thesis, we propose a method which aims to learn parametric ARMA models under the joint wide sense stationary models.

CHAPTER 3

PROPOSED METHOD

This chapter describes the proposed method for estimating time-varying graph signals by learning graph ARMA process models. First, detailed definitions of stationarity for time-varying graph signals and ARMA graph process models are explained. Then, the proposed method for estimating time-varying graph signals is described.

3.1 Preliminaries

3.1.1 Graph Wide Sense Stationary Processes

We first describe how the concept of wide sense stationarity has been extended to graph domains in recent works. Let $x \in \mathbb{R}^N$ be a random vector representing a graph signal. Then x is a graph wide sense stationary process if and only if the following conditions are satisfied [45] :

- x has a constant mean $E[x] = c\,1_N$
- The covariance matrix of x can be written as a graph filter $\Sigma_x = h(\mathcal{L}_\mathcal{G})$,

where c and 1_N respectively denote a real constant and an N-dimensional vector consisting of 1's. We notice that the first condition is a straightforward extension of the concept of stationarity in the mean to graph processes. Similarly, the second condition aims to generalize the condition of covariance stationarity to graph domains as follows: In classical signal processing, covariance stationarity refers to the invariance of the covariance between two process samples to time shifts. However, this definition is not straightforward to extend to graph domains, as vertex shifts are challenging

to define on graphs due to the irregularity of graph domains. The second condition above circumvents this difficulty by formulating the "covariance stationarity" through graph filters: The covariance between the process samples at two graph nodes must be identifiable through how strong the response of some graph filter centered in the first node is in the other node. The characteristics of this graph filter determine the covariance pattern of the graph process then. We also notice that if x is a graph wide sense stationary process, then due to the second condition above, its covariance matrix Σ_x is jointly diagonalizable with the graph Laplacian as it can be decomposed as $\Sigma_x = U_{\mathcal{G}}\, h(\Lambda)\, U_{\mathcal{G}}{}^{\intercal}$. Here, an important property is that the eigenvalues $h(\lambda_n)$ in the diagonals of $h(\Lambda)$ give the power spectral density (PSD) of the graph process x [45]. This property is indeed in line with its counterpart in classical signal processing: Covariance matrices of stationary time processes obtained by filtering white noise are diagonalizable with the DFT matrix, where the eigenvalues correspond to the PSD of the process.

3.1.2 Joint Time-Vertex Wide Sense Stationary Processes

Following the above definition of stationarity for graph processes, the concept of stationarity has next been extended to time-vertex processes in later studies. Let $X \in \mathbb{R}^{N\times T}$ be a random time-vertex process, with vectorized form $\bar{x} \in \mathbb{R}^{N\times T}$. If X satisfies the following conditions, it is called a joint time-vertex wide sense stationary (JWSS) process [1]:

- $\bar{x}$ has a constant mean $E[\bar{x}] = c\, 1_{NT}$.
- The covariance matrix $\Sigma_{\bar{x}}$ of the process $\bar{x}$ is a joint time-vertex filter

$$\Sigma_{\bar{x}} = h(\mathcal{L}_{\mathcal{G}}, \mathcal{L}_T) = U_J\, h(\Lambda, \Omega)\, {U_J}^H, \tag{3.1}$$

where c and 1_{NT} respectively denote a real constant and an NT-dimensional vector consisting of 1's.

Hence, if $\bar{x}$ is a JWSS process, the covariance matrix $\Sigma_{\bar{x}}$ of the process can be jointly diagonalized with the joint Laplacian $\mathcal{L}_J$. Moreover, the eigenvalues $h(\lambda_n, \omega_\tau)$ of

$\Sigma_{\bar{x}}$ give the joint power spectral density (JPSD) of the time-vertex process. We can easily observe that the above conditions required for time-vertex stationarity extend the conditions in Section 3.1.1 to time-vertex processes in a natural way.

Note that there is a relation between JWSS definitions and classical stationarity definitions [1]. If the columns of X are the realizations of a N dimensional time wide-sense stationary process and its rows are the realizations of a T dimensional GWSS process, then the joint process X is a JWSS process [1]. Moreover, if a time-vertex process X is JWSS, it is also a multivariate time wide-sense stationary and multivariate graph wide-sense stationary process [1]. More detailed information about the relation of JWSS definition to classical stationarity definitions can be found in [1].

JWSS definition can be used to model time-varying graph signals. Loukas et al. assume that data come from a JWSS process, and they estimate the JPSD from the available data. They estimated JPSD as the average of JFT of realizations X^r

$$\tilde{h}(\lambda_n, \omega_\tau) = \sum_{r=1}^{R} \frac{\widehat{X^r}(n, \tau)}{R}, \tag{3.2}$$

where $\widehat{X^r}$ is the JFT of the r^{th} realization of X, R is the number of realizations, and $\tilde{h}(\lambda_n, \omega_\tau)$ is the estimate of JPSD of the time-vertex process X. Note that this estimation method is based on the Wiener-Khinchin theorem [48]. After estimating JPSD, an estimate of covariance matrix $\tilde{\Sigma}_X$ can be found using the relation in (3.1). Loukas et al. estimate time-varying graph signals using the covariance matrix $\tilde{\Sigma}_X$ with linear minimum mean square error (LMMSE) approach [1]. It is a nonparametric algorithm, which uses the JWSS assumption. Therefore, we compare the proposed method with this algorithm in Chapter 4. We call this method [1] "JWSS" in Chapter 4.

3.1.3 Autoregressive Moving Average Graph Processes

The concept of autoregressive moving average (ARMA) filters widely used in classical signal processing has been extended to graph domains in several previous works [49, 50]. A JWSS time-vertex process X can be modeled as an ARMA graph process if it is generated by filtering a zero-mean white process w_t with an ARMA graph

filter, where $w_t \sim \mathcal{N}(0, I_N)$ are independent for different t. The graph process x_t at each time t is then related to the past values x_{t-p} of the process and the input white process as

$$x_t = -\sum_{p=1}^{P} a_p(\mathcal{L}_\mathcal{G}) x_{t-p} + \sum_{q=0}^{Q} b_q(\mathcal{L}_\mathcal{G}) w_{t-q}, \tag{3.3}$$

where $a_p(\mathcal{L}_\mathcal{G})$, $b_q(\mathcal{L}_\mathcal{G})$ are graph filters. If $a_p(\mathcal{L}_\mathcal{G})$ and $b_q(\mathcal{L}_\mathcal{G})$ are polynomial graph filters, then the ARMA graph process model is of the form

$$x_t = -\sum_{p=1}^{P}\sum_{k=0}^{K} a_{pk}\mathcal{L}_\mathcal{G}^k\, x_{t-p} + \sum_{q=0}^{Q}\sum_{m=0}^{M} b_{qm}\mathcal{L}_\mathcal{G}^m\, w_{t-q}, \tag{3.4}$$

where a_{pk} and b_{qm} are ARMA graph filter coefficients [49, 50]. The relation in (3.4) can be intuitively interpreted in the way that the process value x_t at each time instant t is obtained as a result of the diffusion of the previous process values x_{t-p} in preceding time instants over the graph, through some polynomial graph filters. At the same time, at each time instant t, a novel external "noise" component is introduced to the system, modeled through the white process w_t, which contributes to x_t by passing through a similar diffusion operation. Polynomial graph filters are typically useful for modeling many natural phenomena over networks. For instance, in an application where the graph process models the spread of an epidemic in a population, the first term in the RHS of (3.4) with x_{t-p}'s would represent the transmission of the infection among the population members, while the second term with w_{t-q}'s would account for the infection sources external to the population. Taking the JFT of the process X, it can be shown that the time-vertex spectral domain representation of the graph filter in (3.4) is given by [49, 50]

$$H(\lambda_n, \omega_\tau) = \frac{\sum_{q=0}^{Q}\sum_{m=0}^{M} b_{qm}\, \lambda_n^M\, e^{-j\omega_\tau q}}{1 + \sum_{p=1}^{P}\sum_{k=0}^{K} a_{pk}\, \lambda_n^K\, e^{-j\omega_\tau p}}. \tag{3.5}$$

Isufi et al. [50] use parametric models to represent time-varying graph signals. They call the model in (3.3) graph vector ARMA (G-VARMA) model. While estimating the model parameters , they fit a univariate temporal ARMA model, for each graph frequency. They use the well-studied ARMA estimation methods to fit the model parameters. They also use a graph polynomial vector AR model given in (3.6) to represent time-varying graph signals. Model parameters a_{pk} are estimated by minimizing the minimum squared error [50]. After model parameters are found, time-varying

graph signals are estimated using AR recursions. We note that both of methods do not use the joint spectrum while estimating model parameters. On the other hand, they aim to model graph and time components together. Therefore, we compare our proposed method with these methods in Chapter 4. We drop the MA part in G-VARMA model and we compare our method with it. It is called "G-VAR" in Chapter 4. The proposed method is also compared with "GP-VAR" algorithm in Chapter 4.

$$x_t = -\sum_{p=1}^{P}\sum_{k=0}^{K} a_{pk}\mathcal{L}_{\mathcal{G}}^{k}\, x_{t-p} + w_t, \tag{3.6}$$

3.2 Proposed Method for Learning Parametric Time-Vertex Processes

In this work, we consider a setting where R realizations $\{X^r\}_{r=1}^{R}$ of a time-vertex process X are available. Each realization $X^r \in \mathbb{R}^{N\times T}$ is assumed to be only partially observed, such that the value X_{it}^r of the realization is known only at some of the graph nodes $i \in \{1,\dots,N\}$ for some of the time instances $t \in \{1,\dots,T\}$. Let S^r denote the index set of node-time pairs for which the realization X^r is observed.

$$S^r = \{(i,t) \mid X_{it}^r \text{ is observed}\}$$

Also, let $\bar{S}^r$ denote the complement of S^r, i.e., the index set of node-time pairs for which the observation of the realization X^r is missing. Our aim in this work is to estimate the missing observations $\{X_{it}^r|\ (i,t) \in \bar{S}^r,\ r = 1,\dots,R\}$, given the available process observations $\{X_{it}^r|\ (i,t) \in S^r,\ r = 1,\dots,R\}$.

Our approach is based on first obtaining an initial rough estimate $\tilde{\Sigma}_{\bar{x}}$ of the covariance matrix $\Sigma_{\bar{x}}$ (3.1) from the available process observations, which yields a rough estimate $\tilde{h}(\lambda_n,\omega_\tau)$ of the joint power spectral density. We then learn the ARMA model parameters by fitting the joint time-vertex spectrum (3.5) of the ARMA filter to the initial estimate $\tilde{h}(\lambda_n,\omega_\tau)$ of the JPSD. Note that, the relation between the joint time-vertex spectrum and the JPSD is given in (3.11). An improved estimate of the covariance matrix $\Sigma_{\bar{x}}$ is finally obtained from the learned ARMA model, from which we infer the initially unknown process values. We discuss these steps in detail in the following sections.

3.2.1 Initial Estimation of the JPSD

We first describe the initial estimation of the JPSD, which will be used in the computation of ARMA process models in Section 3.2.2. The covariance matrix $\Sigma_{\bar{x}} \in \mathbb{R}^{NT \times NT}$ of a time-vertex process $X \in \mathbb{R}^{N \times T}$ is given by

$$\Sigma_{\bar{x}} = E[\bar{x}\bar{x}^{\intercal}] = \begin{bmatrix} \Sigma_{1,1} & \Sigma_{1,2} & \cdots & \Sigma_{1,T} \\ \Sigma_{2,1} & \Sigma_{2,2} & \cdots & \Sigma_{2,T} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{T,1} & \Sigma_{T,2} & \cdots & \Sigma_{T,T} \end{bmatrix},$$

where we assume that X is zero-mean for notational convenience. Here $\bar{x} \in \mathbb{R}^{NT}$ represents the vectorized form of the process X and $\Sigma_{t,u} = E[x_t\, x_u{}^{\intercal}]$ stands for the covariance matrix of the values of the process at time instants t and u. When X is a JWSS process, the covariance matrix $\Sigma_{\bar{x}}$ is known to have the following special property: Each covariance matrix $\Sigma_{t,u}$ is a graph filter $\Sigma_{t,u} = g_{t,u}(\mathcal{L}_{\mathcal{G}})$, which depends only on the time difference $t - u$. This leads to a block-Toeplitz structure in $\Sigma_{\bar{x}}$ [1]. The observation that any graph filter $g(\mathcal{L}_{\mathcal{G}})$ needs to be symmetric, as well as the overall covariance matrix $\Sigma_{\bar{x}}$, leads to the equality $\Sigma_{t,u} = \Sigma_{u,t}$. Hence, the estimation of $\Sigma_{\bar{x}}$ boils down to the estimation of the matrices $\Sigma_{\Delta} \in \mathbb{R}^{N \times N}$, for $\Delta = 0, 1, \ldots, T-1$, where $\Sigma_{t,u} = \Sigma_{\Delta}$ with $\Delta = |t - u|$. We obtain an estimate $\tilde{\Sigma}_{\Delta}$ of each Σ_{Δ} by estimating its entries $[\tilde{\Sigma}_{\Delta}]_{ij}$ from the sample covariance of the observed process value pairs

$$\{(X_{it}^r, X_{ju}^r) |\ \ (i,t),(j,u) \in S^r,\ |t-u| = \Delta,\ \ r = 1, \ldots, R\}.$$

Once we compute the estimate $\tilde{\Sigma}_{\bar{x}}$ of the covariance matrix $\Sigma_{\bar{x}}$ in this way, using the relation in (3.1), we obtain the estimate $\tilde{h}(\lambda_n, \omega_\tau)$ of the JPSD simply by extracting the diagonal entries of the matrix

$$\tilde{h}(\Lambda, \Omega) = {U_J}^H \tilde{\Sigma}_{\bar{x}} U_J. \tag{3.7}$$

3.2.2 Learning ARMA Graph Processes

In order to learn an ARMA process model coherent with the initially estimated JPSD, we first rewrite the filter spectrum in (3.5) as

$$H(\lambda_n, \omega_\tau) = \frac{b^H u_{n,\tau}}{1 + a^H v_{n,\tau}}, \tag{3.8}$$

where the vectors $a \in \mathbb{R}^{P(K+1)\times 1}$ and $b \in \mathbb{R}^{(Q+1)(M+1)\times 1}$ respectively consist of the filter coefficients a_{pk} and b_{qm} as

$$\begin{aligned} a &= [a_{10}\ a_{11}\ \cdots a_{pk}\ \cdots a_{PK}]^H \\ b &= [b_{00}\ b_{01}\ \cdots b_{qm}\ \cdots b_{QM}]^H. \end{aligned} \tag{3.9}$$

The vectors $v_{n,\tau} \in \mathbb{C}^{P(K+1)\times 1}$ and $u_{n,\tau} \in \mathbb{C}^{(Q+1)(M+1)\times 1}$ consist of the coefficients appearing in the joint Fourier transform expression

$$\begin{aligned} v_{n,\tau} &= [\lambda_n^0 e^{j\omega_\tau 1}\ \lambda_n^1 e^{j\omega_\tau 1}\ \cdots\ \lambda_n^K e^{j\omega_\tau p}\ \cdots\ \lambda_n^K e^{j\omega_\tau P}]^H \\ u_{n,\tau} &= [\lambda_n^0 e^{j\omega_\tau 0}\ \lambda_n^1 e^{j\omega_\tau 0}\ \cdots\ \lambda_n^Q e^{j\omega_\tau q}\ \cdots\ \lambda_n^M e^{j\omega_\tau Q}]^H. \end{aligned} \tag{3.10}$$

Similarly to the filtering of white noise processes in classical signal processing, the JPSD $h(\lambda_n, \omega_\tau)$ of the process is related to the filter spectrum in (3.5) as [1]

$$h(\lambda_n, \omega_\tau) = |H(\lambda_n, \omega_\tau)|^2 = \left| \frac{b^H u_{n,\tau}}{1 + a^H v_{n,\tau}} \right|^2. \tag{3.11}$$

Hence, we can formulate the estimation of the ARMA model parameters from the initially estimated JPSD $\tilde{h}(\lambda_n, \omega_\tau)$ as

$$\min_{a,b} \sum_{n}^{N} \sum_{\tau}^{T} \left| \left| \frac{b^H u_{n,\tau}}{1 + a^H v_{n,\tau}} \right|^2 - \tilde{h}(\lambda_n, \omega_\tau) \right|^2. \tag{3.12}$$

The optimization problem in (3.12) is non-convex and difficult to solve due to the fourth-order dependence of the objective function on the model parameters a and b. In order to develop a convex relaxation of this problem, we first reformulate the relation in (3.4) as

$$\sum_{p=0}^{P} \sum_{k=0}^{K} a_{pk} \mathcal{L}_{\mathcal{G}}^k x_{t-p} = \sum_{q=0}^{Q} \sum_{m=0}^{M} b_{qm} \mathcal{L}_{\mathcal{G}}^m w_{t-q}, \tag{3.13}$$

where we set $a_{00} = 1$ and $a_{0k} = 0$ for $k = 1, 2, \cdots, K$. This new formulation has the advantage that the joint frequency response of the resulting graph filter has the relatively simple form

$$H(\lambda_n, \omega_\tau) = \frac{b^H u_{n,\tau}}{a^H v_{n,\tau}}, \tag{3.14}$$

where the vectors $a \in \mathbb{R}^{(P+1)(K+1)\times 1}$ and $v_{n,\tau} \in \mathbb{C}^{(P+1)(K+1)\times 1}$ are redefined as

$$\begin{aligned} a &= [a_{00}\ a_{01}\ \cdots a_{pk}\ \cdots a_{PK}]^H \\ v_{n,\tau} &= [\lambda_n^0 e^{j\omega_\tau 0}\ \lambda_n^1 e^{j\omega_\tau 0}\ \cdots\ \lambda_n^k e^{j\omega_\tau p}\ \cdots\ \lambda_n^K e^{j\omega_\tau P}]^H \end{aligned} \tag{3.15}$$

and the vectors b and $u_{n,\tau}$ remain as in (3.9) and (3.10). The objective function in (3.12) then becomes

$$\begin{aligned} &\sum_n^N \sum_\tau^T \left| \left| \frac{b^H u_{n,\tau}}{a^H v_{n,\tau}} \right|^2 - \tilde{h}(\lambda_n, \omega_\tau) \right|^2 \\ =&\sum_n^N \sum_\tau^T \left| \frac{u_{n,\tau}^H b b^H u_{n,\tau}}{v_{n,\tau}^H a a^H v_{n,\tau}} - \tilde{h}(\lambda_n, \omega_\tau) \right|^2 . \end{aligned} \tag{3.16}$$

The problem in (3.16) is difficult to handle because of the rational function term it involves. Therefore, instead of solving the original problem in (3.16) we replace it with the modified problem in (3.17). Although the modified problem in (3.17) is expected to have a different solution than that of the one in (3.16), it becomes much more tractable as the rational function is replaced with polynomial terms.

$$\sum_n^N \sum_\tau^T \left| u_{n,\tau}^H b b^H u_{n,\tau} - v_{n,\tau}^H a a^H v_{n,\tau}\, \tilde{h}(\lambda_n, \omega_\tau) \right|^2 . \tag{3.17}$$

The dependence of this objective on the vectors a and b is still non-convex. We thus propose to relax it into a convex function of the matrices A and B, where $A = aa^H$ and $B = bb^H$. However, for the decompositions of A and B in terms of a and b to be valid, the matrices A and B must be rank-1 and positive semi-definite. Hence, we get the optimization problem

$$\begin{aligned} \min_{A,B} \sum_n^N \sum_\tau^T & \left| u_{n,\tau}^H B\, u_{n,\tau} - v_{n,\tau}^H A\, v_{n,\tau}\, \tilde{h}(\lambda_n, \omega_\tau) \right|^2 \\ \text{subject to} \quad & \text{rank}(A) = 1,\ \text{rank}(B) = 1, \\ & A \in \mathbf{S}_+^{(P+1)(K+1)},\ \ B \in \mathbf{S}_+^{(Q+1)(M+1)}, \\ & a_{00} = 1,\ \ a_{0k} = 0 \text{ for } k = 1, 2, \cdots, K, \end{aligned} \tag{3.18}$$

where $\mathbf{S}_+^n$ denotes the cone of $n \times n$ positive semi-definite matrices. Lastly, we apply a convex relaxation of the rank constraints as follows. The positive semi-definite matrices A and B can be pushed to be low-rank by minimizing the sums of their singular values, or equivalently, their traces tr(A) and tr(B). We hence obtain the final form of our optimization problem as

$$\begin{aligned} &\min_{A,B} \sum_{n}^{N} \sum_{\tau}^{T} \left| u_{n,\tau}^H B\, u_{n,\tau} - v_{n,\tau}^H A\, v_{n,\tau}\, \tilde{h}(\lambda_n, \omega_\tau) \right|^2 + \mu_A \text{tr}(A) + \mu_B \text{tr}(B) \\ &\text{subject to} \quad A \in \mathbf{S}_+^{(P+1)(K+1)}, \;\; B \in \mathbf{S}_+^{(Q+1)(M+1)} \\ &\qquad\qquad a_{00} = 1, \;\; a_{0k} = 0 \text{ for } k = 1, 2, \cdots, K, \end{aligned} \tag{3.19}$$

where μ_A and μ_B are positive weight parameters. The objective function in (3.19) is quadratic and convex in A and B. We also observe that the constraint set consists of linear equality constraints and the constraint that A and B be positive semi-definite matrices, which can be reformulated as linear matrix inequality constraints. Therefore, the problem in (3.19) is convex, which we solve using convex optimization techniques [51], [52]. Once the matrices A and B are computed by solving (3.19), the ARMA model parameter vectors a and b can be recovered through rank-1 decompositions of A and B, respectively.

3.2.3 Estimation of Missing Observations of the Process

Having estimated the ARMA model parameters a and b as described in Section 3.2.2, we now discuss the estimation of the missing observations $\{X_{it}^r |\ (i,t) \in \bar{S}^r\}$ of the process. Following the relation in (3.11), the learnt model parameters a and b provide an improved estimate of the JPSD, which we may denote as $h^\star(\lambda_n, \omega_\tau)$. Rearranging $h^\star(\lambda_n, \omega_\tau)$ in matrix form as $h^\star(\Lambda, \Omega)$, we can obtain an improved estimate $\Sigma_{\bar{x}}^\star$ of the covariance matrix $\Sigma_{\bar{x}}$ as

$$\Sigma_{\bar{x}}^\star = U_J\, h^\star(\Lambda, \Omega)\, U_J{}^H \tag{3.20}$$

which follows from the relation in (3.1).

Finally, denoting the vectorized form of each realization X^r of the time-vertex process as $\bar{x}^r$, let us form two new vectors $\bar{y}^r$ and $\bar{z}^r$ from the known and the missing entries of $\bar{x}^r$, respectively, i.e., the process values in the sets $\{X_{it}^r |\ (i,t) \in S^r\}$ and

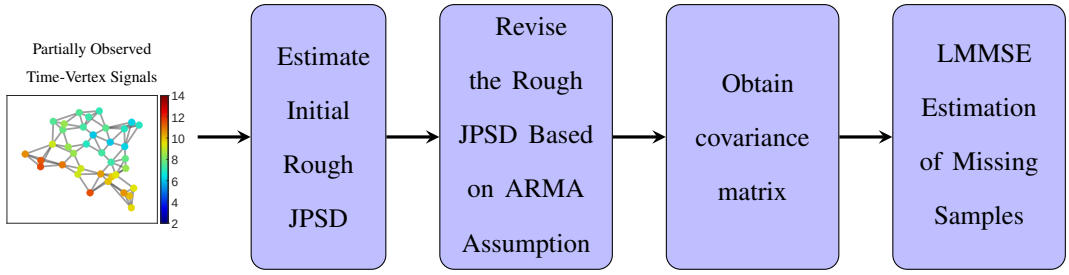


Figure 3.1: The flow chart of the proposed algorithm

$\{X_{it}^r|\ (i,t) \in \bar{S}^r\}$. The vector of missing process values $\bar{z}^r$ can then be estimated with the classical linear minimum mean square error (LMMSE) estimation approach [1] as

$$(\bar{z}^r)^\star = \Sigma_{\bar{z}\bar{y}}^r (\Sigma_{\bar{y}}^r)^{-1}(\bar{y}^r - E[\bar{y}^r]) + E[\bar{z}^r] \tag{3.21}$$

for $r = 1, \ldots, R$. Here $\Sigma_{\bar{z}\bar{y}}^r$ is the cross-covariance matrix of $\bar{z}^r$ and $\bar{y}^r$, and $\Sigma_{\bar{y}}^r$ is the covariance matrix of $\bar{y}^r$, which can be formed by extracting the corresponding entries of the overall covariance matrix $\Sigma_{\bar{x}}^\star$ for each realization X^r.

The pseudo code of the proposed method is given in Algorithm 1, and the flow chart of the proposed method can be found in Figure 3.1.

Algorithm 1 Proposed Method

Input Graph $\mathcal{G}$, available process observations $\{\bar{y}^r\}_{r=1}^R$,

Output Estimated process observations $(\{\bar{z}^r)^\star\}_{r=1}^R$

1: Compute $\tilde{h}$ using $\{\bar{y}^r\}_{r=1}^R$ as explained in Section 3.2.1
2: Compute A and B by solving optimization problem given in (3.19)
3: Find a and b through rank-1 decompositions of A and B
4: Compute the JPSD $h^\star(\lambda_n, \omega_\tau)$ from a and b to using (3.11)
5: Find the covariance matrix $\Sigma_{\bar{x}}^\star$ from JPSD using the relation in (3.20)
6: Find LMMSE estimates $(\bar{z}^r)^\star$ using (3.21)

CHAPTER 4

EXPERIMENTAL RESULTS

In this chapter, we examine the performance of our proposed method. First, we conduct several experiments on synthetic data. We aim to analyze how accurately we can calculate the model with synthetic data experiments. We also examine the parameters that affect the model calculation accuracy. Then, we perform comparative experiments on a real data set.

4.1 Synthetic Data Experiments

In order to test the model computation accuracy of our algorithm in detail, we present results on a synthetically generated ARMA graph process. We use a real graph topology which is generated from the Molène weather data set [44]. It consists of $N = 33$ different meteorological observation stations, each represented as a graph node. We construct a 5-NN graph with Gaussian edge weights based on the geographical proximities of the stations. On this topology, we synthetically generate realizations of an ARMA graph process as given in (3.4).

The purpose of synthetic data experiments is to test how accurately the proposed algorithm can learn a model by solving the relaxed optimization problem in (3.19). We thus learn a model from R complete realizations of the process without any missing observations. The accuracy of the JPSD estimation significantly influences the accuracy of the estimated covariance matrix. Since the estimated covariance matrix is used in the estimation of the missing entries, the eventual algorithm performance depends on the estimation accuracy of the covariance matrix and JPSD. Therefore, we evaluate the model computation accuracy based on how close the estimated JPSD

$h^\star(\lambda_n, \omega_\tau)$ is to the ground truth JPSD $h(\lambda_n, \omega_\tau)$ of the process. We compute the JPSD estimation error as

$$\frac{\|h^\star - h\|_F}{\|h\|_F} \tag{4.1}$$

where, with a slight abuse of notation, $h^\star$ and h denote matrices containing the estimated and the ground truth JPSD values, respectively and $\|.\|_F$ is the Frobenius norm. To compare, we also present the JPSD estimation error of the algorithm in [1] (JWSS), which computes nonparametric JWSS process models based on the JFT of the process realization. The proposed method is also capable of estimating the graph ARMA model parameters a and b in (3.9) and (3.15). The estimation error of a is computed as

$$\frac{\|a^\star - a\|}{\|a\|} \tag{4.2}$$

where $a^\star$ and a denote the estimated and the ground truth vectors, respectively. The operator $\|.\|$ stands for the ℓ_2-norm. Similarly, the estimation error of b is computed as

$$\frac{\|b^\star - b\|}{\|b\|} \tag{4.3}$$

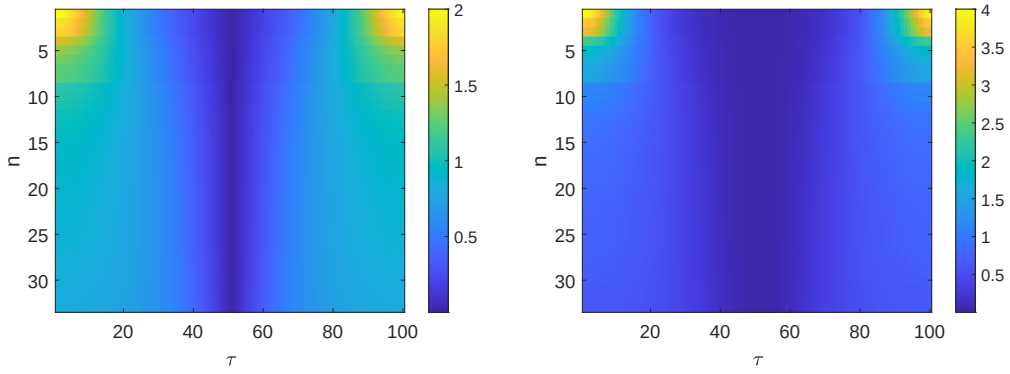
where $b^\star$ and b represent the estimated and the ground truth vectors, respectively.

We first analyze the effect of the number of realizations on the estimation performance of the ARMA graph process model in Section 4.1.1. Then, in Section 4.1.2, we investigate the estimation performance in case noisy observations on the synthetic data. In Section 4.1.3, the effect of model complexity on the estimation performance is examined. The effect of the model mismatch is studied in Section 4.1.4. Lastly, a parameter sensitivity analysis for μ_A and μ_B is given in Section 4.1.5.

4.1.1 Effect of Number of Realizations

We synthetically generate realizations of an ARMA graph process (3.4) of order $P = 1$, $K = 1$, $Q = 1$, $M = 0$ on the Molène graph described in Section 4.1. The time length T of the process is set to 100. The ground truth model parameter vectors in (3.15) and (3.9) are set as $a = \begin{bmatrix} 1 & -0.5 & 0 & 0.5 \end{bmatrix}^H$ and $b = \begin{bmatrix} 0.5 & 0.5 \end{bmatrix}^H$. The joint filter created with these parameters is shown in Figure 4.1a. We filter R realizations of a zero-mean white Gaussian time-vertex process to obtain the realizations

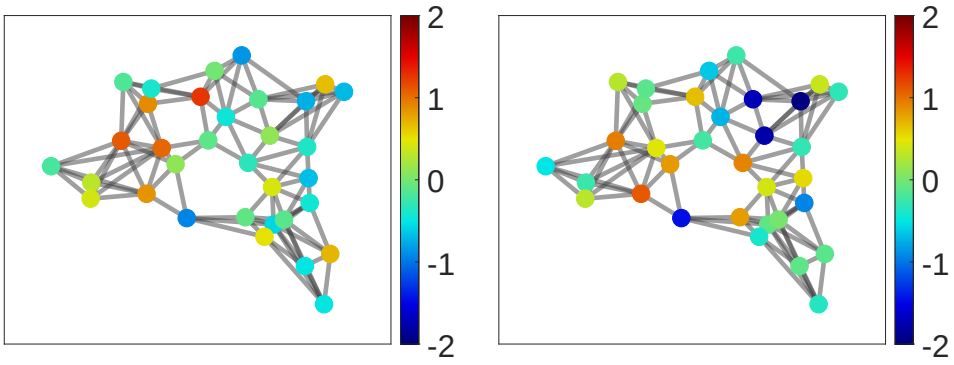
X^r of the synthetically generated ARMA graph process X, for $r = 1, \ldots, R$. The JPSD of the generated process is given in Figure 4.1b. An example realization of the generated process is shown in Figure 4.2. Since the joint filter $H(\lambda_n, \omega_\tau)$ shows low pass characteristics in both the time and the vertex domains, the generated samples are expected to change smoothly over the graph and the time. One can observe from Figure 4.2 that the realization of the generated ARMA graph process are smoothly changing over both domains.



(a) (b)

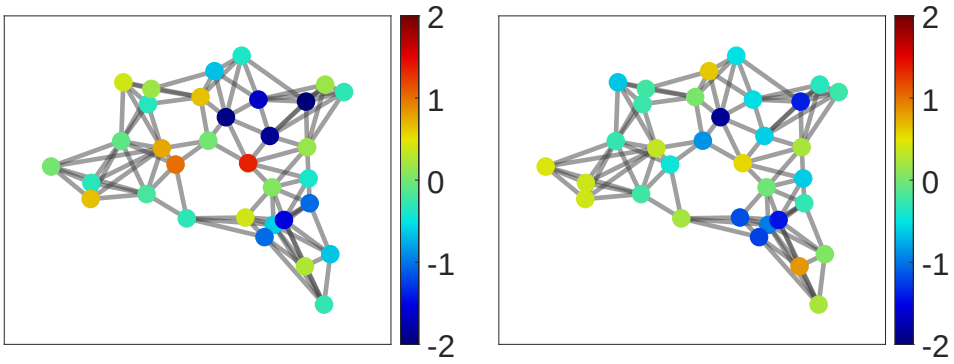
Figure 4.1: (a) The magnitude response of the joint filter $|H(\lambda_n, \omega_\tau)|$, (b) The JPSD $h(\lambda_n, \omega_\tau)$ of the synthetically generated ARMA graph process

In Figure 4.3a, we study the variation of the JPSD error with the number of realizations R of the process. We have performed 20 experiments, and the average estimation performance is shown in Figure 4.3. μ_A and μ_B are set to 10^{-5}. The model orders are assumed to be known. Figure 4.3a shows that the JPSD error of the proposed method is very low compared to the baseline method [1]. Especially in cases where the number of realizations is small, the proposed method estimates JPSD with less error than the nonparametric approach [1]. Although the difference between the nonparametric approach and the proposed method decreases as the number of observations increases, the proposed method still performs better. As observed, the JPSD error approaches zero with an increasing number of realizations shows that after several relaxations, the approximate optimization problem in (3.19) successfully finds the correct solution. It provides an accurate estimate of the parametric process



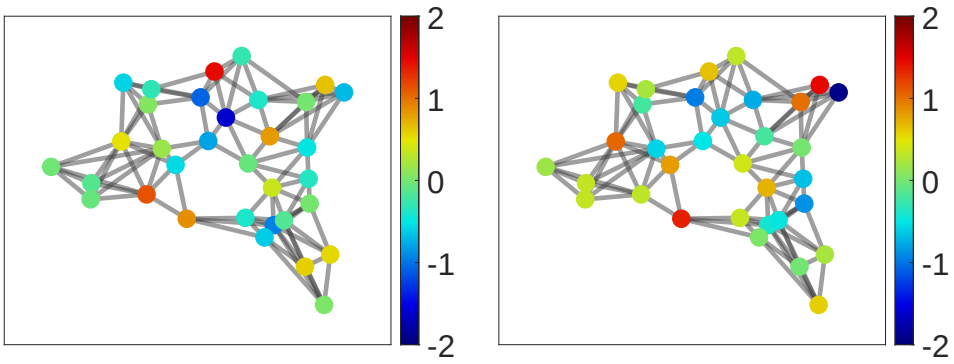
(a) The graph signal at time $t = 25$

(b) The graph signal at time $t = 26$



(c) The graph signal at time $t = 27$

(d) The graph signal at time $t = 28$



(e) The graph signal at time $t = 29$

(f) The graph signal at time $t = 30$

Figure 4.2: Samples from the synthetically generated ARMA graph process

model as well. The proposed method can also calculate ARMA graph process parameters. Therefore, we have also measured how accurately the model parameters are estimated. The estimation error of the parameters a and b according to the changing number of observations are shown in Figure 4.3b and 4.3c, respectively. These results indicate that the proposed method can successfully recover ARMA graph process models.

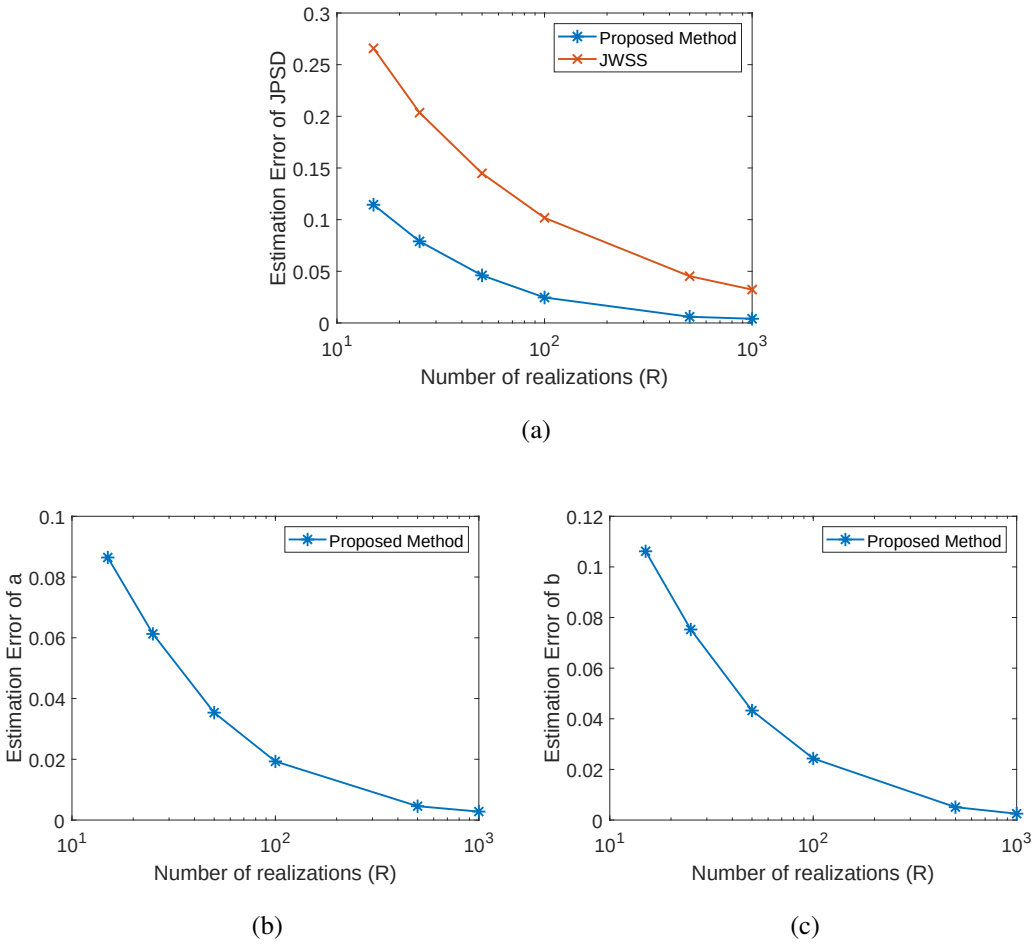


Figure 4.3: (a) JPSD error vs. the number of realizations, (b) Estimation error of a vs. the number of realizations, (c) Estimation error of b vs. the number of realizations

In the light of these experiments, we find that the number of observations significantly affects the model estimation performance. Also, using a parametric approach improves the estimation results, especially when a small amount of data is available. Furthermore, the optimization problem in (3.19) is able to successfully capture the

ARMA graph process model.

4.1.2 Effects of Noisy Observations

In this experiment, we examine how much the model estimation performance is affected by the noise level. To do so, synthetic data with different signal-to-noise ratio (SNR) levels are generated by adding additive white Gaussian noise. We use the same ARMA graph process model described in Section 4.1.1. The number of realizations is fixed as $R = 1000$. μ_A and μ_B are set to 10^{-5}. The model orders are set to their ground truth values.

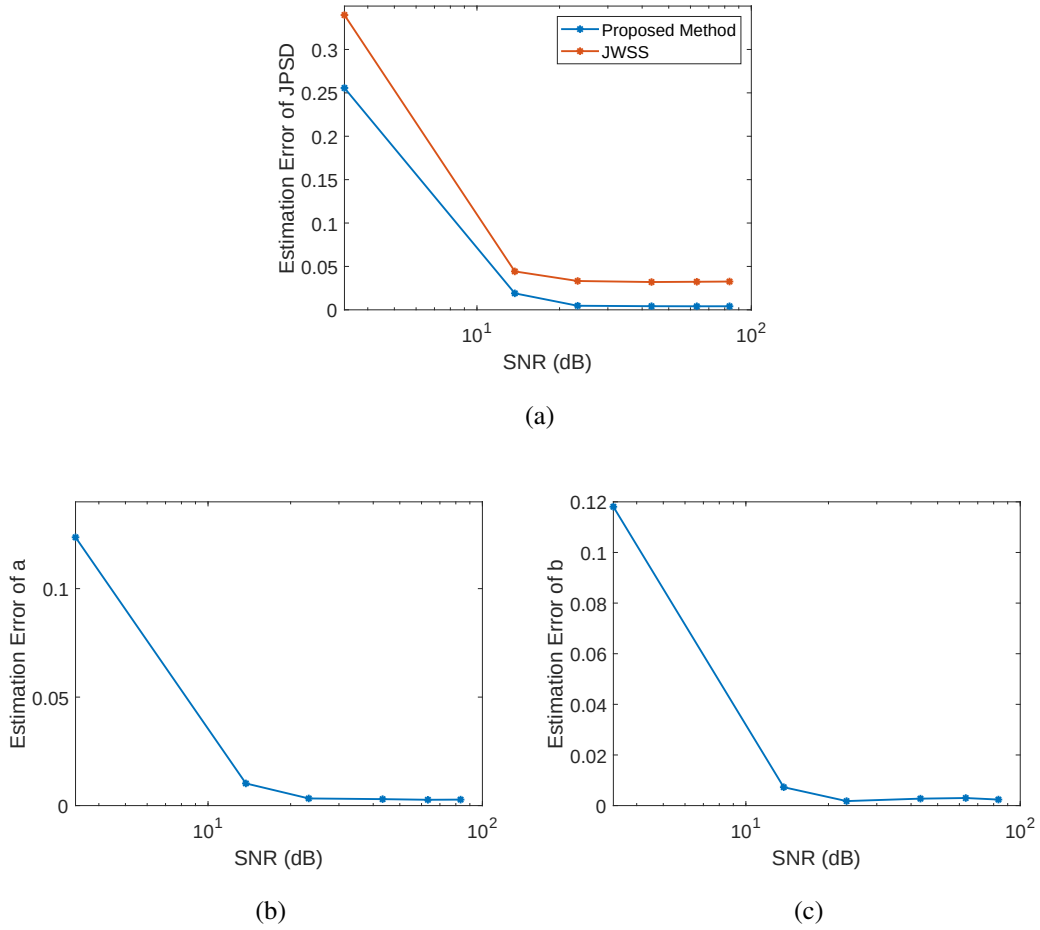


Figure 4.4: (a) JPSD error vs. SNR (dB), (b) Estimation error of a vs. SNR (dB), (c) Estimation error of b vs. SNR (dB)

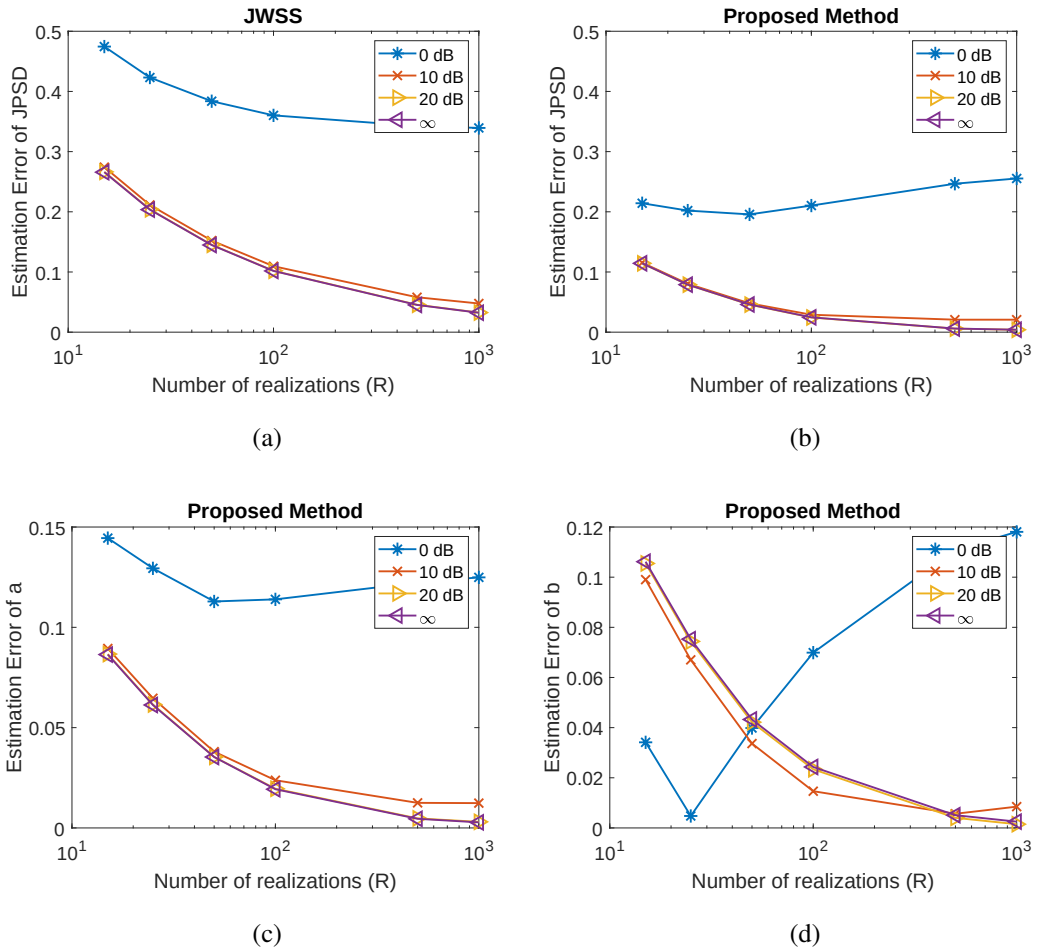


Figure 4.5: (a) JPSD error vs. the number of realizations for nonparametric JWSS model [1] at different SNR levels, b) JPSD error vs. the number of realizations for the proposed method at different SNR levels, (c) Estimation error of a vs. the number of realizations for the proposed method at different SNR levels, (c) Estimation error of b vs. the number of realizations for the proposed method at different SNR levels

In Figure 4.4a, we present the obtained JPSD errors. The experiment is repeated 20 times, and the average of the results is given in Figure 4.4. The figure shows that the proposed parametric method performs much better than the nonparametric JWSS method [1] under noisy observations. It should be noted that there is a considerable performance gap between the proposed method and the nonparametric JWSS method, especially at low SNR levels. The JPSD estimate of the JWSS method [1] is known to be quite comparable to our initial JPSD estimate $h^{\star}(\lambda_n, \omega_\tau)$. These results confirm

that the proposed algorithm can successfully refine the initial noisy JPSD estimate to compute an improved model that better approximates the process spectrum.

Then, the number of realizations experiment explained in Section 4.1.1 is repeated at different noise levels. In our experiments, data are generated in different SNR levels, then the number of realizations are changed during the experiment. The same procedure is repeated 20 times, and the average of the results is given in Figure 4.5.

Comparing Figures 4.5a and 4.5b, it can be observed that the proposed method outperforms the nonparametric method at different noise levels. The results indicate that the proposed method is more robust to noise than the nonparametric method.

Noiseless results and the results when SNR is equal to 20dB in Figure 4.5b are very close to each other. When SNR ratio is equal to 10dB, the estimation error of the proposed method approaches 0.02. Moreover, when the noise power is equal to the signal power, i.e. SNR $= 0$dB, the increase in the number of observations may not increase the estimation performance. However, the proposed method performs much better than the nonparametric method when signal-to-noise ratio is equal to 0 dB.

4.1.3 Effect of Model Complexity

We evaluate the estimation performance for 6 different models. Experiment is repeated 5 times and the average results can be seen in Figure 4.6. μ_A and μ_B values are set to 10^{-5}. The number of realizations R is set to 1000. The purpose of this experiment is to study how the number of realizations required for accurate model estimation evolves in relation to the model complexity. The ground truth model orders P, K, Q, M are thus provided to the algorithm. Figure 4.6a shows that the JPSD estimation with the nonparametric approach is not affected much from the complexity of the model. On the other hand, Figure 4.6b shows that the estimation accuracy of the proposed method highly depends on the model complexity. As the model complexity increases, the estimation performance decreases for a fixed number of realizations. This can be explained in the following way: The optimization variables in the problem given in (3.19) are the A and B matrices. Therefore, the number of optimization variables rapidly increases as the model orders P, K, Q, and M increase. The increasing

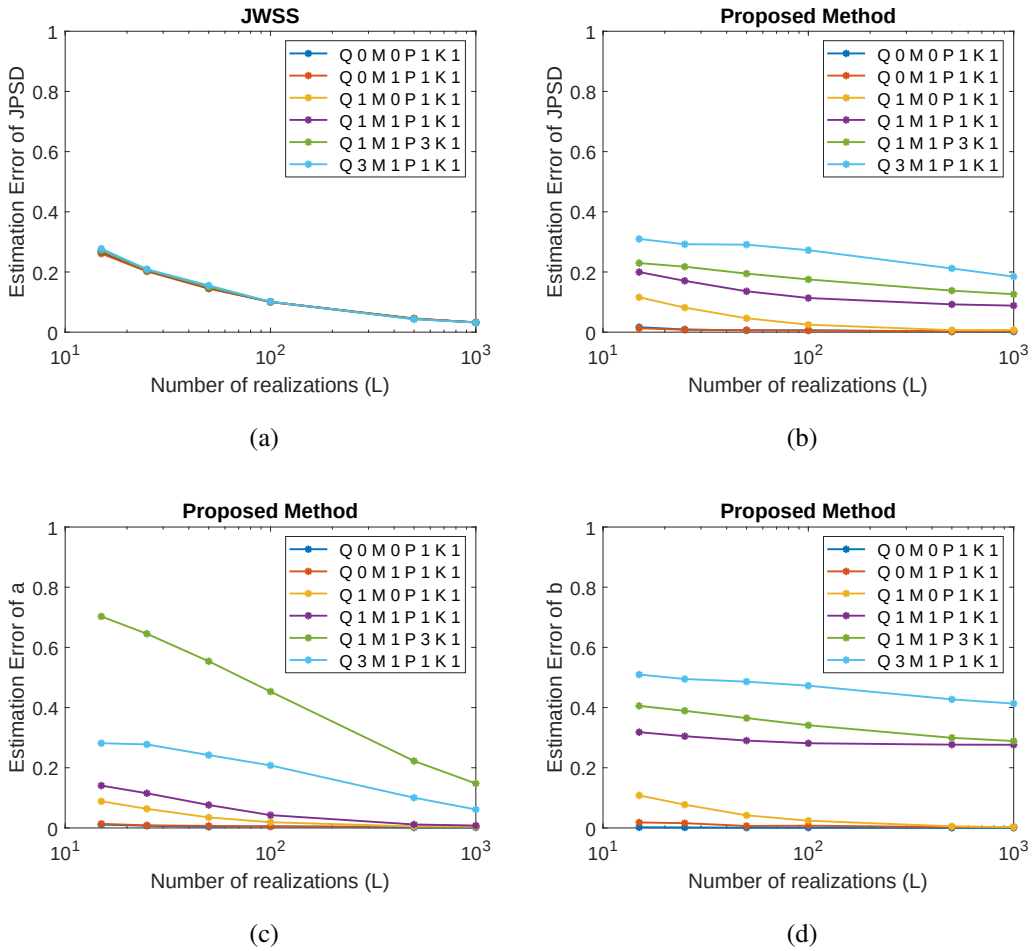


Figure 4.6: (a) JPSD error vs. the number of realizations for the nonparametric JWSS method, (b) JPSD error vs. the number of realizations for the proposed method, (b) Estimation error of a vs. the number of realizations for the proposed method, (c) Estimation error of b vs. the number of realizations for the proposed method

algorithm complexity at high model orders affects the estimation performance, which can be considered as the main drawback of the proposed method. Although the performance of the proposed method decreases with the increase in model complexity, the error keep decreasing as the number of observations increases. This is quite in line with the concept of overfitting in machine learning: A higher model complexity increases the need for training data in order to properly learn a model. The model complexity influences the estimation error in a and b as well. The estimation errors of the a and b vectors are shown in Figure 4.6c and Figure 4.6d, respectively. The highest estimation error for a is observed when model order P is set to 3. On the other

hand, when Q is equal to 3, the highest estimation error of b is observed. This is a natural consequence of the fact that the lengths of the a and b vectors are proportional to the parameters P and Q, respectively.

4.1.4 Effect of Model Mismatch

In this experiment, we study how the estimation accuracy is affected by the mismatch between the ground truth model order parameters and their values selected by the algorithm. For this purpose, we use the process mentioned in Section 4.1.1, which is an ARMA graph process (3.4) of order $P = 1$, $K = 1$, $Q = 1$, $M = 0$, and $a = \begin{bmatrix} 1 & -0.5 & 0 & 0.5 \end{bmatrix}^H$ and $b = \begin{bmatrix} 0.5 & 0.5 \end{bmatrix}^H$. In each part of the experiment, the algorithm is set to learn a model whose P, K, Q, or M order parameter deviates from its ground truth value, and the effect of this deviation on the estimation performance is studied. Each experiment is repeated 5 times and the average results are reported. μ_A and μ_B values are set to 10^{-5}. The number of realizations R is set to 1000.

First, we set the model order parameters K, Q, and M of the algorithm to their ground truth values i.e. $K = 1$, $Q = 1$, and $M = 0$. The value of P is changed during the experiment into $P = 1, 2, \ldots, 6$. The results are shown in Figures 4.7a, 4.7c, and 4.7e. When P is not selected correctly i.e. it is different than the ground truth model order, the accuracy of the model computation starts to decrease. Although the performance drops, the increase in the estimation errors is not dramatic. When P is equal to 6, the JPSD error is still less than 0.02. If we analyze the estimated vector a, we can see that the terms a_{pk} with $p > 1$ are predicted close to 0 when P is chosen greater than its ground truth value 1. These results show that selecting P incorrectly does not affect the performance that much. Although P is directly related to only a parameters, the estimation errors of both a and b increase if the P value is incorrectly assigned. However, an incorrect selection of P affects the estimation performance of a more than that of b, which is natural as K is directly related to the parameter vector a.

Similar experiments are performed for K, Q, and M, and the results of these experiments are shown in Figures 4.7 and 4.8.

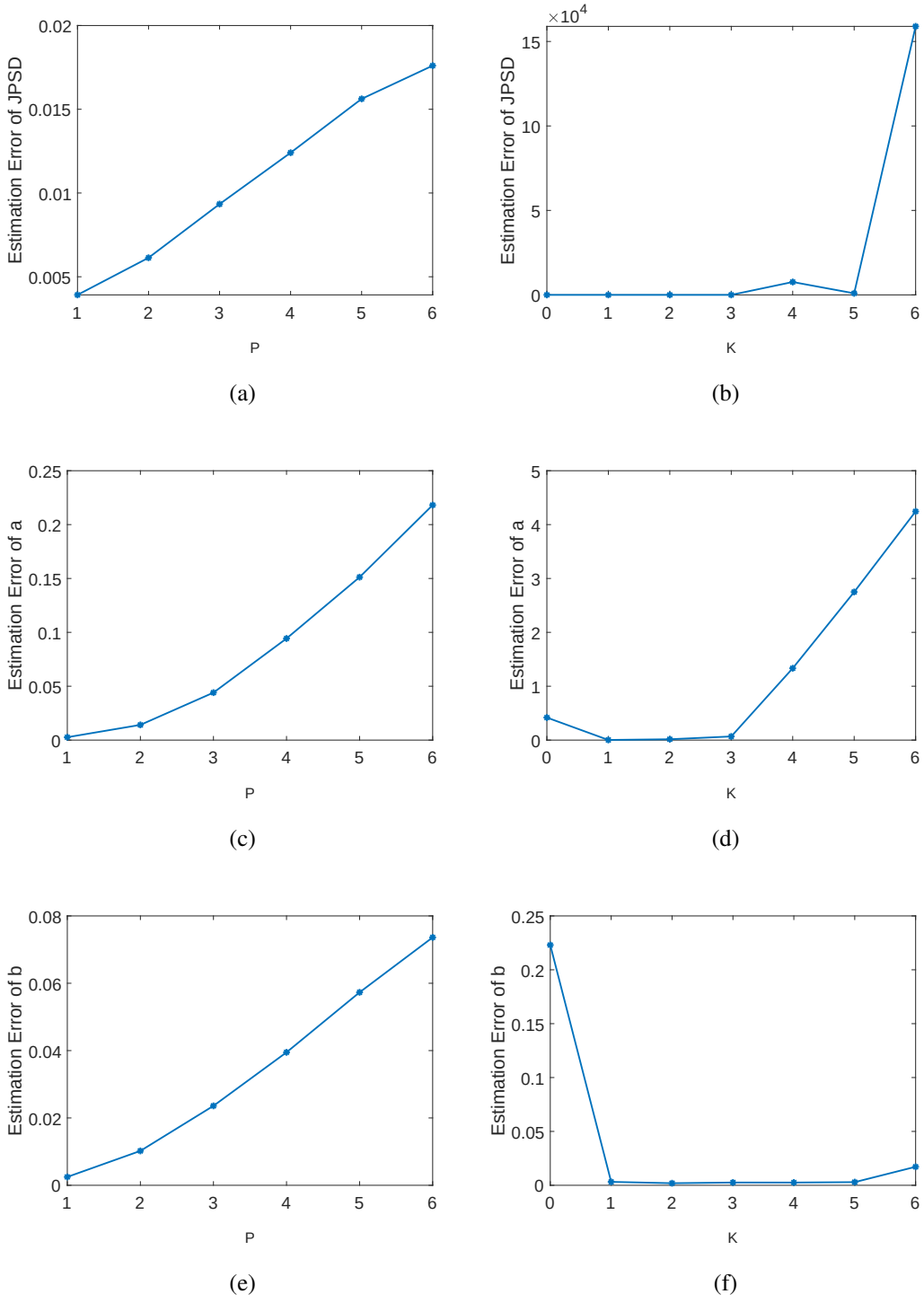


Figure 4.7: Results of parameter mismatch experiments for P and K where the ground truth $P = 1$ and the ground truth $K = 1$. (a) JPSD error vs. P , (b) JPSD error vs. K, (c) Estimation error of a vs. P, (d) Estimation error of a vs. K, (e) Estimation error of b vs. P, (f) Estimation error of b vs. K

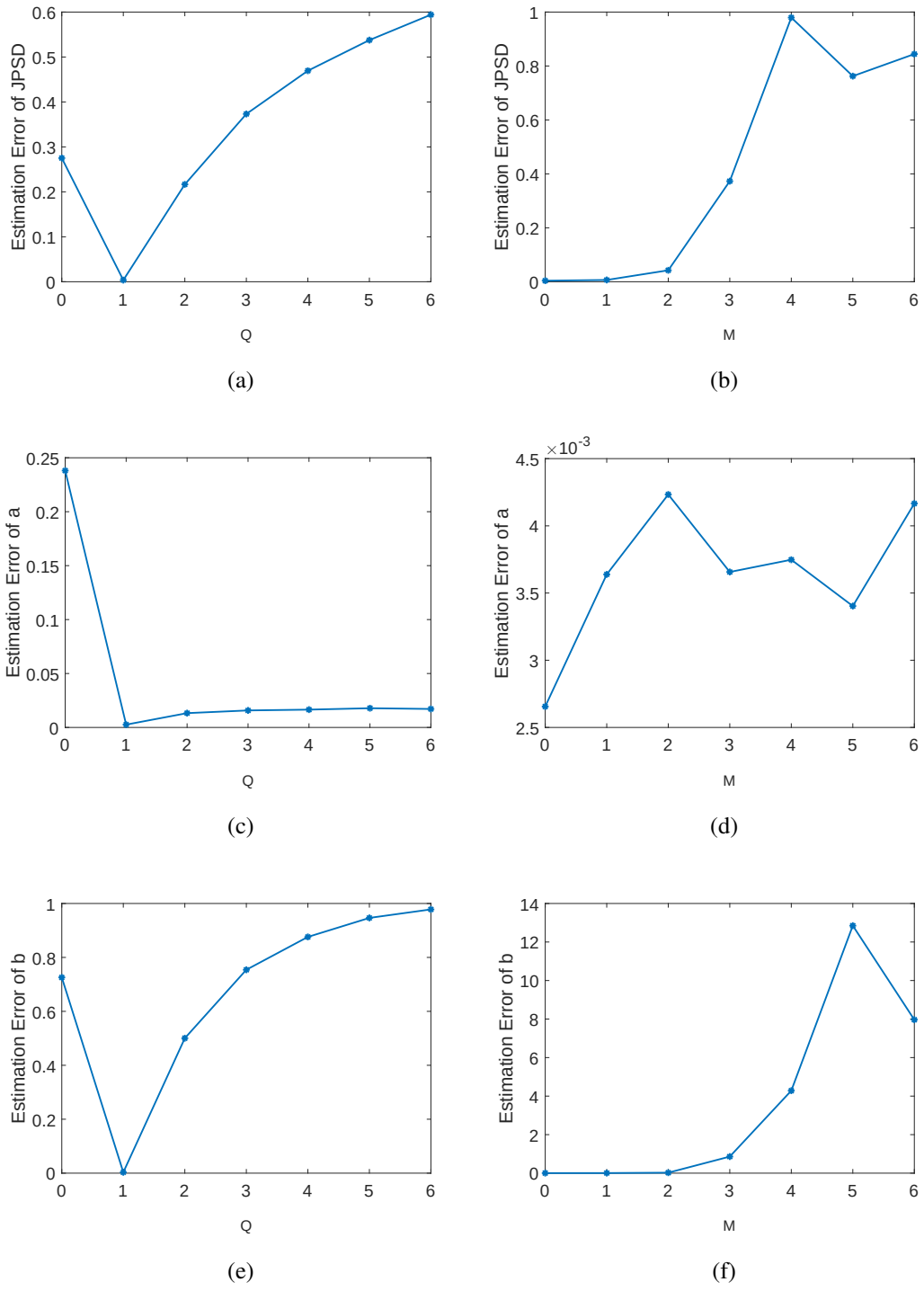


Figure 4.8: Results of parameter mismatch experiments for Q and M where the ground truth $Q = 1$ and the ground truth $M = 0$. (a) JPSD error vs. Q, (b) JPSD error vs. M, (c) Estimation error of a vs. Q, (d) Estimation error of a vs. M, (e) Estimation error of b vs. Q, (f) Estimation error of b vs. M

The results for K are given in Figure 4.7b, Figure 4.7d, and Figure 4.7f. The ground truth value of K is 1. The best results are obtained when K is set to 1. We observe that setting K incorrectly causes sharp performance drop. Moreover, an incorrect selection of K affects the estimation performance a more than that of b. Next, we study the effect of the parameter Q. The ground truth value of Q is 1. The results for Q are given in Figures 4.8a, 4.8c, and 4.8e. The best results are obtained when Q is set to the ground truth value i.e. 1. Choosing Q smaller or larger than the ground truth value causes a drop in the performance, especially resulting from the errors in the estimation of the b vector. Finally, we analyze the effect of M. The results for M are given in Figures 4.8b, 4.8d, and 4.8f. The ground truth value for M is 0. If M is chosen higher than 0, the performance starts to decrease.

The experiments of this section show that when the model order parameters selected by the algorithm deviate from their ground truth values significantly, the estimation performance drops. Therefore, the selection of the model orders is crucial. Setting P incorrectly has a minor impact on the performance, while choosing K incorrectly may affect the performance considerably. Similarly, the algorithm performance seems to be more affected by the choice of M than that of Q. This is an interesting observation, since P and Q control the "time delay" of the process, i.e., to what extent the previous instances of the process values x_{t-p} and the external noise values w_{t-q} are expected to contribute to the current process value x_t. On the other hand, the K and M parameters control the extent of the diffusion of the process values over the graph, through the order of the polynomial graph filters. The results of the experiments suggest that the algorithm has less tolerance to the over-diffusion of the process values on the graph, compared to the overestimation of the length of time-filtering.

4.1.5 Parameter Sensitivity Analysis

In our last experiment on synthetic data, we examine the sensitivity of the proposed method to the weight parameters μ_A and μ_B in the objective function. For this purpose, we use the synthetic model explained in Section 4.1.1, and set the number of realizations R to 1000. The model orders are set to their ground true values, i.e. $P = 1$, $K = 1$, $Q = 1$, and $M = 0$. Then, different μ_A and μ_B values are set

during the experiment and the estimation performance is measured for the different μ_A and μ_B values. The experiment is repeated for 5 times, and the average of the JPSD estimation errors are given in Table 4.1. The results show that in order to achieve high estimation performance, the algorithm hyperparameters μ_A and μ_B must be selected carefully. Setting μ_A and μ_B to sufficiently small values within the range $\mu_A \in [10^{-7}\ 10^{-3}], \mu_B \in [10^{-7}\ 10^{-3}]$ gives the best results for this model. If μ_B is set large, we have observed that the algorithm tends to trivially estimate the b vector as a zero vector, in order to minimize the term values of b becomes 0 to minimize tr(B) in the objective function.

μ_A \ μ_B	10^{-7}	10^{-5}	10^{-3}	10^{-1}	10^{1}	10^{3}	10^{5}	10^{7}
10^{-7}	**0.0038**	**0.0038**	**0.0038**	0.0039	0.0074	0.2809	1.0000	1.0000
10^{-5}	**0.0038**	**0.0038**	**0.0038**	0.0039	0.0074	0.2809	1.0000	1.0000
10^{-3}	**0.0038**	**0.0038**	**0.0038**	0.0039	0.0074	0.2809	1.0000	1.0000
10^{-1}	**0.0038**	**0.0038**	**0.0038**	**0.0038**	0.0074	0.2810	1.0000	1.0000
10^{1}	0.0209	0.0209	0.0209	0.0209	0.0219	0.2895	1.0000	1.0000
10^{3}	0.3487	0.3487	0.3487	0.3487	0.3497	0.4701	1.0000	1.0000
10^{5}	0.4391	0.4391	0.4391	0.4391	0.4391	0.4718	1.0000	1.0000
10^{7}	0.4405	0.4405	0.4405	0.4405	0.4405	0.4711	1.0000	1.0000

Table 4.1: Parameter sensitivity analysis for μ_A and μ_B

4.2 Comparative Experiments

We next evaluate the performance of our method with comparative experiments on the Molène weather data set. We experiment on temperature measurements recorded on $N = 37$ different meteorological observation stations, each represented as a graph node. We construct a 10-NN graph with Gaussian edge weights based on the geographical proximities of the stations. We regard each 24-hour measurement sequence as one realization of a time-vertex graph process X with graph size $N = 37$ and time length $T = 24$, obtaining a total of $R = 31$ realizations.

For each realization, we consider some of the process values as missing observations.

The time and vertex indices of the missing observations are randomly selected. We learn a process model with the known observations, which is then used for obtaining an LMMSE estimate of the missing observations as discussed in Section 3.2. The estimation performance of our method is compared with the following approaches: Nonparametric JWSS process models[1] (JWSS) [1], graph vector autoregressive recursions (G-VAR) [50], graph polynomial vector autoregressive recursions (GP-VAR) [50], vector autoregressive process models (VAR) [53], and prediction via an AR time process model (AR) [9]. Half of the known process observations are used for validation for all algorithms requiring the hyperparameter tuning such as model orders. Hyperparameters of the proposed method is set as follows: $P = 2$, $K = 1$ $Q = 0$, $M = 3$, $\mu_A = 10$ and $\mu_B = 10$.

The performances of the algorithms are compared with respect to the normalized mean square error (NME) of their estimates of missing observations as given by

$$NME = \left(\sum_{r=1}^{R} \|\bar{z}^r - (\bar{z}^r)^\star\|^2 / \sum_{r=1}^{R} \|\bar{z}^r\|^2 \right)^{1/2} \tag{4.4}$$

where $\bar{z}^r$ and $(\bar{z}^r)^\star$ represent the ground truth values of the missing observations and their estimates, respectively.

The variation of the NME with respect to the ratio of missing observations is presented in Figure 4.9 for all methods. We observe that methods based on graph process models perform clearly better than the VAR and AR models, which treat the data as individual time series and do not employ the information of the underlying graph topology. Regarding the comparison among the graph-based methods, it is interesting to note that the proposed method and the JWSS method, which compute models based on the time-vertex joint spectrum of the process, provide better estimation performances than the G-VAR and GP-VAR methods, which do not exploit the information of the joint spectrum. This confirms the expectation that the joint time-vertex spectrum of a time-vertex signal may provide critical information about its characteristics, which cannot be captured with only vertex-based frequency analysis. The proposed method yields a lesser estimation error than JWSS, suggesting that the parametric graph process models considered in our work may be computed more

[1] As the computation of the JFT is not possible with randomly selected missing observations in this setting, are using a variant of the original method [1] by estimating the JPSD from the covariance matrix as in (3.7) and refining it by extracting its diagonal entries.

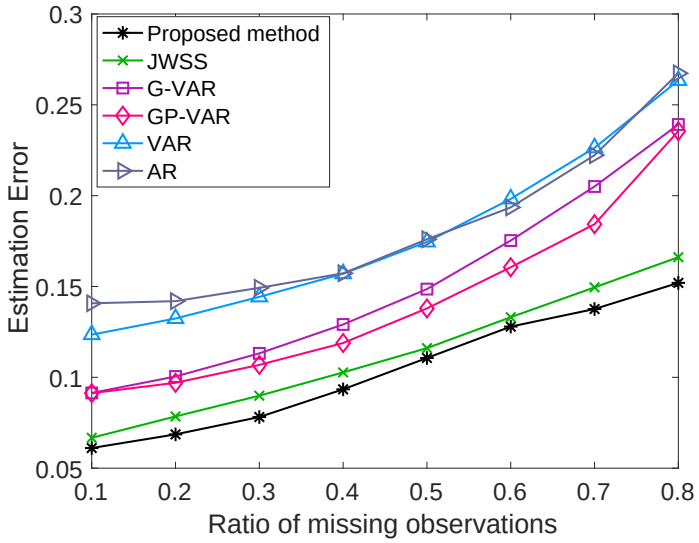


Figure 4.9: NME of compared methods on the Molène weather data

accurately than nonparametric models [1], in settings with incomplete process realizations. An overall consideration of these results indicates that our method is able to successfully combine the efficacy of parametric process models with the information of the time-vertex joint spectral characteristics of the process.

CHAPTER 5

CONCLUSION

In this thesis, we have proposed a method for learning time-vertex joint graph ARMA process models from incomplete observations of the process. First, an empirical estimate of the JPSD of the process is obtained by employing the particular block-Toeplitz structure of its covariance matrix. Then, we have formulated the learning of the ARMA process parameters via an optimization problem where the error between the parametric JPSD of the process and its empirical estimate is minimized. While this initial problem is non-convex, we have proposed to relax it into a convex one by rewriting the ARMA process parameters in terms of low-rank positive semi-definite matrices. Once the process parameters are computed, the initially unavailable observations of the process are inferred with an LMMSE estimation. Compared to recent methods in the literature, the resulting algorithm has much milder assumptions on the structure of the available observations of the process. Since, it does not require a set of "training" observations completely known over the entire graph and during a whole time interval, unlike many state-of-the-art methods.

The foremost part of our proposed method examines the estimation of the JPSD. After estimating the JPSD, the missing observations are simply computed by LMMSE estimation. Therefore, it is crucial to successfully estimate the JPSD if the process comes from the JWSS ARMA process model. Thus, the JPSD estimation performance of the proposed method is investigated for different number of realizations. The experiment has been conducted on synthetically generated JWSS ARMA graph processes. Then, the proposed parametric JPSD estimation method is compared with the reference nonparametric JPSD estimation method proposed in [1]. As the number of realizations increases, the estimation error drops more drastically in the proposed

method. Moreover, when the performance of the estimation of the ARMA model parameters a_{PK} and b_{QM} have also been analyzed, the results show that the JWSS ARMA graph process model can be successfully recovered from data when there is a sufficient amount of realizations available.

In the second experiment, we study the JPSD estimation performance in case of noisy observations of the process. The experiment is conducted on the same synthetically generated ARMA graph process. As the SNR increases, the JPSD is estimated more accurately as expected. The performance drop in the proposed method at large noise levels is less than the nonparametric model. As a result, using the parametric ARMA model provides better robustness to noise, in comparison with the baseline nonparametric model proposed in[1]. Furthermore, the estimation performances of the model parameters a and b are examined in this setting as well. The results show that the JWSS ARMA graph process model can also be successfully learned under noisy measurements.

The effect of model complexity, model mismatch are investigated using synthetic data. As model complexity increases, the performance of the proposed method decreases. If model parameters are not set correctly, it decreases the estimation performance. In addition, sensitivity analysis for μ_A and μ_B is performed. The performance of the proposed method is affected by μ_A and μ_B. In order for the proposed method to give the better results, the most suitable P, K, Q, M, μ_A, and μ_B parameters for the process should be selected.

In the comparative experiments, we use a real-world data set that contains hourly temperature measurements. We have considered an experimentation setting where missing observations occur at arbitrary locations in both vertex and time domains. We aim to estimate the incomplete observations from the observed ones. The performance of the estimation of missing values is measured by the NME. The estimation performance of the proposed method is compared with five different algorithms. The ratio of missing observations is altered during the experiment and its effect on the estimation error is investigated. As expected, as the ratio of the missing observations increases, the estimation error increases. The proposed method and the algorithm proposed in [1], which use the JWSS process model, to outperform the other methods.

This confirms the expectation that modeling a real-world time-varying graph signal with the JWSS assumption may be helpful. Moreover, the results show that modeling the time-varying graph process as a parametric JWSS ARMA model provides better signal estimation performance than a JWSS nonparametric model.

In this thesis, as a proof of concept that the assumption JWSS ARMA process models can improve the estimation performance in a weather data set. As future work, more extensive experiments on real-world data sets should be conducted in order to better understand when modeling a time-varying graph signal as a JWSS ARMA graph process model. We have focused on a setting where the graph is fixed. Another interesting future work on this study can be to explore wide-sense stationarity when the graph topology has a time-varying structure as well as the graph signals. Another potential extension of our study would address the problem of learning or correcting the graph topology along with the process model. Note that the validity of the GWSS assumption directly depends on the graph topology. Therefore, correcting an initially available but often erroneous graph topology so as to achieve a better fit between the process observations and the process model may improve the signal estimation performance in a wide range of applications.

REFERENCES

[1] A. Loukas and N. Perraudin, "Stationary time-vertex signal processing," *EURASIP Journal on Advances in Signal Processing*, vol. 2019, no. 1, p. 36, 2019.

[2] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Processing Magazine*, vol. 30, pp. 83–98, May 2013.

[3] A. Ortega, P. Frossard, J. Kovačević, J. M. F. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," *Proceedings of the IEEE*, vol. 106, pp. 808–828, May 2018.

[4] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs," *IEEE Transactions on Signal Processing*, vol. 61, pp. 1644–1656, April 2013.

[5] F. Zhang and E. R. Hancock, "Graph spectral image smoothing using the heat kernel," *Pattern Recognition*, vol. 41, pp. 3328–3342, Nov. 2008.

[6] D. I. Shuman, P. Vandergheynst, and P. Frossard, "Chebyshev polynomial approximation for distributed signal processing," in *2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)*, pp. 1–8, June 2011.

[7] F. Grassi, A. Loukas, N. Perraudin, and B. Ricaud, "A time-vertex signal processing framework: Scalable processing and meaningful representations for time-series on graphs," *IEEE Transactions on Signal Processing*, vol. 66, pp. 817–829, Feb 2018.

[8] A. G. Marques, S. Segarra, G. Leus, and A. Ribeiro, "Stationary graph processes and spectral estimation," *IEEE Transactions on Signal Processing*, vol. 65, pp. 5911–5926, Nov 2017.

[9] M. Hayes, *Statistical Digital Signal Processing and Modeling*. Wiley, 1996.

[10] X. Dong, D. Thanou, and M. Rabbat, "Learning Graphs From Data," *IEEE Signal Processing Magazine*, vol. 36, no. May, pp. 44–63, 2019.

[11] M. Belkin and P. Niyogi, "Semi-supervised learning on riemannian manifolds," *Machine Learning*, vol. 56, no. 1-3, pp. 209–239, 2004.

[12] X. Zhu, Z. Ghahramani, and J. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, p. 912–919, AAAI Press, 2003.

[13] A. J. Smola and R. Kondor, "Kernels and regularization on graphs," in *Learning Theory and Kernel Machines* (B. Schölkopf and M. K. Warmuth, eds.), (Berlin, Heidelberg), pp. 144–158, Springer Berlin Heidelberg, 2003.

[14] D. Zhou and B. Schölkopf, "A regularization framework for learning from graph data," in *ICML Workshop on Statistical Relational Learning and Its Connections to Other Fields*, January 2004.

[15] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *Advances in Neural Information Processing Systems 16 (NIPS 2003)*, January 2003.

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

[17] D. A. Spielman, "Spectral graph theory," in *Combinatorial Scientific Computing*, Chapman and Hall/CRC Press, 2012.

[18] D. I. Shuman, P. Vandergheynst, and P. Frossard, "Chebyshev polynomial approximation for distributed signal processing," in *2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)*, pp. 1–8, IEEE, jun 2011.

[19] S. Safavi and U. A. Khan, "Revisiting Finite-Time Distributed Algorithms via Successive Nulling of Eigenvalues," *IEEE Signal Processing Letters*, vol. 22, pp. 54–57, jan 2015.

[20] A. Loukas, A. Simonetto, and G. Leus, "Distributed Autoregressive Moving Average Graph Filters," *IEEE Signal Processing Letters*, vol. 22, pp. 1931–1935, nov 2015.

[21] I. Pesenson, "Sampling in Paley-Wiener spaces on combinatorial graphs," *Transactions of the American Mathematical Society*, vol. 360, pp. 5603–5627, jul 2008.

[22] S. K. Narang, A. Gadde, and A. Ortega, "Signal processing techniques for interpolation in graph structured data," *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, pp. 5445–5449, oct 2013.

[23] M. Tsitsvero, S. Barbarossa, and P. Di Lorenzo, "Signals on graphs: Uncertainty principle and sampling," *IEEE Transactions on Signal Processing*, vol. 64, pp. 4845–4860, sep 2016.

[24] A. G. Marques, S. Segarra, G. Leus, and A. Ribeiro, "Sampling of Graph Signals With Successive Local Aggregations," *IEEE Transactions on Signal Processing*, vol. 64, pp. 1832–1843, apr 2016.

[25] S. Chen, R. Varma, A. Sandryhaila, and J. Kovačević, "Discrete Signal Processing on Graphs: Sampling Theory," *IEEE Transactions on Signal Processing*, vol. 63, pp. 6510–6523, dec 2015.

[26] Z. Zhang, Y. Xu, J. Yang, X. Li, and D. Zhang, "A Survey of Sparse Representation: Algorithms and Applications," *IEEE Access*, vol. 3, pp. 490–530, 2015.

[27] E. Simoncelli and E. Adelson, "Noise removal via bayesian wavelet coring," in *Proceedings of 3rd IEEE International Conference on Image Processing*, vol. 1, pp. 379–382 vol.1, 1996.

[28] K. Kreutz-Delgado, J. F. Murray, B. D. Rao, K. Engan, T.-w. Lee, and T. J. Sejnowski, "Dictionary Learning Algorithms for Sparse Representation," *Neural Computation*, vol. 15, pp. 349–396, feb 2003.

[29] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4311–4322, 2006.

[30] Zhuolin Jiang, Zhe Lin, and L. S. Davis, "Label Consistent K-SVD: Learning a Discriminative Dictionary for Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, pp. 2651–2664, nov 2013.

[31] I. Tosic and P. Frossard, "Dictionary Learning," *IEEE Signal Processing Magazine*, vol. 28, pp. 27–38, mar 2011.

[32] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Applied and Computational Harmonic Analysis*, vol. 30, no. 2, pp. 129–150, 2011.

[33] M. Gori, G. Monfardini, and F. Scarselli, "A new model for earning in raph domains," *Proceedings of the International Joint Conference on Neural Networks*, vol. 2, pp. 729–734, 2005.

[34] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.

[35] Y. Li, R. Zemel, M. Brockschmidt, and D. Tarlow, "Gated Graph Sequence Neural Networks," in *International Conference on Learning Representations 2016*, 2016.

[36] H. Dai, Z. Kozareva, B. Dai, A. Smola, and L. Song, "Learning Steady-States of Iterative Algorithms over Graphs," in *Proceedings of the 35th International Conference on Machine Learning* (J. Dy and A. Krause, eds.), vol. 80 of *Proceedings of Machine Learning Research*, pp. 1106–1114, PMLR, 2018.

[37] J. Bruna, W. Zaremba, A. Szlam, and Y. Lecun, "Spectral networks and locally connected networks on graphs," in *International Conference on Learning Representations (ICLR2014), CBLS, April 2014*, 2014.

[38] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering," in *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain* (D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, eds.), pp. 3837–3845, 2016.

[39] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net, 2017.

[40] R. Levie, F. Monti, X. Bresson, and M. M. Bronstein, "CayleyNets: Graph Convolutional Neural Networks With Complex Rational Spectral Filters," *IEEE Trans. Signal Process.*, vol. 67, no. 1, pp. 97–109, 2019.

[41] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, (Red Hook, NY, USA), p. 2001–2009, Curran Associates Inc., 2016.

[42] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, p. 2014–2023, JMLR.org, 2016.

[43] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, p. 1263–1272, JMLR.org, 2017.

[44] B. Girault, "Stationary graph signals using an isometric graph translation," in *2015 23rd European Signal Processing Conference (EUSIPCO)*, pp. 1516–1520, Aug 2015.

[45] N. Perraudin and P. Vandergheynst, "Stationary signal processing on graphs," *IEEE Transactions on Signal Processing*, vol. 65, pp. 3462–3477, July 2017.

[46] B. Girault, P. Gonçalves, and É. Fleury, "Translation on graphs: An isometric shift operator," *IEEE Signal Processing Letters*, vol. 22, pp. 2416–2420, Dec 2015.

[47] N. Perraudin, A. Loukas, F. Grassi, and P. Vandergheynst, "Towards stationary time-vertex signal processing," in *2017 IEEE International Conference on*

Acoustics, Speech and Signal Processing (ICASSP), pp. 3914–3918, March 2017.

[48] A. Papoulis and S. Pillai, *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill series in electrical engineering: Communications and signal processing, McGraw-Hill, 2002.

[49] E. Isufi, A. Loukas, A. Simonetto, and G. Leus, "Separable autoregressive moving average graph-temporal filters," in *Proc. 24th (EUSIPCO)*, pp. 200–204, 2016.

[50] E. Isufi, A. Loukas, N. Perraudin, and G. Leus, "Forecasting time series with VARMA recursions on graphs," *IEEE Trans. Signal Process.*, vol. 67, no. 18, pp. 4870–4885, 2019.

[51] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 2.1," Mar. 2014.

[52] M. Grant and S. Boyd, "Graph implementations for nonsmooth convex programs," in *Recent Advances in Learning and Control*, pp. 95–110, Springer-Verlag Limited, 2008.

[53] H. Lütkepohl, *New introduction to multiple time series analysis*. Springer, 2005.