# Graph-time signal processing

## Filtering and sampling strategies

Isufi, Elvin

**DOI**
10.4233/uuid:e52cc182-457c-4687-baee-d0f72af36950

**Publication date**
2019

**Document Version**
Final published version

**Citation (APA)**
Isufi, E. (2019). *Graph-time signal processing: Filtering and sampling strategies*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:e52cc182-457c-4687-baee-d0f72af36950

**Important note**
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

# GRAPH-TIME SIGNAL PROCESSING

FILTERING AND SAMPLING STRATEGIES

# GRAPH-TIME SIGNAL PROCESSING

## FILTERING AND SAMPLING STRATEGIES

**Proefschrift**

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof. dr. ir. T.H.J.J. van der Hagen
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op vrijdag 28 januari 2019 om 12:30 uur

door

**Elvin ISUFI**

Master of Science in Electrical and Telecommunications Engineering,
University of Perugia, Perugia, Italy,
geboren te Vlore, Albania.

Dit proefschrift is goedgekeurd door de

promotor: Prof. dr. ir. G. Leus

Samenstelling promotiecommissie:

| | |
|---|---|
| Rector Magnificus, | voorzitter |
| Prof. dr. ir. G. Leus, | Technische Universiteit Delft, promotor |
| *Onafhankelijke leden:* | |
| Prof. dr. P. Borgnat | École Normale Supérieure de Lyon, France |
| Dr. ir. R. C. Hendriks, | Technische Universiteit Delft |
| Prof. dr. A. Ribeiro | University of Pennsylvania, USA |
| Prof. dr. ir. A. J. van der Veen | Technische Universiteit Delft |
| Prof. dr. ir. P. F. A. Van Mieghem | Technische Universiteit Delft |
| *Overige leden:* | |
| Prof. dr. P. Banelli | University of Perugia, Italy |


*Keywords:* Graph signal processing, graph filters, graph-time signal processing, graph-time filters, Laplacian, network theory, FIR, ARMA, observability, linear system on graphs, Kalman filter, sampling theory, graph sampling, sparse sensing.



ISBN 978-94-028-1353-1

An electronic version of this dissertation is available at
`http://repository.tudelft.nl/`.

*Man deals with knowledge and science throughout his life.*
*In childhood he learns them, applies them in youth,*
*and teaches them in old age.*

Sami Frashëri

To my parents – my life teachers.

# CONTENTS

# I

# PROLOGUE

# 1

# INTRODUCTION

*When writing something, read and adjust it many times,*
*so that each cluster of letters sounds like a piece of music.*

Faik Konica

Big data comes with big challenges. To deal with its large volume and efficiently use resources such as time, computational power, and storage, it demands novel tools for basic operations such as acquisition, processing, and analysis. Among several approaches adopted in the signal processing community, including compressive sensing and sampling, tensor decomposition and distributed signal processing, this thesis deals with graph signal processing (GSP) which distinguishes itself by exploiting the underlying structure inherent to the data. This structure may be implicit, like data correlations and dependencies, or explicit, like traffic data relative to road networks.

This thesis provides fundamental contributions to the field of GSP and addresses the aforementioned tasks of data acquisition, analysis, and processing. The proposed findings expand our knowledge on the importance of the underlying data structure and show that a substantial performance gain can be achieved when that structure is exploited. This chapter starts with the concept of data living on top of networks along with motivating the importance of the underlying connections. It will then go on to the scope of this thesis and provide an outline of the presented work. At the end, the thesis' main contributions are detailed.

## 1.1. DATA LIVING ON TOP OF NETWORKS

Today, we live in a highly interconnected yet sensitive world. A classic example is the airport network, where a single flight delay or flight cancellation directly incurs several consequences (e.g., a cascade flight delay and/or a cascade flight cancellation), often problematic on a wider scale[1]. The same sort of influence can be observed in price fluc-

[1] According to [2], in the United States the airlines' costs due to flight delays amount to $22 billion per year.

tuations in financial networks, political orientation in blog networks, gossip propagation in social networks and traffic congestion in road networks. Even at a more microscopic scale, correlated interconnections and dependencies are present in neuron-to-neuron or gene-to-gene interactions [3]. These interconnections provide a meaningful structure to the data, which is not entirely exploited by standard signal processing tools. Therefore, for the tasks of delay prediction in airline networks or gossip propagation in social networks, for instance, novel processing tools that incorporate the underlying interconnections in the solution are required.

To date, a number of studies established the efficacy of graphs as a useful mathematical tool to concisely capture and represent the underlying (often hidden) structure between data, see [4–6], and references therein. While the graph structure is an interesting object of study by itself, this work focuses on the data that reside on top of the graph. In the airport network example, the graph captures the airline infrastructure with the airports being the vertices of the graph and the edges indicating the presence of a direct flight between two different airports. The object of study, i.e., the data on top of the graph, could, for instance, be the average delay per flight in each airport. Then, we might be interested in analyzing the delay spread over this network, or the consequences of a fallen edge (e.g., a flight cancellation). This information allows then the implementation of local proactive policies for air traffic control, and thus potentially reduces the overall delay in the network.

A useful example that illustrates the concept of data on top of networks is depicted in Figure 1.1. Here, the graph represents a sensor network with sensors being represented by the graph vertices and the neighborhood information by the graph edges (i.e., the line connecting two of such nodes). The data on top of this network commonly referred to as *the signal on top of the graph*, or *the graph signal* consists of the noisy temperature measurements in a particular region. A common task of interest is to locally denoise the signal by allowing sensors to exchange information only with their direct neighbors.

The research field that approaches big data by incorporating their underlying structure represented by a graph is known as *graph signal processing* [9, 10]. The workhorse of GSP is the notion of signal variation over the graph, which allows us now to extend fundamental signal processing techniques to the graph setting. The most notable are the frequency analysis of graph signals, graph filtering, and graph signal sampling.

This thesis adds to the field of GSP and concerns the question how the underlying structure, inherent to the data, can be exploited to develop novel tools for processing signals that reside on top of networks. Inspired by the tight connection between the graph structure and the graph signal, we propose a series of basic building blocks to answer this fundamental research question. The proposed approaches are accompanied by solid theoretical performance guarantees, and we illustrate that exploring the *graph topology-graph signal connection* yields a performance gain over alternative solutions that ignore this coupling.

The upcoming sections briefly introduce the thesis framework and provide a glimpse of the arguments treated in the following chapters.

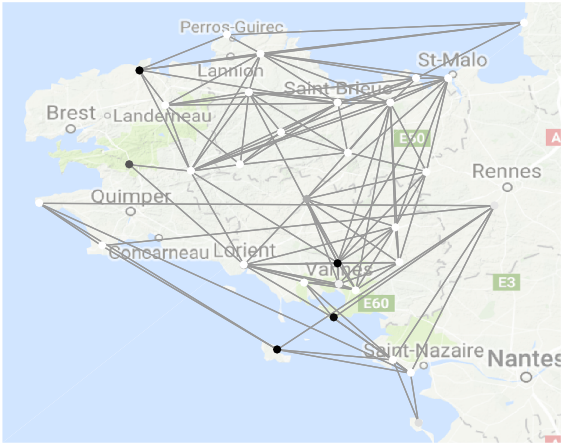



Figure 1.1: Illustration of the Molene temperature sensor network [7]. The graph is built with the approach of [8]. The vertices (black and white) represent the sensors, while the edges represent their neighborhood connections. A GSP task consists of cleaning noisy measurements at the black nodes by local communications, i.e., with connecting nodes.

## 1.2. FILTERING GRAPH SIGNALS

Filtering is one of the basic operations in signal processing. It concerns the preservation of only useful spectral information about the signal. In sensor networks, distributed consensus [11, 12], i.e., finding the network average by local communications with neighbors, may be interpreted as a distributed low-pass spatial filtering of graph signals [13, 14] over the network. However, the reader should note there is a distinction between the notion of a graph in distributed signal processing and GSP. In GSP, when we talk about a graph topology, we mean the signal graph, i.e., the graph that explains the structure. For instance, in the airport network, the signal graph consists of the airline infrastructure which influences the delay on top of this network. The actual graph used for distributed data exchange named the communication graph (e.g., between computers in the different airports) could be similar or not, but we will not cover this aspect in this thesis. We assume that the graph explaining the signal is also used for communications. Hence, in the sequel, the term graph will always refer to the *signal graph* which equals the communication graph.

With this analogy between consensus and filtering of graph signals, we pose our first research question:

(Q1) *How can a sensor network perform more involved distributed filtering tasks than simple averaging by taking into account the underlying data structure?*

Starting from the next chapter, we will give a more formal and detailed answer to this

question, but for now, let us give an intuitive answer with the following example.

**Example 1.1.** (Graph signal denoising) *Consider a sensor network observing a single temporal snapshot of some temperature measurements. We would like the network to estimate the original signal from these measurements by local communications with direct neighbors. Such a scenario is depicted in Figure 1.2, where the top left image depicts the ground truth signal, the top right image the noisy sensor measurements, and the bottom image the cleaned signal. As it can be observed from Figure 1.2 (a), the ground truth signal has similar values among vertices that share an edge. This means that between neighboring sensors we expect the measured signals (e.g., Figure 1.2 (b)) to have dissimilar values only if these sensors are highly corrupted by noise. An efficient noise reduction algorithm will then carefully filter the useless part of the signal by simply taking into account the information provided by its direct neighbors.*

The above denoising example, which makes use of prior information about the original signal (e.g., adjacent vertices share similar values) is well known in the literature (prior to the formalization of GSP) and is commonly referred to as Tikhonov regularization on graphs [16–18]. In Section 2.3.6, we formalize this problem and show how it can be cast in a GSP perspective.

In general, to the distributed filtering operation in (Q1), we will commonly refer to as distributed graph filtering. A significant aspect of distributed graph filters is the number of communication rounds that adjacent nodes need to perform for solving a given task. In analogy with classical signal processing, we distinguish two types of graph filters: *i*) *finite impulse response* (FIR) graph filters, i.e., a network operation that leads to the designed output in finite time and *ii*) *infinite impulse response* (IIR) graph filters, i.e., a network operation that leads to the designed output in infinite time. With these definitions in place, we can pose two relative subquestions of (Q1):

(Q1.1) *How can distributed graph signal processing tasks be performed with FIR graph filters?*

(Q1.2) *How can distributed graph signal processing tasks be performed with IIR graph filters?*

Chapters 3 and 4 in Part II will respectively formalize the concepts of FIR and IIR graph filters to provide a thorough answer to the above questions.

## 1.3. FILTERING GRAPH SIGNALS IN DYNAMIC ENVIRONMENTS

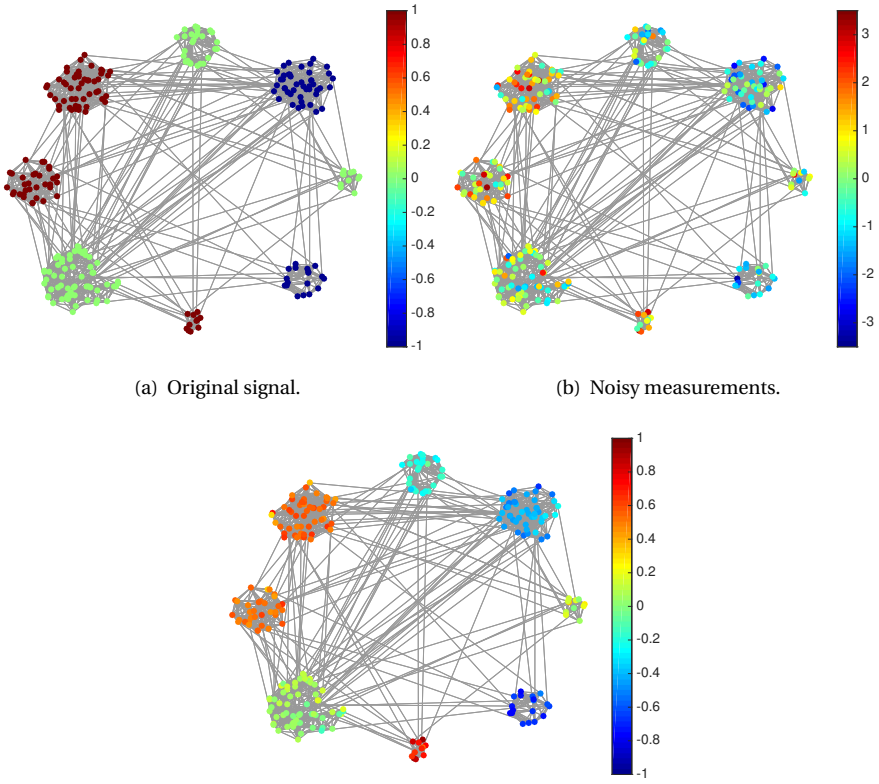
Along with the benefits of local communications that distributed graph filters bring, a crucial challenge is the filters' behavior in dynamic environments, when variations in the graph topology and graph signal occur. Whilst some research has been carried out for distributed consensus [19–21], there is still little understanding on how these dynamics affect the more involved graph filtering operations. The latter leads to our second research question:

(Q2) *What are the implications of dynamic changes, in the graph topology and graph signal, on the graph filter output?*

The answer to this question will be our central topic in Part III. Specifically, in Chapter 6 we address deterministic changes (e.g., moving sensors) and answer the subquestion:

(Q2.1) *How do graph filters behave when the input signal and the graph topology change deterministically over time?*

On the other hand, to address random fluctuations in the graph topology (e.g., link failures) and graph signal (e.g., noise corrupted signals), in Chapter 7 we shift our focus to changes of stochastic nature and answer the subquestion:




(a) Original signal.

(b) Noisy measurements.

(c) Cleaned signal.

Figure 1.2: Illustration of a sensor network that contains different versions of the graph signal. The vertex color indicates the graph signal value. We observe a large noise removal (the color of the nodes within a cluster is more similar in c) than in b)) at the expense of some energy spreading between adjacent nodes. The figures are generated with the GSP toolbox [15].

(Q2.2) *What are the statistical properties of the filter output when the graph topology and the graph signal are random processes?*

## 1.4. OBSERVING TIME-VARYING GRAPH PROCESSES

The interpolation of missing data on a network from partial measurements is another interesting and useful task that alleviates some of the costs of structured big data. Let us illustrate this with the following example.

**Example 1.2.** (Graph signal interpolation.) *Given the Molene temperature sensor network in Figure 1.1, we consider that due to energy constraints, only a subset of sensors is capable to collect measurements (e.g., the black nodes). Then, the task to address consists of interpolating (reconstructing) the missing values (e.g., at the white nodes) from the available ones. One straightforward distributed approach may consist of setting the value of the white sensors to the average of their neighbors. This goes in line with the prior assumption that adjacent sensors share similar values. More sophisticated approaches will exploit Tikhonov regularization or other prior information to accurately determine the missing values.*

A more complex scenario occurs when the graph signal evolves with a predefined model over time. Concisely, we will refer to this as a graph process. We then aim at estimating the initial graph signal realization on all nodes from only a few sampled vertices. Let us illustrate this scenario with the following social network example.

**Example 1.3.** (Observability of graph processes.) *Let us consider a social network with users being the vertices of the graph and edges representing user connections, e.g., friendships or followers. The graph signal is considered to be an opinion on a particular topic like politics, sports, or art. Due to fellow influence, it is reasonable to assume that user opinions will change over time (e.g., from total disinterest in the topic to partial involvement). Subsequently, the observability of the opinion signal relates to finding the initial opinion of all users from a survey performed on a few candidate users at different time instances.*

With this in place, we are now able to pose our third research question:

(Q3) *Under which conditions of the graph topology and the graph process can we estimate the initial network state from a subset of vertices?*

The above question adds to one of the most interesting and elaborated problems in linear system theory: the observability of a linear system. Nevertheless, we are now interested in finding the conditions that both the graph topology and the graph process must satisfy to ensure observability. Moreover, we would like to relate these conditions to the constraint of collecting limited measurements. In Section 8.3, we formalize this problem in a GSP perspective and provide an elegant answer to the above question.

## 1.5. TRACKING TIME-VARYING GRAPH PROCESSES

While observability concerns discovering the initial state of the graph process, in dynamics over networks we are also interested in tracking the temporal evolution of the graph process. With respect to the social network example, this task consists of tracking the user opinions over time. Tracking on, and with sensor networks, has already been investigated in [24–26] and references therein. However, to the best of our knowledge, the underlying network structure is only exploited as a tool to accomplish the tracking goal. Here, we pursue a different path, where the graph structure is considered as an intrinsic part of the problem and tightly relates the tracking performance with both the graph topology and the graph process. In addition, we will exploit process priors with respect to the graph topology to introduce the concept of Kalman filtering (KF) over graphs for tracking from a subset of vertices. It is then crucial to ask:

(Q4) *Which are the conditions that the graph topology and the graph process must satisfy such that Kalman filtering can be employed to track network dynamics from a subset of nodes?*

A detailed answer to this question is provided in Section 8.4, where the involvement of the KF leads to the optimal tracking performance. Additionally, as considered for the adaptive algorithms and the observability study, we would also like to carefully pick the right vertices such that a target mean-square error (MSE) tracking performance is guaranteed.

## 1.6. THESIS OUTLINE AND CONTRIBUTIONS

This thesis is organized into five main parts covering ten chapters. In the next section, we elaborate on each chapter and show the relative contributions, while a complete list of related references and other contributions is shown in Section 1.6.2.

For the sake of obtaining a self-explanatory document, the treated arguments are elaborated in sufficient detail to be followed by an audience with a general signal processing and linear algebra background. The introduced topics are structured to improve readability, rather than respecting their publication time. Interested readers who require more details are redirected to the related works. The abbreviations and notations used throughout the thesis are provided on pages 191 and 193, respectively.

### 1.6.1. THESIS OUTLINE

◇ *Prologue—Part I.* The remainder of this part consists of Chapter 2, which covers the necessary background concepts that will be exploited throughout the thesis.

**Graph signal processing—Chapter 2.** This chapter consists of some background information about the research field of GSP, which forms the backbone of this thesis. We first formulate the graph structure in linear algebra terms, by developing the notion

of *the graph shift operator* as a matrix that captures the graph connectivity. We subsequently focus our attention to the GSP direction, where the notion of *signal variation over a graph* is introduced and formalized. The latter arguments lead to the definition of the graph Fourier transform (GFT), which analogously to the classical temporal Fourier transform introduces the concept of frequency in the graph setting. This specific definition of the GFT allows us to characterize the *bandwidth* of a graph signal and introduce the concept of *graph filtering*. The chapter is concluded with the concept of *wide sense stationarity on graphs*.

◇ *Graph Filtering—Part II*. This part concerns the first topic we listed in Part I, i.e., the graph filters, and answers the research question (Q1) over Chapters 3 and 4. Specifically, Chapter 3 is dedicated to FIR graph filters, i.e., (Q1.1) and includes also our contribution about distributed edge-variant FIR graph filters. Chapter 4 introduces the autoregressive moving average (ARMA) recursions as algorithms to implement distributed IIR graph filters and answers research question (Q1.2).

**Finite impulse response graph filtering—Chapter 3.** With the formalization of the graph frequency content of a graph signal in Chapter 2, in this chapter, we take a step further by analyzing the simplest type of graph filters, i.e., *the FIR graph filters*. As their name suggests, these are filters that act on the graph spectrum and are characterized by a finite impulse response in the vertex domain. Being an operation that acts over a network, we dedicate particular attention to FIR graph filters that can be *implemented distributively*. Further, we show how such filters can be designed. This chapter also contains our first contribution [cf. Section 3.4], where we propose a novel distributed algorithm to implement FIR graph filters. The proposed approach yields notable improvements with respect to prior art solutions and candidates itself as a strong building block in the field of GSP.

**Infinite impulse response graph filtering—Chapter 4.** This chapter extends the distributed graph filtering concept, from the FIR graph filters introduced in Chapter 3, to the class of IIR graph filters. As their name suggests, these filters are characterized by an infinite impulse response in the vertex domain. After introducing the IIR graph filters and their implementation structures, we show how these filters can be implemented distributively with *ARMA recursions on graphs*. A detailed mathematical analysis illustrates the capability of ARMA graph filters to provide exact solutions to some graph signal processing tasks such as graph signal denoising, and diffusion.

◇ *Graph-Time Filtering—Part III*. The task addressed in this part is studying the behavior of the formerly distributed graph filters in dynamic environments. This part is composed of Chapters 5, 6 and 7 and contains our pioneering work on the temporal extension of graph filters. In Chapter 5 we reformulate the problem of graph-time processing and graph-time filtering. The answer to the research question (Q2) is spanned over Chapters 6 and 7. The former chapter concerns the subquestion relative to deterministic topology changes (Q2.1), whilst the latter addresses stochastic topology changes, i.e., (Q2.2).

**Graph-time signal processing—Chapter 5.** This chapter lays the basic groundwork for the extension of GSP to graph-time signal processing (GTSP). It paves the way for the upcoming two chapters by formalizing the concepts of the *graph-time shift operator*,

*graph-time Fourier transform*, and *graph-time filtering*.

**Deterministic analysis of graph-time filtering—Chapter 6.** This chapter analyzes the distributed graph filters of Chapters 3 and 4 when the graph signal and(or) the graph topology change(s) deterministically over time. Here we also introduce the FIR and ARMA *graph-time filters*, i.e., filters that process jointly the graph and temporal spectrum of time-varying graph signals. We show that the introduced filters enjoy an efficient *distributed implementation*, pledging themselves as potential tools for a more incisive analysis of time-varying graph signals.

**Statistical analysis of graph-time filtering—Chapter 7.** This chapter expands the analysis of graph filters to a time-varying stochastic environment. Specifically, we consider the behavior of the filter output when the graph signal and(or) the graph topology change(s) randomly over time. We perform a statistical analysis of the filtering output and characterize the influence of the graph topology and(or) the graph signal statistics on the filter behavior. We conclude the chapter by suggesting a novel approach that *exploits stochasticity* to alleviate the costs of the graph filters introduced in Chapters 3-4.

◇ *Graph-Time Sampling—Part IV*. This part consists of solely Chapter 8 and builds on the extension of GSP to time-varying graph signals by introducing sampling strategies for time-varying graph processes. More specifically, it answers the last two research questions (Q3) and (Q4) by providing sampling conditions for the tasks of observing and tracking a time-varying graph process.

**Observing and tracking graph processes—Chapter 8.** This chapter develops graph sampling strategies for observing and tracking time-varying graph processes from a subset of nodes. Our inspiration comes from the from the applications of graph signal diffusion and wave propagation on graphs. We first show how the aforementioned tasks can be formulated as state-space models on graphs. Then, we introduce the concept of graph process observability from a few collected measurements. Further, we introduce Kalman filtering on graphs, as an optimal algorithm to track changes in the graph signal, again from only a few available nodes. We provide theoretical conditions that the selected subset of nodes should satisfy to guarantee observability and tracking. In addition, we perform a detailed MSE analysis on the observability/tracking performance to highlight the role played by the different actors like the graph topology, the graph process nature with respect to the underlying graph and the sampled set.

◇ *Epilogue—Part V*. This part wraps up the thesis with the concluding Chapter 9.

**Conclusions and future research directions—Chapter 9.** This chapter summarises the thesis contributions and draws the respective conclusions. In addition, future research directions for each of the treated arguments are also proposed in this chapter.

### 1.6.2. LIST OF PUBLICATIONS AND OTHER CONTRIBUTIONS

To summarize the introductory chapter, the work developed during the Ph.D. period resulted in the following peer-reviewed journal and conference papers.

1

## Thesis related contributions

**Journal papers**

**J1** **E.Isufi**, P. Banelli, P. Di Lorenzo and G. Leus, "Observing and Tracking Bandlimited Graph Processes", *submitted to IEEE Transactions on Signal Processing*, Sep. 2018.

**J2** **E. Isufi**, A. Loukas, A. Simonetto and G. Leus,"Filtering Random Graph Processes Over Random Time-Varying Graphs ", *IEEE Transactions on Signal Processing*,vol.65 (16), pages 4406-4421, 2017.

**J3** **E. Isufi**, A. Loukas, A. Simonetto and G. Leus, "Autoregressive Moving Average Graph Filtering", *IEEE Transactions on Signal Processing*, vol.67 (2), pages 274-288, 2017.

**Conference papers**

**C1** **E.Isufi**, P. Banelli, P. Di Lorenzo and G. Leus, "Observing Bandlimited Graph Processes from Subsampled Measurements", Asilomar Conference on Signals, Systems and Computations, Pacific Grove, USA, Oct. 2018.

**C2** M. Coutino, **E. Isufi** and G. Leus, "Distributed Edge-Variant Graph Filters", IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), Curacao, Dutch Antilles, Dec. 2017. **(best student paper award, 3rd ranked)**

**C3** **E. Isufi**, A. Loukas and G. Leus, "Autoregressive Moving Average Graph Filters - A Stable Distributed Implementation", IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), New Orleans, USA, Mar. 2017.

**C4** **E. Isufi** and G. Leus, "Distributed Sparsified Graph Filters for Denoising and Diffusion Tasks", IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), New Orleans, USA, Mar. 2017.

**C5** **E. Isufi**, P. Banelli and G. Leus, "2-Dimensional Finite Impulse Response Graph-Temporal Filters", IEEE Global Conference on Signal and Information Processing (GlobalSIP), Washington DC, USA, Dec. 2016.

**C6** **E. Isufi**, A. Loukas, A. Simonetto and G. Leus, "Separable Autoregressive Moving Average Graph-Temporal Filters", EURASIP European Signal Processing Conference (EUSIPCO), Budapest, Hungary, Aug. 2016.

**C7** **E. Isufi**, A. Simonetto, A. Loukas and G. Leus, "Stochastic Graph Filtering on Time-Varying Graphs", IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), Cancun, Mexico, Dec. 2015.

## Other contributions

**Journal papers**

**J4** **E. Isufi**, A. Loukas, N. Perraudin and G. Leus, "Forecasting Time Series with VARMA Recursions on Graphs", *submitted to IEEE Transactions on Signal Processing*, Oct. 2018.

**J5** M. Coutino, **E. Isufi** and G. Leus, "Advances in Distributed Graph Filtering", *submitted to IEEE Transactions on Signal Processing*, Jul. 2018.

**J6** J. Liu, **E. Isufi** and G. Leus, "Filter Design for Autoregressive Moving Average Graph Filters", *to appear in the IEEE Transactions on Signal and Information Processing over Networks*, 2018.

**J7** **E. Isufi**, A. S. U. Mahabir and G. Leus, "Blind Graph Topology Change Detection", *IEEE Signal Processing Letters*, vol.25 (5), pages 655-659, 2018.

**J8** P. Di Lorenzo, P. Banelli, **E.Isufi**, S. Barbarossa, and G. Leus, "Adaptive Graph Signal Processing: Algorithms and Optimal Sampling Strategies," *IEEE Transactions on Signal Processing*, vol.66 (13), pages 3584-3598, 2018.

**J9** **E. Isufi**, H. Dol and G. Leus, "Advanced Flooding-Based Routing Protocols for Underwater Sensor Networks", *EURASIP Journal on Advances in Signal Processing*, 2016.1 (2016) : 52.

**Conference papers**

**C8** M. Coutino, **E. Isufi**, T. Maehara and G. Leus, "State-Space Based Network Topology Identification", submitted to the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, United Kingdom, May 2019.

**C9** M. Coutino, **E. Isufi**, T. Maehara and G. Leus, "On the Limits of Finite Time Distributed Consensus through Graph Filters", IEEE Asilomar Conference on Signals, Systems and Computations, Pacific Grove, USA, Oct. 2018. **(invited paper)**

**C10** **E.Isufi**, P. Di Lorenzo, P. Banelli and G. Leus, "Distributed Wiener-Based Reconstruction of Graph Signals", IEEE Statistical Signal Processing Workshop (SSP), Freiburg, Germany, Jun. 2018.

**C11** F. Gamma, **E. Isufi**, G. Leus and A. Ribeiro, "Control of Graph Signals Over Random Time-Varying Graphs", IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Calgary, Canada, Apr. 2017.

**C12** A. Loukas, **E. Isufi** and N. Perraudin, "Predicting the evolution of stationary graph signals", IEEE Asilomar Conference on Signals, Systems and Computations, Pacific Grove, USA, Oct.-Nov. 2017. **(invited paper)**

**C13** P. Di Lorenzo, **E. Isufi**, P. Banelli, S. Barbarossa and G. Leus, "Distributed Recursive Least Squares Strategies for Adaptive Reconstruction of Graph Signals", EURASIP European Signal Processing Conference (EUSIPCO), Kos, Greece, Aug.-Sept. 2017.

**C14** J. Liu, **E. Isufi** and G. Leus, "Autoregressive Moving Average Graph Filter Design", 5th IEEE Global Conference on Signal and Information Processing (GlobalSIP), Montreal, Canada, Nov. 2017.

**C15** J. Liu, **E. Isufi** and G. Leus, "Autoregressive Moving Average Graph Filter Design", 6th Joint WIC/IEEE Symposium on Information Theory and Signal Processing in the Benelux, Louvain-la-Neuve, Belgium, May 2016.

**C16** **E. Isufi**, H. Dol and G. Leus, "Network Coding for Flooding-Based Routing in Underwater Sensor Networks", ACM International Conference on Underwater Networks and Systems (WUWNET'14), Rome, Italy, Nov. 2014.

## FURTHER READING


[1] G. B. Giannakis, R. Cendrillon, V. Cevher, A. Swami, and Z. Tian, *Introduction to the issue on signal processing for big data,* IEEE Journal of Selected Topics in Signal Processing **9**, 583 (2015).

[2] J. Rapajic, *Beyond airline disruptions* (Ashgate Publishing, Ltd., 2009).

[3] M. Bansal, V. Belcastro, A. Ambesi-Impiombato, and D. Di Bernardo, *How to infer gene networks from expression profiles,* Molecular systems biology **3**, 78 (2007).

[4] M. Newman, *Networks: An Introduction* (Oxford university press, 2010).

[5] D. Easley and J. Kleinberg, *Networks, crowds, and markets: Reasoning about a highly connected world* (Cambridge University Press, 2010).

[6] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, *A survey on sensor networks,* IEEE Communications magazine **40**, 102 (2002).

[7] *Molene weather dataset,* `https://donneespubliques.meteofrance.fr/donnees_libres/Hackathon/RADOMEH.tar.gz` (2017).

[8] S. P. Chepuri, S. Liu, G. Leus, and A. O. Hero, *Learning sparse graphs under smoothness prior,* in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on* (IEEE, 2017) pp. 6508–6512.

[9] 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 **30**, 83 (2013).

[10] A. Sandryhaila and J. M. Moura, *Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure,* IEEE Signal Processing Magazine **31**, 80 (2014).

[11] W. Yu, G. Chen, Z. Wang, and W. Yang, *Distributed consensus filtering in sensor networks,* IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) **39**, 1568 (2009).

[12] W. Ren and R. W. Beard, *Distributed consensus in multi-vehicle cooperative control* (Springer, 2008).

[13] A. Sandryhaila, S. Kar, and J. M. Moura, *Finite-time distributed consensus through graph filters,* in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on* (IEEE, 2014) pp. 1080–1084.

[14] S. Segarra, A. Marques, and A. Ribeiro, *Optimal graph-filter design and applications to distributed linear network operators,* IEEE Transactions on Signal Processing (2017).

[15] N. Perraudin, J. Paratte, D. Shuman, L. Martin, V. Kalofolias, P. Vandergheynst, and D. K. Hammond, *Gspbox: A toolbox for signal processing on graphs,* arXiv preprint arXiv:1408.5781 (2014).

[16] C. Groetsch, *The theory of Tikhonov regularization for Fredholm equations,* Boston Pitman Publication (1984).

[17] G. H. Golub, P. C. Hansen, and D. P. O'Leary, *Tikhonov regularization and total least squares,* SIAM Journal on Matrix Analysis and Applications **21**, 185 (1999).

[18] A. J. Smola and R. Kondor, *Kernels and regularization on graphs,* in *COLT*, Vol. 2777 (2003) pp. 144–158.

[19] F. Xiao and L. Wang, *Asynchronous consensus in continuous-time multi-agent systems with switching topology and time-varying delays,* IEEE Transactions on Automatic Control **53**, 1804 (2008).

[20] T. Li and J.-F. Zhang, *Consensus conditions of multi-agent systems with time-varying topologies and stochastic communication noises,* IEEE Transactions on Automatic Control **55**, 2043 (2010).

[21] W. Ren, *Multi-vehicle consensus with a time-varying reference state,* Systems & Control Letters **56**, 474 (2007).

[22] S. Joshi and S. Boyd, *Sensor selection via convex optimization,* IEEE Transactions on Signal Processing **57**, 451 (2009).

[23] S. P. Chepuri and G. Leus, *Sparse sensing for statistical inference,* Foundations and Trends® in Signal Processing **9**, 233 (2016).

[24] R. R. Brooks, P. Ramanathan, and A. M. Sayeed, *Distributed target classification and tracking in sensor networks,* Proceedings of the IEEE **91**, 1163 (2003).

[25] H.-T. Kung and D. Vlah, *Efficient location tracking using sensor networks,* in *Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE*, Vol. 3 (IEEE, 2003) pp. 1954–1961.

[26] T. He, P. Vicaire, T. Yan, L. Luo, L. Gu, G. Zhou, R. Stoleru, Q. Cao, J. A. Stankovic, and T. Abdelzaher, *Achieving real-time target tracking usingwireless sensor networks,* in *Real-Time and Embedded Technology and Applications Symposium, 2006* (IEEE, 2006) pp. 37–48.

# 2

# GRAPH SIGNAL PROCESSING

*We ourselves feel that what we are doing is just a drop in the ocean.*
*But the ocean would be less because of that missing drop.*

Anjezë Gonxhe Bojaxhiu, a.k.a. Mother Teresa

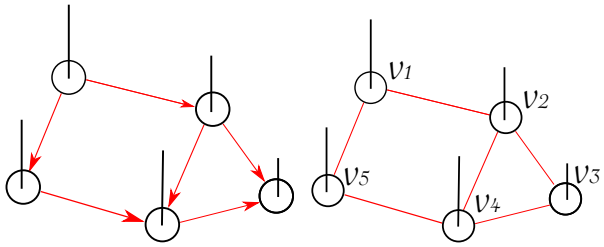
The goal of this chapter is twofold. The first is to provide the reader the necessary background information about GSP that will be called throughout the thesis. The second is to introduce the notation and terminology that bridge the high-level discussion of Chapter 1 to the more detailed mathematical formulation of the succeeding chapters.

Our analysis mainly follows the graph Laplacian-based approach summarized in [1]. For a more in-depth analysis, we also suggest [2], as well as the works by Sandryhaila and Moura [3–5] which treat the arguments from the graph adjacency matrix viewpoint.

This chapter is organized as follows. Section 2.1 briefly recalls the background works in GSP and clarifies the philosophy adopted in this thesis. Section 2.2 then introduces the graph as a mathematical tool to express the interconnections/interdependencies between data in linear algebra terms. The spectral analysis of graph signals is shown in Section 2.3, where the GFT is formalized in Section 2.3.2 and the graph filters in Section 2.3.5. Section 2.4 introduces the concept of stationarity for graph signals and Section 2.5 concludes the chapter.

## 2.1. INTRODUCTION

Being able to formulate a mathematical theory that captures the data interrelations allows us to develop signal processing techniques for analyzing and processing signals that reside on top of networks. In this chapter, we show that graph theory proves to be a useful tool in this regard, and it pledges itself as a standing platform for validating the variation of graph signals [2, 6–8]. This variation on graphs can then be explored to formulate a spectral analysis theory for graph signals, and to advance the concept of the




(a) Directed graph. (b) Undirected graph.

Figure 2.1: Illustration of two different graphs with the respective signal on top. The height of the bar represents the signal value.

graph Fourier transform (GFT). The latter, similar to the frequency analysis of temporal and spatial (image) signals, will now add a harmonic flavour to signals on graphs.

There are two distinct, yet complementary, philosophies that approach GSP from different angles: the graph adjacency matrix-based approach, and the graph Laplacian matrix-based approach. As its name suggests, the former approach builds on the graph adjacency matrix and extends algebraic signal processing [9, 10] from time and space to irregular non-Euclidean domains, see [4, 5]. In this philosophy, the notion of total signal variation is exploited to formalize the signal variation over graphs. The graph Laplacian matrix-based approach leverages the graph spectral theory [6] to extend the Fourier analysis to non-Euclidean spaces, see [1, 2, 7]. In this context, the graph Laplacian matrix theory[1] and the concept of signal smoothness over graphs are exploited to introduce the harmonic analysis for graph signals. While these approaches differ in derivation and interpretation, they both aim at the same thing: The expansion of the graph signal in terms of the oscillating modes of the graph. As we shall see next, these "oscillating modes" of the graph turn out to be the eigenvectors of the considered matrix.

In this thesis, we will develop our theory following the graph Laplacian matrix-based approach. However, as we point out next, the proposed methods can be extended with a few appropriate changes to fit the graph adjacency matrix philosophy as well.

## 2.2. GRAPHS AS A TOOL TO CAPTURE INTERCONNECTIONS

A graph is denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, \ldots, v_N\}$ is the set of $N$ vertices (or nodes) and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the edge set containing all tuples $e_{i,j} = (v_i, v_j)$ for which nodes $v_i$ and $v_j$ are connected. We consider there are $M$ such edges, i.e., $|\mathcal{E}| = M$. A graph is said to be directed if its edges present a direction orientation, meanwhile, $\mathcal{G}$ is said to be undirected if there is no edge orientation for all tuples $(v_i, v_j) \in \mathcal{E}$. Figure 2.1 illustrates this distinction. Throughout this thesis, we consider graphs that satisfy the following:

**Assumption 2.1.** (Considered class of graphs.) *We consider graphs that are connected, without self-loops (i.e., there are no edges of the form $(v_i, v_i)$), and undirected.*

[1] In literature this approach is also encountered as "the graph Laplacian operator theory".

The biggest loss of generality of the above assumption is with respect to directed graphs. In fact, if $\mathcal{G}$ is composed of $R$ components $\mathcal{G}_1, \ldots, \mathcal{G}_R$, we can treat each component as a smaller graph and analyze it separately. The restriction to undirected graphs, which will be more clear in Section 2.3, is imposed by the adopted approach of graph Laplacian matrix theory[2]. However, in practice, it is often sufficient to consider undirected graphs, which have been proven useful on a wide range of applications, see [1, 13] and references therein.

Given that $\mathcal{G}$ is undirected, the weighted graph adjacency matrix $\mathbf{W}$ is an $N \times N$ symmetric matrix with $W_{i,j} = W_{j,i} > 0$ being the weight of the edge $e_{i,j} = (v_i, v_j) \in \mathcal{E}$. $W_{i,j} = 0$ indicates that nodes $v_i$, $v_j$ are not connected. The node degrees are contained in the diagonal matrix $\mathbf{D}$ with as its $i$th diagonal element

$$D_{i,i} = \sum_{j=1}^{N} W_{i,j} \tag{2.1}$$

representing the sum of all edge weights connected to $v_i$. The combinatorial graph Laplacian (for short Laplacian) matrix[3] is defined as [6]

$$\mathbf{L} = \mathbf{D} - \mathbf{W}, \tag{2.2}$$

whereas the normalized Laplacian matrix is defined as

$$\mathbf{L}_{\mathrm{n}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}. \tag{2.3}$$

With this in place, we distinguish two different streams on how to build a graph:

**Physical graphs.** In this category we group all graphs that have a physical meaning, e.g., $i$) road networks with cross-roads being the nodes and streets representing the graph edges; $ii$) sensor networks, where the nodes represent the sensors, while the graph edges match the data exchange links between sensors; $iii$) airport networks, where the nodes in $\mathcal{V}$ represent the different airport terminals and $\mathcal{E}$ contains the tuples $(v_i, v_j)$ if there is a flight connection between airports $v_i$ and $v_j$; $vi$) smart grid networks with nodes denoting the load stations and edges the power lines. A standard choice for the weighted adjacency matrix then is a Gaussian kernel [6, 14], i.e,

$$W_{i,j} = \begin{cases} \exp\left(-\frac{[\mathrm{dist}(v_i,v_j)]^2}{2\theta^2}\right) & \text{if } \mathrm{dist}(v_i, v_j) \leq \gamma \\ 0 & \text{otherwise,} \end{cases} \tag{2.4}$$

for some parameters $\theta$ and $\gamma$. In (2.4), $\mathrm{dist}(v_i, v_j)$ may be the physical distance between the vertices $v_i$ and $v_j$, or the Euclidean distance between two feature vectors describing $v_i$ and $v_j$.

**Learned data graphs.** With learned data graphs, we indicate those approaches that learn the graph topology from a stream of data $\mathbf{x}_1, \ldots, \mathbf{x}_t$. The illustration in Figure 1.1 from [15] is one such example. The key idea is to connect data elements that share

[2] For the sake of completeness, we hereto report the recent works [11, 12] that advocate the use of the graph Laplacian matrix theory for directed graphs as well.

[3] In literature $\mathbf{L}$ is often referred to as the Laplacian operator [6].

similar properties, such as correlations. This includes also approaches that build the graph adjacency matrix, or the graph Laplacian to explain a data distribution $\mathscr{P}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ [16]. Graph learning is recently enjoying an increasing popularity; we refer to [17–22] for some typical references.



### 2.2.1. COMPARISONS: PHYSICAL GRAPHS VERSUS DATA GRAPHS

The above observations yield a graph structure that models in mathematical terms the interdependencies hidden in the data. Having a good and representative structure will be the key to the upcoming spectral analysis of graph signals in Section 2.3. The comparisons between the physical graphs and data graphs are as follows:

- The physical graph model tends to explain the node dependencies by providing a structure to their natural applications and thus is easier to interpret. This is to some extent an intuitive and natural way to proceed and has yielded several accurate models of real phenomena, see [14, 23–25]. However, its natural simplification may be a limitation when the data at hand has hidden dependencies.
- The data graph model employs mathematical tools and data priors to give a meaning to the underlying hidden structure of the data. It is a more versatile and adaptive model and thus it may provide more explanatory power. On the downside, in comparison to the physical graph model, the data graph model may not have explicit meaningful interpretations and the lack of "sufficient" training data may influence the results.

### 2.2.2. THE GRAPH SIGNAL

A signal on top of the graph or a graph signal is defined as a mapping from the vertex set to the set of complex numbers, i.e., $x_i : v_i \to \mathbb{R}$. The examples in Figure 2.1 show two of such graph signals. For convenience, we collect all nodes' signals in the vector $\mathbf{x} \in \mathbb{R}^N$, where the $i$th component of $\mathbf{x}$ represents the signal value at node $v_i$.

### 2.2.3. THE GRAPH SHIFT OPERATOR

Throughout this thesis, we will commonly refer to the three graph representation matrices, i.e., the adjacency matrix $\mathbf{W}$, the graph Laplacian $\mathbf{L}$, and the normalized graph Laplacian $\mathbf{L}_{\text{n}}$, as the graph shift operator matrix $\mathbf{S}$. By construction, $\mathbf{S}$ is symmetric and real-valued. Additionally, for $\mathbf{S}$ we assume the following:

**Assumption 2.2.** (Graphs of bounded norm.) *We consider graphs $\mathscr{G}$ which have a graph shift operator matrix $\mathbf{S}$ with bounded spectral norm $\|\mathbf{S}\| \leq \varrho$, for some constant $\varrho \geq 0$.*

The above assumption focuses our attention to graphs of finite dimensions and with finite weights; both valid considerations in practice and broadly used in literature.

One of the central operations in GSP is the shifting of a graph signal $\mathbf{x}$ over the graph, i.e.,

$$\mathbf{x}^{(1)} = \mathbf{S}\mathbf{x}, \tag{2.5}$$

where $\mathbf{x}^{(1)}$ stands for one-shift of $\mathbf{x}$ by $\mathbf{S}$. Similarly, $\mathbf{x}^{(0)} = \mathbf{I}_N\mathbf{x}$ can be considered as the zero-shifting of $\mathbf{x}$ over the graph, i.e., the graph signal itself. An important aspect of the

shifting operation (2.5) is that it can also be computed locally paving the way to compute $\mathbf{x}^{(1)}$ distributively. That is, each node needs information only from its direct neighbors, without requiring global information about the whole vector $\mathbf{x}$. With reference to Figure. 2.1 (b), the one-shift of $\mathbf{x}$ over that graph is

$$\begin{pmatrix} x_1^{(1)} \\ x_2^{(1)} \\ x_3^{(1)} \\ x_4^{(1)} \\ x_5^{(1)} \end{pmatrix} = \begin{pmatrix} S_{1,1} & S_{1,2} & 0 & 0 & S_{1,5} \\ S_{2,1} & S_{2,2} & S_{2,3} & S_{2,4} & 0 \\ 0 & S_{3,2} & S_{3,3} & S_{3,4} & 0 \\ 0 & S_{4,2} & S_{4,3} & S_{4,4} & S_{5,4} \\ S_{5,1} & 0 & 0 & S_{5,4} & S_{5,5} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix},$$

where it can be noticed that any node $v_i$ can compute its shifted signal $x_i^{(1)} = \sum_{j \in (\mathcal{N}_i \cup i)} S_{i,j} x_j$ by simply obtaining information from its direct neighbors $\mathcal{N}_i$ and potentially the node $v_i$ itself (e.g., if $\mathbf{S}$ is the graph Laplacian or the normalized graph Laplacian). For instance, $x_5^{(1)} = S_{5,1}x_1 + S_{5,4}x_4 + S_{5,5}x_5$ requires information from from nodes $v_1$, $v_4$, and $v_5$.

For the different choices of $\mathbf{S}$, the shifting operation (2.5) takes the form:

- for $\mathbf{S} = \mathbf{W}$, we have $x_i^{(1)} = \sum_{j \in \mathcal{N}_i} W_{i,j} x_j$;
- for $\mathbf{S} = \mathbf{L}$, we have $x_i^{(1)} = \sum_{j \in \mathcal{N}_i} W_{i,j}(x_i - x_j)$;
- for $\mathbf{S} = \mathbf{L}_{\text{n}}$, we have $x_i^{(1)} = \frac{1}{\sqrt{D_{i,i}}} \sum_{j \in \mathcal{N}_i} W_{i,j} \left( \frac{x_i}{\sqrt{D_{i,i}}} - \frac{x_j}{\sqrt{D_{j,j}}} \right)$.

Similarly, higher order shifts can be computed recursively as $\mathbf{x}^{(k)} = \mathbf{S}^k \mathbf{x} = \mathbf{S}\mathbf{S}^{k-1}\mathbf{x} = \mathbf{S}\mathbf{x}^{(k-1)}$, i.e., nodes can perform locally the $k$-shift $\mathbf{x}^{(k)}$ by exchanging with their neighbors information about the previous shifted version of $\mathbf{x}$, $\mathbf{x}^{(k-1)}$. This aspect will play a central role in the distributed implementation of graph filters in Part II.

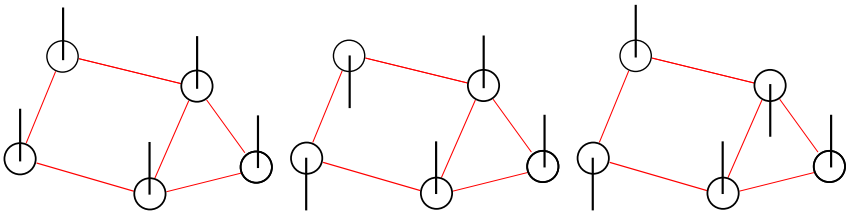
## 2.3. SPECTRAL ANALYSIS OF GRAPH SIGNALS

With the graph playing the role of the signal support, we can now formalize the notion of signal variation over this support, and therefore to provide a spectral analysis for graph signals. To start, let us consider the following example:

**Example 2.1.** (Signal variation on the graph edges.) *Consider the scenario in Figure 2.2 depicting three different graph signals residing on the same topology $\mathcal{G}$. The vertical bars indicate the signal value where positive values are illustrated with a bar oriented upwards and a negative value with a bar oriented downwards.*

*We are interested in finding which of the three graph signals varies the most and which of the three graph signals varies the least on the graph. Visually, we can see that the signals in (b) and (c) vary more on $\mathcal{G}$ than the constant signal in (a).*

*One way to quantify the signal variation over $\mathcal{G}$ is to count the number of times that the graph signal changes sign between nodes that share an edge. Then, for (a) we have no sign change, for (b) two sign changes, and for (c) five sign changes. Thus, we may "more formally" conclude that the signal in (a) is the least varying over $\mathcal{G}$ and the signal in (c) is the most varying over $\mathcal{G}$.*



(a) Constant signal. (b) Slow-varying signal. (c) High-varying signal.

Figure 2.2: Illustration of three different graph signals over the same undirected graph. From left to right the signal variation over the graph increases.

This intuitive characterization of signal variation over a graph will serve as a trigger to mathematically formulate the harmonic analysis for graph signals leading to the concept of the GFT. Finally, observe that the topology plays a central role when counting the sign change. In fact, if an edge between two different nodes of opposite sign is removed this sign change is not counted. In Section 2.3.2 we will come back to this important factor in the definition of the GFT.

### 2.3.1. SIGNAL VARIATION OVER THE GRAPH

For a signal $\mathbf{x}$ over a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the variation of $\mathbf{x}$ with respect to the edge $e_{i,j} = (v_i, v_j)$ valued at vertex $v_i$ is given by the edge derivative

$$\left.\frac{\partial \mathbf{x}}{\partial e_{i,j}}\right|_{v_i} = \sqrt{W_{i,j}}(x_i - x_j), \tag{2.6}$$

that is, the difference of the signal at the end nodes of $e_{i,j} = (v_i, v_j)$ weighted by the square root of the edge weight. Subsequently, the graph gradient of $\mathbf{x}$ at vertex $v_i$ is the vector

$$\nabla_{v_i}\mathbf{x} = \left\{ \left.\frac{\partial \mathbf{x}}{\partial e_{i,j}}\right|_{v_i} \right\}_{e_{i,j}\in\mathcal{E} \text{ s.t. } e_{i,j}=(v_i,v_j) \text{ for some } v_j \in \mathcal{V}} \tag{2.7}$$

containing all partial derivatives of $\mathbf{x}$ at node $v_i$. Then, the $l_2$-norm of (2.7)

$$\|\nabla_{v_i}\mathbf{x}\|_2 = \left( \sum_{v_j \in \mathcal{N}_i} W_{i,j}(x_i - x_j)^2 \right)^{\frac{1}{2}} \tag{2.8}$$

provides a measure of signal variability of $\mathbf{x}$ around vertex $v_i$ with respect to its neighbors $\mathcal{N}_i$. Precisely, the graph signal is said to be smooth in the $v_i$'s neighborhood $\mathcal{N}_i$ if $\|\nabla_{v_i}\mathbf{x}\|_2$ is small, or equivalently if $x_i$ is similar to $x_j$ for $v_j \in \mathcal{N}_j$. Consequently, $\|\nabla_{v_j}\mathbf{x}\|_2 > \|\nabla_{v_i}\mathbf{x}\|_2$ indicates that the signal variation at node $v_j$ is higher than the signal variation at node $v_i$. The constant signal in Figure 2.4 (a) is an exceptional example since it has $\|\nabla_{v_i}\mathbf{x}\|_2 = 0$ for all $v_i \in \mathcal{V}$.

The notion of signal variation can be extrapolated from a particular node to the whole graph by means of the $p-$Dirichlet form of $\mathbf{x}$

$$S_p(\mathbf{x}) = \frac{1}{p}\sum_{v_i\in\mathcal{V}} \|\nabla_{v_i}\mathbf{x}\|_2^p = \frac{1}{p}\sum_{v_i\in\mathcal{V}}\left(\sum_{v_j\in\mathcal{N}_i} W_{i,j}(x_i - x_j)^2\right)^{\frac{p}{2}}, \tag{2.9}$$

which consists of a weighted sum of the signal variations in all nodes. Particular forms of $S_p(\mathbf{x})$ are $i$) $S_1(\mathbf{x})$ referred to as the signal total variation, and $ii$)

$$S_2(\mathbf{x}) = \frac{1}{2}\sum_{v_i\in\mathcal{V}}\sum_{v_j\in\mathcal{N}_i} W_{i,j}(x_i - x_j)^2 = \sum_{(v_i,v_j)\in\mathcal{E}} W_{i,j}(x_i - x_j)^2 = \mathbf{x}^\mathsf{T}\mathbf{L}\mathbf{x} \tag{2.10}$$

also known as the graph Laplacian quadratic form [6]. In relation with Example 2.1, $S_2(\mathbf{x})$ is zero for the signal in Figure 2.4 (a) (the signal has no variation) and $S_2(\mathbf{x})$ of the signal in Figure 2.4 (b) is smaller than $S_2(\mathbf{x})$ of the signal in Figure 2.4 (c).

### 2.3.2. THE GRAPH FOURIER TRANSFORM

The GFT relies on the spectral decomposition of the graph Laplacian. Specifically, since $\mathbf{L}$ is real and symmetric [cf. Assumption 2.1], it enjoys the eigendecomposition

$$\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\mathsf{H}, \tag{2.11}$$

where $\mathbf{U} = (\mathbf{u}_0, \ldots, \mathbf{u}_{N-1})$ is an $N \times N$ orthonormal matrix containing eigenvectors of $\mathbf{L}$ and $\mathbf{\Lambda} = \text{diag}(\lambda_0, \ldots, \lambda_{N-1})$ is an $N \times N$ diagonal matrix with the $i$th diagonal element the $i$th eigenvalue of $\mathbf{L}$. We consider the eigenvalues of $\mathbf{L}$ to be ordered as $0 = \lambda_0 < \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_{N-1} := \lambda_{\max}$, where zero appears as an eigenvalue with the same multiplicity as the number of connected components of the graph [6] [one, cf. Assumption 2.1]. Therefore, $\lambda_i > 0$ for all $i = 1, \ldots, N-1$.

From [26] and as shown in [1], the eigenvalues and eigenvectors of $\mathbf{L}$ can also be defined iteratively by solving the Rayleigh quotient

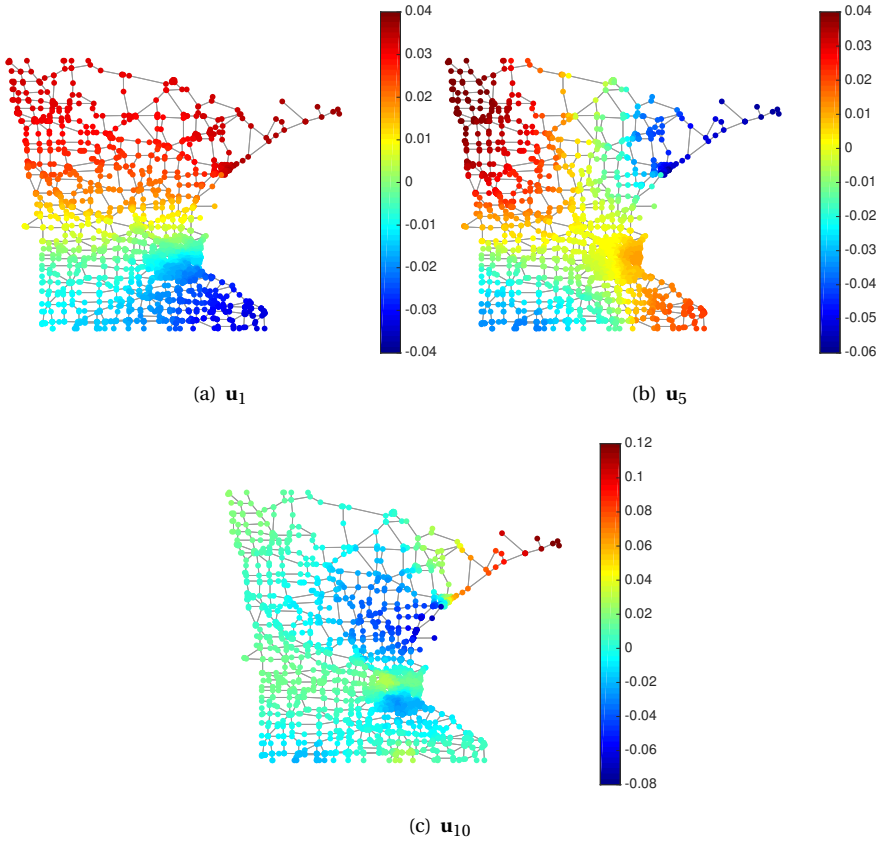
$$\begin{aligned} \lambda_0 = \min_{\mathbf{x}\in\mathbb{R}^N} \quad & \mathbf{x}^\mathsf{T}\mathbf{L}\mathbf{x} \\ \text{s.t.} \quad & \|\mathbf{x}\|_2 = 1 \end{aligned} \tag{2.12}$$

and

$$\begin{aligned} \lambda_l = \min_{\mathbf{x}\in\mathbb{R}^N} \quad & \mathbf{x}^\mathsf{T}\mathbf{L}\mathbf{x} \\ \text{s.t.} \quad & \|\mathbf{x}\|_2 = 1, \\ & \mathbf{x} \perp \text{span}\{\mathbf{u}_0, \ldots, \mathbf{u}_{l-1}\}, \quad l = 1, \ldots, N-1, \end{aligned} \tag{2.13}$$

where the tuple $(\lambda_l, \mathbf{u}_l)$ consists of the minimum and the minimizer of the $l$th problem, respectively.

From (2.10), (2.12) and (2.13), we note that the graph Laplacian eigenvectors are the minimizer of the Laplacian quadratic form. Hence, the eigenvectors associated with a smaller eigenvalue yield a smaller cost and are smoother (i.e., vary less) over $\mathcal{G}$ than the eigenvectors associated with a higher eigenvalue. This suggests that the graph Laplacian eigenvalues and eigenvectors carry some notion of frequency in the graph setting. To illustrate this concept visually, we consider a modified example from [1].




(a) $\mathbf{u}_1$ (b) $\mathbf{u}_5$

(c) $\mathbf{u}_{10}$

Figure 2.3: Variation of the graph Laplacian eigenvectors for the Minnesota roadmap graph. The graph Laplacian quadratic form is (a) $S_2(\mathbf{u}_2) = 8.4e^{-4}$, (b) $S_2(\mathbf{u}_5) = 0.0031$ and (c) $S_2(\mathbf{u}_{10}) = 0.01$.

**Example 2.2.** (Variation of the Minnesota roadmap eigenvectors.) *Suppose $\mathcal{G}$ is the Minnesota roadmap with graph Laplacian $\mathbf{L}$. We evaluate the variability on $\mathcal{G}$ of three different graph signals, namely the third, the sixth and the eleventh eigenvector of $\mathbf{L}$ i.e.,* $\mathbf{x} = \{\mathbf{u}_2, \mathbf{u}_5, \mathbf{u}_{10}\}$.

*Figure 8.6 shows in colormap these signals on top of the Minnesota graph. The respective Laplacian quadratic forms have values $S_2(\mathbf{u}_2) = 8.4e^{-4}$, $S_2(\mathbf{u}_5) = 0.0031$, and $S_2(\mathbf{u}_{10}) = 0.01$. These results enforce the derivations of* (2.10), (2.12) *and* (2.13) *and show that eigenvectors associated with higher eigenvalues are characterized by higher variations over $\mathcal{G}$ and viceversa.*

From the above discussion, we may conclude that the graph Laplacian eigenvectors form an orthonormal basis with each eigenvector carrying some notion of frequency in the graph setting. The following definition formalizes then the expansion of a graph signal on this frequency basis.

**Definition 2.1.** (Graph Fourier transform.) *The graph Fourier transform* $\hat{\mathbf{x}}$ *of a graph signal* $\mathbf{x}$ *living on the graph* $\mathcal{G}$ *with graph Laplacian matrix* $\mathbf{L}$ *is defined as*

$$\hat{\mathbf{x}} = \mathbf{U}^{\mathsf{H}}\mathbf{x}, \tag{2.14}$$

*where* $\mathbf{U}$ *is the eigenvector matrix of* $\mathbf{L}$.



That is, the GFT is the expansion of $\mathbf{x}$ in terms of the eigenvectors of the graph Laplacian. Vector $\hat{\mathbf{x}}$, containing the GFT coefficients, represents the weight that each eigenvector has in this expansion. The graph Laplacian eigenvalues will then serve as the support for each GFT coefficient and will be referred to as *the graph frequencies* [1].

**Interpretation from a system perspective.** The graph Laplacian eigenvectors act as the oscillating modes of the graph. In fact, if we abstract the notion of the graph and consider $\mathbf{L}$ to be the matrix transfer function of a linear system $\mathbf{y} = \mathbf{L}\mathbf{x}$, the eigenvectors of $\mathbf{L}$ are the system oscillating modes [27].Then, if $\mathbf{x}$ equals one of the eigenvectors of $\mathbf{L}$, say $\mathbf{u}_i$, it will make the system oscillate at this particular mode. In this respect, the GFT expresses the input signal $\mathbf{x}$ as a linear combination of the system modes. Following this analogy, for a linear system $\mathbf{L}$ we can imaginatively call this expansion as the "system Fourier transform".

**The role of the graph.** The underlying graph structure plays an important role in the GFT of a signal $\mathbf{x}$. To see this, consider two different graphs $\mathcal{G}_1 = (\mathcal{V}, \mathcal{E}_1)$ and $\mathcal{G} = (\mathcal{V}, \mathcal{E}_2)$ with respective Laplacians $\mathbf{L}_1 = \mathbf{U}_1\mathbf{\Lambda}_1\mathbf{U}_1^{\mathsf{H}}$ and $\mathbf{L}_2 = \mathbf{U}_2\mathbf{\Lambda}_2\mathbf{U}_2^{\mathsf{H}}$ and the same graph signal $\mathbf{x}$. The GFT of $\mathbf{x}$ w.r.t. $\mathcal{G}_1$ and $\mathcal{G}_2$ equals (using (2.14)) $\hat{\mathbf{x}}_1 = \mathbf{U}_1^{\mathsf{H}}\mathbf{x}$ and $\hat{\mathbf{x}}_2 = \mathbf{U}_2^{\mathsf{H}}\mathbf{x}$, respectively. Therefore, the GFT coefficients that yield from these expansions are different. From a linear algebra perspective, this means that $\mathbf{x}$ is expanded on two different bases, and, thus, the basis coefficient expansions are different. From a practical viewpoint, every edge change (e.g., addition, or removal) in $\mathcal{G}$ yields a different GFT interpretation of the same signal $\mathbf{x}$.

Likewise (2.14), the inverse GFT (IGFT) is defined as

**Definition 2.2.** (Inverse graph Fourier transform.) *The inverse graph Fourier transform* $\mathbf{x}$ *of* $\hat{\mathbf{x}}$ *is*

$$\mathbf{x} = \mathbf{U}\hat{\mathbf{x}}, \tag{2.15}$$

*where* $\mathbf{U}$ *is the eigenvector matrix of the graph Laplacian* $\mathbf{L}$.

That is, it expresses the signal $\mathbf{x}$ in the vertex domain from its graph spectral decomposition. Note that operations (2.14)-(2.15) preserve the Parseval property since $\mathbf{U}$ is an orthonormal matrix.

As a final observation, we highlight that the Laplacian eigenvectors are only one of the possible bases to perform the graph spectral decomposition. In fact, any graph shift operator matrix $\mathbf{S}$, whose eigenvectors carry a notion of frequency in the graph setting can be a potential choice. For more details about the graph harmonic expansion when $\mathbf{S} = \mathbf{W}$ or $\mathbf{S} = \mathbf{L}_{\mathrm{n}}$ we refer to [5] and [1], respectively. For the considered shift operator candidates there are a few properties to consider:

- For $\mathbf{S} = \mathbf{W}$, the slow varying eigenvectors are associated to eigenvalues of large magnitude, and viceversa [5].

- For $\mathbf{S} = \mathbf{L}$, the slow varying eigenvectors are associated to eigenvalues of low magnitude, and viceversa [1]. The eigenvector $\mathbf{u}_0$ associated to $\lambda_0 = 0$ is constant, $\mathbf{u}_0 = 1/\sqrt{N}\mathbf{1}_N$.
- For $\mathbf{S} = \mathbf{L}_{\text{n}}$, the slow varying eigenvectors are associated to eigenvalues of low magnitude, and viceversa [1]. In this instance, $\mathbf{u}_0$ is not anymore constant and the maximum eigenvalue satisfies $\lambda_{\max} \leq 2$. Equality holds for the class of bipartite graphs [6].

### 2.3.3. CONNECTION: CLASSICAL FOURIER TRANSFORM AND GRAPH FOURIER TRANSFORM

A number of pioneering studies on spectral analysis of graph signals [1, 2, 4, 7, 10] advocate the specialization of the GFT to the classical Fourier transform when the signal of interest is the time-varying temporal signal $x(t) \in \mathbb{R}$. Although with some differences between them, all these works show that their interpretation of GFT specializes to the classical Fourier transform for temporal signals. Let us here briefly show this for the graph Laplacian.

The classical Fourier transform

$$\hat{x}(f) = \int_{\mathbb{R}} x(t) e^{-2\pi \jmath f t} dt \tag{2.16}$$

expands the temporal signal $x(t) \in \mathbb{R}$ onto the basis of complex exponentials. These complex exponentials correspond to the eigenfunctions of the one-dimensional Laplace operator [1, 2, 7]

$$-\Delta(e^{2\pi \jmath f t}) = -\frac{\partial^2}{\partial t^2} e^{2\pi \jmath f t} = (2\pi f)^2 e^{2\pi \jmath f t}, \tag{2.17}$$

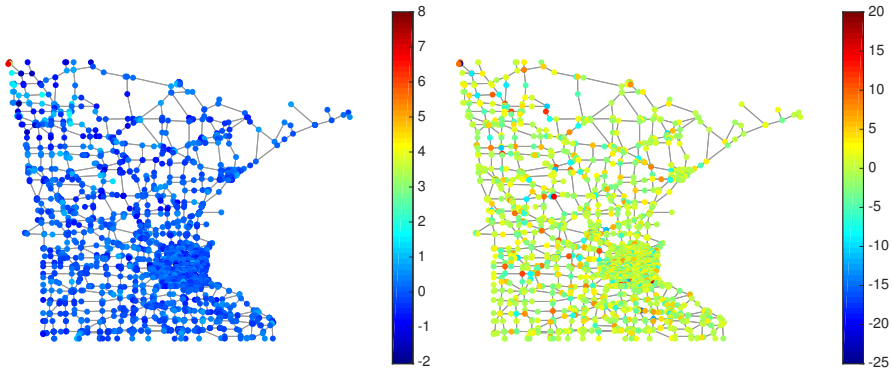
where the eigenvalues $(2\pi f)^2$ carry the notion of frequency. In fact, the eigenvectors (complex exponentials) associated to $f$ close to zero oscillate slowly in time, whereas the eigenvectors (complex exponentials) associated to $f \gg 0$ oscillate more rapidly.

Similarly, the GFT (2.14) expands the graph signal $\mathbf{x}$ onto the basis spanned by the eigenvectors of the $N$-dimensional Laplace operator $\mathbf{L}$. In this case, the notion of frequency is carried by the eigenvalues $\mathbf{\Lambda}$ of $\mathbf{L}$, thus the name *graph frequency*. In fact, as shown in Sections 2.3.1-2.3.2 the graph Laplacian eigenvectors $\mathbf{u}_j$ associated with low graph frequencies $\lambda_j$ vary slowly on $\mathcal{G}$, i.e., the eigenvector's value on two connected vertices is likely to be similar. Analogously, the graph Laplacian eigenvectors $\mathbf{u}_j$ associated with a larger graph frequency $\lambda_j$ vary faster on $\mathcal{G}$ and two connected vertices are more likely to have dissimilar values.

### 2.3.4. GRAPH SIGNAL BANDWIDTH

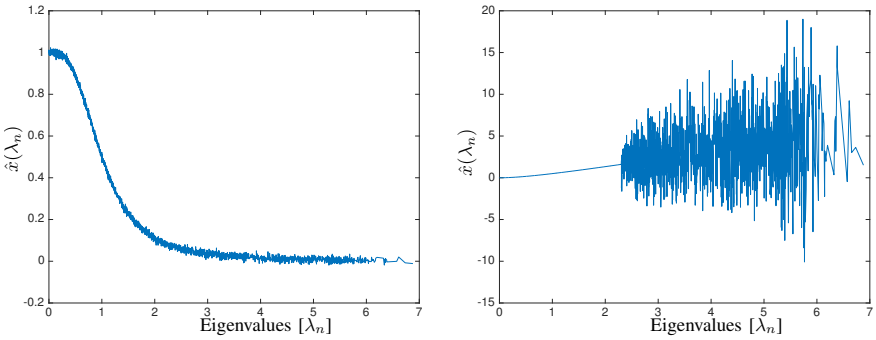
With the formalization of the signal variation over graphs (Section 2.3.1) and with the definition of the GFT (Section 2.3.2), we are now able to provide a more formal answer to questions such as: *What is a graph signal with low/high graph spectral content?* or *What is the bandwidth of a graph signal?*

A graph signal $\mathbf{x}$ is said to have low graph frequency content if its energy is mostly concentrated in the graph frequencies related to the slow-varying eigenvectors. One



(a) Low frequency content graph signal in the vertex domain.

(b) High frequency content graph signal in the vertex domain.




(c) Low frequency content graph signal in the spectral domain.

(d) High frequency content graph signal in the spectral domain.

Figure 2.4: Two different graph signals in both the vertex and graph spectral domains.

such graph signal along with its graph spectral content is shown in Figure 2.4 (a) and (c), respectively.

Similarly, a graph signal $\mathbf{x}$ is said to have high graph frequency content if it has substantial energy in the graph frequencies related to the eigenvectors that vary fast on $\mathcal{G}$. These graph signals are characterized by having dissimilar values in adjacent vertices. Figure 2.4 (b) illustrates one example with respective GFT shown in Figure 2.4 (d).

A graph signal $\mathbf{x}$ is said to be bandlimited on $\mathcal{G}$ with bandwidth $|\mathcal{F}| \leq N$ if and only if its graph spectral content $\hat{\mathbf{x}}$ is different from zero only on a limited set $\mathcal{F}$ of graph frequencies (not necessarily adjacent). More formally, the set $\mathcal{F}$ consists of

$$\mathcal{F} = \{\lambda_i | \hat{x}_i \neq 0, \text{ for } i \in \{0, \ldots, N-1\}\}. \tag{2.18}$$

The graph signal in Figure 2.4 (a) can be said to be bandlimited as its graph spectral content (Figure 2.4 (c)) is close to zero for $\lambda_i > 3$.

Observe that similarly to the role played by the graph structure in the GFT definition [cf. Section 2.3.2], the same graph signal $\mathbf{x}$ may be bandlimited on a particular graph $\mathcal{G}_1$

and not-bandlimited on another graph $\mathcal{G}_2$. The same consideration holds also for the low/high graph frequency content of $\mathbf{x}$.

**Interpretation as sparse representation.** For $\mathbf{x}$ being $\mathcal{F}$-bandlimited on $\mathcal{G}$, its GFT $\hat{\mathbf{x}}$ contains respectively $N-|\mathcal{F}|$ elements that are zero. Let us denote as $\tilde{\mathbf{x}}_{\mathcal{F}} = [\tilde{x}_1, \ldots, \tilde{x}_{|\mathcal{F}|}]^\mathsf{T}$ the $|\mathcal{F}| \times 1$ vector containing only the non-zero elements of $\hat{\mathbf{x}}$. Then, by means of the IGFT, we write

$$\mathbf{x} = \mathbf{U}_{\mathcal{F}} \tilde{\mathbf{x}}_{\mathcal{F}}, \tag{2.19}$$

where $\mathbf{U}_{\mathcal{F}}$ is the $N \times |\mathcal{F}|$ column-trimmed eigenvector matrix relative to $\hat{x}_i \neq 0$. That is, it is composed of the columns of $\mathbf{U}$ associated with the non-zero GFT coefficients of $\mathbf{x}$. Expression (2.19) has an analogy with the sparse representation literature [28]. In fact, $\tilde{\mathbf{x}}_{\mathcal{F}}$ can be interpreted as an $\mathcal{F}$-sparse representation of $\mathbf{x}$ in the space spanned by some of the graph Laplacian eigenvectors. The atoms of this sparse decomposition are now given by the columns of $\mathbf{U}_{\mathcal{F}}$. This property is widely used in GSP, especially for graph signal sampling purposes [29–31]. In Part IV, we will as well leverage this property for the adaptive algorithms and sampling.

### 2.3.5. GRAPH FILTERING

The introduction of harmonic analysis to graph signals creates the opportunity to process the latter in the graph spectral domain as well. In this regard, we define, next, the graph filter as the basic block to alter the spectrum of graph signals.

**Definition 2.3.** (Graph filters.) *A graph filter $h(\lambda_n)$ is defined as a function over the graph frequencies $\{\lambda_n\}_{n=0}^{N-1}$ to the set of real numbers, i.e., $h : \{\lambda_n\}_{n=0}^{N-1} \to \mathbb{R}$, altering the graph frequency content $\hat{\mathbf{x}}$ of $\mathbf{x}$ as a point-wise multiplication in the graph Fourier domain. That is, the graph filter output at the graph frequency $\lambda_n$ is $\hat{y}(\lambda_n) = h(\lambda_n)\hat{x}(\lambda_n)$.*

This definition of graph filters preserves the convolution property from discrete signal processing, i.e., a filtering operation corresponds to a point-wise multiplication in the Fourier domain. By stacking in the vector $\hat{\mathbf{y}} = [\hat{y}(\lambda_0), \ldots, \hat{y}(\lambda_{N-1})]^\mathsf{T}$ the filter output for all graph frequencies, we obtain

$$\hat{\mathbf{y}} = h(\mathbf{\Lambda})\hat{\mathbf{x}}, \tag{2.20}$$

where $h(\mathbf{\Lambda}) = \text{diag}(h(\lambda_0), \ldots, h(\lambda_{N-1}))$ is referred to as *the graph filter frequency response*. Then, by applying (2.14) to (2.20) we have

$$\mathbf{U}^\mathsf{H}\mathbf{y} = h(\mathbf{\Lambda})\mathbf{U}^\mathsf{H}\mathbf{x}, \tag{2.21}$$

which by means of the IGFT (2.15) becomes

$$\mathbf{y} = \mathbf{U}h(\mathbf{\Lambda})\mathbf{U}^\mathsf{H}\mathbf{x} = \mathbf{H}\mathbf{x}. \tag{2.22}$$

Equation (2.22) reformulates the graph filtering operation (2.20) in the vertex domain. We see that the filter output is now expressed as a linear combination of the input signal, with a graph filter impulse response[4] $\mathbf{H} = \mathbf{U}h(\mathbf{\Lambda})\mathbf{U}^\mathsf{H}$. Differently, we can often write (2.22)

[4] The term *filter impulse response* is in fact lend from the analogy with the temporal filtering operation, i.e., the inverse Fourier transform of the filter frequency response.

as $\mathbf{y} = h(\mathbf{S})\mathbf{x}$, which expresses the graph filtering operation as a function of the graph shift operator. Operation (2.22) has a computational cost of the order of a matrix-vector multiplication, i.e., $O(N^2)$. In Part II, we will show how to compute the filtering output in (2.22) with a cost that is lower than $O(N^2)$.

Let us finally illustrate the graph filtering application with a continuation of the denoising Example 1.1.



**Example 2.3.** (Graph signal denoising in the GFT domain.) *Suppose the graph $\mathcal{G}$ consists of the community graph with a piece-wise constant graph signal $\mathbf{x}_d$. We are interested to recover $\mathbf{x}_d$ by filtering noisy measurements $\mathbf{x} = \mathbf{x}_d + \mathbf{w}$ in the GFT domain.*

*This procedure is illustrated in Figure 2.5. We observe that the desired signal $\mathbf{x}_d$ is characterized by a low-pass spectrum, while the noisy signal $\mathbf{x}$ contains a substantial high graph frequency content. From the rightmost figures, we observe that the filtered signal is less noisy. The latter can be either observed in the vertex domain where the signal is more smooth within a community, or in the GFT domain where $\mathbf{y}$ does not contain high graph frequency content.*

### 2.3.6. TIKHONOV REGULARIZATION ON GRAPHS

Tikhonov regularization [32–35] is probably one of the most studied and used regularization techniques used for signal denoising. In the sequel, we briefly review this problem and show how it can be cast as a graph filtering operation.

From a GSP perspective, Tikhonov regularization considers recovering the desired signal $\mathbf{x}_\mathrm{d}$ from a single snapshot of a noisy observation $\mathbf{x} = \mathbf{x}_\mathrm{d} + \mathbf{w}$ given the prior assumption that $\mathbf{x}_\mathrm{d}$ is smooth w.r.t. the underlying graph $\mathcal{G}$. From Section 2.3.1, we observe that requiring $\mathbf{x}_\mathrm{d}$ to be smooth over $\mathcal{G}$ is equivalent to $\mathbf{x}_\mathrm{d}^\mathsf{T}\mathbf{S}\mathbf{x}_\mathrm{d}$ being small for $\mathbf{S} = \mathbf{L}$ or $\mathbf{S} = \mathbf{L}_\mathrm{n}$. Then, the estimate $\mathbf{x}_\mathrm{d}^\star$ of $\mathbf{x}_\mathrm{d}$ is cast as the solution of the optimization problem

$$\mathbf{x}_\mathrm{d}^\star = \underset{\mathbf{x}\in\mathbb{R}^N}{\operatorname{argmin}} \quad \|\mathbf{x} - \mathbf{x}_\mathrm{d}\|_2^2 + w\mathbf{x}_\mathrm{d}^\mathsf{T}\mathbf{S}\mathbf{x}_\mathrm{d} \tag{2.23}$$

for some fixed $w \geq 0$. The left term in the optimization function asks for a signal that is close to $\mathbf{x}_\mathrm{d}$, while the second term uses the quadratic form of $\mathbf{S}$ to enforce the smoothness prior. There are two extreme cases

- when $w = 0$, the objective does not consider any smoothness prior and the solution is $\mathbf{x}_\mathrm{d}^\star = \mathbf{x}$;
- when $w \to +\infty$, the objective emphasises on the smoothness prior and the optimal solution goes to $\mathbf{x}_\mathrm{d}^\star = \mathbf{0}_N$.

Any value of $w$ in between allows for a trade-off between these two cases whose global solution is

$$\mathbf{x}_\mathrm{d}^\star = (\mathbf{I}_N + w\mathbf{S})^{-1}\mathbf{x}. \tag{2.24}$$

By leveraging (2.22), the optimal solution $\mathbf{x}_\mathrm{d}^\star$ can be seen as a graph filtering of $\mathbf{x}$ with the graph filter $\mathbf{H} = (\mathbf{I}_N + w\mathbf{S})^{-1}$. By transforming $\mathbf{H}$ in the GFT domain, the filter frequency

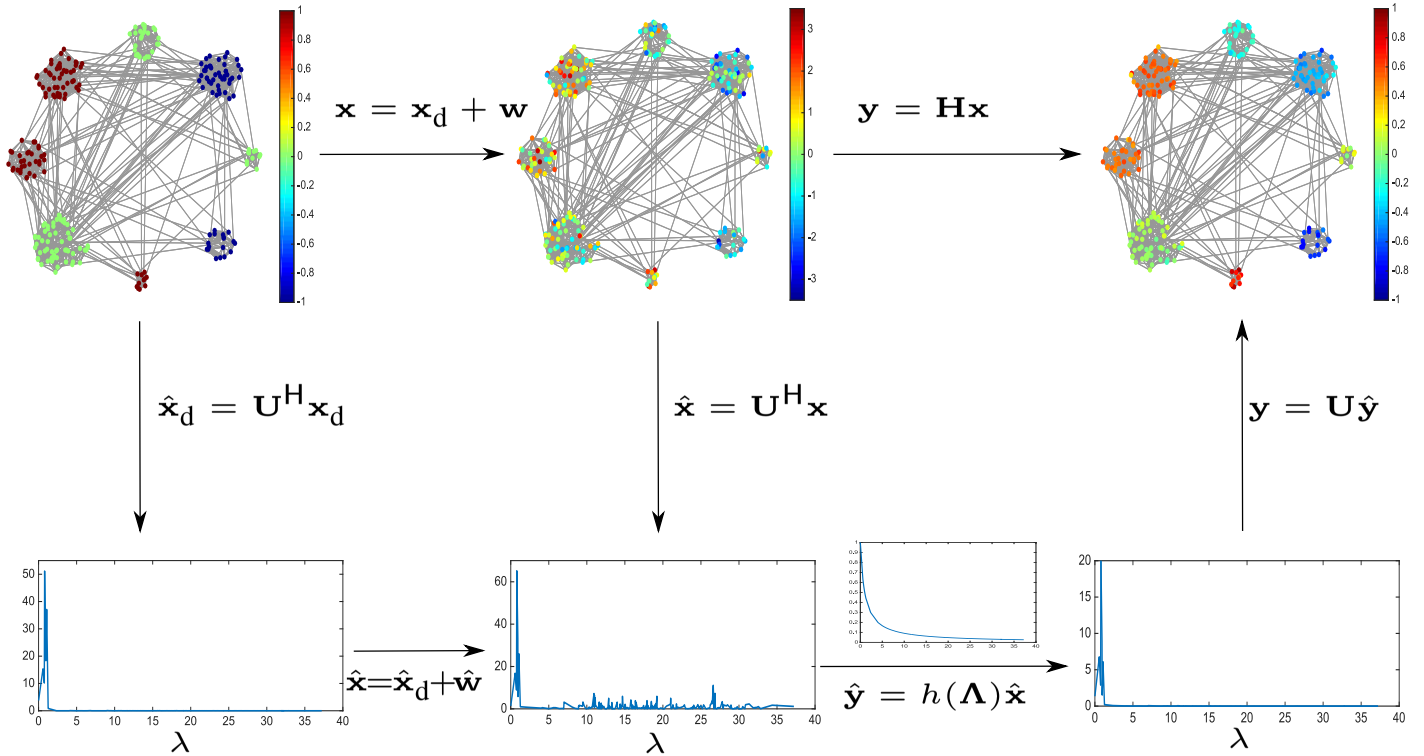



Figure 2.5: Illustration of graph signal denoising example with graph filters. (Top) Signals in the vertex domain. (Bottom) Signals in the GFT domain. Observe how the noisy signal presents high frequency content (bottom center), which is removed with the graph filter (bottom right).

response is[5] $h(\mathbf{\Lambda}) = (\mathbf{I}_N + w\mathbf{\Lambda})^{-1}$. In Chapter 4, we will see that graph filters that present a similar graph frequency response correspond to the class of IIR graph filters.

## 2.4. STATIONARY GRAPH SIGNALS



Stationarity is another useful property that has recently been extended to graph signals [36–38]. In what follows, we briefly provide the definition of graph wide sense stationarity (GWSS) with the aim to introduce the Wiener regularization on graphs. The latter turns out to be a graph filtering operation as well.

### 2.4.1. WIDE SENSE STATIONARITY ON GRAPHS

For a graph signal $\mathbf{x}$ abiding to some distribution $\mathscr{P}(\mathbf{0}_N, \mathbf{\Sigma}_x)$, GWSS is defined as follows[6]:

**Definition 2.4.** (GWSS.) *A random graph signal* $\mathbf{x}$ *is said to be graph wide sense stationary, if and only if its covariance matrix* $\mathbf{\Sigma}_x$ *is diagonalizable by the GFT basis* $\mathbf{U}$*, i.e.,* $\mathbf{\Sigma}_x = \mathbf{U}\text{diag}(\mathbf{p}_x)\mathbf{U}^{\mathsf{H}}$*. The* $N\times 1$ *vector* $\mathbf{p}_x$ *is referred to as the graph power spectral density (GPSD).*

The above definition extends the classical notion of stationarity to graph signals, where now it is assumed that the signal statistics are preserved along the graph dimension[7]. In fact, it requires the translation invariance of the second order moment w.r.t. the graph shift operator.

Similarly to temporal wide sense stationarity, a useful property of GWSS is that $\mathbf{\Sigma}_x$ can be driven by a graph filter with frequency response $h(\mathbf{\Lambda}) = \text{diag}(\mathbf{p}_x)^{\frac{1}{2}}$. The latter suggests that a GWSS graph signal can be generated by filtering a random signal $\mathbf{w} \sim \mathscr{P}(\mathbf{0}_N, \mathbf{I}_N)$ with a graph filter of the form $\mathbf{H} = \mathbf{U}\text{diag}(\mathbf{p}_x)^{\frac{1}{2}}\mathbf{U}^{\mathsf{H}}$.

### 2.4.2. WIENER REGULARIZATION ON GRAPHS

Wiener regularization considers recovering a signal of interest $\mathbf{x}_{\text{d}}$ from a noisy realization $\mathbf{x} = \mathbf{x}_{\text{d}} + \mathbf{w}$ when $\mathbf{x}_{\text{d}} \sim \mathscr{P}_x(\mathbf{0}_N, \mathbf{\Sigma}_{x_{\text{d}}})$ and $\mathbf{w} \sim \mathscr{P}_w(\mathbf{0}_N, \mathbf{\Sigma}_{\text{w}})$. From the GSP perspective, we may say that the desired signal $\mathbf{x}_{\text{d}}$ is stochastic over the graph, in contrast to the Tikhonov-based denoising [cf. Section 2.3.6]. The linear Wiener solution to this denoising problem is obtained by minimizing the mean squared error (MSE)

$$\begin{aligned} \mathbf{H}^{\star} = \underset{\mathbf{H}\in\mathbb{R}^{N\times N}}{\text{argmin}} \quad & \mathbb{E}\left[\|\mathbf{H}\mathbf{x} - \mathbf{x}_{\text{d}}\|_2^2\right] \\ \text{s.t.} \quad & \mathbf{x} = \mathbf{x}_{\text{d}} + \mathbf{w}, \end{aligned} \tag{2.25}$$

and then set the recovered signal as $\mathbf{x}_{\text{d}}^{\star} = \mathbf{H}^{\star}\mathbf{x}$. Problem (2.12) is convex and admits a closed form solution which leads to the optimal recovered signal

$$\mathbf{x}_{\text{d}}^{\star} = \mathbf{\Sigma}_{x_{\text{d}}}\left(\mathbf{\Sigma}_{x_{\text{d}}} + \mathbf{\Sigma}_w\right)^{-1}\mathbf{x} \tag{2.26}$$

[5] From Sylvester's matrix theorem matrices $\mathbf{A}$ and $\mathbf{A}^{-1}$ have the same eigenvectors and the eigenvalues of $\mathbf{A}^{-1}$ are the inverse of the eigenvalues of $\mathbf{A}$.

[6] Although [36–38] introduce stationarity to graph signals from different starting points, they all meet with the result of Definition 2.4.

[7] For $\mathbb{E}(\mathbf{x}) \neq \mathbf{0}_N$ the GWSS has the additional requirements that $\mathbf{x}$ should have a constant mean.

given $\mathbf{\Sigma}_{x_\mathrm{d}} + \mathbf{\Sigma}_w$ is non-singular.

In those cases where $\mathbf{\Sigma}_{x_\mathrm{d}}$ and $\mathbf{\Sigma}_w$ share the eigenvectors with the graph shift operator (i.e., $\mathbf{x}_\mathrm{d}$ and $\mathbf{w}$ are GWSS), the above linear system takes the form of a graph filter. Specifically, for $p_{x_\mathrm{d}}(\lambda_n)$ being the $n$th eigenvalue of $\mathbf{\Sigma}_{x_\mathrm{d}}$ and $p_w(\lambda_n)$ being the $n$th eigenvalue of $\mathbf{\Sigma}_w$, from (2.26) the graph filter frequency response at the $n$th graph frequency is

$$h(\lambda_n) = \frac{p_{x_\mathrm{d}}(\lambda_n)}{p_{x_\mathrm{d}}(\lambda_n) + p_w(\lambda_n)}, \tag{2.27}$$

where once again we exploited the relation between the eigenvectors and eigenvalues of a matrix $\mathbf{A}$ and its inverse.

**Historical note.** Surprisingly, the idea of Wiener regularization on graphs was initially mentioned before the formalization of GWSS. In fact, [39] interpreted semi-supervised learning on graphs as a Wiener graph filter. Subsequently, in our work [40] we interpret the solution of problem (2.26) as an ARMA graph filter with a Wiener frequency response (2.27). Finally, [37] introduces the Wiener regularization as a potential tool to advocate the benefits of stationary graph signal processing.

### 2.4.3. CONNECTION: KARHUNEN-LOÉVE TRANSFORM AND STATIONARY GRAPH SIGNALS

The Karhunen-Loéve transform (KLT) [41] is a classical tool used in image processing mainly for compression purposes. Since both the KLT and the GFT for stationary graph signals exploit the eigendecomposition of the data covariance matrix, there is a connection between the two that we highlight in the following.

Consider a random vector $\mathbf{x}$ following some distribution $\mathscr{P}_x(\mathbf{0}_N, \mathbf{\Sigma}_x)$ with $\mathbf{\Sigma}_x = \mathbf{U}\mathrm{diag}(\mathbf{p}_x)\mathbf{U}^\mathsf{H}$. Up to an ordering of the elements, the KLT of $\mathbf{x}$ is

$$\hat{\mathbf{x}}_\mathrm{KL} = \mathbf{U}^\mathsf{H}\mathbf{x}, \tag{2.28}$$

i.e., it consists of a projection of $\mathbf{x}$ onto the eigenspace spanned by the eigenvectors of the covariance matrix. The convention wants $[\hat{\mathbf{x}}_\mathrm{KL}]_1$ to be the projection of $\mathbf{x}$ onto the eigenvector associated with the largest eigenvalue and $[\hat{\mathbf{x}}_\mathrm{KL}]_N$ to be the relative projection onto the eigenvector associated with the smallest eigenvalue.

In the instance that $\mathbf{x}$ resides on a graph $\mathscr{G}$ with $\mathbf{S} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\mathsf{H}$, from Definition 2.4 we observe that $\mathbf{x}$ belongs to the family of stationary graph signals. Moreover, its GFT $\hat{\mathbf{x}} = \mathbf{U}^\mathsf{H}\mathbf{x}$ is identical (up to a permutation) to the KLT transform of (2.28). Therefore, we conclude that the GFT of stationary graph signals is an alternative formulation of the KLT transform of these data. The advantage though is that no data and only the graph is required to compute the GFT.

## 2.5. CONCLUDING REMARKS

In this chapter, we introduced the basic principles of GSP, which builds the foundation for the remaining part of the thesis. After setting down the basic nomenclature about the graph structure and the graph signal, we showed three graph representation matrices,

namely, the graph adjacency, the graph Laplacian, and the normalized graph Laplacian matrix. These matrices serve as candidates for the so-called graph shift operator matrix; a matrix that captures the network connectivity in linear algebra terms. Additionally, we showed that the graph shifting of a graph signal consists of local operations. This will be our basic tool to perform distributed operations over the graph in Chapters 3-4.



We then characterized the variability of the signal over the graph by means of the $p$-Dirichlet form. This allowed us to introduce the concept of the graph Fourier transform, where the signal is expressed as a linear combination of the oscillating modes of the graph. These modes turned out to be the eigenvectors of the graph shift operator matrix, and they extend the signal harmonic analysis from the temporal domain to the graph dimension.

Subsequently, we introduced the concept of graph filtering to alter the graph frequency content of graph signals. We showed that linear graph filters can be implemented directly in the vertex domain as a matrix-vector multiplication. Additionally, we showed that Tikhonov regularization on graphs consists of a graph filtering operation.

Finally, the notion of weak stationarity on graphs is introduced. The latter allowed us to formulate the graph Wiener regularization problem as a graph filtering operation. We also showed that the GFT of stationary graph signals coincides with the KLT transform.

## FURTHER READING

[1] 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 **30**, 83 (2013).

[2] D. I. Shuman, B. Ricaud, and P. Vandergheynst, *Vertex-frequency analysis on graphs,* Applied and Computational Harmonic Analysis **40**, 260 (2016).

[3] A. Sandryhaila and J. M. Moura, *Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure,* IEEE Signal Processing Magazine **31**, 80 (2014).

[4] A. Sandryhaila and J. M. Moura, *Discrete signal processing on graphs,* IEEE transactions on signal processing **61**, 1644 (2013).

[5] A. Sandryhaila and J. M. Moura, *Discrete signal processing on graphs: Frequency analysis.* IEEE Trans. Signal Processing **62**, 3042 (2014).

[6] F. R. Chung, *Spectral graph theory*, 92 (American Mathematical Soc., 1997).

[7] D. K. Hammond, P. Vandergheynst, and R. Gribonval, *Wavelets on graphs via spectral graph theory,* Applied and Computational Harmonic Analysis **30**, 129 (2011).

[8] C. Godsil and G. F. Royle, *Algebraic graph theory*, Vol. 207 (Springer Science & Business Media, 2013).

[9] M. Puschel and J. M. Moura, *Algebraic signal processing theory: 1-d space,* IEEE Transactions on Signal Processing **56**, 3586 (2008).

[10] M. Puschel and J. M. Moura, *Algebraic signal processing theory: Foundation and 1-d time,* IEEE Transactions on Signal Processing **56**, 3572 (2008).



[11] R. Singh, A. Chakraborty, and B. Manoj, *Graph Fourier transform based on directed Laplacian,* in *Signal Processing and Communications (SPCOM), 2016 International Conference on* (IEEE, 2016) pp. 1–5.

[12] H. Sevi, G. Rilling, and P. Borgnat, *Multiresolution analysis of functions on directed networks,* in *Wavelets and Sparsity XVII*, Vol. 10394 (International Society for Optics and Photonics, 2017) p. 103941Q.

[13] A. Ortega, P. Frossard, J. Kovačević, J. M. Moura, and P. Vandergheynst, *Graph signal processing,* arXiv preprint arXiv:1712.00468 (2017).

[14] A. Barrat, M. Barthelemy, and A. Vespignani, *Dynamical processes on complex networks* (Cambridge university press, 2008).

[15] S. P. Chepuri, S. Liu, G. Leus, and A. O. Hero, *Learning sparse graphs under smoothness prior,* in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on* (IEEE, 2017) pp. 6508–6512.

[16] C. Zhang, D. Florêncio, and P. A. Chou, *Graph signal processing–a probabilistic framework,* Microsoft Res., Redmond, WA, USA, Tech. Rep. MSR-TR-2015-31 (2015).

[17] S. Segarra, A. G. Marques, G. Mateos, and A. Ribeiro, *Network topology inference from spectral templates,* IEEE Transactions on Signal and Information Processing over Networks (2017).

[18] V. Kalofolias, *How to learn a graph from smooth signals,* in *Artificial Intelligence and Statistics* (2016) pp. 920–929.

[19] J. Mei and J. M. Moura, *Signal processing on graphs: Causal modeling of unstructured data,* IEEE Transactions on Signal Processing **65**, 2077 (2017).

[20] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst, *Learning laplacian matrix in smooth graph signal representations,* IEEE Transactions on Signal Processing **64**, 6160 (2016).

[21] N. Thanou, *Graph Signal Processing: Sparse Representation and Applications*, Ph.D. thesis, Ecole Polytechnique Fédérale de Lausanne (2016).

[22] G. B. Giannakis, Y. Shen, and G. V. Karanikolas, *Topology identification and learning over graphs: Accounting for nonlinearities and dynamics,* Proceedings of the IEEE **106**, 787 (2018).

[23] J. Kleinberg, R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins, *The web as a graph: measurements, models, and methods,* Computing and combinatorics , 1 (1999).

[24] M. Newman, *Networks: An Introduction* (Oxford university press, 2010).

[25] X. Zhu, Z. Ghahramani, and J. D. Lafferty, *Semi-supervised learning using gaussian fields and harmonic functions,* in *Proceedings of the 20th International conference on Machine learning (ICML-03)* (2003) pp. 912–919.

[26] R. A. Horn and C. R. Johnson, *Matrix analysis* (Cambridge university press, 2012).

[27] T. Kailath, *Linear systems*, Vol. 156 (Prentice-Hall Englewood Cliffs, NJ, 1980).

[28] M. Elad, *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing* (Springer, 2010).

[29] S. Chen, R. Varma, A. Sandryhaila, and J. Kovačević, *Discrete signal processing on graphs: Sampling theory,* IEEE Transactions on Signal Processing **63**, 6510 (2015).

[30] M. Tsitsvero, S. Barbarossa, and P. Di Lorenzo, *Signals on graphs: Uncertainty principle and sampling,* IEEE Transactions on Signal Processing **64**, 4845 (2016).

[31] A. G. Marques, S. Segarra, G. Leus, and A. Ribeiro, *Sampling of graph signals with successive local aggregations,* IEEE Transactions on Signal Processing **64**, 1832 (2016).

[32] C. Groetsch, *The theory of Tikhonov regularization for Fredholm equations,* Boston Pitman Publication (1984).

[33] G. H. Golub, P. C. Hansen, and D. P. O'Leary, *Tikhonov regularization and total least squares,* SIAM Journal on Matrix Analysis and Applications **21**, 185 (1999).

[34] A. J. Smola and R. Kondor, *Kernels and regularization on graphs,* in *COLT*, Vol. 2777 (2003) pp. 144–158.

[35] A. Elmoataz, O. Lezoray, and S. Bougleux, *Nonlocal discrete regularization on weighted graphs: a framework for image and manifold processing,* IEEE transactions on Image Processing **17**, 1047 (2008).

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

[37] N. Perraudin and P. Vandergheynst, *Stationary signal processing on graphs,* IEEE Transactions on Signal Processing **65**, 3462 (2017).

[38] A. G. Marques, S. Segarra, G. Leus, and A. Ribeiro, *Stationary graph processes and spectral estimation,* IEEE Transactions on Signal Processing (2017).

[39] B. Girault, P. Gonçalves, E. Fleury, and A. S. Mor, *Semi-supervised learning for graph to signal mapping: A graph signal wiener filter interpretation,* in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on* (IEEE, 2014) pp. 1115–1119.

[40] E. Isufi, A. Loukas, A. Simonetto, and G. Leus, *Autoregressive moving average graph filtering,* IEEE Transactions on Signal Processing **65**, 274 (2017).

[41] R. Dony *et al.*, *Karhunen-loeve transform,* The transform and data compression handbook **1**, 1 (2001).

# II

# GRAPH FILTERING

# 3

# FINITE IMPULSE RESPONSE GRAPH FILTERING

*If it's the right chair, it doesn't take too long to get comfortable in it.*

Robert De Niro

This chapter introduces a particular type of graph filter recursion, named *finite impulse response* (FIR) graph filter. This recursion allows us to implement the graph filtering operation described in Section 2.3.5 distributively over the graph. Moreover, by making use of the locality of the graph shift operator [cf. Section 2.2.3], we will show that the FIR filters enjoy an efficient implementation in the vertex domain which has a complexity smaller than $O(N^2)$, i.e., matrix-vector multiplication. Similarly to its temporal homonym, the FIR graph filter will alter the graph signal spectra with an impulse response that is finite, now, in the vertex domain.

Prior works, relevant to this chapter are [2, 3] and [4]. Shuman and co-authors advocate in [2] the use of Chebyshev polynomials to perform distributed signal processing tasks. Sandryhaila and Moura in [3] formalized the concept of FIR filtering on graphs and propose a least-squares (LS) filter design approach. We will refer to these FIR graph filters as *distributed node-invariant FIR graph filters*. Segarra and co-authors developed in [4] the concept of *distributed node-variant FIR filter*, which implements distributively a broader class of signal processing operations. In the co-authored work [1], we take one more step further and introduce the *distributed edge-variant FIR graph filters*, which contains the former FIR approaches as special cases.

The next section highlights our contributions in this direction and lists several applications where the FIRs are adopted. The FIR filtering recursions in the vertex domain are described in Section 3.2. The filter design strategies for the node-invariant and node-variant FIR graph filters are shown in Section 3.3. The proposed distributed edge-variant FIR graph filter is then introduced in Section 3.4. Finally, Section 3.5 provides a chapter summary and draws conclusions.

Parts of this chapter have been published in the IEEE CAMSAP Workshop (2017) [1].

## 3.1. INTRODUCTION

In Section 2.3.5 of Chapter 2, we defined the graph filters as the tool to shape the graph spectral content of a signal on the graph. The described procedure consisted of first applying the GFT to the signal of interest, then applying a desired function to the signal spectral content [cf. (2.20)], and finally obtaining the signal in the vertex domain through the IGFT. In this chapter, we will answer our research question (Q1.1) and develop tools to process the graph signal directly in the vertex domain by a finite number of operations, avoiding the implementation of GFT-IGFT. This way of proceeding brings two main advantages. First, it reduces the computational costs from $O(N^3)$ due to the eigendecomposition of the shift operator to $O(MK)$, where recall that $M$ is the number of edges in the graph and $K$ in the number of operations performed in the vertex domain, referred to as the filter order. Second, through the locality of the shift operator [cf. Section 2.2.3], it will allow us to implement these recursions distributively in the vertex domain. On the downside, the price to pay for this reduced cost is that the obtained operation will be an approximation of the desired operation and the accuracy will depend on the type of recursion and the filter order $K$.

We start our analysis by first revisiting the state-of-the-art distributed FIR recursion on graphs along with their design strategies and then we introduce the first thesis contribution, i.e., the distributed edge-variant FIR graph filter. In the sequel, we conclude this section with a detailed list of contributions and potential applications of the FIR graph filters.

### 3.1.1. CONTRIBUTIONS

Within the context of FIR graph filtering, this chapter introduces the following contributions:

**Contribution 3.1.** *We propose a novel algorithm to implement FIR graph filters distributively.* We introduce an important change to the traditional way of implementing FIR recursions on graphs, which yield an increment of the degrees of freedom (DoFs). The latter allows us to significantly reduce the filter implementation costs while keeping a similar distributed implementation as prior FIR graph filters.

**Contribution 3.2.** *We propose an optimization scheme for designing the filter frequency response of the introduced FIR graph filter.* The proposed filter design approach tackles the challenges introduced by the increment of the DoFs and uses standard convex optimization techniques to find the filter coefficients.

There are yet two important practical aspects with regards to the introduced approach:

- In the filter design phase, the computational cost in solving the optimization problem may often be high. Though being a computational issue, the filter design can be performed offline, and we can allow ourselves to use an extensive computational power to solve it.

- The reduction of the implementation costs for the introduced filter leads to significant computational and communication gain in distributed networks. Specifically, it requires less data exchange between adjacent nodes to meet a target filtering performance.

### 3.1.2. APPLICATIONS

With a bulk of literature on GSP and graph filters, we here illustrate some of the key applications where the latter distinguish themselves better.



- *Graph signal denoising.* As illustrated with both Tikhonov and Wiener-based denoising [cf. Sections 2.3.6 and 2.4.2, respectively], one of the central applications of graph filters is to remove spurious noise perturbations from the signal of interest. The FIR graph filters can subsequently be used to distributively clean noisy sensor measurements, or to suppress outliers.

- *Graph signal interpolation.* In sensor networks, because of energy constraints or local failures, several sensors may not be able to observe the graph signal. In this case, by means of graph filtering, we are interested to distributively learn these missing values by only local exchanges between sensors. This, for instance, can be the case that the signal of interest presents some properties w.r.t. the underlying graph and the graph filter can solve a Tikhonov-, or Wiener-based interpolation problem.

- *Spectral clustering.* One of the main strategies in partitioning a graph $\mathcal{G}$ of $N$ nodes in $k$ inter-connected clusters (a group of nodes that share similar properties, such as political orientation) has as the first step the computation of the first $k$ eigenvectors of the Laplacian matrix [5]. As the number of nodes $N$ grow large, we run in a computational bottleneck since the eigendecomposition cost of the graph Laplacian is of order $O(N^3)$. To this end, graph filters can play a role to isolate the first $k$ eigenvectors of the graph Laplacian with a lower cost, by simply filtering random signals [6]. Subsequently, low order graph filters with high approximation accuracy are required to further reduce the costs in very large graphs.

- *Network coding.* With the massive explosion of sensor deployment in the internet-of-things, network coding sets itself as an effective mechanism to improve the task of routing. With network coding, relay sensors do not simply receive-and-forward a packet, say $x$ , but rather they mix the received packet content with their own packet, say $y$, and transmit the mixed packet, say $z = x + y$. The destination node can then retrieve the original message, by simply collecting a sufficient number of packets from different relay sensors. This strategy has shown to be of great advantage in both wireless sensor networks [7], and underwater sensor networks [8]. In terms of GSP, [4] has shown that analog network coding can be performed distributively with FIR recursion on graphs since the latter allows nodes to mix their own signal with that of their neighbors upon transmission. Then, to improve the overall energy efficiency of the network a low-order FIR filter is required for achieving the imposed performance with fewer communication costs.

- *Analyzing brain signals.* In neuroscience, the partition of the brain in several regions, where interconnections between them can be expressed with a graph has shown promising results in analyzing the neurological and individual behavior of patients [9]. GSP posed itself recently as a novel approach to analyze functional brain networks [10, 11]. In the latter works, a graph spectral analysis of brain signals has shown that users with different level of adaptability throughout learning exhibit different graph spectral content. Then, a graph filter can isolate different parts of the brain signal spectrum, which can then be used to discriminate between users of different learning levels.



Overall, we remark that these are only a few of graph filter applications. As the field of GSP is growing fast, we believe that graph filters will find even more applications in the tomorrow tasks over networks, such as non-Euclidean deep learning [12], recommendation systems [13], and data classification and compression [3], to name a few. Furthermore, in a more technical aspect, graph filters serve also as a basic building block for graph filter banks [14], and graph wavelets [15].

## 3.2. FILTERING IN THE VERTEX DOMAIN

The FIR filter output $y_i$ at vertex $v_i$ consists of a linear combination of the input signal $x_j$ of the $K$-hops away neighbors $\mathcal{N}_{(i,K)}$ of $v_i$, i.e.,

$$y_i = \phi_{i,i} x_i + \sum_{j \in \mathcal{N}_{(i,K)}} \phi_{i,j} x_j, \tag{3.1}$$

for some complex scalars $\phi_{i,j}$ referred to as filter coefficients. That is, (3.1) says that a $K$th order FIR graph filter ($\text{FIR}_K$) is a localized linear operation and considers information from nodes that are at most $K$ edges away from $v_i$. Next, we analyze two recursions that allow the implementation of (3.1) with only local communications.

### 3.2.1. NODE-INVARIANT FIR FILTERING

One way to relate the frequency domain filtering (2.20), with localized operations in the vertex domain is to express $h(\mathbf{\Lambda})$ as polynomials of the form

$$h(\mathbf{\Lambda}) = \sum_{k=0}^{K} \phi_k \mathbf{\Lambda}^k, \tag{3.2}$$

i.e., the filter frequency response at the $i$th graph frequency $h(\lambda_i) = \sum_{k=0}^{K} \phi_k \lambda_i^k$ is a $K$th order polynomial at that particular frequency. The $\phi_k$s are again complex scalar coefficients. By substituting (3.2) into (2.20) the filter output in the spectral domain is

$$\hat{\mathbf{y}} = \sum_{k=0}^{K} \phi_k \mathbf{\Lambda}^k \hat{\mathbf{x}}, \tag{3.3}$$

**Algorithm 3.1.** Distributed computation of the node-invariant FIR output

1: Initialize the filter coefficients $\phi_0, \ldots, \phi_K$ and set $\mathbf{x}^{(0)} = \mathbf{x}$
2: **procedure** COMPUTE LOCALLY THE FILTER OUTPUT
3: **for** k = 1, ..., K **do**
4: Collect $x_m^{(k-1)}$ from all neighbors $m \in \mathcal{N}_n$
5: Compute $x_n^{(k)} = \sum_{m \in \mathcal{N}_n} W_{n,m}\left(x_n^{(k-1)} - x_m^{(k-1)}\right)$
6: Send $x_n^{(k)}$ to all neighbors $\mathcal{N}_n$
7: Set $y_n = \sum_{k=0}^{K} \phi_k x_n^{(k)}$



which by means of the IGFT becomes

$$\begin{aligned} \mathbf{y} = \mathbf{U}\left(\sum_{k=0}^{K} \phi_k \mathbf{\Lambda}^k\right)\mathbf{U}^{\mathsf{H}}\mathbf{x} = \left(\sum_{k=0}^{K} \phi_k \mathbf{U}\mathbf{\Lambda}^k\mathbf{U}^{\mathsf{H}}\right)\mathbf{x} \\ \overset{(a)}{=} \left(\sum_{k=0}^{K} \phi_k (\mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathsf{H}})^k\right)\mathbf{x} = \sum_{k=0}^{K} \phi_k \mathbf{S}^k\mathbf{x}. \end{aligned} \tag{3.4}$$

Equality $(a)$ in (3.4) holds since $(\mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathsf{H}})^k = \mathbf{U}\mathbf{\Lambda}^k\mathbf{U}^{\mathsf{H}}$ due to the normality of matrix $\mathbf{U}$. Finally, from (2.22) and (3.4), the filter input-output are related by the filter impulse response

$$\mathbf{H} = \sum_{k=0}^{K} \phi_k \mathbf{S}^k, \tag{3.5}$$

i.e., it is a polynomial of order $K$ in the graph shift operator $\mathbf{S}$.

To see that (3.4) consists of local operations in the vertex domain and that it can be implemented distributively, let us expand (3.4) as

$$\mathbf{y} = \phi_0\mathbf{I}_N\mathbf{x} + \phi_1\mathbf{S}\mathbf{x} + \ldots + \phi_K\mathbf{S}^K\mathbf{x}, \tag{3.6}$$

where, by recalling the shifted versions of the input signal $\mathbf{x}^{(k)} = \mathbf{S}^k\mathbf{x}$ (cf. Section 2.2.3), we write

$$\mathbf{y} = \phi_0\mathbf{x}^{(0)} + \phi_1\mathbf{x}^{(1)} + \ldots + \phi_K\mathbf{x}^{(K)}. \tag{3.7}$$

Recursion (3.7) expresses the filter output $\mathbf{y}$ as a linear combination of the first $K$ shifted versions of the input signal over the graph. Then, as reported in Section 2.2.3, each node can compute its output $y_i$ by locally exchanging previous shifted versions of the input signal with its direct neighbors, i.e., $\mathbf{x}^{(1)} = \mathbf{S}\mathbf{x}$, $\mathbf{x}^{(2)} = \mathbf{S}\mathbf{x}^{(1)}$ and so forth. The latter suggests that the FIR filtering operation is a localized linear transform in the vertex domain, and it can be performed distributively over the network. Algorithm 3.1 resembles this procedure for $\mathbf{S} = \mathbf{L}$.

Since all vertices apply the same coefficient $\phi_j$ to the $j$th shifted version of $\mathbf{x}$ in (3.4)-(3.7), to this implementation it is referred to as the *node-invariant* FIR graph filter.

**Algorithm 3.2.** Distributed computation of the node-variant FIR output

1: Initialize the filter coefficients $\boldsymbol{\phi}^{(0)},\dots,\boldsymbol{\phi}^{(K)}$ and set $\mathbf{x}^{(0)}=\mathbf{x}$
2: **procedure** COMPUTE LOCALLY THE FILTER OUTPUT
3: **for** k = 1, ..., K **do**
4: Collect $x_m^{(k-1)}$ from all neighbors $m\in\mathcal{N}_n$
5: Compute $x_n^{(k)}=\sum_{m\in\mathcal{N}_n} W_{n,m}\left(x_n^{(k-1)}-x_m^{(k-1)}\right)$
6: Send $x_n^{(k)}$ to all neighbors $\mathcal{N}_n$
7: Set $y_n=\sum_{k=0}^{K}\phi_n^{(k)}x_n^{(k)}$



### 3.2.2. NODE-VARIANT FIR FILTERING

In [4], the authors extend the node-invariant FIR filter (3.5) to a *node-variant* version by allowing nodes to apply different coefficients while shifting the signal. Specifically, they define a node-variant FIR filter of order $K$ as

$$\mathbf{H}_{\text{nv}}=\sum_{k=0}^{K}\text{diag}(\boldsymbol{\phi}^{(k)})\mathbf{S}^k, \tag{3.8}$$

where $\boldsymbol{\phi}^{(k)}=(\phi_1^{(k)},\dots,\phi_N^{(k)})^{\mathsf{T}}$ is the $N\times 1$ vector of filter coefficients that each node applies to all its neighbors at shift $k$. Observe that the node invariant FIR (3.5) is a particular case of (3.8) and can be obtained by setting $\boldsymbol{\phi}^{(k)}=\phi_k\mathbf{1}_N$.

The main advantage of the node-variant graph filter (3.8) is that it does not affect the distributed local implementation of the filter with a larger number of DoFs (i.e., filter coefficients) that can be exploited in the design phase to approximate a larger set of operations on graphs. For consistency, Algorithm 3.2 illustrates how the output

$$\mathbf{y}=\sum_{k=0}^{K}\text{diag}(\boldsymbol{\phi}^{(k)})\mathbf{S}^k\mathbf{x}, \tag{3.9}$$

of a node-variant FIR is computed distributively for $\mathbf{S}=\mathbf{L}$.

There is yet a crucial difference between the node-variant graph filter (3.8) and the node-invariant graph filter (3.5). Specifically, since $\text{diag}(\boldsymbol{\phi}^{(k)})$ and $\mathbf{S}^{(k)}$ do not share the eigenvectors, it is not possible to have a spectral interpretation of (3.8) in terms of (3.2). This, to some extent, limits our understanding on how (3.8) will shape the graph spectral content of the input and, therefore, allows for filter design only in the vertex domain (See later on Section 3.3.4.). However, some insight on the frequency representation of this filer is provided by [4], which we report here for completeness.

By rewriting (3.8) as

$$\mathbf{H}_{\text{nv}}=\sum_{k=0}^{K}\text{diag}(\boldsymbol{\phi}^{(k)})\mathbf{U}\boldsymbol{\Lambda}^k\mathbf{U}^{\mathsf{H}}, \tag{3.10}$$

the effect of the filter on the $v_i$th node is given by the $i$th row of $\mathbf{H}_{\text{nv}}$, i.e.,

$$\begin{aligned}\mathbf{h}_i^{\mathsf{T}} = \mathbf{e}_i^{\mathsf{T}}\mathbf{H}_{\text{nv}} &= \sum_{k=0}^{K} \mathbf{e}_i^{\mathsf{T}} \operatorname{diag}(\boldsymbol{\phi}^{(k)})\mathbf{U}\boldsymbol{\Lambda}^k\mathbf{U}^{\mathsf{H}} \\ &= \sum_{k=0}^{K} [\operatorname{diag}(\boldsymbol{\phi}^{(k)})]_{i,i}\mathbf{e}_i^{\mathsf{T}}\mathbf{U}\boldsymbol{\Lambda}^k\mathbf{U}^{\mathsf{H}},\end{aligned} \tag{3.11}$$

where $[\operatorname{diag}(\boldsymbol{\phi}^{(k)})]_{i,i}$ is the $i$th diagonal element of $\operatorname{diag}(\boldsymbol{\phi}^{(k)})$, i.e., the filter tap that vertex $v_i$ applies to the $k$th shift. Then, being $\mathbf{u}_i^{\mathsf{T}} = \mathbf{e}_i^{\mathsf{T}}\mathbf{U}$ the $i$th row of $\mathbf{U}$, the node-variant output at the $v_i$th node is

$$y_i = \mathbf{h}_i^{\mathsf{T}}\mathbf{x} = \mathbf{u}_i^{\mathsf{T}}\left(\sum_{k=0}^{K} [\operatorname{diag}(\boldsymbol{\phi}^{(k)})]_{i,i}\boldsymbol{\Lambda}^k\right)\hat{\mathbf{x}}. \tag{3.12}$$

Finally, the above equation leads to the following observations:

- $\hat{\boldsymbol{\Phi}}_i = \sum_{k=0}^{K} [\operatorname{diag}(\boldsymbol{\phi}^{(k)})]_{i,i}\boldsymbol{\Lambda}^k$ is the graph frequency response of a node-invariant FIR graph filter of order $K$ having as filter taps the $i$th node coefficients $[\operatorname{diag}(\boldsymbol{\phi}^{(k)})]_{i,i}$.
- The $v_i$th node output $y_i$ can be viewed as the inner product between the GFT of the input signal, $\hat{\mathbf{x}}$, and the impact that the graph spectrum has on $v_i$, $\mathbf{u}_i$ (i.e., how strongly $v_i$ senses all graph frequencies), all modulated by the $v_i$th node FIR graph filter $\hat{\boldsymbol{\Phi}}_i$.

By redirecting the reader to the original work [4] for further detail, we conclude this section with the following remark.

**Remark 3.1.** *An alternative implementation of the node-variant FIR$_K$* (3.8) *is the recursion*

$$\mathbf{H}'_{\text{nv}} = \sum_{k=0}^{K} \mathbf{S}^k \operatorname{diag}(\boldsymbol{\phi}^{(k)}), \tag{3.13}$$

*where now the filter first modulates the input signal with the coefficients* $\operatorname{diag}(\boldsymbol{\phi}^{(k)})$ *and then applies the graph shift. Implementation* (3.8) *considers the opposite, i.e., first to shift the input signal and then to modulate it by the nodes coefficients. Observe that* (3.13) *again specializes in the node-invariant filter* (3.5) *for $\boldsymbol{\phi}^{(k)} = \phi_k \mathbf{1}_N$, and preserves a similar localized distributed implementation as the other two candidates [4].*

### 3.2.3. DISTRIBUTED COSTS

To analyze the distributed costs of the previously introduced FIR recursions, let us first remark that except the filter taps, which are set locally by nodes, both the node-invariant and the node-variant FIRs have the same distributed strategy. Consequently, the following analysis applies to both recursions.

In performing the filter *distributedly* on the network $\mathcal{G}$, we assume that each node $v_i \in \mathcal{V}$ is imbued with memory, computation, and communication capabilities and is in charge of calculating the local filter output $[\mathbf{Hx}]_i$. To do so, the node has to its disposal

direct access to $x_i$, as well as indirect access to the memory of its neighbors. For simplicity of presentation, we pose an additional restriction to the computation model: we will assume that nodes operate in synchronous rounds, each one consisting of a message exchange phase and a local computation phase. In other words, in each round, $v_i$ may compute any (polynomial-time computable) function which has as input, variables from its local memory as well as those from the memory of its neighbors in $\mathcal{G}$. Since the examined algorithms are, in effect, dynamical systems, in the following we will adopt the term *iteration* as being synonymous with rounds. Furthermore, we assume that each iteration lasts exactly one time instant.



For the distributed computation of a $K$th order FIR output, recall that $\mathbf{S}^K\mathbf{x} = \mathbf{S}\left(\mathbf{S}^{K-1}\mathbf{x}\right)$ ,i.e., each node $v_i$ can compute the $K$-th term from the values of the $(K-1)$th term in its neighborhood, in an iterative manner. The algorithm performing the $\text{FIR}_K$ graph filter terminates after $K$ iterations, and if a more efficient recursive implementation is used [2], in total, each node $v_i$ exchanges $K\deg(v_i)$ values with its neighbors, meaning that overall, the network exchanges $NK\deg(v_i)$ variables, amounting to a communication cost of $O(MK)$. This computational complexity is in general much smaller than $O(N^2)$ (i.e., matrix-vector multiplications) since practical graphs are relatively sparse, i.e., they contain few edges, and the filter order $K$ is often chosen much smaller than $N$.

Finally, the above analysis sheds light on the impact that the filter order $K$ has on the filter distributed implementation. On one hand, a high order $K$ is preferred to improve the filter approximation accuracy. On the other hand, this increment of the approximation accuracy is traded with more exchanges between nodes, thus higher costs.

## 3.3. FILTER DESIGN

We now focus on the task of finding the filter taps that approximate a given frequency response $h^*(\mathbf{\Lambda})$, or a filtering operation matrix $\mathbf{H}^*$. As the node-invariant implementation is the most widely used, most of this section is dedicated to this recursion, while in Section 3.3.4 we touch the design strategy for the node-variant FIR filter, as well.

### 3.3.1. FREQUENCY AWARE VERSUS UNIVERSAL DESIGN

There are two different strategies of finding the graph filter coefficients: $(i)$ the *frequency aware design*, where the specific values of the graph frequencies are known; and $(ii)$ the *universal* design, where only the interval range where the graph frequencies are contained is known.

**Frequency aware design.** This scenario considers that the eigendecomposition $\mathbf{S} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathsf{H}}$ is computationally feasible and the goal is to find the filter coefficients $\phi_k$ in (3.5) that approximate a desired frequency response $h^*(\mathbf{\Lambda})$. By exploiting the structure of the $\text{FIR}_K$ graph filter (3.5), the solution for the filter coefficients consists of solving the linear

system

$$\begin{aligned}
\phi_0 + \phi_1\lambda_0 + \ldots + \phi_K\lambda_0^K &= h^*(\lambda_0),\\
\phi_0 + \phi_1\lambda_1 + \ldots + \phi_K\lambda_1^K &= h^*(\lambda_1),\\
&\vdots\\
\phi_0 + \phi_1\lambda_{L-1} + \ldots + \phi_K\lambda_{L-1}^K &= h^*(\lambda_{L-1}),
\end{aligned} \tag{3.14}$$

where we consider $\mathbf{S}$ to have $L \leq N$ distinct eigenvalues. The above linear system of $L$ equations and $K+1$ unknowns can then be written in the matrix form as

$$\begin{pmatrix} 1 & \lambda_0 & \cdots & \lambda_0^K \\ 1 & \lambda_1 & \cdots & \lambda_1^K \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \lambda_{L-1} & \cdots & \lambda_{L-1}^K \end{pmatrix} \begin{pmatrix} \phi_0 \\ \phi_1 \\ \vdots \\ \phi_K \end{pmatrix} = \begin{pmatrix} h^*(\lambda_0) \\ h^*(\lambda_1) \\ \vdots \\ h^*(\lambda_{L-1}) \end{pmatrix}, \tag{3.15}$$

which has a full-rank $L\times(K+1)$ Vandermonde system matrix [16]. Note that (3.15) has: $(i)$ a unique solution if $L = K+1$, i.e., the filter taps are in number the same as the distinct graph frequencies; $(ii)$ infinite many solutions if $L \leq K$; and $(iii)$ no-solution for $L \geq K+1$.

As stated earlier, a major challenge of this design strategy is the eigendecomposition cost of $\mathbf{S}$, which practically would make appropriate if we want to perform an optimal filtering task distributively. In a centralized implementation, on the other hand, the eigendecomposition of $\mathbf{S}$ allows us to directly process the GFT of the signal, rendering this approach suboptimal.

**Universal design.** The universal filter design is blind about the specific eigenvalues $\mathbf{S}$ and assumes knowledge about the interval $[\lambda_{\min}, \lambda_{\max}]$ where the graph frequencies lie. Computationally, both $\lambda_{\min}$ and $\lambda_{\max}$ can be estimated with a much lower cost, or when $\mathbf{S} = \mathbf{L}_{\text{n}}$ is chosen we have $\lambda \in [0,2]$. The desired frequency response $h^*(\lambda)$ is then a continuous function in the graph frequency $\lambda$ and this design strategy aims at finding the filter taps that approximate $h^*(\lambda)$ over the whole interval $[\lambda_{\min}, \lambda_{\max}]$. Figure 3.1 illustrates one example and highlight the differences with the frequency aware design.

Besides avoiding the eigendecomposition of $\mathbf{S}$, the universal design is beneficial in those cases where the underlying structure is unknown to the designer, or in cases the topology changes in time[1]. The corresponding filter coefficients are thus independent of the graph and universally applicable. Therefore, we can design a single set of coefficients that instantiate the same graph frequency response $h^*(\lambda)$ over the different bases.

To further illustrate the universality property, consider the application of a universal graph filter to two different graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ of $N_1$ and $N_2$ nodes with graph frequency sets $\{\lambda_{1,n}\}_{n=0}^{N_1-1}$, $\{\lambda_{2,n}\}_{n=0}^{N_2-1}$, and eigenvectors $\mathbf{U}_1 = \{\mathbf{u}_{1,n}\}_{n=0}^{N_1-1}$, $\mathbf{U}_2 = \{\mathbf{u}_{2,n}\}_{n=0}^{N_2-1}$. The filter will attain the same response $h^*(\lambda)$ over both graphs, but, in each case, supported over a different set of graph frequencies: For $\mathcal{G}_1$, filtering results in $\mathbf{y}_1 = \mathbf{U}_1 \sum_{n=0}^{N_1-1} h^*(\lambda_{1,n})\mathbf{U}_1^{\mathsf{H}}\mathbf{x}$, whereas for $\mathcal{G}_2$ the filtering result is $\mathbf{y}_2 = \mathbf{U}_2 \sum_{n=0}^{N_2-1} h^*(\lambda_{2,n})\mathbf{U}_2^{\mathsf{H}}\mathbf{x}$. Thus, the universality of

[1] An example of topological changes may be moving sensors in a sensor network.

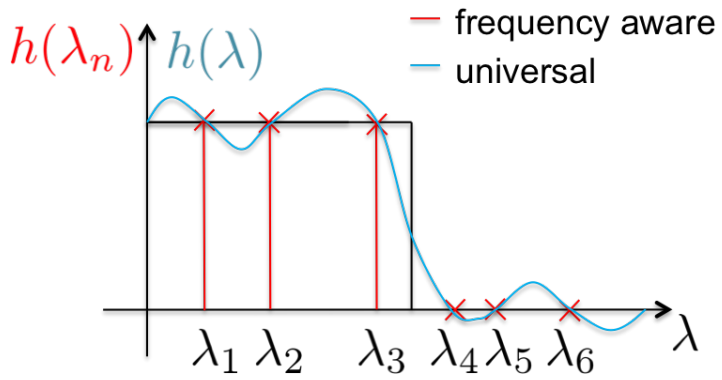



Figure 3.1: Illustration of the desired frequency response (black solid line) and its approximations with a frequency aware and universal design.

this approach lies in the correctness to implement $h^*(\lambda)$ on all graphs, yet with different outputs in different topologies.

In the upcoming three sections, we focus more on technical detail about these design strategies and propose ways to find the FIR filter coefficients.

### 3.3.2. LINEAR LEAST SQUARES-BASED DESIGN

As mentioned in (3.15), for filter orders $K+1 \leq L$ the linear system does not have a solution. However, in practice, we are interested to have $K \ll L$ to keep limited the computational and communication cost. The filter coefficients $\boldsymbol{\phi} = (\phi_0, \ldots, \phi_K)^\mathsf{T}$ are then found with a least squares approach, i.e.,

$$\boldsymbol{\phi} = \boldsymbol{\Psi}_K^\dagger \mathrm{diag}(h^*(\boldsymbol{\Lambda})), \tag{3.16}$$

where $\boldsymbol{\Psi}_K$ denotes the Vandermonde matrix in (3.15) and $h^*(\boldsymbol{\Lambda}) = \mathrm{diag}(h^*(\lambda_0), h^*(\lambda_1), \ldots, h^*(\lambda_{L-1}))$. The founded filter coefficients $\boldsymbol{\phi}$ provide a frequency response that approximates $\mathbf{h}^*(\boldsymbol{\Lambda})$ with a polynomial function, yet the optimal approximation in a squared-error sense.

For the universal design, the least-squares approach finds the filter coefficients $\phi$ as the solution of

$$\boldsymbol{\phi} = \underset{\phi_0,\ldots,\phi_K}{\mathrm{argmin}} \int_\lambda \left| \sum_{k=0}^{K} \phi_k \lambda^k - h^*(\lambda) \right|^2 \mathrm{d}\lambda, \tag{3.17}$$

which by (fine) gridding the frequency range $[\lambda_{\min}, \lambda_{\max}]$ in $L'$ (not necessarily smaller than $N$) points becomes

$$\boldsymbol{\phi} = \underset{\phi_0,\ldots,\phi_K}{\mathrm{argmin}} \sum_{l=0}^{L'-1} \left| \sum_{k=0}^{K} \phi_k \lambda_l^k - h^*(\lambda_l) \right|^2, \tag{3.18}$$

and can reformulated and solved similar to (3.16).

### 3.3.3. CHEBYSHEV POLYNOMIAL-BASED DESIGN

Another approach initially introduced by [15] for approximation purposes, and then elaborated by [2] in the context of graph filtering is to make use of Chebyshev polynomials for approximating the desired frequency response $h^*(\lambda)$.

Recall that for a scalar $x \in [-1,1]$, the $k$th Chebyshev polynomial of the first king (for short Chebyshev polynomial) $T_k(x)$ is

$$T_k(x) = \begin{cases} 1, & \text{if } k=0 \\ x & \text{if } k=1 \\ 2xT_{k-1}(x) - T_{k-2}(x) & \text{if } k \geq 2 \end{cases}. \tag{3.19}$$



These polynomials form an orthogonal basis for any function $h(x)$ on $[-1,1]$ that is square integrable w.r..t. the measure $dx/\sqrt{1-x^2}$. So, $h(x)$ can be represented as

$$h(x) = \frac{1}{2}b_0 + \sum_{k=1}^{\infty} b_k T_k(x), \tag{3.20}$$

where $\{b_k\}_{k=0}^{\infty}$ are the coefficients in the expansion of $h(x)$ in the Chebyshev polynomial basis [17].

To adopt the Chebyshev polynomials in approximating $h^*(\lambda)$ in the interval $[0,\lambda_{\max}]$, consider first the variable transformation $\lambda = \frac{\lambda_{\max}}{2}(x+1)$ to express $h^*(\lambda)$ as

$$h^*(\lambda) = \frac{1}{2}c_0 + \sum_{k=1}^{\infty} c_k \overline{T}_k(\lambda), \text{ for all } \lambda \in [0,\lambda_{\max}], \tag{3.21}$$

with

$$\begin{aligned} \overline{T}_k(\lambda) &:= T_k\left(\frac{\lambda-\gamma}{\gamma}\right), \\ \gamma &:= \frac{\lambda_{\max}}{2}, \\ c_k &:= \frac{2}{\pi}\int_0^{\pi} \cos(k\theta) h\Big(\gamma(\cos(\theta)+1)\Big) \mathrm{d}\theta. \end{aligned} \tag{3.22}$$

Then, for $k \geq 2$, the shifted Chebyshev polynomials can be computed recursively as

$$\overline{T}_k(\lambda) = \frac{2}{\gamma}(\lambda-\gamma)\overline{T}_{k-1}(\lambda) - \overline{T}_{k-2}(\lambda). \tag{3.23}$$

With this in place, we can find the first $K$ Chebyshev coefficients in (3.21) as the ones that approximate the continuous frequency response $h^*(\lambda)$ on the interval $[0,\lambda_{\max}]$. Subsequently, the output of a $K$th order Chebyshev polynomial FIR graph filter can be either computed iteratively as

$$\overline{T}_k(\mathbf{S})\mathbf{x} = \frac{2}{\gamma}(\mathbf{S}-\gamma\mathbf{I}_N)(\overline{T}_{k-1}(\mathbf{S})\mathbf{x}) - \overline{T}_{k-2}(\mathbf{S})\mathbf{x}, \tag{3.24}$$

or as

$$\overline{T}_K(\mathbf{S})\mathbf{x} = \sum_{k=0}^{K} \phi_k \mathbf{S}^k \mathbf{x}. \tag{3.25}$$

Observe that both (3.24)-(3.25) are localized distributed implementations, and (3.25) is simply obtained by expanding (3.24) to all its terms for some specific relation between the coefficients $\phi_k$ and $c_k$.

### 3.3.4. DESIGN IN THE VERTEX DOMAIN

So far we have been focused on designing the filter coefficients $\boldsymbol{\phi}$ in the graph frequency domain to approximate a given frequency response $h^*(\lambda)$. In this section, we consider the case of finding these coefficients in the vertex domain to approximate an operator matrix $\mathbf{H}^*$. An example of the latter consists of approximating a non-local operator $\mathbf{H}^*$ as a polynomial in $\mathbf{S}$ to allow for distributed computation. Moreover, this design strategy allows us to design the filter coefficients also for the node-variant FIR filter.

For the node-invariant FIR, the coefficients are found as the solution of

$$\boldsymbol{\phi} = \underset{\phi_0,\dots,\phi_K}{\operatorname{argmin}} \left\| \sum_{k=0}^{K} \phi_k \mathbf{S}^k - \mathbf{H}^* \right\|_2^2, \tag{3.26}$$

and similarly for the node-variant FIR we have

$$(\boldsymbol{\phi}^{(0)}, \boldsymbol{\phi}^{(1)}, \dots, \boldsymbol{\phi}^{(K)}) = \underset{\boldsymbol{\phi}^{(0)},\boldsymbol{\phi}^{(1)},\dots,\boldsymbol{\phi}^{(K)}}{\operatorname{argmin}} \left\| \sum_{k=0}^{K} \operatorname{diag}(\boldsymbol{\phi}^{(k)}) \mathbf{S}^k - \mathbf{H}^* \right\|_2^2. \tag{3.27}$$

Observe that both (3.26) and (3.27) are least squares convex problems, and can be solved efficiently with off-the-shelf algorithms [18]. Finally, as we now deal with matrix approximations, an alternative to the 2-norm in (3.26)-(3.27) is the Frobenius norm $\|\cdot\|_{\mathrm{F}}$.

### 3.3.5. DISCUSSIONS

With regards to the FIR graph filter design, we state the following:

- The benefit of the frequency aware design is that the obtained filter coefficients are the ones that minimize the approximation error on the specific set of graph frequencies that we are interested in. However, the eigendecomposition cost of $O(N^3)$ may result prohibitive in large graphs. Such an approach is mostly recommended for tasks that require the knowledge of the specific eigenvalues, such as the finite-time consensus [19].

- The universal design trades the eigendecomposition cost with approximation accuracy. Indeed, since the filter taps approximate $h^*(\lambda)$ on the continuous range $[0, \lambda_{\max}]$, the filter frequency response may result in a worse approximation accuracy on the specific graph frequencies $\{\lambda_n\}_{n=0}^{N}$.

- Differently, from the frequency aware design (3.15), the order of an FIR designed universally is not necessarily limited to $N$. In fact, by increasing $K$ we can approximate any square integrable frequency response arbitrarily well. On the other hand, a higher $K$ implies more communications in a distributed setting.

- FIR graph filters of orders $K$ close to $N$ may run into numerical issues when the eigenvalues of the shift operator are numerically close to each other. These issues are amplified in graphs with more than a couple hundreds of nodes.

- In the least-square universal design, it should be paid attention to the number of gridding points $L'$. From what said above, a very fine grid may run in overfitting issues. On the other hand, a sparse grid will encounter opposite challenges, i.e., the filter coefficients do not approximate good enough the frequency response in graph frequencies that are intermediate to two gridding points.

- Finally, observe that though we design the filter taps to approximate a function on the eigenvalues, i.e., $h^*(\lambda)$, the filter frequency response amplifies/suppresses the eigenvector contributions related to different eigenvalues. Specifically, for a graph with $\lambda_i = \lambda_j$, a filter with frequency response $h(\lambda_i) = h(\lambda_j)$ will act in the same way on two different eigenvectors (graph modes) $\mathbf{u}_i$ and $\mathbf{u}_j$.



## 3.4. DISTRIBUTED EDGE-VARIANT FIR GRAPH FILTERS

So far, we have considered two implementations of FIR graph filters: *i*) the node-invariant FIRs [cf. Section 3.2.1] where all nodes weight equally the same all their neighbor's signals; and *ii*) the node-variant FIRs [cf. Section 3.2.2] where different nodes weight differently all their neighbor's signals. In this section, we introduce a generalization of the NV FIRs, named the edge-variant (EV) FIR graph filter, where different nodes weight differently each of their neighbor's signals. As we shall see next, this improvement of the DoFs will lead to a reduction of the filter order, and hence of the distributed implementation costs.

### 3.4.1. EDGE-VARIANT FIR FILTERING

We define an EV FIR graph filter in the vertex domain as

$$\begin{aligned}\mathbf{H}_{\text{ev}} \triangleq \quad & \boldsymbol{\Phi}_0 + \boldsymbol{\Phi}_1 \odot \mathbf{S} + (\boldsymbol{\Phi}_2 \odot \mathbf{S})(\boldsymbol{\Phi}_1 \odot \mathbf{S}) + \dots \\ & + (\boldsymbol{\Phi}_K \odot \mathbf{S})(\boldsymbol{\Phi}_{K-1} \odot \mathbf{S}) \cdots (\boldsymbol{\Phi}_1 \odot \mathbf{S}) \\ = \quad & \sum_{k=1}^{K} \prod_{j=1}^{k} (\boldsymbol{\Phi}_j \odot \mathbf{S}) + \boldsymbol{\Phi}_0, \end{aligned} \tag{3.28}$$

where $\boldsymbol{\Phi}_j \in \mathbb{R}^{N \times N}$ are edge-weighting matrices that apply different weights to the entries of $\mathbf{S}$. That is, the EV filter (3.28) concedes the opportunity to each node to weight differently and independently the contribution of its neighbors in different shifts. Moreover, there is no symmetric structure imposed on the matrices $\boldsymbol{\Phi}_j$ and $\boldsymbol{\Phi}_0$ is considered to be a diagonal for (3.28) to enjoy a distributed implementation. Finally, observe that recursion (3.28) presents the same distributed implementation of the node-invariant and node-variant FIRs. Algorithm 3.3 illustrates the EV distributed implementation for $\mathbf{S} = \mathbf{L}$.

Given then the desired output

$$\mathbf{y}^* = \mathbf{H}^* \mathbf{x}, \tag{3.29}$$

**Algorithm 3.3.** Distributed computation of the edge-variant FIR output

1: Initialize the filter coefficients $\mathbf{\Phi}_0, \ldots, \mathbf{\Phi}_K$ and set $\mathbf{x}^{(0)} = \mathbf{x}$
2: **procedure** COMPUTE LOCALLY THE FILTER OUTPUT
3: **for** k = 1, . . . , K **do**
4: Collect $x_m^{(k-1)}$ from all neighbors $m \in \mathcal{N}_n$
5: Compute $x_n^{(k)} = \sum_{m \in \mathcal{N}_n} [\mathbf{\Phi}_k]_{n,m} W_{n,m} \left( x_n^{(k-1)} - x_m^{(k-1)} \right)$
6: Send $x_n^{(k)}$ to all neighbors $\mathcal{N}_n$
7: Set $y_n = [\mathbf{\Phi}_0]_{n,n} x_n^{(0)} + \sum_{k=1}^{K} x_n^{(k)}$



the EV graph filter $\mathbf{H}_{\text{ev}}$ (3.28) is designed in the vertex domain as

$$\underset{\mathbf{\Phi}_j}{\text{minimize}} \left\| \mathbf{H}^* - \sum_{k=1}^{K} \prod_{j=1}^{k} (\mathbf{\Phi}_j \odot \mathbf{S}) + \mathbf{\Phi}_0 \right\|_{\text{F}}. \tag{3.30}$$

Unfortunately, (3.30) is a high-dimensional nonconvex problem that might lead to suboptimal results due to its multiple local minima. To address this issue, in the sequel, we propose a two-step approach for designing the weighting matrices $\{\mathbf{\Phi}_j\}_{j=0}^{K}$.

First, let us consider the following decomposition for approximating a given filter $\mathbf{H}^*$ by a finite matrix series:

$$\mathbf{H}^* \approx \sum_{k=1}^{K} \tilde{\mathbf{\Phi}}_k \odot \mathbf{S}^k + \tilde{\mathbf{\Phi}}_0, \tag{3.31}$$

where matrices $\tilde{\mathbf{\Phi}}_k$ share the support with $\mathbf{S}^k$ and $\tilde{\mathbf{\Phi}}_0$ is diagonal. This decomposition perfectly describes the filter $\mathbf{H}^*$ for sufficiently large $K$ and/or well-connected graphs, as all entries of $\mathbf{S}^K$ are nonzero. The decomposition (3.31) allows us to approximate any filter as a set of matrices, $\{\mathbf{M}_k\}_{k=0}^{K}$, that can be decomposed through an element-wise operation, i.e., $\mathbf{M}_k = \tilde{\mathbf{\Phi}}_k \odot \mathbf{S}^k$. Then, by employing the definition of the Frobenius norm, we find the weighting matrices $\{\tilde{\mathbf{\Phi}}\}_{k=0}^{K}$ from (3.31) as the solution of

$$\underset{\{\tilde{\boldsymbol{\phi}}_k\}}{\text{minimize}} \left\| \tilde{\mathbf{h}} - \sum_{k=0}^{K} \text{diag}(\mathbf{s}_k) \tilde{\boldsymbol{\phi}}_k \right\|_2, \tag{3.32}$$

where $\mathbf{h}^* := \text{vec}(\mathbf{H}^*)$, $\tilde{\boldsymbol{\phi}}_k := \text{vec}(\tilde{\mathbf{\Phi}}_k)$ and $\mathbf{s}_k := \text{vec}(\mathbf{S}^k)$. As problem (3.32) is clearly a convex, methods for its efficient solution are readily available [18]. After the weighting matrices $\{\tilde{\mathbf{\Phi}}_k\}$ are obtained, as the second step, we require to obtain the weighting matrices of the edge-variant filter in (3.28). To do so we propose to fit the matrices $\{\mathbf{\Phi}_k\}_{k=0}^{K}$ by sequentially solving the problem:

$$\underset{\mathbf{\Phi}_k}{\text{minimize}} \left\| \tilde{\mathbf{\Phi}}_k \odot \mathbf{S}^k - (\mathbf{\Phi}_k \odot \mathbf{S}) \prod_{j=1}^{k-1} (\mathbf{\Phi}_j \odot \mathbf{S}) \right\|_F, \tag{3.33}$$

where it is assumed that the matrices $\{\mathbf{\Phi}_j\}_{j<k}$ have been already obtained with $\mathbf{\Phi}_0 = \tilde{\mathbf{\Phi}}_0$ and $\mathbf{\Phi}_1 = \tilde{\mathbf{\Phi}}_1$.

**Algorithm 3.4.** Distributed computation of the constrained edge-variant FIR output

1: Initialize the filter coefficients $\mathbf{\Phi}_0, \ldots, \mathbf{\Phi}_K$, set $\mathbf{x}^{(0)} = \mathbf{x}$ and $\mathbf{y}^{(0)} = \mathbf{\Phi}_0\mathbf{x}$
2: **procedure** COMPUTE LOCALLY THE FILTER OUTPUT
3: **for** k = 1, ..., K **do**
4: Collect $x_m^{(k-1)}$ from all neighbors $m \in \mathcal{N}_n$
5: Compute $x_n^{(k)} = \sum_{m \in \mathcal{N}_n} W_{n,m}\left(x_n^{(k-1)} - x_m^{(k-1)}\right)$
6: Store locally $y_n^{(k)} = \sum_{m \in \mathcal{N}_n} [\mathbf{\Phi}_k]_{n,m} W_{n,m}\left(x_n^{(k-1)} - x_m^{(k-1)}\right)$
7: Send $x_n^{(k)}$ to all neighbors $\mathcal{N}_n$
8: Set $y_n = \sum_{k=0}^{K} y_n^{(k)}$

Even though the proposed EV graph filter (3.28) is the most general version of a linear graph filter, the two-step design of the weighting matrices might result in an involved and numerically unstable task. Considering these facts, in the next section, we develop a constrained version of the EV FIR that provides a simpler design for the weighting matrices and still preserves the efficient implementation with complexity $O(MK)$.

### 3.4.2. CONSTRAINED EDGE-VARIANT FIR FILTERING

Instead of considering the filter structure in (3.28), consider the following description for a ready-to-distribute constrained edge-variant (C-EV) FIR

$$\mathbf{H}_{\text{c-ev}} \triangleq \sum_{k=1}^{K} (\mathbf{\Phi}_k \odot \mathbf{S})\mathbf{S}^{k-1} + \mathbf{\Phi}_0. \tag{3.34}$$

Similarly to (3.28), the matrix $\mathbf{\Phi}_k \in \mathbb{R}^{N \times N}$ is an edge-weighting matrix whose support is equal to the one of the shift operator matrix, $\mathbf{S}$ and $\mathbf{\Phi}_0$ is still diagonal.

From (3.34) it can be seen that the C-EV FIR is distributed in nature since at each step the previous intermediate result, $\mathbf{x}^{(k)} = \mathbf{S}^{k-1}\mathbf{x}^{(k-1)}$, is locally weighted and combined at each node, i.e., $\mathbf{y}^{(k)} = (\mathbf{\Phi}_k \odot \mathbf{S})\mathbf{x}^{(k)}$, similarly to the node-invariant and NV graph filters. Algorithm 3.4 further illustrates the distributed implementation of the C-EV FIR graph filter for $\mathbf{S} = \mathbf{L}$.

Following the same strategy as in the previous section, we can design the weighting matrices $\{\mathbf{\Phi}_k\}_{k=0}^{K}$ through a least squares approach when the criterion (3.30) is employed. That is, the optimal weighting matrices are the solution to the optimization problem

$$\underset{\{\boldsymbol{\phi}_k\}}{\text{minimize}} \quad \left\| \mathbf{h}^* - \sum_{k=1}^{K} (\mathbf{S}^{k-1} \otimes \mathbf{I}_N)\text{diag}(\mathbf{s})\boldsymbol{\phi}_k + \boldsymbol{\phi}_0 \right\|_2, \tag{3.35}$$

where $\mathbf{s} = \text{diag}(\mathbf{S})$ and $\boldsymbol{\phi}_k = \text{vec}(\mathbf{\Phi}_k)$. The vector $\mathbf{h}^* = \text{vec}(\mathbf{H}^*)$ is the vectorized desired filter response matrix. As the support of the weighting matrices is known, i.e., the locations of the nonzero entries are given by the nonzero entries of the shift operator, the

filter (in its vectorized form) can be expressed as

$$\mathbf{h}_{\text{c-ev}} = \sum_{k=1}^{K} \bar{\mathbf{S}}_k \bar{\boldsymbol{\phi}}_k + \tilde{\boldsymbol{\phi}}_0, \tag{3.36}$$

where $\mathbf{h}_{\text{c-ev}} = \text{vec}(\mathbf{H}_{\text{c-ev}})$, $\bar{\boldsymbol{\phi}}_k$ is the vector with the nonzero entries of $\boldsymbol{\phi}_k$, and $\bar{\mathbf{S}}_k$ is the matrix $(\mathbf{S}^{k-1} \otimes \mathbf{I}_N)\text{diag}(\mathbf{s})$ with the columns related to the zero entries of $\boldsymbol{\phi}_k$ removed. That is, as the support for each $\boldsymbol{\Phi}_k$ is known and only its nonzero entries must be estimated.



Therefore, the optimal weights for the edges are given by the solution to the linear system

$$\begin{aligned} \mathbf{h} &= \begin{bmatrix} \mathbf{I}_N & \bar{\mathbf{S}}_1 & \dots & \bar{\mathbf{S}}_K \end{bmatrix} \begin{bmatrix} \bar{\boldsymbol{\phi}}_0 \\ \bar{\boldsymbol{\phi}}_1 \\ \vdots \\ \bar{\boldsymbol{\phi}}_K \end{bmatrix} \\ &= \bar{\mathbf{S}}_{0:K} \bar{\boldsymbol{\phi}}_{0:K}, \end{aligned} \tag{3.37}$$

where $\bar{\mathbf{S}}_{0:K} \in \mathbb{R}^{N^2 \times \text{nnz}(\mathbf{S})\cdot K+N}$ and $\bar{\boldsymbol{\phi}}_{0:K} \in \mathbb{R}^{\text{nnz}(\mathbf{S})\cdot K+N}$. Here, $\text{nnz}(\cdot)$ is the number of nonzero entries of a matrix. The linear system (3.37) has a unique solution as long the condition $\text{rank}(\bar{\mathbf{S}}_{0:K}) = \text{nnz}(\mathbf{S}) \cdot K + N$ holds, otherwise regularization should be used in order to achieve a unique solution for (3.37). We conclude this section with the following remark:

**Remark 3.2.** *It is worth noticing that the NV graph filter [cf. Section 3.2.2]:*

$$\mathbf{H}_{\text{nv}} = \sum_{k=1}^{K} diag(\boldsymbol{\phi}_k)\mathbf{S}^k + diag(\boldsymbol{\phi}_0), \tag{3.38}$$

*is a particular case of the C-EV graph filter, in which each row of the matrices $\{\boldsymbol{\Phi}_k\}_{k=1}^K$ have all their non-zero elements equal, i.e.,*

$$\mathbf{H}_{\text{nv}} = \sum_{k=1}^{K} \left((\boldsymbol{\phi}_k \mathbf{1}_N^T) \odot \mathbf{S}\right)\mathbf{S}^{k-1} + diag(\boldsymbol{\phi}_0). \tag{3.39}$$

### 3.4.3. NUMERICAL RESULTS

In this section, we test the proposed C-EV graph filter for approximating a user-provided frequency response. For these tests, we consider a random community graph generated with the GSP toolbox [20], with $N = 256$ nodes and shift operator $\mathbf{S} = \mathbf{L}_{\text{n}}$, i.e., the normalized Laplacian. Further, the maximum order of the FIR graph filters that is used as the baseline is $K_{\max} = 25$. For analysis, we consider two frequency responses commonly used in the graph community: $(i)$ an exponential kernel, i.e.,

$$h(\lambda) := e^{-\gamma(\lambda-\mu)^2},$$

and $(ii)$ an ideal low pass filter, i.e.,

$$h(\lambda) = \begin{cases} 1 & 0 \le \lambda \le \lambda_{\text{c}} \\ 0 & \text{otherwise,} \end{cases}$$

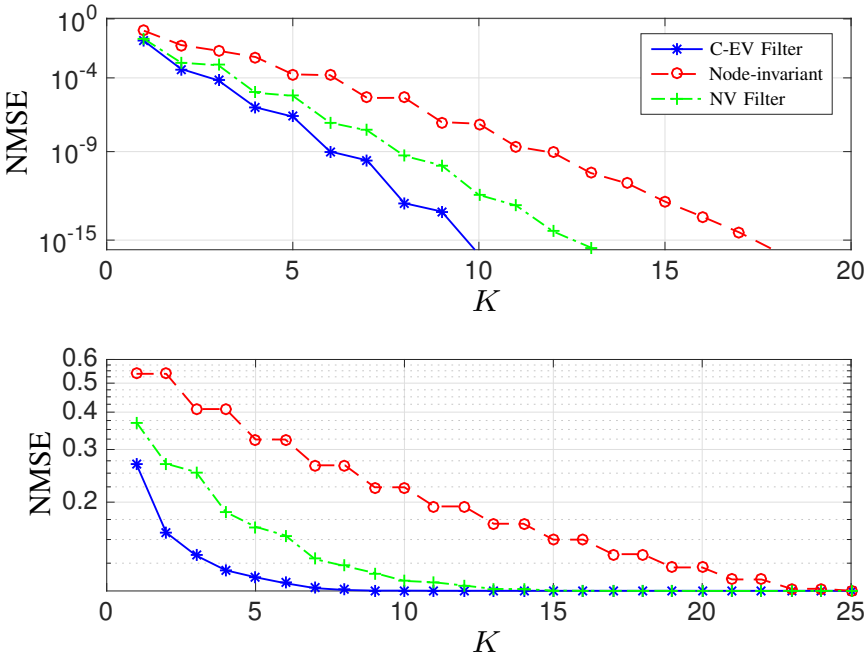



Figure 3.2: Error comparison between the proposed EV graph filter, the NV graph filter and the node-invariant FIR for different orders. (Top) Performance in the ideal low pass scenario. (Bottom) Performance in the exponential kernel.

with $\gamma$ and $\mu$ being the spectrum decaying factor and the central parameter for the exponential kernel, respectively, and $\lambda_c$ the cut-off frequency in the ideal low pass filter[2]. The matrix filter responses used to solve (3.37) are the desired filter response, i.e., $(\vec{\mathbf{H}}^*) = (\vec{\mathbf{U}}h(\mathbf{\Lambda})\mathbf{U}^{\mathsf{H}})$.

Figure 3.2 shows a comparison in terms of the normalized mean squared error (NMSE) between the approximated and the desired frequency responses for both scenarios. In the low pass scenario, we remark that the C-EV graph filter achieves the error floor for $K = 8$, while the NV graph filter for $K = 14$ and the node-invariant FIR for $K = 25$. In the exponential kernel scenario, we observe that the gap for a fixed NMSE between the constrained EV and NV graph filters is only two orders while the node-invariant FIR requires much higher orders to meet the imposed NMSE target.

In Figure 3.3 we depict the approximation accuracy of the C-EV graph filter ($K = 8$) with that of the node-invariant FIRs of different orders for the low pass filter case. We note that the approximation quality for the C-EV graph filter is similar to that of the

[2]The exponential kernel frequency response is preferred as it resembles the Tikhonov denoising problem on graphs (see also Section 2.3.6 in Chapter 2). The low-pass step function, on the other hand, is used in compressive clustering [6] and in graph filter banks [14].

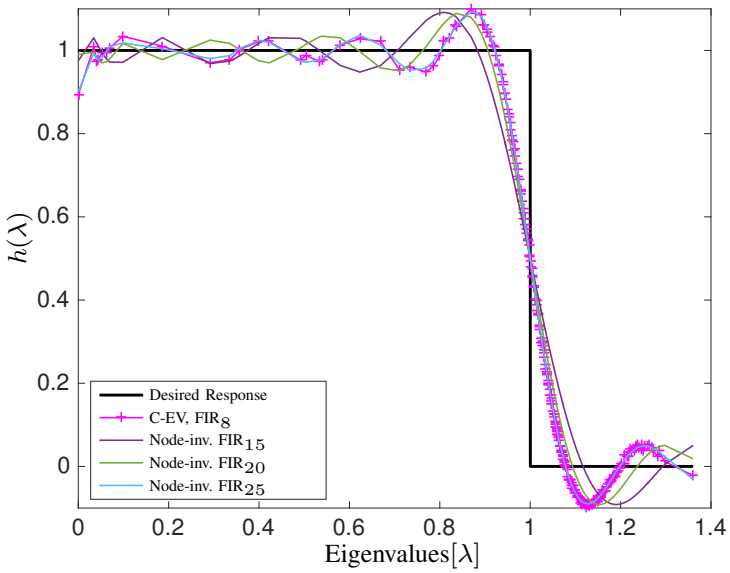



Figure 3.3: Comparison of the frequency response for a FIR filter of increasing order with the response of the C-EV filter of order $K = 8$ when approximating a perfect low pass filter.

node-invariant FIR with $K = 25$.[3] This order reduction implies that fewer communication rounds in the graph are required to implement such low pass filter. Similarly, the comparison between the response of the different filters approximating an exponential kernel filter is shown in Figure 3.4. In this case, the exponential kernel, with parameters $\{\mu = 0.75, \gamma = 3\}$, is well-approximated by the C-EV graph filter for $K = 3$. This approximation produces a similar result with a node-invariant FIR of order $K = 5$, providing fewer communication rounds. Notice that the C-EV filter of order $K = 2$ outperforms all node-invariant FIRs with $K \leq 4$.

It is worth noticing that the improved accuracy of the C-EV graph filter is due to its larger DoFs, i.e., $\text{DoFs}_{\text{c-ev}} = \text{nnz}(\S) \cdot K + N > N \cdot (K+1) = \text{DoFs}_{\text{nv}}$ leading to a saving in terms of communications and computational complexity compared to a distributed implementation of FIR graph filters. However, due to this design freedom, we would like to remark that the filter design for the C-EV graph filters might suffer from numerical issues for large filter orders due to the conditioning of the matrix $\bar{\S}_{0:K}$, i.e., the number of parameters to be estimated $(\text{nnz}(\mathbf{S}) \cdot K + N)$ become larger than the effective rank of $\bar{\S}_{0:K}$. Similar problems might arise for NV graph filters of high orders, although the number of parameters to estimate are smaller compared to the C-EV graph filter.

[3] We also remark that the approximation accuracy of the node-invariant FIR does not improve for $K > 25$.

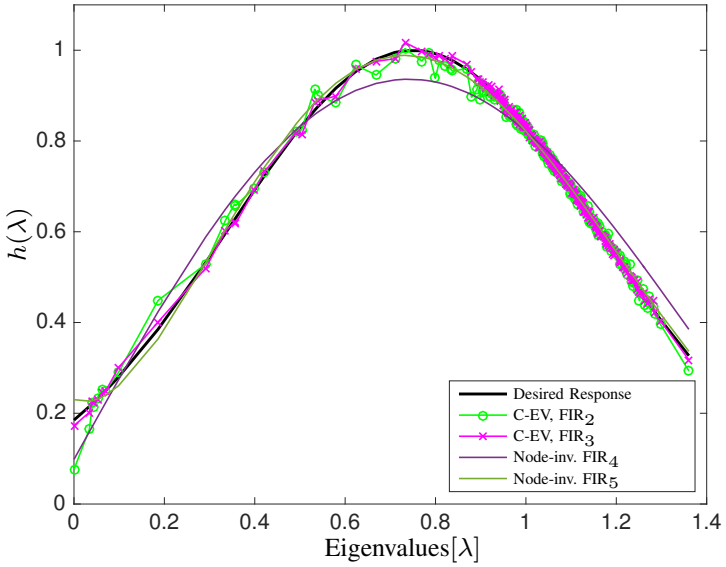



Figure 3.4: Comparison of the frequency response for a FIR filter of increasing order with the response of the C-EV filter of order $K = 8$ when approximating an exponential kernel with parameters $\mu = 0.75$ and $\gamma = 3$.

## 3.5. CONCLUDING REMARKS

This chapter introduced the first type of graph filters, the finite impulse response graph filter. FIR graph filters, the direct homonym of their temporal counterpart are characterized by a finite impulse response now in the vertex domain. We first introduced two state-of-the-art architectures to implement distributed FIR graph filtering, namely the node-invariant and node-variant FIR graph filters. These architectures enjoy an efficient distributed implementation in the vertex domain, allowing for a number of local communications that are related to the number of graph edges and the filter order.

We then introduced the filter design task such that a given frequency response is approximated. We illustrated two approaches to do so. The first approach designs the filter coefficients by exploiting the eigendecomposition of the graph shift operator matrix, allowing the filter coefficients to accurately match the desired response on the specific graph frequencies. The second approach avoids the eigendecomposition costs and designs the filter coefficients in a continuous range of graph frequencies, where the eigenvalues of the graph shift operator are supposed to lie. As a consequence, by aiming to minimize the approximation error over all the frequency interval, the approximation accuracy w.r.t. the first approach is potentially lower.

Finally, we introduced a novel architecture to implement FIR graph filters, namely edge-variant graph filter. These EV graph filters generalize the former two architectures and reduce the communication and computational complexity of the distributed imple-

mentation. Similarly, we proposed design strategies for the EV filters and show that a better approximation accuracy is achieved when approximating an exponential kernel-like and a low-pass step function graph frequency response.

## FURTHER READING



[1] M. Coutino, E. Isufi, and G. Leus, *Distributed Edge-Variant Graph Filters,* in *International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)* (IEEE, 2017).

[2] D. I. Shuman, P. Vandergheynst, and P. Frossard, *Distributed signal processing via chebyshev polynomial approximation,* arXiv preprint arXiv:1111.5239 (2011).

[3] A. Sandryhaila and J. M. Moura, *Discrete signal processing on graphs,* IEEE transactions on signal processing **61**, 1644 (2013).

[4] S. Segarra, A. Marques, and A. Ribeiro, *Optimal graph-filter design and applications to distributed linear network operators,* IEEE Transactions on Signal Processing (2017).

[5] M. Belkin and P. Niyogi, *Laplacian eigenmaps for dimensionality reduction and data representation,* Neural computation **15**, 1373 (2003).

[6] N. Tremblay, G. Puy, R. Gribonval, and P. Vandergheynst, *Compressive spectral clustering,* in *International Conference on Machine Learning* (2016) pp. 1002–1011.

[7] R. Ahlswede, N. Cai, S.-Y. Li, and R. W. Yeung, *Network information flow,* IEEE Transactions on information theory **46**, 1204 (2000).

[8] E. Isufi, H. Dol, and G. Leus, *Advanced flooding-based routing protocols for underwater sensor networks,* EURASIP Journal on Advances in Signal Processing **2016**, 52 (2016).

[9] S. Achard, R. Salvador, B. Whitcher, J. Suckling, and E. Bullmore, *A resilient, low-frequency, small-world human brain functional network with highly connected association cortical hubs,* Journal of Neuroscience **26**, 63 (2006).

[10] W. Huang, L. Goldsberry, N. F. Wymbs, S. T. Grafton, D. S. Bassett, and A. Ribeiro, *Graph frequency analysis of brain signals,* IEEE Journal of Selected Topics in Signal Processing **10**, 1189 (2016).

[11] J. D. Medaglia, W. Huang, E. A. Karuza, A. Kelkar, S. L. Thompson-Schill, A. Ribeiro, and D. S. Bassett, *Functional alignment with anatomical networks is associated with cognitive flexibility,* Nature Human Behaviour **2**, 156 (2018).

[12] M. Defferrard, X. Bresson, and P. Vandergheynst, *Convolutional neural networks on graphs with fast localized spectral filtering,* in *Advances in Neural Information Processing Systems* (2016) pp. 3844–3852.

[13] J. Ma, W. Huang, S. Segarra, and A. Ribeiro, *Diffusion filtering of graph signals and its use in recommendation systems,* in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on* (IEEE, 2016) pp. 4563–4567.

[14] S. K. Narang and A. Ortega, *Perfect reconstruction two-channel wavelet filter banks for graph structured data,* IEEE Transactions on Signal Processing **60**, 2786 (2012).

[15] D. K. Hammond, P. Vandergheynst, and R. Gribonval, *Wavelets on graphs via spectral graph theory,* Applied and Computational Harmonic Analysis **30**, 129 (2011).

[16] F. R. Gantmakher, *The theory of matrices*, Vol. 131 (American Mathematical Soc., 1998).

[17] J. C. Mason and D. C. Handscomb, *Chebyshev polynomials* (CRC Press, 2002).

[18] S. Boyd and L. Vandenberghe, *Convex optimization* (Cambridge university press, 2004).

[19] A. Sandryhaila, S. Kar, and J. M. Moura, *Finite-time distributed consensus through graph filters,* in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on* (IEEE, 2014) pp. 1080–1084.

[20] N. Perraudin, J. Paratte, D. Shuman, L. Martin, V. Kalofolias, P. Vandergheynst, and D. K. Hammond, *Gspbox: A toolbox for signal processing on graphs,* arXiv preprint arXiv:1408.5781 (2014).

# 4

# INFINITE IMPULSE RESPONSE GRAPH FILTERING

*Time flies over us, but leaves its shadow behind.*

Nathaniel Hawthorne

In this chapter, we discuss the class of distributed graph filters that attain a rational graph frequency response. These filters recast in the family of *infinite impulse response* (IIR) graph filters. Being characterized by a rational frequency response, IIR graph filters have the potential to achieve higher approximation accuracies regarding their FIR counterparts. They will provide exact closed-form solutions for tasks such as Tikhonov- (2.24) and Wiener- (2.26) based denoising and graph signal interpolation under smoothness assumption. On the downside, these filters will achieve their graph frequency response after infinite iterations in the vertex domain, yet characterized by a linear convergence rate.

IIR filtering over networks has been approached in recent years from different perspectives. The authors in [3, 4] introduced an infinite recursion, named the *potential kernel*, to perform inverse filtering for identifying information peaks within a sensor network. In [5], inverse problems over a graph were solved with the steepest descent approach. In [6] and later extended to the co-authored works [1, 2] we formalized the inverse filtering on graphs with autoregressive moving average (ARMA) graph filters. In this chapter, we show how this formalization highlights the role played by the graph shift operator and the desired filter response in the design phase, and how it links to the FIR graph filters.

We begin the chapter with introducing IIR filtering in the graph Fourier domain in Section 4.1. The ARMA recursions to implement these filters in the vertex domain are described in Sections 4.2 and 4.3. These sections contain also the filters' design strategies

Parts of this chapter have been published in the IEEE Transactions on Signal Processing [1] (2017) and in the IEEE ICASSP [2] (2017).

and the numerical evaluation. In Section 4.4 we conclude the chapter with a summary of the main results.

## 4.1. INTRODUCTION

The primary aim of this chapter is to introduce recursions that implement a graph filter with a frequency response

$$h(\lambda) = \frac{h_n(\lambda)}{h_d(\lambda)} = \frac{\sum_{q=0}^{Q} b_q \lambda^q}{1 + \sum_{p=1}^{P} a_p \lambda^p}, \tag{4.1}$$



where $h(\lambda)$ consists of the ratio of two polynomial $h_{\mathrm{n}}(\lambda) = \sum_{q=0}^{Q} b_q \lambda^q$ and $h_{\mathrm{d}}(\lambda) = 1 + \sum_{p=1}^{P} a_p \lambda^p$. The positive scalars $P$ and $Q$ denote the orders of the denominator and numerators, while $a_p$ and $b_q$ are complex coefficients. This improvement of degrees of freedom allows us to compute a larger family of responses with respect to the FIR counterpart [cf. (3.2)]. We refer to form (4.1) as an $\mathrm{ARMA}_{P,Q}$ graph filter, while for $P = Q = K$ we use the shorthand notation $\mathrm{ARMA}_K$.

These ARMA graph filters will be our tool to provide an answer to the research question (Q1.2). As in the FIR case, we are interested in recursions that perform (4.1) distributively. However, the ARMA filters introduce a crucial challenge in the design phase; the filter stability. In fact, the denominator coefficients $a_p$ must satisfy $1 + \sum_{p=1}^{P} a_p \lambda^p \neq 0$ to have a stable filter. As we will show from the next section, the IIR structures that implement (4.1) in the vertex domain will require infinite iterations between nodes to attain the frequency response. Therefore, the coefficients $a_p$ and $b_q$ should be designed to ensure a fast and convergent[1] recursion.

In the sequel, we provide the chapter contributions, while the potential applications of the ARMA graph filters are the same as those of the FIR [cf. Section 3.1.2].

### 4.1.1. CONTRIBUTIONS

In the context of IIR graph filters, this chapter provides the following contributions.

**Contribution 4.1.** *We formalize the concept of inverse filtering on graphs through ARMA recursions.* These filters pledge themselves as direct contenders of the FIR graph filters for processing graph signals, now, with a rational frequency response.

**Contribution 4.2.** *We propose three types of ARMA recursions, namely the parallel, the periodic and the feedback-looped implementation, to attain a rational graph frequency response.* All methods can be implemented distributedly, attain fast convergence, and have communication and memory requirements linear in the number of graph edges and the approximation order.

[1] We remark that the concepts of *stability* and *convergence* are different in this thesis. With stable filters, we require the denominator of (4.1) to be non zero. Meanwhile, for a recursion to be convergent we intend an algorithm capable to achieve at steady-state the designed frequency response (4.1) and not diverging to a different value. The latter is similar to the convergence analysis performed for gradient descent algorithms [7, 8]. In this context, an IIR graph filter can be stable but not convergent.

**Contribution 4.3.** *We propose a filter design strategy to account for distributable graph filtering.* We use a variant of the Shanks' method, to approximate any desired graph frequency response. We provide exact design strategies for inverse problems graphs, such as the Tikhonov- and Wiener-based graph signal denoising and graph signal interpolation under smoothness assumptions.

## 4.2. ARMA GRAPH FILTERS

In complete analogy with the temporal IIR filters, an ARMA graph filter requires the nodes to exchange information of both the filter's input and output signals. The $\text{ARMA}_{P,Q}$ output $y_i$ at vertex $v_i$ can be written as

$$y_i = \sum_{j \in \mathcal{N}(i,P)} a_{i,j} y_j + \sum_{j \in \mathcal{N}(i,Q)} b_{i,j} x_j + b_{i,i} x_i, \tag{4.2}$$



where $a_{i,j}$ and $b_{i,j}$ are complex coefficients. To compute $y_i$, vertex $v_i$ requires indirect access to its $P$-hops neighbors' output signal $y_j$ for $j \in \mathcal{N}(i,P)$ and to its $Q$-hops neighbors' input signal $x_j$ for $j \in \mathcal{N}(i,Q)$, and direct access to its input signal $x_i$. In Chapter 3, we saw that FIR graph filters allowed us to have indirect access to the input signal of the neighbors. Therefore, the practical challenge of (4.2) is in accessing the neighboring output signal.

To render (4.2) practical, nodes can compute their outputs iteratively over time as

$$[\mathbf{y}_t]_i = \sum_{j \in \mathcal{N}(i,P)} a_{i,j} [\mathbf{y}_{t-1}]_j + \sum_{j \in \mathcal{N}(i,Q)} b_{i,j} x_j + b_{i,i} x_i, \tag{4.3}$$

where $[\mathbf{y}_t]_i$ is the $i$th element of the filter output at iteration $t$, $\mathbf{y}_t$. Differently from (4.2), with the iterative implementation (4.3) node $v_i$ computes its output at iteration $t$, $[\mathbf{y}_t]_i$, as a linear combination of its $P$-hop neighbors' previous output $\mathbf{y}_{t-1}$ and that of the $Q$-hop neighbors' input signal $\mathbf{x}$. In this way, the network can perform the ARMA filtering for $t \to \infty$, yielding the notion of IIR graph filtering in the vertex domain.

In the sequel, we analyze different recursions that follow (4.3) to implement ARMA graph filters with graph frequency response (4.1).

### 4.2.1. $\text{ARMA}_1$ GRAPH FILTER

Before describing the full-fledged ARMA graph filters, it helps first to consider a first order graph filter. These $\text{ARMA}_1$ recursions, as we show next, are the basic building block for creating filters with a rational frequency response of any order. We get an $\text{ARMA}_1$ graph filter as an extension of the potential kernel [3]. Consider the following first order recursion

$$\mathbf{y}_t = \psi \mathbf{S} \mathbf{y}_{t-1} + \varphi \mathbf{x} \tag{4.4a}$$

$$\mathbf{z}_t = \mathbf{y}_t + c\mathbf{x} \tag{4.4b}$$

where $\mathbf{y}_0$ is arbitrary and $\psi$, $\varphi$ and $c$ are complex coefficients. In (4.4), $\mathbf{y}_t$ is an iterative auxiliary variable that takes into account the filter state, while $\mathbf{z}_t$ is the filter output. For this recursion, we prove our first result.

**Algorithm 4.1.** Distributed computation of the $\text{ARMA}_1$ output

1: Initialize the filter coefficients $\psi, \varphi, c$ and $\mathbf{y}_0$ arbitrary
2: **procedure** COMPUTE LOCALLY THE FILTER OUTPUT
3: **for** $t = 1, \dots T$ **do**
4: Collect $[\mathbf{y}_{t-1}]_m$ from all neighbors $m \in \mathcal{N}_n$
5: Compute $[\mathbf{y}_t]_n = \psi \sum_{m \in \mathcal{N}_n} W_{n,m}\Big([\mathbf{y}_{t-1}]_n - [\mathbf{y}_{t-1}]_m\Big) + \varphi x_n$
6: Send $[\mathbf{y}_t]_n$ to all neighbors $\mathcal{N}_n$
7: Set $[\mathbf{z}_t]_n = [\mathbf{y}_t]_n + c x_n$

**Theorem 4.1.** *The frequency response of the ARMA$_1$ implementation* (4.4) *is*

$$h(\lambda) = c + \frac{r}{\lambda - p}, \quad \textit{subject to} \quad |p| > \varrho \tag{4.5}$$

*with residual $r = -\varphi/\psi$, pole $p = 1/\psi$, and $\varrho$ being the spectral radius bound of $\mathbf{S}$. Recursion* (4.4) *converges to it linearly, irrespective of the initial condition $\mathbf{y}_0$ and shift operator $\mathbf{S}$.*

(The proof can be found in Appendix 4.A.)

The convergence constraint in (4.5) can also be understood from a dynamical system perspective. Comparing recursion (4.4) to a discrete-time state-space equation, it becomes apparent that, when the convergence constraint $|\psi\varrho| < 1$ holds, recursion (4.4) achieves the frequency response (4.5).

It is useful to observe that, since $|p| > \varrho$, an increment of the convergence region can be attained if we work with shift operators that center the eigenvalues of the Laplacian/adjacency matrix and decrease the spectral radius bound $\varrho$. For instance, the following shift operators can be considered: $\mathbf{S} = \mathbf{L} - \lambda_{\max}(\mathbf{L})/2\mathbf{I}_N$ with $\lambda_{\min}(\mathbf{S}) = -\max(\mathbf{L})/2$ and $\lambda_{\max}(\mathbf{S}) = \max(\mathbf{L})/2$, or $\mathbf{S} = \mathbf{L}_{\text{n}} - \mathbf{I}_N$ with $\lambda_{\min}(\mathbf{S}) = -1$ and $\lambda_{\max}(\mathbf{S}) = 1$. Due to these benefits, it is recommended to use translated versions of the Laplacians in the filter design phase.[2].

Recursion (4.4) leads to a simple distributed implementation of a graph filter with first order rational frequency response: at each iteration $t$, each node $v_i$ updates its value $[\mathbf{y}_t]_i$ based on its local signal $x_i$ and a weighted combination of the values of $[\mathbf{y}_{t-1}]_j$ of its neighbors $v_j$. Since each node must exchange its value with each of its neighbors, the message complexity at each iteration is of order $O(M)$, which is similar to the efficient implementation of the FIR graph filters. Algorithm 4.1 illustrates this distributed implementation for $\mathbf{S} = \mathbf{L}$.

**Connection: FIR and $\text{ARMA}_1$ graph filters.** There is an equivalence between the $\text{ARMA}_1$ graph filter and the FIR graph filters. Indeed, if we expand (4.4a) to all its terms and consider $t \to \infty$ and $\mathbf{y}_0 = \mathbf{0}_N$ we get:

$$\mathbf{z} = \lim_{t \to \infty} \mathbf{z}_t = \varphi \sum_{\tau=0}^{\infty} (\psi \mathbf{S})^{\tau} \mathbf{x} + c\mathbf{x}, \tag{4.6}$$

[2] Note that from the Sylvester matrix theorem, the translated version of the Laplacians share the same eigenvectors as the original ones.

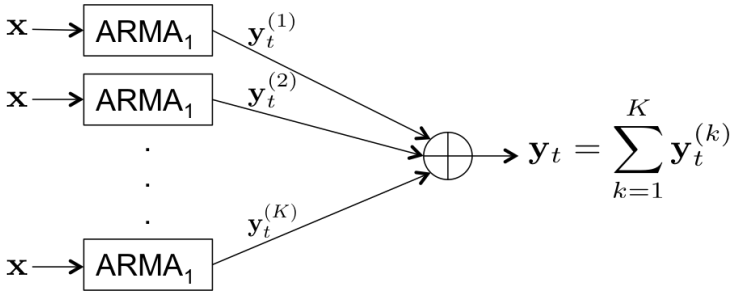



Figure 4.1: Illustration of the parallel ARMA$_K$ graph filter. The filter state at time $t$, $\mathbf{y}_t$, consists of the sum of $K$ the filter states from each ARMA$_1$ branch.



which from Section 3.2.1 is the output of a node-invariant FIR filter of order $K = \infty$ with coefficients $[\varphi + c, \varphi\psi, \varphi\psi^2, \ldots, \varphi\psi^\infty]$. Similar results hold also for $t = T < \infty$.

This equivalence between filters suggests that the same output of an ARMA$_1$ can be obtained with an equivalent FIR filter. That is, the ARMA$_1$ graph filter can be used to design FIR coefficients in approximating rational frequency responses. Later on, in Chapters 6-7, we will see that this ARMA recursion results in a more robust implementation when the graph topology or the graph signal change in time.

### 4.2.2. ARMA$_K$ GRAPH FILTER

Next, we use the ARMA$_1$ block (4.4) to derive distributed graph filters with a more complex frequency response. We present two constructions: The first uses a parallel bank of $K$ ARMA$_1$ filters, attaining linear convergence with a per iteration communication and memory cost of $O(KM)$. The second uses periodic coefficients to reduce the communication costs to $O(M)$ while preserving the linear convergence as the parallel ARMA$_K$ filters.

**Parallel ARMA$_K$ filters.** A larger variety of filter responses can be obtained by adding the states of a parallel ARMA$_1$ filter bank. With reference to Fig. 4.1, let us denote with the superscript $(k)$ the terms corresponding to the $k$th ARMA$_1$ filter for $k = 1, \ldots, K$. Then, the output $\mathbf{z}_t$ of the ARMA$_K$ filter at time instant $t$ is

$$\mathbf{y}_t^{(k)} = \psi^{(k)} \mathbf{S}\mathbf{y}_{t-1}^{(k)} + \varphi^{(k)} \mathbf{x}, \tag{4.7a}$$

$$\mathbf{z}_t = \sum_{k=1}^{K} \mathbf{y}_t^{(k)} + c\mathbf{x} \tag{4.7b}$$

where $\mathbf{y}_0^{(k)}$ is arbitrary. The following theorem characterizes the frequency response of the ARMA$_K$ recursion (4.7).

**Theorem 4.2.** *The frequency response of the parallel ARMA$_K$ is*

$$h(\lambda) = c + \sum_{k=1}^{K} \frac{r_k}{\lambda - p_k} \quad \textit{subject to} \quad |p_k| > \varrho, \tag{4.8}$$

**Algorithm 4.2.** Distributed computation of the parallel $\text{ARMA}_K$ output

1: Initialize the filter coefficients $\psi^{(1)},\dots,\psi^{(K)}$, $\varphi^{(1)},\dots,\varphi^{(K)}$, $c$, and $\mathbf{y}_0^{(1)},\dots,\mathbf{y}_0^{(K)}$ arbitrary
2: **procedure** COMPUTE LOCALLY THE FILTER OUTPUT
3: **for** $t=1,\dots T$ **do**
4: Collect $[\mathbf{y}_{t-1}^{(k)}]_m$ from all neighbors $m\in\mathcal{N}_n$, for $k=1,\dots,K$
5: Compute $[\mathbf{y}_t^{(k)}]_n=\psi^{(k)}\sum_{m\in\mathcal{N}_n}W_{n,m}\left([\mathbf{y}_{t-1}^{(k)}]_n-[\mathbf{y}_{t-1}^{(k)}]_m\right)+\varphi^{(k)}x_n$, for $k=1,\dots,K$
6: Send $[\mathbf{y}_t^{(k)}]_n$, for $k=1,\dots,K$ to all neighbors $\mathcal{N}_n$
7: Set $[\mathbf{z}_t]_n=\sum_{k=1}^K[\mathbf{y}_t^{(k)}]_n+cx_n$

*with residues $r_k=\varphi^{(k)}/\psi^{(k)}$, poles $p_k=1/\psi^{(k)}$, and $\varrho$ being the spectral radius of $\mathbf{S}$. Recursion* (4.7) *converges to it linearly, irrespective of the initial conditions $\mathbf{y}_0^{(k)}$ and graph shift operator $\mathbf{S}$.*

(The proof follows from Theorem 4.1.)

The frequency response of a parallel $\text{ARMA}_K$ is, therefore, a rational function with numerator and denominator polynomials of order $K$ (presented here in a partial fraction form). In addition, since we are running $K$ $\text{ARMA}_1$ filters in parallel, the communication and memory complexities are $K$ times that of the $\text{ARMA}_1$ graph filter. Algorithm 4.2 illustrates the distributed implementation of the $\text{ARMA}_K$ filter for $\mathbf{S}=\mathbf{L}$. Note also that the equivalence between $\text{ARMA}_1$ and FIR filters extends to the parallel $\text{ARMA}_K$ filter.

**Periodic $\text{ARMA}_K$ filters.** We can decrease the memory requirements of the parallel implementation by letting the filter coefficients to vary periodically in time. The periodic filter take the form

$$\mathbf{y}_t=(\theta_t\mathbf{I}_N+\psi_t\mathbf{S})\mathbf{y}_{t-1}+\varphi_t\mathbf{x} \tag{4.9a}$$

$$\mathbf{z}_t=\mathbf{y}_t+c\mathbf{x} \tag{4.9b}$$

where $\mathbf{y}_0$ is arbitrary, the output $\mathbf{z}_{t+1}$ is valid every $K$ iterations, and the coefficients $\theta_t$, $\psi_t$, $\varphi_t$ are periodic with period $K$: $\theta_t=\theta_{t-iK}$, $\psi_t=\psi_{t-iK}$, $\varphi_t=\varphi_{t-iK}$, with $i$ an integer in $[0,t/K]$. The frequency behavior of the periodic $\text{ARMA}_K$ filter is characterized by the following theorem.

**Theorem 4.3.** *The graph frequency response of a periodic $\text{ARMA}_K$ graph filter is*

$$h(\lambda)=c+\frac{\sum_{k=0}^{K-1}\left(\prod_{\tau=k+1}^{K-1}(\theta_\tau+\psi_\tau\lambda)\right)\varphi_k}{1-\prod_{k=0}^{K-1}(\theta_k+\psi_k\lambda)}, \tag{4.10}$$

*subject to the convergence constraints*

$$\left|\prod_{k=0}^{K-1}(\theta_k+\psi_k\varrho)\right|<1 \tag{4.11}$$

*with $\varrho$ being the spectral radius bound of $\mathbf{S}$. Recursion* (4.9) *converges to it linearly, irrespective of the initial condition $\mathbf{y}_0$ and graph shift operator $\mathbf{S}$.*

**Algorithm 4.3.** Distributed computation of the periodic ARMA$_K$ output

1: Initialize the filter coefficients $\theta_0,\dots,\theta_{K-1}$, $\psi_0,\dots,\psi_{K-1}$, $\varphi_0,\dots,\varphi_{K-1}$, $c$, and $\mathbf{y}_0$ arbitrary
2: **procedure** COMPUTE LOCALLY THE FILTER OUTPUT
3: **for** $t = 1,\dots T$ **do**
4: Set $\theta_t = \theta_{t-iK}$, $\psi_t = \psi_{t-iK}$, $\varphi_t = \varphi_{t-iK}$ with integer $i \in [0, t/K]$
5: Collect $[\mathbf{y}_{t-1}]_m$ from all neighbors $m \in \mathcal{N}_n$
6: Compute $[\mathbf{y}_t]_n = \theta_t[\mathbf{y}_{t-1}]_n + \psi_t \sum_{m\in\mathcal{N}_n} W_{n,m}\Big([\mathbf{y}_{t-1}]_n - [\mathbf{y}_{t-1}]_m\Big) + \varphi_t x_n$
7: Send $[\mathbf{y}_t]_n$ to all neighbors $\mathcal{N}_n$
8: Set $[\mathbf{z}_t]_n = [\mathbf{y}_t]_n + cx_n$

(The proof can be found in Appendix 4.B.)

With some algebraic manipulation, it can be shown that the frequency response of periodic ARMA$_K$ is also a rational function of order $K$. We can also observe that the convergence criterion (4.11) is more involved than that of the parallel implementation. As now we are dealing with $K$ ARMA$_1$ graph filters interleaved in time, to guarantee their joint convergence one does not necessarily have to examine them independently (requiring for instance that, for each $k$, $|\theta_k + \psi_k\varrho| < 1$). Instead, it is sufficient that the product $|\prod_{k=0}^{K-1}(\theta_k + \psi_k\varrho)|$ is smaller than one. To illustrate this, notice that if $\theta_k = 0$, the periodic ARMA$_K$ can be stable even if some of the ARMA$_1$ graph filters it is composed of are unstable.

In the distributed computation of the periodic ARMA$_K$, in each iteration each node $v_i$ stores and exchanges $\deg(v_i)$ values with its neighbors. This yields a memory complexity of $O(M)$, rather than the $O(KM)$ of the parallel one (after each iteration, the values are overwritten). Since the output of the periodic ARMA$_K$ is only valid after $K$ iterations, the communication complexity is again $O(KM)$. The low memory requirements of the periodic ARMA$_K$ render it suitable for resource-constrained devices. The distributed computation of the periodic ARMA$_K$ graph filters for $\mathbf{S} = \mathbf{L}$ is illustrated in Algorithm 4.3.

### 4.2.3. FILTER DESIGN

In this section we show how to design the ARMA filter coefficients that approximate any response, using a variant of the Shanks' method [9]. This approach gives us convergent recursions, ensuring the same selectivity as the universal node-invariant FIR graph filters.

Given a desired frequency response $h(\lambda) : [\lambda_{\min}, \lambda_{\max}] \to \mathbb{R}$ and a filter order $K$, our aim is to find the complex polynomials $h_n(\lambda)$ and $h_d(\lambda)$ of order $K$ that minimize

$$\int_\lambda \left| \frac{h_n(\lambda)}{h_d(\lambda)} - h^*(\lambda) \right|^2 \mathrm{d}\lambda = \int_\lambda \left| \frac{\sum_{q=0}^{Q} b_q \lambda^q}{1 + \sum_{p=1}^{P} a_p \lambda^p} - h^*(\lambda) \right|^2 \mathrm{d}(\lambda) \tag{4.12}$$

while ensuring that the chosen coefficients result in a convergent system (see constraints in Theorems 4.2 and 4.3). From $h_n(\lambda)/h_d(\lambda)$ one computes the filter coefficients ($\psi^{(k)}, \varphi^{(k)}$, or $\theta_k, \psi_k$) by algebraic manipulation.

**Design method.** Following the Shanks' method, we approximate the filter coefficients as follows:

*Denominator.* Determine $a_k$ for $k = 1, \dots, K$ by finding a $K' > K$ order polynomial approximation $h'(\lambda) = \sum_{k=0}^{K'} \phi_k \lambda^k$ for $h^*(\lambda)$ using polynomial regression, and solving the coefficient-wise system of equations $h_d(\lambda)h'(\lambda) = h_n(\lambda)$.

*Numerator.* Determine $b_k$ for $k = 1, \dots, K$ by solving the least squares problem

$$\underset{b_0,\dots,b_K}{\text{minimize}} \int_\lambda \left| \frac{h_n(\lambda)}{h_d(\lambda)} - h^*(\lambda) \right|^2 \mathrm{d}\lambda \tag{4.13}$$

over a uniform gridding of the interval $[\lambda_{\min}(\mathbf{S}), \lambda_{\max}(\mathbf{s})]$.

Once the numerator $b_k$ and denominator $a_k$ coefficients are found:

(*i*) *Parallel design.* Perform the partial fraction expansion to find the residuals $r_k$ and poles $p_k$. Then, the filter coefficients $\psi^{(k)}$ and $\varphi^{(k)}$ can be found by exploiting their relation with $r_k$ and $p_k$ in Theorem 4.2.

(*ii*) *Periodic design.* Identify $\psi_k$ by computing the roots of the (source) denominator polynomial $1 - \prod_{k=0}^{K-1}(\theta_k + \psi_k \lambda)$ in (4.10) and equation them to the roots of the (target) denominator $1 + \sum_{k=1}^{K} a_k \lambda^k$. It is suggested to set $\theta_1 = 0$ and $\theta_k = 1$ for $k > 0$, which has the effect of putting the two polynomials in similar form. Once the coefficients $\psi_k$ (and $\theta_k$) have been set, we get $\varphi_k$ by evaluating the numerator target and source polynomials.

The method is also suitable for numerator and denominator polynomials of different orders. We advocate however to use equal orders because it yields the highest approximation accuracy for a given communication/memory complexity.

The most crucial step is the approximation of the denominator coefficients. By fitting $h_d(\lambda)$ to $h'(\lambda)$ instead of $h^*(\lambda)$, we are able to compute coefficients $a_k$ independently of $b_k$. Increasing $K' \gg K$ often leads to a (slight) increase in accuracy but at the price of slower convergence and higher sensitivity to numerical errors. Especially for sharp functions, such as the ones of ideal low-pass filtering, a high order polynomial approximation results in very large coefficients, which affect the numerical convergence of the filters and push the poles closer to the unit circle. In the reminder of this thesis, we consider $K' = K + 1$.

**Stability-convergence concerns.** The proposed design method does not come with theoretical convergence and stability guarantees. The latter is is inherited from the adoption of the Shank's method, which in the temporal domain has yielded stable IIR filters. The convergence guarantee, on the other hand, results in a challenging task for the proposed ARMA$_K$ since it is related to the partial form decomposition coefficients (e.g. $\psi^{(K)}$ and $\varphi^{(k)}$ for the parallel ARMA$_K$), while the design method is performed on the full rational function coefficients (e.g., $a_k$ and $b_k$ in (4.12)). However, this impossibility to act on the coefficients limits our handle to improve the filter convergence speed. In the sequel, we propose exact ARMA designs for some of the most common graph signal processing applications. In Section 4.3 we introduce the feedback-looped ARMA recursion which is designed with theoretical convergence guarantees and controlled convergence speed.

### 4.2.4. EXACT GRAPH FILTER DESIGNS

We proceed to present exact (and in certain cases explicit) graph filter constructions for particular graph signal denoising and interpolation problems. The proposed filters are

universal, that is, they are designed without knowledge of the graph structure. Indeed, the filter coefficients are found independently from the eigenvalues of the graph shift operator. This makes the ARMA filters suitable for any graph, and ideal for cases when the graph structure is unknown or when the $O(N^3)$ complexity of the eigenvalue decomposition becomes prohibitive.

**Tikhonov-based denoising.** Let us recall the Tikhonov regularization problem in Section 2.3.6. A generalization of (2.24) consists of

$$\mathbf{x}_0^\star = \underset{\mathbf{x}\in\mathbb{C}^N}{\operatorname{argmin}} \quad \|\mathbf{x}-\mathbf{x}_0\|_2^2 + w\mathbf{x}_0^\mathsf{T}\mathbf{S}^K\mathbf{x}_0, \tag{4.14}$$

with optimal solution

$$\mathbf{x}_0^\star = (\mathbf{I}_N + w\mathbf{S}^K)^{-1}\mathbf{x}. \tag{4.15}$$

In the above equation, the graph filter $\mathbf{H} = (\mathbf{I}_N + w\mathbf{S}^K)^{-1}$ that provides the optimal denoised solution is an ARMA$_K$ with frequency response

$$h(\lambda) = \frac{1}{1+w\lambda^K} = \frac{1}{\prod_{k=1}^K(\lambda - p_k)} \tag{4.16}$$

with $p_k = -e^{\jmath\gamma_k}/\sqrt[k]{w}$ and $\gamma_k = (2k+1)\pi/K$. From the convergence condition of the parallel ARMA$_K$ (4.8), convergent filters require $|p_k| > \varrho$, which for the particular expression of $p_k$ becomes $\sqrt[k]{w}\varrho < 1$.

By implementing (4.15) with an ARMA$_K$ operating on the translated Laplacians, a notable improvement on the filter convergence is gained. For $\mathbf{S} = \mathbf{L} - \lambda_{\max}(\mathbf{L})/2\mathbf{I}_N$ (4.16) is written as

$$h(\lambda(\mathbf{S})) = \frac{1}{1+w\left(\lambda(\mathbf{L}) + \frac{\lambda_{\max}(\mathbf{L})}{2}\right)^K} = \frac{1}{\prod_{k=1}^K(\lambda(\mathbf{L}) - p_k)}, \tag{4.17}$$

where $\lambda(\mathbf{A})$ and $\lambda_{\max}(\mathbf{A})$ stand for the eigenvalues and the maximum eigenvalue of $\mathbf{A}$, $p_k = -\lambda_{\max}(\mathbf{L})/2 + e^{\jmath\gamma_k}/\sqrt[k]{w}$ for $\gamma_k = (2k+1)\pi/K$. Again, from the stability of the ARMA$_K$ $|p_k| > \varrho$, or equivalently $|p_k|^2 > \varrho^2$, we get convergent filters as long as

$$\left(-\frac{\lambda_{\max}(\mathbf{L})}{2} + \frac{\cos(\gamma_k)}{\sqrt[K]{w}}\right)^2 + \frac{\sin^2(\gamma_k)}{\sqrt[K]{w^2}} > \varrho^2, \tag{4.18}$$

or equivalently

$$\left(\frac{\lambda_{\max}(\mathbf{L})}{4} - \varrho^2\right)\sqrt[K]{w}^2 - l\cos(\gamma_k)\sqrt[K]{w} + 1 > 0, \tag{4.19}$$

are satisfied. For the shifted normalized Laplacian, $\varrho = 1$ and $\lambda_{\max}(\mathbf{L}_\mathrm{n}) = 2$, (4.19) simplifies to

$$2\cos(\gamma_k)\sqrt[K]{w} < 1, \tag{4.20}$$

which is always met for the standard choices of $K = 1$ (quadratic regularization) and $K = 2$ (total variation)[3] [cf. (2.9)-(2.10)].

[3] Even though $w$ is a free parameter, for K = 1, 2 the value $\cos(\gamma_k)$ in (4.20) will be either 0 or -1, due to the expression of $\gamma_k$.

For both (4.16) and (4.17), the denominator coefficients $\psi^{(k)}$ of the corresponding parallel ARMA$_K$ filter can be found as $\psi^{(k)} = 1/p_k$. Meanwhile, the numerator coefficients $\varphi^{(k)}$ are found in two steps: $(i)$ express (4.16), (4.17) in the partial form as in (4.8) to find the residuals $r_k$ and $(ii)$ take $\varphi^{(k)} = -r_k\psi^{(k)}$.

**Wiener-based denoising.** Let us now consider the Wiener-based regularization problem in Section 2.4.2. Recall from (2.27) the optimal Wiener denoising problem corresponds to an ARMA$_K$ graph filter of the form

$$h(\lambda_n) = \frac{p_{x_\mathrm{d}}(\lambda_n)}{p_{x_\mathrm{d}}(\lambda_n) + p_w(\lambda_n)}. \tag{4.21}$$

Notice that this filter is still universal, as the ARMA$_K$ coefficients depend on the rational functions $p_{x_d}(\lambda)$ and $p_w^2(\lambda)$, but not on the specific eigenvalues of the graph graph shift operator.

Let us illustrate the above with the following example. Suppose that $\mathbf{x}_d$ is normally distributed with covariance equal to the pseudoinverse of the Laplacian $\mathbf{L}^\dagger$. This is a popular and well-understood model for smooth signals on graphs with strong connections to Gaussian Markov random fields [10]. In addition, let the noise power be $p_w = w$. Substituting this into (2.27), we find

$$\mathbf{x}_d^\star = \sum_{n=1}^{N} \frac{1}{1+w\lambda_n}(\mathbf{x}^\top \mathbf{u}_n)\mathbf{u}_n, \tag{4.22}$$

which is identical to the Tikhonov-based denoising for $K = 1$. In fact, it corresponds to an ARMA$_1$ with $\varphi = 2/(2+wl)$ and $\psi = -2w/(2+w\lambda_{\max}(\mathbf{L}))$, which as previously shown has a convergent implementation. Note that even though the convergence is ensured for the considered case, it does not necessarily hold for every covariance matrix. Indeed, the convergence of the filter must be examined in a problem-specific manner.

**Graph signal interpolation.** Suppose that only $r$ out of the $N$ values of a signal $\mathbf{x}_d$ are known, and let $\mathbf{x}$ be the $N \times 1$ vector which contains the known values and zeros otherwise. Under the assumption of $\mathbf{x}_d$ being smooth w.r.t. $\mathbf{S} = \mathbf{L}_\mathrm{d}$ or $\mathbf{S} = \mathbf{L}_\mathrm{n}$, we can estimate the unknowns by the regularized problem

$$\mathbf{x}_d^\star = \underset{\mathbf{x}_d \in \mathbb{R}^N}{\operatorname{argmin}} \|\mathbf{C}(\mathbf{x}_d - \mathbf{x})\|_2^2 + w\mathbf{x}_d^\top \mathbf{S}^K \mathbf{x}_d, \tag{4.23}$$

where $\mathbf{C}$ is the diagonal matrix with $C_{i,i} = 1$ if $[\mathbf{x}_d]_i$ is known and $C_{i,i} = 0$ otherwise. Such formulations have been used widely, both in the context of graph signal processing [11, 12] and earlier by the semi-supervised learning community [13, 14]. Similar to (4.15), this optimization problem is convex and its global minimum is found as

$$\mathbf{x}_d^\star = \left(\mathbf{C} + w\mathbf{S}^K\right)^{-1}\mathbf{x}. \tag{4.24}$$

Most commonly, $K = 1$, (4.24) becomes

$$\mathbf{x}_d^\star = \left(\mathbf{I}_N - \mathbf{S}'\right)^{-1}\mathbf{x}, \tag{4.25}$$

which is an ARMA$_1$ filter designed for the shift operator matrix $\mathbf{S}' = \mathbf{C} - \mathbf{I}_N + w\mathbf{S}$. For larger values of $K$, the interpolation cannot be computed distributedly using the developed ARMA filters. This is because the corresponding basis matrix $\mathbf{S}' = \mathbf{C} + w\mathbf{S}^K$ cannot be appropriately factorized into a series of local matrices.

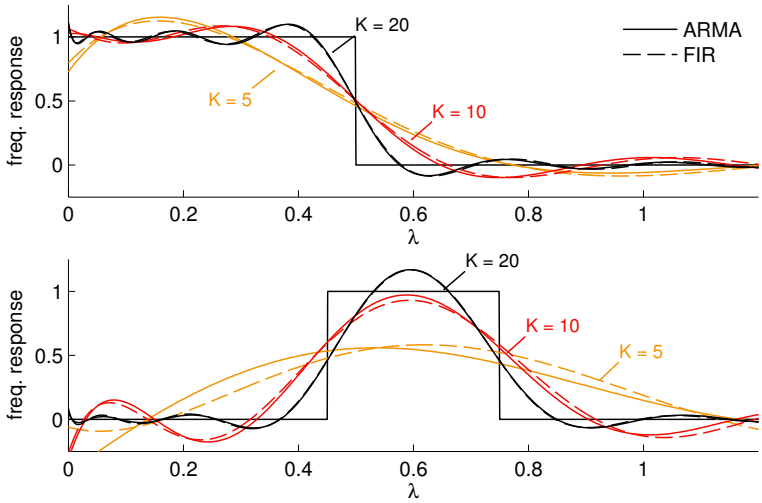



Figure 4.2: The frequency response of ARMA$_K$ filters designed by Shanks' method and the FIR responses of corresponding order. Here, $h^*(\lambda)$ is a step function (top) and a window function (bottom).



### 4.2.5. NUMERICAL RESULTS

In this section we test the approximation accuracy of the ARMA$_K$ graph filters and compare the performance with their direct contenders, i.e., the node-invariant FIR (for short FIR) graph filters[4].

**Approximation.** Figure 4.2 illustrates in solid lines the frequency responses of three ARMA$_K$ filters ($K = 5, 10, 20$), designed to approximate a low-pass step function (top) and a band-pass window function (bottom). In analogy with the connection between ARMA and FIR, we observe that both types of filters achieve a similar approximation accuracy. Table 8.1 in Appendix **??** summarizes some filter coefficients of the parallel implementation used to obtain these results.

**Tikhonov denoising.** Next, we evaluate the filtering performance solving distributively the Tikhonov denoising problem (4.15). We considered a graph $\mathcal{G}$ composed of $N = 100$ nodes placed randomly in a square area. Two nodes are connected if they are closer than 15% of the maximum distance in the area. The average node degree is 11.8. The desired signal $\mathbf{x}_d$ has a graph spectrum $\hat{x}_n = e^{-5\lambda_n}$ and the noise is considered i.i.d. Gaussian zero-mean with unitary variance. The results are compared in terms of root normalized mean squared error (rNMSE) defined as $\|\mathbf{z}_t - \mathbf{x}_d^\star\|_2 / \|\mathbf{x}_d^\star\|_2$

Figure 4.4 shows the filtering error over time in terms of root normalized mean squared error (rNMSE), between the output of the ARMA$_K$ recursion and the solution of the optimization problem (4.15). We remark the fast convergence of the ARMA filters that hit the computer's numerical precision within 10-20 iterations.

[4] A fair comparison of the ARMA graph filters with the NV-FIR and EV-FIR would consist of developing the NV- and the EV-ARMA graph filters. As we address this topic for future research in Section 9.2.1, we do not present here this comparison.

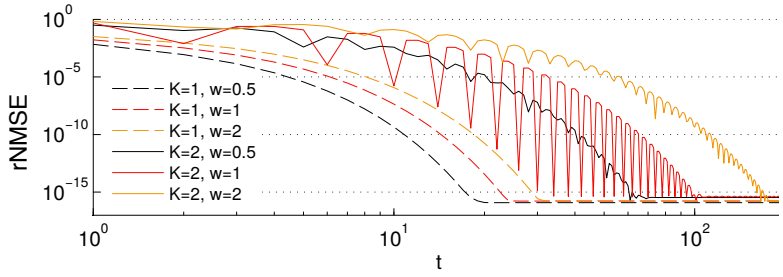



Figure 4.3: Convergence of a denoising parallel ARMA$_K$ filter for $K = 1,2$ and $w = 0.5,1,2$. The residual error is a consequence of the computer's bounded numerical precision.



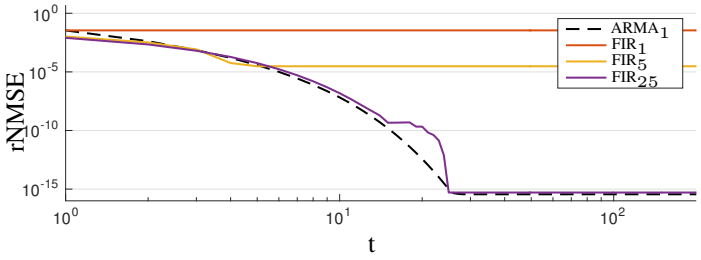



Figure 4.4: Normalized error relative to the Tikhonov denoising problem with different graph filters. For the FIRs, $t$ indicates the iteration of the filter.

Figure 4.4 compares the ARMA$_1$ performance with FIRs of different orders. As expected, we can see that in the static case the ARMA graph after K iterations has the same performance as the FIRK filter and they both match the solution of the optimization problem. In this instance, the FIR coefficients are obtained with a least squares-based design [cf. Section 3.3.2][5]. Later in Chapter 6, we will come back to this scenario and analyze the performance in time-varying topologies.

## 4.3. FEEDBACK-LOOPED ARMA GRAPH FILTERS

While the previous ARMA$_K$ implementations build up on the potential kernel, in this section we extend (4.3) to an ARMA$_{P,Q}$ recursion on graphs. Specifically, as we transformed (3.1) into an FIR$_K$, we observe that the terms $\sum_{j\in\mathcal{N}(i,Q)} b_{i,j}x_j + b_{i,i}x_i$ in (4.2) correspond to an FIR$_Q$ filter operating on $\mathbf{x}$, while the term $\sum_{j\in\mathcal{N}(i,P)} a_{i,j}[\mathbf{y}_{t-1}]_j$ is another FIR filter or order $P$ operating on $\mathbf{y}_{t-1}$. Therefore, an ARMA$_{P,Q}$ graph filter can be

[5] In a subsequent work [15], the authors show this is not always the case, and there are scenarios where the Chebyshev polynomial-based design might perform better. In the follow-up joint work [16, 17], we demonstrate that ARMA filters can in fact compare well also with the Chebyshev polynomial-based design, but in a centralized implementation.

implemented distributively through the feedback recursion

$$\mathbf{y}_t = -\sum_{p=1}^{P} a_p \mathbf{S}^p \mathbf{y}_{t-1} + \sum_{q=0}^{Q} b_q \mathbf{S}^q \mathbf{x}, \tag{4.26}$$

where the first term on the right hand-side of (4.26) consists of the feedback loop of the previous output[6] $\mathbf{y}_{t-1}$. The complex scalars $a_p$ and $b_q$ denote the filter coefficients. Note that the FIR$_P$ filter acting on $\mathbf{y}_{t-1}$ starts from $p = 1$. To ease the notation, let us indicate as $\mathbf{P} = -\sum_{p=1}^{P} a_p \mathbf{S}^p$ and $\mathbf{Q} = \sum_{q=0}^{Q} b_q \mathbf{S}^q$. With this in place, the following proposition shows that recursion (4.26) implements a graph filter with a frequency response of the form (4.1).

**Proposition 4.1.** *The graph frequency response of the ARMA$_{P,Q}$ recursion* (4.26) *is*

$$h(\lambda) = \frac{\sum_{q=0}^{Q} b_q \lambda^q}{1 + \sum_{p=1}^{P} a_p \lambda^p} \quad \textit{subject to} \quad \sum_{p=1}^{P} |a_p| \varrho^p < 1. \tag{4.27}$$

*Recursion* (4.26) *converges to it linearly independently from the initial condition* $\mathbf{y}_0$ *and the choice of the shift operator* $\mathbf{S}$.

(The proof can be found in Appendix 4.C.)

The graph frequency response (4.27) confirms our intuition about (4.3) and shows that recursion (4.26) is a way to implement graph filters with rational frequency responses. Differently from the parallel (4.7) and periodic (4.9) implementations, filters with different orders can now be designed by changing the values of $P$ and $Q$ in (4.26).

### 4.3.1. RECURSION ANALYSIS

From Proposition 4.1, the ARMA$_{P,Q}$ recursion (4.26) will attain the frequency response (4.27) for $t \to \infty$. However, in practice, we are interested to arrest the filter after few iterations. Then, there are two important aspects to address for its distributed implementation: $(i)$ the convergence time, and $(ii)$ the communication and computational costs. While the above were challenging tasks for the parallel and periodic ARMA$_K$, in the sequel, we show this is not the case for the feedback-looped ARMA (4.26). In fact, the feedback-looped ARMA allows a trading of the convergence speed with the approximation accuracy, yet guaranteeing a convergent implementation.

**Convergence analysis.** The following proposition characterizes the linear convergence time of the algorithm and shows our handle to tune it such that a desired error from the steady-state is achieved.

**Proposition 4.2.** *For a stable ARMA$_{P,Q}$ filter of the form* (4.26)*, a sufficient iterations $t$ to be $\epsilon$-close to the frequency response* (4.27) *is*

$$t \geq ln(\epsilon/\alpha) \left(ln\left(\|\mathbf{P}\|\right)\right)^{-1}, \tag{4.28}$$

*for a desired small error $\epsilon$ and $\alpha = \|\mathbf{y}_0\|_2 + (1 - \|\mathbf{P}\|)^{-1} \|\mathbf{Q}\| \|\mathbf{x}\|_2$.*

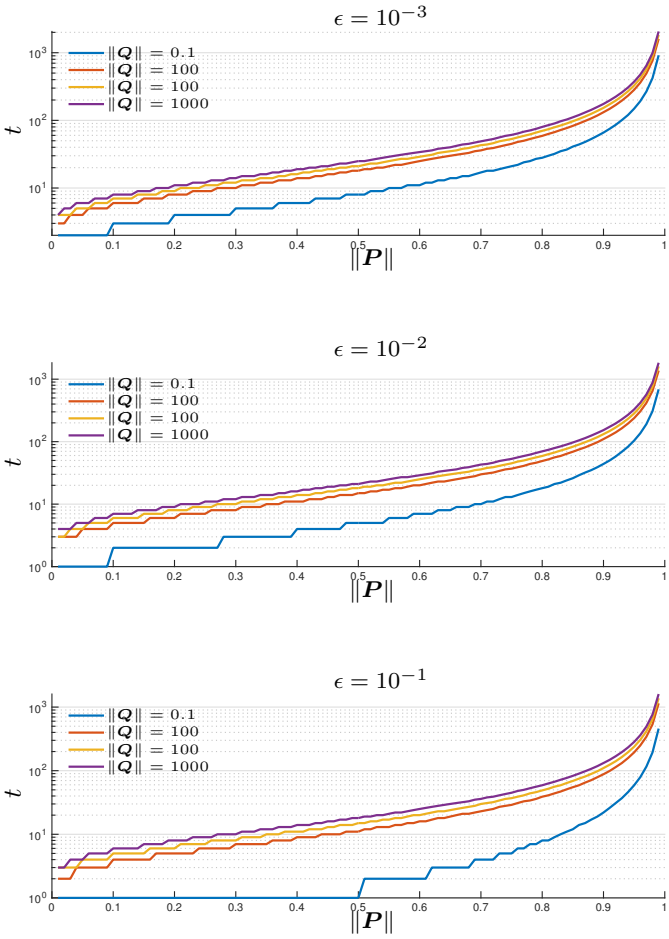



Figure 4.5: Convergence time of the feedback-looped ARMA implementation to be $\epsilon$-close to its steady state as a function of the denominator coefficients. The results are analyzed for different numerator coefficients and different approximation errors $\epsilon$.

(The proof can be found in Appendix 4.D.)

Result (4.28) provides the minimum number of iterations $t$ that algorithm (4.26) must run to be $\epsilon$-close from the steady state. As we can see from (4.28), a handle to reduce $t$ is the spectral norm of $\mathbf{P}$. From $(ln(\|\mathbf{P}\|))^{-1}$ (where $\|\mathbf{P}\| \leq 1$ to ensure convergence) $t$ reduces for a $\mathbf{P}$ with a smaller spectral norm. However, this is in contrast to the effect of $\|\mathbf{P}\|$ in $ln(\epsilon/\alpha)$, where a smaller $\|\mathbf{P}\|$ yields a larger $ln(\epsilon/\alpha)$ and thus potentially a higher $t$. The latter is observed to have a lower influence on $t$. Further, since the $ln(\|\mathbf{P}\|) < 0$,

[6] In relation with the earlier ARMA implementations((4.4), (4.7), and (4.9)) the filter output is the filter state, i.e., $\mathbf{z}_t = \mathbf{y}_t$.

$t$ reduces by choosing the coefficients $\varphi_q$ to reduce the spectral norm of $\mathbf{Q}$ (this is true when $\ln(\epsilon/\alpha) < 0$, otherwise $t = 0$). Therefore, $\|\mathrm{P}[\|]$ has contradictory effects on $t$, while $t$ increases with $\|\mathbf{Q}\|$.

To quantify what said above, consider a scenario with $\epsilon = [10^{-3}, 10^{-2}, 10^{-1}]$, $\|\mathbf{y}_0\|_2 = \mathbf{0}_N$ and $\|\mathbf{x}\|_2 = 1$. Figure 4.5 shows the convergence time (rounded to the next integer) of the ARMA$_{P,Q}$ when (4.28) is met with equality, as a function of $\|\mathbf{P}\| \in [0,1[$ and for different values of $\|\mathbf{Q}\|$. First, notice that the impact of $\|\mathbf{P}\|$ on $(ln(\|\mathbf{P}\|))^{-1}$ prevails on that of $ln(\epsilon/\alpha)$. Indeed, $t$ increases monotonically with $\|\mathbf{P}\|$. Second, a bigger value $\|\mathbf{Q}\|$ yields a larger convergence time. From these results, we suggest avoiding values of $\|\mathbf{P}\| \approx 1$ to obtain reasonable convergence times. This, on the other hand, reduces the convergence region and, thus, we have fewer DoFs to design the coefficients. As we will see later, this might affect the approximation accuracy. Similar considerations hold also for $\|\mathbf{Q}\|$, but since it has less effect on the convergence time we can allow a higher $\|\mathbf{Q}\|$ to improve the approximation accuracy. In summary, the filter coefficients are designed as a trade-off between approximation accuracy and convergence time, with the denominator coefficients being the most sensitive.



### 4.3.2. FILTER DESIGN

The coefficient design of the ARMA$_{P,Q}$ recursion (4.26) has the main advantage that the convergence constraint in (4.27) is related to the coefficients of the full rational frequency response (4.1) rather than to its partial fraction decomposition.

Thus, given the filter orders $(P,Q)$ and a desired frequency response $h^*(\lambda) : [\lambda_{\min}, \lambda_{\max}] \to \mathbb{R}$, we would like to find the coefficients $a_p$ and $b_q$ that make the frequency response (4.27) as close as possible to $h^*(\lambda)$, yet ensuring a convergent implementation. Differently, we would like to minimize the error

$$e'(\lambda) = h^*(\lambda) - \frac{\sum_{q=0}^{Q} b_q \lambda^q}{1 + \sum_{p=1}^{P} a_p \lambda^p}. \tag{4.29}$$

Finding the filter coefficients by minimizing, in a least squares sense, (4.29) leads to a set of nonlinear equations in the filter coefficients. Similar to the Padé approximation in the time domain [18], we can multiply both sides of (4.29) by the denominator expression of the frequency response to obtain the new (not equivalent) error

$$e(\lambda) = h^*(\lambda) + h^*(\lambda) \sum_{p=1}^{P} a_p \lambda^p - \sum_{q=0}^{Q} b_q \lambda^q, \tag{4.30}$$

which is now linear in $a_p$ and $b_q$. Then, by staking the errors for different $\lambda_n$ (e.g., from the eigendecomposition of $\mathbf{S}$ or by gridding the interval $[\lambda_{\min}, \lambda_{\max}]$ into $N$ points) in $\mathbf{e} = [e(\lambda_1), \dots e(\lambda_N)]^\mathsf{T}$ and defining $\mathbf{a} = [a_1, \dots, a_P]^\mathsf{T}$, $\mathbf{b} = [b_1, \dots, b_Q]^\mathsf{T}$ as the vectors containing the filter coefficients, we rewrite (4.30) as

$$\mathbf{e} = \mathbf{h}^* + \mathrm{diag}(\mathbf{h}^*)\boldsymbol{\Psi}_P \mathbf{a} - \boldsymbol{\Psi}_{Q+1}\mathbf{b}, \tag{4.31}$$

where $\mathbf{h}^* = [h^*(\lambda_1), \dots h^*(\lambda_N)]^\mathsf{T}$ is the $N \times 1$ vector containing the desired frequency response, $\boldsymbol{\Psi}_P$ is a Vandermonde-like $N \times P$ matrix with $(r,p)$-th entry $[\boldsymbol{\Psi}_P]_{r,p} = \lambda_r^p$ and

$\mathbf{\Psi}_{Q+1}$ is the $N \times (Q+1)$ matrix still with a Vandermonde-like structure, but with $(r,q)$-th entry $[\mathbf{\Psi}_{Q+1}]_{r,q} = \lambda_r^{q-1}$. With the new notation in place, the convergence condition of the filter $\|\mathbf{P}\| < 1$ can be expressed as $\|\mathbf{\Psi}_P \mathbf{a}\|_\infty < 1$, where the latter inequality can be derived from the expression of $\mathbf{P}$ and by considering that $\mathbf{S} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathsf{H}}$, or from gridding. Then, convergent filter coefficients that minimize $\mathbf{e}$ are obtained by solving the convex constrained least squares problem

$$\begin{aligned} \underset{\mathbf{a},\mathbf{b}}{\text{minimize}} \quad & \|\mathbf{h}^* + \text{diag}(\mathbf{h}^*)\mathbf{\Psi}_P \mathbf{a} - \mathbf{\Psi}_{Q+1}\mathbf{b}\|_2 \\ \text{subject to} \quad & \|\mathbf{\Psi}_P \mathbf{a}\|_\infty \leq \delta_P, \quad \delta_P < 1, \end{aligned} \tag{4.32}$$

where $\delta_P$ is our handle to trade convergence speed with approximation accuracy. We have observed that a good choice for $\delta_P$ is in the range $[0.6, 0.8]$. Such consideration comes from the fact that a higher value of $\delta_P$ lead to slower convergence (see also Fig. 4.5) and thus larger computational costs. On the other hand, values of $\delta_P < 0.6$ are generally not recommended since the approximation accuracy is severely compromised.

While solving (4.32) will produce the stable filter coefficients that minimize $e(\lambda_n)$, we are interested in minimizing $e'(\lambda_n)$. As we previously said, this problem is non convex due to the denominator expresion in $\mathbf{a}$. However, once the optimal solution $\mathbf{a}^\star$ of (4.32) is obtained, we can plug it back into (4.29) and minimize the latter solely w.r.t. $\mathbf{b}$. This step remainds once again the Shanks' method in Section 4.2.3, but now with theoretical guarantees that (4.26) converges to the designed graph frequency response.

### 4.3.3. NUMERICAL RESULTS

Let us now illustrate and compare the performance of the parallel ARMA$_K$ (4.7) with the feedback-looped ARMA$_{P,Q}$ (4.26). In addition, we analyze the FastIDIIR algorithm of [5], which uses the gradient descent to implement an inverse filtering distributively[7].

We consider a random geometric graph of $N = 100$ nodes randomly distributed in a squared area with two nodes being neighbors if they are closer than 15% of the maximum distance. The shift operator is $\mathbf{S} = \mathbf{L}_{\text{n}} - \frac{\lambda_{\max}(\mathbf{L}_{\text{n}})}{2}\mathbf{I}$, with $\lambda_{\max}(\mathbf{L}_{\text{n}})$ the maximum eigenvalue of $\mathbf{L}_{\text{n}}$ for the *particular* graph. All the filters are designed to approximate an ideal low pass filter in the frequency domain of $\mathbf{S}$. The filter cut-off frequency is $\lambda_c = \lambda_{N/2}(\mathbf{S})$, i.e., the half of the band. The input signal $\mathbf{x}$ has a white unitary spectrum w.r.t. the underlying graph and the filters are initialized as $\mathbf{y}_0 = \mathbf{x}$. Our results are averaged over 10 iterations.

We compare the convergence time of the three distributed algorithms that are characterized by a rational frequency response for $K = 10$, 20. Note that the *per iteration* complexity of the ARMA$_{P,Q}$ is at most equal to that of the other two approaches (equality when $P + Q = K$). For the ARMA$_K$ and for the FastIDIIR the Shanks' method in Section 4.2.3 is used to design the filter coefficients, while for the ARMA$_{P,Q}$ we followed the approach in Section 4.3.2. In the latter case, for each graph, the values of $P$ and $Q$ ($P + Q \leq K$ to have at most same cost) with the smallest approximation error are selected with $\delta_P = 0.65$.

[7] In the original paper [2], we compared the performance also with the Chebyshev polynomial-based design FIR. This is not presented here as we were not able to reproduce the same results. We attribute this mismatch is to numerical issues due to the high randomness of the scenario and the few averaging.

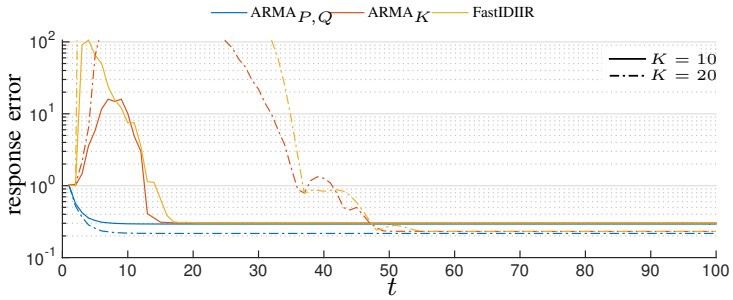



Figure 4.6: Normalized response error as a function of time for the three distributed algorithms to implement a rational frequency response. The results are shown for orders $K = 10, 20$. The ARMA$_{P,Q}$ is designed such that $P + Q \leq K$ to ensure the same distributed cost.



Figure 4.6 depicts the response error in terms of NMSE between the filter frequency responses and the desired one as a function of time. We can see that ARMA$_{P,Q}$ converges faster than the other algorithms yet ensuring the same approximation accuracy. We believe the large values in the transition phase of ARMA$_K$ and FastIDIIR are due to the design strategy that does not aim to minimize the approximation error and focuses only in the minimization after $K$ iterations.

## 4.4. CONCLUDING REMARKS

In this chapter, we introduced the ARMA graph filters on graphs with the aim to implement rational, rather than polynomial, graph frequency responses. These filters can be implemented distributively in the vertex domain in an IIR fashion. We proposed three different constructions and for each of them we introduced a Shanks's-based design method to approximate a given frequency response.The ARMA graph filters gave exact closed-form solutions for the tasks of Tikhonov-, Wiener-based denoising, and signal interpolation under smoothness assumption.

Their IIR implementation allows the ARMA filters to achieve the designed frequency response theoretically after infinite iterations. We showed for all architectures that they are characterized by a linear convergence, and they can be arrested within few iterations with an error less than 1% from the theoretical steady state.

Our theoretical and numerical evaluations showed that the ARMA architectures yield a similar performance compared to FIR graph filters. Later in Chapters 6-7 we analyze both the FIR and ARMA in time-varying scenarios (e.g., time-varying graphs and graph signals) and show that the graph filters deal differently with the time-varying phenomena.

# APPENDICES

## 4.A. PROOF OF THE ARMA$_1$ FREQUENCY RESPONSE THEOREM

By expanding (4.4a) to all its terms we obtain

$$\mathbf{y}_t = (\psi\mathbf{S})^t\mathbf{y}_0 + \varphi\sum_{\tau=0}^{t-1}(\psi\mathbf{S})^\tau\mathbf{x}. \tag{4.33}$$

Then for $|\psi\varrho| < 1$ and $t \to \infty$, (4.33) approaches the steady state

$$\mathbf{y} = \lim_{t\to\infty}\mathbf{y}_t = \varphi\sum_{\tau=0}^{\infty}(\psi\mathbf{S})^\tau\mathbf{x} = \varphi(\mathbf{I}_N - \psi\mathbf{S})^{-1}\mathbf{x}, \tag{4.34}$$

irrespectively of $\mathbf{y}_0$. From Sylvester's matrix theorem, matrix $(\mathbf{I}_N - \psi\mathbf{S})$ has the same eigenvectors as $\mathbf{S}$ and its eigenvalues are equal to $1-\psi\lambda_n$. It is also well known that invertible matrices have the same eigenvectors as their inverse and eigenvalues, that are the inverse of the eigenvalues of their inverse. Thus,

$$\mathbf{z} = \lim_{t\to\infty}\mathbf{z}_t = \mathbf{y} + c\mathbf{x} = \sum_{n=1}^{N}\left(c + \frac{\varphi}{1-\psi\lambda_n}\right)\hat{x}_n\mathbf{u}_n, \tag{4.35}$$

and the frequency response (4.5) follows by simple algebra. We arrived at (4.35) by considering a specific realization of $\mathbf{S}$, thus the set of eigenvalues $\lambda_n \in [\lambda_{\min}, \lambda_{\max}]$ is discrete. However, the same result is achieved for every other graph realization matrix $\mathbf{S}$ with a potentially different set of eigenvalues, still in $\lambda_n \in [\lambda_{\min}, \lambda_{\max}]$. Therefore, we can write (4.5) for all $\lambda \in [\lambda_{\min}, \lambda_{\max}]$.

## 4.B. PROOF OF THE PERIODIC ARMA$_K$ FREQUENCY RESPONSE THEOREM

Define matrices $\mathbf{\Gamma}_t = \theta_t\mathbf{I}_N + \psi_t\mathbf{S}$ and $\mathbf{\Phi}(t,t') = \mathbf{\Gamma}_t\mathbf{\Gamma}_{t-1}\ldots\mathbf{\Gamma}_{t'}$ if $t \geq t'$, whereas $\mathbf{\Phi}_\Gamma(t,t') = \mathbf{I}_N$ otherwise. The output at the end of each period can be rewritten as a time-invariant system

$$\mathbf{y}_{(i+1)K} = \overbrace{\mathbf{\Phi}_\Gamma(K-1,0)}^{\triangleq\mathbf{A}}\mathbf{y}_{iK} + \overbrace{\sum_{k=0}^{K-1}\mathbf{\Phi}_\Gamma(K-1,k+1)\varphi_k\mathbf{x}}^{\triangleq\mathbf{B}} \tag{4.36a}$$

$$\mathbf{z}_{(i+1)K} = \mathbf{y}_{(i+1)K} + c\mathbf{x}. \tag{4.36b}$$

Both $\mathbf{A}$ and $\mathbf{B}$ have the same eigenvectors $\mathbf{U}$ as $\mathbf{S}$. Notice that (4.36) resembles (4.4) and we can proceed in an identical manner. As such, when the maximum eigenvalue of $\mathbf{A}$ is bounded by $|\lambda_{\max}(\mathbf{A})| < 1$, the steady state of (4.36) is

$$\mathbf{z} = (\mathbf{I}_N - \mathbf{A})^{-1}\mathbf{B}\mathbf{x} + c\mathbf{x} = \sum_{n=1}^{N}\left(c + \frac{\lambda_n(\mathbf{B})}{1-\lambda_n(\mathbf{A})}\right)\hat{x}_n\mathbf{u}_n. \tag{4.37}$$

To derive the exact response, we exploit the backward product in the definition of $\mathbf{\Phi}_\Gamma(t_1, t_2)$ and obtain

$$\lambda_n(\mathbf{\Phi}_\Gamma(t_1, t_2)) = \prod_{\tau=t_1}^{t_2} \lambda_n(\mathbf{\Gamma}_t) = \prod_{\tau=t_1}^{t_2} (\theta_\tau + \psi_\tau \lambda_n), \tag{4.38}$$

which, by the definition of $\mathbf{A}$ and $\mathbf{B}$, yields the desired frequency response. The linear convergence rate and convergence condition follow from the linear convergence of (4.36) to $\mathbf{z}$ with rate $|\lambda_{\max}(\mathbf{A})|$.

## 4.C. PROOF OF THE FEEDBACK-BASED ARMA$_{P,Q}$ FREQUENCY RESPONSE PROPOSITION

We write recursion (4.26) as

$$\mathbf{y}_t = \mathbf{P}^t \mathbf{y}_0 + \sum_{\tau=0}^{t-1} \mathbf{P}^\tau \mathbf{Q}\mathbf{x}. \tag{4.39}$$

When $\|\mathbf{P}\| < 1$ and for $t \to \infty$ we approach the steady state

$$\mathbf{y} = \lim_{t\to\infty} \mathbf{y}_t = \sum_{\tau=0}^{\infty} \mathbf{P}^\tau \mathbf{Q}\mathbf{x} = (\mathbf{I} - \mathbf{P})^{-1} \mathbf{Q}\mathbf{x}. \tag{4.40}$$

Substituting back the expressions for $\mathbf{P}$ and $\mathbf{Q}$ in (4.40) and applying the GFT, we obtain the relationship between the $n$th frequency component of the output $\hat{y}_n$ and input $\hat{x}_n$:

$$\hat{y}_n = \left(1 + \sum_{p=1}^{P} a_p \lambda_n^p\right)^{-1} \left(\sum_{q=0}^{Q} b_q \lambda_n^q\right) \hat{x}_n. \tag{4.41}$$

The frequency response $\hat{y}_n$ in (4.27) can simply be obtained by a pointwise division between $\hat{y}_n$ and $\hat{x}_n$. The sufficient condition for convergence can be obtained by substituting the expression of $\mathbf{P}$ in $\|\mathbf{P}\| < 1$ and then applying the triangle and Cauchy-Schwarz inequality while remembering that $\|\mathbf{S}\| = \varrho$.

## 4.D. PROOF OF THE ARMA$_{P,Q}$ CONVERGENCE TIME PROPOSITION

Given the ARMA$_{P,Q}$ output at time $t$ in (4.26), the error norm w.r.t. the steady state $\mathbf{y} = \lim_{t\to\infty} \mathbf{y}_t$ is

$$\|\mathbf{y}_t - \mathbf{y}\| \leq \|\mathbf{P}\|^t \|\mathbf{y}_0\| + \sum_{\tau=t}^{\infty} \|\mathbf{P}\|^\tau \|\mathbf{Q}\| \|\mathbf{x}\|, \tag{4.42}$$

where in (4.42) we have applied the triangle and Cauchy-Schwarz inequality of the norms. Then, by considering that ARMA$_{P,Q}$ is stable, i.e., $\|\mathbf{P}\| < 1$, we can express the geometric series in closed form

$$\begin{aligned} \|\mathbf{y}_t - \mathbf{y}\| &\leq \|\mathbf{P}\|^t \|\mathbf{y}_0\| + (1 - \|\mathbf{P}\|)^{-1} \|\mathbf{P}\|^t \|\mathbf{Q}\| \|\mathbf{x}\| \\ &\leq \|\mathbf{P}\|^t \left(\|\mathbf{y}_0\| + (1 - \|\mathbf{P}\|)^{-1} \|\mathbf{Q}\| \|\mathbf{x}\|\right). \end{aligned} \tag{4.43}$$

Then, for $\|\mathbf{y}_t - \mathbf{y}\| \leq \epsilon$ we have

$$t\,(ln\,(\|\mathbf{P}\|)) \leq ln\left(\frac{\epsilon}{\|\mathbf{y}_0\| + (1 - \|\mathbf{P}\|)^{-1} \|\mathbf{Q}\| \|\mathbf{x}\|}\right), \tag{4.44}$$

which can be reformulated into (4.28) by dividing both sides of (4.44) by $ln\,(\|\mathbf{P}\|) < 0$.

**4.E.** Parallel ARMA$_K$ coefficients

Table 4.1: Residues $r_k$ and poles $p_k$ of parallel ARMA$_K$ filter, for $K = 3, 5$ and 7.

| *order* | $r_0, p_0$ | $r_1, p_1$ | $r_2, p_2$ | $r_3, p_3$ | $r_4, p_4$ | $r_5, p_5$ | $r_6, p_6$ |
|---|---|---|---|---|---|---|---|
| K=3 | 10.954 + 0$i$,<br>-6.666 + 0$i$ | 1.275 + 1.005$i$,<br>0.202 + 1.398$i$ | 1.275 - 1.005$i$,<br>0.202 - 1.398$i$ | - | - | - | - |
| K=5 | -7.025 + 0$i$,<br>-3.674 + 0$i$ | -1.884 - 1.298$i$,<br>-0.420 + 1.269$i$ | -1.884 + 1.298$i$,<br>-0.420 - 1.269$i$ | 1.433 - 1.568$i$,<br>0.703 + 1.129$i$ | 1.433 + 1.568$i$,<br>0.703 + 1.129$i$ | - | - |
| K=7 | -46.398 + 0$i$,<br>–3.842 + 0$i$ | -20.207 - 8.343$i$,<br>0.102 + 1.427$i$ | -20.207 + 8.343$i$,<br>0.102 + 1.427$i$ | -5.205 + 4.946$i$,<br>-0.785 + 1.128$i$ | -5.205 - 4.946$i$,<br>-0.785 + 1.128$i$ | 3.124 - 10.622$i$,<br>0.902 + 1.011$i$ | 3.124 + 10.622$i$,<br>0.902 - 1.011$i$ |

## FURTHER READING

[1] E. Isufi, A. Loukas, A. Simonetto, and G. Leus, *Autoregressive moving average graph filtering,* IEEE Transactions on Signal Processing **65**, 274 (2017).

[2] E. Isufi, A. Loukas, and G. Leus, *Autoregressive moving average graph filters a stable distributed implementation,* in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, EPFL-CONF-223825 (2017).

[3] A. Loukas, M. Zuniga, M. Woehrle, M. Cattani, and K. Langendoen, *Think globally, act locally: On the reshaping of information landscapes,* in *Proceedings of the 12th international conference on Information processing in sensor networks* (ACM, 2013) pp. 265–276.

[4] A. Loukas, M. Zuniga, I. Protonotarios, and J. Gao, *How to identify global trends from local decisions? event region detection on mobile networks,* in *INFOCOM, 2014 Proceedings IEEE* (IEEE, 2014) pp. 1177–1185.

[5] X. Shi, H. Feng, M. Zhai, T. Yang, and B. Hu, *Infinite impulse response graph filters in wireless sensor networks,* IEEE Signal Processing Letters **22**, 1113 (2015).

[6] A. Loukas, A. Simonetto, and G. Leus, *Distributed autoregressive moving average graph filters,* IEEE Signal Processing Letters **22**, 1931 (2015).

[7] D. P. Bertsekas, A. Nedi, A. E. Ozdaglar, *et al., Convex analysis and optimization,* (2003).

[8] S. Boyd and L. Vandenberghe, *Convex optimization* (Cambridge university press, 2004).

[9] J. L. Shanks, *Recursion filters for digital processing,* Geophysics **32**, 33 (1967).

[10] C. Zhang, D. Florêncio, and P. A. Chou, *Graph signal processing–A probabilistic framework,* Microsoft Res., Redmond, WA, USA, Tech. Rep. MSR-TR-2015-31 (2015).

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

[12] Y. Mao, G. Cheung, and Y. Ji, *Image interpolation for dibr viewsynthesis using graph fourier transform,* in *3DTV-Conference: The True Vision-Capture, Transmission and Display of 3D Video (3DTV-CON), 2014* (IEEE, 2014) pp. 1–4.

[13] M. Belkin and P. Niyogi, *Semi-supervised learning on riemannian manifolds,* Machine learning **56**, 209 (2004).

[14] T. Zhang, A. Popescul, and B. Dom, *Linear prediction models with graph regularization for web-page categorization,* in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining* (ACM, 2006) pp. 821–826.

[15] D. I. Shuman, P. Vandergheynst, D. Kressner, and P. Frossard, *Distributed signal processing via chebyshev polynomial approximation,* arXiv preprint arXiv:1111.5239v3 (2017).

[16] J. Liu and E. Isufi and G. Leus, *Autoregressive moving average graph filter design,* in *2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)* (2017) pp. 593–597.

[17] J. Liu, E. Isufi, and G. Leus, *Filter design for autoregressive moving average graph filters,* arXiv preprint arXiv:1711.09086 (2017).

[18] M. H. Hayes, *Statistical digital signal processing and modeling* (John Wiley & Sons, 2009).

# III

# GRAPH-TIME FILTERING

# 5

# GRAPH-TIME SIGNAL PROCESSING

*The best prophet of the future is the past.*

Lord Byron

In the last three chapters, we considered the graph signal to be invariant over time. In fact, the developed definitions in Chapter 2 and filters in Chapters 3-4 consider a single temporal snapshot as graph signal. This chapter takes one step further by considering the graph signal to be time-varying. By introducing the time into the analysis, we will merge discrete signal processing (DSP) and GSP concepts, which, if properly exploited, yield substantial benefits compared to the case where the temporal, or the graph dimension are processed separately. Although it would be possible to move directly to Chapter 6, the reader should look through this chapter to become familiar with the introduced concepts, which will help to better focalize the arguments treated in Chapters 6-7.

This chapter is organized as follows. Section 5.1 motivates the joint graph-time processing of time-varying graph signals and briefly surveys the state-of-the-art works that exploit this coupling. Section 5.2 formalizes the notion of time-varying graph signals and introduces the concepts of product graph and joint graph-time shift operator. The joint graph and time Fourier transform (GTFT) is described in Section 5.3. In this section, the graph-time filters are introduced as well. Finally, Section 5.4 concludes the chapter with a summary of the treated arguments.

## 5.1. INTRODUCTION

To date, few recent works such as [1–8] have mentioned the potential extension of GSP to a graph-time signal processing framework and formalize the concepts we saw in Part II. This analysis finds practical application in temperature observations over sensor networks, or traffic diffusion in road networks, to name a couple where the graph signal is time-varying in nature, e.g., temperature measurements in a given temporal window, or the number of cars in the crossroads throughout the day. It is then reasonable to process

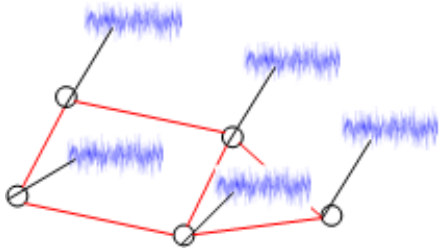


Figure 5.1: Illustration of a time-varying signal on the vertices of a graph.

these signals in a joint graph-time fashion, by considering their joint variation on both the graph and the temporal dimension.

The first notions of extending GSP to time is proposed in [1], where the so-called product graphs are used to link the nodes though time. Later, [2] and [3] introduce the concept of graph-time filtering, which is further extended in [5, 6]. The work in [4] leverages the graph wavelet theory to enable visual analysis of time-varying graph signals. The authors in [7] focus on learning a graph topology that captures the signal evolution through a restricted vector autoregressive process. Finally, the recent work [8] provides a good overview of the several concepts used in processing time-varying graph signals.

This chapter recalls the joint spectral analysis and filtering of time-varying graph signals. By introducing the concept of the joint graph-time shift operator– an operator that captures the node interactions among graph and time–the definition of the GTFT boils down to that of projecting the time-varying graph signal onto the eigenspace of this new, yet more complex, shift operator.

## 5.2. TIME-VARYING SIGNALS ON GRAPHS

We indicate as $\mathbf{x}_t$ a time-varying graph signal over the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ of $N$ nodes. The $N \times T$ matrix $\mathbf{X}_T = [\mathbf{x}_1, \ldots, \mathbf{x}_T]^\mathsf{T} \in \mathbb{R}^{N \times T}$ collects $T$ successive temporal measurements of $\mathbf{x}_t$. Figure 5.1 depicts one example of a time-varying graph signals on a graph. Through the DFT we can perform a temporal spectral analysis of $\mathbf{X}_T$ by decomposing *each row* of $\mathbf{X}_T$ independently in the temporal oscillating modes (i.e., the complex exponentials). More specifically, we have

$$[\hat{\mathbf{X}}_T]_{\text{DFT}} = \mathbf{X}_T \mathbf{F}^*, \tag{5.1}$$

where $[\hat{\mathbf{X}}_T]_{\text{DFT}}$ denotes the DFT of the row of $\mathbf{X}_T$ and $\mathbf{F}$ is the normalized DFT matrix defined as

$$[\mathbf{F}]_{t,k} = \frac{e^{\jmath \omega_i t}}{\sqrt{T}}, \qquad \text{with} \qquad \omega_i = \frac{2\pi(i-1)}{T}, \tag{5.2}$$

with $t, i = 1, \ldots, T$. Similarly, through the GFT we can perform a graph spectral analysis of $\mathbf{X}_T$ by decomposing *each column* of $\mathbf{X}_T$ independently (each column of $\mathbf{X}_T$ represents a graph signal snapshot) in the graph oscillating modes (i.e., the eigenvectors of the shift operator). That is,

$$[\hat{\mathbf{X}}_T]_{\text{GFT}} = \mathbf{U}^\mathsf{H} \mathbf{X}_T, \tag{5.3}$$

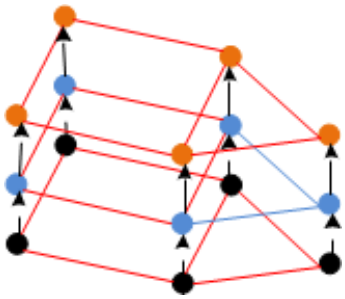


Figure 5.2: Figure illustrating an example of a joint graph as the exentsion of the orignial graph $\mathcal{G}$ to $\mathcal{G}_3$.

where $[\hat{\mathbf{X}}_T]_{\text{GFT}}$ is the $N \times T$ matrix collecting the GFTs of all realizetions of $\mathbf{x}_t$ in $\mathbf{X}_T$.

Transforms (5.1) and (5.3) represent two different ways of processing $\mathbf{x}_t$ in each domain, separately. They will be our starting point to formalize the GTFT for $\mathbf{X}_T$. In this regard, next, we introduce an equivalent representation of $\mathbf{x}_t$ on $\mathcal{G}$, and the joint graph-time shift operator.



### 5.2.1. THE JOINT GRAPH

An alternative way of representing $\mathbf{X}_T$ on $\mathcal{G}$, is that of considering a time-invariant graph signal $\mathbf{x}_T = \text{vec}(\mathbf{X}_T)$ that lives on a joint extended graph $\mathcal{G}_T$ including $T$ copies of $\mathcal{G}$. With the illustration in Figure 5.2 for $T = 3$, the extended graph $\mathcal{G}_T$ has a vertex set $\mathcal{V}_T = \cup_{t=1}^{T} \mathcal{V}_t$ with $NT$ nodes, and where each node has as graph signal the time-invariant entries of $\mathbf{x}_T$. The edge set of $\mathcal{G}_T$ consists of the $MT$ edges it inherents from the unconnected copies of $\mathcal{G}$, besides the $NT$ extra edges connecting the consecutive realizations of $\mathbf{x}_t$ on a particular node. That is, a directed edge connecting the node $v_{i,t}$ (node $v_i$ in the $t$th realization) with the node $v_{i,t+1}$ (node $v_i$ in the $t+1$th realization).

This extended graph $\mathcal{G}_T$ can be obtained as $\mathcal{G}_T = \mathcal{G} \overset{c}{\times} \mathcal{G}'$, with $\overset{c}{\times}$ denoting the Cartesian product and $\mathcal{G}'$ the directed chain graph[1] [9].

### 5.2.2. THE JOINT GRAPH-TIME SHIFT OPERATOR

From the Cartesian product relation of $\mathcal{G}_T$ with $\mathcal{G}$, we can compute the shift operator $\mathbf{S}_T$ of $\mathcal{G}_T$ as

$$\mathbf{S}_T = \mathbf{S}' \otimes \mathbf{I}_N + \mathbf{I}_T \otimes \mathbf{S} \tag{5.4}$$

where $\mathbf{S}'$ is the shift operator matrix of $\mathcal{G}'$, and $\otimes$ denotes the Kronecker product. Though $\mathcal{G}_T$ and $\mathcal{G}'$ are directed graphs, $\mathbf{S}_T$ can always be diagonalized as [8]

$$\begin{aligned} \mathbf{S}_T &= (\mathbf{F}\boldsymbol{\Omega}\mathbf{F}^{\mathsf{H}}) \otimes \mathbf{I}_N + \mathbf{I}_T \otimes (\mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^{\mathsf{H}}) \\ &= (\mathbf{F} \otimes \mathbf{U})(\boldsymbol{\Omega} \oplus \boldsymbol{\Lambda})(\mathbf{F} \otimes \mathbf{U})^{\mathsf{H}} = \mathbf{U}_T \boldsymbol{\Lambda}_T \mathbf{U}_T^{\mathsf{H}}, \end{aligned} \tag{5.5}$$

where $\boldsymbol{\Omega}$ is the diagonal matrix containing the complex exponential of the angular frequencies, i.e., $[\boldsymbol{\Omega}]_{i,i} = e^{-\jmath\omega_i}$ with $\omega_i$ defined in (5.2).

[1] A directed chain graph of $N$ nodes has each node being connected by a direct graph to only another node, i.e., $v_1 \to v_2 \to \ldots \to v_N$.

## 5.3. GRAPH-TIME FREQUENCY ANALYSIS

### 5.3.1. GRAPH AND TIME FOURIER TRANSFORM

Similarly as the GFT, the GTFT of $\mathbf{x}_T$ is defined as

$$[\hat{\mathbf{x}}_T]_{\text{GTFT}} = \mathbf{U}_T^{\mathsf{H}} \mathbf{x}_T, \tag{5.6}$$

which from the properties of the vec operator and from the structure of $\mathbf{U}_T$ becomes

$$[\hat{\mathbf{X}}_T]_{\text{GTFT}} = \mathbf{U}^{\mathsf{H}} \mathbf{X}_T \mathbf{F}^*. \tag{5.7}$$

Put simply, the GTFT (5.7) consists of applying the DFT and the GFT to the matrix $\mathbf{X}_T$.

Since $\mathbf{U}_T$ is a unitary matrix by construction, the inverse GTFT is $\mathbf{x}_t = \mathbf{U}_T[\hat{\mathbf{x}}_T]_{\text{GTFT}}$, or in the matrix form $\mathbf{X}_T = \mathbf{U}[\hat{\mathbf{X}}_T]_{\text{GTFT}}\mathbf{F}^{\mathsf{T}}$. Moreover, the Parseval result on the energy preservation holds. Finally, we remark that the GTFT is a composed linear operation, and thus the $[\hat{\mathbf{X}}_T]_{\text{GTFT}}$ can be obtained as

$$[\hat{\mathbf{X}}_T]_{\text{GTFT}} = \mathbf{U}^{\mathsf{H}}[\hat{\mathbf{X}}_T]_{\text{DFT}} = [\hat{\mathbf{X}}_T]_{\text{GFT}}\mathbf{F}^*, \tag{5.8}$$

i.e., first apply the DFT and then the GFT to $\mathbf{X}_T$, or vice-versa.

### 5.3.2. GRAPH-TIME FILTERING

Following the definition of the graph filtering operation in Section 2.3.5, we now formalize the notion of joint graph-time filtering for a time varying graph signal.

**Definition 5.1** (Joint graph-time filters)**.** *A joint graph-time filter $h(\lambda_n, \omega_i)$ is defined as a function over the graph frequencies $\{\lambda_n\}_{n=0}^{N-1}$ and angular frequencies $\{\omega_i\}_{i=1}^{T}$ to the set of real numbers, i.e., $h : (\{\lambda_n\}_{n=0}^{N-1} \times \{\omega_i\}_{i=1}^{T}) \to \mathbb{R}$, altering the joint graph-temporal frequency content $[\hat{\mathbf{x}}_T]_{GTFT}$ of $\mathbf{x}_T$ as a point-wise multiplication in the joint Fourier domain. The graph filter output at the frequency tuple $(\lambda_n, \omega_i)$ is $[\hat{\mathbf{y}}_T]_{GTFT}(\lambda_n, \omega_i) = h(\lambda_n, \omega_i)[\hat{\mathbf{x}}_T]_{GTFT}(\lambda_n, \omega_i)$.*

In vector form, the filter output writes as

$$[\hat{\mathbf{y}}_T]_{\text{GTFT}} = h(\mathbf{\Lambda}, \mathbf{\Omega})[\hat{\mathbf{x}}_T]_{\text{GTFT}}, \tag{5.9}$$

where $h(\mathbf{\Lambda}, \mathbf{\Omega})$ is an $N \times T$ matrix containing the *joint graph-time frequency response*. By means of the inverse GTFT, we have

$$\mathbf{y}_T = \mathbf{U}_T[\hat{\mathbf{y}}_T]_{\text{GTFT}} = \mathbf{U}_T h(\mathbf{\Lambda}, \mathbf{\Omega})\mathbf{U}_T^{\mathsf{H}} \mathbf{x}_T, \tag{5.10}$$

where now the graph-time filtering matrix $\mathbf{H} = \mathbf{U}_T h(\mathbf{\Lambda}, \mathbf{\Omega})\mathbf{U}_T^{\mathsf{H}}$ is of dimensions $NT \times NT$ and is referred to as the joint graph-time impulse response. To implement distributively the filtering operation (5.10), in Chapter 6 we introduce the joint graph-time FIR and ARMA filters.

## 5.4. CONCLUDING REMARKS

This chapter introduced the framework of processing time-varying graph signals by exploiting the latter variation on both the graph and temporal dimension. By introducing the notion of the joint graph, i.e., a larger graph that is expanded in both the number of nodes and edges to capture the temporal dimension of the signal, we introduced an equivalent graph shift operator that captures the signal relations in a graph-time fashion. Similarly to the GFT, the eigendecomposition of this graph-time shift operator carries a notion of frequency and characterizes the signal joint variation over the graph and time. Finally, the concept of graph-time filtering as a function acting jointly on the graph and temporal frequencies is introduced.

## FURTHER READING

[1] A. Sandryhaila and J. M. Moura, *Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure,* IEEE Signal Processing Magazine **31**, 80 (2014).

[2] A. Loukas, A. Simonetto, and G. Leus, *Distributed autoregressive moving average graph filters,* IEEE Signal Processing Letters **22**, 1931 (2015).

[3] E. Isufi, A. Loukas, A. Simonetto, and G. Leus, *Autoregressive moving average graph filtering,* IEEE Transactions on Signal Processing **65**, 274 (2017).

[4] P. Valdivia, F. Dias, F. Petronetto, C. T. Silva, and L. G. Nonato, *Wavelet-based visualization of time-varying data on graphs,* in *Visual Analytics Science and Technology (VAST), 2015 IEEE Conference on* (IEEE, 2015) pp. 1–8.

[5] E. Isufi, A. Loukas, A. Simonetto, and G. Leus, *Separable autoregressive moving average graph-temporal filters,* in *Signal Processing Conference (EUSIPCO), 2016 24th European* (IEEE, 2016) pp. 200–204.

[6] E. Isufi, G. Leus, and P. Banelli, *2-dimensional finite impulse response graph-temporal filters,* in *Signal and Information Processing (GlobalSIP), 2016 IEEE Global Conference on* (IEEE, 2016) pp. 405–409.

[7] J. Mei and J. M. Moura, *Signal processing on graphs: Causal modeling of unstructured data,* IEEE Transactions on Signal Processing **65**, 2077 (2017).

[8] F. Grassi, A. Loukas, N. Perraudin, and B. Ricaud, *A time-vertex signal processing framework,* arXiv preprint arXiv:1705.02307 (2017).

[9] W. Imrich, S. Klavžar, and D. F. Rall, *Topics in graph theory: Graphs and their Cartesian product* (AK Peters/CRC Press, 2008).

# 6

# DETERMINISTIC ANALYSIS OF GRAPH-TIME FILTERING

*The two most powerful warriors are patience and time.*

Leo Tolstoy

In Chapters 3 and 4 we analyzed graph filters in static scenarios, i.e., when both the graph topology and the graph signal do not change in time. We now depart from this philosophy to investigate the graph filters' behavior in dynamic environments, i.e., when the graph topology and (or) the graph signal change(s) deterministically in time. This analysis is crucial since it characterizes the filters' robustness in scenarios involving moving sensors and time-varying graph signals, such as temperature measurements. A salient contribution of the distributed graph filters is their natural extension to joint graph-time filters when the input signal has a time-varying nature. So, the filters developed in Chapters 3 and 4 will now be capable to carry out distributively the graph-time filtering operation described in Chapter 5.

To the best of our knowledge, prior contributions that analyze the filters' robustness in deterministic dynamic environments are limited to [4] and [5]. The former work concerns the convergence of the potential kernel in dynamic environments. The latter work proposes joint graph-time ARMA filtering and tests it. In the co-authored work [1], we start from these two works to extend the potential kernel analysis to more involved ARMA filters. We give a more in-depth analysis of the filter behavior and characterize with closed-form expression the filter behavior in dynamic scenarios. In the follow-up works [2, 3], we introduce the distributed FIR and ARMA graph-time filters, respectively.

The next section of this chapter motivates the deterministic analysis of graph-time filtering, along with our contributions and potential applications. Section 6.2 analyzes the behavior of the parallel $\text{ARMA}_K$ graph filter operating on a time-varying graph with

Parts of this chapter have been published in the IEEE Transactions on Signal Processing [1] (2017), in the EURASIP EUSIPCO [2] (20016), and in the IEEE GlobalSIP [3] (2016).

a time-(in)variant graph signal as input. The two-dimensional distributed graph-time filters are developed in Section 6.3. Section 6.4 recaps the chapter main concepts.

## 6.1. INTRODUCTION

Distributed graph filters, in both their FIR and ARMA implementations, showed promising results in performing distributively GSP tasks such as clustering, denoising, and interpolation. However, the analysis conducted in Chapters 3 and 4 is performed for static inputs over static graphs. One can then raise the question: *"Do the shown graph filter properties hold when the graph and/or the graph signal are a function of time?"*. The answer to this question along with the answers to the related research questions ($Q2.1$) will be the central arguments of this chapter.

Temporal variations in the graph signal are encountered in a wide range of applications, starting from sequential sensor measurements to brain signal monitoring. In this instance, we are interested in denoising and/or interpolating the signal from the temporal stream of data. To address this aspect, we show that the introduced ARMA graph filters naturally inherit the property to perform a distributed graph-time filtering when the graph signal is time-varying. We first discuss the filter behavior to characterize its robustness to graph signal variations. Then, to achieve a distributed joint graph-time filtering, we introduce two architectures that produce two-dimensional FIR and ARMA graph-temporal filters. These filters have the ability to extract relevant information of the time-varying graph signal, such as an interferer in sensor networks.

Along with the variations in the graph signal, we consider the graph topology to change under a specific model over time, e.g., sensor movements. We show that the introduced ARMA graph filters are able to handle these variations, and we give a closed-form expression of the filter output when both the graph topology and graph signal change over time.

### 6.1.1. CONTRIBUTIONS

This chapter brings the following contributions to the GSP research field.

**Contribution 6.1.** *Through the analysis of the ARMA graph filter for time-varying input signals, we introduce the notion of joint graph-time filtering.* The proposed recursions naturally extend to two-dimensional filters operating simultaneously in the graph frequency domain and in the temporal frequency domain. We show that the ARMA filter output remains close to the correct time-varying solution (under sufficient conditions on the input), which characterizes the robustness to dynamics of our algorithms.

**Contribution 6.2.** *We give theoretical guarantees about the ARMA graph filter behavior when both the graph topology and the graph signal are time-varying.* Specifically, we provide sufficient conditions for filter stability and show that a decomposition basis exists (uniquely determined by the sequence of graph realizations), over which the filters achieve the same frequency response as in the static case. We find that a graph naturally dampens temporal frequencies in a manner depending on its spectrum. Exploiting this

finding, we extend the ARMA designs presented in Chapter 4 to allow also a measure of control over the temporal frequency response of the filters.

**Contribution 6.3.** *We propose distributed two-dimensional graph-temporal filters to process the joint spectrum of the time-varying graph signal.* These filters are generalizations of the introduced FIR and ARMA graph filters to account for the input signal dynamics. For both filters, we analyze the architectures, the design strategies, and distributed costs. Our results show that these two-dimensional filters can process more than one graph signal in parallel by making them orthogonal in the temporal domain.

### 6.1.2. APPLICATIONS

The main aim of the time-varying analysis for graph filters is to offer guarantees on the consequences that variations in the graph topology and graph signal have on the filter output. This is the case of the tasks listed in Section 3.1.2. As likewise mentioned in [4] an application is to localize information potentials within a dynamic network, such as safe areas in large events, e.g.,

- *Crowd management.* In large concerts, or public manifestations high crowd density can be a threaten in case of an emergency rush. Here, to keep density on safe levels attendees can be provided with a wearable sensor being the vertices of a graph, and through local communications with neighboring (moving) sensors, each node can check the density in its surrounding (e.g., the graph signal can be the number of neighbors). Since, low-pass graph filtering can localize information potentials (e.g., high crowd density) [4], distributed implementations of these filters will allow self-monitoring of density peaks and inform attendees to move towards to safer areas. Differently from the potential kernel, an ARMA$_K$ can achieve sharper filters, so, it can improve the localization accuracy of the information potentials. In addition, as the environment is dynamic we aim at providing filtering solutions robust to topology and crowd density variations.

Nevertheless, due to their ability to perform joint graph-time processing, these filters can be useful for the following applications.

- *Modeling time-varying graph signals.* We aim at modeling time-varying signals on graphs with a two-dimensional graph filter (FIR or ARMA). Thus, by designing few filter coefficients that fit the training signal the best, we can achieve graph signal compression and prediction.

- *Interference cancellation on graphs.* In distributed graph signal denoising over sensor networks the signal of interest may be affected by an interferer working at a different temporal frequency, e.g., self-inference from sensor malfunction, or from an external sensor network working in parallel. In this instance, by exploiting the two-dimensional graph-temporal filters, the interference can be suppressed by nulling its contribution in the temporal domain. Since the filter will now work with time-varying signals, it has the potential to clean more spurious noise in a graph-time fashion which is not possible if the joint processing is ignored.

- *Time-varying signal interpolation.* To improve the energy efficiency in sensor networks, we aim at setting different sensors in steady-state in different time-instants. Then, by means of two-dimensional distributed graph filters, the goal is to interpolate the missing values by receiving local information from neighbors.

Finally, in a more technical aspect graph-time filters are a useful tool to model and generate joint graph-time stationary signals, i.e., graph signals that are WSS over both the graph and the temporal domain [6, 7].

## 6.2. ARMA GRAPH FILTERS AND THEIR INHERENT TEMPORAL PROCESSING

In this section, we extend the parallel $\text{ARMA}_K$ graph filters presented in Chapter 4 to capture the temporal dynamics. We focus on this implementation since it provides the most meaningful results among the three.

### 6.2.1. JOINT GRAPH AND TEMPORAL PROCESSING

The $\text{ARMA}_1$ recursion with a time-varying input $\mathbf{x}_t$ has the form

$$\mathbf{y}_{t+1} = \psi \mathbf{S}\mathbf{y}_t + \varphi \mathbf{x}_t, \tag{6.1}$$

where the subscript $t$ indicates the input time dependency. The dimension of the above recursion can be reduced by restricting the input graph signal to lie in the subspace of an eigenvector $\mathbf{u}$ with associated eigenvalue $\lambda$, i.e., $\mathbf{x}_t = x_t \mathbf{u}$, where now $x_t$ is a scalar and similarly, $\mathbf{y}_0 = y_0 \mathbf{u}$.[1] By orthogonality of the basis, the filter only alters the magnitude $x_t$ relative to the eigenvector $\mathbf{u}$ and not the direction of $\mathbf{x}_t$. So, (6.1) is equivalent to

$$y_{t+1} = \psi \lambda y_t + \varphi x_t \tag{6.2}$$

where $x_t, y_t \in \mathbb{R}$ are the magnitudes of $\mathbf{x}_t, \mathbf{y}_t \in \mathbb{C}^N$ lying on the eigenspace of $\mathbf{u}$, and we can write $\mathbf{y}_t = y_t \mathbf{u}$. By applying the z-transform on both sides, we get the joint graph and temporal frequency response

$$h(z, \lambda) = \frac{\varphi z^{-1}}{1 - \psi \lambda z^{-1}}. \tag{6.3}$$

The temporal-impulse response for each graph frequency $\lambda$ is

$$\mathbf{h}_{t+1}(\lambda) = \left(\varphi (\psi \lambda)^t\right) \mathbf{u}, \tag{6.4}$$

with filter region of convergence (ROC) $\{|z| > |\psi \lambda|, \text{ for all } \lambda\}$ and that the filter is causal.

The joint transfer function characterizes the behavior of $\text{ARMA}_1$ graph filters for an arbitrary yet time-invariant graph: when $z \to 1$, we return to the constant $\mathbf{x}$ result and convergence condition of Theorem 4.1, while for all other $z$ we obtain the standard frequency response and the graph frequency one. As one can see, recursion (6.1) is an

[1] This is a standard way to derive the frequency response of the system.

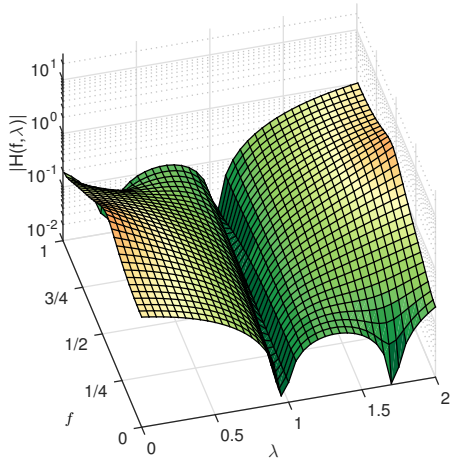



Figure 6.1: The joint graph and temporal frequency response of a parallel graph filter designed to approximate an ideal low pass (step) response with cut-off frequency $\lambda_c = 0.5$ and $K = 3$ w.r.t. the normalized graph Laplacian. The temporal frequencies $f$ are normalized ($\times\pi$ rad/sample).



$\text{ARMA}_1$ filter in the graph domain as well as in the time domain. Observe also that the poles of $h(z,\lambda)$ obey the fixed relationship $z = \lambda\psi$. This yields an interesting insight: the temporal frequency response of the filter differs along each graph frequency $\lambda$, meaning that temporal dynamics affecting signals lying in low graph frequency eigenspaces are dampened to a smaller extent.

As Theorems 6.1 below show, these results are readily generalized to higher order filters.

**Theorem 6.1.** *The joint graph and temporal frequency transfer function of a parallel $\text{ARMA}_K$ is*

$$h(z,\lambda) = \sum_{k=1}^{K} \frac{\varphi^{(k)} z^{-1}}{1 - \psi^{(k)} \lambda z^{-1}}, \tag{6.5}$$

*subject to the conditions of Theorem 4.2 .*

(The proof can be found in Appendix 6.A.)

As in the first order case, Theorem 6.1 describes the behavior of the parallel and periodic implementations. The filer is now an $\text{ARMA}_K$ filter in the graph and temporal domain. In particular, the parallel filter has up to $K$ distinct poles abiding to

$$z = \psi^{(k)} \lambda. \tag{6.6}$$

To give further insight, Figure. 6.1 plots the joint graph and temporal frequency response of the parallel graph filter of third order, designed (only in the graph domain) to approximate an ideal low pass response with cut-off frequency $\lambda_c = 0.5$. In the figure,

the horizontal axis measures the graph frequency with smaller $\lambda$ corresponding to lower variation terms. The temporal axis measures the normalized temporal frequency $f$ such that, for $f = 0$, one obtains the standard graph frequency response.

We conclude the following observation.

**Remark 6.1.** *The ARMA$_K$ graph filter ensures almost the same frequency response as for the static case ($f = 0$) for low temporal variations $f \leq 1/8$. This suggests that these filters are more appropriate for slow temporal variations. While for graph signals lying in eigenspaces with $\lambda$ close to $\lambda = 1$ all temporal frequencies are damped. This is a phenomenon that transcends the filter implementation and the particular filter coefficients. It is attributed to the shifting of the Laplacian $\mathbf{S} = \mathbf{L}_n - \mathbf{I}_N$ in the design phase and to the multiplicative relation of the response poles.*

### 6.2.2. TIME-VARYING GRAPHS AND SIGNALS

Time variations on the graph topology bring new challenges to the graph filtering problem. *First, they make approaches that rely on knowledge of the graph spectrum ineffective.* Approaches which ensure stability by designing the poles to lie outside the set of the Laplacian eigenvalues of a graph, may lead to unstable filters in a different graph where some eigenvalues may over-shoot one of the poles. Due to their different design philosophy, the presented ARMA graph filters handle naturally the aforementioned issues. We can, for instance, think that the different graph realizations among time enjoy an upper bound on the largest eigenvalue $\lambda_{\max}$. In case this is not possible, or difficult to determine, we can always work with the normalized Laplacian and thus take $\lambda_{\max} = 2$. In this way, by designing the filter coefficients to guarantee convergence w.r.t. $\lambda_{\max}$, we impose convergence for all different graph realizations. Furthermore, by designing once the filter coefficients for a continuous range of frequencies, the ARMA recursions also preserve the desired frequency response for different graph realizations.

*The second major challenge is characterizing the graph filter behavior.* Time-varying affine systems are notoriously difficult to analyze when they own no special structure [8]. We devise a new methody for time-varying graph filter analysis. We show that a decomposition basis always exists, over which ARMA$_1$ graph filters (and as a result parallel ARMA$_K$ filters) have the same frequency response as in the static case. This decomposition basis depends only on the sequence of graph realizations.

For a time-varying graph topology, yet with a fixed number of nodes $N$, and a time-varying graph signal, the ARMA$_1$ recursion (4.4) can be written as

$$\mathbf{y}_{t+1} = \psi \mathbf{S}_t \mathbf{y}_t + \varphi \mathbf{x}_t \tag{6.7}$$

where the time-varying graph is shown by indexing $\mathbf{S}_t$ with the subscript $t$. Expanding the recursion we find that, for any sequence of graph realizations $\{\mathcal{G}_0, \mathcal{G}_1, \ldots, \mathcal{G}_t\}$ with corresponding graph shift operators $\{\mathbf{S}_0, \mathbf{S}_1, \ldots, \mathbf{S}_t\}$, the output signal is

$$\mathbf{y}_{t+1} = \psi^{t+1} \mathbf{\Phi}_{\mathbf{S}}(t, 0)\mathbf{y}_0 + \varphi \sum_{\tau=0}^{t} \psi^{\tau} \mathbf{\Phi}_{\mathbf{S}}(t, t-\tau+1)\mathbf{x}_{t-\tau}, \tag{6.8}$$

where $\mathbf{\Phi}_{\mathbf{S}}(t, t') = \mathbf{S}_t \mathbf{S}_{t-1} \ldots \mathbf{S}_{t'}$ for $t \geq t'$, and $\mathbf{\Phi}_{\mathbf{S}}(t, t') = \mathbf{I}_N$ otherwise.

Since the output $\mathbf{y}_t$ depends on the entire sequence of graph realizations, the spectrum of any individual graph shift operator is insufficient to derive the graph frequency of the filter. To extend the spectral analysis to the time-varying setting, we define a joint graph shift operator matrix $\mathbf{S}_{\text{tv}}$ that encompasses all the individual shift operators. The intuition behind our approach is to think of a time-varying graph as one large graph $\mathcal{G}_{\text{tv}}$ that contains all nodes of the graphs $\mathcal{G}_0, \mathcal{G}_1, \ldots, \mathcal{G}_t$, as well as directional edges connecting the nodes at different time steps. We then interpret the spectrum of the shift operator matrix $\mathbf{S}_{\text{tv}}$ as the basis for our time-varying graph Fourier transform. This idea generalizes the joint graph construction introduced in Chapter 5, used to define a Fourier transform for graph signals which change with time. We will construct $\mathcal{G}_{\text{tv}}$ by replicating each node $v_i \in \mathcal{V}$ once for each time step $t$. Denote the $t$th replication of the $i$th node as $v_{t,i}$. For each $t$ and $i$, $\mathcal{G}_{\text{tv}}$ will contain directed edges between $v_{t-1,j}$ and $v_{t,i}$ with $v_j$ being a neighbor of $v_i$ in $\mathcal{G}_{t-1}$. Therefore, here the edges between nodes $v_i$ and $v_j$ are a function of time. By its construction, $\mathcal{G}_{\text{tv}}$ captures not only the topology of the different graphs, but also the temporal relation between them: since the exchange of information between two neighbors incurs a delay of one unit of time, at each time step $t$, a node has access to the values of its neighbors at $t-1$.

To proceed, define $\mathbf{S}_{\text{c}}$ to be the $(t+1)\times(t+1)$ cyclic shift matrix with ones below the diagonal and construct $\mathbf{S}_{\text{tv}}$ as the $N(t+1)\times N(t+1)$ permuted block-diagonal matrix

$$\mathbf{S}_{\text{tv}} = \text{blkdiag}[\mathbf{S}_0, \mathbf{S}_1, \ldots, \mathbf{S}_t](\mathbf{S}_{\text{c}} \otimes \mathbf{I}_N), \tag{6.9}$$



For consistency with the established theory on GFT, when $t=0$ and the graph is time-invariant, we define $\mathbf{S}_{\text{c}} = 1$. Let $\mathbf{e}_\tau$ be the $(t+1)$dimensional canonical unit vector with $[\mathbf{e}_\tau]_i = 1$ if $i=\tau$ and $[\mathbf{e}_\tau]_i = 0$, otherwise. By defining $\mathbf{x}_{0:t} = [\mathbf{x}_0^\mathsf{T}, \mathbf{x}_1^\mathsf{T}, \ldots, \mathbf{x}_t^\mathsf{T}]^\mathsf{T}$ as the vector of dimension $N(t+1)$ which encompasses all input graph signals, we can write

$$\mathbf{\Phi}_{\mathbf{S}}(t, t-\tau+1)\mathbf{x}_{t-\tau} = (\mathbf{e}_{t+1}^\mathsf{T} \otimes \mathbf{I}_N)\mathbf{S}_{\text{tv}}^\tau \mathbf{x}_{0:t}. \tag{6.10}$$

In those cases when the non-symmetric matrix $\mathbf{S}_{\text{tv}}$ enjoys an eigendecomposition, we have $\mathbf{S}_{\text{tv}} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1}$ with $k$th eigenpair $(\lambda_k, \mathbf{u}_k)$. In particular, $\lambda_k$ is the $k$th diagonal element of $\mathbf{\Lambda}$ and $\mathbf{u}_k$ is the $k$th column of $\mathbf{U}$. The total number of eigenpairs of $\mathbf{S}_{\text{tv}}$ is $K = N\times(t+1)$. To ease the notation, we will denote as $\mathbf{u}_k^{-1}$ the respective $k$th column of $\mathbf{U}^{-1}$.

Substituting (6.10) into the second term of (6.8) and rearranging the sums, we get

$$\begin{aligned}
\varphi \sum_{\tau=0}^{t} \psi^\tau \mathbf{\Phi}_{\mathbf{S}}(t, t-\tau+1)\mathbf{x}_{t-\tau} &= \varphi(\mathbf{e}_{t+1}^\mathsf{T} \otimes \mathbf{I}_N) \sum_{\tau=0}^{t} (\psi \mathbf{S}_{\text{tv}})^\tau \mathbf{x}_{0:t} \\
&= \varphi(\mathbf{e}_{t+1}^\mathsf{T} \otimes \mathbf{I}_N) \sum_{\tau=0}^{t} \sum_{k=1}^{K} (\psi\lambda_k)^\tau \mathbf{x}_{0:t}^\mathsf{H} \mathbf{u}_k^{-1} \mathbf{u}_k \\
&= (\mathbf{e}_{t+1}^\mathsf{T} \otimes \mathbf{I}_N) \sum_{k=1}^{K} \varphi \frac{1-(\psi\lambda_k)^{t+1}}{1-\psi\lambda_k} \mathbf{x}_{0:t}^\mathsf{H} \mathbf{u}_k^{-1} \mathbf{u}_k.
\end{aligned} \tag{6.11}$$

Similarly,

$$\psi^{t+1}\mathbf{\Phi}_{\mathbf{S}}(t,0)\mathbf{y}_0 = (\mathbf{e}_{t+1}^{\top}\otimes\mathbf{I}_N)\left(\psi\mathbf{S}_{\text{tv}}\right)^{t+1}(\mathbf{e}_{t+1}\otimes\mathbf{y}_0)$$
$$= (\mathbf{e}_{t+1}^{\top}\otimes\mathbf{I}_N)\sum_{k=1}^{K}(\psi\lambda_k)^{t+1}(\mathbf{e}_{t+1}\otimes\mathbf{y}_0)^{\mathsf{H}}\mathbf{u}_k^{-1}\mathbf{u}_k. \tag{6.12}$$

Without loss of generality, when $t$ is sufficiently large we can ignore terms of the form $(\psi\lambda_k)^{t+1}$ as long as $|\psi\lambda_k|<1$, which also shows that the impact of the filter initialization $\mathbf{y}_0$ on the filter output vanishes with time. This condition is met when $\|\psi\mathbf{S}_{\text{tv}}\|<1$, which as a direct consequence of Gershgorin's circle theorem, this stability condition is met if, for every $\tau$, the sum of the elements of each row of $\mathbf{S}_\tau$, in absolute value, is smaller than $|1/\psi|$ (which also implies that the eigenvalues of $\mathbf{S}_\tau$ are bounded by $|1/\psi|$). For the $\mathbf{S}$ being the normalized (translated) Laplacian this means $|\psi<2|$ ($|\psi<1|$), matching the convergence conditions of the static case. Under this sufficient condition, the filter output approaches

$$\mathbf{y}_{t+1}\approx(\mathbf{e}_{t+1}^{\top}\otimes\mathbf{I}_N)\sum_{k=1}^{K}\left(\frac{\varphi}{1-\psi\lambda_k}\right)\mathbf{x}_{0:t}^{\mathsf{H}}\mathbf{u}_k^{-1}\mathbf{u}_k. \tag{6.13}$$

Notice that the ARMA$_1$ keeps the same graph frequency response as in the time-invariant case (4.5), now expressed in the basis of $\mathbf{S}_{\text{tv}}$. It is not difficult to show that the ARMA$_1$ graph filter converges asymptotically. Let us denote the distance between the filter output at two different time instants $t_1>t_2$ as

$$\epsilon_{t_1,t_2}=\frac{\|\mathbf{z}_{t_1}-\mathbf{z}_{t_2}\|}{x_{\max}}. \tag{6.14}$$

where $x_{\max}=\max_{t=1,\dots,t_1}\|\mathbf{x}_t\|_2$ is an upper bound on the energy of the input. We can now claim

**Theorem 6.2.** *Given the ARMA$_1$ recursion* (6.8) *and given that the graph shift operators are uniformly bounded for every $t$ $\|\mathbf{S}_t\|\le\varrho$, the distance $\epsilon_{t_1,t_2}$ between the filter output at time instants $t_1$ and $t_2$ is upper-bounded as*

$$\epsilon_{t_1,t_2}\le\|\mathbf{y}_0\|\frac{|\psi\varrho|^{t_1}+|\psi\varrho|^{t_2}}{x_{max}}+|\varphi|\frac{|\psi\varrho|^{t_2}-|\psi\varrho|^{t_1}}{1-|\psi\varrho|}. \tag{6.15}$$

(The proof can be found in Appendix 6.B.)

Consider $t_1$ big enough such that the term $|\psi\varrho|^{t_1}\approx 0$. Then, from (6.15) we can find the value of $t_2$ such that the error between the two is smaller than a desired positive constant $\varepsilon$, i.e.,

$$t_2\ge\log(\alpha/\varepsilon)\quad\Rightarrow\quad\epsilon_{t_1,t_2}\le\varepsilon, \tag{6.16}$$

with $\alpha=\|\mathbf{y}_0\|/x_{\max}+\|\varphi\|/(1-\|\psi\varrho\|)$.

These results can be further extended to the general ARMA$_K$ graph filter. For the parallel implementation, we can proceed in the same way as for the ARMA$_1$ by considering that the output signal is the sum of $K$ ARMA$_1$ graph filters.

The main result of Theorem 6.2 stands in the fact that the ARMA output will not diverge as long as the graph shift operators of each realization $\mathcal{G}_t$ has uniformly bounded spectral norm and from (6.16) the distance decreases exponentially. Further, for $t$ big enough and if $\mathbf{S}_{\text{tv}}$ enjoys an eigendecompositon the result in (6.13) gives us insights where the ARMA output converges. Numerical results suggest that the obtained output is close to the designed frequency response of the ARMA filter.

### 6.2.3. NUMERICAL RESULTS

To illustrate our results we simulate two different case-studies: one with a fixed graph and a time-varying graph signal, and one where both the graph and the graph signal are time-varying. In the latter case, the ARMA performance is compared to the state-of-the-art FIR filters designed in a universal manner [9]. With the first case-study, we aim to show how the proposed filters operate on graph signals that have spectral content in both graph and temporal frequency domains. Meanwhile, with the second the goal is to illustrate the ARMA performance when the underlying graph topology is not static anymore but varies with time. For all our simulations, the ARMA filters, if not differently mentioned, are initialized to zero (i.e., $\mathbf{y}_0 = \mathbf{0}_N$ and $\mathbf{y}_0^{(k)} = \mathbf{0}_N$ for all $k$) and the filter design is performed in a universal setting.

### 6.2.4. VARIAITONS ON THE GRAPH SIGNAL



In this section, we present simulation results for time-varying signals. We consider a 0.5-bandlimited graph signal $\mathbf{x}_t^*$ oscillating with a fixed temporal frequency $\pi/10$, meaning that

$$\mathbf{u}_n^{\mathsf{H}}\mathbf{x}_t^* = \begin{cases} e^{J\pi t/10} & \text{if } \lambda_n < 0.5 \\ 0 & \text{otherwise,} \end{cases} \tag{6.17}$$

where $\lambda_n$ is the $n$th eigenvalue of the normalized graph Laplacian and $t$ it the time index. The signal is corrupted with a second interfering signal $\mathbf{i}_t$, oscillating with a temporal frequency $9\pi/10$ with graph spectrum defined in the following in two different ways.. In addition, the signal at each node is corrupted with i.i.d. Gaussian noise $\mathbf{w}_t$, with zero mean and variance $\sigma^2 = 0.1$. We then attempt to recover $\mathbf{x}_t^*$ by filtering it with a parallel ARMA$_5$ graph filter canceling the interference $\mathbf{i}_t$ and attenuating the out of band noise. The ARMA filter is designed only in the graph frequency domain based on the GFT of $\mathbf{x}_t^*$, i.e., to approximate an ideal low-pass filter in the graph domain with cut-off frequency $\lambda_c = 0.5$. Regarding the temporal part, we exploit the property of the filter to preserve the same graph frequency response as the static case for low temporal oscillations, while attenuating the contribution of high temporal frequencies. Our simulations were conducted using a random geometric graph $\mathcal{G}$ composed of $N = 100$ nodes placed randomly in a square area, with any two nodes being connected if they are closer than 15% of the maximum distance in the area.

Depending whether or not the interference is correlated with the signal, we distinguish between two scenarios:

**i) Correlated signal interference.** In this scenario, the interference is self-induced, meaning that at an instant $t$, $\mathbf{i}_t$ and $\mathbf{x}_t^*$ share the same graph spectrum, but oscillating

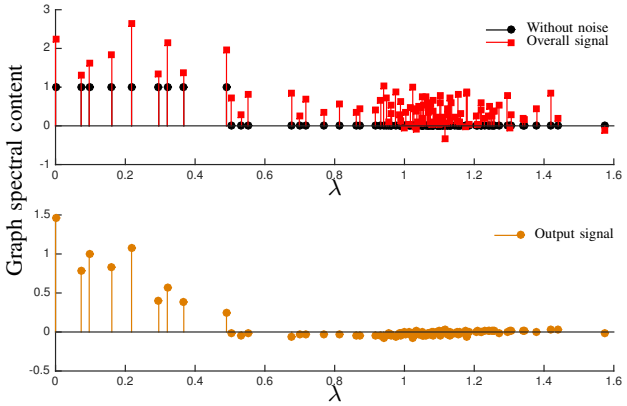



Figure 6.2: Graph spectral content of the input signal as well as of the overall signal affected by interference and noise *a*) (top), and of the filter output signal *b*) (bottom). The output signal graph spectrum is shown for $t = 100$.

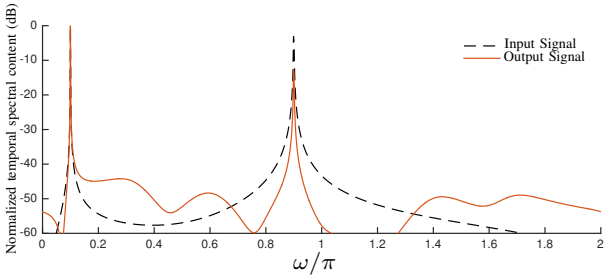



Figure 6.3: Average time spectral content over all nodes of the input and output signal. The values are normalized with respect to the maximum.

at a higher temporal frequency (due for instance to electronics problems). To give intuition, in Figure. 6.2.a), we show the graph spectral content of $\mathbf{x}_0^*$ and $\mathbf{x}_0^* + \mathbf{i}_0 + \mathbf{w}_0$. We can see that once corrupted by noise and interference, the graph signal presents significant spectral content across the graph spectrum, thus losing its bandlimited nature. Meanwhile, Figure. 6.2 b) depicts the real part of the graph spectral content of the filter output after 100 iterations (i.e., well after the initial state is forgotten). Even though the figure cannot capture the effect of dynamics (as it focuses on $t = 100$), it shows that all frequencies above $\lambda_c = 0.5$ have been attenuated and the interference contribution in the band is reduced.

To illustrate the filtering of the temporal frequencies of the signal, in Figure. 6.3 we show the average spectrum over all nodes of the input and output signal. To increase visibility, the values in the figure are normalized with respect to the maximum. We note that the content relative to the interfering frequency $9\pi/10$ of the output signal is attenuated around 13 dB with respect to the main temporal frequency content of $\pi/10$.

**ii) Uncorrelated signal interference.** Let us now consider a more involved scenario,

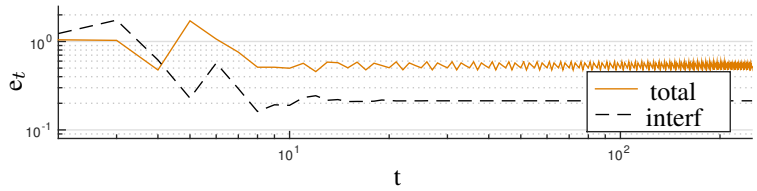



Figure 6.4: Error of the ARMA recursion when the time-vaying input signal is affected by uncorrelated interference.

in which the interfering graph signal satisfies

$$\mathbf{u}_n^{\mathsf{H}}\mathbf{i}_t = e^{J9\pi t/10}e^{-\lambda_n}, \tag{6.18}$$

i.e., it is a signal having a heat kernel-like graph spectrum oscillating in time with a pulsation $\omega = 9\pi/10$. We will examine two types of errors: i) The first compares for each time $t$ the ARMA$_5$ output GFT $\hat{\mathbf{y}}_t$ to that of the signal of interest

$$e_t^{(total)} = \frac{\|\hat{\mathbf{y}}_t - \hat{\mathbf{x}}_t^*\|}{\|\hat{\mathbf{x}}_t^*\|}. \tag{6.19}$$

Achieving a small error $e_t^{(total)}$ is a very challenging problem since an algorithm has to simultaneously overcome the addition of noise and the interference, while operating in a time-varying setting (see Figure. 6.4. ii) The second error focuses on interference and compares $\mathbf{y}_t$ to the output $\mathbf{y}_t^*$ of the same ARMA$_5$ operating on $\mathbf{x}_t^*+\mathbf{w}_t$ (but not $\mathbf{x}_t^*+\mathbf{i}_t+\mathbf{w}_t$)

$$\mathbf{e}_t^{(interf)} = \frac{\|\hat{\mathbf{y}}_t - \hat{\mathbf{y}}_t^*\|}{\|\hat{\mathbf{y}}_t^*\|}, \tag{6.20}$$

where $\hat{\mathbf{y}}_t^*$ is the GFT of $\mathbf{y}_t^*$.

We can see from Figure. 6.4 that after a few iterations this error becomes relatively small, meaning that the ARMA output spectrum when the signal is affected by interference is similar to when the interference-less signal is used. This gives a first insight, that using the ARMA recursion we can manage multiple signals on a graph by making them orthogonal in the temporal frequency domain. By a specific design of the filter coefficients, one can distributively operate on the graph signal of interest and ignore the others. Such a result cannot be achieved with FIR filters for two reasons: i) they suffer from handling time-varying input signals, and ii) the shown FIR filters do not act on the temporal frequency content of the graph signals, thus such a distinction between overlapping signals is difficult to achieve. Next, in Section 6.3.1 we will introduce two-dimensional FIR graph-temporal filters capable to perform this task.

The above results illustrate the conclusions of Section 6.2.1, and also quantify how much we can attenuate the signal at a specific graph/temporal frequency.

### 6.2.5. VARIATIONS ON THE GRAPH TOPOLOGY

We examine the influence of graph variations for two filtering objectives. The first, which corresponds to Tikhonov denoising, can be computed *exactly* using ARMA. In the second

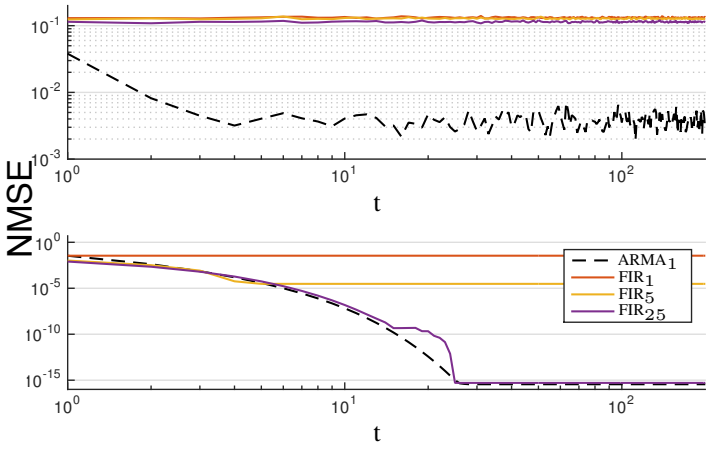



Figure 6.5: Normalized error related to the solution of the denoising problem in a distributed way with graph filters. Results relative to random time-varying graph (top) and static graph (bottom). We compare the results of $\text{ARMA}_1$ with different FIR graph filters. The $\text{FIR}_K$ output at time $t$ is calculated as $\mathbf{y}_t = \sum_{k=0}^{K} h_k \mathbf{\Phi}_{\mathbf{S}}(t, t-k+1)\mathbf{x}$ and is not arrested after $K$ time instants.

6

objective, the graph filter is designed to *approximate* an ideal low-pass graph filter, i.e., a filter that eliminates all graph frequency components higher than some specific $\lambda_c$. In addition, we consider two different graph dynamics: *random edge failures*, where the edges of a graph disappear at each iteration with a fixed probability, and the standard model of *random waypoint* mobility [10]. The above setup allows us to test and compare universal ARMA and FIR graph filters (designed using the least-squares method) over a range of scenarios, each having different characteristics.

**Exact design (denoising).** We simulate the denoising problem (as defined by (4.15), with $w = 0.5$ and $K = 1$) over the same graph topology of Section 6.2.4, where the probability of an edge going down at each time instant is $p = 0.05$. The input signal $\mathbf{x} = \mathbf{x}_0 + \mathbf{w}$ is given by a linear combination of a smooth signal $\mathbf{x}_0$ and noise $\mathbf{w}$. To ensure that the graph signal is smooth, we set its spectrum, w.r.t. the first graph, as $\mathbf{u}_n^{\mathsf{H}}\mathbf{x}_0 = e^{-5\lambda_n}$. The noise $\mathbf{w}$ is i.i.d. Gaussian distributed with zero mean and unit variance.

To compare the results, we calculate the normalized error between the graph filter output and the analytical solution of the optimization problem (4.15) solved w.r.t. the initial graph. In Figure. 6.5, we plot the normalized error of solving the denoising problem via distributed graph filtering. We consider an $\text{ARMA}_1$ graph filter (designed according to Section 4.2.4 with $\mathbf{y}_0 = \mathbf{x}_0$) and we compare its performance with FIR graph filters of different orders.

As expected, we can see that in the static case the ARMA graph after $K$ iterations has the same performance as the $\text{FIR}_K$ filter and they both match the optimal solution. In the random time-varying graph the ARMA filter outperforms all the FIRs. This is mainly due to its implementation strategy, which allows the ARMAs to handle the graph variations better. Also, note that the result got from the $\text{ARMA}_1$ in the time-varying scenario

quantifies the theoretical derivations in (6.13) and Theorem 6.2. Indeed, we notice that the output is close (up to an order $10^{-3}$) to the desired frequency response and the convergence is linear.

We can see that, for both the random time-varying and static graph the ARMA graph filter gives a lower error with respect to the solution of the optimization problem. As we have seen before, for static graphs the ARMA filter matches correctly the analytical solution. Meanwhile, when the graph is generated randomly it approximates quite well the latter. On the other hand, the FIR filters performance is limited because they only approximate the optimal solution. Notice that the FIR output is given after $K$ time instants and then the filter is reset, hence the spikes in the figure.

**Approximate design (ideal low pass).** We use graph filters of increasing orders, $K = 2, 4$ and 6, to universally approximate a low-pass graph filter with frequency response $h^*(\lambda) = 1$ if $\lambda < 0.5$, and zero otherwise. The graph has 100 nodes living in a square of 1000 × 1000 meters, with a communication range of 180 meters. We simulated node mobility according to the random waypoint model [10] with a constant speed selected in $[0, 3]$ $m/s$.

We start with a scenario where only the graph topology changes in time while the graph signal remains invariant. Later, we simulate a more general case, where both the graph topology and the graph signal are time-varying. For both scenarios, we do 20 distinct runs, each lasting 10 minutes and comprising 600 iterations (one iteration per second). We compare the response error $\|\mathbf{h} - \mathbf{h}^*\|_2 / \|\mathbf{h}^*\|_2$ of the ARMA filters with that of the analogous FIR filters while accounting for the initialization phase (we ignore the first 100 iterations). At each time instant, we compute $h(\lambda_n) = \hat{y}_n / \hat{x}_n$, where the points $\hat{x}_n \approx 0$ are not considered. Then, it is compared with the desired frequency response at the particular graph frequency $\lambda_n$, i.e., $h^*(\lambda_n)$. The statistical significance of our results stems not only by the 20 distinct repetitions, but also by the large number of graph topologies experienced in each run.

*Time-varying graph, constant graph signal.* For this scenario, $\mathbf{x}$ is a random vector with entries selected uniformly distributed in [0, 1]. In Figure. 6.6 (top) we show the response error for increasingly higher node speeds. As expected, the error increases with speed. The ARMA filters show a better performance compared to their analogous FIR filters. This indicates that the proposed approach handles better time-varying settings than the FIR filters. Further, we can see that higher order ARMA filters approximate better the desired frequency response (smaller error) when the graph is static. On the other hand, when mobility is present, higher order ARMA recursions lead to a rough approximation due to their slower convergence and the fact that the poles go closer to the unit circle (larger coefficients).

*Time-varying graph and graph signal.* To conclude, we simulate the more general case where both the graph structure and the graph signal change. Simulating a target tracking scenario, we let the signal at each node take a value of zero, unless a node was within 100 meters from a target point, residing at the middle of the 1000 × 1000 meter simulation area, in which case the node's value was set to one. In Figure. 6.6 (bottom) we show the response error as a function of the node's speed. It is not surprising that letting the graph signal change over time makes the graph filtering problem harder and the corresponding errors of all graph filters larger. As expected, the error increases with

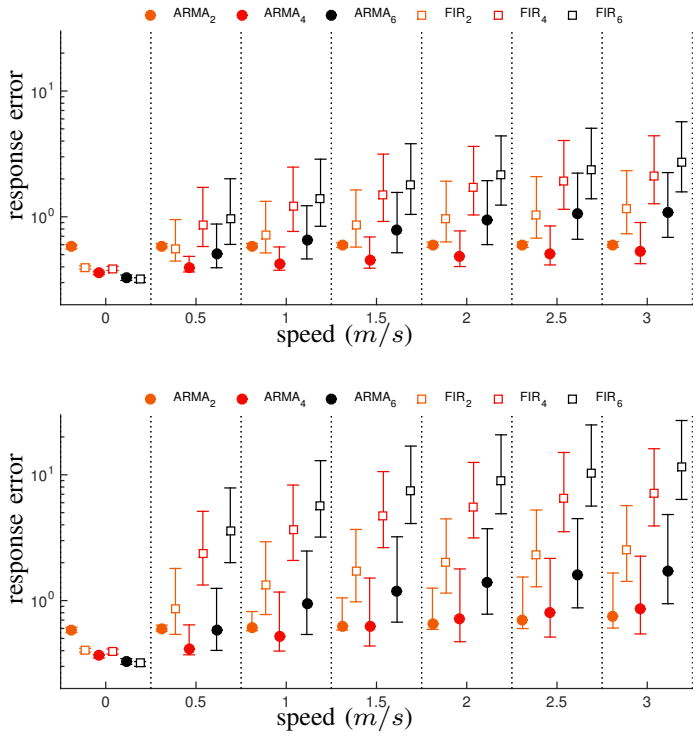



Figure 6.6: The effects of the variations only on the graph topology (top) and on both graph and graph signal (bottom). The response error is calculated as $\|h(\lambda) - h^*(\lambda)\|_2 / \|h^*(\lambda)\|_2$. Each error bar shows the standard deviation of the approximation error over 20 runs. A small horizontal offset is included to improve visibility.

speed. The ARMA filters show a better performance compared to their analogous FIR filters for all cases other than when $K = 2$ and speed is zero (the latter is an artifact of the Shank's method).

## 6.3. DISTRIBUTED TWO-DIMENSIONAL GRAPH-TIME FILTERS

So far, we treated graph signal variations as perturbations in the input signal and we wanted to characterize the filter robustness in such conditions. However, we saw that to some extent *the graph-based designed ARMA filter* showed some filtering behavior also in the temporal domain. To properly process the joint graph-time spectrum of time-varying graph signals, in this section we extend our analysis to more involved two-dimensional FIR and ARMA filters. The proposed filters can process a time-varying graph signal in a distributed manner, and by tuning the filter coefficients they can approximate any desired two-dimensional graph frequency response.

### 6.3.1. FIR GRAPH-TEMPORAL FILTERS

In this section, we present the recursions that implement a two-dimensional FIR (FIR$^{2D}$) filter.

**FIR$^{2D}$ (intuitive extension).** Temporal variations of the input signal can be captured by the FIR filter considering its temporal history. Let us consider extending (3.6) to the recursion

$$\mathbf{y}_t = \sum_{k=0}^{K} \phi_k \mathbf{S}^k \mathbf{x}_{t-k}, \tag{6.21}$$

where now the output at time $t \geq K$ , i.e., $\mathbf{y}_t$, depends on the past $K$ realizations of the input signal, where $\mathbf{x}_{t-k}$ is graph-shifted with $\mathbf{S}^k$ (this favors a distributed implementation). Recursion (6.21) provides the intuition it is an FIR$_K$ filter in both the graph and temporal frequency domain. To see this, we calculate the joint transfer function of the filter (6.21). Applying first the GFT and then the $Z$-transform to (6.21), the joint graph-temporal transfer function can be written as

$$h(z,\lambda) = \sum_{k=0}^{K} \phi_k \lambda^k z^{-k}. \tag{6.22}$$

We can now formally see that, the joint transfer function (6.22) implements an FIR filter of order $K$ in the graph domain and an FIR filter of the same order in the temporal domain. In a distributed computation, for computing $\mathbf{y}_t$ we need access to the terms $\mathbf{S}\mathbf{x}_{t-1}, \mathbf{S}^2\mathbf{x}_{t-2}, \ldots, \mathbf{S}^K\mathbf{x}_{t-K}$. To reduce the computation effort, we can consider that each node memorizes the terms $\mathbf{x}_{t-1}, \mathbf{S}\mathbf{x}_{t-2}, \ldots, \mathbf{S}^K\mathbf{x}_{t-K-1}$ while computing $\mathbf{y}_{t-1}$. In this way, each node can compute $\mathbf{S}^k\mathbf{x}_{t-k}$ from $\mathbf{S}^{k-1}\mathbf{x}_{t-k}$, which leads to the same computational effort as computing (3.6). From (6.22), we note that the zeros of the polynomial in $\lambda$ and in $z$ are correlated to each other[2]. This affects the joint design, and we can expect the same behavior as the ARMA filter, i.e., there is a tradeoff between the filter approximation in each domain.

We further illustrate this in Figure 6.7, where we approximate with an FIR$_3$ filter an ideal step function in the graph frequency domain with cut-off frequency $\lambda_c = 0.5$. We can see that for a high normalized temporal frequency the filter response differs from the case of $f = 0$, for which the filter has been designed. This behavior is attributed to the fact that the joint transfer function (6.22) is not a complete two-dimensional polynomial of order $K$. All the cross term monomials of the form $\lambda^\alpha z^{-\beta}$ with $\alpha \neq \beta$ are missing. However, even considering all these challenges, this approach can still approximate specific two-dimensional filter masks or to characterize the FIR filter robustness to perturbations of the input signal as we did for the ARMA filter in Section 6.2.

**General FIR$^{2D}$.** The approximation accuracy of the FIR$^{2D}$ filter can be improved if we incorporate also the missing cross term monomials in (6.21). This is achieved by considering all $K$ graph-shifts for every past input of the graph signal. This approach considers more data exchanges and computational power to implement a filter of the same order

[2] In the before introduced ARMA filter this is observed for the poles (cf. eq. (6.6)), which affects both the filter design and stability.

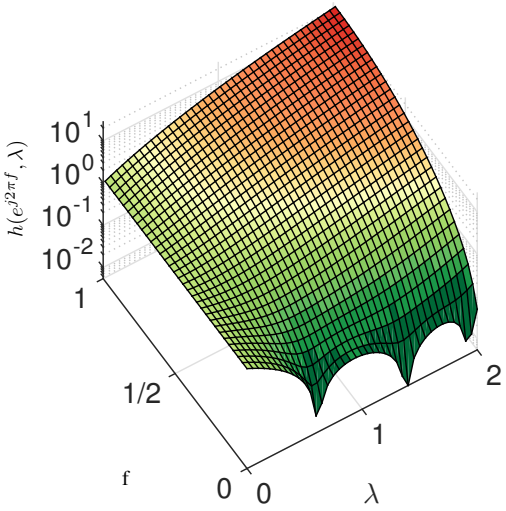



Figure 6.7: The joint graph and temporal frequency response of the $\text{FIR}_3$ graph filter, designed to approximate an ideal low pass (step) response with cut-off frequency $\lambda_c = 0.5$. A normalized Laplacian has been used, $\lambda \in [0,2]$ and the temporal normalized frequency $f \in [0,1]$.



in both the graph and temporal domain. More formally, consider the recursion

$$\mathbf{y}_t = \sum_{k=0}^{K_\text{g}} \sum_{l=0}^{K_\text{t}} \phi_{k,l} \mathbf{S}^k \mathbf{x}_{t-l}, \tag{6.23}$$

with $K_\text{g}$ and $K_\text{t}$ the memory of the filter in the graph and temporal domain, respectively. We can now calculate the joint transfer function of (6.23) in the same way as we did for (6.21). By applying the GFT and the $Z$-transform we get

$$h(z,\lambda) = \sum_{k=0}^{K_\text{g}} \sum_{l=0}^{K_\text{t}} \phi_{k,l} \lambda^k z^{-l}, \tag{6.24}$$

which is now an FIR of order $K_\text{g}$ in the graph domain and of order $K_\text{t}$, not necessarily equal to $K_\text{g}$, in the temporal domain. From the joint frequency response (6.24), we can see that now we have a full polynomial in the variables $z$ and $\lambda$. Thus with this expression we can act on all the $K_\text{g}K_\text{t}$ coefficients $\phi_{k,l}$, instead of the $K$ offered by (6.21), to approximate a two-dimensional frequency response. Further, regarding (6.21), recursion (6.23) has the potential to achieve filters of different orders in each domain. Similar to the distribution implementation of (6.21), the computational efforts of computing the output $\mathbf{y}_t$ can be reduced by allowing the nodes to keep in memory the terms $\mathbf{S}^k \mathbf{x}_{t-l}$ for $k \in [0, K_g]$ and $l \in [1, K_t]$ obtained while computing $\mathbf{y}_{t-1}$. Thus, for the computation of $\mathbf{y}_t$ only the terms $\mathbf{S}\mathbf{x}_t, \ldots, \mathbf{S}^{K_g}\mathbf{x}_t$ have to be computed. For a $\text{FIR}^{2\text{D}}$ filter of the form (6.23) with order $K$ in both graph and time, this means we need only $K$ times more computational efforts compared to (3.6).

**Graph restricted causal FIR$^{2D}$.** Given the general form (6.23) and its particular version (6.21) there is room to use an intermediary approach, which can be implemented in a causal way with relaxed implementation costs. The causal FIR$^{2D}$ with most degrees of freedom can be implemented as

$$\mathbf{y}_t = \sum_{l=0}^{K_t} \sum_{k=0}^{l} \phi_{k,l} \mathbf{S}^k \mathbf{x}_{t-l}. \tag{6.25}$$

Being a restricted causal implementation, the terms $\mathbf{S}\mathbf{x}_t, \ldots, \mathbf{S}^{K_g}\mathbf{x}_t$ are not anymore necessary to be computed in (6.25), thus the distributed implementation costs are the same as (3.6).

**Separable FIR$^{2D}$.** A particular subclass of interest of (6.23), are two-dimensional filters that achieve a *separable* two-dimensional frequency response in graph and time. By setting $\phi_{k,l} = \phi_k b_l$ we can express (6.23) as

$$\mathbf{y}_t = \left( \sum_{k=0}^{K_g} \phi_k \mathbf{S}^k \right) \left( \sum_{l=0}^{K_t} b_l \mathbf{x}_{t-l} \right), \tag{6.26}$$

where $\phi_k$ are the filter coefficients relative to the graph part and $b_l$ to time. Similar to the derivation of (6.23), the transfer function of (6.26) can be written as $h(z,\lambda) = h_t(z)h_g(\lambda)$ where $h_t(z) = \sum_{l=0}^{K_t} b_l z^{-l}$ and $h_g(\lambda) = \sum_{k=0}^{K_g} \phi_k \lambda^k$. This separable approach offers us the freedom to handle the filter specifications independently in the graph and temporal domain. We can also see (6.26) as first computing locally at each node the temporal filtering, i.e., $\mathbf{y}'_t = \sum_{l=0}^{K_t} b_l \mathbf{x}_{t-l}$ and then performing the overall output $\mathbf{y}_t$ by filtering $\mathbf{y}'_t$ on the graph, i.e., $\mathbf{y}_t = \sum_{k=0}^{K_g} \phi_k \mathbf{S}^k \mathbf{y}'_t$. In contrast to the latter, (6.26) allows for an online processing of the time-varying graph signal. In the sense that, when $\mathbf{x}_{t+1}$ becomes available, (6.26) requires processing only this signal among the graph and not calculating locally $\mathbf{y}'_{t+1}$ and then perform a graph filter. This means we can track the time variations of the graph signal. However, notice that the separable filters have only $K_g + K_t$ degrees of freedom instead of $K_g K_t$ of the general case (6.23). Thus it can address a limited class of two-dimensional frequency responses, but a practical class.

**Filter design.** We now discuss the task in designing the filter coefficients give a two-dimensional frequency mask $h^*(e^{J\omega}, \lambda)$. We consider the following cases:

*General design.* By using the general two-dimensional frequency response (6.24), the filter coefficients can be designed with a two-dimensional polynomial fitting of $h(z,\lambda)$ (for $z = e^{J\omega}$) to $h^*(e^{J\omega}, \lambda)$.[3]

*Design by signal modeling.* When we are interested to model a temporal stream of graph signals $\{\mathbf{x}_0, \ldots, \mathbf{x}_T\}$ with few coefficients, we might consider fitting one of the FIR$^{2D}$ implementations to the data and the filter coefficients can be found by solving

$$\min_{\phi_{k,l}} \sum_{\tau=0}^{T} \left\| \mathbf{x}_{\tau+1} - \sum_{k=0}^{K_g} \sum_{l=0}^{K_t} \phi_{k,l} \mathbf{S}^k \mathbf{x}_{\tau-l} \right\|_2^2, \tag{6.27}$$

[3] In practice, we have observed that a direct two-dimensional fit does not lead to good approximations. As a deeper analysis is needed, we consider this aspect as future research in Section 9.2.2.

where in (6.28) we consider the most general FIR$^{2D}$ (6.23). As used in [11] for graph topology inference from streaming data, and as as we will see in Section 6.3.2, the approach in (6.28) can also be interpreted as a graph-temporal autoregressive modeling as $\mathbf{y}_t$ is substituted with $\mathbf{x}_{t+1}$.

An alternative to the design (6.28) is to inject unitary white noise $\mathbf{w}_t$ into the filter and exploit statistics of the streaming data to design the filter coefficients as the solution of

$$\min_{\phi_{k,l}} \mathbb{E} \left\| \mathbf{x}_t - \sum_{k=0}^{K_g} \sum_{l=0}^{K_t} \phi_{k,l} \mathbf{S}^k \mathbf{w}_{t-l} \right\|_2^2, \tag{6.28}$$

with $\mathbf{x}_t$ abiding to some known distribution on the graph.

*Separable design.* When the desired frequency response is separable, i.e., $h^*(e^{j\omega}, \lambda) = h_t^*(e^{j\omega}) h_g^*(\lambda)$ we have the advantage to separate the filter design as well. Thus, we can use any desired method to find the coefficients $\phi_k$ that approximate $h_g^*(\lambda)$ and any of the well-known techniques to find the coefficients $b_l$ for approximating $h_t^*(e^{j\omega})$. The latter renders the separable approach practical since we can give our specifications independently in the graph and temporal domain.

### 6.3.2. ARMA GRAPH-TEMPORAL FILTERS

We can extend the general FIR$^{2D}$ (6.23) by adding a feedback loop on the filter to obtain the general two-dimensional ARMA (ARMA$^{2D}$) graph-temporal recursion defined as

$$\sum_{p=0}^{P} \sum_{l=0}^{L_p} \psi_{l,p} \mathbf{S}^l \mathbf{y}_{t-p} = \sum_{q=0}^{Q} \sum_{k=0}^{K_q} \varphi_{k,q} \mathbf{S}^k \mathbf{x}_{t-q}, \tag{6.29}$$

for some autoregressive coefficients $\psi_{l,p}$ and moving average coefficients $\varphi_{k,q}$. Scalars $P$, $L_p$, $Q$ and $K_q$ denote the filter orders. Recursion (6.29) extends the ARMA graph filters in Chapter 4, as now the computation of the output $\mathbf{y}_t$ involves shifts of different orders in the graph domain (indicated by the powers of $\mathbf{S}$), and in the time domain (indicated by the temporal memory of the past output and input signal). These joint shifts once again show that we can jointly capture signal variations over the graph and time. On the negative side, ARMA$^{2D}$ introduces stability issues which once again relate filter instabilities in both the graph and temporal domain.

The following proposition asserts the above intuition.

**Proposition 6.1.** *Given a set of stable coefficients $\psi_{l,p}$, $\varphi_{k,q}$ (see proof for the exact conditions), the joint transfer function of* (6.29) *is*

$$h(z, \lambda) = \frac{\sum_{q=0}^{Q} \sum_{k=0}^{K_q} \varphi_{k,q} \lambda^k z^{-q}}{\sum_{p=0}^{P} \sum_{l=0}^{L_p} \psi_{l,p} \lambda^l z^{-p}}. \tag{6.30}$$

(The proof can be found in Appendix 6.C.)

Recursion (6.29) implements ARMA($P$, $Q$) in the time domain and an ARMA$_{L_p,K_q}$ in the graph domain for different temporal shifts $p$ and $q$. The ARMA$^{2D}$ filters thus have the potential to achieve a better approximation quality of a two-dimensional desired

response, as compared to the ARMA graph filters in Section 6.2. Notice that (6.29) is a special form of implementing two-dimensional filters, thus it will not collapse to a pure ARMA graph filter when the graph signal is time-invariant. This is because $\mathbf{y}_t$ will appear on both sides of (6.29).

By tuning the coefficients $\psi_{l,p}$, $\varphi_{k,q}$ and the shift orders $(P,Q)$, $(L_p,K_q)$, (6.29) specializes to the following forms:

- For $\mathbf{P}_0=\sum_{l=0}^{L_0}\psi_{l,0}\mathbf{S}^l$ being non singular, we get the causal ARMA$^{2D}$ form

$$\mathbf{P}_0\mathbf{y}_t=-\sum_{p=1}^{P}\sum_{l=0}^{L_p}\psi_{l,p}\mathbf{S}^l\mathbf{y}_{t-p}+\sum_{q=0}^{Q}\sum_{k=0}^{K_q}\varphi_{k,q}\mathbf{S}^k\mathbf{x}_{t-q}, \tag{6.31}$$

  which is a predictor recursion on graph, since $\mathbf{y}_t$ depends only on the past filter input and output signals. To avoid singularity issues of $\mathbf{P}_0$, we can set $\mathbf{P}_0=\mathbf{I}_N$.

- For $L_p=p$ and $K_q=q$, we get the graph restricted causal ARMA$^{2D}$ form

$$\psi_{0,0}\mathbf{y}_t=-\sum_{p=1}^{P}\sum_{l=0}^{p}\psi_{l,p}\mathbf{S}^l\mathbf{y}_{t-p}+\sum_{q=0}^{Q}\sum_{k=0}^{q}\varphi_{k,q}\mathbf{S}^k\mathbf{x}_{t-q}, \tag{6.32}$$

  which considers the output at time $t$ as a linear combination of the neighbors distant at most as the temporal history of the input-output signals (i.e., like (6.25)).

- For $Q=K_q=0$, (6.29) reduces to a two-dimensional AR filter on graphs.

- For $\mathbf{P}_0=\mathbf{I}_N$ and $\mathbf{P}_\tau=\sum_{l=0}^{L_\tau}\psi_{l,0}\mathbf{S}^l=\mathbf{0}_N\mathbf{0}_N^\mathsf{T}$ for all $\tau=1,\ldots,P$, the ARMA$^{2D}$ reduces to the FIR$^{2D}$ filter on graphs.

- For $\psi_{l,p}=\psi_l a_p$ and $\varphi_{k,q}=\varphi_k b_q$, recursion (6.29) implements a separable ARMA$^{2D}$ with frequency response $h(z,\lambda)=h_t(z)h_g(\lambda)$, where $h_t(z)$ and $h_g(\lambda)$ are respectively the rational frequency responses in each domain.

- Finally, (6.29) allows us to achieve hybrid forms, i.e., FIR in one domain, say graph, and IIR in the other domain, say time. The latter example can be achieved by setting $L_p=0$ for all $p=0,\ldots,P$ and $\psi_{0,0}=1$, which leads to the recursion

$$\mathbf{y}_t=-\sum_{p=1}^{P}\mathbf{y}_{t-p}+\sum_{q=0}^{Q}\sum_{k=0}^{q}\varphi_{k,q}\mathbf{S}^k\mathbf{x}_{t-q}. \tag{6.33}$$

  Then, under the conditions of Proposition 6.1, (6.33) achieves the two-dimensional frequency response

$$h(z,\lambda)=\frac{\sum_{q=0}^{Q}\sum_{k=0}^{K_q}\varphi_{k,q}\lambda^k z^{-q}}{\sum_{p=0}^{P}\psi_{l,0}z^{-p}}. \tag{6.34}$$

  The main advantage of (6.34) is that it achieves a two-dimensional graph-time frequency response with an alleviated stability that depends only on the temporal domain (i.e., poles within the unit circle).

**Distributed computation.** We consider the standard message-passing model [12] where the input graph is the same as the network over which the computation is performed. The message exchange is assumed to take much short than the sampling period of the signal. We quantify the efficiency of the filtering in terms of the *per-timestep communication complexity*, defined as the number of bits the network needs to exchange to compute $\mathbf{y}_t$ given $\mathbf{y}_{t-1},\dots,\mathbf{y}_0$. Let us focus on the special case that $\mathbf{P}_0 = \mathbf{I}_N$, for which recursion (6.29) is efficiently computable. Additionally, consider $L_p = L$ and $K_q = K$ for all $p,q$. The recursion involves the terms $\mathbf{S}^l\mathbf{y}_{t-p}$ and $\mathbf{S}^k\mathbf{x}_{t-q}$ for all $k,l,p,q>0$, but, only the terms $\mathbf{S}^1\mathbf{y}_{t-1},\dots,\mathbf{S}^l\mathbf{y}_{t-1}$ and $\mathbf{S}^1\mathbf{x}_t,\dots,\mathbf{S}^k\mathbf{x}_t$ have not been computed during a previous timestep. Since in both cases we are dealing with successive powers of $\mathbf{S}$, we can reduce the computation effort by obtaining $\mathbf{S}^l\mathbf{y}_{t-1}$ from $\mathbf{S}^{l-1}\mathbf{y}_{t-1}$ and so on (the same holds for $\mathbf{S}^k\mathbf{x}_t$). Thus, we will need $K+L$ multiplications with matrix $\mathbf{S}$. Since $\mathbf{S}$ is a local matrix, the network can perform the multiplication of $\mathbf{S}$ with any graph signal distributedly, by exchanging $2M$ values. The per-timestep communication complexity is $2M(K+L)\times c_r = O(MK)$ bits, where $c_r$ is the architecture-dependent representation length of each scalar and, w.l.o.g., we assume that $K \geq L$. The per-timestep computational complexity of ARMA$^{2D}$ is less than twice that of ARMA$_K$ graph filter (which is $2MK\times c_r$ bits per-timestep) in the general case, and equivalent when $L=0$.

**Filter design.** We now discuss the challenges that ARMA$^{2D}$ introduces in the design phase in meeting given filtering specifications.

*General design.* As in the FIR$^{2D}$, we would like to approximate a desired two-dimensional desired response $h^*(e^{j\omega},\lambda)$ with the two-dimensional frequency response (6.30) for $z = e^{j\omega}$. We remark that finding the filter coefficients $\psi_{l,p}$ and $\varphi_{k,q}$ that minimize the approximation error results in a challenging task since the fractional nature of (6.30) leads to a set of non-linear equations. The added convergence/stability issues that the ARMA$^{2D}$ brings into the play make this aspect more challenging. While we leave a detailed analysis of this aspect for future research, the hybrid form (6.33) is also a valid candidate with alleviated convergence/stability issues.

*Design by signal modeling.* By following the same approach as the FIR$^{2D}$ modeling, the ARMA$^{2D}$ coefficients that model a stream of time-varying graph signals $\{\mathbf{x}_0,\dots,\mathbf{x}_T\}$ can be found by minimizing

$$\min_{\psi_{l,p},\varphi_{k,q}} \sum_{\tau=0}^{T} \left\| \mathbf{x}_{\tau+1} + \sum_{p=1}^{P}\sum_{l=0}^{L_p} \psi_{l,p}\mathbf{S}^l\mathbf{x}_{\tau-p} - \sum_{q=0}^{Q}\sum_{k=0}^{K_q} \varphi_{k,q}\mathbf{S}^k\boldsymbol{\delta}_{\tau-q} \right\|_2^2, \tag{6.35}$$

with $\boldsymbol{\delta}_\tau$ being a know deterministic driving graph signal (not necessarily Dirac impulse). For known statistics of $\mathbf{x}_t$ the ARMA$^{2D}$ filter can be driven by white noise and the filter coefficients, similarly to (6.28), can be found as the solution of

$$\min_{\psi_{l,p},\varphi_{k,q}} \mathbb{E}\left\| \mathbf{x}_{\tau+1} + \sum_{p=1}^{P}\sum_{l=0}^{L_p} \psi_{l,p}\mathbf{S}^l\mathbf{x}_{\tau-p} - \sum_{q=0}^{Q}\sum_{k=0}^{K_q} \varphi_{k,q}\mathbf{S}^k\mathbf{w}_{\tau-q} \right\|_2^2, \tag{6.36}$$

for some unitary zero-mean white noise $\mathbf{w}_t$. In (6.35) and(6.36), we implicitly considered the causal form (6.31) with $\mathbf{P}_0 = \mathbf{I}_N$ due to its simplistic implementation, but other alternatives are also possible.

*Separable design.* For a desired filter frequency response of the form $h^*(e^{J\omega},\lambda) = h_t^*(e^{J\omega})h_g^*(\lambda)$. The design problem can then be reformulated as finding the respective filters taps that approximate each frequency response independently, i.e., by minimizing

$$\int_{\omega}\left|h_t(e^{J\omega})-h_t^*(e^{J\omega})\right|^2 \mathrm{d}\omega \quad \text{and} \quad \int_{\mu}\left|h_g(\lambda)-h_g^*(\lambda)\right|^2 \mathrm{d}\lambda \tag{6.37}$$

for the temporal and graph approximations, respectively. In this way, we can use any of the techniques shown in the previous chapters to approximate a desired graph frequency response and any of the well-established techniques for temporal filter design [13].

### 6.3.3. NUMERICAL RESULTS

This section tests the FIR$^{2D}$ and ARMA$^{2D}$ filters with some preliminary numerical results. For both architectures, we evaluate the approximation accuracy of the separable approaches and their ability to achieve jointly graph signal denoising and interference suppression. We start our evaluation with the FIR$^{2D}$ and then we focus on the ARMA$^{2D}$ filter. For our simulations we consider a network of $N$ =100 nodes randomly placed in a squared area, with two nodes being neighbors if they are physically closer than 15% of the largest distance in the area.

**FIR$^{2D}$.** We consider FIR filter with orders $K_g = K_t = 10$. For the design in the graph domain, we use the polynomial approximation [9] and the windowing method for the time domain.

*Filter approximation.* With reference to Figure 6.8, we note that the proposed approach can approximate different desired two-dimensional separable frequency responses. For this case, the cut-off frequencies in both domains are the half of the respective bands. We remark that these results extend those of the ARMA$_K$ in Figure 6.6 to suppress desired temporal frequencies.

*Denoising and interference suppression.* Consider a graph signal of the form $\mathbf{x}_t = \mathbf{x}_t^* + \mathbf{i}_t + \mathbf{w}_t$, where

$$\mathbf{u}_n^{\mathsf{H}}\mathbf{x}_t^* = \begin{cases} e^{j\pi t/4} & \text{if } \lambda_n < 0.5 \\ 0 & \text{otherwise,} \end{cases} \tag{6.38}$$

is the signal of interest, $\mathbf{u}_n^{\mathsf{H}}\mathbf{i}_t = e^{J3\pi t/4}$, for all$\lambda_n$, is the interfering signal and $\mathbf{w}_t$ is a zero mean additive Gaussian noise with $\mathbf{\Sigma}_w = \sigma_w^2\mathbf{I}_N$ and $\sigma_w^2 = 0.1$. Our goal is to recover the graph signal of interest $\mathbf{x}_t^*$ using the two-dimensional FIR graph filtering approach. In this way, we aim to use the FIR filter to suppress the out-of-band noise in the graph and time domain and to suppress the interferer in the temporal domain. Similarly to the ARMA$_K$ we consider the errors (6.19) and (6.20) to measure the filtering performance. For our simulations, the coefficients $\phi_k$ approximate an ideal low pass step function with $\lambda_c = 0.5$, meanwhile the coefficients $b_l$ are found by approximating a low pass temporal filter with cut-off frequency $\omega_c = \pi/2$.

In Fig. 6.9, we can see that after some initial assessment time of the filter, both errors reduce. Specifically, $e_t^{(\text{interf})}$ is reduced by an order of two[4]. This shows the robustness

[4]To further reduce the error the filter can be designed as a band-pass around the oscillating frequency of $\mathbf{x}_t^*$.

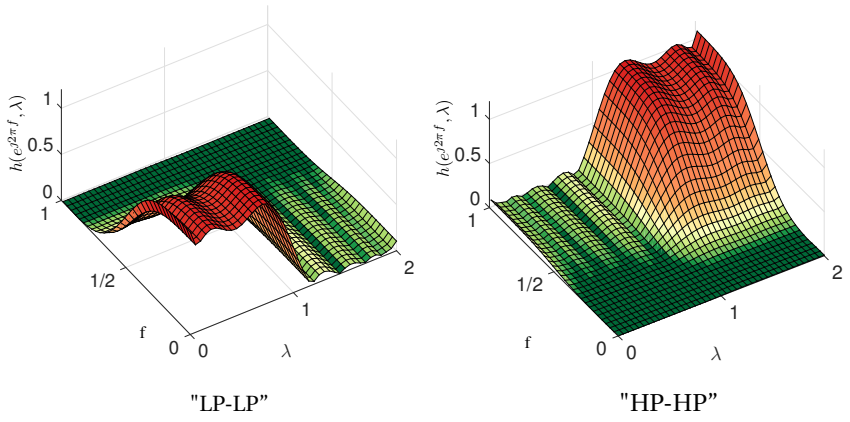



Figure 6.8: Different two-dimensional FIR approximations. From left to right, we have a low-pass (LP) filter in both graph and temporal frequency domain and a high-pass (HP) filter in both domains. A normalized Laplacian has been used and also the temporal frequencies are normalized ($\times\pi$ rad/sample).

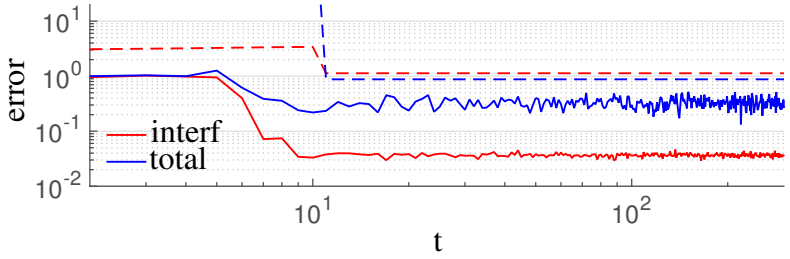
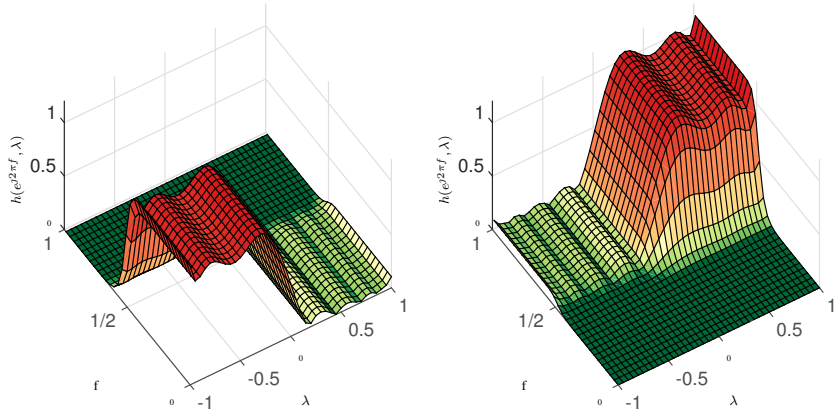



Figure 6.9: Errors as a function of time for the two-dimensional FIR graph-temporal filter (solid line) and for the classical FIR graph filter (dashed line).

of the two-dimensional FIR filter (6.26) to interference, where the interfering signal is attenuated by the filter. An FIR graph filter that considers the input signal only once when starts filtering, it produces errors that are much higher due to their impossibility to operate on the temporal frequencies. These results suggest that the network can process jointly different time-invariant graph signals by modulating them in orthogonal temporal frequencies. Then, we can process them with different two-dimensional filters to separate the mixed signals in the temporal domain.

**ARMA$^{\mathbf{2D}}$.** For these simulations we consider a hybrid ARMA$^{2D}$ consisting of an FIR$_{10}$ in the graph domain and a 6th order Butterworth filter for the temporal domain.

*Filter approximation.* With reference to Figure. 6.10, we can see that the proposed approach can approximate different desired separable two-dimensional frequency responses. Also in this instance, the cut-off frequencies in both domains are the half of the respective bands. In comparison with the FIR$^{2D}$ we remark a higher accuracy in the temporal domain due to the Butterworth approach.




"LP-LP" "HP-HP"

Figure 6.10: Different two-dimensional filter approximations. From left to right, we go from a low-pass (LP) filter in both graph and temporal frequency domain to a high-pass (HP) filter in both domains. The joint filter is an $\text{FIR}_{10}$ in the graph domain and a Butterworth of order 6 in time. The depicted results are w.r.t. the shift operator $\mathbf{S} = \mathbf{L}_{\text{n}} - \mathbf{I}_N$.



*Signal recovery.* We now consider only the task of time-varying signal recovery in noise. For this purpose, we assume a graph signal of the form $\mathbf{x}_t = \mathbf{x}_t^* + \mathbf{w}_t$, where $\mathbf{x}_t^*$ is characterized by $\mathbf{u}_n^{\mathsf{H}} \mathbf{x}_t^* = e^{j\pi t/4}$ if $\lambda_n < \lambda_{\max}/2$ and zero otherwise. We consider zero-mean white Gaussian noise with different noise powers. To recover our signal $\mathbf{x}_t^*$ we adopt the double LP filter of Figure. 6.10 with graph cut-off graph frequency $\lambda_{\max}/2$ and temporal cut-off frequency $f_c = 1/2$. Considering that not only the signal but also the noise is time-varying, we expect the two-dimensional filter to cancel the noise spectral part outside the band of interest not only in the graph domain but also in the temporal domain. To quantify the performance, we define a new measure of quality, named the signal-to-error ratio (SER) as

$$\text{SER}_t = \frac{\|\hat{\mathbf{x}}_t^*\|^2}{\|\hat{\mathbf{y}}_t - \hat{\mathbf{x}}_t^*\|^2}, \tag{6.39}$$

which quantifies how well we cancel the out-of-band noise and approximate the filter (notice that our desired response in the graph frequency domain is $\hat{\mathbf{x}}_0^*$).

In Figure 6.11 we show the SER (in dB) for different input signal-to-noise ratios (SNRs), for both hybrid approaches and the universal FIR filter which operates only on the graph spectral domain. The pure FIR graph filter assumes that $\mathbf{x}_t^*$ does not oscillates in time (i.e., $\mathbf{x}_t^* = \mathbf{x}_0^*$) and only the noise is time-varying. For a time-varying $\mathbf{x}_t^*$, the FIR performance degrades since it does not consider the temporal spectrum of the input signal. As seen in the figure, $\text{ARMA}^{2\text{D}}$ outperforms the FIR graph filter for all noise levels. When the noise level decreases, i.e., the input SNR is higher, the SER is lower than the input SNR due to the filter approximation accuracy. We conclude that the proposed two-dimensional filters, as expected, better suit time-varying environments.

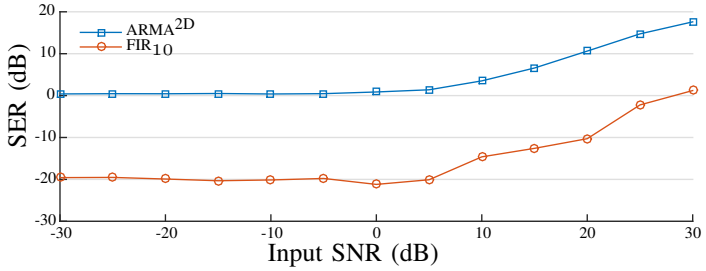



Figure 6.11: SER (dB) vs. input SNR (dB) for our presented ARMA$^{2D}$ approach (FIR$_{10}$ in graph and Butterworth 6 in time) and for the classical universal FIR$_{10}$ graph filters. The results are averaged over 10 iterations.

## 6.4. CONCLUDING REMARKS

This chapter concerned the analysis of graph filters under deterministic variations in the graph topology and/or graph signal. We first showed that the ARMA$_K$ graph filters extend naturally to joint graph-temporal filters when the input signal is time-varying. We characterized the filter robustness to temporal variations of the input signal and show that a purely graph-based design can cope well with slow variations of the filter input. Next, we analyzed the robustness of the parallel ARMA$_K$ graph filter to variations in the graph topology. We provided closed-form expressions for the ARMA$_K$ output when the graph structure and the graph signal change over time. To process time-varying graph signals we introduced two architectures that allow distributed implementation for the joint graph-temporal (FIR and ARMA) filters. Different implementation strategies such as causal and separable are shown, along with their design challenges. The filters' performance is assessed with numerical results to quantify their ability in dealing with deterministic dynamics in the graph topology and/or graph signal.

## APPENDICES

### 6.A. PROOF OF THE JOINT ARMA$_K$ GRAPH AND TEMPORAL FREQUENCY RESPONSE THEOREM

The recursion of a parallel ARMA$_K$ with time-varying input is

$$\mathbf{y}_{t+1}^{(k)} = \psi^{(k)}\mathbf{S}\mathbf{y}_t^{(k)} + \varphi^{(k)}\mathbf{x}_t \tag{6.40a}$$

$$\mathbf{y}_{t+1} = \sum_{k=1}^{K} \mathbf{y}_{t+1}^{(k)}, \tag{6.40b}$$

where $\mathbf{y}_t^{(k)}$ is the state of the $k$th ARMA$_1$, whereas $\mathbf{x}_t$ and $\mathbf{y}_t$ are the input and output graph signals, respectively. Using the Kronecker product the above takes the more com-

pact form

$$\mathbf{y}_{t+1}^{(1:K)} = (\boldsymbol{\Psi} \otimes \mathbf{S})\,\mathbf{y}_t^{(1:K)} + \boldsymbol{\varphi} \otimes \mathbf{x}_t \tag{6.41a}$$

$$\mathbf{y}_{t+1} = (\mathbf{1}_K^\top \otimes \mathbf{I}_N)\mathbf{y}_t^{(1:K)}, \tag{6.41b}$$

with $\mathbf{y}_t^{(1:K)} = \left[\mathbf{y}_t^{(1)\top}, \mathbf{y}_t^{(2)\top}, \cdots, \mathbf{y}_t^{(K)\top}\right]^\top$ the $NK \times 1$ stacked state vector, $\boldsymbol{\Psi} = \text{diag}(\psi^{(1)}, \psi^{(2)}, \cdots, \psi^{(K)})$ a diagonal $K \times K$ coefficient matrix, $\boldsymbol{\varphi} = (\varphi^{(1)}, \varphi^{(2)}, \cdots, \varphi^{(K)})^\top$ a $K \times 1$ coefficient vector, and $\mathbf{1}_K$ the $K \times 1$ one-vector.

We therefore have

$$\begin{aligned}\mathbf{y}_{t+1}^{(1:K)} &= (\boldsymbol{\Psi} \otimes \mathbf{S})^t \mathbf{y}_0^{(1:K)} + \sum_{\tau=0}^{t} (\boldsymbol{\Psi} \otimes \mathbf{S})^\tau \left(\boldsymbol{\varphi} \otimes \mathbf{x}_{t-\tau}\right) \\ &= \left(\boldsymbol{\Psi}^t \otimes \mathbf{S}^t\right)\mathbf{y}_0^{(1:K)} + \sum_{\tau=0}^{t} \left(\boldsymbol{\Psi}^\tau \boldsymbol{\varphi}\right) \otimes \left(\mathbf{S}^\tau \mathbf{x}_{t-\tau}\right).\end{aligned} \tag{6.42}$$

Notice that, when the convergence condition $\|\psi^{(k)}\mathbf{S}\| < 1$ is met, $\lim_{t\to\infty} \|\left(\boldsymbol{\Psi}^t \otimes \mathbf{S}^t\right)\mathbf{y}_0\|_2 = 0$. Hence, for sufficiently large $t$, the ARMA$_K$ output is

$$\begin{aligned}\lim_{t\to\infty} \mathbf{y}_{t+1} &= \lim_{t\to\infty} \sum_{\tau=0}^{t} (\mathbf{1}^\top \otimes \mathbf{I}_N) \left(\boldsymbol{\Psi}^\tau \boldsymbol{\varphi}\right) \otimes \left(\mathbf{S}^\tau \mathbf{x}_{t-\tau}\right) \\ &= \lim_{t\to\infty} \sum_{\tau=0}^{t} \left(\mathbf{1}_K^\top \boldsymbol{\Psi}^\tau \boldsymbol{\varphi}\right) \otimes \left(\mathbf{S}^\tau \mathbf{x}_{t-\tau}\right) \\ &= \lim_{t\to\infty} \sum_{\tau=0}^{t} \sum_{k=1}^{K} \varphi^{(k)} \left(\psi^{(k)}\mathbf{S}\right)^\tau \mathbf{x}_{t-\tau} + c\,\mathbf{x}_t,\end{aligned} \tag{6.43}$$

where we have used the Kronecker product property $(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = (\mathbf{AC}) \otimes (\mathbf{BD})$ and expressed the Kronecker product as the sum of $K$ terms. The transfer matrix $\mathbf{H}(z)$ is obtained by taking the Z-transform in both sides and re-arranging the terms

$$\mathbf{H}(z) = z^{-1} \sum_{k=1}^{K} \varphi^{(k)} \sum_{\tau=0}^{\infty} \left(\psi^{(k)}\mathbf{S}\right)^\tau z^{-\tau}. \tag{6.44}$$

Finally, applying the GFT and using the properties of geometric series we obtain the joint transfer function in the closed-form expression

$$\begin{aligned}H(z,\lambda) &= z^{-1} \sum_{k=1}^{K} \varphi^{(k)} \sum_{\tau=0}^{\infty} \left(\psi^{(k)}\lambda\right)^\tau z^{-\tau} \\ &= \sum_{k=1}^{K} \frac{\varphi^{(k)} z^{-1}}{1 - \psi^{(k)}\lambda z^{-1}}\end{aligned} \tag{6.45}$$

and our claim follows.

## 6.B. PROOF OF ARMA OUTPUT DISTANCE IN TIME-VARYING SCENARIOS THEOREM

We start the proof by substituting the expression (6.8) for $t_1$ and $t_2$ into the numerator of (6.14). Then, we can write

$$\begin{aligned}\|\mathbf{y}_{t_1+1}-\mathbf{y}_{t_2+1}\|_2 = \|&\psi^{t_1+1}\mathbf{\Phi}_{\mathbf{S}}(t_1,0)\mathbf{y}_0-\psi^{t_2+1}\mathbf{\Phi}_{\mathbf{S}}(t_2,0)\mathbf{y}_0\\&+\varphi\sum_{\tau=0}^{t_1}\psi^{\tau}\mathbf{\Phi}_{\mathbf{S}}(t_1,t_1-\tau+1)\mathbf{x}_{t_1-\tau}\\&-\varphi\sum_{\tau=0}^{t_2}\psi^{\tau}\mathbf{\Phi}_{\mathbf{S}}(t_2,t_2-\tau+1)\mathbf{x}_{t_2-\tau}\|_2.\end{aligned} \tag{6.46}$$

Rearranging the terms, we have

$$\begin{aligned}\|\mathbf{y}_{t_1+1}-\mathbf{y}_{t_2+1}\|_2 = \|&\psi^{t_1+1}\mathbf{\Phi}_{\mathbf{S}}(t_1,0)\mathbf{y}_0-\psi^{t_2+1}\mathbf{\Phi}_{\mathbf{S}}(t_2,0)\mathbf{y}_0\\&+\varphi\sum_{\tau=t_2+1}^{t_1}\psi^{\tau}\mathbf{\Phi}_{\mathbf{S}}(t_1,t_1-\tau+1)\mathbf{x}_{t_1-\tau}\|_2\end{aligned} \tag{6.47}$$

By using the Cauchy-Schwarz property, the triangle inequality of the spectral norm, and a uniform bound $\varrho$ on the eigenvalues of matrices $\mathbf{S}_t$, the above expression simplifies

$$\begin{aligned}\|\mathbf{y}_{t_1+1}-\mathbf{y}_{t_2+1}\|_2 \leq &\left(|\psi\varrho|^{t_1+1}+|\psi\varrho|^{t_2+1}\right)\|\mathbf{y}_0\|_2\\&+|\varphi|\sum_{\tau=t_2+1}^{t_1}|\psi\varrho|^{\tau}\|\mathbf{x}_{t_1-\tau}\|_2.\end{aligned} \tag{6.48}$$

Leveraging the fact that $|\psi\varrho|<1$, as well as that $\|\mathbf{x}_t\|_2 \leq x_{\max}$ for every $t$, we can express the sum in a closed form

$$\sum_{\tau=t_2+1}^{t_1}|\psi\varrho|^{\tau}\|\mathbf{x}_{t_1-\tau}\|_2 \leq x_{\max}\left(\frac{|\psi\varrho|^{t_2+1}-|\psi\varrho|^{t_1+1}}{1-|\psi\varrho|}\right). \tag{6.49}$$

We obtain the desired bound on $\epsilon_{t_1,t_2}$ by dividing the above expressions with $x_{\max}$ and adjusting the indices.

## 6.C. PROOF OF THE TWO-DIMENSIONAL ARMA FREQUENCY RESPONSE PROPOSITION

Assume that the filter is stable and consider a reduced dimension problem for (6.29). This can be done by setting $\mathbf{x}_t = x_t\mathbf{u}$, with $x_t$ the scalar magnitude of $\mathbf{x}_t$ in the eigenspace of $\mathbf{u}$. By considering the orthogonality of the shift operator eigenbasis, we can then rewrite (6.29) as

$$\sum_{p=0}^{P}\sum_{l=0}^{L_p}\psi_{l,p}\lambda^l y_{t-p} = \sum_{q=0}^{Q}\sum_{k=0}^{K_q}\varphi_{k,q}\lambda^k x_{t-q}, \tag{6.50}$$

where $y_t \in \mathbb{C}$ is the magnitude of $\mathbf{y}_t \in \mathbb{C}^N$ in the eigenspace of $\mathbf{u}$. Taking the Z-transform on both sides and by rearranging the terms we obtain the joint transfer function (6.30).

Let us analyze the stability condition for which recursion (6.29) converges. For simplicity, we focus on the dimensionality reduced equivalent problem (6.50), but the result also holds for (6.29). We start by rewriting (6.50) as

$$\sum_{l=0}^{L_0} \psi_{l,0} \lambda^l y_t + \sum_{p=1}^{P} \sum_{l=0}^{L_p} \psi_{l,p} \lambda^l y_{t-p} = \sum_{q=0}^{Q} \sum_{k=0}^{K_q} \varphi_{k,q} \lambda^k x_{t-q}. \tag{6.51}$$

We then define the $P \times 1$ vector $\mathbf{y}_{P,t} = [y_{t-P+1}, y_{t-P+2}, \ldots, y_{t-1}, y_t]^\top$, the $(Q+1) \times 1$ vector $\mathbf{x}_{Q,t} = [x_{t-Q}, x_{t-Q+1}, \ldots, x_{t-1}, x_t]^\top$, $\mathbf{\Psi}_p = \sum_{l=0}^{L_p} \psi_{l,p} \lambda^l$ and $\mathbf{\Phi}_q = \sum_{k=0}^{K_q} \varphi_{k,q} \lambda^k$. With these definitions in place, we rewrite (6.51) as

$$a_0 \mathbf{\Psi}_0 \mathbf{y}_{P,t} = \mathbf{P} \mathbf{y}_{P,t-1} + \mathbf{Q} \mathbf{x}_{Q,t}, \tag{6.52}$$

with $\mathbf{P}$ and $\mathbf{Q}$ the $P \times P$ and $(Q+1) \times (Q+1)$ matrices, respectively, defined as

$$\mathbf{P} = \begin{bmatrix} 0 & \mathbf{\Psi}_0 & \ldots & 0 \\ 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \mathbf{\Psi}_0 \\ -\mathbf{\Psi}_P & -\mathbf{\Psi}_{P-1} & \ldots & -\mathbf{\Psi}_1 \end{bmatrix}, \quad \mathbf{Q} = \begin{bmatrix} 0 & \ldots & 0 \\ 0 & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & 0 \\ \mathbf{\Phi}_Q & \ldots & \mathbf{\Phi}_0 \end{bmatrix}.$$

We can rewrite (6.52) as

$$\mathbf{y}_{P,t} = \mathbf{P}' \mathbf{y}_{P,t-1} + \mathbf{Q}' \mathbf{x}_{Q,t}, \tag{6.53}$$

with $\mathbf{P}' = (\mathbf{\Psi}_0)^{-1} \mathbf{P}$ and $\mathbf{Q}' = (\mathbf{\Psi}_0)^{-1} \mathbf{Q}$. It is then clear that (6.53) and thus (6.29) converges if the eigenvalues of $\mathbf{P}'$ are lower than one in magnitude, i.e., $\|\mathbf{P}\| < |\mathbf{\Psi}_0|$, and if $(\mathbf{\Psi}_0)^{-1}$ is different from zero for all eigenvalues $\lambda$ of $\mathbf{S}$.



## FURTHER READING


[1] E. Isufi, A. Loukas, A. Simonetto, and G. Leus, *Autoregressive moving average graph filtering,* IEEE Transactions on Signal Processing **65**, 274 (2017).

[2] E. Isufi, A. Loukas, A. Simonetto, and G. Leus, *Separable autoregressive moving average graph-temporal filters,* in *Signal Processing Conference (EUSIPCO), 2016 24th European* (IEEE, 2016) pp. 200–204.

[3] E. Isufi, G. Leus, and P. Banelli, *2-dimensional finite impulse response graph-temporal filters,* in *Signal and Information Processing (GlobalSIP), 2016 IEEE Global Conference on* (IEEE, 2016) pp. 405–409.

[4] A. Loukas, *Distributed graph filters,* (2015).

[5] A. Loukas, A. Simonetto, and G. Leus, *Distributed autoregressive moving average graph filters,* IEEE Signal Processing Letters **22**, 1931 (2015).

[6] A. Loukas and N. Perraudin, *Stationary time-vertex signal processing,* arXiv preprint arXiv:1611.00255 (2016).

[7] E. Isufi, A. Loukas, N. Perraudin, and G. Leus, *Forecasting time series with varma recursions on graphs,* arXiv preprint arXiv:1810.08581 (2018).

[8] B. Touri, *Product of random stochastic matrices and distributed averaging* (Springer Science & Business Media, 2012).

[9] D. I. Shuman, P. Vandergheynst, and P. Frossard, *Distributed signal processing via chebyshev polynomial approximation,* arXiv preprint arXiv:1111.5239 (2011).

[10] N. Aschenbruck, R. Ernst, E. Gerhards-Padilla, and M. Schwamborn, *Bonnmotion: a mobility scenario generation and analysis tool,* in *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques* (ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2010) p. 51.

[11] J. Mei and J. M. Moura, *Signal processing on graphs: Causal modeling of unstructured data,* IEEE Transactions on Signal Processing **65**, 2077 (2017).

[12] E. Ghadimi, *Accelerating convergence of large-scale optimization algorithms*, Ph.D. thesis, KTH Royal Institute of Technology (2015).

[13] A. V. Oppenheim and R. W. Schafer, *Digital signal processing. 1975,* Englewood Cliffs, New York .

# 7

# STATISTICAL ANALYSIS OF GRAPH-TIME FILTERING

*If your experiment needs statistics,*
*you ought to have done a better experiment.*

Ernest Rutherford

We now depart from the *deterministic* analysis of graph-time filtering in which the graph topology and the graph signal were assumed to change. Instead, in this chapter, we assume the graph topology and the graph signal to vary *stochastically* with known statistical properties. The motivation to embrace this approach is twofold. First, the graph can be affected by edge losses, e.g., link failures in sensor networks, which can be modeled better as a stochastic process. Second, the graph signal can have a stochastic nature as well, e.g., affected by noise. We then would like to characterize the filter output from a statistical viewpoint and express its relation to the statistics of the graph and graph signal. By assigning a statistical distribution to the filter output, we can, therefore, argue in terms of *filtering output in the mean* and *filtering output deviation* from the mean value. To the best of our knowledge, this is the first attempt that approaches GSP and graph filtering from this perspective.

The organization of this chapter is a follows. Section 7.1 motivates the research and provides a list of contributions. The considered stochastic model and the assumptions the developed theory relies on are shown in Section 7.2. The statistical analysis of the filter's output is carried out in Section 7.3. Section 7.4 exploits the stochasticity in the graph signal to perform joint graph-temporal signal denoising, while in Section 7.5 we leverage stochasticity to implement distributed graph filtering in a sparsified way. Section 7.6 wraps up the chapter with a summary of the results.

Parts of this chapter have been published in the IEEE Transactions on Signal Processing [1] (2017), in the IEEE CAMSAP [2] (2015), and in the IEEE ICASSP [3] (2017).

## 7.1. INTRODUCTION

Stochasticity is encountered in applications like communication networks, social networks, smart grids, and road networks. It occurs, for instance, when the graph signal is corrupted with random additive noise, when it obeys a certain distribution, or when –owing to random link and node failures– the graph topology becomes random. Characterizing the impact of random graph perturbations has been considered for example in distributed optimization [4–7], but not in graph filters.

In the sequel, we analyze the graph filters' behavior when the input signal on the graph *and* the graph topology are random processes over *time* with given statistical properties, yet with independent realizations. We remark that when we talk about a random graph topology; we mean the signal graph, i.e., the graph that explains the signal structure, which also corresponds to the graph used for processing the graph signals. For instance, in a road network, the signal graph can be random due to accidents. The actual graph used for data exchange named the communication graph (e.g., between sensor nodes placed on the crossroads of a road network) could be similar to the signal graph or not, but we will not discuss this issue in this chapter. Hence, the term graph will always refer to the *signal graph*. We believe that computing the filter output by incorporating the stochasticity of the graph is more meaningful since they are an integral part of the graph signal realizations. Coming back to the road network example, the traffic at a particular time instant is a consequence of the past realizations of the graph.

Finally, we view stochasticity as a tool to ease the computational and communication costs in distributed graph filtering over a deterministic graph. We show that the introduced framework can serve as a standing platform to perform the distributed filtering operations in Chapters 3 and 4 with lower costs.



### 7.1.1. CONTRIBUTIONS

This chapter brings the following contributions to the field of GSP.

**Contribution 7.1.** *Statistical analysis of the filter output.* We provide closed-form expression and upper bounds for the first and second order moments of the filter output, respectively. Our expressions show that state-of-the-art graph filters are equipped to handle stochastic settings (for graphs and signals).

- $1-a$) For a random time-varying graph signal characterized by *temporal stationarity with independent realizations*, living on a random time-varying graph, we show that the expected filter output is equal to the output of the same filter operating on the expected signal living on the expected graph.

- $1-b$) For a random time-varying, yet *non-stationary* graph signal with independent realizations, (i.e., a time-varying mean and covariance) over a random time-varying graph, we show that the expected output signal is equal to the output of a deterministic two-dimensional filter, operating on the time-varying expected signal over the expected graph. In the latter case, the filter jointly captures the variations in the mean of the graph signal over the graph and temporal domains.

- $1-c$) We show that the average node variance of the filter output is upper bounded. Further, we introduce a recursive way to track the variance of the ARMA output iteratively based only on statistical knowledge of the previous time instant.

**Contribution 7.2.** *Graph signal denoising in the mean.* We propose to use stochasticity for improving graph signal denoising tasks under smoothness priors, a.k.a. Tikhonov denoising, when multiple realizations of the graph signal are available. The proposed approach performs an online joint denoising over the graph and time domains exploiting the new realizations of the input signal.

**Contribution 7.3.** *Sparsified implementation of graph filters.* By leveraging tools from sparsification [8] and gossiping [9], we make use of stochasticity to do classical graph filtering with lower communication and complexity costs. The proposed approach considers filtering the signal over a sparsified graph where the information is exchanged only with some randomly chosen neighbors.

### 7.1.2. APPLICATIONS

Besides characterizing the graph filters behavior when involved in the distributed tasks detailed in Section 3.1.2, the proposed statistical analysis may result effective in the following applications.

- *Signal diffusion on graphs affected by edge losses.* The graph filter statistical viewpoint can be useful to address graph signal diffusion over networks affected by edge losses. This includes, but not limited to: $i$) gossip propagation in social networks, where friendships are lost and restored randomly; $ii$) energy diffusion over smart grids, where link losses may occur due to random grid problems; and $iii$) traffic propagation in road networks, where a street (edge) can be closed over time due to accidents. Since the steady-state diffused signal has an $\text{ARMA}_1$ (cf. Chapter 4) interpretation[1], we are interested to analyze how the random fluctuations in the graph topology affect the steady state diffused signal.

- *Sparser implementation of distributed graph filters.* With the goal to increase the network lifetime, we consider solving the distributed graph filtering tasks (e.g., signal denoising, interpolation, and network coding) with random communications with neighbors. In this scenario, each node, for each iteration, decides randomly the neighbors with whom to exchange information with. As a results, the number of the communications drops, but the output has a stochastic nature. Through the statistical analysis of graph filters, we aim designing the communication probabilities such that a target performance guaranteed.

## 7.2. STOCHASTIC MODELING

We here formalize the stochastic model used in the rest of this chapter. Our definitions consider the stochasticity of the graph and the graph signal.

[1] This aspect is covered in more detail in Section 8.2.

### 7.2.1. GRAPH MODEL

We consider a random time-varying graph $\mathcal{G}_t$ as a random edge sampling of an arbitrary, but time-invariant underlying graph $\mathcal{G}$. More formally:

**Definition 7.1.** (Random edge sampling (RES) graph model.) *The probability that a link* $(i, j)$ *in the edge set* $\mathcal{E}$ *is activated at time* $t$ *is* $p_{i,j}$*, with* $0 < p_{i,j} \leq 1$*. The edges are activated independently across time and the graph realizations are considered mutually independent with the random graph process.*

Thus, at each time step $t$, we draw a graph realization $\mathcal{G}_t = (\mathcal{V}, \mathcal{E}_t)$ from the underlying graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the edge set $\mathcal{E}_t \subseteq \mathcal{E}$ is generated via an i.i.d. Bernoulli process; this is a standard way of studying link failures in the literature on network algorithms [10, 11]. In a graph signal processing perspective, the RES model suits better cases where the graph has physical meaning, e.g., smart grids, which will have stochastic repercussions on the filtering output.

Within the context of this chapter, let us refer to $\mathbf{L}$ as the Laplacian relative to the underlying graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, from which the graph realizations at different time instants are drawn, and to $\mathbf{L}_t$ as the Laplacian of the graph realization at time instant $t$. Note that each node locally derives the application of the instantaneous Laplacian matrix $\mathbf{L}_t$ on the graph signal $\mathbf{x}$ by communicating with its neighbors. For convenience, denote the expected Laplacian $\mathbb{E}[\mathbf{L}_t]$ as $\bar{\mathbf{L}}$ related to the expected graph $\bar{\mathcal{G}}$.

For Laplacians $\mathbf{L}$ belonging to a set $\mathcal{L}$ and $\mathcal{E}_t \subseteq \mathcal{E}$, the instantaneous graph Laplacian $\mathbf{L}_t$ of $\mathcal{G}_t$ belongs also to $\mathcal{L}$, meaning that all $\mathbf{L}_t$ have bounded eigenvalues. Further, due to the interlacing property [12], the spectral radius bound $\varrho$ satisfies[2] the property $\|\mathbf{L}_t\| \leq \|\mathbf{L}\| \leq \varrho$ for all $t$. In general, we will say $\mathbf{S} = \mathbf{L}$ (or $\mathbf{S} = \mathbf{L}_{\mathrm{n}}$), $\mathbf{S}_t = \mathbf{L}_t$ (or $\mathbf{S}_t = \mathbf{L}_{\mathrm{n},t}$) and $\bar{\mathbf{S}} = \bar{\mathbf{L}}$ (or $\bar{\mathbf{S}} = \bar{\mathbf{L}}_{\mathrm{n},t}$).



### 7.2.2. SIGNAL MODEL

We consider random time-varying graph signals which can be correlated among the nodes for a fixed time instant, but have independent realizations at different time instants. More formally:

**Definition 7.2.** (Random graph process model.) *The graph signal* $\mathbf{x}_t \in \mathbb{R}^N$*, at time instant* $t$*, is a realization of a random process, with time-invariant mean* $\bar{\mathbf{x}}$ *and an* $N \times N$ *covariance matrix* $\mathbf{\Sigma}_x$*. Signal* $\mathbf{x}_t$ *has independent realizations over time.*

The random graph process model presented above is a subclass of stationary temporal signals, while not being restricted to stationary graph signals (cf. Section 2.4). In this chapter, we work with a broader model, where we do not need the expected graph signal $\mathbf{x}$ to be a constant vector, or the covariance matrix $\mathbf{\Sigma}_x$ to be jointly diagonalizable with the graph shift operator. Further, the considered signal model has a fundamental difference with the work of [13], where the authors assume that the covariance matrix of the graph process is related to the underlying graph Laplacian $\mathbf{L}$. This approach can be seen as a particular case of the stochastic model presented here, assuming that the

[2] For the normalized Laplacian we can set $\varrho = 2$.

covariance matrix of the graph signal is only related to the main graph $\mathcal{G}$ and not to its instantaneous realizations $\mathcal{G}_t$.

Besides the assumed random graph process model, in Section 7.3.2 we extend some of our results to the case where only the second-order moment of the signal is time-invariant, while the mean is time-varying. This model encompasses time-varying deterministic signals corrupted with node-correlated noise, such as sensor noise.

## 7.3. GRAPH FILTERS IN THE MEAN

In this section, we analyze the first and second order statistics of graph filters when the graph topology and graph signal are of stochastic nature. We start by considering our random graph process model and then extend our results to the more general case when the random graph process has a time-varying mean, yet a time-invariant covariance matrix.

### 7.3.1. RANDOM GRAPH PROCESSES

We start with the simpler FIR graph filters. The more involved case of ARMA graph filters is discussed next.

**FIR.** With the definition of the transition matrix $\mathbf{\Phi}_{\mathbf{S}}(t', t) := \prod_{\tau=t}^{t'} \mathbf{S}_\tau$ if $t' \geq t$ and $\mathbf{\Phi}_{\mathbf{S}}(t', t) := \mathbf{I}_N$ if $t' < t$, the output $\mathbf{y}_{t+1}$ of a $K$-th order FIR graph filter is

$$\mathbf{y}_{t+1} = \sum_{k=0}^{K} \phi_k \mathbf{\Phi}_{\mathbf{S}}(t, t-k+1)\mathbf{x}_{t-k+1}. \tag{7.1}$$

Notice that, in contrast to the time-invariant setting, the output which is computed at time $t+1$ is a function of all graph realizations in the time interval $[t-K+1, t]$ and graph signals starting from time $t-K+1$ up to time $t+1$. According to the following proposition, the above graph filter is well behaved when examined in expectation.

**Proposition 7.1.** *Consider the FIR$_K$ graph filter* (7.1)*, the RES graph model and the random graph process model. Then, the expected output of the graph filter after $K$ time steps is*

$$\bar{\mathbf{y}}_{t+1} = \sum_{k=0}^{K} \phi_k \bar{\mathbf{S}}^k \bar{\mathbf{x}}. \tag{7.2}$$

(The proof can be found in Appendix 7.A.)

Proposition 7.1 confirms that, in the mean, an FIR$_K$ filter has the same behavior as if this filter was applied to the expected graph, having the expected graph signal as input.

**ARMA.** The output $\mathbf{y}_t$ of the parallel ARMA$_K$ filter at time instant $t$ operating on a time-varying graph signal $\mathbf{x}_t$ over a time-varying graph $\mathcal{G}_t$ can be expressed as

$$\mathbf{y}_{t+1}^{(k)} = \psi^{(k)} \mathbf{S}_t \mathbf{y}_t^{(k)} + \varphi^{(k)} \mathbf{x}_t \tag{7.3a}$$

$$\mathbf{y}_{t+1} = \sum_{k=1}^{K} \mathbf{y}_{t+1}^{(k)}, \tag{7.3b}$$

where $\mathbf{y}_0^{(k)}$ is arbitrary. Assuming time-varying stochasticity, the subscript $t$ indicates the random time-variations of the graph (captured by the shift operator $\mathbf{S}_t$) and input signal $\mathbf{x}_t$. Theorem 7.1 in the sequel describes the expected behavior of an ARMA$_K$ graph filter.

**Theorem 7.1.** *Under the RES graph model, the random graph process model and the stability of the parallel ARMA$_K$ filter, i.e., $|\psi^{(k)}| < 1/\rho$ for all $k = 1, \dots, K$, the steady state of the expected value of the ARMA$_K$ recursion* (7.3) *is*

$$\bar{\mathbf{y}} = \lim_{t\to\infty} \mathbb{E}[\mathbf{y}_{t+1}] = \sum_{k=1}^{K} \varphi^{(k)} \left(\mathbf{I}_N - \psi^{(k)} \bar{\mathbf{S}}\right)^{-1} \bar{\mathbf{x}}. \tag{7.4}$$

*Recursion* (7.3) *converges in the mean to* (7.4) *linearly, irrespective of the initial condition $\mathbf{y}_0^{(k)}$ and graph realizations $\mathbf{S}_t$.*

(The proof can be found in Appendix 7.B.)

We can see that (7.4) is a parallel ARMA$_K$ filter having as input $\mathbf{x}$ over the graph $\bar{\mathcal{G}}$. Theorem 7.1 asserts that the expected value of the steady state of the filter output is only influenced by the expected value of the signal and by the expected graph, but is independent of any changes in the graph topology.

### 7.3.2. RANDOM GRAPH PROCESSES WITH TIME-VARYING STATISTICS

In the sequel, we characterize the expected filter output in a more general context, i.e., when $\mathbf{x}_t$ is characterized by a time-varying mean and covariance. The generalized signal model is:

**Definition 7.3.** (Random graph process with time-varying statistics.) *The input signal $\mathbf{x}_t \in \mathbb{C}^N$, at time instant $t$, is a realization of a random process, with time-dependent first order moment $\bar{\mathbf{x}}_t$ and time-dependent covariance matrix $\mathbf{\Sigma}_x[t]$. Signal $\mathbf{x}_t$ has independent realizations over time.*

To extend our results to the case of random graph processes with time-varying statistics, we have to perform our analysis in a two-dimensional frequency domain: the graph-frequency domain using the GFT, and the temporal-frequency domain using the Z-transform. Such an analysis is similar to the previously presented for the deterministic setting in Chapter 6.

**FIR.** The following proposition is a generalization of Proposition 7.1.

**Proposition 7.2.** *Consider the FIR$_K$ graph filter* (7.1)*, and let the graph be RES and the graph process be random with a time-varying statistics. Then, the expected output of the graph filter is*

$$\bar{\mathbf{y}}_{t+1} = \sum_{k=0}^{K} \phi_k \bar{\mathbf{S}}^k \bar{\mathbf{x}}_{t-k+1}, \tag{7.5}$$

*which describes a filter on the deterministic time-varying mean signal over the expected graph with the two-dimensional filter transfer function*

$$h(z, \lambda) = \sum_{k=0}^{K} \phi_k \left(\frac{\lambda}{z}\right)^k, \tag{7.6}$$

*where $z$ stems from the Z-transform and $\lambda$ from the GFT.*

(The proof can be found in Appendix 7.C.)

From Proposition 7.2, we can see that the result of Proposition 7.1 extends to the two-dimensional case, capturing jointly the mean signal variations over the expected graph and time. As we will see next, the same result holds also for ARMA graph filters.

**ARMA.** We can now prove the following claim.

**Theorem 7.2.** *Consider the RES graph model and the random graph process model with time-varying statistics. The relation between the expected filter output and the expected graph signal over the expected graph is given by a two-dimensional ARMA$_K$ transfer function*

$$h(z,\lambda) = \sum_{k=1}^{K} \frac{\varphi^{(k)} z^{-1}}{1-\psi^{(k)}\lambda z^{-1}}, \tag{7.7}$$

*subject to the stability conditions of Theorem 7.1.*

(The proof can be found in Appendix 7.D.)

As for the FIR$_K$, Theorem 7.2 claims that, in expectation, we achieve a two-dimensional filter operating jointly on the mean variations over the expected graph and time signal.

The following remarks conclude this section.

**Remark 7.1.** *Throughout our analysis, we have assumed that the graph realizations $\mathcal{G}_t$ (and thus $\mathbf{S}_t$) are independent at different times $t$. This assumption is crucial for our derivations: since $\mathbf{y}_t$ is a function of all $\mathbf{S}_0,\ldots,\mathbf{S}_{t-1}$, when the graphs at different times are dependent $\mathbb{E}[(\boldsymbol{\Psi}\otimes\mathbf{S}_t)\mathbf{y}_t] \neq (\boldsymbol{\Psi}\otimes\mathbf{S})\mathbf{y}_t$, due to $\mathbb{E}[\mathbf{S}_{t1}\mathbf{S}_{t2}] \neq \mathbb{E}[\mathbf{S}_{t1}]\mathbb{E}[\mathbf{S}_{t2}]$. It is therefore a fact that our results do not generalize to many stochastic graph processes (e.g., human mobility, failure cascade). Our findings, however, do suggest that graph filters are robust to some stochastic phenomena common in networks such as noise and edge fluctuations. Nonetheless, the graph signal $\mathbf{x}_t$ might still be a function of the graph topology, e.g., i) the underlying graph $\mathcal{G}$ could be related to $\boldsymbol{\Sigma}_x$ or $\boldsymbol{\Sigma}_x^{-1}$ by construction, including also the case where $\boldsymbol{\Sigma}_x$ shares the graph Laplacian eigenvectors; and ii) as shown above the graph signal mean and covariance can depend arbitrarily (though deterministic) on the graph topology changes. As a side note, the considered approach with mutual independence between the graph signal and graph topology is also a way to jointly handle the particular cases where only one of them is stochastic.*

**Remark 7.2.** *Our second remark is based on the parallelism between stochastic graph filtering and linear system theory [14]. Other works have exploited this analogy to propose graph signal control [15, 16], or graph signal estimation [17, 18]. Differently from linear system theory, the GSP perspective not only gives more insights into what is evolving over the network but also introduces new strategies related to the characteristics (e.g., smoothness prior, stationarity and bandlimitedness) of the graph signal w.r.t. the underlying graph. More specifically, we can view the ARMA graph filter as a linear state-space model with $\mathbf{x}_t$, $\mathbf{y}_t^{(1:K)} = \left[\mathbf{y}_t^{(1)\top},\mathbf{y}_t^{(2)\top},\cdots,\mathbf{y}_t^{(K)\top}\right]^\top$ and $\mathbf{y}_t$ being the stochastic input signal, the $NK\times 1$ stacked state vector of the system, and the system output at time $t$, respectively. The stochastic matrix $\mathbf{S}_t$, which now captures the stochastic graph topology at time instant $t$, is the stochastic transfer operator of the system. Thus, we say that the expected*

*behavior of the graph filter is related to the expected behavior of the system and the same holds for the output signal. While in deterministic [19, 20] and stochastic control [21–23] of linear systems the focus is on designing the optimal controller, this work considers the analysis/design of the filter transfer function under stochastic variations in the graph topology and/or graph signal.*

### 7.3.3. VARIANCE ANALYSIS

As we saw in the previous section, when examined in expectation, graph filters are impervious to stochasticity of the graph topology and/or graph signal. To quantify how far from the mean a realization can be, in the following, we characterize the covariance of the filter output $\mathbf{\Sigma}_y$ as a function of that of the input signal $\mathbf{\Sigma}_x$. We derive the limiting average variance of the filter output over all nodes, defined as

$$\lim_{t\to\infty} \overline{\text{var}}[\mathbf{y}_t] = \lim_{t\to\infty} \frac{\text{tr}\left(\mathbf{\Sigma}_y[t]\right)}{N} = \lim_{t\to\infty} \frac{\text{tr}\left(\mathbb{E}[\mathbf{y}_t\mathbf{y}_t^{\mathsf{H}}] - \mathbf{y}\mathbf{y}^{\mathsf{H}}\right)}{N}. \tag{7.8}$$

For the FIR graph filters, the limit can be omitted. Expression (7.8), often used for statistical characterizations in signal processing, presents a simple way to quantify the experienced variance of the filter output at each node.

**FIR.** The following proposition formally states an upper bound on the average variance of the output signal of the $\text{FIR}_K$ filter.

**Proposition 7.3.** *Consider the $\text{FIR}_K$ graph filter (7.1) with a random graph process living on a RES graph. The average variance among all nodes of the $\text{FIR}_K$ filter output is bounded by*

$$\overline{var}[\mathbf{y}_{t+1}] \leq (\boldsymbol{\varrho}^{\mathsf{T}}\boldsymbol{\phi})^2 \left(\overline{var}[\mathbf{x}_t] + \frac{\|\mathbf{x}\|_2^2}{N}\right), \tag{7.9}$$

*with vectors $\boldsymbol{\varrho}$ and $\boldsymbol{\varphi}$ respectively, defined as $\boldsymbol{\varrho} = [1, \varrho, \varrho^2, \ldots, \varrho^K]^{\mathsf{T}}$ and $\boldsymbol{\phi} = [\phi_0, \phi_1, \ldots, \phi_K]^{\mathsf{T}}$.*

(The proof can be found in Appendix 7.E.)

The result of Proposition 7.3 suggests that despite the drastic variation of the graph topology, the average variance of the output signal over the nodes is finite and thus also the variance of the output signal at a given node. We can notice that the filter order $K$ has indeed an impact on the bound.

**ARMA.** As for the FIR, the following theorem gives an expression on how to upper bound the average variance across all nodes of the ARMA graph filter output.

**Theorem 7.3.** *Consider the $\text{ARMA}_K$ graph filter (7.3) with a random graph process living on a RES graph and additionally assume that the initial filter state $\mathbf{y}_0^{(1:K)}$ is independent of $\mathbf{x}_t$ with zero mean $\mathbb{E}[\mathbf{y}_0^{(1:K)}] = \mathbf{0}_{KN}$. The limiting average variance among all nodes of the $\text{ARMA}_K$ filter output is bounded by*

$$\lim_{t\to\infty} \overline{var}[\mathbf{y}_{t+1}] \leq \frac{K\|\boldsymbol{\varphi}\|_2^2}{(1-\varrho|\psi_{max}|)^2} \left(\overline{var}[\mathbf{x}_t] + \frac{\|\bar{\mathbf{x}}\|_2^2}{N}\right) \tag{7.10}$$

*where $\psi_{max} = \max\{\psi^{(1)}, ..., \psi^{(K)}\}$.*

(The proof can be found in Appendix 7.F.)

According to Theorem 7.3, the bound depends linearly on the filter order $K$ and on the sum of the squared coefficients $\boldsymbol{\varphi}$.

We now continue with the recursive approach to calculate exactly the variance of the filter output. We consider the case of graph process with time-varying statistics, since, differently from the bounds (7.9)-(7.10), the recursive variance is less challenging to compute. Recalling that the graph shift operator at time $t$, i.e., $\mathbf{S}_t$, is a random realization with mean $\mathbf{S}$, it can be written as $\mathbf{S}_t = \mathbf{S} + \tilde{\mathbf{S}}_t$, with $\tilde{\mathbf{S}}_t$ being the related zero-mean process. We then define the matrices

$$\mathbf{A} = \boldsymbol{\Psi} \otimes \mathbf{S} \text{ and } \tilde{\mathbf{A}}_t = \boldsymbol{\Psi} \otimes \tilde{\mathbf{S}}_t \tag{7.11}$$

and rewrite the ARMA$_K$ filter (7.3) as

$$\mathbf{y}_{t+1}^{(1:K)} = \mathbf{A}\mathbf{y}_t^{(1:K)} + \tilde{\mathbf{A}}_t \mathbf{y}_t^{(1:K)} + \boldsymbol{\varphi} \otimes \mathbf{x}_t \tag{7.12a}$$

$$\mathbf{y}_{t+1} = \mathbf{C}\mathbf{y}_{t+1}, \tag{7.12b}$$

with $\boldsymbol{\Psi} = \text{diag}(\psi^{(1)}, \psi^{(2)}, \cdots, \psi^{(K)})$ a diagonal $K \times K$ coefficient matrix and $\mathbf{C} = \mathbf{1}_K^\mathsf{T} \otimes \mathbf{I}_N$. The instantaneous expected value of the output signal of (7.12) is calculated as

$$\mathbf{y}_{t+1}^{(1:K)} = \mathbf{A}\mathbf{y}_t + \boldsymbol{\varphi} \otimes \mathbf{x}_t \tag{7.13a}$$

$$\mathbf{y}_{t+1} = \mathbf{C}\mathbf{y}_{t+1}^{(1:K)}, \tag{7.13b}$$

which suggests a way to track recursively the expected value of the output signal at the time instant $t+1$ based on the knowledge of the first order statistics at the previous time instant $t$. With this in place, we state the following

**Proposition 7.4.** *Given the ARMA$_K$ graph filter* (7.12) *operating on a random graph process with time-varying statistics over a RES graph model and given first and second order moments of the filter output at time instant $t$. Then, the covariance matrix of the output signal at time instant $t+1$ can be calculated as*

$$\boldsymbol{\Sigma}_y[t+1] = \mathbf{C}\left(\mathbf{A}\boldsymbol{\Sigma}_{y^{(1:K)}}[t]\mathbf{A}^\mathsf{T} + \boldsymbol{\varphi}\boldsymbol{\varphi}^\mathsf{T} \otimes \boldsymbol{\Sigma}_x[t] + \mathbb{E}_{\tilde{\mathbf{A}}}\left[\tilde{\mathbf{A}}_t \mathbf{R}_{y^{(1:K)}}[t]\tilde{\mathbf{A}}_t^\mathsf{T}\right]\right)\mathbf{C}^\mathsf{T} \tag{7.14}$$

*where* $\mathbf{R}_{y^{(1:K)}}[t] = \mathbb{E}_{\mathbf{y}^{(1:K)}}\left[\mathbf{y}_t^{(1:K)}\mathbf{y}_t^{(1:K),\mathsf{T}}\right]$ *with* $\mathbb{E}_{\tilde{\mathbf{A}}}[\cdot]$ *the expected value only regarding the random variable* $\tilde{\mathbf{A}}$.

*The entry $(i,j)$ of* $\mathbb{E}_{\tilde{\mathbf{A}}}[\tilde{\mathbf{A}}_t \mathbf{R}_{\mathbf{y}^{(1:K)}}[t]\tilde{\mathbf{A}}_t^\mathsf{T}]$ *can be calculated as*

$$\begin{aligned}\mathbb{E}_{\tilde{\mathbf{A}}}\left[\tilde{\mathbf{A}}_t \mathbf{R}_{\mathbf{y}^{(1:K)}}[t]\tilde{\mathbf{A}}_t^\mathsf{T}\right]_{i,j} &= \sum_{n=1}^{N} C_{A_{j1},A_{in}}\left[\mathbf{R}_{\mathbf{y}^{(1:K)}}[t]\right]_{n,1} \\ &+ \sum_{n=1}^{N} C_{A_{j2},A_{in}}\left[\mathbf{R}_{\mathbf{y}^{(1:K)}}[t]\right]_{n,2} + \cdots + \sum_{n=1}^{N} C_{A_{jN},A_{in}}\left[\mathbf{R}_{\mathbf{y}^{(1:K)}}[t]\right]_{n,N},\end{aligned} \tag{7.15}$$

*with* $C_{A_{oq},A_{rs}} = cov(A_{oq}[t], A_{rs}[t])$ *the covariance between the $(o,q)$-th and $(r,s)$-th entries of the matrix* $\tilde{\mathbf{A}}_t$, *which can be expressed as a function of the link activation probability and filter coefficients.*

(The proof can be found in Appendix 7.G.)

The result of Proposition 7.4 gives a way to track the variance of the filter output signal, and thus to characterize stochastically how far a realization may be at each node.

We conclude this section with the following remarks.

**Remark 7.3.** *We see from both bounds and the recursive variance that the order of the filter and the sum of the squared coefficients have an influence on the average variance. Thus, our handle on getting a small variance is the filter order and the coefficient values. The latter can be a problem for the FIR filter where the magnitude of the coefficients is much greater than for the ARMA filter. Further, their value increases with the filter order. One way to reduce the filter coefficients value and therefore the variance is to work with the shift operators that are translated Laplacians, i.e.,* $\mathbf{S} = \mathbf{L} - \lambda_{max}(\mathbf{L})/2\mathbf{I}_N$, *or* $\mathbf{S} = \mathbf{L}_n - \mathbf{I}_N$, *instead of using the discrete Laplacian* $\mathbf{L}$, *or normalized Laplacian* $\mathbf{L}_n$. *Also, considering that the spectral norm of* $\mathbf{S}$ *(i.e.,* $\varrho$*) impacts the bounds* (7.9) - (7.10)*, the use of shift operator matrices with small spectral norms, such as the ones mentioned above, is recommended to achieve a small variance in the output signal.*

**Remark 7.4.** *While the above derivations concern the graph filter behavior in a stochastic environment, one could extend the analysis in the properties of random graph signals over random graphs. Thus, for a deterministic graph signal* $\mathbf{x}$ *living on a random graph* $\mathcal{G}_t$, *the 2-Dirichlet form (*$S_2(\mathbf{x}) = \mathbf{x}^\mathsf{T}\mathbf{L}\mathbf{x}$*) defined in Section 2.3.1 will be a random variable, with mean* $\mathbb{E}[S_2(\mathbf{x})] = \mathbf{x}^\mathsf{T}\bar{\mathbf{L}}\mathbf{x}$ *and variance* $var(S_2(\mathbf{x})) = \mathbb{E}[(\mathbf{x}^\mathsf{T}(\mathbf{L} - \bar{\mathbf{L}})\mathbf{x})^2]$. *In this setting, smooth graph signals over random graphs are characterized by being smooth in the mean and by having* $var(S_2(\mathbf{x}))$ *as small as possible. Similarly, when* $\mathbf{x}$ *is a random process, the reasoning extends by incorporating also the statistics of* $\mathbf{x}$ *in the* $S_2(\mathbf{x})$ *analysis.*

7

### 7.3.4. NUMERICAL RESULTS

We consider the scenario with $N = 100$ nodes randomly placed in a square area and two nodes are neighbors if they are closer than 15% of the maximum distance of this area. The comparison of the filter output is done at steady-state, i.e., after $K$ time instants for the FIR and $20 \times K$ iterations. The ARMA filter is initialized as $\mathbf{y}_0^{(1:K)} = \mathbf{0}_{KN}$. The empirical results are averaged over 2000 simulation runs. We run the filters w.r.t. the translated normalized Laplacian $\mathbf{S} = \mathbf{L}_\mathrm{n} - \mathbf{I}_N$. Such a choice has the benefit to improve the ARMA stability region and produces FIR coefficients of lower magnitude.

We aim to show that graph filters can handle stochastic variations of the graph topology and graph signal. This scenario can be considered as the case when we want to perform the filtering on a deterministic graph with a deterministic signal, but we have only random realizations of the latter. We also aim to quantify the variance of the output signal, both empirically and with upper-bounds. We analyze both FIR and ARMA filters for $K = 1, 5, 10$ applied to an input signal of the form $\mathbf{x}_t = \bar{\mathbf{x}} + \mathbf{w}_t$, with

$$\mathbf{u}_n^\mathsf{H}\bar{\mathbf{x}} = \begin{cases} 1 & \text{if } \lambda_n < 1 \\ 0 & \text{if } \lambda_n \in [1, 2] \end{cases} \tag{7.16}$$

being low pass, and noise $\mathbf{w}_t$ which follows a zero mean normal distribution with covariance matrix $\mathbf{\Sigma}_w = \sigma_w^2\mathbf{I}_N$. All graph filters (FIR and ARMA) approximate a frequency

response identical to the one of the graph signal $\bar{\mathbf{x}}$. We analyze the filtering performance for different link-activation probabilities $p$ and noise powers $\sigma_w^2$. We compare the filtered signal under the stochastic model $\mathbf{y}^{(\mathrm{s})}$ with the deterministic output signal $\mathbf{y}^{(\mathrm{d})}$ of the same filters operating on the expected graph with graph signal $\bar{\mathbf{x}}$. The expected (translated) normalized Laplacian is estimated over 1000 runs.

To quantify the performance we define the error

$$\mathbf{e} = \mathbf{y}^{(\mathrm{s})} - \mathbf{y}^{(\mathrm{d})}. \tag{7.17}$$

Given then the zero mean-error (7.17), our conclusions will be based on the empirical average standard deviation among all nodes and realizations

$$\bar{\sigma}_e = \left[\mathrm{tr}\left(\mathbb{E}[\mathbf{e}\mathbf{e}^{\mathsf{H}}]\right)/N\right]^{\frac{1}{2}} = \left[\mathrm{tr}\left(\mathbf{\Sigma}_{y^{(\mathrm{s})}}\right)/N\right]^{\frac{1}{2}}, \tag{7.18}$$

which for a sufficiently large number of realizations approaches the square root of the average variance used in our bounds, i.e., $\bar{\sigma}_e \approx \sqrt{\overline{\mathrm{var}}[\mathbf{y}^{(\mathrm{s})}]}$.

Figure 7.1 depicts the empirical average standard deviation for different values of $p$ and $\sigma_w^2$. We observe the following:

First, ARMA graph filters yield smaller variance for low filter orders ($K = 1$ and 5), as compared to FIR. The order under which ARMA filters are preferable to FIR is $K = 8$. The increased variance of high order ARMA filters examined here is mostly related to their higher convergence time, thus a higher error is introduced in the filter memory through $\mathbf{y}_t^{(1:K)}$ while computing the "steady state" output. The effect could be improved by imposing a stricter stability constraint in the design problem, asking now that $|p_k| \geq c > 1$ for some constant $c$, which is translated in a faster convergence at the expense of a worse filter approximation.

Second, there is an exponential dependence between the output and input variance. This phenomenon becomes clear when the noise is the only source of variance ($p = 1$). The results in Figure 7.1 suggest that graph filters can attain a reasonable performance with stochastically time-varying signals as long as the noise variance is lower than $\sigma_w^2 < 10^{-1}$. However, as we show in Section 7.4, filtering a stochastic signal can have a beneficial outcome, for example, with denoising, if compared not to the deterministic filter output $\mathbf{y}^{(\mathrm{d})}$ but to the denoising solution. For $\sigma_w^2 \geq 10^{-3}$, $p$ has a smaller influence on $\bar{\sigma}_e$.

Last, lower-order filters (especially ARMA) tolerate a significant amount of edge fluctuations when the noise variance is small. This finding motivates the use of graph filters in stochastic time-varying scenarios, as occurring in distributed systems with message loss. A more comprehensive analysis is presented in stochastic sparsification in Section 7.5.

Table 8.1 depicts the square roots of the upper bounds (7.9) and (7.10) for the FIR and ARMA filters, respectively. Despite the difference between the variance guaranteed by the upper bounds and the empirical variance, the results in Table I and Figure 7.1 suggest that *(i)* the filter order mpacts the actual variance as well and *(ii)* in practice the actual variance is much lower.

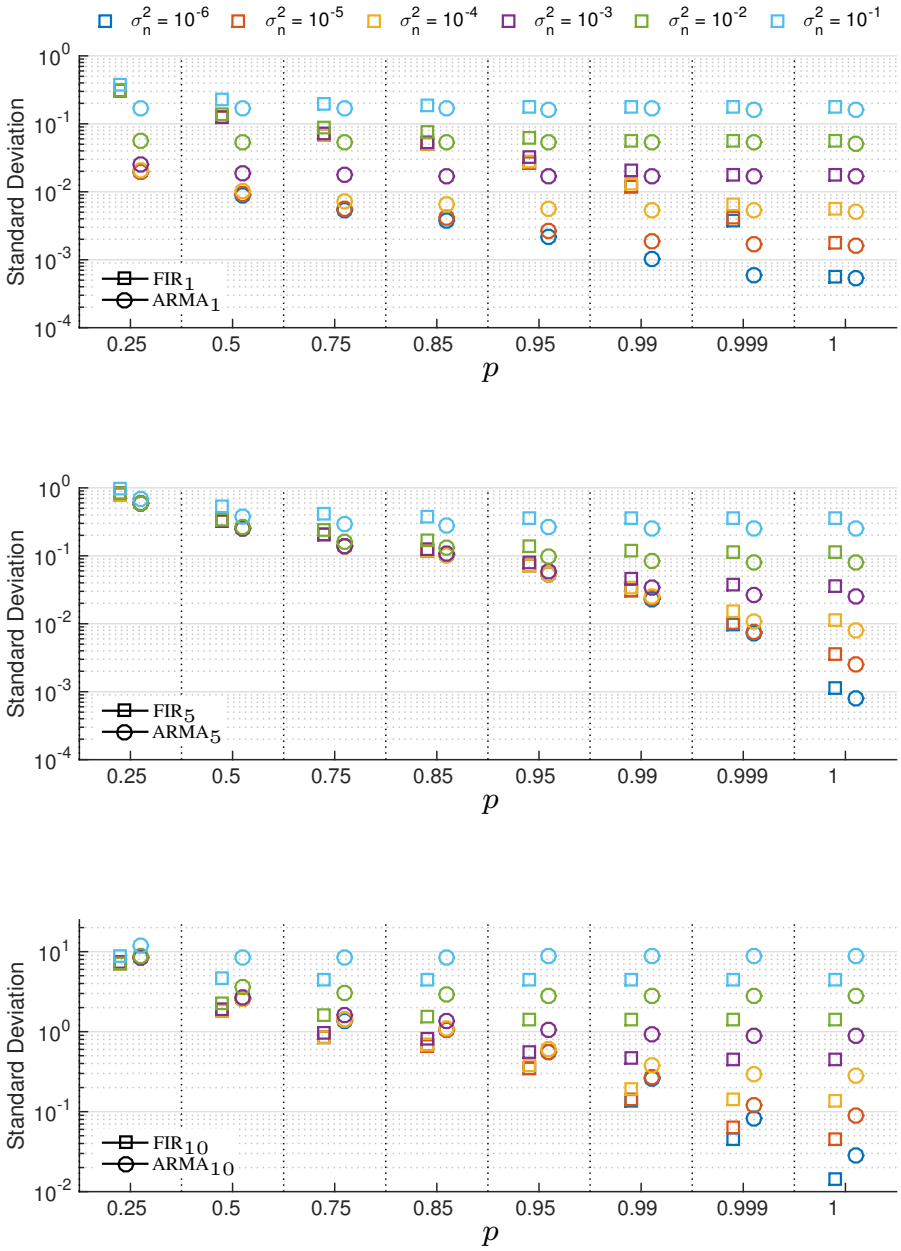



Figure 7.1: Empirical average standard deviation among all nodes and all realizations vs. the link activation probability $p$ and for different noise levels $\sigma_w^2$. From top to bottom, we have K = 1, 5, and 10, respectively. ARMA filters handle stochasticity better for low orders while FIR filters perform better for K = 10.

Table 7.1: Square roots of (7.9) / (7.10) for the FIR / ARMA filters.

| *order* | $\sigma_w^2 = 10^{-6}$ | $\sigma_w^2 = 10^{-3}$ | $\sigma_w^2 = 10^{-1}$ |
|---|---|---|---|
| K=1 | 0.88 / 0.58 | 0.88 / 0.58 | 0.93 / 0.6 |
| K=3 | 0.58 / 6.9 | 0.58 / 6.9 | 0.66 / 7.2 |
| K=5 | 0.82 / 22.8 | 0.82 / 22.8 | 0.88 / 23.9 |

## 7.4. GRAPH SIGNAL DENOISING IN THE MEAN

Consider a noisy graph signal of the form $\mathbf{x} = \mathbf{x}_0 + \mathbf{w}$, with $\mathbf{x}_0$ the signal of interest and $\mathbf{w}$ the zero mean additive noise. The goal is to recover $\mathbf{x}_0$ using the a priori knowledge on its behavior w.r.t. the underlying graph. From the results in Section 4.2.4, Tikhonov denoising recovers $\mathbf{x}_0$ through an $\text{ARMA}_1$ graph filter. Here, the task can be solved without knowing the full graph topology since *i*) the filter coefficients can be found avoiding the eigendecomposition, i.e., only the value of $w$ is necessary; and *ii*) in a distributed implementation nodes need to know who are their neighbors to run the filter.

### 7.4.1. TIKHONOV GRAPH SIGNAL DENOISING IN THE MEAN

When multiple realizations of $\mathbf{x}$ are available (e.g., in a sensor network) one may consider generalizing the Tikhonov solution to improve the accuracy of signal denoising. In this case, we have for every realization $\mathbf{x}_t = \mathbf{x}_0 + \mathbf{w}_t$ with $\mathbf{x}_0$ equal to the desired signal. The graph signal denoising problem can then be performed *in the mean*. The related optimization problem becomes

$$\mathbf{x}_0^\star = \underset{\mathbf{x}_0 \in \mathbb{R}^N}{\operatorname{argmin}} \, \mathbb{E}\left[\|\mathbf{x}_0 - \mathbf{x}_t\|_2^2\right] + w\mathbf{x}_0^\mathsf{T}\mathbf{S}\mathbf{x}_0, \tag{7.19}$$

where we aim to solve it in the MSE sense over different time realizations. Analogous to (4.15), the optimal solution of (7.19) can still be written as an $\text{ARMA}_1$ filter of the form

$$\mathbf{x}_0^\star = \sum_{n=1}^{N} \frac{1}{1 + w\lambda_n} \mathbb{E}[\mathbf{x}_t]^\mathsf{H} \mathbf{u}_n \mathbf{u}_n, \tag{7.20}$$

which has two interpretations: (i) it can be considered the output of an $\text{ARMA}_1$ graph filter applied to the mean $\bar{\mathbf{x}}_t$ or (ii) the mean output of an $\text{ARMA}_1$ applied to $\mathbf{x}_t$.

**i) Disjoint average and denoise (DAD).** One approach for solving (7.20) is first to do a running average up to time $t$, independently at each node $\bar{x}_{t,i} = \frac{1}{t}((t-1)\bar{x}_{t-1,i} + x_{t,i})$, and then run a recursive $\text{ARMA}_1$ on the average signal $\bar{\mathbf{x}}_t = [\bar{x}_{t,1}, \ldots, \bar{x}_{t,N}]^\mathsf{T}$. Hence, the recursions are not in time, but carried out for every $t$ separately. This procedure is summarized in Algorithm 7.1 for $\mathbf{S} = \mathbf{L}$. Since the noise is zero-mean, the local average [cf. line 3 of Algorithm 7.1] reduces the noise level proportionally to the number of collected samples, effectively facilitating the work of graph denoising. However, this algorithm needs to run the graph filter till convergence for each available sample of the input signal $\mathbf{x}_t$. We refer to this approach as *disjoint average and denoise*.

**Algorithm 7.1.** *Disjoint average and denoise*

1: Given $w$, initialize $\psi$, $\varphi$, $\mathbf{y}_0 = \mathbf{0}_N$ and the ARMA$_1$ iterations $I$
2: **for** each $t > 0$
3: *compute local average* $\bar{x}_{t,i} = \frac{1}{t}((t-1)\bar{x}_{t-1,i} + x_{t,i})$
4: **procedure** COMPUTE THE ESTIMATE OF $[\mathbf{x}_0^\star]_{t,i}$
5: **for** $\iota = 1$ **to** $I$
6: *collect* $y_j[\iota-1]$ *from all neighbors* $j \in \mathcal{N}_i$
7: *compute* $y_i[\iota] = \psi \sum_{j\in\mathcal{N}_i} W_{i,j}\left(y_i[\iota-1] - y_j[\iota-1]\right) + \varphi\bar{x}_{t,i}$
8: *send* $y_i[\iota]$ *to all neighbors* $\mathcal{N}_i$
9: **end for**
10: *set* $[\mathbf{x}_0^\star]_{t,i} = y_i[I]$
11: **end for**

**ii) Joint denoise in the mean.** The complexity of denoising in the mean can be reduced by performing the local averaging and the denoising jointly, an approach which we refer to as *joint denoising in the mean.* In virtue of the results in Section 7.3, we distinguish between three different cases to do the local averaging.

*ii − a*) The more general case of joint denoising in the mean with *input-output averaging* (JDMIOA), where a regular ARMA filter[3] is run in time with an input signal at time $t$ the local *running average* of the history of $\mathbf{x}_t$ and as the final solution of (7.20) at node $i$ the local *running average* of the filter output up to time $t$, i.e., $[\mathbf{x}_0^\star]_{t,i} = 1/t((t-1)[\mathbf{x}_0^\star]_{t-1,i} + y_{t,i})$. This procedure is summarized in Algorithm 7.2 for $\mathbf{S} = \mathbf{L}$.

*ii − b*) The particular case of joint denoising in the mean with *input averaging* (JDMIA), which differently from JDMIOA sets the final solution of (7.20) at node $i$ as the filter output at time $t$, i.e., in Algorithm 2 this differs only in line 8 which should be "*set* $[\mathbf{x}_0^\star]_{t,i} = y_{t,i}$". Note that this procedure is a direct online implementation of the DAD algorithm.

*ii − c*) The particular case of joint denoising in the mean with *output averaging* (JDMOA), which differently from the JDMIOA has as input signal at time $t$ the collected samples, $\mathbf{x}_t = \mathbf{x}_0 + \mathbf{w}_t$. Referring again to Algorithm 7.2, this differs only in line 3 which should be "*collect the local sample* $x_{t,i} = [\mathbf{x}_0]_i + w_{t,i}$".

Despite being particular cases of JDMIOA, the JDMIA and JDMOA perform differently depending on the scenario. Note that the joint denoising in the mean approaches follow the same filtering strategy as a classical graph signal denoising filter, and thus do not need extra memory, computations or communication efforts. Again, the noise being zero mean, averaging the filter input and/or output reduces the noise power that will be introduced by the filter in every successive iteration while performing the online denoising in the mean.

*iii*) **Local averaging (LA).** As a benchmark we consider the local averaging, i.e., the nodes ignore the graph and set $[\mathbf{x}_0^\star]_t = 1/t((t-1)[\mathbf{x}_0^\star]_{t-1} + \mathbf{x}_t)$.

We conclude this section stressing that the above ideas are not restricted to denoising or simple ARMA$_1$ filtering. They can be used for any filter, ARMA$_K$ or FIR$_K$, as long as

[3] From Section 4.2.4, the ARMA$_1$ enjoys a stable implementation for all $w > 0$.

**Algorithm 7.2.** *Joint denoise in the mean with input-output averaging (JDMIOA)*

1: Given $w$, initialize $\psi$, $\varphi$ and $\mathbf{y}_0 = \mathbf{0}_N$
2: **for** each $t > 0$
3: *compute local average* $\bar{x}_{t,i} = \frac{1}{t}((t-1)\bar{x}_{t-1,i} + x_{t,i})$
4: **procedure** COMPUTE THE ESTIMATE OF $[\mathbf{x}_0^\star]_{t,i}$
5: *collect* $y_{t-1,j}$ *from all neighbors* $j \in \mathcal{N}_i$
6: *compute* $y_{t,i} = \psi \sum_{\in \mathcal{N}_i} W_{i,j}\left(y_{t-1,i} - y_{t-1,j}\right) + \varphi \bar{x}_{t,i}$
7: *send* $y_{t,i}$ *to all neighbors* $\mathcal{N}_i$
8: *set* $[\mathbf{x}_0^\star]_{t,i} = \frac{1}{t}((t-1)[\mathbf{x}_0^\star]_{t-1,i} + y_{t,i})$
9: **end for**

multiple noisy observations of the signal are available.

### 7.4.2. NUMERICAL RESULTS

Under the same scenario of Section 7.3.4, consider a graph signal $\mathbf{x}_t = \mathbf{x}_0 + \mathbf{w}_t$, where $\mathbf{x}_0$ varies smoothly over the graph. One way to generate these signals is by an exponentially decaying spectrum, i.e., $\mathbf{x}_0^{\mathsf{H}}\mathbf{u}_n = \exp(-25\lambda_n)$. The noise follows a normal distribution with i.i.d. realizations and variance $\sigma_w^2 = 1$.

We illustrate the denoising in the mean problem and compare all joint approaches with both the DAD and the LA approach. To quantify the performance, we calculate the error between the filter output and the signal $\mathbf{x}_0$ at each node as in (7.18).

There is a subtle yet crucial difference between the inner-workings of the three compared algorithms: LA and joint denoising in the mean are online algorithms, meaning that the algorithmic output is obtained in real-time[4]. The error depicted in the figures at time $t$ is therefore computed w.r.t. the denoising output at $t$. The DAD algorithm performs the averaging and denoising steps in sequence and solves one filtering problem for each computed local average (i.e., at each iteration $t$). Since each disjoint filtering requires enough iterations to converge, the error depicted in the figure at $t$ is in this case is w.r.t. the filter output at convergence, occurring much later than $t$ (we took $t + 100$ in the experiments, but also $t + 10$ can be considered as a choice).

In Figure 7.2(a) we compare the LA (solid black line), the DAD and all three joint denoising in the mean algorithms as a function of time for Tikhonov denoising. Here $w$ is 0.5. The first thing to notice is that incorporating the graph knowledge via the smoothness outperforms the LA. The role of the graph is more important when fewer samples are available, i.e., small values of $t$. Second, while the JDMIA immediately approaches the DAD (being it an online implementation of the latter), the JDMIOA and JDMOA for $t \leq 10$ perform better. Then, when more samples become available, the JDMOA matches the DAD. The JDMIOA, on the other hand, prioritizes the noise reduction over the smoothness prior due to the double averaging of the filter input and output (lines 3 and 8 in Algorithm 7.2, respectively). Thus, when more samples are available the related $\bar{\sigma}_e$ decays as that of the LA algorithm. To avoid overcrowded plots, in the sequel,

[4] In the experiments, the signal changes with the same rate as the iteration rate. We use the term iteration or more simply time to refer to both concepts.

we do not illustrate show the JDMIA, since it consistently matches the DAD algorithm.

Figure 7.2(b) compares the JDMIOA, the JDMOA, and the DAD algorithm when the weight given to the smoothness prior is twice that of the denoising, i.e., $w = 2$. As we can see, considering the graph structure (i.e., $w$ is higher w.r.t. Figure 7.2(a)) improves the performance as it exploits the smoothness prior to remove the noise. Once again, when $t$ increases the JDMOA approaches the DAD algorithm, while the JDMIOA performs between the latter and the LA.

Figure 7.2(c) compares the algorithms for $w = 20$. In this case, when numerous samples are available, a large $w$ may overly prioritize the graph smoothness resulting in a performance degradation. Since the noise becomes negligible, the smoothness prior is only useful when the number of samples is limited ($t < 100$). This is not only an issue of the joint approach but of graph signal denoising in general.

This effect is further illustrated in Figure 7.2(d) for $\sigma_w^2 = 10^{-4}$ and $w = 0.5$. Here, also $w = 0.5$ gives more weight than necessary to the smoothness thus resulting in a lower performance than the pure LA. This includes also the JDMIOA, which suggests that a Tikhonov prior should be considered in high noise regimes.

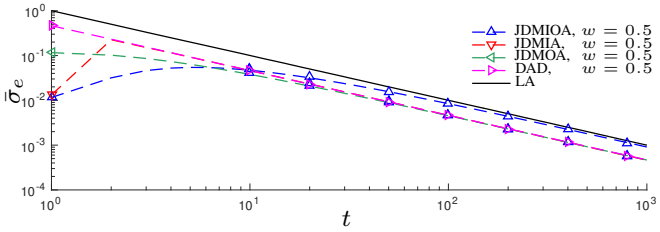
As a general observation w.r.t. the DAD algorithm, we say that even though we compute the joint denoising in the mean online, i.e., the graph filter output is averaged for every collected sample, our approach gives a better performance than the offline alternative when the number of samples is limited. The initial transient behavior is because the filters are initialized to zero. On the other hand, both approaches perform the same when the number of samples increases.

Solving the joint Tikhonov denoising in the mean leads to a reduced computational and communication complexity as compared to the disjoint approach. This is because the joint approach does not require many iterations between the nodes to reach a reasonable performance. For the same reason, the joint denoising is more efficient also in terms of latency.

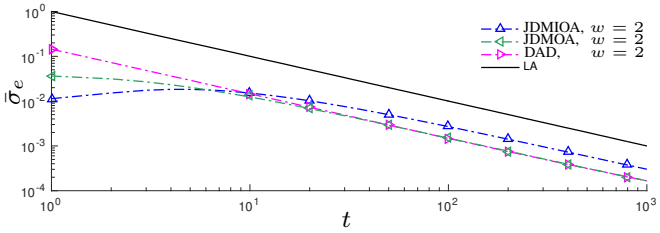
As a particular case, the pure Tikhonov denoising error can be seen for $t = 1$, when only one signal realization is available, in the DAD algorithm. We see that taking multiple realizations into account can improve the performance up to one order of magnitude in only 10 iterations (Figure 7.2 (b), $w = 2$). To conclude, notice that the proposed approach even with a node variance of $\sigma_w^2 = 1$ and with 10 samples, can achieve an error of $10^{-2}$ for the signal of interest in only 10 iterations.

## 7.5. STOCHASTICALLY SPARSIFIED GRAPH FILTERING

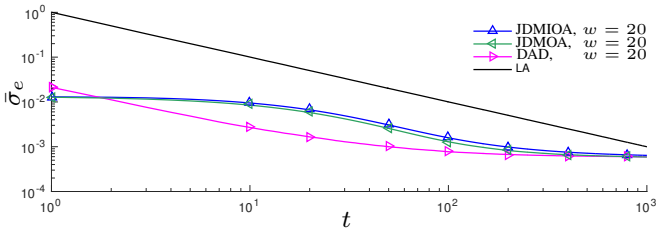
To decrease the communication and computational complexity of graph filtering, we propose, instead of filtering a deterministic signal $\mathbf{x}$ on the deterministic graph $\mathcal{G}$, to filter $\mathbf{x}$ using realizations of a sparser random graph $\mathcal{G}_t$ obtained as a random edge sampling of $\mathcal{G}$. This means that the filtering will be performed on a sequence of time-varying graphs which abide to the RES graph model with a uniform probability $p \in (0,1]$ for all links. What we prove in the following is that, when the filter coefficients are changed accordingly, the filter output in $\mathcal{G}_t$ will be statistically close (the first and second moments of the error are zero and bounded, respectively) to the original filter in $\mathcal{G}$. Our results im-



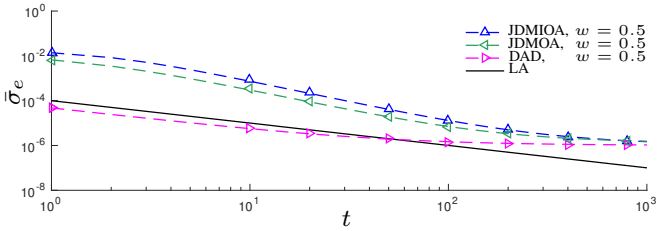

(a)




(b)




(c)




(d)

Figure 7.2: Standard deviation of the denoising algorithms for different time instances. (a) compares the joint denoising in the mean algorithms with the DAD and LA algorithms for $w = 0.5$ and $\sigma_w^2 = 1$. (b) shows the performance for $w = 2$ (i.e., where smoothness is prioritised over denoising) and $\sigma_w^2 = 1$. (c) compares the algorithms for $w = 20$ and $\sigma_w^2 = 1$. Finally, in (d) is shown the error standard deviation for $w = 0.5$ and $\sigma_w^2 = 10^{-4}$.

ply a reduction in the communication and computational complexity of graph filtering that is linear in $p$.

Denote by $\mathbf{S}$ the shift operator of the deterministic graph $\mathcal{G}$, where (for now) $\mathbf{S}$ is chosen as the discrete Laplacian $\mathbf{L}$.

### 7.5.1. SPARSIFIED FIR GRAPH FILTERS

With a slight modification of the results in Section 7.3 (since the graph signal is now deterministic), we find that in expectation the filter output will be

$$\begin{aligned}\bar{\mathbf{y}}_{t+1}^{(\mathrm{s})} = \mathbb{E}\left[\mathbf{y}_{t+1}^{(\mathrm{s})}\right] &= \mathbb{E}\left[\sum_{k=0}^{K} \phi_k^{(\mathrm{s})} \mathbf{\Phi}_{\mathbf{S}}(t, t-k+1)\mathbf{x}\right] \\ &= \sum_{k=0}^{K} \phi_k^{(\mathrm{s})} \bar{\mathbf{S}}^k \mathbf{x} = \sum_{k=0}^{K} \phi_k^{(\mathrm{s})} (p\mathbf{S})^k \mathbf{x}, \end{aligned} \tag{7.21}$$

where $\mathbf{y}_{t+1}^{(\mathrm{s})}$ is the stochastic sparsification output and $\phi_k^{(\mathrm{s})}$ are the new coefficients. Therefore, if each coefficient $\phi_k^{(\mathrm{s})}$ is $\phi_k$ scaled by $p^{-k}$, the expected output will be identical to what would have been obtained if the original FIR filter was used in $\mathcal{G}$

$$\mathbb{E}\left[\mathbf{y}_{t+1}^{(\mathrm{s})}\right] = \mathbf{y}_{t+1} \quad \text{for} \quad \phi_k^{(\mathrm{s})} = \phi_k p^{-k}, \tag{7.22}$$

or equivalently, the expected sparsification error is zero

$$\mathbb{E}\left[\mathbf{y}_{t+1}^{(\mathrm{s})} - \mathbf{y}_{t+1}\right] = \mathbf{0}_N. \tag{7.23}$$

The above expression implies that we can linearly reduce the communication and computational complexity of FIR graph filters from $O(MK)$ to $O(pMK)$. To see how $p$ influences how far the error can lie from the mean, in the following, we derive an upper bound on the average variance of the output signal. From Proposition 7.3, we have

$$\overline{\mathrm{var}}[\mathbf{y}_{t+1}^{(\mathrm{s})}] \leq (\boldsymbol{\varrho}^{\mathsf{T}} \boldsymbol{\phi}^{(\mathrm{s})})^2 \left(\overline{\mathrm{var}}[\mathbf{x}] + \frac{\|\bar{\mathbf{x}}\|_2^2}{N}\right) = (\mathbf{r}^{\mathsf{T}} \boldsymbol{\phi})^2 \frac{\|\mathbf{x}\|_2^2}{N}, \tag{7.24}$$

with $\mathbf{r} = [(\varrho/p)^0, (\varrho/p)^1, \ldots, (\varrho/p)^K]^{\mathsf{T}}$. Though the provided bound is not tight (as is witnessed by the fact that for $p = 1$ the variance is not zero), it illustrates the impact of $1/p$ on the error variance.

### 7.5.2. SPARSIFIED ARMA GRAPH FILTERS

The expected sparsified output $\bar{\mathbf{y}}_{t+1}^{(\mathrm{s})}$ of the parallel ARMA$_K$ filter is

$$\begin{aligned}\bar{\mathbf{y}}_{t+1}^{(\mathrm{s},k)} &= \psi^{(\mathrm{s},k)} (p\mathbf{S}) \bar{\mathbf{y}}_t^{(\mathrm{s},k)} + \varphi^{(k)} \mathbf{x} \\ \bar{\mathbf{y}}_{t+1}^{(\mathrm{s})} &= \sum_{k=1}^{K} \bar{\mathbf{y}}_{t+1}^{(\mathrm{s},k)}, \end{aligned} \tag{7.25}$$

where again the superscript (s) refers to the sparsified filtering. Notice that as long as $\psi^{(s,k)} = \psi^{(k)}/p$ the expected output of (7.25) is identical to the same ARMA filter operating on the complete graph. This sparsification will reduce the communication and computational complexity by the same order as for the FIR. Further, from Theorem 7.3, the average variance of the sparsified filter is upper bounded by

$$\lim_{t\to\infty} \overline{\text{var}}[\mathbf{y}_{t+1}^{(s)}] \leq \frac{K\|\boldsymbol{\varphi}\|_2^2\|\mathbf{x}\|_2^2}{N(1-\varrho|\frac{\psi_{\max}}{p}|)^2} \tag{7.26}$$

As for the FIR, the bound (7.26) does not reach zero for $p = 1$, but it shows that the variance is upper bounded by a quadratic rate of $1/p$.

We conclude this section with the following observations.

**Remark 7.5.** *Our numerical simulations show that even though in practice the sparsification error variance is much smaller than* (7.24) *and* (7.26)*, $p$ has indeed the claimed impact on the output variance.*

**Remark 7.6.** *The above theoretical derivations provide a useful way to change the filter coefficients, to avoid a bias in the sparsified approach. These derivations stand only for the discrete Laplacian $\mathbf{S} = \mathbf{L}$ (or its translated version $\mathbf{S} = \mathbf{L} - \lambda_{max}(\mathbf{L})/2\mathbf{I}_N$), meaning that the expected graph Laplacian can be expressed as a scaled version of the Laplacian of the underlying graph $\mathcal{G}$. This is not the case for the normalized Laplacian $\mathbf{L}_n$ (or its translated version $\mathbf{S} = \mathbf{L}_n - \mathbf{I}_N$). Finding a closed form expression of the mean normalized Laplacian is a challenging task. Even if it can be found, its $(i,j)$th entry will depend on the node degrees in $\mathcal{G}$, which renders formulating $\bar{\mathbf{L}}_n$ as a scaled version of the normalized Laplacian $\mathbf{L}_n$ difficult. However, the (translated) normalized Laplacian is useful to improve the stability region of the ARMA filter and in some cases it provides FIR coefficients with lower magnitude (and as a result leads to a lower error variance). As we will see in the simulation results, $\mathbf{S} = \mathbf{L}_n - \mathbf{I}_N$, even without changing the filter coefficients, will lead to an error with a small mean and variance.*

### 7.5.3. NUMERICAL RESULTS

We considered the same settings of Section 7.3.4 with a graph signal characterized by a white unitary graph spectrum on the graph $\mathcal{G}$. We derived the results for both the FIR and ARMA graph filters for $K = 1, 3, 5, 7$ designed as ideal low-pass filters with the cut-off frequency equal to half the signal bandwidth. Further, the same error measuring criterion as in (7.17) is used, where $\mathbf{y}^{(s)}$ indicates the filter output of the sparsified filter.

Since choices of $\mathbf{S}$ with a low spectral norm are preferred to have a lower variance of the filter output and a larger stability region for the ARMA filters (and as a result higher approximation accuracy), we perform the results for two different scaled shift operators, i.e., $\mathbf{S} = \mathbf{L}_n - \mathbf{I}_N$ and $\mathbf{S} = \frac{1}{\lambda_{\max}(\mathbf{L})}\mathbf{L} - 0.5\mathbf{I}_N$. In case the latter Laplacian is used, the filter coefficients can be changed according to the above results. Meanwhile, for $\mathbf{S} = \mathbf{L}_n - \mathbf{I}_N$ this cannot be done since we do not have a closed form expression of the expected value. In this instance, we run the filters with the same filter coefficients as used in calculating the deterministic output, i.e., they do not change. As a consequence, the zero mean error is not guaranteed.

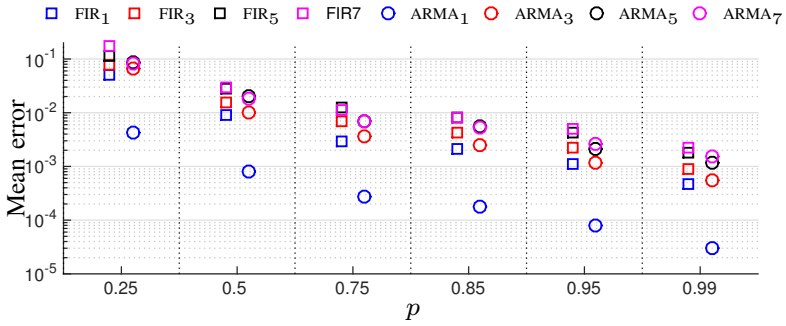



Figure 7.3: Mean error over all nodes and realizations between sparsified and deterministic output for different values of $p$ when $\mathbf{S} = \mathbf{L}_{\text{n}} - \mathbf{I}_N$.

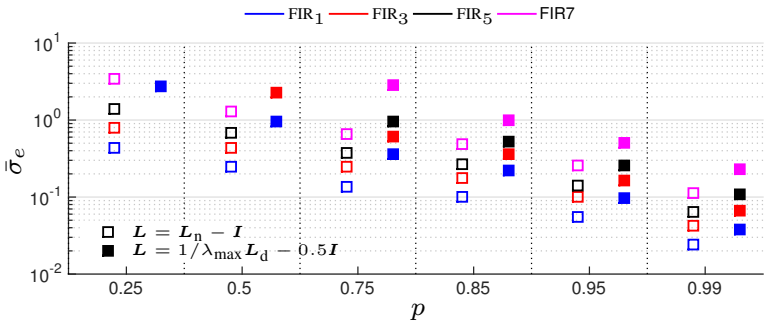



Figure 7.4: Average empirical standard deviation among all nodes of the error between the sparsified FIR filter output and the original graph output for different values of $p$. The results are shown for two different Laplacians. For $\mathbf{S} = \mathbf{L}_{\text{n}} - \mathbf{I}_N$, the filter coefficients are not changed.

Our aim is to show simulation results when the desired operation is not carried out over the given deterministic graph $\mathcal{G}$, but on its sparsified version. In Figure 7.3, we illustrate the mean error for different values of $p$ and for $\mathbf{S} = \mathbf{L}_{\text{n}} - \mathbf{I}_N$. We see that the introduced bias is negligible, and it reduces furthermore when the filter order is lower. Fori $p \geq 0.5$, the mean error is smaller than $10^{-2}$.

Figures 7.4 and 7.5 show the empirical standard deviation of the considered error for different values of $p$. As a common phenomenon for all filters, the standard deviation of the error reduces with higher values of $p$. Further, for all filters, a lower filter order yields a lower standard deviation. A notable result is observed for the $\text{ARMA}_1$ filter, where we can solve tasks like Tikhonov denoising, signal interpolation under smoothness assumptions, some Wiener based denoising, and graph diffusion processes, with communication and computational costs reduced by 75% with little or no error. Indeed, for $p = 0.25$, the average standard deviation is $\bar{\sigma}_e \approx 0.035$ and a mean error smaller than 0.004. The latter shows that the signal output got in the sparsified graph is closer to the same output obtained operating on underlying graph deterministically. For higher filter

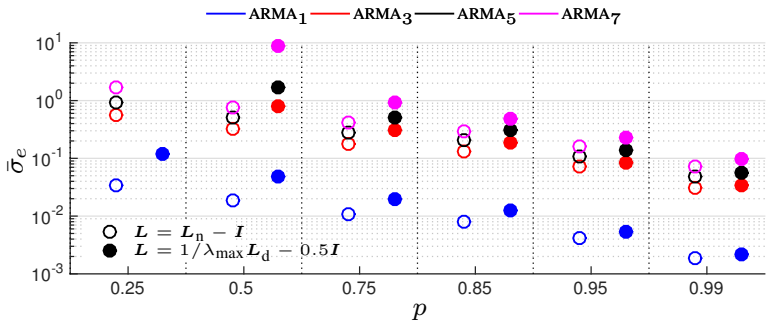



Figure 7.5: Average empirical standard deviation among all nodes of the error between the sparsified ARMA filter output and the original graph output for different values of $p$. The results are shown for two different Laplacians. For $\mathbf{S} = \mathbf{L}_{\rm n} - \mathbf{I}_N$ the filter coefficients are not changed. Notice that for the ARMA$_1$ filter (useful for diffusion, interpolation, and denoising) we can save up to 75% of communication and computational costs.

orders, we suggest $p$ should be around 0.75 to achieve a reasonable performance.

Similar to Figure 7.1, the ARMA filters give a lower $\bar{\sigma}_e$ than the FIR filters for the presented orders, while for higher values of $K$ the FIR filters perform better. To conclude, note that $\mathbf{S} = \mathbf{L}_{\rm n} - \mathbf{I}_N$ without changing the filter coefficients yields a favorable performance which suggests that the normalized Laplacian is a good choice to be robust to link fluctuations.

## 7.6. CONCLUDING REMARKS



In this chapter, we introduced a statistical analysis of graph-time filtering. Due to the practical necessity to deal with randomness in the graph topology (e.g., edge fluctuation) and graph signal (e.g., noise), we first performed a stochastic characterization for both FIR and ARMA graph filters. Under the assumption that the graph randomness is independent of the graph signal, both the FIR and the ARMA graph filters behaved in the mean as the same deterministic filter, having as input the mean signal, operating on a deterministic graph being the expected graph. We further showed that when the graph signal is a stochastic process with a time-varying mean and covariance, the graph filters operated as a two-dimensional filter and they captured the variations in the mean of the graph signal on the expected graph and time jointly. For both FIR and ARMA graph filters, we proved the variance of the filter output is upper bounded and numerical results showed that the empirical variance in different simulated scenarios is small. Thus the signal realizations stayed close to the expected value.

The chapter is concluded by leveraging stochasticity for the tasks of graph signal denoising in the mean and stochastically sparsified graph filtering. Through an ARMA$_1$ graph filter, we showed an online recursive approach to clean signal by exchanging time-varying signals with the neighbors. In addition, both graph filters (FIR and ARMA) can be performed in a stochastic way leading to amenable savings in terms of distributed communication and computation costs, with very little difference from the determinis-

tic setting.

# APPENDICES

## 7.A. PROOF OF THE FIR$_K$ EXPECTED OUTPUT PROPOSITION

By applying the expectation operator to (7.1) and considering that the realizations of the graph topology and graph signal are independent we have

$$\bar{\mathbf{y}}_{t+1} = \mathbb{E}\left[\sum_{k=0}^{K} \phi_k \mathbf{\Phi}_{\mathbf{S}}(t, t-k+1)\mathbf{x}_{t-k+1}\right] \tag{7.27}$$

$$= \sum_{k=0}^{K} \phi_k \left(\prod_{\tau=t}^{t-k+1} \mathbb{E}[\mathbf{S}_\tau]\right) \mathbb{E}[\mathbf{x}_{t-k+1}] = \sum_{k=0}^{K} \phi_k \bar{\mathbf{S}}^k \bar{\mathbf{x}}, \tag{7.28}$$

where $\bar{\mathbf{y}}_{t+1}$ is the constant expected value of $\mathbf{y}_{t+1}$ after $K$ time steps. In (7.27) we substitute the back going product $\mathbf{\Phi}_{\mathbf{S}}(t, t-k+1)$ and apply the linearity of the expectation.

## 7.B. PROOF OF THE PARALLEL ARMA$_K$ EXPECTED OUTPUT THEOREM

Recursion (7.3) can be differently written as

$$\mathbf{y}_{t+1}^{(1:K)} = (\mathbf{\Psi} \otimes \mathbf{S}_t)\mathbf{y}_t^{(1:K)} + \boldsymbol{\varphi} \otimes \mathbf{x}_t \tag{7.29a}$$

$$\mathbf{y}_{t+1} = (\mathbf{1}^\mathsf{T} \otimes \mathbf{I}_N)\mathbf{y}_{t+1}^{1:K}, \tag{7.29b}$$

with $\mathbf{y}_t^{(1:K)} = \left[\mathbf{y}_t^{(1)\mathsf{T}}, \mathbf{y}_t^{(2)\mathsf{T}}, \cdots, \mathbf{y}_t^{(K)\mathsf{T}}\right]^\mathsf{T}$ the $NK \times 1$ stacked state vector, $\mathbf{\Psi} = \text{diag}(\psi^{(1)}, \psi^{(2)}, \cdots, \psi^{(K)})$ a diagonal $K \times K$ coefficient matrix, $\boldsymbol{\varphi} = [\varphi^{(1)}, \varphi^{(2)}, \cdots, \varphi^{(K)}]^\mathsf{T}$ a $K \times 1$ coefficient vector, and $\mathbf{1}_K$ the $K \times 1$ one-vector. Then, in the following we consider that all the branches of the parallel filter bank are stable, i.e., $|\psi^{(k)}| = \|\mathbf{\Psi}\|_\infty < 1/\rho$ imposed in the filter design phase. Then, by using the linearity of the expectation, and the independency among time between the realizations of the random variables we rewrite (7.29) as

$$\bar{\mathbf{y}}_{t+1}^{(1:K)} = \mathbb{E}[(\mathbf{\Psi} \otimes \mathbf{S}_t)]\,\bar{\mathbf{y}}_t^{(1:K)} + \mathbb{E}\left[\boldsymbol{\varphi} \otimes \mathbf{x}_t\right] \tag{7.30a}$$

$$\bar{\mathbf{y}}_{t+1} = (\mathbf{1}^\mathsf{T} \otimes \mathbf{I}_N)\bar{\mathbf{y}}_{t+1}^{(1:K)}, \tag{7.30b}$$

where $\bar{\mathbf{y}}_{t+1}$ and $\bar{\mathbf{y}}_{t+1}^{(1:K)}$ denote the expected values of $\mathbf{y}_{t+1}$ and $\mathbf{y}_{t+1}^{(1:K)}$, respectively. Then by using once again the independency of the graph realizations and graph signal together with the properties of the Kronecker product we rewrite (7.30) as

$$\bar{\mathbf{y}}_{t+1}^{(1:K)} = (\mathbf{\Psi} \otimes \bar{\mathbf{S}})\bar{\mathbf{y}}_t^{(1:K)} + \boldsymbol{\varphi} \otimes \bar{\mathbf{x}} \tag{7.31a}$$

$$\bar{\mathbf{y}}_{t+1} = (\mathbf{1}_K^\mathsf{T} \otimes \mathbf{I}_N)\bar{\mathbf{y}}_{t+1}^{(1:K)}, \tag{7.31b}$$

with $\bar{\mathbf{x}}$ being the expected values of $\mathbf{x}_t$. Then we expand (7.31) to express it as a function of the initial condition $\mathbf{y}_0^{(1:K)}$ and the inputs as

$$\begin{aligned}\bar{\mathbf{y}}_{t+1}^{(1:K)} &= \left(\boldsymbol{\Psi}\otimes\bar{\mathbf{S}}\right)^{t+1}\mathbf{y}_0^{(1:K)} + \sum_{\tau=0}^{t}\left(\boldsymbol{\Psi}\otimes\bar{\mathbf{S}}\right)^{\tau}\left(\boldsymbol{\varphi}\otimes\bar{\mathbf{x}}\right)\\ &= \left(\boldsymbol{\Psi}^{t+1}\otimes\bar{\mathbf{S}}^{t+1}\right)\mathbf{y}_0^{(1:K)} + \sum_{\tau=0}^{t}\left(\boldsymbol{\Psi}^{\tau}\boldsymbol{\varphi}\right)\otimes\left(\bar{\mathbf{S}}^{\tau}\bar{\mathbf{x}}\right),\end{aligned} \tag{7.32}$$

where we have used the Kronecker product property

$$(\mathbf{A}\otimes\mathbf{B})(\mathbf{C}\otimes\mathbf{D}) = (\mathbf{AC})\otimes(\mathbf{BD}). \tag{7.33}$$

By using once again the stability condition for all the branches, we have

$$\lim_{t\to\infty}\|\left(\boldsymbol{\Psi}^{t+1}\otimes\bar{\mathbf{S}}^{t+1}\right)\mathbf{y}_0^{(1:K)}\|_2 = 0. \tag{7.34}$$

Hence, the limiting steady state of the expected value can be written as

$$\begin{aligned}\bar{\mathbf{y}} &= \lim_{t\to\infty}\sum_{\tau=0}^{t}(\mathbf{1}_K^{\mathsf{T}}\otimes\mathbf{I}_N)\left(\boldsymbol{\Psi}^{\tau}\boldsymbol{\varphi}\right)\otimes\left(\bar{\mathbf{S}}^{\tau}\bar{\mathbf{x}}\right)\\ &= \lim_{t\to\infty}\sum_{\tau=0}^{t}\left(\mathbf{1}^{\mathsf{T}}\boldsymbol{\Psi}^{\tau}\boldsymbol{\varphi}\right)\otimes\left(\bar{\mathbf{S}}^{\tau}\bar{\mathbf{x}}\right)\\ &= \lim_{t\to\infty}\sum_{\tau=0}^{t}\sum_{k=1}^{K}\varphi^{(k)}\left(\psi^{(k)}\bar{\mathbf{S}}\right)^{\tau}\bar{\mathbf{x}}\end{aligned} \tag{7.35}$$

where once again we made use of (7.33) and expressed the Kronecker product as the sum of $K$ terms. By leveraging once again the fact that all the branches of the filter bank are stable ($|\psi^{(k)}| < 1/\rho$), we rewrite (7.35) as

$$\bar{\mathbf{y}} = \sum_{k=1}^{K}\varphi^{(k)}\left(\mathbf{I}_N - \psi^{(k)}\bar{\mathbf{S}}\right)^{-1}\bar{\mathbf{x}} \tag{7.36}$$

which proves the first part (7.4). Then from (7.32) we can see that the dependency on the initial state $\mathbf{y}_0^{(1:K)}$ decreases exponentially with $t$, thus we can say that (7.3) converges linearly to the limiting steady state value of the expected value (7.4).

## 7.C. PROOF OF THE FIR$_K$ EXPECTED OUTPUT WITH NON-STATIOIONARY INPUT PROPOSITION

By using the same arguments as in the proof of Proposition 7.1, one finds that the output of an FIR$_K$ is in expectation

$$\bar{\mathbf{y}}_{t+1} = \sum_{k=0}^{K}\phi_k\bar{\mathbf{S}}^k\bar{\mathbf{x}}_{t-k+1}. \tag{7.37}$$

Then, by projecting the signal using the GFT into the subspace spanned by an eigenvector $\mathbf{u}_n$ of $\bar{\mathbf{S}}$ with associated eigenvalue $\lambda_n$ (7.37) is reduced to

$$\bar{y}_{t+1} = \sum_{k=0}^{K} \phi_k \lambda_n^k \bar{x}_{t-k+1}, \tag{7.38}$$

where $\bar{x}_t = \mathbf{u}_n^{\mathsf{H}} \bar{\mathbf{x}}_t$ and $\bar{y}_{t+1} = \mathbf{u}_n^{\mathsf{H}} \bar{\mathbf{y}}_{t+1}$ are respectively the magnitude of the projections of the filter input and output on the chosen subspace. By taking the Z-transform and dividing both sides by $z^{t+1}$ the claim (7.6) follows.

## 7.D. PROOF OF THE ARMA$_K$ EXPECTED OUTPUT WITH NON-STATIOIONARY INPUT THEOREM

By rewriting the parallel ARMA$_K$ in the form (7.29) and by following the same procedure as in the derivation of (7.31), we have that the expected ARMA output at time instant $t+1$ is

$$\begin{aligned} \bar{\mathbf{y}}_{t+1}^{(1:K)} &= \mathbb{E}\left[ (\boldsymbol{\Psi} \otimes \mathbf{S}_t) \mathbf{y}_t^{(1:K)} + \boldsymbol{\varphi} \otimes \mathbf{x}_t \right] = (\boldsymbol{\Psi} \otimes \bar{\mathbf{S}}) \bar{\mathbf{y}}_t^{(1:K)} + \boldsymbol{\varphi} \otimes \bar{\mathbf{x}}_t \\ \bar{\mathbf{y}}_{t+1} &= \mathbb{E}\left[ (\mathbf{1}_K^{\mathsf{T}} \otimes \mathbf{I}_N) \mathbf{y}_{t+1}^{(1:K)} \right] = (\mathbf{1}_K^{\mathsf{T}} \otimes \mathbf{I}_N) \bar{\mathbf{y}}_{t+1}^{(1:K)}. \end{aligned} \tag{7.39}$$

In analogy to Theorem 6.1 in Chapter 6, (7.39) represents a deterministic two-dimensional ARMA$_K$ filter with deterministic time-varying input signal $\bar{\mathbf{x}}_t$, filter memory $\bar{\mathbf{y}}_t^{(1:K)}$ and output $\bar{\mathbf{y}}_t$ operating over the deterministic time-invariant graph $\bar{\mathbf{S}}$. With these analogies (7.7) can be proven from Theorem 6.1 in Chapter 6.

## 7.E. PROOF OF THE FIR$_K$ VARIANCE BOUND PROPOSITION

We start by computing the trace of the covariance matrix of the filter output at time instant $t+1$ as

$$\mathrm{tr}(\boldsymbol{\Sigma}_y[t+1]) = \mathrm{tr}(\mathbb{E}[\mathbf{y}_{t+1}\mathbf{y}_{t+1}^{\mathsf{H}}]) - \mathrm{tr}(\mathbb{E}[\mathbf{y}_{t+1}]\mathbb{E}[\mathbf{y}_{t+1}]^{\mathsf{H}}). \tag{7.40}$$

Using the linearity of the expectation, the first term on the right hand side of (7.40) can be expanded as

$$\mathrm{tr}\left( \mathbb{E}[\mathbf{y}_{t+1}\mathbf{y}_{t+1}^{\mathsf{H}}] \right) = \sum_{k=0,l=0}^{K} \phi_k \phi_\ell T(k,\ell), \tag{7.41}$$

where

$$T(k,\ell) = \mathrm{tr}\left( \mathbb{E}\boldsymbol{\Phi}_{\mathbf{S}}(t,t-k+1)\mathbf{x}_{t-k+1}\mathbf{x}_{t-\ell+1}^{\mathsf{H}}\boldsymbol{\Phi}_{\mathbf{S}}(t,t-l+1)^{\mathsf{H}}] \right). \tag{7.42}$$

To proceed, note that by the commutativity of the trace with respect to the expectation and using the cyclic property of the trace, we can write

$$\begin{aligned} T(k,\ell) &= \mathbb{E}\left[ \mathrm{tr}\left( (\boldsymbol{\Phi}_{\mathbf{S}}(t,t-l+1)^{\mathsf{H}} \boldsymbol{\Phi}_{\mathbf{S}}(t,t-k+1)\mathbf{x}_{t-k+1}\mathbf{x}_{t-\ell+1}^{\mathsf{H}} \right) \right] \\ &= \mathrm{tr}\left( \mathbb{E}\left[ \boldsymbol{\Phi}_{\mathbf{S}}(t,t-l+1)^{\mathsf{H}} \boldsymbol{\Phi}_{\mathbf{S}}(t,t-k+1) \right] \mathbb{E}\left[ \mathbf{x}_{t-k+1}\mathbf{x}_{t-\ell+1}^{\mathsf{H}} \right] \right), \end{aligned} \tag{7.43}$$

with matrix

$$\mathbb{E}\left[\mathbf{x}_{t-k+1}\mathbf{x}_{t-\ell+1}^{\mathsf{H}}\right] = \begin{cases} \boldsymbol{\Sigma}_x + \bar{\mathbf{x}}\bar{\mathbf{x}}^{\mathsf{H}}, & \text{if } k = \ell, \\ \bar{\mathbf{x}}\bar{\mathbf{x}}^{\mathsf{H}}, & \text{otherwise,} \end{cases} \tag{7.44}$$

being positive semi-definite. We can therefore use inequality

$$\operatorname{tr}(\mathbf{AB}) \leq \frac{\|\mathbf{A}+\mathbf{A}^{\mathsf{H}}\|}{2}\operatorname{tr}(\mathbf{B}) \leq \|\mathbf{A}\|\operatorname{tr}(\mathbf{B}) \tag{7.45}$$

valid for any square matrix $\mathbf{A}$ and positive semi-definite matrix $\mathbf{B}$ ($\mathbf{B} \succeq 0$) of appropriate dimensions [24], together with the triangle inequality and the fact that the realizations of the graph Laplacian are upper bounded as $\|\mathbf{S}_t\| \leq \varrho$ to bound $T(k,\ell)$

$$\begin{aligned} T(k,\ell) &\leq \operatorname{tr}\left(\mathbb{E}\left[\mathbf{x}_{t-k+1}\mathbf{x}_{t-\ell+1}^{\mathsf{H}}\right]\right)\|\mathbb{E}\left[(\prod_{\tau=t}^{t-\ell+1}\mathbf{S}_\tau)(\prod_{\tau=t}^{t-k+1}\mathbf{S}_\tau)\right]\| \\ &\leq \operatorname{tr}\left(\boldsymbol{\Sigma}_x + \bar{\mathbf{x}}\bar{\mathbf{x}}^{\mathsf{H}}\right)\varrho^{k+\ell}, \end{aligned} \tag{7.46}$$

where in (7.46) we have also applied the Jensen's inequality for the spectral norm ($\|\mathbb{E}[\mathbf{A}]\| \leq \mathbb{E}[\|\mathbf{A}\|]$). We now can notice that the second term in the right-hand side of (7.40) is always positive. Thus it can be lower bounded as[5]

$$\operatorname{tr}(\mathbb{E}[\mathbf{y}_{t+1}]\mathbb{E}[\mathbf{y}_{t+1}^{\mathsf{H}}] \geq 0. \tag{7.47}$$

Then, combining the results (7.46), (7.47) and (7.41) we can upper bound (7.40) as

$$\operatorname{tr}(\boldsymbol{\Sigma}_y[t+1]) \leq \sum_{k=0,l=0}^{K} \phi_k\phi_\ell \operatorname{tr}\left(\boldsymbol{\Sigma}_x + \bar{\mathbf{x}}\bar{\mathbf{x}}^{\mathsf{H}}\right)\varrho^{k+\ell}, \tag{7.48}$$

which then can be reformulated as (7.9) dividing both sides by $N$ and with simple algebra.

## 7.F. PROOF OF THE ARMA$_K$ VARIANCE BOUND THEOREM

Let us set $\mathbf{C} = \mathbf{1}_K^{\mathsf{T}} \otimes \mathbf{I}_N$ for brevity. For the parallel ARMA$_K$ (7.29), we can express

$$\lim_{t\to\infty} \operatorname{tr}(\operatorname{var}(\mathbf{y}_{t+1})) = \lim_{t\to\infty} \operatorname{tr}(\operatorname{var}(\mathbf{C}\mathbf{y}_{t+1}^{(1:K)})) \tag{7.49}$$

Then, applying the definition and using the linearity of expectation, we have

$$\begin{aligned} \operatorname{var}(\mathbf{C}\mathbf{y}_{t+1}^{(1:K)}) &= \mathbf{C}\operatorname{var}(\mathbf{y}_{t+1}^{(1:K)})\mathbf{C}^{\mathsf{T}} \\ &= \mathbf{C}\mathbb{E}[\mathbf{y}_{t+1}^{(1:K)}\mathbf{y}_{t+1}^{(1:K),\mathsf{H}}]\mathbf{C}^{\mathsf{T}} - \mathbf{C}\mathbb{E}[\mathbf{y}_{t+1}^{(1:K)}]\mathbb{E}[\mathbf{y}_{t+1}^{(1:K)}]^{\mathsf{H}}\mathbf{C}^{\mathsf{T}}. \end{aligned} \tag{7.50}$$

[5] Notice that the goal is to present *a* bound. A tighter bound can be obtained using the results of Proposition 7.1 and writing $\operatorname{tr}(\bar{\mathbf{y}}_{t+1}\bar{\mathbf{y}}_{t+1}^{\mathsf{H}}) = \sum_{k=0,\ell=0}^{K} \phi_k\phi_\ell \bar{\mathbf{x}}^{\mathsf{H}}\bar{\mathbf{S}}^{k+\ell}\bar{\mathbf{x}}$.

To proceed, we use inequality (7.45) and exploit the fact that $\text{var}(\mathbf{y}_{t+1}^{(1:K)}) \succeq 0$ is positive semidefinite matrix being the covariance matrix of $\mathbf{y}_{t+1}$. We can therefore apply (7.45) to (7.50) as

$$\begin{aligned}\lim_{t\to\infty} \text{tr}(\text{var}(\mathbf{C}\mathbf{y}_{t+1}^{(1:K)})) &= \lim_{t\to\infty} \text{tr}(\mathbf{C}^{\mathsf{T}}\mathbf{C}\text{var}(\mathbf{y}_{t+1}^{(1:K)})) \\ &\leq \lim_{t\to\infty} \|\mathbf{C}^{\mathsf{T}}\mathbf{C}\|_2 \Big(\text{tr}(\mathbb{E}[\mathbf{y}_{t+1}^{(1:K)}\mathbf{y}_{t+1}^{(1:K),\mathsf{H}}]) - \text{tr}(\mathbb{E}[\mathbf{y}_{t+1}^{(1:K)}]\mathbb{E}[\mathbf{y}_{t+1}^{(1:K)}]^{\mathsf{H}})\Big) \\ &= \lim_{t\to\infty} K\text{tr}(\mathbb{E}[\mathbf{y}_{t+1}^{(1:K)}\mathbf{y}_{t+1}^{(1:K),\mathsf{H}}]) - K\text{tr}(\mathbb{E}[\mathbf{y}_{t+1}^{(1:K)}]\mathbb{E}[\mathbf{y}_{t+1}^{(1:K)}]^{\mathsf{H}})\end{aligned} \tag{7.51}$$

where, in the last step, we made the substitution $\|\mathbf{C}^{\mathsf{T}}\mathbf{C}\|_2 = \|\left(\mathbf{1}_K^{\mathsf{T}} \otimes \mathbf{I}_N\right)^{\mathsf{T}}\left(\mathbf{1}_K^{\mathsf{T}} \otimes \mathbf{I}_N\right)\|_2 = K$ and we used the linearity of the expectation and trace: $\mathbb{E}[\text{tr}(\mathbf{A})] = \text{tr}(\mathbb{E}[\mathbf{A}])$. To ease notation, define $\boldsymbol{\Phi}(t',t) := \prod_{\gamma=t}^{t'} \boldsymbol{\Psi} \otimes \mathbf{S}_\gamma$ for $t' \geq t$, and $\boldsymbol{\Phi}(t',t) := \mathbf{I}_{KN}$ if $t' < t$ and $\check{\mathbf{x}}_t = \boldsymbol{\varphi} \otimes \mathbf{x}_t$, after which the parallel $\text{ARMA}_K$ recursion (7.29a) becomes

$$\mathbf{y}_{t+1}^{(1:K)} = \boldsymbol{\Phi}(t,0)\mathbf{y}_0^{(1:K)} + \sum_{\tau=0}^{t} \boldsymbol{\Phi}(t, t-\tau+1)\check{\mathbf{x}}_{t-\tau}. \tag{7.52}$$

After some algebraic manipulation and using the properties of the trace, we can write

$$\begin{aligned}\mathbb{E}\left[\text{tr}(\mathbf{y}_{t+1}^{(1:K)}\mathbf{y}_{t+1}^{(1:K),\mathsf{H}}\right])) &= \mathbb{E}\left[\text{tr}(\boldsymbol{\Phi}(t,0)^{\mathsf{H}}\boldsymbol{\Phi}(t,0)\mathbf{y}_0^{(1:K)}\mathbf{y}_0^{(1:K),\mathsf{H}})\right] \\ &+ \sum_{\tau=0}^{t} \mathbb{E}\left[\text{tr}(\boldsymbol{\Phi}(t,t-\tau+1)^{\mathsf{H}}\boldsymbol{\Phi}(t,0)\mathbf{y}_0^{(1:K)}\check{\mathbf{x}}_{t-\tau}^{\mathsf{H}})\right] \\ &+ \sum_{\tau=0}^{t} \mathbb{E}\left[\text{tr}(\boldsymbol{\Phi}(t,0)^{\mathsf{H}}\boldsymbol{\Phi}(t,t-\tau+1)\check{\mathbf{x}}_{t-\tau}\mathbf{y}_0^{(1:K),\mathsf{H}})\right] \\ &+ \sum_{\tau_1,\tau_2=0}^{t} \mathbb{E}\Big[\text{tr}(\boldsymbol{\Phi}(t,t-\tau_2+1)^{\mathsf{H}}\boldsymbol{\Phi}(t,t-\tau_1+1)\check{\mathbf{x}}_{t-\tau_1}\check{\mathbf{x}}_{t-\tau_2}^{\mathsf{H}})\Big].\end{aligned} \tag{7.53}$$

Due to the independence of signal and graph, as well as since $\mathbb{E}[\mathbf{y}_0^{(1:K)}] = \mathbf{0}_{KN}$, the second and third terms above are equal to zero. Notice however that $\mathbf{y}_0^{(1:K)}\mathbf{y}_0^{(1:K),\mathsf{H}} \succeq 0$ and $\check{\mathbf{x}}_{t-\tau_1}\check{\mathbf{x}}_{t-\tau_2}^{\mathsf{H}} \succeq 0$: $\mathbf{y}_0^{(1:K)}\mathbf{y}_0^{(1:K),\mathsf{H}} \succeq 0$ is symmetric with one non-zero eigenvalue equal to $\|\mathbf{y}_0^{(1:K)}\|_2^2$, whereas $\mathbb{E}\left[\check{\mathbf{x}}_{t-\tau_1}\check{\mathbf{x}}_{t-\tau_2}^{\mathsf{H}}\right] \succeq 0$ is a Kronecker product of positive semi-definite matrices[6]

$$\begin{aligned}\mathbb{E}\left[\check{\mathbf{x}}_{t-\tau_1}\check{\mathbf{x}}_{t-\tau_2}^{\mathsf{H}}\right] &= \mathbb{E}\left[\left(\boldsymbol{\varphi} \otimes \mathbf{x}_{t-\tau_1}\right)\left(\boldsymbol{\varphi} \otimes \mathbf{x}_{t-\tau_2}\right)^{\mathsf{H}}\right] \\ &= \boldsymbol{\varphi}\boldsymbol{\varphi}^{\mathsf{H}} \otimes \mathbb{E}\left[\mathbf{x}_{t-\tau_1}\mathbf{x}_{t-\tau_2}^{\mathsf{H}}\right] \\ &= \begin{cases} \boldsymbol{\varphi}\boldsymbol{\varphi}^{\mathsf{H}} \otimes (\boldsymbol{\Sigma}_x + \bar{\mathbf{x}}\bar{\mathbf{x}}^{\mathsf{H}}), & \text{if } \tau_1 = \tau_2, \\ \boldsymbol{\varphi}\boldsymbol{\varphi}^{\mathsf{H}} \otimes \bar{\mathbf{x}}\bar{\mathbf{x}}^{\mathsf{H}}, & \text{otherwise.} \end{cases}\end{aligned} \tag{7.54}$$

[6] If $\mathbf{A} \succeq 0$ and $\mathbf{B} \succeq 0$, then $\mathbf{A} \otimes \mathbf{B} \succeq 0$.

We can therefore use again inequality (7.45), as well as Jensen's inequality for the spectral norm ($\|\mathbb{E}[\mathbf{A}]\| \leq \mathbb{E}[\|\mathbf{A}\|]$), and the linearity of the expectation and the trace, to write

$$\mathbb{E}\left[\mathrm{tr}(\mathbf{y}_{t+1}^{(1:K)}\mathbf{y}_{t+1}^{(1:K),\mathsf{H}})\right] \leq \mathbb{E}\left[\|\boldsymbol{\Phi}(t,0)^{\mathsf{H}}\|\|\boldsymbol{\Phi}(t,0)\|\right]\mathrm{tr}\left(\mathbb{E}\left[\mathbf{y}_0^{(1:K)}\mathbf{y}_0^{(1:K),\mathsf{H}}\right]\right) \tag{7.55}$$
$$+\sum_{\tau_1,\tau_2=0}^{t}\mathbb{E}\left[\|\boldsymbol{\Phi}(t,t-\tau_2+1)^{\mathsf{H}}\|\|\boldsymbol{\Phi}(t,t-\tau_1+1)\|\right]\times\mathrm{tr}\left(\mathbb{E}\left[\check{\mathbf{x}}_{t-\tau_1}\check{\mathbf{x}}_{t-\tau_2}^{\mathsf{H}}\right]\right).$$

Next, we examine $\mathbb{E}[\|\boldsymbol{\Phi}(t_1,t_2)\|]$. Considering that, for all $p$-norms and for any matrix $\mathbf{A}$ and $\mathbf{B}$ $\|\mathbf{A}\otimes\mathbf{B}\|_p = \|\mathbf{A}\|_p\|\mathbf{B}\|_p$, the expected spectral norm of $\boldsymbol{\Phi}(t_1,t_2)$ is bounded by

$$\begin{aligned}\mathbb{E}[\|\boldsymbol{\Phi}(t_1,t_2)\|] &\leq \mathbb{E}\left[\prod_{\gamma=t_2}^{t_1}\|\boldsymbol{\Psi}\otimes\mathbf{S}_\gamma\|\right] = \mathbb{E}\left[\prod_{\gamma=t_2}^{t_1}\|\boldsymbol{\Psi}\|\|\mathbf{S}_\gamma\|\right]\\ &= \prod_{\gamma=t_2}^{t_1}\|\boldsymbol{\Psi}\|\mathbb{E}\left[\|\mathbf{S}_\gamma\|\right] = \prod_{\gamma=t_2}^{t_1}|\psi_{\max}|\|\mathbb{E}\left[\mathbf{S}_\gamma\right]\|\\ &\leq (\varrho|\psi_{\max}|)^{t_1-t_2+1}.\end{aligned} \tag{7.56}$$

Therefore, in the limit, the first term of (7.55) vanishes

$$\begin{aligned}&\lim_{t\to\infty}\mathbb{E}\left[\|\boldsymbol{\Phi}(t,0)^{\mathsf{H}}\|\|\boldsymbol{\Phi}(t,0)\|\right]\mathrm{tr}\left(\mathbb{E}\left[\mathbf{y}_0^{(1:K)}\mathbf{y}_0^{(1:K),\mathsf{H}}\right]\right)\\ &\quad\leq \lim_{t\to\infty}(\varrho|\psi_{\max}|)^{2t+2}\mathrm{tr}\left(\mathbb{E}\left[\mathbf{y}_0^{(1:K)}\mathbf{y}_0^{(1:K),\mathsf{H}}\right]\right) = 0.\end{aligned} \tag{7.57}$$

The last step above is because the filter is stable $|\varrho\psi_{\max}| \leq 1$. Putting (7.55) and (7.56) together, while eliminating all terms that vanish, we obtain a bound for the first term of (7.51)

$$\begin{aligned}\lim_{t\to\infty}K\mathbb{E}\left[\mathrm{tr}(\mathbf{y}_{t+1}^{(1:K)}\mathbf{y}_{t+1}^{(1:K),\mathsf{H})}\right] &\leq \lim_{t\to\infty}K\sum_{\tau_1,\tau_2=0}^{t}(\varrho|\psi_{\max}|)^{\tau_1+\tau_2}\,\mathrm{tr}(\mathbb{E}\left[\check{\mathbf{x}}_{t-\tau_1}\check{\mathbf{x}}_{t-\tau_2}^{\mathsf{H}}\right])\\ &\leq \lim_{t\to\infty}K\sum_{\tau_1,\tau_2=0}^{t}(\varrho|\psi_{\max}|)^{\tau_1+\tau_2}\,\mathrm{tr}\left(\boldsymbol{\varphi}\boldsymbol{\varphi}^{\mathsf{H}}\otimes\left(\boldsymbol{\Sigma}_x+\bar{\mathbf{x}}\bar{\mathbf{x}}^{\mathsf{H}}\right)\right)\\ &= K\,\mathrm{tr}\left(\frac{\boldsymbol{\varphi}\boldsymbol{\varphi}^{\mathsf{H}}\otimes\left(\boldsymbol{\Sigma}_x+\bar{\mathbf{x}}\bar{\mathbf{x}}^{\mathsf{H}}\right)}{(1-\varrho|\psi_{\max}|)^2}\right),\end{aligned} \tag{7.58}$$

whereas for the remaining term from (7.50) we can see that it cannot be negative. Thus, we can lower bound[7] (notice that this term has to be subtracted) it as

$$\lim_{t\to\infty}K\mathrm{tr}(\mathbb{E}[\mathbf{y}_{t+1}^{(1:K)}]\mathbb{E}[\mathbf{y}_{t+1}^{(1:K)}]^{\mathsf{H}}) \geq 0. \tag{7.59}$$

Putting together (7.49), (7.58)-(7.59) and after some algebraic manipulation, we reach the bound

$$\lim_{t\to\infty}\overline{\mathrm{var}}[\mathbf{y}_{t+1}] \leq (K/N)\,\mathrm{tr}\left(\frac{\boldsymbol{\varphi}\boldsymbol{\varphi}^{\mathsf{H}}\otimes\left(\boldsymbol{\Sigma}_x+\bar{\mathbf{x}}\bar{\mathbf{x}}^{\mathsf{H}}\right)}{(1-\varrho|\psi_{\max}|)^2}\right). \tag{7.60}$$

[7] As for the FIR upper bound, also here one can exploit the structure of $\lim_{t\to\infty}K\mathrm{tr}(\mathbb{E}[\mathbf{y}_{t+1}^{(1:K)}]\mathbb{E}[\mathbf{y}_{t+1}^{(1:K)}]^{\mathsf{H}})$ to achieve a tighter bound.

We now use the property of the trace and Kronecker product, $\mathrm{tr}(\mathbf{A}\otimes\mathbf{B}) = \mathrm{tr}(\mathbf{A})\mathrm{tr}(\mathbf{B})$ with $\mathrm{tr}(\boldsymbol{\varphi}\boldsymbol{\varphi}^{\mathsf{H}}) = \|\boldsymbol{\varphi}\|^2$, since $\boldsymbol{\varphi}$ is a diagonal matrix. With the latter consideration (7.60) can be then reformulated as (7.10) with simple algebra.

## 7.G. PROOF OF THE RECURSIVE ARMA$_K$ VARIANCE COMPUTATION

The covariance matrix $\boldsymbol{\Sigma}_y[t+1]$ of the filter output $\mathbf{y}_{t+1}$ in (7.12) can be expressed as

$$\begin{aligned}\boldsymbol{\Sigma}_y[t+1] &= \mathbf{C}\boldsymbol{\Sigma}_{y^{(1:K)}}[t+1]\mathbf{C}^{\mathsf{T}} \\ &= \mathbf{C}\mathbb{E}\left[\mathbf{y}_{t+1}^{(1:K)}\mathbf{y}_{t+1}^{(1:K),\mathsf{T}}\right]\mathbf{C}^{\mathsf{T}} - \mathbf{C}\mathbf{y}_{t+1}^{(1:K)}\mathbf{y}_{t+1}^{(1:K),\mathsf{T}}\mathbf{C}^{\mathsf{T}},\end{aligned} \tag{7.61}$$

where in the above derivation we made use of the fact that $\mathbf{y}_t^{(1:K)}$ is independent from $\mathbf{x}_t$. Then, we can notice that only the autocorrelation matrix of the system memory $\mathbf{R}_{y^{(1:K)}}[t+1] = \mathbb{E}\left[\mathbf{y}_{t+1}^{(1:K)}\mathbf{y}_{t+1}^{(1:K),\mathsf{T}}\right]$ at time instant $t+1$ is unknown in (7.61). The term $\mathbf{C}\mathbf{y}_{t+1}^{(1:K)}\mathbf{y}_{t+1}^{(1:K),\mathsf{T}}\mathbf{C}^{\mathsf{T}}$ can be recursively calculated using the statistical knowledge of time instant $t$. Substituting the expression of $\mathbf{y}_{t+1}^{(1:K)}$ in (7.13a) into $\mathbf{R}_{y^{(1:K)}}[t+1]$ we have

$$\begin{aligned}\mathbf{R}_{y^{(1:K)}}[t+1] = \bar{\mathbf{A}}\mathbf{R}_{y^{(1:K)}}[t]\bar{\mathbf{A}}^{\mathsf{T}} + \bar{\mathbf{A}}\bar{\mathbf{y}}_t^{(1:K)}(\boldsymbol{\varphi}^{\mathsf{T}}\otimes\bar{\mathbf{x}}_t^{\mathsf{T}}) + (\boldsymbol{\varphi}\otimes\bar{\mathbf{x}}_t)\bar{\mathbf{y}}_t^{(1:K),\mathsf{T}}\bar{\mathbf{A}}^{\mathsf{T}} \\ + \mathbb{E}_{\tilde{\mathbf{A}}}\left[\tilde{\mathbf{A}}_t\mathbf{R}_{y^{(1:K)}}[t]\tilde{\mathbf{A}}_t^{\mathsf{T}}\right] + \boldsymbol{\varphi}\boldsymbol{\varphi}^{\mathsf{T}}\otimes\left(\boldsymbol{\Sigma}_x[t] + \bar{\mathbf{x}}_t\bar{\mathbf{x}}_t^{\mathsf{T}}\right)\end{aligned} \tag{7.62}$$

where once again we used the Kronecker properties $(\mathbf{A}\otimes\mathbf{B})^{\mathsf{T}} = \mathbf{A}^{\mathsf{T}}\otimes\mathbf{B}^{\mathsf{T}}$ and (7.33). In (7.62), $\mathbb{E}\left[\tilde{\mathbf{A}}_t\mathbf{y}_t^{(1:K)}\mathbf{y}_t^{(1:K),\mathsf{T}}\tilde{\mathbf{A}}_t^{\mathsf{T}}\right] = \mathbb{E}_{\tilde{\mathbf{A}}}\left[\tilde{\mathbf{A}}_t\mathbb{E}_{y^{(1:K)}}\left[\mathbf{y}_t^{(1:K)}\mathbf{y}_t^{(1:K),\mathsf{T}}\right]\tilde{\mathbf{A}}_t^{\mathsf{T}}\right]$ due to the independency between the graph realization at time instant $t+1$ and system memory at time instant $t$ and linearity of the expecation. Then, we can see that each entry of the matrix $\mathbb{E}_{\tilde{\mathbf{A}}}[\tilde{\mathbf{A}}_t\mathbf{R}_y[t]\tilde{\mathbf{A}}_t^{\mathsf{T}}]$ is derived from the standard three matrix multiplication which can be expressed as (7.15). Let us now focus on the term $\mathbf{C}\mathbf{y}_{t+1}^{(1:K)}\mathbf{y}_{t+1}^{(1:K),\mathsf{T}}\mathbf{C}^{\mathsf{T}}$. Using (7.13a) we can expand it as

$$\begin{aligned}\mathbf{C}\bar{\mathbf{y}}_{t+1}^{(1:K)}\bar{\mathbf{y}}_{t+1}^{(1:K),\mathsf{T}}\mathbf{C}^{\mathsf{T}} = \mathbf{C}\bar{\mathbf{A}}\bar{\mathbf{y}}_t^{(1:K)}\bar{\mathbf{y}}_t^{(1:K),\mathsf{T}}\bar{\mathbf{A}}^{\mathsf{T}}\mathbf{C}^{\mathsf{T}} + \mathbf{C}\bar{\mathbf{A}}\bar{\mathbf{y}}_t^{(1:K)}(\boldsymbol{\varphi}^{\mathsf{T}}\otimes\bar{\mathbf{x}}_t^{\mathsf{T}})\mathbf{C}^{\mathsf{T}} \\ + \mathbf{C}(\boldsymbol{\varphi}\otimes\bar{\mathbf{x}}_t)\bar{\mathbf{y}}_t^{(1:K),\mathsf{T}}\bar{\mathbf{A}}^{\mathsf{T}}\mathbf{C}^{\mathsf{T}} + \mathbf{C}(\boldsymbol{\varphi}\boldsymbol{\varphi}^{\mathsf{T}}\otimes\bar{\mathbf{x}}_t\bar{\mathbf{x}}_t^{\mathsf{T}})\mathbf{C}^{\mathsf{T}}.\end{aligned} \tag{7.63}$$

Then, by substituting (7.62), (7.63) and $\mathbb{E}_{\tilde{\mathbf{A}}}[\tilde{\mathbf{A}}_t\mathbf{R}_{\mathbf{y}}[t]\tilde{\mathbf{A}}_t^{\mathsf{T}}]$ into (7.61) the claim (7.14) follows.

## FURTHER READING

[1] E. Isufi, A. Loukas, A. Simonetto, and G. Leus, *Filtering random graph processes over random time-varying graphs,* IEEE Transactions on Signal Processing (2017).

[2] E. Isufi, A. Simonetto, A. Loukas, and G. Leus, *Stochastic graph filtering on time-varying graphs,* in *Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), 2015 IEEE 6th International Workshop on* (IEEE, 2015) pp. 89–92.

[3] E. Isufi and G. Leus, *Distributed sparsified graph filters for denoising and diffusion tasks,* in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on* (IEEE, 2017) pp. 5865–5869.

[4] I. D. Schizas, G. B. Giannakis, S. I. Roumeliotis, and A. Ribeiro, *Consensus in ad hoc wsns with noisy linksópart ii: Distributed estimation and smoothing of random signals,* IEEE Transactions on Signal Processing **56**, 1650 (2008).

[5] H. Zhu, G. B. Giannakis, and A. Cano, *Distributed in-network channel decoding,* IEEE Transactions on Signal Processing **57**, 3970 (2009).

[6] J. Tsitsiklis, D. Bertsekas, and M. Athans, *Distributed asynchronous deterministic and stochastic gradient optimization algorithms,* IEEE transactions on automatic control **31**, 803 (1986).

[7] K. Srivastava and A. Nedic, *Distributed asynchronous constrained stochastic optimization,* IEEE Journal of Selected Topics in Signal Processing **5**, 772 (2011).

[8] D. A. Spielman and S.-H. Teng, *Spectral sparsification of graphs,* SIAM Journal on Computing **40**, 981 (2011).

[9] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, *Randomized gossip algorithms,* IEEE/ACM Transactions on Networking (TON) **14**, 2508 (2006).

[10] F. Iutzeler, P. Bianchi, P. Ciblat, and W. Hachem, *Asynchronous distributed optimization using a randomized alternating direction method of multipliers,* in *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on* (IEEE, 2013) pp. 3671–3676.

[11] E. Wei and A. Ozdaglar, *On the o (1= k) convergence of asynchronous distributed alternating direction method of multipliers,* in *Global conference on signal and information processing (GlobalSIP), 2013 IEEE* (IEEE, 2013) pp. 551–554.

[12] G. Chen, G. Davis, F. Hall, Z. Li, K. Patel, and M. Stewart, *An interlacing result on normalized laplacians,* SIAM Journal on Discrete Mathematics **18**, 353 (2004).

[13] C. Zhang, D. Florêncio, and P. A. Chou, *Graph signal processing–a probabilistic framework,* Microsoft Res., Redmond, WA, USA, Tech. Rep. MSR-TR-2015-31 (2015).

[14] T. Kailath, *Linear systems*, Vol. 156 (Prentice-Hall Englewood Cliffs, NJ, 1980).

[15] S. Segarra, A. G. Marques, G. Leus, and A. Ribeiro, *Reconstruction of graph signals through percolation from seeding nodes,* IEEE Transactions on Signal Processing **64**, 4363 (2016).

[16] S. Barbarossa, S. Sardellitti, and A. Farina, *On sparse controllability of graph signals,* in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on* (IEEE, 2016) pp. 4104–4108.

[17] P. Di Lorenzo, S. Barbarossa, P. Banelli, and S. Sardellitti, *Adaptive least mean squares estimation of graph signals,* IEEE Transactions on Signal and Information Processing over Networks **2**, 555 (2016).

[18] D. Romero, M. Ma, and G. B. Giannakis, *Kernel-based reconstruction of graph signals,* IEEE Transactions on Signal Processing **65**, 764 (2017).

[19] J. A. Fax and R. M. Murray, *Information flow and cooperative control of vehicle formations,* IEEE transactions on automatic control **49**, 1465 (2004).

[20] P. Massioni and M. Verhaegen, *Distributed control for identical dynamically coupled systems: A decomposition approach,* IEEE Transactions on Automatic Control **54**, 124 (2009).

[21] D. P. Bertsekas, D. P. Bertsekas, D. P. Bertsekas, and D. P. Bertsekas, *Dynamic programming and optimal control*, Vol. 1 (Athena scientific Belmont, MA, 1995).

[22] W. H. Fleming and R. W. Rishel, *Deterministic and stochastic optimal control*, Vol. 1 (Springer Science & Business Media, 2012).

[23] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming* (John Wiley & Sons, 2014).

[24] S.-D. Wang, T.-S. Kuo, and C.-F. Hsu, *Trace bounds on the solution of the algebraic matrix riccati and lyapunov equation,* IEEE Transactions on Automatic Control **31**, 654 (1986).

# IV

# GRAPH-TIME SAMPLING

# 8

# OBSERVING AND TRACKING GRAPH PROCESSES

*Accuracy of observation is the equivalent of accuracy of thinking.*

Wallace Stevens

In the previous three chapters we saw how the extension of GSP to the temporal dimension could bring substantial benefits to filtering. In this chapter, we show that time can also be exploited to observe and track a time-varying graph signal from a few nodes. To these graph signals, we will refer in short as *graph processes*. Since we are interested to observe a track a graph process by not collecting all the measurements, we would like to derive conditions when this is possible. To the best of our knowledge, this is the first contribution in this direction, and the following sections are based on our works [1, 2].

This chapter begins with an introduction to the tasks of observability and tracking for graph processes in Section 8.1. The state-space models on graphs are described in Section 8.2. The observability of graph processes is treated in Section 8.3, while the derivations for the graph process tracking are performed in Section 8.4. The chapter is concluded in Section 8.5 with a summary.

## 8.1. INTRODUCTION

An important branch in GSP is the *sampling theory* for signals on graphs. This theory mainly relies on the observation that graph signals are often sparse (i.e., bandlimited) in the graph frequency domain. Practical examples of bandlimited graph signals include temperature measurements, where adjacent sensors measure similar values, fMRI data of brain networks [3], ratings in recommendation systems [4], protein networks [5], and

Part of this chapter is submitted in the IEEE Transactions on Signal Processing [1] (2017) and is presented in [2] (2018).

networks that exhibit a clustering behavior such as opinion networks. A substantial contribution to sampling of bandlimited graph signals is provided by [6], which is then extended in a number works including [7–13] to propose signal reconstruction strategies from a few measurements.

The aforementioned works propose sampling and reconstruction strategies only for a single snapshot of the graph signal ignoring its time-varying nature. However, time-varying graph signals, named here graph processes, are encountered in consecutive sensor measurements, biological signal evolution prone to stimuli, financial networks, and information diffusion over networks. The latter has gave rise to a couple of recent works that propose sampling and reconstruction strategies for graph processes [12, 14–18]. In this chapter, we continue on this path to propose a graph-time sampling theory for the tasks of observing and tracking a bandlimited graph process, i.e., a graph process that has a sparse graph frequency content over time.

Observability of network processes has been considered in [19, 20] for sensor placement and in [21] for designing observable topologies. While these findings are of particular importance, these works do not consider sampling strategies for observing the network process. Differently, we exploit GSP tools and in particular the bandlimited prior to bring the graph sampling theory into the temporal dimension. This allows us to derive by theoretical guarantees when a bandlimited network process is observable from a few measurements and to propose effective graph-time sampling schemes.

The bandlimited assumption has also been exploited in [14, 17] for tracking slowly time-varying graph signals from a few nodes. However, since the main goal in these works is to develop sampling strategies for adaptive signal reconstruction, signal tracking comes as a byproduct and, therefore, without theoretical guarantees. Here, we tackle this challenge and we propose sampling strategies directly to track the process by means of Kalman filtering (KF), which is known to achieve the optimal performance.

The works in [22–24] exploited the graph structure to improve the prediction accuracy of AR and ARMA models. However, the findings in these works need all the data to stand. As we show later, this is not the case for the proposed KF. The work in [12] casts the tracking of a graph process from limited measurements as a regularized-based interpolation problem on a larger graph. Subsequently, KF-like methods are used to alleviate the computational burden. We identify three main differences with the proposed framework. First, no conditions on the minimum number of required samples are provided in [12]. Second, the sampling is performed uniformly at random, which is well-known to be a suboptimal choice. Finally, since our framework is not a regularized interpolation problem, we avoid the task of designing the regularizer and its weight to track the process.

Differently from the above, KF has also been used to track network dynamics [25, 26]. However, in these cases, the graph embeds the communication links between the sensors. To the best of our knowledge, this is the first attempt that conciliates process tracking with a designed sampling strategy and with the nature of the graph process itself, i.e., its bandlimitedness.

### 8.1.1. CONTRIBUTIONS

We can divide the chapter contributions into two parts.

**Contribution 8.1.** *Observability of graph processes.* We extend the sampling theory for graph signals to the observability of graph processes.

- We derive *necessary* and *sufficient* conditions for observing a bandlimited graph process from a subset of nodes.
- We propose two approaches for observability: *i*) observability with deterministic sampling (cf. Section 8.3.1), i.e., when the selected subset of nodes is chosen deterministically; and *ii*) observability with random sampling (cf. Section 8.3.2), i.e., when a subset of nodes is sampled with a given probability. We perform a mean-square error (MSE) analysis of the state estimation performance to show the connection between the graph topology, the process bandwidth, and the sampling set.
- We propose sampling techniques based on sparse sensing to pick the minimum number of samples such that a target MSE estimation performance is guaranteed.

**Contribution 8.2.** *Kalman filtering for graph processes.* We propose KF to track bandlimited graph processes that follow a predefined model. We first consider KF for time-varying models (cf. Section 8.4.1) and then extend our derivations to steady-state KF (cf. Section 8.4.2), i.e., when the model is time-invariant.

- We derive necessary conditions on the minimum required number of nodes for tracking the graph process and conciliate these conditions with those derived for observability. The MSE analysis, given by the posterior (or steady-state) error covariance matrix highlights the role played by the graph topology, the graph process bandwidth, and the sampling set in the tracking (steady-state) performance.
- We propose a sparse sensing strategy to sample the graph, which ensures a predefined MSE tracking cost.

### 8.1.2. APPLICATIONS



Within the context of observability of a graph process we consider the following applications.

- *Gossip propagation in social networks.* As nowadays social networks are one of the fastest platforms to spread information in large scale, the identification of information sources has an important role in classifying these sources as relevant, trustable, or threaten. Since most of the users do not render their data public (e.g., privacy concerns), we consider observing the gossip source by collecting the data only from the available users. Then, the proposed framework can be exploited to carefully pick the information from the right users and to localize the source.
- *Chemical agent diffusion in air.* After the Chernobyl nuclear disaster in 1986[1], international concern about the diffusion of chemical agents, such as radioactive materials, increased and several efforts were put to understand the dynamics of

[1] According to studies summarized in the Wikipedia article [27], radioactive material was revealed in a large area spanning from Russia to Italy.

such diffusion. One example is the ETEX experiment [28], where two tracers are released in the atmosphere and sampled over a period of three days over different stations in Europe. In this experiment, we may consider the different stations on the vertices of a graph and exploit the GSP theory to observe the place from where the chemical agent was released. This information can be then used to isolate the chemical agent spread and to take further controlling actions. A main challenge, faced also in the ETEX experiment, was that not all stations could detect the released tracer, thus the proposed graph-time sampling theory can be explored to reconstruct the original signal from the few available measurements.

As observability is applicable to the above applications in identifying the graph signal diffusion source, we believe that KF on graphs applies as well to the above cases for the tasks of tracking the graph signal evolution. Let us briefly illustrate this with the gossip application.

- *Gossip tracking in social networks.* We consider that information about a particular topic is being shared over a social network, and that a set of selected users inject new related information to orient the network opinion at a particular direction. Through the use of KF on graphs, we aim collecting subsampled measurements such that we can optimally track the gossip trend, and as such to design more efficient controlling actions. In this way, the graph signal subsampling must be performed per each time instant to collect measurements recursively from the most relevant users.

## 8.2. STATE-SPACE MODELS ON GRAPHS

In this section we introduce the state-space models on graphs that will be used throughout the chapter. We start our discussion by reformulating in mathematical terms the system on graphs. Then, we define the bandlimited graph process and reformulate the system on graphs as a sparse representation in the GFT domain.



### 8.2.1. SYSTEMS ON GRAPHS

Consider the $N-$state discrete linear time-varying system

$$\mathbf{x}_t = \mathbf{A}_{t-1}\mathbf{x}_{t-1} + \mathbf{B}_{t-1}\mathbf{u}_{t-1} \tag{8.1a}$$

$$\mathbf{y}_t = \mathbf{C}_{\mathscr{S}_t}(\mathbf{x}_t + \mathbf{v}_t), \tag{8.1b}$$

where $\mathbf{x}_t$ is the state vector containing the graph signal at time $t$, $\mathbf{u}_t$ is the input signal, and $\mathbf{A}_t$ and $\mathbf{B}_t$ are the time-varying state-transition and input matrices, respectively. $\mathbf{y}_t \in \mathbb{C}^N$ is the measurement vector and $\mathbf{C}_{\mathscr{S}_t} = \text{diag}(c_{t,1}, \ldots, c_{t,N})$ is the sampling matrix with $c_{t,n} = 1$ if the $n$th node belongs to the instantaneous sampling set $\mathscr{S}_t$ defined as $\mathscr{S}_t = \{n \in \{1, \ldots, N\} | c_{t,n} = 1\}$. $\mathbf{v}_t$ is white zero-mean noise with covariance matrix $\mathbf{\Sigma}_v = \sigma_v^2 \mathbf{I}_N$.

Model (8.1) comprises the following network processes.

**Signal diffusion.** For $\mathbf{x}_0$ being the initial signal state on the graph, its instantaneous diffused [29] realization is expressed through the exponential matrix product

$$\mathbf{x}_t = e^{-w\mathbf{L}_\text{d}t}\mathbf{x}_0 = e^{-w\mathbf{L}_\text{d}}e^{-w\mathbf{L}_\text{d}(t-1)}\mathbf{x}_0 \triangleq \mathbf{A}\mathbf{x}_{t-1} \tag{8.2}$$

where $w > 0$ is the diffusion rate and $\mathbf{A} = e^{-w\mathbf{L}_{\mathrm{d}}}$ is the time-invariant state-transition matrix. In (8.2) we can also incorporate an input $\mathbf{u}_{t-1}$ which may represent additional sources that become available at $t-1>0$. The diffusion model has found several practical applications including temperature diffusion, chemical substances dispersion, opinion propagation over networks [30], and brain signal analysis [31].

**Wave propagation.** The discretised wave equation on graphs [32] follows the two-step recursion

$$\mathbf{w}_t = (2\mathbf{I}_N - c^2\mathbf{L}_{\mathrm{d}})\mathbf{w}_{t-1} - \mathbf{w}_{t-2}, \tag{8.3}$$

with initial state $\mathbf{w}_0$ and wave speed $c$. Recursion (8.3) can be reformulated as (8.1a) by defining $\mathbf{x}_t = [\mathbf{w}_{t-1}, \mathbf{w}_t]\mathsf{T}$ with $\mathbf{A} = \left[[\mathbf{I}_N, (2\mathbf{I}_N - c^2\mathbf{L}_{\mathrm{d}})]\mathsf{T}, [\mathbf{0}_N\mathbf{0}_N\mathsf{T}, \mathbf{I}_N]\mathsf{T}\right]$ and $\mathbf{w}_{-1} = \mathbf{0}_N$. The latter is of practical interest for instance in seismic data [33].

**ARMA graph processes.** We denote an ARMA graph process as

$$\mathbf{x}_t = f(\mathbf{S})\mathbf{x}_{t-1} + g(\mathbf{S})\mathbf{u}_{t-1}, \tag{8.4}$$

where $f(\mathbf{S})$ and $g(\mathbf{S})$ are matrix functions of $\mathbf{S}$ that share the eigenvectors with $\mathbf{S}$, such as polynomials of a given power.

A particular form of (8.4) is the first-order recursion

$$\mathbf{x}_t = -w\mathbf{S}\mathbf{x}_{t-1} + \mathbf{u}_0 \text{ with } \quad \mathbf{x}_0 = \mathbf{0}_N, \tag{8.5}$$

which for $0 < w < 1/\lambda_{\max}(\mathbf{S})$ reaches the steady-state

$$\mathbf{x} = \lim_{t\to\infty} \mathbf{x}_t = (\mathbf{I}_N + w\mathbf{S})^{-1}\mathbf{u}_0. \tag{8.6}$$

Expression (8.6) is the solution of the so-called aggregate diffusion model which plays an important role in image smoothing [34], Tikhonov denoising [35], and recommendation systems [36].

### 8.2.2. BANDLIMITED SYSTEMS ON GRAPHS



To proceed with the graph Fourier analysis of (8.1), we define the following.

**Definition 8.1.** *A graph process $\mathbf{x}_t$ with instantaneous GFT $\hat{\mathbf{x}}_t = \mathbf{U}^{\mathsf{H}}\mathbf{x}_t$ is $\mathscr{F}$−bandlimited if $\hat{\mathbf{x}}_t$ has non-zero frequency content only on a subset of graph frequency indices $\mathscr{F}$.*

The set $\mathscr{F} = \{n \in \{1,\dots,N\} | \hat{x}_{t,n} \neq 0, t \geq 0\}$ is considered to be a common *time-invariant* set for all realizations of $\mathbf{x}_t$. Said differently, $\mathscr{F}$ is the union of all instantaneous sets $\mathscr{F}_t = \{n \in \{1,\dots,N\} | \hat{x}_{t,n} \neq 0\}$. We then write

$$\mathbf{x}_t = \mathbf{U}_{\mathscr{F}}\tilde{\mathbf{x}}_t, \tag{8.7}$$

where $\tilde{\mathbf{x}}_t \in \mathbb{C}^{|\mathscr{F}|}$ is the vector containing the entries of $\hat{\mathbf{x}}_t$ indicated by $\mathscr{F}$.

We further assume the following.

**Assumption 8.1.** *The system evolution matrices $\mathbf{A}_t$ and $\mathbf{B}_t$ share the eigenvectors with the graph shift operator $\mathbf{S}$.*

**Assumption 8.2.** *The input $\mathbf{u}_t$ is an $\mathscr{F}$−bandlimited graph process.*

Assumption 8.1 focuses our attention to linear time-varying systems on graphs that are a function of the graph shift operator. In fact, for all network processes in Section 8.2.1 this assumption holds. Assupmtion 8.2 requires the input signal to have a sparse GFT over time. That is, $\mathbf{u}_t$ should be a (piece-wise) smooth input signal on the graph, or have properties similar to the signals studied in [3–5, 7–11, 14–17, 22, 23, 37, 38]. From Definition 8.1, we should note that the bandlimitedness of $\mathbf{x}_t$ considers the sparsity in the GFT domain of all past realizations including those of $\mathbf{u}_0, \ldots, \mathbf{u}_{t-1}$. Furthermore, we will not consider Assumption 8.2 for the task of observability and will leverage it only for tracking.

The subsequent proposition formalizes the above.

**Proposition 8.1.** *Let $\mathbf{x}_t$ be a graph process that follows model* (8.1) *and let Assumptions 8.1 and 8.2 hold. Then, $\mathbf{x}_t$ is an $\mathscr{F}-$bandlimited graph process if and only if $\mathbf{x}_0$ is an $\mathscr{F}-$bandlimited graph signal.*

(The proof follows from simple algebra.)

With this in place, we can write the evolution of $\mathbf{x}_t$ as

$$\tilde{\mathbf{x}}_t = \tilde{\mathbf{A}}_{t-1}\tilde{\mathbf{x}}_{t-1} + \tilde{\mathbf{B}}_{t-1}\tilde{\mathbf{u}}_{t-1} \tag{8.8a}$$

$$\mathbf{y}_{\mathscr{S}_t} = \mathbf{C}_{\mathscr{S}_t}(\mathbf{U}_{\mathscr{F}}\tilde{\mathbf{x}}_t + \mathbf{v}_t), \tag{8.8b}$$

where $\tilde{\mathbf{A}}_t = \mathbf{U}_{\mathscr{F}}^{\mathsf{H}}\mathbf{A}_t\mathbf{U}_{\mathscr{F}}$ and $\tilde{\mathbf{B}}_t = \mathbf{U}_{\mathscr{F}}^{\mathsf{H}}\mathbf{B}_t\mathbf{U}_{\mathscr{F}}$ are diagonal matrices containing the in-band spectrum of $\mathbf{A}_t$ and $\mathbf{B}_t$, respectively.

Hereinafter, we will refer to systems of the form (8.1) that can be written in the form (8.8) as $\mathscr{F}-$bandlimited systems on graphs. The $\mathscr{F}-$bandlimited graph processes considered in this paper follow the evolutions (8.1a) and (8.8a) in the vertex and graph spectral domain, respectively. In the next section, we provide conditions for the observation of an $\mathscr{F}-$bandlimited graph process.

8

## 8.3. OBSERVING GRAPH PROCESSES

We start this section by adapting the definition of observability to our context [39].

**Definition 8.2.** *An $\mathscr{F}-$bandlimited system on graph is observable over the set $\mathscr{S}_{0:T} = \bigcup_{t=0}^{T}\mathscr{S}_t = \{n \in \{1,\ldots,N\}; t \in \{0,\ldots,T\} |\ c_{t,n} = 1\}$ if for any $\mathscr{F}-$bandlimited initial state $\mathbf{x}_0$ and some final time $T$, the initial state $\mathbf{x}_0$ can be uniquely determined in the absence of noise by the knowledge of the input $\mathbf{u}_t$ and measurement $\mathbf{y}_t$ for all $t \in \{0,\ldots,T\}$.*

The set $\mathscr{S}_{0:T}$ specifies all graph-time locations where and when the nodes are sampled in the interval $\{0,\ldots,T\}$. Since for observability we need the knowledge of the input signal, Assumption 8.2 is not necessary here.

With the above formulation in place, we can answer the questions: *Under which conditions is an $\mathscr{F}-$bandlimited graph process observable from a few measurements? When and where should we collect noisy measurements to estimate $\mathbf{x}_0$ up to a desired accuracy?*

To provide an answer, we write the relation between the measurement $\mathbf{y}_t$ and the initial $\mathscr{F}-$bandlimited signal $\tilde{\mathbf{x}}_0$ as

$$\mathbf{y}_t = \mathbf{C}_{\mathscr{S}_t}\mathbf{U}_{\mathscr{F}}\tilde{\mathbf{A}}_{t,0}\tilde{\mathbf{x}}_0 + \mathbf{C}_{\mathscr{S}_t}\mathbf{U}_{\mathscr{F}}\sum_{\tau=0}^{t-1}\tilde{\mathbf{A}}_{t,\tau+1}\tilde{\mathbf{B}}_{\tau}\tilde{\mathbf{u}}_{\tau} + \mathbf{C}_{\mathscr{S}_t}\mathbf{v}_t, \tag{8.9}$$

with

$$\tilde{\mathbf{A}}_{t,\tau} = \begin{cases} \tilde{\mathbf{A}}_{t-1}\tilde{\mathbf{A}}_{t-2}\ldots\tilde{\mathbf{A}}_{\tau}, & t > \tau \\ \mathbf{I}_{|\mathscr{F}|}, & t = \tau \\ \mathbf{0}_{|\mathscr{F}|}\mathbf{0}_{|\mathscr{F}|}^{\mathsf{T}}, & t < \tau. \end{cases} \tag{8.10}$$

Let $\mathbf{y}_{0:T} = [\mathbf{y}_0^{\mathsf{T}}, \mathbf{y}_1^{\mathsf{T}}, \ldots, \mathbf{y}_T^{\mathsf{T}}]^{\mathsf{T}}$ be the vector of measurements collected in the interval $\{0,\ldots,T\}$. Then, from (8.9) we have

$$\mathbf{y}_{0:T} = \mathbf{O}_{0:T}\tilde{\mathbf{x}}_0 + \mathbf{J}_{0:T}\mathbf{u}_{0:T-1} + \mathbf{C}_{\mathscr{S}_{0:T}}\mathbf{v}_{0:t}, \tag{8.11}$$

where

$$\begin{aligned} \mathbf{O}_{0:T} &= [(\mathbf{C}_{\mathscr{S}_0}\mathbf{U}_{\mathscr{F}}\tilde{\mathbf{A}}_{0,0})^{\mathsf{T}}, (\mathbf{C}_{\mathscr{S}_1}\mathbf{U}_{\mathscr{F}}\tilde{\mathbf{A}}_{1,0})^{\mathsf{T}}, \ldots, (\mathbf{C}_{\mathscr{S}_T}\mathbf{U}_{\mathscr{F}}\tilde{\mathbf{A}}_{T,0})^{\mathsf{T}}]^{\mathsf{T}} \\ &= \mathbf{C}_{\mathscr{S}_{0:T}}(\mathbf{I}_{T+1}\otimes\mathbf{U}_{\mathscr{F}})\tilde{\mathbf{A}}_{0:T}, \end{aligned} \tag{8.12}$$

$\mathbf{C}_{\mathscr{S}_{0:T}} = \text{blkdiag}(\mathbf{C}_{\mathscr{S}_0},\ldots,\mathbf{C}_{\mathscr{S}_T})$ $\tilde{\mathbf{A}}_{0:T} = \left[\mathbf{I}_{|\mathscr{F}|}, \tilde{\mathbf{A}}_{1,0}^{\mathsf{T}}, \ldots, \tilde{\mathbf{A}}_{T,0}^{\mathsf{T}}\right]^{\mathsf{T}}$, $\mathbf{u}_{0:T-1} = [\mathbf{u}_0^{\mathsf{T}}, \mathbf{u}_1^{\mathsf{T}}, \ldots, \mathbf{u}_{T-1}^{\mathsf{T}}]^{\mathsf{T}}$, and $\mathbf{v}_{0:T} = [\mathbf{v}_0^{\mathsf{T}}, \mathbf{v}_1^{\mathsf{T}}, \ldots, \mathbf{v}_T^{\mathsf{T}}]^{\mathsf{T}}$. $\mathbf{J}_{0:T}$ is the input evolution matrix in the interval $\{0,\ldots,T\}$ whose expression is not required for our derivations, but can be obtained from (8.9).

In the next section, we answer the above questions for $\mathbf{C}_t$ being a deterministic sampler, while in Section 8.3.2 we consider the case where the entries of $\mathbf{C}_t$ follow a Bernoulli distribution.

### 8.3.1. OBSERVABILITY WITH DETERMINISTIC SAMPLING

In this section, we consider the task of observability when the sampled nodes are chosen deterministically. Recall in this context that $\mathbf{C}_{\mathscr{S}_{0:T}}$ plays the role of the set projection matrix over the set $\mathscr{S}_{0:T}$. Given then $\mathbf{C}_{\mathscr{S}_{0:T}}$, system (8.8) is observable over $\mathscr{S}_{0:T}$ iff the observability matrix $\mathbf{O}_{0:T}$ in (8.12) is full rank [39], i.e., $\text{rank}(\mathbf{O}_{0:T}) = |\mathscr{F}|$. Then, we have

$$\tilde{\mathbf{x}}_0^o = \mathbf{O}_{0:T}^{\dagger}\left(\mathbf{y}_{0:T} - \mathbf{J}_{0:T}\mathbf{u}_{0:T-1}\right), \tag{8.13}$$

which is also the least squares (LS) estimate of $\tilde{\mathbf{x}}_0$ in the presence of noise $\mathbf{v}_t \neq 0$. From the structure of $\mathbf{O}_{0:T}$, a *sufficient* condition for observability is that at least one of the block matrices $\mathbf{C}_{\mathscr{S}_t}\mathbf{U}_{\mathscr{F}}\tilde{\mathbf{F}}_{t,0}$ is of rank $|\mathscr{F}|$, which requires at least $|\mathscr{F}|$ nodes to be active for the specific $t$ (i.e., $|\mathscr{S}_t| \geq |\mathscr{F}|$). This condition is similar to that of graph signal recovery via LMS [14], or RLS [17] on graphs. However, while in adaptive graph signal recovery the goal is to reconstruct $\tilde{\mathbf{x}}_0$ from multiple noisy realizations of the latter, here, we extend the recovery such that it encompasses also the model evolution (i.e., $\tilde{\mathbf{A}}_{t,0}$) into the analysis. The latter allows to take measurements in a graph-time fashion, resulting in so-called *graph-time* samples. In this context, we claim the following.

**Proposition 8.2.** *An $\mathscr{F}-$bandlimited system on graph is observable over the set $\mathscr{S}_{0:T}$ only if at least $|\mathscr{F}|$ graph-time samples are taken in the time interval $\{0,\dots,T\}$. These samples can be taken by $|\mathscr{F}|$ nodes at a fixed time instant, by one node in $|\mathscr{F}|$ time instants, or a combination of the two.*

Put simply, the condition in Proposition 8.2 is equivalent to

$$|\mathscr{S}_{0:T}| \geq |\mathscr{F}|, \tag{8.14}$$

i.e., the cardinality of the sampling set must be greater than or equal to the process bandwidth. However, (8.14) is only a necessary condition for observability. In fact, $\mathbf{O}_{0:T}$ may be easily ill-conditioned depending on the particular location of these samples and the spectral support of $\tilde{\mathbf{x}}_0$.

It is then paramount to carefully pick the samples in a graph-time fashion such that $\mathbf{O}_{0:T}$ is of full rank $|\mathscr{F}|$, and in the presence of noise $\mathbf{v}_t \neq 0$, possibly also well-conditioned. Put differently, the sampling set should satisfy

$$\operatorname{rank}\Big(\sum_{t=0}^{T} \tilde{\mathbf{A}}_{t,0}^{\mathsf{H}} \mathbf{U}_{\mathscr{F}}^{\mathsf{H}} \mathbf{C}_{\mathscr{S}_t} \mathbf{U}_{\mathscr{F}} \tilde{\mathbf{A}}_{t,0}\Big) = |\mathscr{F}|, \tag{8.15}$$

where the single shot graph signal reconstruction [11] is the special case $T = 0$. The following theorem generalizes the reconstruction of graph signals to a necessary and sufficient condition for the observability of an $\mathscr{F}$-bandlimited graph process over a sampling set.

**Theorem 8.1.** *An $\mathscr{F}-$bandlimited system on graph is observable over the set $\mathscr{S}_{0:T}$ if and only if*

$$\|\mathbf{C}_{\mathscr{S}_{0:T}^c}(\mathbf{I}_{T+1} \otimes \mathbf{U}_{\mathscr{F}})\| < \frac{s_{min}^2(\tilde{\mathbf{A}}_{0:T})}{s_{max}^2(\tilde{\mathbf{A}}_{0:T})}, \tag{8.16}$$

*where $\mathbf{C}_{\mathscr{S}_{0:T}^c} = \mathbf{I}_{N(T+1)} - \mathbf{C}_{\mathscr{S}_{0:T}}$ is the operator that projects onto the complementary set $\mathscr{S}_{0:T}^c = \{n \in \{1,\dots,N\}; t \in \{0,\dots,T\} \mid c_{t,n} = 0\}$ and $s_{min}(\tilde{\mathbf{A}}_{0:T})$, $s_{max}(\tilde{\mathbf{A}}_{0:T})$ indicate the minimum and maximum singular values of $\tilde{\mathbf{A}}_{0:T}$, respectively.*



Condition (8.16) is related to the localization properties of graph signals involving also the evolution model of the latter. It implies that in their *evolution* there are no $\mathscr{F}$-bandlimited graph processes perfectly localized on the complementary set $\mathscr{S}_{0:T}^c$. The single shot condition [11] is obtained for $T = 0$.

We conclude this part with the following observation.

**Remark 8.1.** *While (8.13) is an option to estimate $\mathbf{x}_0$ in the LS sense, one can also rely on the time-invariant results by considering only one realization $\mathbf{y}_t$. In this case, the presence of $\tilde{\mathbf{A}}_{t,0}$ should also be considered. Thus, when rank$(\tilde{\mathbf{A}}_{t,0}) < |\mathscr{F}|$, the recovery over singular observations is not possible. In a time-varying fashion, we exploit the successive realizations for estimating $\mathbf{x}_0$. Furthermore, since we must deal with noise, operating in a graph-time fashion makes the recovery more robust to bad noise realizations for a particular $t$.*

**MSE analysis.** We here quantify how the sampling set $\mathscr{S}_{0:T}$ affects the MSE of the LS estimate (8.13). The latter will be then used as a criterion to collect the graph-time samples. The main result is given by the following proposition.

**Proposition 8.3.** *Given an $\mathscr{F}$−bandlimited graph process following the model* (8.8) *and assuming the result of Theorem 8.1 holds. Then, the* MSE of the LS observed signal $\tilde{\mathbf{x}}_0^o$ is

$$\begin{aligned}\text{MSE} &= \mathbb{E}\left\{\|\tilde{\mathbf{x}}_0^o - \tilde{\mathbf{x}}_0\|^2\right\} = \mathbb{E}\left\{\text{tr}\left[(\tilde{\mathbf{x}}_0^o - \tilde{\mathbf{x}}_0)(\tilde{\mathbf{x}}_0^o - \tilde{\mathbf{x}}_0)^{\mathsf{H}}\right]\right\} \\ &= \sigma_v^2 \text{tr}\left\{\left[\tilde{\mathbf{A}}_{0:T}^{\mathsf{H}}(\mathbf{I}_{T+1}\otimes\mathbf{U}_{\mathscr{F}})^{\mathsf{H}}\mathbf{C}_{\mathscr{S}_{0:T}}(\mathbf{I}_{T+1}\otimes\mathbf{U}_{\mathscr{F}})\tilde{\mathbf{A}}_{0:T}\right]^{-1}\right\}.\end{aligned} \tag{8.17}$$

(The claim follows from the covariance matrix of the LS estimator [40].)

Besides characterizing the impact of the graph-time samples on the MSE[2], expression (8.17) shows that not only the number of selected samples plays a role, but also their location in graph and time. In the sequel, we show how to select these samples such that a target MSE (8.17) is guaranteed.

**Sampling strategy.** Given (8.17), we follow a sparse sensing approach [41, 42] to design the sampling set $\mathscr{S}_{0:T}$ such that a target MSE estimation performance is guaranteed. The latter is achieved as the solution of the convex problem

$$\begin{aligned}\underset{\mathbf{c}_{0:T}}{\text{minimize}} \quad & \mathbf{1}_{N\times(T+1)}^{\mathsf{T}}\mathbf{c}_{0:T} \\ \text{subject to} \quad & \text{tr}\left[\left(\mathbf{\Psi}_{0:T}^{\mathsf{H}}\mathbf{C}_{\mathscr{S}_{0:T}}\mathbf{\Psi}_{0:T}\right)^{-1}\right] \leq \frac{\gamma}{\sigma_v^2}, \\ & \mathbf{C}_{\mathscr{S}_{0:T}} = \text{diag}(\mathbf{c}_{0:T}), \\ & \mathbf{\Psi}_{0:T} = (\mathbf{I}_{T+1}\otimes\mathbf{U}_{\mathscr{F}})\tilde{\mathbf{A}}_{0:T}, \\ & 0 \leq c_{0:T,i} \leq 1, \quad i = 1,\ldots,N(T+1),\end{aligned} \tag{8.18}$$

where the objective function is the $l_1$-surrogate of the $l_0$-norm and imposes sparsity in $\mathscr{S}_{0:T}$; the constant $\gamma > 0$ imposes a target MSE performance; and the last constraint is the relaxation of the Boolean constraint $c_{0:T,i} \in \{0,1\}$ to the box one[3]. Alternatively, one can adopt a greedy approach similar to [13] for building $\mathscr{S}_{0:T}$. Obviously, we can also consider the opposite problem where the aim is to minimize the MSE, while imposing a fixed budget on the selected number of samples. The latter translates as well into a convex problem.



### 8.3.2. OBSERVABILITY WITH RANDOM SAMPLING

In this section, we consider the case where the entries of $\mathbf{C}_{\mathscr{S}_t}$ in (8.1b) are i.i.d. in time Bernoulli random variables with expected value $\bar{\mathbf{C}} = \text{diag}(\bar{\mathbf{c}})$. Let then $\bar{\mathscr{S}} = \{n \in \{1,\ldots,N\} | \bar{c}_n > 0\}$ be the expected sampling set, i.e., the set of nodes that are sampled with a probability greater than zero. The task here is to above questions w.r.t. $\bar{\mathscr{S}}$. As we show at the end of this section, one major benefit of this approach is that the sparse sensing design of $\bar{\mathscr{S}}$ avoids the relaxation techniques used in (8.18).

Given the measurements in the interval $\{0,\ldots,T\}$ (8.11), for a realization of $\mathbf{C}_{\mathscr{S}_{0:T}}$, we define

$$\mathbf{z}_{0:T} = \mathbf{y}_{0:T} - \mathbf{J}_{0:T}\mathbf{u}_{0:T} = \mathbf{O}_{0:T}\tilde{\mathbf{x}}_0 + \mathbf{C}_{\mathscr{S}_{0:T}}\mathbf{v}_{0:T}, \tag{8.19}$$

[2] The absence of model noise in (8.1a) allows us to find a closed-form expression for the MSE, rather than an upper bound. Moreover, it matches perfectly models (8.2)-(8.6).

[3] In a second step, randomized rounding or thresholding can be used to project the optimal solution $\mathbf{c}_{0:T}^*$ of (8.18) to the $\{0,1\}^{N(T+1)}$ space [41].

i.e., we subtract each realization[4] of the input signal before analyzing the observability properties. From (8.14), a necessary condition for the instantaneous observability matrix $\mathbf{O}_{0:T}$ to be full rank is that the instantaneous sampling set $\mathcal{S}_{0:T}$ in $\{0,\dots,T\}$ has a cardinality greater than, or equal to, the signal bandwidth. Given the structure of $\mathbf{O}_{0:T}$ ($\mathbf{C}_{\mathcal{S}_{0:T}}$) in (8.12), it is obvious that $\text{rank}(\mathbf{O}_{0:T})$ $\big(\text{rank}(\mathbf{C}_{\mathcal{S}_{0:T}})\big)$ depends on the $\text{rank}(\bar{\mathbf{C}})$, i.e., on the cardinality of the nodes that are sampled with a probability strictly greater than zero. The subsequent proposition formalizes the above as a necessary condition.

**Proposition 8.4.** *Consider an $\mathcal{F}-$bandlimited system on graph and given the diagonal sampling matrix $\mathbf{C}_{\mathcal{S}_t}$ with i.i.d. in time Bernoulli entries and expected value $\bar{\mathbf{C}}$. A necessary condition for the observability of the system from samples taken randomly in the interval $\{0,\dots,T\}$ is that at least $\lceil|\mathcal{F}|/(T+1)\rceil$ nodes are sampled with a probability greater than zero.*

That is, differently from the deterministic node sampling, the observability of a graph process is now related to the expected sampling set $\bar{\mathcal{S}}$. Put simply, the constraint in Proposition 8.4 is equivalent to

$$|\bar{\mathcal{S}}| \geq \lceil|\mathcal{F}|/(T+1)\rceil. \tag{8.20}$$

It must be noted that for $T \geq |\mathcal{F}|$ there is the potential to observe an $\mathcal{F}-$bandlimited graph process by allowing only one node to randomly take measurements. The above result, though novel from the random sampling viewpoint, is not entirely surprising. In fact, in [10] it has been seen that a graph signal can be reconstructed also by sampling successive aggregations of a single node. Hence, by bringing the time into the play, one node can collect different linearly independent measurements in time and will be able to observe the process.

However, since the node sampling is random in a finite interval $\{0,\dots,T\}$, there is always a possibility that the instantaneous sampling set $\mathcal{S}_{0:T}$ has a cardinality smaller than $|\mathcal{F}|$. The following corollary quantifies the latter.

**Corollary 8.1.** *Given the sampling matrix $\mathbf{C}_{\mathcal{S}_t}$ in (8.1) with i.i.d. in time Bernoulli entries and expected value $\bar{\mathbf{C}}$. The probability that the cardinality of the instantaneous sampling set $\mathcal{S}_{0:T}$ is smaller than the process bandwidth $|\mathcal{F}|$ is*

$$\Pr\big(|\mathcal{S}_{0:T}| < |\mathcal{F}|\big) = \sum_{k=0}^{|\mathcal{F}|-1} \frac{\alpha^k e^{-\alpha}}{k!}, \tag{8.21}$$

*where $\alpha = (T+1)\mathbf{1}_N^\top \bar{\mathbf{c}}$ is the mean of the Poisson distribution.*

This probability drops to zero even for moderate values $\bar{\mathbf{c}}$ as long as $N$ is of the order of 100 nodes and $T$ is relatively large. In Section **??**, we show with real data that this probability drops below machine precision. To further quantify the impact of $\bar{\mathbf{C}} = \text{diag}(\bar{\mathbf{c}})$ on the process observability, we perform next an MSE analysis of the LS estimated state.

[4] This is analogous to our graph deterministic observability, or observability in linear systems, where the realizations of the input signal should be known. This is not a problem since the realization of $\mathbf{C}_{\mathcal{S}_{0:T}}$ is known.

**MSE analysis.** To render a MSE analysis tractable, we follow a similar procedure as used for the Cramér-Rao lower bound[5] (CRLB) [40], which quantifies the lowest MSE estimate $\tilde{\mathbf{x}}_0^o = \mathbf{O}_{0:T}^{\dagger}\mathbf{z}_{0:T}$. The following proposition quantifies this finding.

**Proposition 8.5.** *Given an $\mathscr{F}$ – bandlimited graph process following model* (8.8) *and given $\mathbf{C}_{\mathscr{S}_t}$ a diagonal sampling matrix with i.i.d. in time Bernoulli entries and expected value $\bar{\mathbf{C}}$. The* MSE of the LS observed signal $\tilde{\mathbf{x}}_0^o = \mathbf{O}_{0:T}^{\dagger}\mathbf{z}_{0:T}$ is then lower-bounded by

$$\text{MSE} \geq \sigma_{\text{v}}^2 \text{tr}\left\{\left[\tilde{\mathbf{A}}_{0:\text{T}}^{\text{H}}\left(\mathbf{I}_{\text{T}+1} \otimes \mathbf{U}_{\mathscr{F}}^{\text{H}}\bar{\mathbf{C}}\mathbf{U}_{\mathscr{F}}\right)\tilde{\mathbf{A}}_{0:\text{T}}\right]^{-1}\right\}. \tag{8.22}$$

Besides providing a statistical measure of the lowest achievable MSE for a particular $\bar{\mathbf{C}}$, the lower bound (8.22) can also be used as a design criterion to find these sampling probabilities. This aspect is covered in more detail next.

**Sampling strategy.** Following the same principle as in [42], we design the expected sampling set $\bar{\mathscr{S}}$ in a sparse sensing fashion, where instead of using the CRLB as a design criterion, we consider the lower bound (8.22). Then, $\bar{\mathbf{c}}$ and therefore $\bar{\mathscr{S}}$ are found as the solution of the convex problem

$$\begin{aligned}
\underset{\bar{\mathbf{c}},\boldsymbol{\gamma}\in\mathbb{R}^{|\mathscr{F}|}}{\text{minimize}} \quad & \mathbf{1}_N^{\text{T}}\bar{\mathbf{c}} \\
\text{subject to} \quad & \text{tr}\left\{\left[\tilde{\mathbf{A}}_{0:T}^{\text{H}}\left(\mathbf{I}_{T+1} \otimes \mathbf{U}_{\mathscr{F}}^{\text{H}}\bar{\mathbf{C}}\mathbf{U}_{\mathscr{F}}\right)\tilde{\mathbf{A}}_{0:T}\right]^{-1}\right\} \leq \frac{\gamma}{\sigma_v^2}, \\
& \bar{\mathbf{C}} = \text{diag}(\bar{\mathbf{c}}), \\
& c_{\min} \leq \bar{c}_n \leq c_{\max}, \quad n = 1, 2, \ldots, N, \\
& 0 \leq c_{\min} \leq c_{\max} \leq 1.
\end{aligned} \tag{8.23}$$

Even though conceptually equivalent to (8.18), problem (8.23) differs in two main aspects, which preserve the optimality of the solution. First, the objective function is not a surrogate anymore of the $l_0$-norm. Rather, it is the true function (i.e., the overall sampling rate) that we want to minimize. Second, the convex box constraint $c_{\min} \leq \bar{c}_n \leq c_{\max}$ is not a relaxation anymore, since now we directly optimize over the sampling probabilities for some rate allocation bounds $c_{\min}$ and $c_{\max}$. In (8.23), one can add also a constraint on the probability criterion (8.21). However, the latter should be upper-bounded since it is not a convex function. As we show in the next section with the ETEX dataset, the latter is not necessary since a small enough $\gamma$ on the MSE will trade well the sampling probabilities with the performance.



### 8.3.3. NUMERICAL RESULTS

We start our numerical analysis with the observability with deterministic sampling on the Molene weather data set[6] and then we analyze the observability with random sampling on the European tracer experiment (ETEX) data set[7] [28].

[5] Since the measurements are the product of a Bernoulli and a Gaussian random variable, the joint pdf does not satisfy the CRLB regularity condition.

[6] Data publicly available at `https://donneespubliques.meteofrance.fr/donnees_libres/Hackathon/RADOMEH.tar.gz`.

[7] Data publicly available at https://rem.jrc.ec.europa.eu/RemWeb/etex/ .

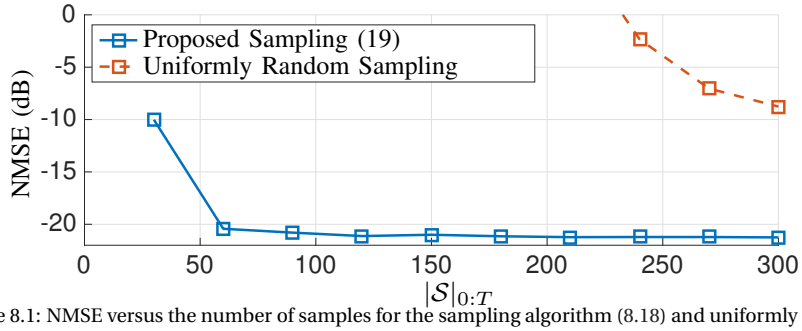



Figure 8.1: NMSE versus the number of samples for the sampling algorithm (8.18) and uniformly random sampling. The graph process has a perfectly localized spectrum on $|\mathscr{F}| = N = 32$, i.e., the entire bandwidth.

Table 8.1: Theoretical, empirical NMSE, and the cardinality of the sampling set $\mathscr{S}_{0:T}$ for different values of $\gamma$ in (8.18).

| | $\gamma = 2.05$ | $\gamma = 2.5$ | $\gamma = 3$ | $\gamma = 3.5$ |
|---|---|---|---|---|
| Theo. (8.17) | $-21.26$dB | $-20.42$dB | $-19.64$dB | $-19.32$dB |
| Emp. | $-21.22$dB | $-20.37$dB | $-19.57$dB | $-19.28$dB |
| $\lvert\mathscr{S}_{0:T}\rvert$ | 277 | 61 | 37 | 32 |

**Obs. with deterministic sampling.** The Molene weather data set consists of $R = 744$ hourly temperature recordings collected in January 2014 over 32 cities in the region of Brest, France. The graph is a $k$-nearest neighbour ($k$NN) [43] graph with $k = 3$. We consider a single recording[8] and then diffuse it following model (8.2) with $w = 1.5$ and $T = 10$.

First, we analyze the effect of the sampling set $\mathscr{S}_{0:T}$ when the graph process is perfectly $\mathscr{F}-$bandlimited. In this regard, we considered $|\mathscr{F}| = N = 32$ (i.e., the entire bandwidth) and corrupted the measurements with a zero-mean Gaussian noise with $\sigma_v^2 = 10^{-1}$, which corresponds to an average signal-to-noise ratio (SNR) of 19.3dB computed as

$$\overline{\text{SNR}} = 10\log_{10}\left[\frac{\sum_{\tau=1}^{R}\|\mathbf{r}_\tau\|_2^2}{NR\sigma_v^2}\right]. \tag{8.24}$$

Here, $\mathbf{r}_\tau$ stands for the $\tau$th recording. The $|\mathscr{S}_{0:T}|$ samples are chosen by solving the opposite of problem (8.18) as the ones that minimize the MSE (8.17) in a sparse sense fashion. As a performance evaluation criterion, we use the normalized MSE (NMSE) between the estimated (observed) $\tau$th recording $\mathbf{r}_\tau^o$ and the true one $\mathbf{r}_\tau$, defined as

$$\text{NMSE} = \frac{\sum_{\tau=1}^{R}\|\mathbf{r}_\tau^o - \mathbf{r}_\tau\|^2}{\sum_{\tau=1}^{R}\|\mathbf{r}_\tau\|^2}. \tag{8.25}$$

Fig. 8.1 shows the obtained NMSE as a function of $|\mathscr{S}_{0:T}|$. It can be seen that even with 60 samples (out of 320) an NMSE of $-20$dB is achieved. On the contrary, the uniformly

[8] The graph signal consists of the measured temperature after subtracting their average value.

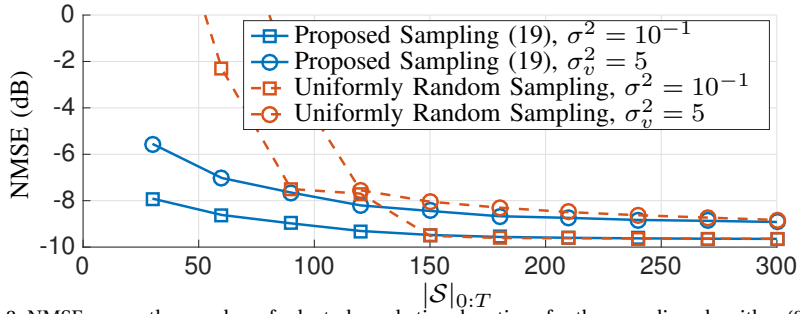



Figure 8.2: NMSE versus the number of selected graph-time locations for the sampling algorithm (8.18) and a uniformly random sampling. The state spectral evolution is considered localized on $\mathcal{F}$ with $|\mathcal{F}| = 8$.

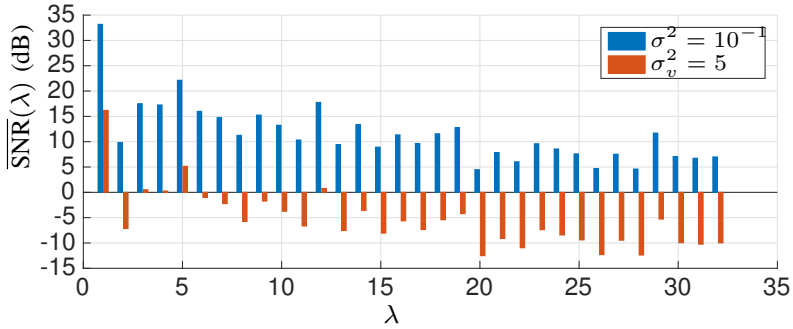



Figure 8.3: Average SNR per graph frequency for the two different noise powers computed as $\overline{\text{SNR}}(\lambda_n) = \sum_{\tau=1}^{744} \hat{r}_n^2 / T\sigma_v^2$. In the high noise regime, we observe that most frequencies experience a negative SNR.



random sampling[9] requires far more measurements to give a comparable performance. *This finding suggests that the sparse observability approach can also be implemented for graph processes that have a contribution on the entire bandwidth.*

To provide more insights, Table 8.1 shows the theoretical and empirical NMSE as a function of the target value $\gamma$ in (8.18). In addition, we show also the cardinality of $\mathcal{S}_{0:T}$. We observe that a stricter NMSE requirement in (8.18) leads to a higher $|\mathcal{S}_{0:T}|$ and, vice-versa, a bigger $\gamma$ leads to a sparser $\mathcal{S}_{0:T}$.

In the second scenario, we restrict the process bandwidth to the first $|\mathcal{F}| = 8$ graph frequencies and analyze two different noise variances $\sigma_v^2 = \{10^{-1}, 5\}$ ($\overline{\text{SNR}} = \{19.3\text{dB}, 2.3\text{dB}\}$). The sampling set $\mathcal{S}_{0:T}$ is built as in the previous scenario and is again compared with the uniformly random sampling.

Fig. 8.2 depicts the average NMSE as a function of $|\mathcal{S}_{0:T}|$, where the proposed selection strategy outperforms again the uniformly random sampling. We further observe that the NMSE has a lower floor much higher than for the full bandwidth case and its value does not reduce even by increasing $|\mathcal{S}_{0:T}|$. We attribute this limitation to the re-

[9] To account for the randomness in this sampling strategy the NMSE is further averaged over 100 iterations.

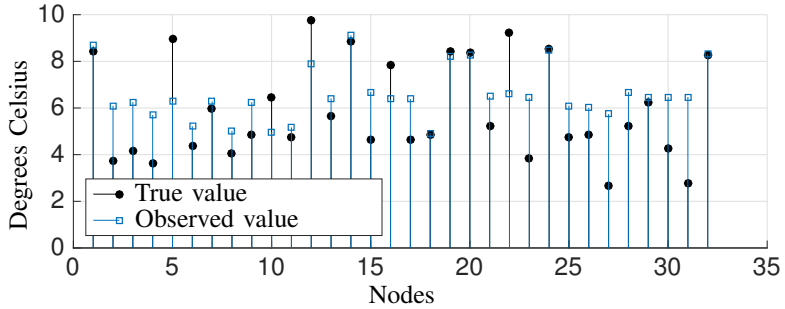



Figure 8.4: True temperature values and the observed ones for a random recording; $|\mathscr{F}| = 8$, $|\mathscr{S}_{0:T}| = 60$, and $\sigma_v^2 = 10^{-3}$. Further improvement can be obtained by increasing $|\mathscr{F}|$.

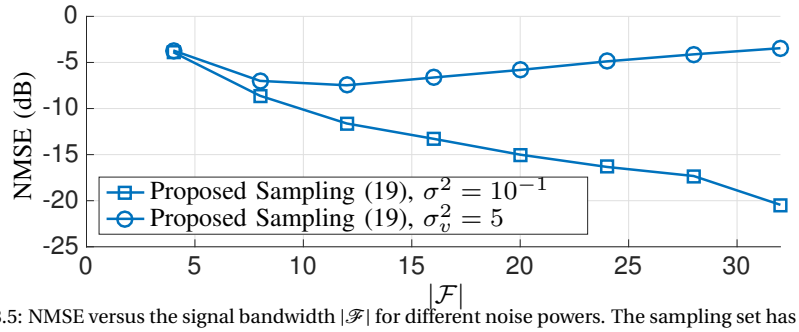



Figure 8.5: NMSE versus the signal bandwidth $|\mathscr{F}|$ for different noise powers. The sampling set has cardinality $|\mathscr{S}_{0:T}| = 100$ chosen by minimizing the MSE (8.17). Observe that a larger bandwidth is not favorable when the measurement noise has high power.

8

stricted bandwidth, since the out-of-band signal contribution seems playing a role in improving further the performance. In fact, w.r.t. Fig. 8.3, we observe that in low noise regimes it is beneficial to consider a larger bandwidth since the average SNR per frequency is high. On the contrary, this might not be the case for $\sigma_v^2 = 5$, since the $\overline{\text{SNR}}(\lambda_n)$ is negative for high graph frequencies. In the sequel, we show that indeed the $\overline{\text{SNR}}(\lambda_n)$ plays a crucial role in the observability performance. Fig. 8.4 concludes this scenario by plotting the true signal and the corresponding observed signal with $|\mathscr{S}_{0:T}| = 60$ and $\sigma_v^2 = 10^{-1}$ for a random recording.

In this third scenario, we analyze the effects of the signal bandwidth on the observability performance. We fix $|\mathscr{S}_{0:T}| = 60$ samples (i.e., almost twice the full bandwidth) and compute the NMSE for different values of $|\mathscr{F}|$ and $\sigma_v^2$. These results are shown in Fig. 8.5.

We observe an increasing trend of the NMSE in high noise regimes (i.e., $\sigma_v^2 = 5$). This suggests that the meaningful information is concentrated in the first few frequencies and, therefore, the graph process is bandlimited. As highlighted in Fig. 8.3, by increasing $|\mathscr{F}|$ we only add more noise resulting in a performance degradation. *This result suggests that in the presence of noise the process bandwidth should not be determined solely by*

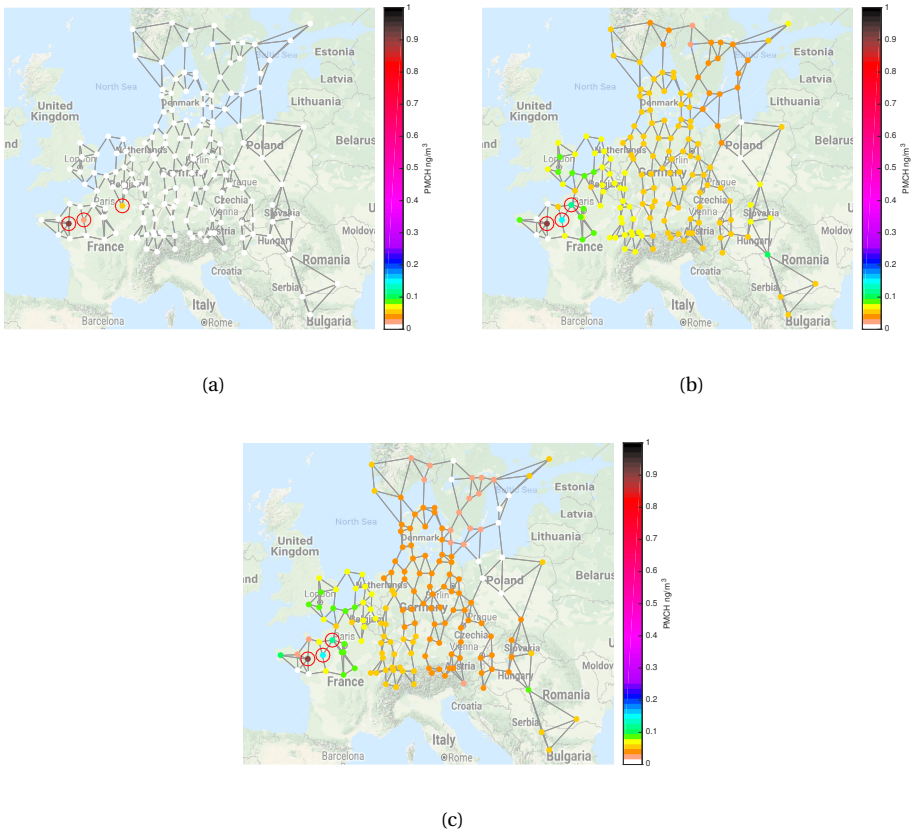



Figure 8.6: Tracer (PMCH) concentration in the 168 stations. The top three nodes with the highest PMCH concentration are circled in red. (a) Ground truth concentration at $t = 0$. (b) Observed tracer concentration by the intantaneous diffusion model with $w = 3.5$ on the 3NN graph with all nodes collecting samples. (c) Mean observed tracer concentration following the observability with random sampling and overall sampling rate $\mathbf{1}_N^\mathsf{T}\bar{\mathbf{c}} = 60$ (out of 168) by solving the opposite of problem (8.23). The NMSE between the observed signal with random sampling (right) and the reconstructed ground truth (center) is $-16.2$ dB with a variance of $-48.1$dB around this value.

*the signal energy, but by the signal-to-noise ratio (SNR).* Indeed, a larger bandwidth (although the signal has energy content) degrades the overall $\overline{\text{SNR}}$. This finding is further reinforced in the low noise regime, where a larger bandwidth is preferred to exploit the SNR on the high frequencies for better observing the graph process.

**Obs. with random sampling.** The ETEX experiment [28] contains measurements of an identifiable perfluorocarbon concentration, released near Rennes, France, and then diffused over Europe. Thirty concentration measurements were collected over a period of 72 hours at $N = 168$ ground-level stations. These stations will serve as the nodes of a $k$NN graph and the 30 collected measurements in time will be the graph process. For several reasons, the measurements are not always available and in these cases, the tracer

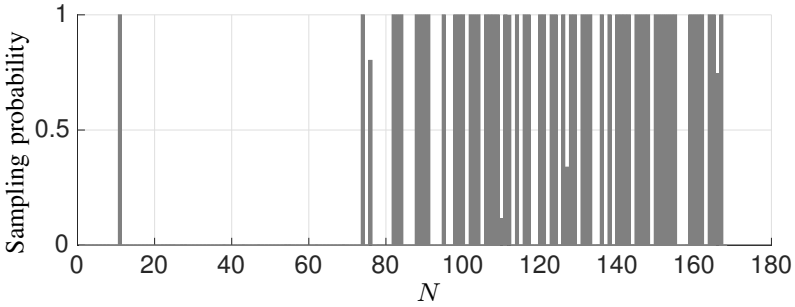



Figure 8.7: Optimal sampling probabilities over the nodes obtained from solving the opposite of problem (8.23). We observe that several nodes are sampled with probability one and that the overall solution is highly sparse.

concentration is set to zero.

We considered model (8.2) to capture the signal evolution over time. The set $\mathscr{F}$ contains the frequency indices where 30% of the process energy is concentrated ($|\mathscr{F}| = 6$). Since diffused graph signals are often bandlimited, our intuition is that also in this experiment most of the frequencies will not have useful information. With this setup, we found heuristically that $k = 3$ and $w = 3.5$ lead to the smallest observability error by using all $(168 \times 72)$ recordings. We then use this result to test the observability with random sampling. To account for the measurement noise, the signal is corrupted with a zero-mean Gaussian noise of variance $\sigma_v^2 = 10^{-4}$ ($\overline{\text{SNR}} = 29.2$dB). The obtained results are averaged over 2000 iterations.

In Fig. **??** (a), we plot the original signal at $t = 0$, wherein red circles highlight the top three nodes with the highest concentration. Then, in Fig. **??** (b) we plot the initial signal reconstructed by using model (8.2) when all nodes collect data. For this instance, the fitted graph and the used model are capable to identify the region (specifically the top two highest concentrations) where the tracer was released, but at the same time a tracer concentration around 0.04ng/m$^3$ is also observed over all nodes[10]. However, for this work, we will use this fully observed signal (Fig. **??** (b)) as a benchmark since it is the best that we can reconstruct with the fitted model. In Fig. **??** (c), we show the average observed signal with a sampling rate of 60. The sampling probabilities are obtained by solving the opposite of problem (8.23) and are illustrated in Fig. 8.7. We achieved an average NMSE between the observed signal and the reconstructed ground truth (Fig. **??** (b)) of $-16.2$dB with a variance around this performance of $-48.1$dB.

*The above results lead to the following conclusions: i) the deviation of a particular realization from the averaged observed signal is in general negligible, yielding a good practical result; and ii) similarly to the approaches that use the CRLB to perform sparse sampling, the lower bound* (8.22) *is a suitable cost function to design a sparse sampler.*

Finally, in Table 8.2 we address the impact of $\gamma$ in (8.23) on the lower bound (8.22), the empirical NMSE, the overall sampling rate, and the value of $\alpha$ in (8.21). We observe

[10] We attribute this concentration leakage to the missing values that are set to zero and to the absence of wind information on the specific days.

Table 8.2: Theoretical, empirical NMSE, overall sampling rate, and $\alpha$ in (8.21) for different values of $\gamma$ ($\times 10^{-4}$) in (8.18).

| | $\gamma = 3.12$ | $\gamma = 3.15$ | $\gamma = 3.18$ | $\gamma = 3.21$ |
|---|---|---|---|---|
| Theo. Lower Bound (8.22) | $-36.47$dB | $-36.42$dB | $-36.38$dB | $-36.34$dB |
| Emp. | $-24.52$dB | $-18.77$dB | $-16.48$dB | $-16.13$dB |
| $\mathbf{1}_N^{\mathsf{T}}\bar{\mathbf{c}}$ | 130.4 | 83.9 | 62.7 | 50.2 |
| $\alpha$ | 3910 | 2515 | 1881 | 1505 |

that despite the gap between the theoretical lower bound and the empirical NMSE, a looser requirement on (8.22) induces a lower sampling rate. Moreover, all the reported values of $\alpha$ lead to a probability (8.21) below machine precision. This demonstrates the use of (8.22) for sparse sampling design and that a reasonable NMSE is achieved even by collecting 1/3 of the measurements.

## 8.4. TRACKING GRAPH PROCESSES

We now consider the task of tracking a bandlimited graph process from a subset of nodes chosen deterministically. We make use of the Kalman filter, which matches perfectly the system (8.8). First, we introduce the KF algorithm for time-varying scenarios and provide conditions on the sampling set to optimally track the graph process. Then, we show how the proposed KF specializes for time-invariant models (as the ones in Section 8.2.1) and provide conditions for the sampling set to ensure a steady-state performance. For both cases, we provide sampling strategies for designing the sampling set with given tracking guarantees.

### 8.4.1. KALMAN FILTERING FOR TIME-VARYING MODELS

Following [39], for the $\mathscr{F}$−bandlimited system (8.8), the KF on graphs evolves as described in Algorithm 8.1. It initializes the *a posteriori* state estimate $\tilde{\mathbf{x}}_0^+$ to a random vector and the *a posteriori* error covariance matrix $\mathbf{P}_0^+$ to a scaled identity. The update of $\mathbf{P}_t^-$ in step $ii$) accounts also for the state model noise $\tilde{\mathbf{w}}_t = \mathbf{U}_{\mathscr{F}}\mathbf{w}_t$, which is considered zero-mean with covariance matrix $\tilde{\mathbf{\Sigma}}_w = \mathbf{U}_{\mathscr{F}}^{\mathsf{H}}\mathbf{\Sigma}_w\mathbf{U}_{\mathscr{F}}$ and independent from $\mathbf{v}_t$.

The Kalman gain matrix $\mathbf{K}_t$ computed in step $iii$) leads to the minimum average *a posteriori* MSE, i.e., $\mathrm{tr}(\mathbf{P}_t^+)$. From its expression, it is clear that $\mathbf{K}_t$ (and thus $\mathbf{P}_t^+$) is highly correlated with the sampling set at time $t$, i.e., $\mathbf{C}_{\mathscr{S}_t}$. In fact, for $\mathrm{rank}(\mathbf{C}_{\mathscr{S}_t}) = R < |\mathscr{F}|$ and assuming $\mathrm{rank}(\mathbf{P}_t^-) = |\mathscr{F}|$, then $\mathrm{rank}(\mathbf{C}_{\mathscr{S}_t}\mathbf{U}_{\mathscr{F}}\mathbf{P}_t^-\mathbf{U}_{\mathscr{F}}^{\mathsf{H}}\mathbf{C}_{\mathscr{S}_t} + \mathbf{C}_{\mathscr{S}_t}\mathbf{\Sigma}_v\mathbf{C}_{\mathscr{S}_t}) = R$. As a consequence $\mathrm{rank}(\mathbf{K}_t) \leq R < |\mathscr{F}|$. Thus, the $|\mathscr{F}| \times N$ Kalman gain matrix $\mathbf{K}_t$ can be full rank only if

$$|\mathscr{S}_t| \geq |\mathscr{F}|. \tag{8.26}$$

That is, KF on graphs will fully exploit its Kalman gain (and thus track better) only if the number of sampled nodes *for each* $t$ is greater than or equal to the signal bandwidth.

**Algorithm 8.1. : Kalman filtering on graphs**

Initialize $\tilde{\mathbf{x}}_0^+$ and $\mathbf{P}_0^+$. For $t>0$ repeat:
i) Update the *a priori* state estimate $\tilde{\mathbf{x}}_t^-$ as:

$$\tilde{\mathbf{x}}_t^- = \tilde{\mathbf{A}}_{t-1}\tilde{\mathbf{x}}_{t-1}^+ + \tilde{\mathbf{B}}_{t-1}\tilde{\mathbf{u}}_{t-1};$$

ii) Update the *a priori* error covariance matrix $\mathbf{P}_t^-$ as:

$$\mathbf{P}_t^- = \tilde{\mathbf{F}}_{t-1}\mathbf{P}_{t-1}^+\tilde{\mathbf{F}}_{t-1}^{\mathsf{H}} + \tilde{\mathbf{\Sigma}}_w;$$

iii) Compute the Kalman gain matrix $\mathbf{K}_t$ as:

$$\mathbf{K}_t = \mathbf{P}_t^-\mathbf{U}_{\mathscr{F}}^{\mathsf{H}}\mathbf{C}_{\mathscr{S}_t}\left(\mathbf{C}_{\mathscr{S}_t}\mathbf{U}_{\mathscr{F}}\mathbf{P}_t^-\mathbf{U}_{\mathscr{F}}^{\mathsf{H}}\mathbf{C}_{\mathscr{S}_t} + \mathbf{C}_{\mathscr{S}_t}\mathbf{\Sigma}_v\mathbf{C}_{\mathscr{S}_t}\right)^{\dagger};$$

iv) Update the *a posteriori* state estimate $\tilde{\mathbf{x}}_t^+$ as:

$$\tilde{\mathbf{x}}_t^+ = \tilde{\mathbf{x}}_t^- + \mathbf{K}_t\left(\mathbf{y}_t - \mathbf{C}_{\mathscr{S}_t}\mathbf{U}_{\mathscr{F}}\tilde{\mathbf{x}}_t^-\right);$$

v) Update the *a posteriori* error covariance matrix $\mathbf{P}_t^+$ as:

$$\begin{aligned}\mathbf{P}_t^+ = \mathbf{P}_t^- &- \mathbf{P}_t^-\mathbf{U}_{\mathscr{F}}^{\mathsf{H}}\mathbf{C}_{\mathscr{S}_t}\mathbf{K}_t^{\mathsf{H}} - \mathbf{K}_t\mathbf{C}_{\mathscr{S}_t}\mathbf{U}_{\mathscr{F}}\mathbf{P}_t^- \\ &+ \mathbf{K}_t\mathbf{C}_{\mathscr{S}_t}\mathbf{\Sigma}_v\mathbf{K}_t^{\mathsf{H}} + \mathbf{K}_t\mathbf{C}_{\mathscr{S}_t}\mathbf{U}_{\mathscr{F}}\mathbf{P}_t^-\mathbf{U}_{\mathscr{F}}^{\mathsf{H}}\mathbf{C}_{\mathscr{S}_t}\mathbf{K}_t^{\mathsf{H}}.\end{aligned}$$

The necessary condition (8.26) extends condition (8.14) from the observability of a bandlimited graph process in an interval to the tracking task.

The impact of the sampled nodes $\mathbf{C}_{\mathscr{S}_t}$ in KF is highlighted in $\mathbf{P}_t^+$ (e.g., step v) in Algorithm 8.1). In the sequel, we will exploit this benefit to design $\mathscr{S}_t$ such that a target *a posteriori* MSE estimation accuracy is guaranteed.

**Sampling strategy.** Condition (8.26) suggests that there is a minimum number of nodes, tightly related to the signal bandwidth, that must be sampled to fully exploit the Kalman gain matrix. However, from the expression of $\mathbf{P}_t^+$, it is once again clear that their location in the graph is as important as $|\mathscr{S}_t|$ to achieve a good tracking performance.

To optimally select the sampled nodes, we adopt a sparse sensing approach that involves the posterior CRB (PCRB) for the state estimation $\tilde{\mathbf{x}}_t$ [42, 44, 45]. Given the log-likelihood of the measurements satisfies the regularity condition $\mathbb{E}[\partial \ln p(\mathbf{y}_t;\tilde{\mathbf{x}}_t)/\partial\tilde{\mathbf{x}}_t] = \mathbf{0}_N$, for all $t$, the PCRB satisfies

$$\mathbf{P}_t^+ = \mathbb{E}\left[\left(\tilde{\mathbf{x}}_t^+ - \tilde{\mathbf{x}}_t\right)\left(\tilde{\mathbf{x}}_t^+ - \tilde{\mathbf{x}}_t\right)^{\mathsf{H}}\right] \geq \mathbf{F}_t^{-1}(\tilde{\mathbf{x}}_t), \tag{8.27}$$

where $\mathbf{F}_t(\tilde{\mathbf{x}}_t)$ is the posterior Fisher information matrix.

For linear systems in additive Gaussian noise, as the one considered in this work, relation (8.27) holds with equality (i.e., the Kalman filter is optimal) and is independent of $\tilde{\mathbf{x}}_t$. This suggests that designing the sampling set w.r.t. $\mathbf{F}_t(\tilde{\mathbf{x}}_t)$ leads to the same result as working with the *a posteriori* error covariance matrix $\mathbf{P}_t^+$. For the KF in Algorithm 8.1

the posterior FIM is [45]:

$$\mathbf{F}_t(\tilde{\mathbf{x}}_t) = \left(\tilde{\mathbf{A}}_t \mathbf{F}_{t-1}^{-1}(\tilde{\mathbf{x}}_t)\tilde{\mathbf{A}}_t^{\mathsf{T}} + \mathbf{\Sigma}_{\tilde{w}}\right)^{-1} + \sum_{n=1}^{N} c_{t,n}\mathbf{F}_{t,n}^{o}(\tilde{\mathbf{x}}_t), \tag{8.28}$$

where $c_{t,n}$ is the $n$th diagonal entry of $\mathbf{C}_{\mathscr{S}_t}$, and $\mathbf{F}_{t,n}^{o}(\tilde{\mathbf{x}}_t) = \sigma_v^{-2} c_{t,n}\mathbf{u}_{\mathscr{F},n}\mathbf{u}_{\mathscr{F},n}^{\mathsf{T}}$ is the FIM related to the $n$th node observation at time $t$. The first term in (8.28) denotes the prior FIM related to the tracking history up to $t-1$.

By substituting the expression for $\mathbf{F}_{t,n}^{o}(\tilde{\mathbf{x}}_t)$ and rearranging the sum, we obtain the *a posteriori* FIM

$$\mathbf{F}_t(\tilde{\mathbf{x}}_t) = \left(\tilde{\mathbf{A}}_t \mathbf{F}_{t-1}^{-1}(\tilde{\mathbf{x}}_t)\tilde{\mathbf{A}}_t^{\mathsf{T}} + \mathbf{\Sigma}_{\tilde{w}}\right)^{-1} + \mathbf{U}_{\mathscr{F}}^{\mathsf{H}}\mathbf{C}_{\mathscr{S}_t}\mathbf{\Sigma}_v^{-1}\mathbf{U}_{\mathscr{F}}. \tag{8.29}$$

Following once again the sparse sensing idea, the instantaneous sampling set $\mathscr{S}_t$ can be built by solving

$$\begin{aligned} \underset{\mathbf{c}_t}{\text{minimize}} \quad & \mathbf{1}^{\mathsf{T}}\mathbf{c}_t \\ \text{subject to} \quad & \mathbf{F}_t(\tilde{\mathbf{x}}_t) \geq \gamma \mathbf{I}_{|\mathscr{F}|}, \\ & \mathbf{C}_{\mathscr{S}_t} = \text{diag}(\mathbf{c}_t), \\ & \mathbf{0}_N \leq \mathbf{c}_t \leq \mathbf{1}_N. \end{aligned} \tag{8.30}$$

Problem (8.30) generalizes (8.18) to the time-varying case, where all the remarks about the optimallity of the solution extend also here.

**Remark 8.2.** *From a practical viewpoint, the presented KF approach presents two main challenges in large graphs. First, the computation* $\mathbf{K}_t$ *involves the pseudo-inverse of an* $|\mathscr{F}| \times N$ *matrix with at most* $|\mathscr{S}_t|^2$ *(by construction) non zero elements. The latter may result computationally prohibitive for* $|\mathscr{S}_t| \to N$*. This issue can be easily addressed with the sequential implementation of the Kalman filter [39]. Second, the node selection strategy involves solving for each time instant t an SDP problem, which for large N may result in prohibited costs [46]. The latter issue can be addressed with greedy solutions such as [47].*



### 8.4.2. STEADY-STATE KALMAN FILTERING ON GRAPHS

We focus here on a time-invariant $\mathscr{F}-$bandlimited system on graphs, which specializes the above derivations to models (8.2)-(8.6). These systems often lead to a convergent state, which can be exploited to design a fixed sampling set for all $t$, i.e., $\mathbf{C}_{\mathscr{S}_t} = \mathbf{C}_{\mathscr{S}}$ for all $t$ with given steady-state performance guarantees.

From [39], the *a priori* error covariance matrix $\mathbf{P}_t^-$ will converge to the unique limit $\mathbf{P}_\infty$ if:

*i*) the pair $(\tilde{\mathbf{A}}, \tilde{\mathbf{B}})$ is stabilizable;

*ii*) the pair $(\tilde{\mathbf{A}}, \mathbf{C}_{\mathscr{S}}\mathbf{U}_{\mathscr{F}})$ is detectable.

The first condition is a characteristic of the graph process and is application specific. However, for stable time-invariant graph processes (as the one of interest in this section) this condition is satisfied. The second condition restricts the sampled nodes and their location in the graph to guarantee the steady-state convergence. From linear systems theory, a useful result is that an observable system is also detectable. Thus, by exploiting

the findings in Section 8.3.1, the KF on graphs is convergent if the limiting observability matrix

$$\mathbf{O}_\infty = \lim_{T\to\infty} \mathbf{O}_{0:T} = \lim_{T\to\infty} (\mathbf{I}_{T+1} \otimes \mathbf{C}_{\mathscr{S}}\mathbf{U}_{\mathscr{F}})\tilde{\mathbf{A}}_{0,T} \tag{8.31}$$

is full rank, or with similar arguments as in Theorem 8.1 if

$$\lim_{T\to\infty} \|\mathbf{I}_{T+1} \otimes \mathbf{U}_{\mathscr{F}}^\top \mathbf{D}_{\mathscr{S}^c}\mathbf{U}_{\mathscr{F}}\| \leq \lim_{T\to\infty} \frac{s^2_{\min}(\tilde{\mathbf{A}}_{0:T})}{s^2_{\max}(\tilde{\mathbf{A}}_{0:T})}, \tag{8.32}$$

where $\mathscr{S}^c = \mathscr{V}/\mathscr{S}$ denotes again the complementary sampling set. Conditions (8.31), once again relates the complementary sampling set with the localization properties of the graph process throughout its temporal evolution (though in practice it is sufficient to hold for $T \gg 0$).

For a fixed $\mathscr{S}$, the *a priori* error covariance matrix satisfies the discrete algebraic Riccatti equation (DARE)

$$\begin{aligned}\mathbf{P}_\infty = \tilde{\mathbf{A}}\mathbf{P}_\infty\tilde{\mathbf{A}}^\top + \tilde{\mathbf{\Sigma}}_w - \tilde{\mathbf{A}}\mathbf{P}_\infty\mathbf{U}_{\mathscr{F}}^\top\mathbf{C}_{\mathscr{S}} \times \\ \left(\mathbf{C}_{\mathscr{S}}\mathbf{U}_{\mathscr{F}}\mathbf{P}_\infty\mathbf{U}_{\mathscr{F}}^\top\mathbf{C}_{\mathscr{S}} + \mathbf{C}_{\mathscr{S}}\mathbf{\Sigma}_v\mathbf{C}_{\mathscr{S}}\right)^\dagger \mathbf{C}_{\mathscr{S}}\mathbf{U}_{\mathscr{F}}\mathbf{P}_\infty\tilde{\mathbf{A}}^\top.\end{aligned} \tag{8.33}$$

Consequently, the steady-state Kalman gain matrix[11] is

$$\mathbf{K}_\infty = \mathbf{P}_\infty\mathbf{U}_{\mathscr{F}}^\top\mathbf{C}_{\mathscr{S}}\left(\mathbf{C}_{\mathscr{S}}\mathbf{U}_{\mathscr{F}}\mathbf{P}_\infty\mathbf{U}_{\mathscr{F}}^\top\mathbf{C}_{\mathscr{S}} + \mathbf{C}_{\mathscr{S}}\mathbf{\Sigma}_v\mathbf{C}_{\mathscr{S}}\right)^\dagger, \tag{8.34}$$

with posterior state estimate

$$\tilde{\mathbf{x}}_t^+ = \left(\mathbf{I}_{|\mathscr{F}|} - \mathbf{K}_\infty\mathbf{C}_{\mathscr{S}}\mathbf{U}_{\mathscr{F}}\right)\tilde{\mathbf{A}}\tilde{\mathbf{x}}_{t-1}^+ + \mathbf{K}_\infty\mathbf{y}_t. \tag{8.35}$$

Differently from the time-varying scenario, the above steady-state KF is only asymptotically optimal. To avoid the matrix inversion in (8.34), we can rely once again on the sequential implementation [39]. From the expression of $\mathbf{K}_\infty$, a necessary condition to fully exploit the steady-state Kalman gain matrix is that the cardinality of the sampling set should satisfy $|\mathscr{S}| > |\mathscr{F}|$. That is, the graph structure, the process bandwidth, and the cardinality of the sampling set are once again tightly related to fully exploit the KF benefits in ensuring a predefined performance.

**Sampling strategy.** Similar to the previous selection strategies, the optimal sampling set that minimizes the steady-state performance for a fixed number of available nodes is found as

$$\begin{aligned}\underset{\mathbf{c}}{\text{minimize}} \quad & \text{tr}(\mathbf{P}_\infty) \\ \text{subject to} \quad & \mathbf{C}_{\mathscr{S}} = \text{diag}(\mathbf{c}), \\ & \|\mathbf{c}\|_0 = |\mathscr{S}|, \\ & \mathbf{c} \in \{0,1\}^N.\end{aligned} \tag{8.36}$$

Problem (8.36) provides the optimal solution for the sampling set that guarantees the best steady-state estimation accuracy. However, even by relaxing the non convex constraint as in (8.18) and (8.30), the impossibility of having a closed form solution for the

[11] Despite not having a closed form solution, the DARE equation (8.33) admits a numerical solution [39].

**Algorithm 8.2. : Greedy node sampling algorithm from [49] for problem** (8.36)

Start with an empty sampling set $\mathcal{S} = \emptyset$, a fixed cardinality $|\mathcal{S}|$ and counter $c = 0$
i) **FOR** $c \leq |\mathcal{S}|$
ii) **WHILE** $n \in \mathcal{S}^c$
iii) Compute $\text{tr}(\mathbf{P}_\infty(\mathcal{S} \cup \{n\}))$ in (8.33);
iv) **END FOR**
v) Select $n$ as $\text{argmin}_n \text{tr}(\mathbf{P}_\infty(\mathcal{S} \cup \{n\}))$;
vi) Update the sampling set $\mathcal{S} = \mathcal{S} \cup \{n\}$;
vii) Update the counter $c = c + 1$;
viii) **END WHILE**

DARE (8.33) renders (8.36) intractable. This result is not entirely surprizing, since the latter issue is commonly present in the sensor selection literature [48–50].

A common way to tackle problem (8.36) is by greedy algorithms [49, 51, 52]. For our specific case, we adopt the strategy from [49], where the node sampling proceeds as described in Algorithm 8.2. The sampling strategy considers starting with an empty sampling set and greedily adding the nodes that give the smallest increment in the steady-state estimation error (e.g., step v)). The solution of DARE $\mathbf{P}_\infty(\mathcal{S} \cup \{n\})$ in step iii) considers solving numerically (8.33) for $\mathcal{S} = \mathcal{S} \cup \{n\}$. Finally the algorithm stops when the desired cardinality of $\mathcal{S}$ is achieved.

For the steady-state KF on graphs (8.33)-(8.35), the greedy Algorithm 8.2 is optimal with respect to problem (8.36) if [49]:

*i*) the measurement noise $\mathbf{v}_t$ is uncorrelated;

*ii*) the set of sensor information matrices $\{\mathbf{F}_1, \ldots, \mathbf{F}_N\}$ with $\mathbf{F}_n = \sigma_v^{-2} \mathbf{u}_{\mathcal{F},n} \mathbf{u}_{\mathcal{F},n}^\mathsf{T}$ is totally ordered w.r.t. the order relation of positive semidefiniteness.

The first condition is easily met in practice. The second condition relates the graph topology and the graph signal bandwidth with the optimal sampling set. It implies that for two different sets $\mathcal{S}'$ and $\mathcal{S}''$ with $\mathbf{F}(\mathcal{S}) = \sum_{n=1}^{|\mathcal{S}|} \sigma_v^{-2} \mathbf{u}_{\mathcal{F},n} \mathbf{u}_{\mathcal{F},n}^\mathsf{T} = \mathbf{U}_\mathcal{F}^\mathsf{T} \mathbf{C}_\mathcal{S} \mathbf{\Sigma}_v^{-1} \mathbf{U}_\mathcal{F}$, if $\mathbf{F}(\mathcal{S}') \succeq \mathbf{F}(\mathcal{S}'')$ it holds that $\text{tr}(\mathbf{P}_\infty(\mathcal{S}')) \leq \text{tr}(\mathbf{P}_\infty(\mathcal{S}''))$. Thus the node sampled in the set $\mathcal{S}'$ is a better choice than the node sampled in the set $\mathcal{S}''$ [49].

As we illustrate next, the proposed KF on graph optimally tracks graph processes and outperforms other alternatives in terms of estimation accuracy.

### 8.4.3. NUMERICAL RESULTS

We now analyze the tracking performance of the KF approaches. We first consider KF for time-varying models and then focus on steady-state KF. The results are averaged over 500 different realizations.

**KF for time-varying models.** We consider tracking instantaneous graph signal diffusion on the Molene data set. The graph is a 3NN, $\mathcal{F}$ consists of the first 16 graph frequencies, and $w = 1$ in equation (8.2). The state $\mathbf{x}_0$ is initialized as zero and $\mathbf{u}_t$ for $t \in \{1, 101, \ldots, 401\}$ consists of five temperature recordings from the data set with $\mathbf{B}_t = \mathbf{I}_N$. In a nutshell, the state evolution considers the temperature diffusion for 100 iterations and then a new input is introduced. We consider a zero-mean model and measurement

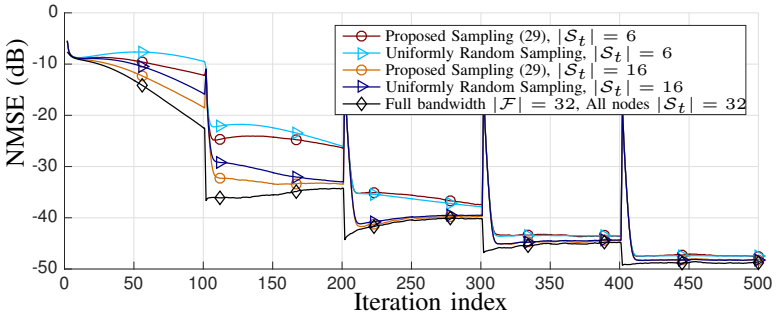



Figure 8.8: Tracking performance of KF. Estimated NMSE versus iteration index for different numbers of sampled nodes. The results are analyzed for the sampling approach (8.30), uniformly random sampling, and when all the nodes are sampled.

noises with respective covariance matrixes $\mathbf{\Sigma}_w = 10^{-4}\mathbf{I}_N$ and $\mathbf{\Sigma}_v = 10^{-1}\mathbf{I}_N$. We initialize the Kalman filter with $\tilde{\mathbf{x}}_0^+ = \mathbf{1}_{|\mathcal{F}|}$ and $\mathbf{P}_0^+ = \mathbf{\Sigma}_{\tilde{w}}$. We compare the sparse sensing approach (i.e., the opposite problem of (8.30) that selects $|\mathcal{S}|$ nodes with minimum MSE) and uniformly random sampling whose performance is averaged over 500 additional realizations.

Fig. 8.8 illustrates the tracking performance as a function of the iteration index for different values of $|\mathcal{S}_t|$. We observe that an increment of $|\mathcal{S}_t|$ leads to a smaller NMSE, especially in the first iterations. However, compared to the case of full bandwidth and $|\mathcal{S}|_t = 32$, these results show that 50% of the samples can be saved by the proposed approach with a little tradeoff on the NMSE. Further, as in [42], uniformly random sampling can be an option for tracking the process for large $t$. We additionally remark that (8.30) may not always give a sparse solution for higher $t$ and since it is an SDP relaxation, it might often lead to solutions that are far from the possible minimum MSE. Finally, note that the spikes in the estimated NMSE are related to the presence of the input signal and are common for both sampling approaches.

Next, we compare the tracking performance of KF with that of LMS [14] and RLS [17] on graphs. For the KF approach $\mathcal{S}_t$ consists of one node, sampled at random for each $t$. The RLS sampling probabilities are found with $\beta_{\text{RLS}} = 0.95$ and $\gamma_{\text{RLS}} = 7 \times 10^{-2}$ following the optimal design of [17]. The latter results in an average sampling rate of 16.08 (greater than $|\mathcal{F}| = 16$) for each $t$, with five nodes sampled with probability one. With the same sampling probabilities, the LMS step size is $\mu_{\text{LMS}} = 0.0875$ such that it meets the RLS steady-state MSE. Both algorithms are initialized as the KF.

The results of Fig. 8.9 show that the KF suffers only in the first iterations, but as the system evolution is learned better it outperforms both the LMS and RLS and, as a consequence, other state-of-the-art tracking algorithms [14, 15, 17] with which LMS and RLS compare. This result highlights the potential of the proposed approach to optimally track the signal by sampling only one node per time instant, while exploiting its dynamics.

**Steady-state KF.** We now track a heat diffusion process evolving on a binary weighted

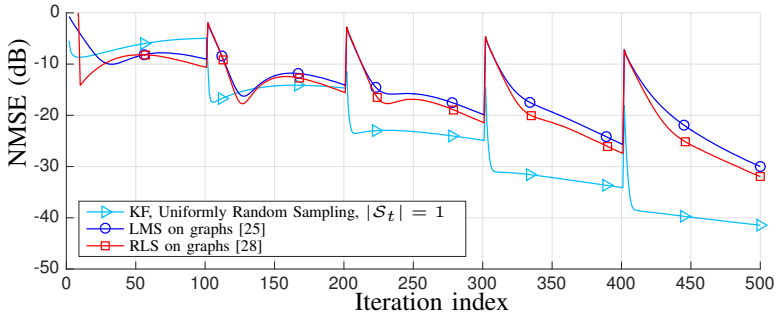



Figure 8.9: Estimated NMSE versus iteration index for KF, LMS ($\mu = 0.125$) [14] and RLS ($\beta = 0.95$) [17]. For KF, one node is sampled for each iteration, while LMS and RLS have an average sampling rate of 16.08 (greater than $|\mathcal{F}| = 16$) with five nodes sampling with probability one.

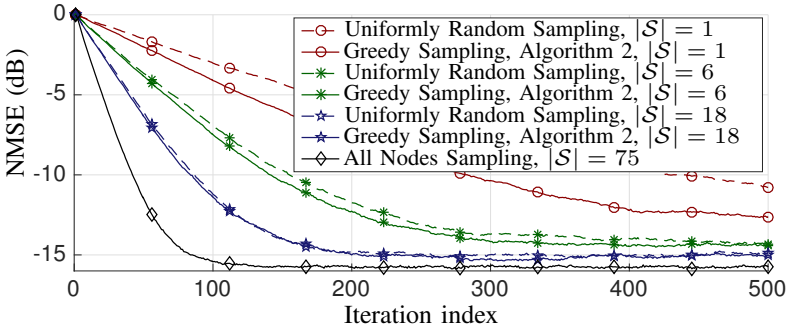



Figure 8.10: Estimated NMSE versus iteration index for steady-state KF with different numbers of sampled nodes. The results are analyzed for the sampling approach in Algorithm 8.2, uniformly random sampling, and when all the nodes are sampled.



two-dimensional rectangular grid of $N = 75$ nodes ($5 \times 15$) by making use of the steady-state KF approach. Here, we aim at providing insights into how GSP can be exploited in temperature monitoring systems. The initial signal $\mathbf{x}_0$ is set to one at the five nodes of the leftmost column of the grid and zero elsewhere. This signal is diffused following the heat propagating model (8.2) with $w = 10$ for $T = 500$ instances. $\mathcal{F}$ consists of the frequency indices where 99% of the energy of $\mathbf{x}_0$ is concentrated, resulting in $|\mathcal{F}| = 18$ active frequencies (not necessarily adjacent). The model and measurement noises have covariance matrices $\mathbf{\Sigma}_w = 10^{-4}\mathbf{I}_N$ and $\mathbf{\Sigma}_v = 10^{-1}\mathbf{I}_N$, respectively.

Fig. 8.10 shows the NMSE as a function of the diffusion time for different cardinalities of the sampling set. We observe that a larger $|\mathcal{S}|$ improves the steady-state NMSE and the convergence rate. Additionally, the sampling strategy in Algorithm 8.2 is beneficial for low values of $|\mathcal{S}|$, while the uniformly random sampling can only be adopted for larger $|\mathcal{S}|$.

Finally, we compare the tracking performance of the steady-state KF with LMS and

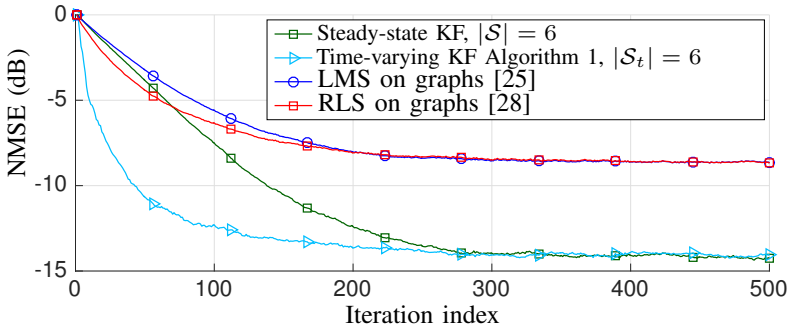



Figure 8.11: Estimated NMSE versus iteration index for steady-state KF ($|\mathcal{S}| = 6$ chosen with Algorithm 8.2), time-varying KF ($|\mathcal{S}_t| = |\mathcal{S}| = 6$ chosen randomly for each iteration), as well as RLS ($\beta_{\text{RLS}} = 0.99$) and LMS ($\mu_{\text{LMS}} = 0.041$) with maximum sampling rate $|\mathcal{S}_t| = |\mathcal{F}| = 18$.

RLS on graphs. The LMS and RLS parameters are chosen as before yielding $\mu_{\text{LMS}} = 0.041$ (LMS), $\beta_{\text{RLS}} = 0.99$ (RLS), and an overall sampling rate of 18 samples per iteration (i.e., the same as $|\mathcal{F}|$ to guarantee asymptotic MSE reconstruction) [17]. Additionally, KF with random time-varying sampling from Algorithm 8.1 is considered as a benchmark. The instantaneous sampling set $\mathcal{S}_t$ is chosen uniformly at random for each $t$ with $|\mathcal{S}_t| = |\mathcal{S}| = 6$.

The results in Fig. 8.11 show that the KF strategies outperform the adaptive algorithms in both steady-state performance and convergence speed. We also remark that the steady-state performance of both KF approaches is identical with a sampling rate that is three times lower than the other alternatives. The additional improvement in convergence speed of the time-varying KF comes at the expense of complexity, i.e., updating the Kalman gain matrix and the *a priori* and *a posteriori* error covariance matrices in each iteration.

8

## 8.5. CONCLUDING REMARKS

In this chapter, we saw how two basic and well studied concepts, such as observability and Kalman filtering can be extended to a process that evolve on top of a graph. We exploited the bandlimitedness of the time-varying graph process to perform the above tasks from few relevant nodes.

For the task of observability, we extend the graph sampling theory to a graph-time fashion and derived necessary and sufficient conditions for the observability of a graph process. In addition, two graph-time sampling schemes, namely deterministic and stochastic sampling were proposed. An MSE analysis on the observed signal is carried out, and sparse sensing-based sampling strategies are used to collect samples.

Similarly, for the KF on graphs we derived necessary conditions on the minimum number of nodes that should collect samples and conciliate these conditions with those of observability. When the system matrices are time-varying we proposed a sparse sensing sampling strategy to select, for each time instant, the minimum number of nodes

such that a desired tracking performance is guaranteed. For the cases, where the system matrices are time-invariant a steady state KF on graph approach is used. In the latter case, we showed how state-of-the-art greedy sampling can be used to carefully pick a predefined number of nodes to optimally track the steady state.

# APPENDICES

## 8.A. PROOF OF THE NECCESSARY NUMBER OF NODES REQUIRED FOR DETERMINISTIC OBSERVABILITY

By applying the rank inequality

$$\text{rank}(\mathbf{AB}) \leq \min\{\text{rank}(\mathbf{A}), \text{rank}(\mathbf{B})\} \tag{8.37}$$

to $\mathbf{O}_{0:T} = \mathbf{C}_{\mathscr{S}_{0:T}}(\mathbf{I}_{T+1} \otimes \mathbf{U}_{\mathscr{F}})\tilde{\mathbf{A}}_{0:T}$ in (8.12), we have that $\mathbf{O}_{0:T}$ can be full column rank $|\mathscr{F}|$ only if

$$\text{rank}\left(\mathbf{C}_{\mathscr{S}_{0:T}}\right) \geq |\mathscr{F}|, \tag{8.38}$$

which from the structure of $\mathbf{C}_{\mathscr{S}_{0:T}}$ is always true when the claimed conditions are satisfied. □

## 8.B. PROOF OF THE CONDITIONS FOR OBSERVABILITY THEOREM

By substituting $\mathbf{C}_{\mathscr{S}_{0:T}} = \mathbf{I}_{N(T+1)} - \mathbf{C}^{c}_{\mathscr{S}_{0:T}}$ into the rank argument of (8.15) we can write the vector form expression

$$\begin{aligned}\tilde{\mathbf{A}}^{\mathsf{H}}_{0:T}(\mathbf{I}_{T+1} \otimes \mathbf{U}^{\mathsf{H}}_{\mathscr{F}})\mathbf{C}_{\mathscr{S}_{0:T}}(\mathbf{I}_{T+1} \otimes \mathbf{U}_{\mathscr{F}})\tilde{\mathbf{A}}_{0:T} = \tilde{\mathbf{A}}^{\mathsf{H}}_{0:T}\tilde{\mathbf{A}}_{0:T} \\ - \tilde{\mathbf{A}}^{\mathsf{H}}_{0:T}(\mathbf{I}_{(T+1)} \otimes \mathbf{U}^{\mathsf{H}}_{\mathscr{F}})\mathbf{C}_{\mathscr{S}^{c}_{0:T}}(\mathbf{I}_{T+1} \otimes \mathbf{U}_{\mathscr{F}})\tilde{\mathbf{A}}_{0:T},\end{aligned} \tag{8.39}$$

which is invertible if

$$\|\tilde{\mathbf{A}}^{\mathsf{H}}_{0:T}(\mathbf{I}_{T+1} \otimes \mathbf{U}^{\mathsf{H}}_{\mathscr{F}})\mathbf{C}_{\mathscr{S}^{c}_{0:T}}(\mathbf{I}_{T+1} \otimes \mathbf{U}_{\mathscr{F}})\tilde{\mathbf{A}}_{0:T}\| < \lambda_{\min}(\tilde{\mathbf{A}}^{\mathsf{T}}_{0:T}\tilde{\mathbf{A}}_{0:T}), \tag{8.40}$$

where $\lambda_{\min}(\mathbf{A})$ is the minimum eigenvalue of $\mathbf{A}$. Here, we are exploiting that both matrices on the right-hand side of (8.39) are positive semidefinite. Then, from the Cauchy-Schwarz inequality we have

$$\begin{aligned}&\|\tilde{\mathbf{A}}^{\mathsf{H}}_{0:T}(\mathbf{I}_{T+1} \otimes \mathbf{U}^{\mathsf{H}}_{\mathscr{F}})\mathbf{C}_{\mathscr{S}^{c}_{0:T}}(\mathbf{I}_{T+1} \otimes \mathbf{U}_{\mathscr{F}})\tilde{\mathbf{A}}_{0:T}\| \leq \\ &\|(\mathbf{I}_{T+1} \otimes \mathbf{U}^{\mathsf{H}}_{\mathscr{F}})\|\|\mathbf{C}_{\mathscr{S}^{c}_{0:T}}(\mathbf{I}_{T+1} \otimes \mathbf{U}_{\mathscr{F}})\|\|\tilde{\mathbf{A}}_{0:T}\|^2 < \lambda_{\min}(\tilde{\mathbf{A}}^{\mathsf{H}}_{0:T}\tilde{\mathbf{A}}_{0:T}),\end{aligned}$$

which then leads to ($\|(\mathbf{I}_{T+1} \otimes \mathbf{U}^{\mathsf{T}}_{\mathscr{F}})\| = 1$)

$$\|\mathbf{C}_{\mathscr{S}^{c}_{0:T}}(\mathbf{I}_{T+1} \otimes \mathbf{U}_{\mathscr{F}})\| < \frac{\lambda_{\min}(\tilde{\mathbf{A}}^{\mathsf{T}}_{0:T}\tilde{\mathbf{A}}_{0:T})}{\|\tilde{\mathbf{A}}_{0:T}\|^2} = \frac{s^2_{\min}(\tilde{\mathbf{A}}_{0:T})}{s^2_{\max}(\tilde{\mathbf{A}}_{0:T})}. \tag{8.41}$$

The equality in (8.41) derives from the definition of the spectral norm and the relation between the singular and the eigenvalues of a matrix. To prove that (8.16) is a

neccessary and sufficient condition we follow similar arguments as in [11, 14]. From (8.40), $\mathbf{O}_{0:T}$ is full rank if the sufficient condition (8.16) holds. Conversely, if $\|\mathbf{C}_{\mathscr{S}^c_{0:T}}(\mathbf{I}_{T+1}\otimes\mathbf{U}_{\mathscr{F}})\| = \lambda_{\min}(\tilde{\mathbf{A}}^{\mathsf{H}}_{0:T}\tilde{\mathbf{A}}_{0:T})/\|\tilde{\mathbf{A}}_{0:T}\|^2$ for $T=0$ and thus $\tilde{\mathbf{A}}_{0:0}=\mathbf{I}_N$ we have $\|\mathbf{C}_{\mathscr{S}^c_0}(\mathbf{I}_1\otimes\mathbf{U}_{\mathscr{F}})\|=1$ which goes in contradiction with the conventional observability (recovery) of bandlimited graph signals. This proves that (8.16) is also necessary. □

## 8.C. PROOF OF THE NECCESSARY NUMBER OF NODES REQUIRED FOR STOCHASTIC OBSERVABILITY

From the structure of $\mathbf{C}_{\mathscr{S}_{0:T}}$, we have

$$\operatorname{rank}(\mathbf{C}_{\mathscr{S}_{0:T}}) \leq \operatorname{rank}(\mathbb{E}[\mathbf{C}_{\mathscr{S}_{0:T}}]) = \operatorname{rank}(\mathbf{I}_{T+1}\otimes\bar{\mathbf{C}}). \tag{8.42}$$

A necessary condition then for $\operatorname{rank}(\mathbf{C}_{\mathscr{S}_{0:T}})$ to be $|\mathscr{F}|$ is that

$$\operatorname{rank}(\mathbf{I}_{T+1}\otimes\bar{\mathbf{C}}) \geq |\mathscr{F}|. \tag{8.43}$$

From $\operatorname{rank}(\mathbf{A}\otimes\mathbf{B}) = \operatorname{rank}(\mathbf{A})\operatorname{rank}(\mathbf{B})$, (8.43) writes as

$$\operatorname{rank}(\bar{\mathbf{C}}) \geq |\mathscr{F}|/(T+1). \tag{8.44}$$

Then, since $\bar{\mathbf{C}}$ is diagonal means that at least $\lceil|\mathscr{F}|/(t+1)\rceil$ nodes must be sampled with a probability different from zero. The latter concludes the proof. □

## 8.D. PROOF OF THE RANDOM SAMPLING COROLLARY

Denote by $\mathbf{c}_t = \operatorname{diag}(\mathbf{C}_{\mathscr{S}_t})$ the random sampling vector with expectation $\bar{\mathbf{c}}$ for $t\in\{0,\dots,T\}$. Let also $d = |\mathscr{S}_{0:T}| = \sum_{t=0}^{T}\sum_{n=1}^{N} c_{t,n}$ be an auxiliary variable that characterises the cardinality of the instantaneous sampling set $\mathscr{S}_{0:T}$. Then, $d$ is a Poisson random variable being it the sum of $N(T+1)$ independent Bernoulli random variables. The claim (8.21) follows by simple statistical properties.

## 8.E. PROOF OF THE MSE PERFORMANCE FOR THE DETERMINISTIC OBSERVABILITY THEOREM

By rewriting the MSE as

$$\text{MSE} = \mathbb{E}_{\mathbf{C}}\left\{\mathbb{E}_{\mathbf{v}}\left[\operatorname{tr}\left[(\tilde{\mathbf{x}}^{\text{o}}_0-\tilde{\mathbf{x}}_0)(\tilde{\mathbf{x}}^{\text{o}}_0-\tilde{\mathbf{x}}_0)^{\mathsf{H}}\right]\right]\right\}, \tag{8.45}$$

from (8.17) we have that

$$\text{MSE} = \sigma^2_{\text{v}}\mathbb{E}_{\mathbf{C}}\left\{\operatorname{tr}\left[\left(\tilde{\mathbf{A}}^{\mathsf{H}}_{0:\text{T}}(\mathbf{I}_{\text{T}+1}\otimes\mathbf{U}_{\mathscr{F}})^{\mathsf{H}}\mathbf{C}_{\mathscr{S}_{0:\text{T}}}(\mathbf{I}_{\text{T}+1}\otimes\mathbf{U}_{\mathscr{F}})\tilde{\mathbf{A}}_{0:\text{T}}\right)^{-1}\right]\right\}. \tag{8.46}$$

Then, since the function $\varphi:\mathbf{X}\to\operatorname{tr}[\mathbf{X}^{-1}]$ is convex, we apply the Jensen inequality $\varphi(\mathbb{E}[\mathbf{X}])\leq\mathbb{E}[\varphi(\mathbf{X})]$ to lower bound (8.46) as in (8.22). □

## FURTHER READING

[1] E. Isufi, P. Banelli, P. Di Lorenzo, and G. Leus, *Observing and tracking bandlimited graph processes,* submitted to IEEE Transactions on Signal Processing (2017).

[2] E. Isufi, P. Banelli, P. Di Lorenzo, and G. Leus, *Observing bandlimited graph processes from subsampled measurements,* in *IEEE Asilomar Conference on Signals, Systems and Computations* (IEEE, 2018) pp. 405–409.

[3] C. Hu, J. Sepulcre, K. A. Johnson, G. E. Fakhri, Y. M. Lu, and Q. Li, *Matched signal detection on graphs: Theory and application to brain imaging data classification,* NeuroImage **125**, 587 (2016).

[4] C. A. Gomez-Uribe and N. Hunt, *The netflix recommender system: Algorithms, business value, and innovation,* ACM Transactions on Management Information Systems (TMIS) **6**, 13 (2016).

[5] Y. Yamanishi, J.-P. Vert, and M. Kanehisa, *Protein network inference from multiple genomic data: a supervised approach,* Bioinformatics **20**, i363 (2004).

[6] I. Pesenson, *Sampling in Paley-Wiener spaces on combinatorial graphs,* Transactions of the American Mathematical Society **360**, 5603 (2008).

[7] S. K. Narang, A. Gadde, E. Sanou, and A. Ortega, *Localized iterative methods for interpolation in graph structured data,* in *Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE* (IEEE, 2013) pp. 491–494.

[8] S. Chen, R. Varma, A. Sandryhaila, and J. Kovacevic, *Discrete signal processing on graphs: Sampling theory,* IEEE Transactions on Signal Processing **63**, 6510 (2015).

[9] X. Wang, P. Liu, and Y. Gu, *Local-set-based graph signal reconstruction,* IEEE transactions on signal processing **63**, 2432 (2015).

[10] A. G. Marques, S. Segarra, G. Leus, and A. Ribeiro, *Sampling of graph signals with successive local aggregations,* IEEE Transactions on Signal Processing **64**, 1832 (2016).

[11] M. Tsitsvero, S. Barbarossa, and P. Di Lorenzo, *Signals on graphs: Uncertainty principle and sampling,* IEEE Transactions on Signal Processing **64**, 4845 (2016).

[12] D. Romero, V. N. Ioannidis, and G. B. Giannakis, *Kernel-based reconstruction of space-time functions on dynamic graphs,* IEEE Journal of Selected Topics in Signal Processing **11**, 856 (2017).

[13] L. F. Chamon and A. Ribeiro, *Greedy sampling of graph signals,* IEEE Transactions on Signal Processing **66**, 34 (2018).

[14] P. Di Lorenzo, S. Barbarossa, P. Banelli, and S. Sardellitti, *Adaptive least mean squares estimation of graph signals,* IEEE Transactions on Signal and Information Processing over Networks **2**, 555 (2016).

[15] P. Di Lorenzo, P. Banelli, S. Barbarossa, and S. Sardellitti, *Distributed adaptive learning of graph signals,* IEEE Transactions on Signal Processing (2017).

[16] P. Di Lorenzo, E. Isufi, P. Banelli, S. Barbarossa, and G. Leus, *Distributed Recursive Least Squares Strategies for Adaptive Reconstruction of Graph Signals,* in *EURASIP European Signal Processing Conference (EUSIPCO), Kos, Greece, Aug.-Sept. 2017* (2017).

[17] P. Di Lorenzo, P. Banelli, E. Isufi, S. Barbarossa, and G. Leus, *Adaptive graph signal processing: Algorithms and optimal sampling strategies,* arXiv preprint arXiv:1709.03726 (2017).

[18] K. Qiu, X. Mao, X. Shen, X. Wang, T. Li, and Y. Gu, *Time-varying graph signal reconstruction,* IEEE Journal of Selected Topics in Signal Processing **11**, 870 (2017).

[19] Y. Xue, S. Pequito, J. R. Coelho, P. Bogdan, and G. J. Pappas, *Minimum number of sensors to ensure observability of physiological systems: A case study,* in *Communication, Control, and Computing (Allerton), 2016 54th Annual Allerton Conference on* (IEEE, 2016) pp. 1181–1188.

[20] S. Pequito, P. Bogdan, and G. J. Pappas, *Minimum number of probes for brain dynamics observability,* in *Decision and Control (CDC), 2015 IEEE 54th Annual Conference on* (IEEE, 2015) pp. 306–311.

[21] S. Pequito, F. Rego, S. Kar, A. P. Aguiar, A. Pascoal, and C. Jones, *Optimal design of observable multi-agent networks: A structural system approach,* in *Control Conference (ECC), 2014 European* (IEEE, 2014) pp. 1536–1541.

[22] J. Mei and J. M. Moura, *Signal processing on graphs: Causal modeling of unstructured data,* IEEE Transactions on Signal Processing **65**, 2077 (2017).

[23] A. Loukas, E. Isufi, and N. Perraudin, *Predicting the evolution of stationary graph signals,* in *Signals, Systems, and Computers, 2017 51st Asilomar Conference on* (IEEE, 2017) pp. 60–64.

[24] E. Isufi, A. Loukas, N. Perraudin, and G. Leus, *Forecasting time series with varma recursions on graphs,* arXiv preprint arXiv:1810.08581 (2018).

[25] A. Soule, K. Salamatian, A. Nucci, and N. Taft, *Traffic matrix tracking using kalman filters,* ACM SIGMETRICS Performance Evaluation Review **33**, 24 (2005).

[26] F. S. Cattivelli and A. H. Sayed, *Diffusion strategies for distributed kalman filtering and smoothing,* IEEE Transactions on automatic control **55**, 2069 (2010).

[27] Wikipedia, *Chernobyl disaster,* (2017).

[28] K. Nodop, R. Connolly, and F. Girardi, *The field campaigns of the european tracer experiment (etex): Overview and results,* Atmospheric Environment **32**, 4095 (1998).

[29] R. I. Kondor and J. Lafferty, *Diffusion kernels on graphs and other discrete input spaces,* in *ICML*, Vol. 2 (2002) pp. 315–322.

[30] J. C. Dittmer, *Consensus formation under bounded confidence,* Nonlinear Analysis: Theory, Methods & Applications **47**, 4615 (2001).

[31] A. Tarun and D. Van De Ville, *Extrapolating functional mri data into white matter via structurally-informed graph diffusion,* in *Organization for Human Brain Mapping (OHBM) meeting* (2018).

[32] J. Friedman and J.-P. Tillich, *Wave equations for graphs and the edge-based laplacian,* Pacific Journal of Mathematics **216**, 229 (2004).

[33] F. Grassi, A. Loukas, N. Perraudin, and B. Ricaud, *A time-vertex signal processing framework,* arXiv preprint arXiv:1705.02307 (2017).

[34] F. Zhang and E. R. Hancock, *Graph spectral image smoothing using the heat kernel,* Pattern Recognition **41**, 3328 (2008).

[35] 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 **30**, 83 (2013).

[36] J. Ma, W. Huang, S. Segarra, and A. Ribeiro, *Diffusion filtering of graph signals and its use in recommendation systems,* in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on* (IEEE, 2016) pp. 4563–4567.

[37] S. Barbarossa, S. Sardellitti, and A. Farina, *On sparse controllability of graph signals,* in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on* (IEEE, 2016) pp. 4104–4108.

[38] F. Gamma, E. Isufi, G. Leus, and A. Ribeiro, *Control of graph signals over random time-varying graphs,* in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2018).

[39] D. Simon, *Optimal state estimation: Kalman, H infinity, and nonlinear approaches* (John Wiley & Sons, 2006).

[40] S. M. Kay, *Fundamentals of statistical signal processing: Practical algorithm development*, Vol. 3 (Pearson Education, 2013).

[41] S. Joshi and S. Boyd, *Sensor selection via convex optimization,* IEEE Transactions on Signal Processing **57**, 451 (2009).

[42] S. P. Chepuri and G. Leus, *Sparse sensing for statistical inference,* Foundations and Trends® in Signal Processing **9**, 233 (2016).

[43] N. Perraudin, J. Paratte, D. Shuman, L. Martin, V. Kalofolias, P. Vandergheynst, and D. K. Hammond, *Gspbox: A toolbox for signal processing on graphs,* arXiv preprint arXiv:1408.5781 (2014).

[44] L. Zuo, R. Niu, and P. K. Varshney, *Posterior crlb based sensor selection for target tracking in sensor networks,* in *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, Vol. 2 (IEEE, 2007) pp. II–1041.

[45] P. Tichavsky, C. H. Muravchik, and A. Nehorai, *Posterior cramér-rao bounds for discrete-time nonlinear filtering,* IEEE Transactions on signal processing **46**, 1386 (1998).

[46] L. Vandenberghe and S. Boyd, *Semidefinite programming,* SIAM review **38**, 49 (1996).

[47] A. Krause, *Optimizing sensing: Theory and applications*, Ph.D. thesis, Carnegie Mellon University (2008).

[48] N. K. Dhingra, M. R. Jovanović, and Z.-Q. Luo, *An admm algorithm for optimal sensor and actuator selection,* in *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on* (IEEE, 2014) pp. 4039–4044.

[49] H. Zhang, R. Ayoub, and S. Sundaram, *Sensor selection for kalman filtering of linear dynamical systems: Complexity, limitations and greedy algorithms,* Automatica **78**, 202 (2017).

[50] V. Gupta, T. H. Chung, B. Hassibi, and R. M. Murray, *On a stochastic sensor selection algorithm with applications in sensor scheduling and sensor coverage,* Automatica **42**, 251 (2006).

[51] C. Yang, J. Wu, X. Ren, W. Yang, H. Shi, and L. Shi, *Deterministic sensor selection for centralized state estimation under limited communication resource,* IEEE transactions on signal processing **63**, 2336 (2015).

[52] V. Tzoumas, A. Jadbabaie, and G. J. Pappas, *Sensor placement for optimal kalman filtering: Fundamental limits, submodularity, and algorithms,* in *American Control Conference (ACC), 2016* (IEEE, 2016) pp. 191–196.

# V

# EPILOGUE

# 9

# CONCLUDING REMARKS AND FUTURE RESEARCH QUESTIONS

*The general was happy. Very happy. He, himself, didn't know why this sudden wave of joy hit him. This was the joy of the traveler who finds a shelter after a dangerous path in bad weather.*

Ismail Kadare,
The general of the dead army (translated from)

In this chapter, we revisit the research questions in Chapter 1 to provide the thesis concluding remarks in Section 9.1 and to pose the future research questions in Section 9.2.

## 9.1. CONCLUDING REMARKS

The definition of a Fourier transform for signals sitting on the vertices of a graph aided us with a new processing tool for such signals. Similarly to the Fourier decomposition of temporal signals, the graph Fourier transform expressed the graph signal as a linear combination of the modes of the underlying support. These modes resulted to be the eigenvectors of the graph Laplacian. This thesis built on this paradigm and provided fundamental contributions to this rising field.

In Part I, we reformulated the research path in Chapter 1 and covered the preliminary material in Chapter 2. The latter chapter included the notions of graph shift operator, graph Fourier transform, graph filtering, and stationary graph signals.

Part II focused on graph filtering as the most basic block for processing the graph signal spectrum. In specific, Chapter 3 is dedicated to the FIR graph filters while Chapter 4 to the IIR graph filters. The FIR graph filters are filters that shape the graph signal spectrum with a finite recursion in the vertex domain. We first introduced two state-the-art FIR implementations, namely the *node-invariant* and *node-variant* FIR graph filters and

showed their distributed implementation. Afterward, we introduced a more involved FIR implementation that we referred to as the *edge-variant* FIR graph filter. This filter enjoyed a distributed implementation and achieved the same approximation accuracy as the other alternatives with less communication and computational costs. Yet, a thorough theoretical analysis of its frequency response is not covered in this thesis. Then, in Chapter 4 we proposed the ARMA recursions to implement IIR graph filters. These filters achieved a rational frequency response and could be implemented distributively in the vertex domain in an IIR fashion, i.e., requiring in theory infinite iterations to converge to their designed frequency response. However these filters are characterized by a linear convergence requiring only a few exchanges between nodes.

In Part III, comprising Chapters 5, 6 and 7, we extended the filtering of graph signals to the temporal dimension. Since most of the signals of interest are time-varying, we proposed the idea to process jointly the variations over both the graph and the temporal domain. After rephrasing the joint graph-time processing in GSP terminology in Chapter 5, in Chapter 6 we performed a deterministic analysis of graph-time filtering. We showed that the ARMA graph filters naturally inherent a joint temporal processing when the graph signal is time-varying leading to the two-dimensional graph-temporal filters. Then, strategies to implement two-dimensional FIR and ARMA graph-temporal filters were introduced. In this chapter, we also characterized in closed-form the ARMA graph filter output when the graph topology is time-varying. Chapter 7 introduced a statistical analysis for graph filters when the graph topology and the graph signal change stochastically over time. We provided expressions on the first and second order moments of the filter output, which yielded different insights on the filter behavior in stochastic environments. Finally, this statistical analysis is explored to perform graph signal denoising in the mean and to implement sparse graph filters.

Part IV extended the graph-temporal analysis to observability and tracking from subsampled measurements in Chapter 8. We focused on analyzing linear state-space models on graphs from subsampled measurements. Here, we derived conditions to observe the initial realization of the graph process and we introduced the Kalman filtering on graphs to track the temporal evolution of the graph signal from few samples collected in a graph-time fashion.

In the remainder of this section, we extend our concluding remarks to the specific questions posed in Chapter 1.



### 9.1.1. ANSWER TO THE POSED RESEARCH QUESTIONS

(Q1) *How can a sensor network perform more involved distributed filtering tasks than simple averaging by considering the underlying data structure?*

We showed in Chapters 3 and 4 that it is possible through the introduced edge-variant FIR, or ARMA graph filters to approximate any desired filtering operations with recursions that enjoy a distributed implementation over the graph. The distributed edge-variant FIR, in fact, allowed us to approximate a wider class of operations which go beyond filtering. However, a graph spectral analysis of this operation is still an open question. The ARMA recursions can implement distributed graph filters with a well defined spectral response that is rational in the graph frequencies. Moreover, the ARMA filter

gave an exact solution to two important problems, namely graph signal denoising and data interpolation from missing values.

(Q2) *What are the implications of dynamic changes in the graph topology and graph signal on the graph filter output?*

We spread the answer to this questions among Chapters 6 and 7. Chapter 6 analyzed the filter behavior in a deterministic environment. Here, we provided closed-form expressions of the distributed ARMA output operating over a known time-varying graph. These results showed the influence of the graph variations on the filter output characterizing completely its behavior. Compared to the FIR graph filter, the ARMA filters were more robust to topological changes as their output performed on time-varying graphs remained closer to the filter output obtained on the time-invariant topology. We additionally found that filters of higher orders suffer more the topological changes.

An important finding of the ARMA graph filters is their natural extension to capture variations of the graph signal. When the signal on the graph is time-varying, the ARMA graph filter behaved as a two-dimensional filter processing the signal over the graph frequencies on one hand and on the temporal frequencies on the other hand. After this observation, we developed more involved graph-temporal FIR and ARMA filters. These filters could achieve two-dimensional frequency responses and shape the joint graph-temporal spectrum to the desired frequency response. We provided several implementation strategies for these filters and provided design strategies for them. These filters had a distributed implementation at the expenses of slightly increased costs w.r.t. the pure graph filtering.

In Chapter 7 we performed a statistical analysis of the graph filters' output when both the graph topology and the graph signal change randomly in time. Under the assumption that the graph and signal variations are independent, we showed that the graph filters behave in the mean as the same deterministic filter, operating on a deterministic graph being the expected graph. We derived upper-bounds on the filter output variance which characterizes the impact of the graph stochasticity on the filtering output.

(Q3) *Under which conditions of the graph topology and the graph process is the initial network state observable from a subset of vertices?*

We answered this question in Chapter 8 for a process evolving according to a defined linear model over the graph. Under the assumption that the model evolution is bandlimited w.r.t. the underlying topology, we derived necessary and sufficient conditions when the graph process is observable from subsampled measurements. We derived an MSE analysis to quantify the effects of the process graph-bandwidth on the reconstruction performance. The latter allowed us to design a sampling strategy for collecting samples in a graph-time fashion such that a predefined MSE performance on the observed signal is met.

(Q4) *Which are the conditions that the graph topology and the graph process must satisfy such that Kalman filtering can be employed to track network dynamics from a subset of nodes?*

Following on the graph process dynamics considered in (Q4), in Chapter 8 we derived necessary conditions to track bandlimited graph processes from subsampled measurements. Our main finding suggested that the number of collected measurement in each time instant must be greater than, or equal to the process bandwidth. This condition resulted useful in tracking graph-processes with time-varying models, and, thus, the sampling set had to be updated in each time instant. For time-invariant models, we considered a steady-state KF approach and selected the sampling set such that a target steady-state tracking performance was achieved.

## 9.2. FUTURE RESEARCH QUESTIONS

This thesis introduced new research directions that fall under the umbrella of graph signal processing. As such, several new research questions arise. In the sequel, we conclude the thesis with some recommendations for future research.

### 9.2.1. GRAPH FILTERING

In Chapters 3-4, we focused on different distributed strategies to perform graph filtering. Our first open question regards the extension of the edge-variant philosophy to an ARMA implementation. In specific:

(FQ1) *How to implement and design node- and edge-variant ARMA graph filters?*

From the parallelism between the $\text{ARMA}_1$ and the $\text{FIR}_K$ graph filter in Chapter 4, a node-variant $\text{ARMA}_1$ graph filter can be implemented as

$$\mathbf{y}_{t+1} = \text{diag}(\boldsymbol{\psi}_{\text{nv}})\mathbf{S}\mathbf{y}_t + \text{diag}(\boldsymbol{\varphi}_{\text{nv}})\mathbf{x}, \tag{9.1}$$

with $\boldsymbol{\psi}_{\text{nv}} = [\psi_1, \ldots, \psi_N]^{\mathsf{T}}$ and $\boldsymbol{\varphi}_{\text{nv}} = [\varphi_1, \ldots, \varphi_N]$ being the node-varying coefficients. While the filtering output and the convergence properties of (9.2) can be derived from Theorem 4.1, the more challenging aspect of the NV-$\text{ARMA}_1$ is the filter design. A higher order NV-ARMA filter can be obtained by either extending (9.2) with a parallel bank as in (4.7), or with a feedback implementation as in (4.26).

By extending (9.2), an EV-$\text{ARMA}_1$ recursion has the form

$$\mathbf{y}_{t+1} = (\boldsymbol{\Psi}_{\text{ev}} \odot \mathbf{S})\,\mathbf{y}_t + \text{diag}(\boldsymbol{\varphi}_{\text{ev}})\mathbf{x}, \tag{9.2}$$

where the coefficients matrix $\boldsymbol{\Psi}_{\text{ev}}$ shares the support with $\mathbf{S}$ and $\boldsymbol{\varphi}_{\text{ev}}$ is a vector of coefficients. Compared to the FIR counterparts, we believe the node/edge-variant ARMA filters might lead to a more robust implementation in dynamic environments (similarly to the results in Chapters 6 and 7 for the node-invariant case). Further, by having more DoFs w.r.t. the FIR counterpart, these filters can implement distributively a larger set of operations on graphs. Check the follow-up work [1] for some initial answers to this question.

Within the context of edge-variant filtering on graphs, we pose our second question:

(FQ2) *How do graph filter banks behave with an edge-variant implementation?*

A thorough answer to the above question will shed light on the potential of the edge-variant filters to improve the performance of graph filter banks [2–4]. The perfect recovery required by graph filter banks is often affected by the limited DOFs that FIR graph filters have. Therefore, an edge-variant graph filter bank can be a suitable choice to tackle this issue.

Our next research question regards the recursive implementation of ARMA graph filters.

(FQ3) *What other recursions implement distributively an ARMA graph filter?*

Following the general implementation (4.3), the above question asks if the recursion

$$[\mathbf{y}_t]_i = \sum_{j\in\mathcal{N}(i,P)} a_{i,j}^{(1)}[\mathbf{y}_{t-1}]_j + \sum_{j\in\mathcal{N}(i,P)} a_{i,j}^{(2)}[\mathbf{y}_{t-2}]_j + \ldots + \sum_{j\in\mathcal{N}(i,P)} a_{i,j}^{(P)}[\mathbf{y}_{t-P}]_j + \sum_{j\in\mathcal{N}(i,Q)} b_{i,j}x_j + b_{i,i}x_i, \tag{9.3}$$

can be a choice for implementing ARMA graph filers. Though (9.3) is an IIR recursion on graphs, it is unclear what frequency response it achieves. The relation between (9.3) and the ARMA$_1$ (4.4), ARMA$_K$ (4.7), and ARMA$_{P,Q}$ (4.26) is another interesting direction to take.

Our last research suggestion on graph filters regards the effects of quantization in sensor networks.

(FQ4) *How to use subtractive dithering for ameliorating the effects of data quantization in distributed graph filters?*

The graph filter analysis carried in this thesis has not dealt with finite precision effect when filtering a signal. However, in distributed sensor networks the transmitted data are often quantized with few bits. To cope well with these quantization effects, the graph filters should be designed to account for the finite precision. A first attempt is considered for the FIR filter in [5, 6]. We believe more benefits might be achieved by introducing the subtractive dithering idea [7] into the filter recursions. The latter has shown benefits in distributed signal processing algorithms [8] and can be applied to ARMA filters. Being the distributed ARMA graph filters an iterative algorithm, it will require only a few bits for the successive iterations yielding a low implementation cost.



### 9.2.2. DETERMINISTIC GRAPH-TIME FILTERING

The focus of Chapter 6 was on analyzing the filtering output when the graph topology and(or) the graph signal change(s) deterministically over time. In the following, we pose three potential future directions.

(FQ5) *How graph filters behave when the graph topology changes over time with a specific model?*

This questions follows the analysis carried out in Section 6.2.2 by asking a detailed frequency analysis for the graph filters when the underlying topology changes over time following a pre-defined model. An example of the latter is a fixed sensor movement pattern (e.g., the graph changes cyclically from $\mathcal{G}_1$ to $\mathcal{G}_\tau$).

(FQ6) *How to account for the effects of asynchronous communications in designing graph filters?*

Along with the signal quantization, asynchronous communication is another challenge to face in distributed sensor networks. Some initial results were carried out for the potential kernel in [9]. However, a filter design strategy that accounts for the amount of asynchrony in the distributed implementation is still unsolved. A filter design performed in this way will lead to more robust and practical filters. If this effect is ignored, serious repercussions might be present on the filter output.

(FQ7) *How to design the more general graph-temporal filters?*

In Section 6.3, we introduced the two-dimensional FIR and ARMA graph filters with the aim to process time-varying graph signals. Though the design problem and some preliminary design approaches were presented, it was not conducted a thorough analysis for the more general form. As a future recommendation, we propose a filter design strategy for a complete two-dimensional graph-time filter to approximatet an arbitrary two-dimensional frequency mask.

### 9.2.3. STATISTICAL GRAPH-TIME FILTERING

In Chapter 7, we introduced a new perspective on graph filtering that considered the stochasticity in the topology and in the graph signal. Even though this is an unexplored area with many open questions, in the sequel we pose two potential research directions.

(FQ8) *What are the statistical properties of the filter output when the graph topology and the signal are correlated with each-other?*



This question extends the filter statistical analysis performed in this thesis and we believe it is an important, yet challenging, direction to take. It will address several real-world phenomena, such as airline delay diffusion affected by random flight cancelations.

(FQ9) *What is the sparsest graph filter that guarantees a target output performance?*

In Section 7.5, we proposed a way to implement distributed graph filtering in a sparsified way. Yet, we did not present any strategy to impose a desired sparsity level. Within the framework of node/edge-variant graph filters, further savings can be achieved by selecting the sparsest communication probabilities that guarantee a given variance on the filtering output.

### 9.2.4. OBSERVING AND TRACKING GRAPH PROCESSES

In Chapter 8, we derived conditions for observing and tracking a time-varying graph process from few noisy measurements. Then, our next research question conciliates this theory with the statistical analysis of graph filtering.

(FQ10) *Which are the conditions to observe and track a graph signal on a random time-varying graph?*

As the graph topology is random over time (e.g., following the RES model of Chapter 7), we can exploit its statistical moments to formulate a statistical observability theory where the estimated realization will be a random variable. Then, by characterizing the first and second order moments of the observed signal, we can draw conclusions without knowing the specific graph realizations.

### 9.2.5. GENERAL GRAPH SIGNAL PROCESSING

Besides the above questions, which yielded from the thesis arguments, there is a number of other research question to be considered for future works. In the sequel, we list a couple of more high-level questions that might be of the reader interest.

(FQ11) *How signal value decomposition can be exploited in graph signal processing?*

Singular value decomposition (SVD) is a handle tool in classical signal processing. However, this is not yet the case for signals on graphs. As many GSP tools extend intuitively from the structured temporal and spatial domains, we feel that SVD has the potential to be useful in GSP as well. This might be a valid surrogate of the numerical unstable Jordan decomposition in directed graphs, but a series of properties need to be addressed. If we follow the developed GSP theory, these properties include the notion of frequency in the graph setting through SVD, for instance. However, if we stay at a broader level, it might be interesting to analyze the usefulness of SVD in improving GSP tasks such as sampling with no particular graph-graph frequency dualism.

(FQ12) *What is a local stationary graph signal?*

The analysis of stationary for graph signals is an interesting direction to consider as it gives rise to several benefits among which the Wiener graph filter. However, so far, real data do not present a high degree of WSS [10, 11]. The latter leaded us with the intuition that graph-based data might present local WSS rather than a global one. An example of the latter is a two connected opinion communities graph, where the first and second order moments of the signal satisfy the WSS on graphs conditions in each community separately but not on the full graph. In that case, it is reasonable to perform the Wiener filtering locally only on each community. It is then necessary to define the notion of local stationary for graph signals and to provide a tool for quantifying it.

We finally remark that the treated arguments along with the future research suggestions are only a small slice of the GSP area. We hope that this thesis triggered the reader interest to provide innovative solutions and novel research directions within this context.

## FURTHER READING

[1] M. Coutino, E. Isufi, and G. Leus, *Advances in distributed graph filtering,* arXiv preprint arXiv:1808.03004 (2018).

[2] S. K. Narang and A. Ortega, *Perfect reconstruction two-channel wavelet filter banks for graph structured data,* IEEE Transactions on Signal Processing **60**, 2786 (2012).

[3] Y. Tanaka and A. Sakiyama, *m-channel oversampled graph filter banks,* IEEE Transactions on Signal Processing **62**, 3578 (2014).

[4] D. B. Tay and Z. Lin, *Design of near orthogonal graph filter banks,* IEEE Signal Processing Letters **22**, 701 (2015).

[5] L. F. Chamon and A. Ribeiro, *Finite-precision effects on graph filters,* in *5th IEEE Global Conference on Signal and Information Processing (GlobalSIP)* (2917).

[6] N. Thanou, *Graph signal processing: Sparse representation and applications*, Ph.D. thesis, Ecole Polytechnique Fédérale de Lausanne (2016).

[7] R. M. Gray and T. G. Stockham, *Dithered quantizers,* IEEE Transactions on Information Theory **39**, 805 (1993).

[8] S. Zhu and B. Chen, *Quantized consensus by the admm: probabilistic versus deterministic quantizers,* IEEE Transactions on Signal Processing **64**, 1700 (2016).

[9] A. Loukas, *Distributed graph filters,* (2015).

[10] N. Perraudin and P. Vandergheynst, *Stationary signal processing on graphs,* IEEE Transactions on Signal Processing **65**, 3462 (2017).

[11] A. G. Marques, S. Segarra, G. Leus, and A. Ribeiro, *Stationary graph processes and spectral estimation,* IEEE Transactions on Signal Processing (2017).

# List of Abbreviations

| | |
|---|---|
| ARMA | AutoRegressive Moving Average |
| C-EV | Constrained Edge Variant |
| CRB | Cramér-Rao Bound |
| DAD | Disjoint Average and Denoise |
| DARE | Discrete Algebric Riccatti Equation |
| DFT | Discrete Fourier Transform |
| DoFs | Degrees of Freedom |
| EV | Edge Variant |
| FIM | Fisher Information Matrix |
| FIR | Finite Impulse Response |
| GFT | Graph Fourier Transform |
| GSP | Graph Signal Processing |
| GPSD | Graph Power Spectral Density |
| GWSS | Graph Wide Sense Stationary |
| HP | High Pass |
| IIR | Infinite Impulse Response |
| JDMIA | Joint Denoising in the Mean with Input Averaging |
| JDMIOA | Joint Denoising in the Mean with Input-Output Averaging |
| JDMOA | Joint Denoising in the Mean with Output Averaging |
| JFT | Joint Fourier Transform |
| KF | Kalman Filtering |
| LMS | Least Mean Squares |
| LP | Low Pass |
| MSE | Mean Square Error |
| NMSE | Normalized Mean Square Error |
| NV | NodeVariant |
| pdf | probabilitydensity function |
| PCRB | PosteriorCramér-Rao Bound |
| RES | RandomEdge Sampling |
| RLS | RecursiveLeast Squares |
| ROC | RegionOf Convergence |
| SDP | Semi-Definite-Programming |
| SIS | Susceptible-Infected-Susceptible |
| sKF | sequential Kalman Filtering |
| SVD | Singular Value Decomposition |

# NOTATION

| | |
|---|---|
| $a$ | scalar variable |
| $\mathbf{a}$ | vector variable |
| $\mathbf{A}$ | matrix variable |
| $a_i$, $[\mathbf{a}]_i$ | indicate the $i$th element of $\mathbf{a}$ |
| $A_{i,j}$, $[\mathbf{A}]_{i,j}$ | indicate the $(i,j)$th element of $\mathbf{A}$ |
| $\mathbf{0}_N$ | the $N \times 1$ vector of all zeros |
| $\mathbf{1}_N$ | the $N \times 1$ vector of all ones |
| $\mathbf{I}_N$ | $N \times N$ identity matrix |
| $\mathbf{e}_i$ | $N \times 1$ canonical vector |
| $\mathscr{S}$ | set variable |
| $\mathbf{1}_{\mathscr{S}}$ | set vector operator, $[\mathbf{1}_{\mathscr{S}}]_s = 1$ if $s \in \mathscr{S}$, and zero otherwise |

**Mathematical Operations**

| | |
|---|---|
| $\mathbf{A}^*$ | element-wise conjugate of $\mathbf{A}$ |
| $\mathbf{A}^\mathsf{T}$ | transpose of $\mathbf{A}$ |
| $\mathbf{A}^\mathsf{H}$ | Hermitian (conjugate transpose) of $\mathbf{A}$ |
| $\mathbf{A}^\dagger$ | Moore-Penrose pseudo-inverse of $\mathbf{A}$ |
| $\text{rank}(\mathbf{A})$ | rank of the matrix $\mathbf{A}$ |
| $\|\mathbf{a}\|_p$ | p-norm of vector $\mathbf{a}$ |
| $\|\mathbf{A}\|$ | spectral norm of matrix $\mathbf{A}$ |
| $\mathbf{A} = \text{diag}(\mathbf{a})$ | builds a diagonal matrix $\mathbf{A}$ with $\mathbf{a}$ in the main diagonal |
| $\mathbf{a} = \text{diag}(\mathbf{A})$ | stores the main diagonal of $\mathbf{A}$ in $\mathbf{a}$ |
| $\mathbf{a} = \text{vec}(\mathbf{A})$ | stores the columns of matrix $\mathbf{A}$ in the vector $\mathbf{a}$ |
| $\mathbf{A} = \text{vec}^{-1}(\mathbf{a})$ | the inverse vectorized operator |
| $\text{nnz}(\mathbf{A})$ | counts the number of non-zero elements of $\mathbf{A}$ |
| $\mathbb{E}(\mathbf{a})$ | expected value of the random vector $\mathbf{a}$ |
| $\mathbf{A} \odot \mathbf{B}$ | Hadamard, element-wise product to matrices $\mathbf{A}$ and $\mathbf{B}$ |
| $\overset{c}{\times}$ | Cartesian product |
| $\|\mathscr{S}\|$ | cardinality of set $\mathscr{S}$ |
| $\lceil \cdot \rceil$ | ceiling operator |

# SUMMARY

The necessity to process signals living in non-Euclidean domains, such as signals defined on the top of a graph, has led to the extension of signal processing techniques to the graph setting. Among different approaches, graph signal processing distinguishes itself by providing a Fourier analysis of these signals. Analogously to the Fourier transform for time and image signals, the graph Fourier transform decomposes the graph signals in terms of the harmonics provided by the underlying topology. For instance, a graph signal characterized by a slow variation between adjacent nodes has a low frequency content.

Along with the graph Fourier transform, graph filters are the key tool to alter the graph frequency content of a graph signal. This thesis focuses on graph filters that are performed distributively in the node domain–that is, each node needs to exchange information only within its neighbor to perform a given filtering operation. Similarly to the classical filters, we propose ways to design and implement distributed finite impulse response and infinite impulse response graph filters.

One of the key contributions of this thesis is to bring the temporal dimension to graph signal processing and build upon a graph-time signal processing framework. This is done in different ways. First, we analyze the effects that the temporal variations on the graph signal and graph topology have on the filtering output. Second, we introduce the notion of joint graph-time filtering. Third, we present a statistical analysis of the distributed graph filtering when the graph signal and the graph topology change randomly in time. Finally, we extend the sampling framework from the reconstruction of graph signals to the observation and tracking of time-varying graph processes.

We characterize the behavior of the distributed autoregressive moving average (ARMA) graph filters when the graph signal and the graph topology are time-varying. The latter analysis is exploited in two ways: $i$) to quantify the limitations of graph filters in a dynamic environment, such as a moving sensors processing a time-varying signal in a sensor network; and $ii$) to provide ways for filtering with low computation and communication complexity time-varying graph signals.

We develop the notion of distributed graph-time filtering, which is an operation that jointly processes the graph frequencies of a time-varying graph signal on one hand and its temporal frequencies on the other hand. We propose distributed finite impulse response and infinite impulse response recursions to implement a two-dimensional graph-time filtering operation. Finally, we propose design strategies to find the filter coefficients that approximate a desired two-dimensional frequency response.

We extend the analysis of graph filters to a stochastic environment, i.e., when the graph topology and the graph signal change randomly over time. By characterizing the first and second order moments of the filter output, we quantify the impact of the graph signal and the graph topology randomness into the distributed filtering operation. The latter allows us to develop the notion of graph filtering in the mean, which is also used to ease the computational burden of classical graph filters.

Finally, we propose a sampling framework for time-varying graph signals. Particularly, when the graph signal changes over time following a state-space model, we extend the graph signal sampling theory to the tasks of observing and tracking the time-varying graph signal from a few relevant nodes. The latter theory considers the graph signal sampling as a particular case and shows that tools from sparse sensing and sensor selection can be used for sampling.

# SAMENVATTING

De noodzaak om signalen in niet-Euclidische domeinen, zoals grafen, te kunnen verwerken heeft geleid tot een uitbreiding van de klassieke signaalverwerking naar grafen. Verschillende technieken zijn hiervoor ontwikkeld, maar de zogenaamde graaf signaalverwerking is interessant omdat het een Fourier analyse toelaat van deze graaf signalen. Zoals de klassieke Fourier transformatie voor audio en beelden ontbindt de graaf Fourier transformatie een graaf signaal in verschillende harmonische componenten die gerelateerd zijn aan de onderliggende graaf. Een graaf signaal dat slechts een kleine variatie vertoont tussen aangrenzende knopen heeft bijvoorbeeld een lage frequentie-inhoud.

Samen met de graaf Fourier transformatie spelen graaf filters een belangrijke rol om de frequentie-inhoud van graaf signalen te veranderen. Deze thesis richt zich op graaf filters waarvan de implementatie kan gedistribueerd worden over de knopen van de graaf ? dit wil zeggen dat iedere knoop enkel informatie uitwisselt met zijn buren om een filteroperatie uit te voeren. Meer specifiek stellen wij verschillende technieken voor om zowel eindige als oneindige impulsresponsie filters te ontwerpen en gedistribueerd te implementeren.

Een van de sleutelbijdragen van deze thesis is het toevoegen van de tijdsdimensie aan graaf signaalverwerking en het opbouwen van een graaf-tijd signaalverwerkingsomgeving. Dit wordt gedaan op verschillende manieren. Eerst analyseren we het effect van tijdsvariaties in het signaal en de graaf op de filteruitgang. Daarna ontwikkelen we het begrip graaf-tijd filter. Ten derde presenteren we een statistische analyse van de gedistribueerde filteroperatie wanneer zowel het signaal als de graaf op een arbitraire manier variëren in de tijd. En tenslotte breiden we de bemonstering en reconstructie van laagfrequente graaf signalen uit naar de observatie en het volgen van graaf-tijd signalen.

# ACKNOWLEDGEMENTS

It amazes me how many things have changed in my personal and professional life in just four years– and how fast all this happened. Undoubtedly, this has been my most interesting, challenging, and inspiring experience. Such experience would have not been the same without the contribution of the following people to whom I want to express my gratitude.

First, I would like to thank my supervisor Geert Leus for his mentorship during this journey. Geert, thank you for believing in me in the first place, for your guidance, advice, and feedback. If today I am a better researcher than four years ago, a big part of this merit is yours and for this, I will always be thankful. Geert was more a colleague and friend figure to me and this rendered his supervision great. I always enjoyed our talks about the Giro and Tour, the beers (always Belgian), and the cycling routes. You even had the patience to teach me how to climb the Ardennes, a challenge that resulted to be tougher than writing a paper. These are only a few of the great memories that made this experience unique.

I am thankful to two friends, Andrea Simonetto and Andreas Loukas, who had the patience to advise me during the first steps of this journey. Andrea, thank you for being an excellent teacher, colleague, and friend. Thanks also for the great memories in Mexico, particularly the one related to the food in Ticul :). Andreas, thanks for all our discussions, brainstormings, and for wisely listening to all my crazy ideas about graph-time signal processing. Your Ph. D. defense, almost four years ago, was the starting point of my research in graph signal processing and I am happy you participated in it.

A special thank goes to all my other collaborators during these four years I had the pleasure to work with and solve challenging problems. Particularly, I would like to mention *Mario Coutino, Paolo Di Lorenzo, Fernando Gamma, Jiani Liu*, and *Alejandro Ribeiro* with whom relation went more than a joint work. Thank you for your feedback, thoughts, ideas, and advice. They all help me improve both professionally and personally and I hope our collaboration will continue in the near future with more interesting problems.

In addition to the above list, my sincere gratitude goes to *Paolo Banelli*, who, in the last nine years, has spanned different roles in my career. Thanks Paolo for bringing me into the world of signal processing back in the bachelor's time. You have always been a reference point in my undergraduate studies and a great colleague and supervisor in the recent years.

It is funny how a random place assignment can transform the everyday office life from boring to an adventure. I was lucky I ended up in the biggest and noisiest office in the CAS group, where thanks to my terrific officemates *Jamal Amini, Andreas Koutrouvelis, Andrea Pizzo, Aydin Rajabzadeh, Thomas Sherson, Pim van der Meulen, Wangyang Yu, Jie Zhang,* and *Jörn Zimmerling* every day was special. Thank you guys for all our discussions spanning from the imaginary Elvin's cryptocurrency to matrix SVDs, for the chess matches, Friday cakes, and for often being a better research engine than Google.

I wish to thank all the past and current people in the CAS group I connected with even with a simple good morning. An exhaustive list would require a thesis apart. Symbolically, I would like to mention the group head Alle-Jan van der Veen for being always available to listen about any topic, and Minaksie Ramsoekh and Irma Zomerdijk for their help with the paperwork.

Mom and dad, thanks! Thanks for making me the person I am. Thanks for your unconditional love and thanks for your wise words. The Ph.D. degree is the highest academic degree that one can aspire and it requires years of sacrifice get it. This thesis could not have been possible without your sacrifice and it cannot be different than being dedicated to you.

Qendresa, these four years have been a challenge for you more than they have been for me. Thank you for your patience, your love, and for being always present. Thanks also for the great moments in this rainy, yet romantic place.

*Elvin Isufi*
*Delft, September 2018*

# BIOGRAPHY

Elvin Isufi was born in Vlore, Albania, in 1989. He received his B. Sc. degree in 2012 with full marks and his M. Sc. degree in 2014 *cum laude* both from University of Perugia, Italy, under the supervision of Prof. Paolo Banelli. In 2014, he was granted by the Erasmus Placement program for mobility exchange to develop his master thesis at the Circuits and Systems group at TU Delft. Subsequently, in Nov. 2014 he started his Ph. D. degree in the same group under the supervision of Prof. Geert Leus. He received the Student of Excellence Award from University of Perugia in 2013, the Erasmus+ mobility grant in 2016, and in 2017 his coauthored paper got the Best Student Paper in the IEEE CAMSAP workshop. His research interests lie in the intersection of signal and data processing, mathematical modeling, machine learning, and network theory.