Signal Processing and Optimization on Graphs

Learning Time-Varying Structures and Generalizing Convolution Principles

SIGNAL PROCESSING AND OPTIMIZATION ON GRAPHS

LEARNING TIME-VARYING STRUCTURES AND GENERALIZING CONVOLUTION PRINCIPLES

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus, prof. dr. ir. T.H.J.J. van der Hagen,
chair of the Board for Doctorates
to be defended publicly on
Monday 7, October 2024 at 15:00 o'clock

by

Alberto NATALI

Dottore Magistrale in Computer Engineering and Robotics,
University of Perugia, Italy
born in Umbertide, Italy

Printed by: Gildeprint

Cover by: Wendy Bour-van Telgen

Peace

CONTENTS

SUMMARY

Graph signal processing is a field that focuses on extracting valuable information from data collected in networks, such as social, transportation, and brain networks. This doctoral thesis makes significant contributions to two important aspects of graph signal processing: network topology identification and the convolution theorem.

The thesis begins by introducing the fundamental concepts of signal processing, such as shifting, convolution, and filtering, in the discrete-time domain. It then extends these concepts to a graph-based context, where classical discrete-time signal processing can be seen as a special case of graph signal processing.

The thesis then presents novel theories and algorithms in three main areas. Specifically:

(1) **Graph Topology and Filter Estimation**: It proposes an algorithmic approach to jointly learn the graph structure and filter coefficients from input-output graph-based data. The method addresses the non-convexity of the problem using an alternating-minimization scheme, ensuring global convergence.

(2) **Time-Varying Graph Topology Inference**: A new framework based on time-varying convex optimization tools is introduced for inferring time-varying network structures from graph-based data. This framework offers flexibility to users in balancing execution speed and algorithm accuracy through tunable parameters.

(3) **Generalizing the Convolution Theorem**: A generalization of the convolution theorem that encompasses both the graph convolution theorem and the one related to time-varying filters is introduced. This generalization has implications for (non-)stationarity and spectral analysis of signals and enables the casting of a graph learning problem to infer a potential graph structure for the frequency domain.

In summary, this thesis significantly contributes to advancing graph signal processing by addressing fundamental challenges and introducing innovative methodologies. It holds the potential to inspire further innovation in the field and deepen our understanding of complex network dynamics.

SAMENVATTING

Grafische signaalverwerking is een vakgebied dat zich richt op het extraheren van waardevolle informatie uit gegevens verzameld in netwerken, zoals sociale, transport- en hersennetwerken. Dit doctoraatsproefschrift levert aanzienlijke bijdragen aan twee belangrijke aspecten van de grafische signaalverwerking: identificatie van netwerktopologie en het convolutietheorema.

Het proefschrift begint met het introduceren van de fundamentele concepten van signaalverwerking, zoals verschuiving, convolutie en filtering, in het discrete-tijd domein. Vervolgens worden deze concepten uitgebreid naar een grafisch context, waarbij klassieke discrete-tijd signaalverwerking kan worden gezien als een speciaal geval van grafische signaalverwerking.

Vervolgens presenteert het proefschrift nieuwe theorieën en algoritmen op drie belangrijke gebieden. Specifiek:

(1) **Identificatie van grafiek-topologie en filterestimatie**: Er wordt een algoritmische aanpak voorgesteld om gezamenlijk de grafiekstructuur en filtercoëfficiënten te leren van input-output grafiekgebaseerde gegevens. De methode behandelt de niet-convexiteit van het probleem met een afwisselend minimalisatieschema, waarbij wereldwijde convergentie wordt gegarandeerd.

(2) **Inferentie van tijdvariërende grafiek-topologie**: Een nieuw kader gebaseerd op tijdvariërende convexe optimalisatietools wordt geïntroduceerd voor het afleiden van tijdvariërende netwerkstructuren uit grafiekgebaseerde gegevens. Dit kader biedt gebruikers flexibiliteit in het balanceren van uitvoeringssnelheid en algoritmenauwkeurigheid door middel van instelbare parameters.

(3) **Algemene formulering van het convolutietheorema**: Er wordt een algemene formulering van het convolutietheorema geïntroduceerd die zowel het grafiekconvolutietheorema als dat van de tijdvariërende filters omvat. Deze generalisatie heeft implicaties voor de (niet-)stationariteit en spectrale analyse van signalen en maakt het mogelijk om een grafiek-leerprobleem te casten om een potentiële grafiekstructuur voor het frequentiedomein af te leiden.

Samengevat draagt dit proefschrift aanzienlijk bij aan de vooruitgang van de grafische signaalverwerking door fundamentele uitdagingen aan te pakken en innovatieve methodologieën te introduceren. Het heeft het potentieel om verdere innovatie in het vakgebied te inspireren en ons begrip van complexe netwerkdynamiek te verdiepen.

PREFACE

And so the journey has come to an end.

It has been an honor and a privilege to purse a doctoral path in this lovely city, within a great university. The environment needs to nurture you, and environment encompasses also the people around you. I will not list all the possible people which contributed in one way or another to support this journey. If you find yourself seeking your name within these pages, this means you have in one way or another—thank you! If you do not know me personally, I am glad you are reading this thesis. However, there are a few to whom I owe special thanks.

Thank you, my friends, for being there no matter the circumstances. I hope you will give me water; I will do the same for you.

Thank you, my PhD fellows, for our time spent together in and out the university. Over these four years, I've undoubtedly grown and changed, in part due to our interactions. I hope I've been able to transmit something to you as well.

Geert, I will always be grateful to you for giving me this magnificent opportunity and for believing in me even when you barely knew me. Your guidance has been invaluable, and over the years, you have enabled me to learn and grow. I hope that I have also been able to leave you something, and that in the future, our paths can still be intertwined.

Elvin, thank you for making my Delft onboarding very smooth and for providing me a warm start in my academic path. You acted as a senior brother to me, and I am very grateful for that.

Mario, thank you for your technical guidance in my first years, and for our time outside the university. You have been a source of inspiration and motivation to me.

Prof. Banelli, thank you for introducing me to graph signal processing which eventually led me to Geert. Networks are important not only on papers.

Finally, to my parents, thank you for giving me the freedom of choosing my (place in) life. This is not a given: the chains of culture and traditions are often strong.

Alberto Natali
Delft, April 2024

1

HUMANS AND KNOWLEDGE

A nice book, without a nice introduction, is not a nice book.

A. Natali

Since our earliest years, we are fascinated by the mysteries of the world both within and beyond us. The questions "why" and "how" are a constant of childhood, echoing through the ages as we seek to unravel the complexities of our existence. Our intrinsic drive for exploration compels us to ponder profound philosophical inquiries, such as the origin of the universe and the fundamental principles governing life itself. Ancient civilizations, driven by practical concerns such as agricultural needs, delved into the realms of cosmology and sought partial answers and patterns in these existential questions.

However, it was during the scientific revolution that a profound shift took place. Mathematics, already firmly established as a fundamental discipline, revealed its unparalleled versatility as a language capable of encapsulating and encoding the essence of our discoveries. Not only, it also reveled its capability of creating new knowledge in a mathematical form which could then be translated back into the real physical world. It is indeed during the 16th and 17th centuries that figures like Galileo Galilei, Johannes Kepler, and Isaac Newton, made of matemathics a predictive tool for a better understanding of our world and unlocking new knowledge. The scientific revolution has begun. This pivotal moment in the annals of science was eloquently captured by the Armenian-American science author Aram D'Abro in his seminal work 'The Evolution of Scientific Knowledge':

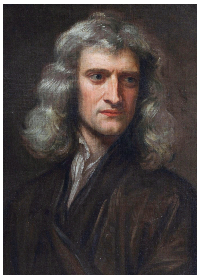


Figure 1.1: Sir Isaac Newton

"But with Newton all the resources of mathematics were turned to advantage in the solution of physical problems. Thenceforth mathematics appeared as an instrument of discovery, the most powerful one known to man, multiplying the power of thought just as in the mechanical domain

the lever multiplied our physical action. [...] Thus problems of physics were metamorphosed into problems of mathematics."

He then continues:

"The mathematical physicist then enters upon the scene, assigns certain letters of the alphabet to the physical entities involved (in the present case electric current designated by i and magnetic intensity designated by H) and by this means translates the numerical relationships discovered by the experimenter into mathematical form. He thus obtains a mathematical relationship or equation α which is assumed to constitute the mathematical image of the concrete physical phenomenon A. His task will now be to extract from his mathematical equation or equations α all their necessary mathematical consequences. In this way, provided his technique does not fail him, he may be led to new equations β. These new equations β, when translated back from the mathematical to the physical, will express new physical relationships B. The mathematician assumes that just as his equations α were the necessary mathematical consequences of his original equations, so also must the physical translation of β constitute a physical phenomenon B, which follows as a necessary consequence of the existence of the physical phenomenon A. If A occurs, B must ensue."

This merits emphasis. It became clear that by identifying the entities involved in a phenomenon of interest, we could construct mathematical models to depict the relationships between them and validate our theories through successive, rigorous experimentation. The hallmark of the scientific method. This breakthrough was made possible by the adoption of new measurement instruments, a crucial element that was notably absent in ancient times.

1.1. Data, Networks and Signal Processing

1.1.1. Data

And so, as mathematics evolved and matured day by day, we found ourselves in need of the essential ingredients: *data*. And they arrived, in abundance. Suddenly, everything we did was meticulously recorded and measured. But measuring, from a scientific point of view, is obviously not without an end goal. It serves the grand scope of deciphering patterns, unraveling relationships, and unearthing hidden associations that lie beneath the surface. These are tasks that the human mind does constantly and unconsciously, but as technology advances, we seeked to automatize this process. Automatic cat recognition became the fundamental problem of modern society.

How do data reach us? First, let's clarify what we mean by "data". From a computer's perspective, data simply comprises collections of numbers, making them amenable to mathematical computations. Then, what sets them apart? We can agree that a song differs in nature from an image, which, in turn, differs from a movie. Specifically, we can view a song as a collection of discretized numbers, where each number is a sampled version at a regular interval of a continuous sound; an image can be conceived as a grid of pixels, with each pixel storing color information; finally, a video can be thought of as an ordered sequence of images, and so, an ordered sequence of grids of pixels storing color information. Moreover, the tasks we wish to perform on them dictate differing processing architectures. Rather than delve into how to execute this processing, we focus on a commonality shared by these examples: one aspect regarding their domain and another concerning their values (which we will call signals[1]).

Domain. Now, let's consider their domain. Concerning the domain, all the aforementioned examples exhibit a well-defined structure with a clear notion of ordering and precedence. For instance, in the realm of sound, there exists a distinct temporal ordering: time instant t precedes time instant $t+1$. Similarly, each pixel within an image maintains a spatial relationship, such as left-right or top-bottom, with respect to neighboring pixels. Likewise, a movie can be understood as a sequential arrangement of images. Typically, each domain point has a consistent number of neighbors, except for those positioned at the domain's borders. Hence, this property of well-ordering or locality among samples is shared across their domains.

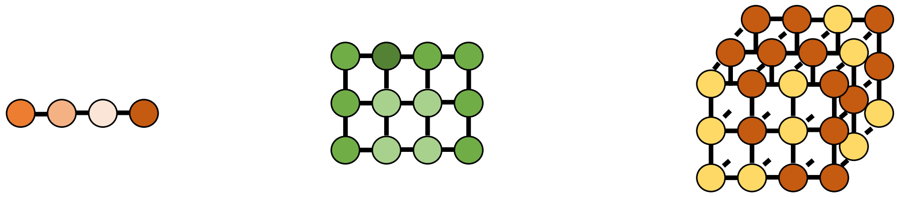


Figure 1.2: An illustration of different data types: a discrete-time signal (left), an image (center), and a video (right). Each circle symbolizes an element of the domain: for the discrete-time signal, it denotes an index t; for the image, it signifies an ordered pair (x, y) representing the pixel location; and for the video, it indicates the triplet (x, y, t) denoting the pixel location at a specific time instant t. The color of each circle represents the value (i.e., the signal) at the indexed domain point.

Signal. Another property pertinent to this type of data concerns their signals, that is, the value that these types of data assume on their domain. In audio, samples at time t and $t+1$ tend to be similar. Similarly, neighboring pixels in images typically share similar colors, barring edges. In a video, successive frames have similar content except when a change of scene is present. Hence, it is generally true, particularly for natural signals, that adjacent samples display similar values — a characteristic

[1]Notice how we use a loose definition of the term signal here, since formally a signal is a function and so it comprehends both the domain and codomain.

leveraged in many signal processing tasks through specific architectural approaches, such as smoothing or denoising. This structural similarity among samples effectively informs processing architectures designed to operate on such data.

1.1.2. NETWORKS

However, we often encounter data where the concept of precedence among samples is not immediately evident or does not exist at all. In other words, the domain of such data is unstructured and irregular, yet it still maintains some form of relationship. Examples of such data, as illustrated in Figure 1.3, include social networks, cooperative autonomous agents, molecules, transportation networks, brain networks, and local area networks. Despite their structural dissimilarity to time and space, there may still be a necessity to conduct signal processing on such data.

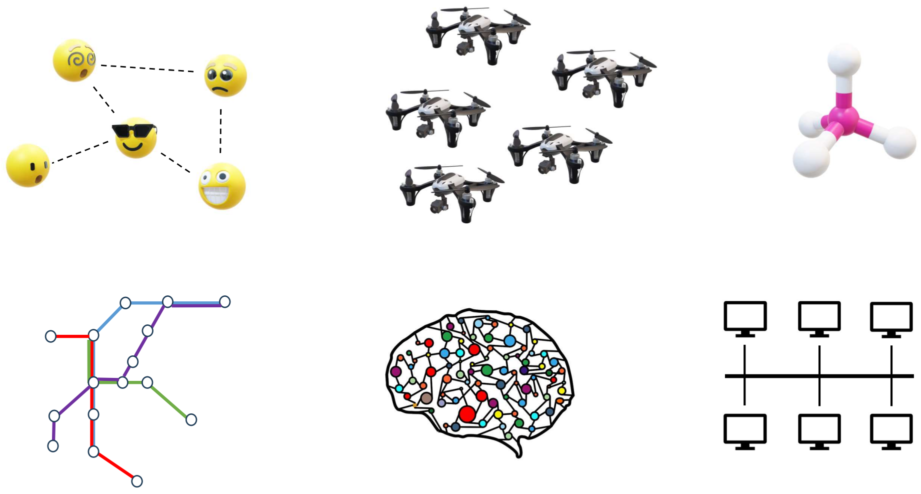


Figure 1.3: Examples of networks: each network is composed by a set of entities (for instance users, drones and atoms) and a set of relationships between different entities (for instance friendship, communication range and atomic bond).

For example, in a social network, we might attempt to deduce an individual's preferences based on the preferences of his/her "neighbors" within the network, i.e., individuals with whom they share a connection. Similarly, in the realm of molecules, we might endeavor to evaluate its suitability for specific antibiotic treatments by analyzing its atomic compositions. In all these scenarios, the data is supported by what is known as a "network" — a mathematical abstraction utilized to depict a system of entities with diverse types of relationships among them. In the realm of signal processing, each entity also carries data.

These networks are typically visually represented using circles, denoted as nodes or vertices, to represent the entities of interest, and lines connecting two entities, known as links or edges, to signify the presence of some form of relationship. We purposefully maintain the term "some form" ambiguous, as it is the responsibility of the researcher to model the network and encapsulate the relationships relevant to the given objective. We will delve into how we can formally describe such a concept using graph theory, but for now, we will limit our discussion to a high-level overview. Let's now provide some examples of networks with their relative entities and relationships:

- *social networks*: each entity of the network is a user and different users can be related based on their friendship or not. The data that we can gather are the demographic information of users;
- *transportation networks*: think of air, maritime, road and logistical networks. These systems are all made of hubs (airports, ports, junctions), and the link represents the connection between two different hubs. The data represent for instance the number of daily passengers in each hub;
- *brain networks*: each entity of the network represents a region of interest of the brain composed of neurons performing a specific task, while the link between each entity represents a physical “pathway” between two different regions, or a virtual functional relationship of interest, for instance the case in which two different regions activate together in response to a stimulus. Data in this case can be gathered in the form of electroencephalography (EEG) measurements.
- *molecules*: each entity is an atom and the connections between atoms describe the chemical bonds. The data we can gather are the atomic and mass number, electronic configuration, spatial configuration.

Thus, we come to comprehend how data are not merely a collection of numbers: they are objects possessing an underlying geometry, which is leveraged in processing such data within architectures. Traditional methods developed for standard data modalities (audio, images, videos) would fail to exploit a more general geometrical structure. This is when graph signal processing (GSP) emerged as a tool to process networked data, blending together notions from signal processing and graph theory.

1.1.3. Graph Signal Processing

Graphs have long served as a fundamental tool for modeling complex systems, with entire branches of mathematics devoted to their study and their properties. One of the earliest formal publications on this subject is credited to Leonhard Euler, whose work on the Seven Bridges of Königsberg, published in 1736, is considered the inaugural paper in the history of graph theory [1]. Consequently, a significant portion of scientific inquiry has been devoted to investigating networks and their intrinsic characteristics.

However, in the early 2000s, the field of signal processing experienced a paradigm shift when researchers recognized the applicability of graph theory to classical signal processing tasks such as denoising, sampling, and interpolation. In essence, it became apparent that the domain, previously implicitly understood when defining operations on signals, could now be explicitly represented through a graph structure. The instantiation of this graph, along with the corresponding architectures designed to operate on the signals residing within it, ultimately defines the modality of the data. For instance, the "time-domain" could be understood of as a path-graph; an image through a grid graph; a video through a lattice graph.

Although the data typically manifests itself as a vector (or more general as a tensor), imparting an underlying structure to this data vector can significantly impact the efficacy of a processing architecture. In this context, the delineation between a proficient and suboptimal processing architecture often hinges on the ability to associate a meaningful structure with the data vector.

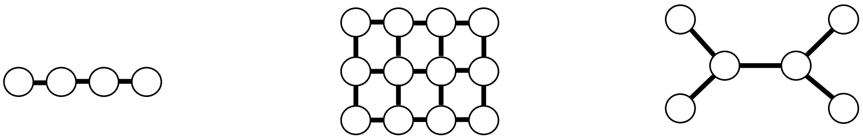


Figure 1.4: Three examples of graphs: (left) a line graph, where each node can be the index of a discrete time signal; a grid graph, where each node represents a pixel; a more general graph, for instance describing the atomic composition of ethylene C_2H_4.

This marked the inception of GSP [2, 3]: harnessing the network structure to enrich signal processing tasks on network data. Over the course of a decade, significant strides were made in this realm, with numerous classical signal processing tasks finally finding their graph-based counterparts. We will lay down the GSP background in Chapter 2.

What is missing then? Not every aspect of graph-based processing seamlessly aligns with the natural flow experienced in time and space domains; that is why fundamental research in GSP was and still is needed. We will limit ourselves here to only briefly mention which are the gaps that we have tried to fill within the research thrusts of this dissertation.

RESEARCH THRUST I: GRAPH LEARNING

Graph signal processing facilitates the analysis of signals defined over networks. But what if we possess the data without the corresponding graph structure? Is it possible to deduce the graph solely from the data? This problem goes under the name of graph topology inference or graph learning, and it has gained a lot of attention in the past decade due to the increased pervasiveness of graph-based architectures (which do need a graph structure to operate on) and due to the interpretability given by a graph structure. We refer the interested reader to the excellent overview papers [4, 5].

As we will see, many network processes can be effectively modeled using graph filters, which are architectures designed to process signals defined over graphs.

Typically, these architectures are parametric in both the filter coefficients and the matrix representation of the graph. Thus, upon data availability, one would like to estimate the graph filter which maps input signals to output signals. This is tantamount to estimating its parameters. While many studies assume complete knowledge of the graph and the strength of its connections, and focus solely on estimating the filter coefficients, there are scenarios where only partial information about the graph structure is available. In this case, can we simultaneously learn both the filter coefficients and a complete description of the graph in a principled way? This question is explored in Chapter 4.

In addition, many works miss a fundamental aspect often arising in real-world networks: time-variability. In other words, many networks are time-varying, i.e., the connection between the entities changes over time. Think for instance of mobile communication networks or functional connectivity networks in the brain. How to tackle the dynamism of such networks in a principled way? We addressed this question in Chapter 5.

RESEARCH THRUST II: CONVOLUTION AND FREQUENCY DOMAIN

From basic system theory, we know that a convolution in one domain (be it time or frequency) equates to pointwise multiplication in the other domain: this is perhaps the most fundamental theorem of signal processing, i.e., the convolution theorem. This holds true for GSP only in one direction, namely a graph convolution (defined in the vertex domain) equates to pointwise multiplication in the frequency domain. In the other direction, i.e., from the frequency to the vertex domain, there is a lack of intuitiveness around this concept; what does graph convolution in the frequency domain mean if a graph does not inherently exist within this domain? An answer to this question has been given by [6, 7] with the introduction of a "frequency-domain graph" which essentially captures the structure of the frequency domain; in this case, a graph convolution in the frequency domain also equates to a pointwise multiplication in the vertex domain.

As we will demonstrate, the *graph convolution theorem* (along with the classical convolution theorem) operates under the assumption that the convolution's implementation architecture is shift-invariant, meaning it commutes with the shift operation. This assumption is akin to considering the data to exhibit some sort of "stationarity" property. However, recognizing that not all systems exhibit shift-invariance and encompassing a broader class of operators, we illustrate how a node-variant graph convolution in one domain can be expressed as another node-variant convolution in the other domain. This extends the scope of the convolution theorem, encompassing scenarios such as time-varying filters. Different instantiations of the architecture's coefficients and of the shift lead to different convolution theorems. This innovative concept is discussed in Chapter 6.

1.2. OUTLINE

This section provides a description for each chapter of this dissertation, starting from Chapter 2.

Chapter 2 This chapter lays down the formal (graph) signal processing background necessary to understand the following chapters. We start with the definition of *shift* which is built into the definition of convolution and Fourier analysis in the time domain. Then, we translate these concepts into the graph domain. Finally, we outline the problem of network topology identification from data, with illustrative examples.

Chapter 3 This chapter presents a more formal and mathematical treatment of the research thrusts introduced in the previous section. The focus of this chapter is not to delve deeply into the technical solution to these problems but rather to introduce them in a manner accessible to a broader audience, gradually transitioning towards a formalized treatment of the issues.

Chapter 4 This chapter presents a data-driven method aimed at simultaneously estimating both the filter coefficients and the graph structure defining a graph filter, which characterizes the dynamics observed in the data. This approach, building on a sequential convex approximation scheme, operates under the assumption that the sparsity pattern of the graph is known.

Chapter 5 This chapter proposes an algorithmic framework to learn time-varying graphs from online data. The generality offered by the framework renders it model-independent, i.e., it can be theoretically analyzed in its abstract formulation and then instantiated under a variety of model-dependent graph learning problems.

Chapter 6 This chapter proposes a novel convolution theorem which encompasses the well known convolution theorem in (graph) signal processing as well as the one related to time-varying filters. Specifically, we show how a node-wise convolution for signals supported on a graph can be expressed as another node-wise convolution in a frequency domain graph, different from the original graph.

Chapter 7 This conclusive chapter summarizes the findings of the research period and delineate possible future avenues of research.

1.3. List of Contributions

We conclude this chapter with an itemized list of the peer-reviewed scientific contributions made during the Ph.D. period. Only the articles with a * symbol will be however the focus of this thesis.

Journals

$*$**J2** **Natali, Alberto** and Geert Leus. “A Generalization of the Convolution Theorem and its Connections to Non-Stationarity and the Graph Frequency Domain." *IEEE Transactions on Signal Processing* 72 (2024): 3424 - 3437.

∗**J1** **Natali, Alberto**, Elvin Isufi, Mario Coutino, and Geert Leus. “Learning time-varying graphs from online data." *IEEE Open Journal of Signal Processing* 3 (2022): 212-228.

CONFERENCES AND WORKSHOPS

C7 Van der Hoeven, Jelmer, **Natali, Alberto**, and Geert Leus. "Forecasting Graph Signals with Recursive MIMO Graph Filters." *2023 31st European Signal Processing Conference (EUSIPCO)*. IEEE, 2023.

C6 **Natali, Alberto** and Geert Leus. “Blind Polynomial Regression" In *"Blind polynomial regression." In ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1-5. IEEE, 2023.

C5 **Natali, Alberto** and Geert Leus. “A General Convolution Theorem for Graph Data" In *2022 56th Asilomar Conference on Signals, Systems, and Computers*, pp. 48-52. IEEE, 2022.

C4 **Natali, Alberto**, Mario Coutino, Elvin Isufi, and Geert Leus. “Online Time-Varying Topology Identification Via Prediction-Correction Algorithms." In *ICASSP2021: The IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 5400-5404. IEEE, 2021.

C3 **Natali, Alberto**, Elvin Isufi, Mario Coutino, and Geert Leus. “Online Graph Learning From Time-Varying Structural Equation Models." In *2021 55th Asilomar Conference on Signals, Systems, and Computers*, pp. 1579-1585. IEEE, 2021.

∗**C2** **Natali, Alberto**, Mario Coutino, and Geert Leus. “Topology-aware joint graph filter and edge weight identification for network processes." In *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1-6. IEEE, 2020.

C1 **Natali, Alberto**, Elvin Isufi, and Geert Leus. “Forecasting multi-dimensional processes over graphs." In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5575-5579. IEEE, 2020.

1.4. CONCLUSIONS

In this chapter, we introduced the concepts of data and networks and illustrated various scenarios where networked data are encountered. We highlighted how the distinct characteristics of networked data, different from data supported on traditional grid-like domains, shape the properties of the data. We then presented graph signal processing as a valuable tool for managing and analyzing this type of data, covering tasks ranging from interpolation and denoising to classification and sampling. Additionally, we outlined two significant areas for further research advancement in the field: learning time-varying networks from data and the development of a generalized convolution theorem for shift-variant systems. Moving forward to Chapter 2, we will delve into the essential mathematical background required to comprehend the problem formulations discussed in Chapter 3, which will receive comprehensive treatment in subsequent chapters (Chapter 4, Chapter 5, and Chapter 6).

REFERENCES

[1] J. A. Bondy, U. S. R. Murty, *et al. Graph theory with applications.* Vol. 290. Macmillan London, 1976.

[2] A. Sandryhaila and J. M. Moura. “Discrete signal processing on graphs”. In: *IEEE transactions on signal processing* 61.7 (2013), pp. 1644–1656.

[3] 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”. In: *IEEE signal processing magazine* 30.3 (2013), pp. 83–98.

[4] G. Mateos, S. Segarra, A. G. Marques, and A. Ribeiro. “Connecting the dots: Identifying network structure via graph signal processing”. In: *IEEE Signal Processing Magazine* 36.3 (2019), pp. 16–43.

[5] X. Dong, D. Thanou, M. Rabbat, and P. Frossard. “Learning graphs from data: A signal representation perspective”. In: *IEEE Signal Processing Magazine* 36.3 (2019), pp. 44–63.

[6] G. Leus, S. Segarra, A. Ribeiro, and A. G. Marques. “The dual graph shift operator: Identifying the support of the frequency domain”. In: *Journal of Fourier Analysis and Applications* 27.3 (2021), p. 49.

[7] J. Shi and J. M. Moura. “Graph signal processing: Dualizing gsp sampling in the vertex and spectral domains”. In: *IEEE Transactions on Signal Processing* 70 (2022), pp. 2883–2898.

2 Signal Processing: Time and Graphs

Learn. Teach. Repeat.

A.N.

Signal processing is the discipline which studies, analyzes and synthesizes signals. For the scope of this dissertation, signals can be thought of as objects to encode information in a mathematical form. Why do we want to encode information? And why do we bring this information in a mathematical form and not into another one? Possible answers to these questions can be found in the interesting readings [1]. In general, we can say that we want to encode information in order to predict, make decisions, or learn about a phenomenon of interest. Since the information we are concerned about is encoded in numerical values, we use mathematics as the language to process it, as we would use English to encode verbal communication. For classic introductory texts on signals and systems, we refer the reader to [2, 3].

In this chapter, our goal is to introduce signal processing from an "operational" perspective, focusing on the methods by which we manipulate signals. We begin our discussion by starting from the time domain, where readers may encounter more recognizable concepts like convolution and filters. Subsequently, we broaden the scope to encompass a more universal domain, represented by a graph.

2.1. Signal Processing in time

We start our journey with a deep treatment of the most important mathematical operation in signal processing: the *convolution*. The reader might have come across this term several times, perhaps in introductory signal processing courses or in the context of convolutional neural networks for image processing [4]. Here, we use a pedagogic approach to build its definition, starting from basic principles. Specifically, we start from its basic building block, the most atomic operation in signal processing, which will be crucial for the rest of the manuscript: the *shift*.

Consider a scalar function $x(\cdot)$ describing a phenomenon of interest, for instance the temperature (in °C) over time of your favorite city. Without any loss of generality and for simplicity of exposition, we assume that the domain and codomain of the function x are not restricted to specific values (as temperature should) but span the entire real line $\mathbb{R}$. Thus, for any value $t \in \mathbb{R}$ representing a specific time instant, we have an associated function value $x(t) \in \mathbb{R}$ representing the temperature at time t. Because we want to process this signal into the digital world, we use a *discretized* version of the time signal $x(t)$ sampled at regular time intervals, i.e., we sample $x(\cdot)$ at multiple time instants of a period of T_s. That is, we sample at $0, T_s, 2T_s, \ldots, (N-1)T_s$, obtaining the discrete version of the signal $x(0), x(T_s), x(2T_s), \ldots, x((N-1)T_s)$. Concisely, for $n = 0, \ldots, N-1$, we have the values $x(nT_s)$. We will omit the specific sampling period T_s and simply denote $x(nT_s)$ as $x_n = x(nT_s)$. Fig. 2.1 (top) shows an example of a continuous time-signal $x(t)$ and its discretized version x_n at the N distinct points. We conveniently collect the points $x_0, \ldots, x_{N-1}$ in the vector $\mathbf{x} = [x_0, x_1, \ldots, x_{N-1}]^\top$, so as to make it amenable to algebraic computations.

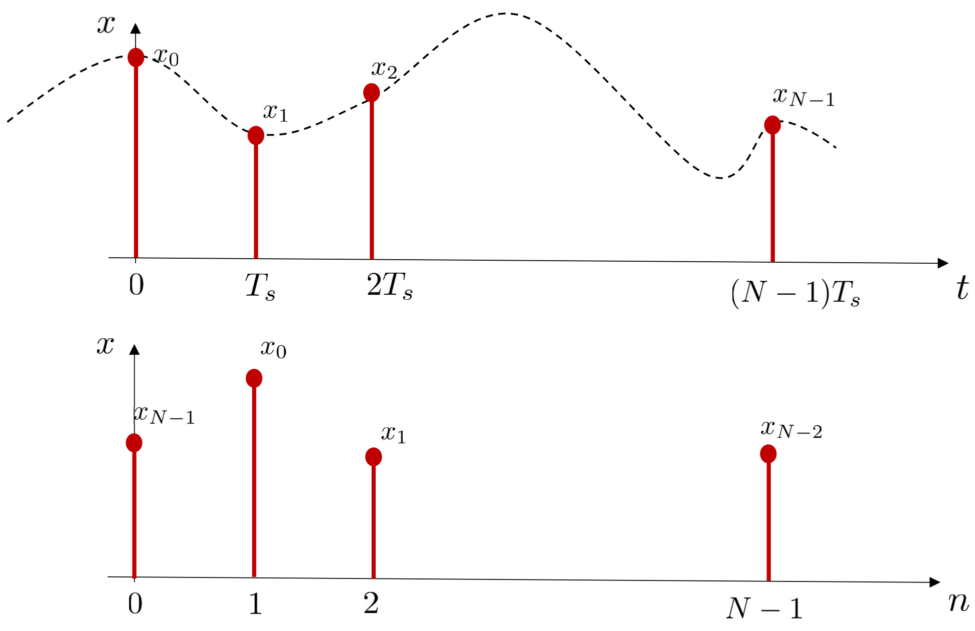


Figure 2.1: (Top) A continuous time signal $x(t)$ (black dotted line) is sampled at N distinct points $n = 0, \ldots, N-1$, yielding a discrete-time signal $x_0, \ldots, x_{N-1}$. (Bottom) The shifted signal $\mathbf{x}^{(1)}$: each entry of the vector $\mathbf{x}$ is shifted one position to the right.

Introducing a delay on a signal $\mathbf{x}$ involves shifting each entry of the signal one position to the right, as represented in its transpose form. This operation results in a new signal, denoted as $\mathbf{x}^{(1)}$, where each entry is shifted by one position to the right.

In mathematical terms, $\mathbf{x}^{(1)}$ is defined as:

$$\mathbf{x}^{(1)} = [x_{N-1}, x_0, \dots, x_{N-2}]^\top, \tag{2.1}$$

where the superscript $^{(1)}$ indicates that one consecutive shift has been applied to the signal. For further reference, the number l in the superscript $^{(l)}$ denotes the number of consecutive shifts applied to the signal.

Since we are dealing with finite-length vectors, we use the notion of circular shift, or *shift modulo* N. In other words:

$$x_n^{(1)} = x_{(n-1)(\text{mod} N)}, \tag{2.2}$$

where we recall that an integer $n \in \mathbb{N}$ is equal to $l(\text{mod}N)$ if and only if $l-n$ is an integer multiple of N. Mathematically:

$$n = l(\text{mod}N) \iff l - n = iN \quad \text{for some } i \in \mathbb{Z}. \tag{2.3}$$

The mapping $\mathbf{x} \to \mathbf{x}^{(1)}$ can be algebraically achieved by means of an important linear operator, the circular *shift operator* (or delay matrix) $\mathbf{S}_c$, defined as:

$$\mathbf{S}_c = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & 0 \\ 0 & \cdots & 0 & 1 & 0 \end{bmatrix} \tag{2.4}$$

We can then express the shifting operation (2.2) as the matrix-vector product (see also Fig. 2.1 (bottom) for an illustration):

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

Consecutive applications, say l, of the matrix $\mathbf{S}_c$ to the signal $\mathbf{x}$, perform l consecutive shifts of the signal $\mathbf{x}$; formally:

$$\mathbf{x}^{(l)} = \mathbf{S}_c^l \mathbf{x} = \mathbf{S}_c(\mathbf{S}_c^{l-1}\mathbf{x}) = \mathbf{S}_c \mathbf{x}^{(l-1)} \tag{2.6}$$

with $\mathbf{x}^{(0)}$ initialized as $\mathbf{x}^{(0)} = \mathbf{x}$.

The recursive implementation offered in (2.6) can be appreciated by noticing that $\mathbf{S}_c$ is a sparse matrix. This, from a computational standpoint, reduces the computational workload significantly during matrix-vector product operations; from a conceptual standpoint, its sparsity pattern reveals that any signal value $x_n^{(l)}$ for a given entry n only requires the knowledge of the value at the preceding time instant $n-1$, denoted as $x_{n-1}^{(l-1)}$. This notion of *locality*, as we will see, extends to more general domains, beyond time, which will be modeled as graphs. We can then say that the structure of $\mathbf{S}_c$ captures dependencies among various signal values. Thus, $\mathbf{S}_c$ not only acts as an operator that transforms signals to their shifted versions but can also be viewed as a matrix representing the domain on which these signals are defined. In essence, we can consider $\mathbf{S}_c$ as a potential representation of the time domain.

2.1.1. Circular Convolution

Matrix $\mathbf{S}_c$ belongs to an important class of matrices called *circulant* matrices, i.e., matrices in which all columns (and rows) are composed of the same elements, and each column (row) is a shifted/delayed version of the previous one. As we now will illustrate, this class of matrices is pivotal in describing the convolution operation among two signals, and to introduce the notion of *frequency* from a purely algebraic approach. Although many different types of convolution do exist in different contexts and for different scopes, we focus here on the *circular convolution*.

Given a signal $\mathbf{x} = [x_0, \dots, x_{N-1}]^\top$, often referred to as input, and another signal with the same length $\mathbf{p} = [p_0, \dots, p_{N-1}]^\top$, often called *filter*[1] or kernel, the circular convolution between $\mathbf{p}$ and $\mathbf{x}$, denoted as $\mathbf{p} \star \mathbf{x}$, is the operation resulting in the output signal $\mathbf{y} = [y_0, \dots, y_{N-1}]^\top$ with:

$$y_n = p_0 x_n + p_1 x_n^{(1)} + \ldots + p_{N-1} x_n^{(N-1)} = \sum_{l=0}^{N-1} p_l x_{(n-l)(\mathrm{mod} N)} \tag{2.7}$$

That is, the (circular) convolution represents, for each output value y_n, the weighted average of the entries of the input signal $\mathbf{x}$, where the weights are specified by the filter coefficients $\mathbf{p}$. Rearranging (2.7) for all $n = 0, \dots, N-1$, we obtain the convolution $\mathbf{y} = \mathbf{p} \star \mathbf{x}$ in matrix-vector form:

$$\underbrace{\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{N-1} \end{bmatrix}}_{\mathbf{y}} = \underbrace{\begin{bmatrix} p_0 & p_{N-1} & p_{N-2} & \ddots & p_2 & p_1 \\ p_1 & p_0 & p_{N-1} & \ddots & p_3 & p_2 \\ p_2 & p_1 & p_0 & \ddots & \ddots & p_3 \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ p_{N-1} & p_{N-2} & \ddots & \ddots & p_1 & p_0 \end{bmatrix}}_{\mathbf{H}(\mathbf{p})} \underbrace{\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{N-1} \end{bmatrix}}_{\mathbf{x}} \tag{2.8}$$

where we denoted with $\mathbf{H}(\mathbf{p})$ the matrix implementing the convolution operation, thus responsible to map $\mathbf{x}$ to $\mathbf{y}$. We can then state the following: the convolution between two signals $\mathbf{p}$ and $\mathbf{x}$ can be implemented as a matrix-vector product, where the matrix $\mathbf{H}(\mathbf{p})$ is circulant and fully specified by the filter response $\mathbf{p}$[2].

An important observation here, which will be crucial for understanding the ensuing chapters, is the relationship between the circular shift $\mathbf{S}_c$ and the matrix representation of the filter $\mathbf{H}(\mathbf{p})$. Specifically, each column is obtained from the preceding one when pre-multiplied by the circular shift $\mathbf{S}_c$, i.e., $\mathbf{H}(\mathbf{p}) = [\mathbf{p}, \mathbf{S}_c\mathbf{p}, \dots, \mathbf{S}_c^{N-1}\mathbf{p}]$. This suggests a functional dependence of $\mathbf{H}(\mathbf{p})$ not only on the vector $\mathbf{p}$ but also on the matrix $\mathbf{S}_c$. It is easy to show that $\mathbf{H}(\mathbf{p})$ can be written as:

$$\mathbf{H}(\mathbf{p}, \mathbf{S}_c) = \sum_{l=0}^{N-1} p_l \mathbf{S}_c^l. \tag{2.9}$$

[1]The vector $\mathbf{p}$ is called filter impulse response, since it describes the output of the filter when excited with a unit impulse.

[2]Since convolution respects the associative property, it holds $\mathbf{y} = \mathbf{x} \star \mathbf{p} = \mathbf{H}(\mathbf{x})\mathbf{p}$

In other words, the circulant matrix $\mathbf{H}(\mathbf{p})$ can be expressed as a matrix polynomial involving the circulant shift matrix $\mathbf{S}_c$, where the entries of $\mathbf{p}$ are the coefficients of such polynomial. In (2.9), we explicitly treat $\mathbf{S}_c$ as an argument of the matrix $\mathbf{H}(\cdot)$ to underscore its significant role as a parameter of the matrix; this will come handy when we will introduce filters in a graph context. A crucial observation is that *any* circulant matrix can be represented by a matrix polynomial in terms of $\mathbf{S}_c$. This observation carries implications for defining a frequency domain and its corresponding *transforms*. More specifically, variations in shifts result in distinct frequency concepts, diverse filter shapes, and varied transforms[5].

By substituting the filter parameterization (2.9) into the expression of $\mathbf{H}(\mathbf{p}, \mathbf{S}_c)$ in (2.8), we get the convolution operation in matrix-vector form (which is the vector-counterpart of (2.7)):

$$\mathbf{y} = \sum_{l=0}^{N-1} p_l \mathbf{S}_c^l \mathbf{x} = p_0 \mathbf{x} + p_1 \mathbf{x}^{(1)} + \ldots + p_{N-1} \mathbf{x}^{(N-1)}, \tag{2.10}$$

which shows once again, this time in vector form, how the output $\mathbf{y}$ is a linear combination of shifted (delayed) versions of the input signal $\mathbf{x}$, where each shifted signal $\mathbf{x}^{(l)}$ is scaled by the filter tap p_l. Filters of this type are referred to as *time-invariant*, because for any y_n, the filter coefficients $p0, \ldots, p_{N-1}$ do not change, i.e., they are independent from the time instant n. We will see how this type of filters simplifies the analysis of signals.

2.1.2. THE FREQUENCY DOMAIN

We now gradually move to the notion of frequency, a central concept in signal processing, as it provides an alternative representation for (processing) signals, and for the architectures operating on them. To define it, we rely on an algebraic approach. We start by recalling what it means for a matrix to be diagonalizable.

Definition 2.1.1. *A square matrix* $\mathbf{A}$ *is said to be diagonalizable if there exists an invertible matrix* $\mathbf{V}$ *and a diagonal matrix* $\mathbf{\Lambda}_a$ *such that:*

$$\mathbf{V}^{-1}\mathbf{A}\mathbf{V} = \mathbf{\Lambda}_a \tag{2.11}$$

In other words, matrix $\mathbf{A}$ is diagonalizable if we can identify a change of basis matrix $\mathbf{V}$ such that $\mathbf{A}$, when expressed in terms of $\mathbf{V}$, becomes diagonal. The resulting diagonal matrix, denoted as $\mathbf{\Lambda}_a$, represents the linear transformation $\mathbf{A}$ in this newly defined basis. Working with $\mathbf{\Lambda}_a$ in the new basis streamlines computations compared to operating with $\mathbf{A}$ in the original basis.

For example, matrix-vector multiplication simplifies to a straightforward element-wise multiplication of the diagonal elements of $\mathbf{\Lambda}_a$ and the corresponding entries of the vector. Consequently, this operation essentially represents a stretching operation on the vector. Additionally, understanding powers of a diagonal matrix becomes intuitive as it involves simply raising each individual diagonal element to the desired power. Notice that an alternative but equivalent condition to (2.11) is given by stating that

$\mathbf{A}$ admits an eigenvalue decomposition (EVD), represented as $\mathbf{A} = \mathbf{V}\mathbf{\Lambda}_a\mathbf{V}^{-1}$, where $\mathbf{V}$ denotes the matrix of eigenvectors and $\mathbf{\Lambda}_a$ the matrix of eigenvalues.

An important question arises: does there exist a set of matrices $\mathcal{M}$ such that they can all be diagonalized by the same change of basis matrix $\mathbf{V}$? This leads to another definition.

Definition 2.1.2. *A set $\mathcal{M}$ of matrices is called simultaneous diagonalizable if there exists a unique invertible matrix $\mathbf{V}$, such that for every matrix $\mathbf{M} \in \mathcal{M}$, the matrix:*

$$\mathbf{V}^{-1}\mathbf{M}\mathbf{V} \text{ is diagonal,} \tag{2.12}$$

that is, if matrix $\mathbf{V}$ is a matrix of eigenvectors for any $\mathbf{M} \in \mathcal{M}$.

But how can we know whether a given set $\mathcal{M}$ of matrices is simultaneously diagonalizable? It can be shown that *a set of matrices $\mathcal{M}$ is simultaneously diagonalizable if and only if they all mutually commute*, that is, if $\mathbf{M}_1\mathbf{M}_2 = \mathbf{M}_2\mathbf{M}_1 \,\forall\, \mathbf{M}_1, \mathbf{M}_2 \in \mathcal{M}$, and at least one of the matrices has simple (non-repeated) eigenvalues.

The importance of the preceding statement resides in the fact that upon the identification of a matrix with simple eigenvalues belonging to the set $\mathcal{M}$, finding its eigenvector matrix $\mathbf{V}$ implies finding the eigenvector matrix for all the other matrices of the set at no additional cost.

It is easy to show that the set of circulant matrices is simultaneously diagonalizable, i.e., taken any two circulant matrices, they commute. Therefore, upon identifying a circulant matrix with simple eigenvalues, its eigenvectors serve to diagonalize the entire collection of circulant matrices simultaneously. Given that any circulant matrix can be expressed as a matrix polynomial of $\mathbf{S}_c$ (and thus, they commute), the circular shift operator $\mathbf{S}_c$ emerges as a promising candidate for such an exploration. Without delving into the mathematical proof, we can state that $\mathbf{S}_c$ *is diagonalized by the inverse discrete Fourier transform (DFT) matrix*, thereby implying that any circulant matrix is likewise diagonalized by the inverse DFT matrix. Let us now revisit the definition of the DFT matrix.

Consider the Nth primitive root of unity, i.e., $\omega = e^{-j2\pi/N}$, where j is the imaginary unit with $j^2 = -1$. The DFT matrix is the $N \times N$ Vandermonde matrix $\mathbf{F}$ having in the (i,k)th entry the value $F_{ik} = \omega^{(i-1)(k-1)}$, for $i,k = 1,\dots,N$, i.e.,:

$$\mathbf{F} = \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \cdots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \cdots & \omega^{2(N-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \cdots & \omega^{3(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \omega^{3(N-1)} & \cdots & \omega^{(N-1)(N-1)} \end{bmatrix}. \tag{2.13}$$

Its inverse, $\mathbf{F}^{-1}$ represents the eigenvector matrix of $\mathbf{S}_c$, a change of basis matrix under which $\mathbf{S}_c$ becomes a diagonal operator.

To show this, first notice that $\mathbf{F}^{-1}$ is given by:

$$\mathbf{F}^{-1} = \frac{1}{N}\mathbf{F}^{*}, \tag{2.14}$$

where $\mathbf{F}^{*}$ is the complex conjugate of $\mathbf{F}$. Denoting with $\mathbf{f}_l^{*}$ the lth column (starting from index 0) of $\mathbf{F}^{*}$, we have:

$$\mathbf{S}_c\mathbf{f}_l^{*} = \omega^l\mathbf{f}_l^{*}, \quad l = 0,\dots,N-1, \tag{2.15}$$

that is, $\mathbf{f}_l^{*}$ is an eigenvector of $\mathbf{S}_c$ with eigenvalue equal to ω^l; in matrix form:

$$\begin{aligned}\mathbf{S}_c &= \mathbf{F}^{-1}\,\mathrm{Diag}(\mathbf{f}_1)\mathbf{F} && (2.16)\\ &= \frac{1}{N}\mathbf{F}^{*}\,\mathrm{Diag}(\mathbf{f}_1)\mathbf{F} && (2.17)\\ &= (\frac{1}{\sqrt{N}}\mathbf{F}^{*})\,\mathrm{Diag}(\mathbf{f}_1)(\frac{1}{\sqrt{N}}\mathbf{F}). && (2.18)\end{aligned}$$

We present three diagonalizations of $\mathbf{S}_c$ for clarity and to highlight various aspects of the decomposition. The first one, represented by (2.16), corresponds to the standard EVD, further extended in (2.17), where we explicitly substitute the inverse $\mathbf{F}^{-1}$ as in (2.14). Lastly, (2.18) illustrates the DFT and its inverse symmetrically, with the eigenvector matrix being unitary, unlike (2.16) and (2.17). We will opt for (2.16) due to its concise representation, while acknowledging its lack of unitarity.

The eigenvectors $\{\mathbf{f}_l^{*}\}_l$ carry a notion of variation. To see this, remember that $e^{j\theta} = \cos(\theta) + j\sin(\theta)$, i.e., each eigenvector $\mathbf{f}_l^{*}$ is a sampled version of a sinusoidal-like wave with frequency $2\pi l/N$; thus they are often called oscillating or resonant modes. Specifically, $\mathbf{f}_0^{*}$ is the constant mode (often referred to as the DC mode), and $\{\mathbf{f}_l^{*}\}_{l=1}^{N-1}$ is a set of waves with increasing frequency for increasing l (up to $(N-1)/2$), represented by its associated eigenvalue $\omega^l = e^{-2\pi l/N}$.

Discrete Fourier Transform

The application of matrix $\mathbf{F}$ to a signal $\mathbf{x}$ to obtain a new signal $\hat{\mathbf{x}}$, i.e., $\hat{\mathbf{x}} := \mathbf{F}\mathbf{x}$, performs the so-called discrete Fourier transform (DFT) of the signal. It provides an alternative representation for the signal $\mathbf{x}$ (a change of basis) over which properties and information of the signal are exposed. Indeed, since $\mathbf{F}^{-1} = \mathbf{F}^{*}/N$, we can see that the signal $\mathbf{x}$ can be expressed through the inverse DFT (iDFT) as:

$$\mathbf{x} = \frac{1}{N}\mathbf{F}^{*}\hat{\mathbf{x}} = \frac{1}{N}\sum_{n=0}^{N-1}\hat{x}_n\mathbf{f}_n^{*} \tag{2.19}$$

This expression illustrates that $\mathbf{x}$ can be reconstructed as a combination of oscillating modes, each characterized by distinct weights determined by the entries of $\hat{\mathbf{x}}$. Through a straightforward examination of these weights, one can discern whether the signal behaves akin to a low-pass or high-pass signal.

Perhaps in a subtle way we have discovered an important fact: the eigenvectors of the shift matrix $\mathbf{S}_c$ provide a *transformation* from the space of discrete-time signals to the space of frequency-domain signals. While in an N-dimensional Euclidean space, the vector $\mathbf{x}$ and its frequency-domain counterpart $\hat{\mathbf{x}}$ remain identical (only their coefficients in their respective bases vary), our focus typically lies in analyzing their coefficients along a real number line. Specifically, we consider the original domain of our signal $\mathbf{x}$ to be the set of natural numbers, while the domain of the transformed DFT signal $\hat{\mathbf{x}}$ to be the eigenvalues of the shift matrix, corresponding to the complex exponentials, or essentially, the frequencies. Fig. 2.2 illustrates this concept. We

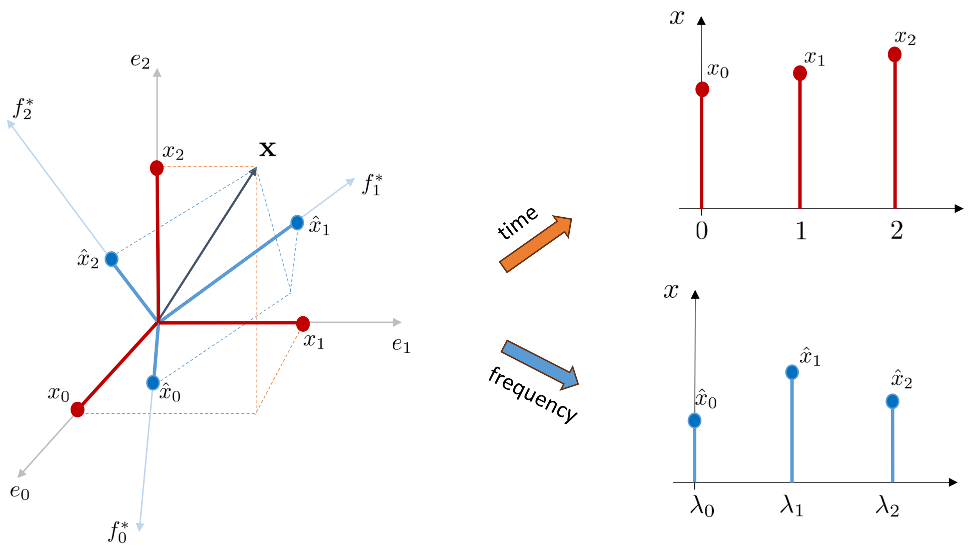


Figure 2.2: Representation of a signal $\mathbf{x} \in \mathbb{R}^3$. Such vector can be described by the vector $\mathbf{x}$ collecting the weighting coefficients of the canonical basis $\{\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2\}$ or by its frequency representation $\hat{\mathbf{x}}$, which collects the weighting coefficients of the "frequency basis" of resonant modes $\{\mathbf{f}_0, \mathbf{f}_1, \mathbf{f}_2\}$.

shall not forget why we computed the eigenvectors of $\mathbf{S}_c$ in the first place. This stemmed from its property of commuting with circulant matrices. Consequently, by determining its eigenvector matrix, we effectively obtain the eigenvectors for all circulant matrices without additional effort. As a result, we can assert that the set of circulant matrices can be simultaneously diagonalized using the iDFT matrix (2.14).

What about the eigenvalues of this set of matrices? Remember that we can generate the set of circulant matrices by means of the filter $\mathbf{H}(\mathbf{p}, \mathbf{S}_c)$ [cf. (2.9)], for different $\mathbf{p}$.

We can then substitute the EVD (2.16) of $\mathbf{S}_c$ in the filter definition (2.9), obtaining:

$$\begin{aligned}\mathbf{H}(\mathbf{p},\mathbf{S}_c) &= \sum_{l=0}^{N-1} p_l \left(\mathbf{F}^{-1}\,\mathrm{Diag}(\mathbf{f}_1)\mathbf{F}\right)^l \qquad (2.20)\\ &= \mathbf{F}^{-1}\left(\sum_{l=0}^{N-1} p_l\,\mathrm{Diag}(\mathbf{f}_1)^l\right)\mathbf{F}\\ &= \mathbf{F}^{-1}\left(\sum_{l=0}^{N-1} p_l\,\mathrm{Diag}(\mathbf{f}_l)\right)\mathbf{F}\\ &= \mathbf{F}^{-1}\,\mathrm{Diag}(\mathbf{F}\mathbf{p})\mathbf{F}\\ &= \mathbf{F}^{-1}\,\mathrm{Diag}(\hat{\mathbf{p}})\mathbf{F} \qquad (2.21)\end{aligned}$$

where we defined $\hat{\mathbf{p}}$ as $\hat{\mathbf{p}} := \mathbf{F}\mathbf{p}$, i.e., the DFT of the filter $\mathbf{p}$. We have then found the following: given a circulant matrix parametrized by the vector $\mathbf{p}$, its eigenvalues are given by $\hat{\mathbf{p}}$, the DFT of $\mathbf{p}$. This, leads us to the renowned convolution theorem, which is the central tenet of signal processing; but first, let us summarize our findings.

We started with the definition of circular convolution between two signals: an operation respecting the scale-shift-sum principle, where the shift can be captured by the circular shift $\mathbf{S}_c$ and the scaling by the filter coefficients $\mathbf{p}$. We have also seen how it can be implemented as a matrix-vector product, where the matrix representation of the filter $\mathbf{H}(\mathbf{p},\mathbf{S}_c)$ is circulant; as such it can be written as a polynomial of $\mathbf{S}_c$, where the coefficients of the polynomial are given by the entries of the vector $\mathbf{p}$. Since $\mathbf{S}_c$ and $\mathbf{H}(\mathbf{p},\mathbf{S}_c)$ commute, they share the same eigenvectors, which coincide with the inverse DFT matrix $\mathbf{F}^{-1}$. In terms of eigenvalues, we have found that the eigenvalues of the shift $\mathbf{S}_c$ are the N primitive roots of unity, while for $\mathbf{H}(\mathbf{p},\mathbf{S}_c)$ they are given by the DFT of the parameter $\mathbf{p}$; more in general the eigenvalues of any circulant matrix are given by the DFT of its first column[3].

2.1.3. The Convolution Theorem

From the diagonalizability of the filter $\mathbf{H}(\mathbf{p},\mathbf{S}_c)$ the convolution theorem emerges: computing the DFT of the output $\mathbf{y}$ in (2.10), and incorporating equation (2.20) into the filter representation $\mathbf{H}(\mathbf{p},\mathbf{S}_c)$, we obtain:

$$\begin{aligned}\hat{\mathbf{y}} = \mathbf{F}\mathbf{y} = \mathbf{F}\mathbf{F}^{-1}\,\mathrm{Diag}(\hat{\mathbf{p}})\mathbf{F}\mathbf{x} &= \qquad (2.22)\\ &= \mathrm{Diag}(\hat{\mathbf{p}})\hat{\mathbf{x}} \qquad (2.23)\\ &= \hat{\mathbf{p}} \odot \hat{\mathbf{x}}. \qquad (2.24)\end{aligned}$$

In other words, the DFT of the output signal $\mathbf{y}$ can be computed as a pointwise multiplication between $\hat{\mathbf{p}}$ and $\hat{\mathbf{x}}$, that is, between the DFT of the filter impulse response $\mathbf{p}$ and the DFT of the input signal $\mathbf{x}$; this is illustrated in Fig. 2.3. We conclude this section with the following observations:

[3]Indeed notice that since $\mathbf{S}_c$ is itself circulant, its eigenvalues are equal to the DFT of its first column, i.e., the DFT of the canonical vector $[0,1,0,\ldots,0]^\top$, which essentially selects the second column of the DFT matrix containing the Nth primitive roots of unity.

- In the frequency domain, a convolution is given by a pointwise multiplication between $\hat{\mathbf{p}}$ and $\hat{\mathbf{x}}$, that is, between the frequency representations of the filter impulse response $\mathbf{p}$ and of the signal $\mathbf{x}$;
- In the frequency domain, the filter $\mathbf{H}(\mathbf{p}, \mathbf{S}_c)$ can be represented as a diagonal operator $\text{Diag}(\hat{\mathbf{p}})$. As a result, its operation on a frequency signal $\hat{\mathbf{x}}$ does not involve intermixing the signal's frequency components. This characteristic facilitates the efficient application of spectral processing techniques such as filtering, windowing, and spectral estimation. The vector $\hat{\mathbf{p}}$, which corresponds to the eigenvalues of $\mathbf{H}(\mathbf{p}, \mathbf{S}_c)$, is obtained as the DFT of the filter impulse response $\hat{\mathbf{p}}$.

$$\begin{tikzcd}
\mathbf{x} \arrow[r] \arrow[d] & \boxed{\mathbf{H}(\mathbf{p}, \mathbf{S}_c)} \arrow[r] & \mathbf{y} \arrow[d] \\
\boxed{\mathbf{F}} \arrow[d] & & \boxed{\mathbf{F}} \arrow[d] \\
\hat{\mathbf{x}} \arrow[r] & \boxed{\text{Diag}(\hat{\mathbf{p}})} \arrow[r] & \hat{\mathbf{y}}
\end{tikzcd}$$

Figure 2.3: Convolution Theorem: a convolution in the discrete time domain is equivalent to a pointwise multiplication in the frequency domain. The filter coefficients $\hat{\mathbf{p}}$ in the frequency domain are given by the DFT of the impulse response $\mathbf{p}$.

2.2. SIGNAL PROCESSING ON GRAPHS

In the preceding section, we delved into concepts like shifting, convolution, and Fourier analysis concerning signals in the (discrete-)time domain. Although not explicitly emphasized, these definitions and their functional representations are inherently shaped by the structure of the signal domain, in the case of discrete-time coinciding with the natural numbers $\mathbb{N}$. To be more concrete, a notion of "order" and "precedence" is evident between successive samples x_i and x_{i+1}, where i and $i+1$ correspond to tangible time intervals, underscoring the physical interpretation associated with them.

However, we frequently come across signals that are defined in a less uniform domain. Take, for example, the high-speed rail system depicted in Fig. 2.4, illustrating a potential configuration of a railway network linking major European cities. This data type presents significant complexity. Suppose we can calculate

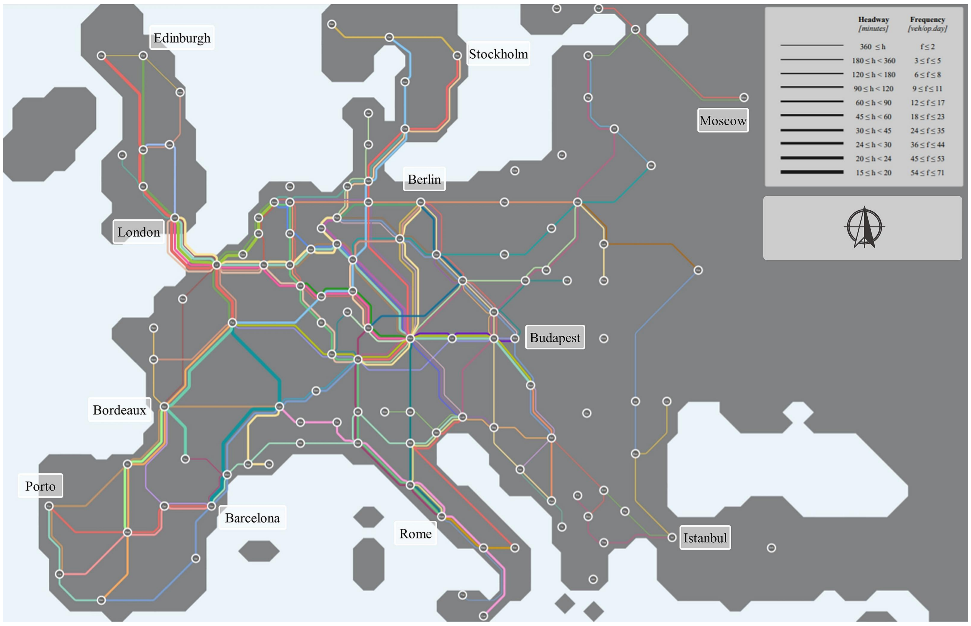

Edinburgh
Stockholm
Moscow
Berlin
London
Budapest
Bordeaux
Porto
Barcelona
Rome
Istanbul
Headway [minutes]
Frequency [veh/op.day]
360 ≤ h | f ≤ 2
180 ≤ h < 360 | 3 ≤ f ≤ 5
120 ≤ h < 180 | 6 ≤ f ≤ 8
90 ≤ h < 120 | 9 ≤ f ≤ 11
60 ≤ h < 90 | 12 ≤ f ≤ 17
45 ≤ h < 60 | 18 ≤ f ≤ 23
30 ≤ h < 45 | 24 ≤ f ≤ 35
24 ≤ h < 30 | 36 ≤ f ≤ 44
20 ≤ h < 24 | 45 ≤ f ≤ 53
15 ≤ h < 20 | 54 ≤ f ≤ 71

Figure 2.4: The European high-speed rail network plan, as envisioned in [6].

pertinent statistics, such as the number of passengers passing through a specific station on any given day of the year, for each station in the network (represented by circles in the figure). We might then gather these values into a vector $\mathbf{x} \in \mathbb{R}^N$, where N denotes the total number of stations. This prompts a question: in what sequence should we arrange these stations (and thus, which entry corresponds to which station)? Should we start to number stations from left to right, top to bottom in the map?

Evidently, there are countless potential ways to order these values. However, by doing so, we overlook another crucial aspect of the data—its underlying geometry or the inherent relationships between stations, which could be described as "connected to" relationships. Furthermore, these relationships themselves might contain numerical information, such as the distance or length between the interconnected stations. By focusing solely on the vector $\mathbf{x}$ we risk losing this geometric context. Undoubtedly, the number of passengers at a given station is influenced by its neighboring stations. This underscores the necessity for a framework that captures the intricacies of this networked data domain. We address this need by employing the concept of graphs. In what comes next, we explore how to expand the concepts of signals, filters, and transformations discussed in the preceding section, to accommodate signals defined within a networked structure.

2.2.1. Graphs

We represent the network of interest using a graph denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \omega)$. Here $\mathcal{V} = \{1, \ldots, N\}$ constitutes the set of N nodes (or vertices), which symbolize the entities or objects under scrutiny; $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ represents the set of M edges, where each edge is characterized by a tuple $e_{ij} = (i, j)$, capturing the connection between nodes i and j[4]; lastly $\omega : \mathcal{E} \rightarrow \mathbb{R}$ serves as a weight function that assigns a real number to each edge. A greater edge weight implies a more pronounced similarity or interdependency between the vertices linked by that specific edge. The categorization of a graph as directed or undirected depends on the characteristics of ω. Specifically, a graph is deemed *undirected* if $\omega(i, j) = \omega(j, i)\ \forall i, j \in \mathcal{V}$. This implies that the weight associated with the edge (i, j) is identical to that of edge (j, i). Conceptually, even though these edges are formally distinct, they are often viewed as a single, unique edge due to their identical weights. Differently, a graph is said to be *directed* when there exists a notion of orientation or directionality among the nodes i and j, and $\omega(i, j) \neq \omega(j, i)$ for some $i, j \in \mathcal{V}$. This directional aspect is evident in systems like water and road networks, where there's a distinct concept of information or flow moving from one node to another. Fig. 2.5 illustrate an undirected and a directed graph.

Graph Signals

To comprehensively study and manipulate data associated with networks, it is essential to establish a rigorous mathematical framework. We introduce the concept of a *graph signal*: a vector $\mathbf{x}$ associated to a graph $\mathcal{G}$, with each node $i \in \mathcal{V}$ having an associated value $x_i \in \mathbb{C}$; that is, $x_i : \mathcal{V} \rightarrow \mathbb{R}$. This formalism is adaptable even when nodes hold vectors rather than individual scalar values. Even tough it would be more accurate to formally represent $\mathbf{x}$ as $\mathbf{x}_{\mathcal{G}}$, to underscore its reliance on and connection to the particular network domain at hand, we will use $\mathbf{x}$ for ease of notation, while keeping in mind its essential connection to the underlying network structure. Fig. 2.6 shows an example of a graph signal.

[4]We adopt a "to-from" notation, indicating that the first node is the endpoint of the edge, while the second node serves as its origin.

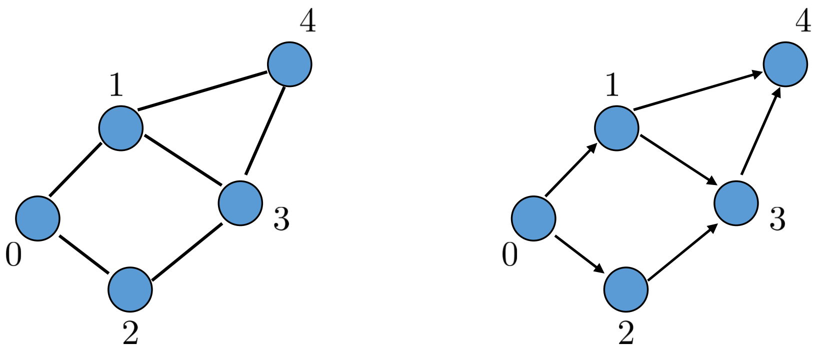


Figure 2.5: An undirected (left) and directed (right) graph composed of 5 nodes.

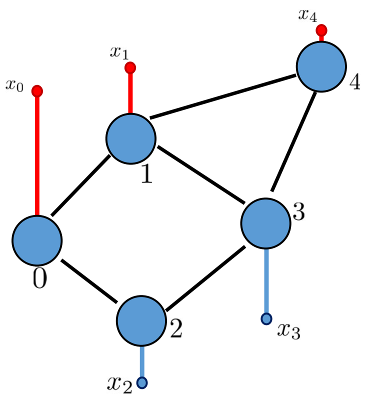


Figure 2.6: A graph signal on a graph composed of 5 nodes. For illustrative purposes, the length of the bars above each node i indicates the magnitude of the associated value x_i, while its color denotes the sign, red for positive numbers and blue for negative ones.

Graph Representation

Having introduced what constitutes a graph, we now want to formally encode it in computational friendly form. The algebraic realm seems to be a good venue for this purpose; thus, we encode a graph by means of matrix representations. While numerous matrices can represent a graph, we will focus here on those most frequently employed in the literature, namely the adjacency matrix and the Laplacian matrix. Given a graph $\mathscr{G}$ with N nodes, the *weighted adjacency matrix* associated to the graph is the square $N \times N$ matrix $\mathbf{W}$ with entry $W_{ij} = \omega(e_{ij})$ if $(i,j) \in \mathscr{E}$, otherwise $W_{ij} = 0$. In the case each edge of the graph merely indicates the presence or absence of a connection between nodes, without carrying any specific value, the matrix simplifies to the *binary adjacency matrix* $\mathbf{A}$, with $\mathbf{A} \in \{0,1\}^{N \times N}$. Here, the entry $A_{ij} = 1$ signifies the existence of an edge originating from j and ending in i, which corresponds to having a constant weight function $\omega : \mathcal{E} \to 1$. In reference to Fig. 2.5, if $\mathbf{A}$ is the binary adjacency matrix of the undirected graph, we have that $A_{32} = A_{23} = 1$, indicating the presence of the edges $(2,3)$ and $(3,2)$; in general for an undirected graph we have $\mathbf{A} = \mathbf{A}^\top$. For the directed graph we have $A_{32} = 1$ and$A_{23} = 0$, since we only have an edge starting from node 2 and ending at node 3. For directed graphs the adjacency matrix is, in general, not symmetric.

Now, let us assume to have a graph whose binary adjacency matrix $\mathbf{A}$ is given by $\mathbf{A} = \mathbf{S}_c$, i.e., it coincides with the circular shift matrix [cf. (2.4)]. This graph would be as the one in Fig. 2.7. That is, the shift operator used to model the circular

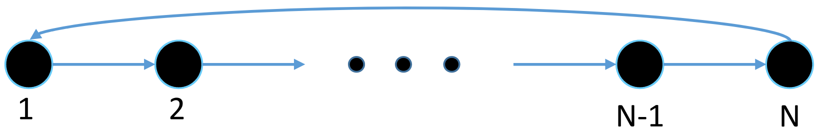


Figure 2.7: The domain of a discrete time signal can be thought as a (directed) cyclic graph.

convolution in discrete-time signal processing can be viewed as a (circular) graph capturing the relationship "precedes" between different nodes, each one representing consecutive time instants. We informally say that $\mathbf{S}_c$ "models the time domain". Under these considerations, we can view $\mathbf{S}_c$ both as an operator acting on signals (to perform the shifting operation), and as the domain of such signals (to model the underlying network domain).

Another related graph matrix representation is the *Laplacian matrix*, which is defined for undirected graphs as:

$$\mathbf{L} = \mathbf{D} - \mathbf{W}, \tag{2.25}$$

where $\mathbf{D} := \mathrm{Diag}([d_0, \ldots, d_{N-1}])$ is the *degree matrix* of the graph, a diagonal matrix where each element d_i is given by the sum of the edge weights incident to node i. When the graph is unweighted, so that $\mathbf{W}$ coincides with the binary adjacency matrix $\mathbf{A}$, d_i simply counts the number of neighbors of node i.

As we will see soon, the Laplacian matrix plays a very important role in graph theory and for the Fourier analysis of graph signals; but before delving into that, we conclude this section by introducing two other matrices matrices often encountered in literature to represent graphs. These are:

- the normalized adjacency matrix $\tilde{\mathbf{W}} = \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$, whose eigenvalues are located in the interval $[-1, 1]$;
- the normalized Laplacian matrix $\tilde{\mathbf{L}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$ whose eigenvalues are located in the interval $[0, 2]$.

They are frequently favored over other matrix representations because raising them to powers does not result in exploding eigenvalues.

The Graph Shift Operator

We've explored various matrix representations for encoding a graph. But which one should we use? The answer to this question is, perhaps unfortunately, not unique. The choice of matrix representation depends on the specific application – a somewhat nuanced decision. While there exists a one-to-one correspondence between a graph's description and each one of its matrix representations, these matrices, despite conveying the same fundamental structural information, are distinct algebraic entities. Consequently, they operate as fundamentally different linear operators; indeed, in the next section, we will observe that these matrices not only serve as descriptive tools for the graph but can also be interpreted as linear operators acting on the graph signals defined on them.

To create a layer of abstraction, without particularizing our discussion to one single matrix representation, we will denote with $\mathbf{S}$ the matrix representation of the graph under analysis, which can be any of the introduced matrices. We will call this matrix the *graph shift operator* (GSO), since it assumes a similar role as the circular shift operator $\mathbf{S}_c$; that is, it defines the notion of locality and shift on the graph context. Formally, given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \omega)$, the GSO is any matrix $\mathbf{S}$ such that $S_{ij} = 0$ for $i \neq j$ if $e_{ij} \notin \mathcal{E}$. As such, matrix $\mathbf{S}$ captures the sparsity pattern of the network [7, 8].

A notion of frequency

Frequency analysis stands as a foundational principle in signal processing, providing the ability to break down a signal into constituent components with distinct rates of variation, as exemplified in (2.19) for discrete scenarios. In that context, we briefly explored the merits of this signal representation, emphasizing its utility in comprehending signal behavior and executing filtering operations. The components employed in that context were the eigenvectors of the circular shift $\mathbf{S}_c$, oscillating modes representing a discretized form of complex exponentials. Was the selection of this basis a pure coincidence, or does it hold unique properties that make it particularly advantageous among an infinite array of possible bases for expressing signals? A possible answer resides on the study of one of the most important mathematical objects in physics and partial differential equations: the *Laplace operator* or *Laplacian*.

Laplacian The Laplace operator $\boldsymbol{\Delta}$ on $\mathbb{R}^N$ is a differential operator defined as the divergence of the gradient of a scalar function; that is, given a twice-differentiable function $f : \mathbb{R}^N \to \mathbb{R}$, the Laplacian of $f(\cdot)$ is the real-valued function:

$$\boldsymbol{\Delta} f(\mathbf{x}) = \frac{\partial^2 f(\mathbf{x})}{\partial x_1^2} + \frac{\partial^2 f(\mathbf{x})}{\partial x_2^2} + \cdots + \frac{\partial^2 f(\mathbf{x})}{\partial x_N^2} \tag{2.26}$$

for $\mathbf{x} \in \operatorname{dom} f$. That is, it is given by the sum of second partial derivatives of the function $f(\cdot)$ with respect to each independent variable evaluated at $\mathbf{x}$; equivalently, it corresponds to the trace of the Hessian matrix of f at $\mathbf{x}$. Thus, it quantifies the local spatial variation or curvature of a function around a specific point $\mathbf{x}$.

The significance of the Laplace operator lies in its relationships with the domain where it is defined. Notably, it commutes with all the isometries of its domain, such as translation and rotation. Formally we say that it exhibits *equivariance* under such isometries[5]. Intriguingly, it can be demonstrated that all differential operators that commute with Euclidean isometries are essentially polynomials in the Laplace operator $\boldsymbol{\Delta}$. This implies that the algebra of differential operators with constant coefficients, which commute with all Euclidean isometries, is isomorphic to the polynomial algebra generated by the Laplace operator [9]. A detour on abstract algebra here might help, but it is not the scope of this introduction, and we refer the interested reader to [10].

This commutation property and the polynomial term should ring a bell. We have previously seen how a special class of filters of interest, linear shift invariant filters, can be represented as circulant matrices. We have also seen how any circulant matrix can be written as a polynomial of the circular shift operator $\mathbf{S}_c$. The shift $\mathbf{S}_c$ generates the commutative algebra of LTI filters, which is a polynomial algebra [11]. Thus, the Laplace operator and the shift seem to have something in common. The shift can be seen as an approximation of the Laplace operator in Euclidean domains. Indeed, it turns out that the eigendecomposition of the Laplace operator also leads to sinusoidal functions (in the continuous domain this time). For instance, for the one-dimensional Laplace operator it holds:

$$-\boldsymbol{\Delta} e^{j2\pi\xi x} = -\frac{\partial^2 e^{j2\pi\xi x}}{\partial x^2} = (2\pi\xi)^2 e^{j2\pi\xi x}. \tag{2.27}$$

In other words, the complex exponentials are eigenfunctions of the Laplace operator with associated eigenvalue $(2\pi\xi)^2$. In classical Fourier analysis, the term $(2\pi\xi)^2$ carries the notion of *frequency*: for ξ close to zero (low frequency), the associated eigenfunction $e^{j2\pi\xi x}$ is smooth, i.e., it exhibits a low-oscillating behavior; increasing ξ renders such eigenfunctions increasingly oscillating. Using these eigenfunctions (or a discretization thereof) as a basis to express signals constitutes the pillar of frequency-based signal processing.

Does the Laplacian matrix introduced for representing graphs incorporate a concept of frequency analogous to that of the Laplace operator? In other words, do the eigenvectors of the Laplacian matrix constitute a basis that exhibits a notion of variation analogous to that of the Laplace operator in continuous (Euclidean) domains? If that is the case, we might use this basis to represent signals on graphs, and we would have a signal decomposition similar to that of classical signal processing. The answer is affirmative as we will show now.

First, notice that $\mathbf{L}$ is a symmetric matrix, and as such, for the spectral theorem, it admits the eigenvalue decomposition:

$$\mathbf{L} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^\top \tag{2.28}$$

where $\mathbf{V} = [\mathbf{v}_0, \ldots, \mathbf{v}_{N-1}]$ is a square matrix of orthogonal eigenvectors, i.e., $\mathbf{V}^\top\mathbf{V} = \mathbf{V}\mathbf{V}^\top = \mathbf{I}_N$ and $\boldsymbol{\Lambda} = \mathrm{Diag}(\boldsymbol{\lambda})$ is the diagonal matrix of real eigenvalues $\boldsymbol{\lambda} = [\lambda_0, \lambda_1, \ldots, \lambda_{N-1}]^\top$; specifically we have $0 = \lambda_0 \leq \lambda_1 \leq \ldots \leq \lambda_{N-1}$.

[5]This property is in many texts confusingly called as invariance.

We show now that different eigenvectors of the Laplacian showcase diverse levels of variation across their respective domains. What does the variation mean in the context of graphs? It is pertinent to recall that the Laplace operator $\mathbf{\Delta}$ [cf. (2.26)] acting on a function f evaluated at point $\mathbf{x}$ captures the *smoothness* of f around that point. In the context of graph signals, applying the Laplacian $\mathbf{L}$ to a signal $\mathbf{x}$ yields a new graph signal $\mathbf{y}$, where the ith component y_i at the ith node is given by:

$$y_i = d_i x_i - \sum_{j \in \mathcal{N}(i)} W_{ij} x_j \tag{2.29}$$

where we recall that d_i represents the degree of node i. This represents a quantification of the "curvature" in the vicinity of node i. It is important to note that a y_i value of 0 does not necessarily imply $x_i = x_j$ for all j. Furthermore, our interest typically extends to evaluating the overall smoothness of the graph signal. Therefore, a metric for assessing this global smoothness is provided by the Laplacian quadratic form, which, for a given graph signal $\mathbf{x}$ and a given Laplacian $\mathbf{L}$, is defined as:

$$\mathrm{LQ}(\mathbf{x}, \mathbf{L}) := \mathbf{x}^\top \mathbf{L} \mathbf{x} = \sum_{(i,j) \in \mathcal{E}} W_{ij} (x_i - x_j)^2 \tag{2.30}$$

It measures the squared value difference among adjacent nodes of the graph. It is clear how $\mathrm{LQ}(\mathbf{x}, \mathbf{L}) = 0$ if and only if $x_i = x_j \forall i, j \in \mathcal{V}$, i.e., if the graph signal is a constant vector. The higher the weight W_{ij} connecting nodes i and j, the bigger the impact of the difference between the node values.

Notice that the eigenvectors $\{\mathbf{v}_i\}_{i=0}^{N-1}$ can be viewed as signals defined over the graph Laplacian. Thus, we can assess their level of variation according to (2.30). It is easy to show that:

$$\mathrm{LQ}(\mathbf{v}_i, \mathbf{L}) := \mathbf{v}_i^\top \mathbf{L} \mathbf{v}_i = \lambda_i \mathbf{v}_i^\top \mathbf{v}_i = \lambda_i \tag{2.31}$$

that is, each eigenvector $\mathbf{v}_i$ exhibits a graph variation equal to its associated eigenvalue λ_i. Because we have $\lambda_i \leq \lambda_j$ for $i < j$ it follows that eigenvectors with smaller eigenvalues are smoother with respect to the underlying domain than eigenvectors associated to higher eigenvalues. Fig. 2.8 show the 1st, the 2nd and the 20th eigenvectors of the Laplacian associated to the Minnesota road graph; we can see how the first eigenvector is the constant signal (associated to $\lambda_0 = 0$), while the 2nd and the 20th exhibit higher variability on the graph, equal to their associated eigenvalues, in this case $\lambda_1 = 8.4 \times 10^{-4}$ and $\lambda_{19} = 1.8 \times 10^{-2}$, respectively. It does not seem then too exotic to start thinking about a Fourier-like theory for data defined on graphs, and to think about the eigenvalues as *graph frequencies*. We will talk more about this in the next section, but first, we spend two more words on some Laplacian properties.

From the definition of $\mathbf{L}$ [cf. (2.25)], we have that each row (and column) of $\mathbf{L}$ sums to 0, i.e., $\mathbf{L1} = \mathbf{0}$. This means that any constant vector is always an eigenvector of $\mathbf{L}$ with associated eigenvalue 0. This makes sense: a constant vector is a signal which does not exhibit any variation on the graph, and as such $\mathrm{LQ}(\cdot, \mathbf{L}) = 0$. Moreover, it can be shown that the multiplicity of the eigenvalue 0, that is, the number of

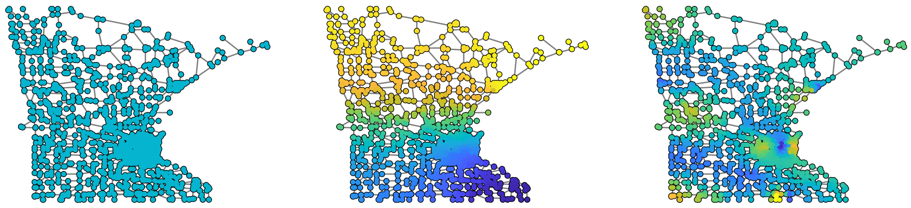


Figure 2.8: Minnesota road graph: 1st, 2nd and 20th eigenvectors of the graph Laplacian as color-coded signals: darker colors indicate lower values.

eigenvectors with associated eigenvalue equal to 0, reveals the number of connected components of the graph; thus by simple inspection of the EVD we can determine whether the graph is connected or not.

2.3. Spectral Analysis and Graph Filtering

We have observed how the eigenvectors of the Laplacian matrix carry a concept of variation akin to that of the Laplace operator and circular shift. Eigenvectors corresponding to higher eigenvalues exhibit greater variability when interpreted as signals on the graph. Similar to the circulant shift $\mathbf{S}_c$, which could be interpreted as the adjacency matrix of a graph, it is possible to show that also the eigenvectors of $\mathbf{W}$ can be understood as frequency modes. Without any loss of generality, for the rest of the chapter, we consider a general GSO $\mathbf{S}$, where we assume it admits the eigenvalue decomposition:

$$\mathbf{S} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^{-1} \tag{2.32}$$

where $\mathbf{V}$ is the square $N \times N$ matrix whose ith column is the eigenvector $\mathbf{v}_i$ of $\mathbf{S}$, and $\boldsymbol{\Lambda} = \text{Diag}(\boldsymbol{\lambda})$ is the diagonal matrix containing the eigenvalues $\boldsymbol{\lambda} = [\lambda_0, \ldots, \lambda_{N-1}]^\top$. For undirected graphs, since $\mathbf{S}$ is real symmetric, we can choose the eigenvectors to be real and orthonormal, such that $\mathbf{V}^{-1} = \mathbf{V}^\top$.

Definition 2.3.1. *The projection of a graph signal $\mathbf{x}$ onto the the eigenvectors $\mathbf{V}$ of the GSO $\mathbf{S}$, denoted with $\hat{\mathbf{x}}$, is called graph Fourier Transform (GFT). Mathematically, it is obtained with the following operation:*

$$\hat{\mathbf{x}} = \mathbf{V}^{-1}\mathbf{x}. \tag{2.33}$$

We will often refer to $\hat{\mathbf{x}}$ as the GFT signal, which represents the spectral representation of the graph signal $\mathbf{x}$ when expressed in the basis $\mathbf{V}$. Since $\mathbf{V}$ is invertible, $\mathbf{x}$ can be synthesized by means of the inverse GFT (iGFT), defined as:

$$\mathbf{x} = \mathbf{V}\hat{\mathbf{x}} = \sum_{n=0}^{N-1} \hat{x}_n \mathbf{v}_n. \tag{2.34}$$

That is, a graph signal is expressed as the linear combination of the GSO eigenvectors $\mathbf{v}_i$, where the weight is given by the spectral coefficient $\hat{x}_i$ of the signal at that particular eigenvalue λ_i. The inverse DFT in (2.19) can be seen as a particular case of (2.34) when $\mathbf{S}$ is a circulant matrix. Fig. 2.9(left) shows a graph signal $\mathbf{x}$ which is given by the linear combination of the first 400 eigenvectors; the weights of such linear combination are shown in 2.9(right). As we can notice, many of the GFT

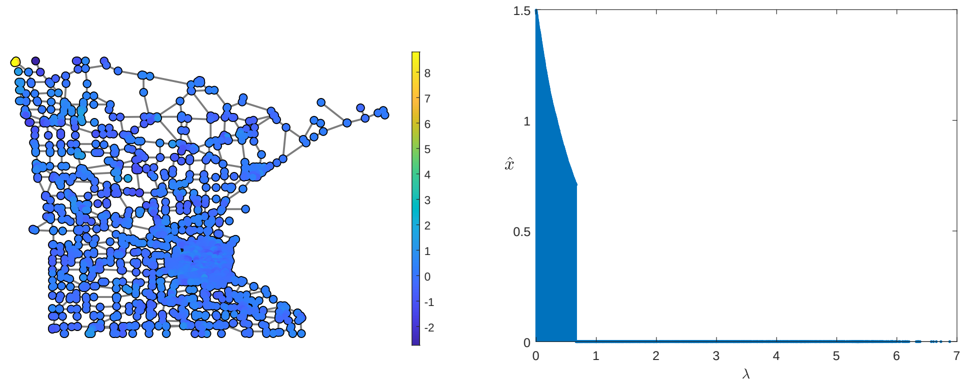


Figure 2.9: (Left) A graph signal $\mathbf{x}$ and (right) its GFT representation $\hat{\mathbf{x}}$. The signal $\mathbf{x}$ is given by a linear combination of the first 400 eigenvectors of the GSO $\mathbf{S}$, thus making it a low-pass signal.

coefficients are zeros: in this case we say that the graph signal $\mathbf{x}$ is *sparse* in the frequency domain. Sparse signals find use in numerous applications, such as data compression and signal denoising.

In section 2.1.1, we discussed how the set of shift-invariant filters, responsible for performing convolution operations, is determined by polynomials involving shifts; that is, every element $\mathbf{H}(\mathbf{p}, \mathbf{S}_c)$ of the set can be generated by varying the polynomial coefficients collected in $\mathbf{p}$. As expected, this concept also applies within the context of graphs for a general $\mathbf{S}$. We will reach this conclusion through a different path, notably stemming from the frequency domain and then returning to the vertex domain. We exemplify this with a practical demonstration. Keep in mind that since $\mathbf{S}$ is (in general) not circulant, also the matrix representation of the filter will not be circulant.

Spectral Filtering

Let us assume that the graph signal $\mathbf{x}$ of Fig. 2.9 undergoes some noisy process, i.e., $\mathbf{x} = \mathbf{x}_0 + \mathbf{n}$, where $\mathbf{x}_0$ is the "clean" component of the signal and $\mathbf{n} \in \mathbb{R}^N$ is noise (so $\mathbf{x}_0$ is essentially the $\mathbf{x}$ of Fig. 2.9; this for notational simplicity). Fig. 2.10 shows the GFT $\hat{\mathbf{x}}$ for the noisy graph signal $\mathbf{x}$. We can directly observe how the noise is responsible for the spectral content in the middle band, i.e., with frequency components $2.7 \leq \lambda \leq 4$. We would like to process such signal to obtain its clean version $\mathbf{x}_0$ back.

If we have prior knowledge that the clean signal $\mathbf{x}_0$ is low-pass, we might create a spectral filter $\hat{\mathbf{p}}$ acting on the GFT signal $\hat{\mathbf{x}}$ such that only the low-spectral content of the signal is maintained. Let us call $\hat{\mathbf{y}}$ the filtered version of $\hat{\mathbf{x}}$; then we can perform this *spectral filtering* as:

$$\hat{\mathbf{y}} = \hat{\mathbf{p}} \odot \hat{\mathbf{x}} = \mathrm{Diag}(\hat{\mathbf{p}})\hat{\mathbf{x}} \tag{2.35}$$

where $\mathrm{Diag}(\hat{\mathbf{p}})$ is the matrix representation of the filter in the frequency domain and $\hat{\mathbf{p}}$ is called *graph filter frequency response*; we will soon see how we can make $\hat{\mathbf{p}}$ parametric. Fig. 2.10 shows, together with the noisy GFT signal $\hat{\mathbf{x}}$, a low-pass graph frequency response $\hat{\mathbf{p}}$ which removes the noise spectral content.

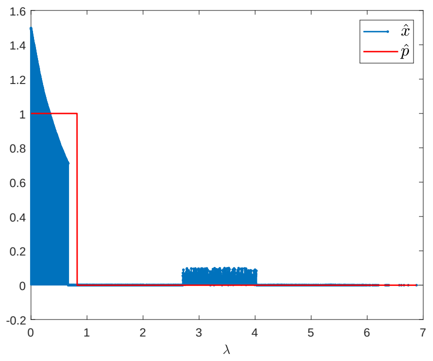


Figure 2.10: GFT $\hat{\mathbf{x}}$ of a noisy graph signal $\mathbf{x}$ and graph frequency response $\hat{\mathbf{p}}$.

In other words, (2.35) shows how spectral filtering is a frequency-wise multiplication of each graph signal coefficient $\hat{x}_i$ and the associated graph filter response $\hat{p}_i$ which attenuates or amplifies $\hat{x}_i$.

How does this spectral operation translate in the vertex domain? By simply computing the iGFT of $\hat{\mathbf{y}}$, we have:

$$\mathbf{y} = \mathbf{V}\hat{\mathbf{y}} = \mathbf{V}\mathrm{Diag}(\hat{\mathbf{p}})\mathbf{V}^{-1}\mathbf{x} = \mathbf{H}\mathbf{x} \tag{2.36}$$

where $\mathbf{H} := \mathbf{V}\mathrm{Diag}(\hat{\mathbf{p}})\mathbf{V}^{-1}$ is a so-called *graph filter* (GF), a linear operator acting on the graph signal $\mathbf{x}$ in the vertex domain. The filtering operation (2.36) in the vertex domain can then be seen as three consecutive operations acting on the signal $\mathbf{x}$: first *i)* the GFT $\hat{\mathbf{x}}$ of $\mathbf{x}$ is obtained as $\hat{\mathbf{x}} = \mathbf{V}^{-1}\mathbf{x}$; then *ii)* a pointwise multiplication between $\hat{\mathbf{x}}$ and the filter frequency response $\hat{\mathbf{p}}$ is performed, obtaining a new GFT signal $\hat{\mathbf{y}}$; and finally *iii)* the new GFT signal $\hat{\mathbf{y}}$ is brought back to the vertex domain by means of the iGFT to get $\mathbf{y}$, that is, $\mathbf{y} = \mathbf{V}\hat{\mathbf{y}}$. The design of the graph frequency response $\hat{\mathbf{p}}$ is application-dependent and an entire branch of literature is devoted to it; for a thorough discussion on graph filter design and applications, we refer the reader to [12]. As a concluding note, the matrix-vector multiplication in (2.36) requires $\mathcal{O}(N^2)$ operations, since matrix $\mathbf{H}$ is full in general; we will see how this cost can be lowered by exploiting the graph structure in the next section.

2.3.1. Graph Convolution

Consider again (2.36). It is visible how $\mathbf{H}$ is diagonalized by $\mathbf{V}$, i.e., by the eigenvectors of the GSO $\mathbf{S}$. Thus the graph filter $\mathbf{H}(\mathbf{p})$ commutes with the GSO $\mathbf{S}$:

$$\mathbf{HS} = \mathbf{SH}. \tag{2.37}$$

Graph filters with this property are termed *shift-invariant* (or node-invariant in this context) and they are the graph counterpart to the time-invariant filters defined in Section (2.1.1), where the distinctive feature there was that the matrix representation of the filter was circulant. This is not the case for a graph filter $\mathbf{H}$, in general.

We have also seen how, upon the definition of a shift, the class of shift invariant filters is given by matrix polynomials in the shift. Thus it seems natural to parametrize the class of shift-invariant graph filters as polynomials of the GSO $\mathbf{S}$ similar to (2.9); formally, a graph filter of order $L-1$ is the matrix polynomial:

$$\mathbf{H}(\mathbf{p},\mathbf{S}) = \sum_{l-0}^{L-1} p_l \mathbf{S}^l \tag{2.38}$$

where $\mathbf{p} := [p_0,\dots,p_{L-1}]^\top$ collects the L graph filter coefficients, also called graph *filter taps*. Notice once again how in (2.38) we made explicit the dependence of $\mathbf{H}$ in the filter taps $\mathbf{p}$ and $\mathbf{S}$.

The application of $\mathbf{H}(\mathbf{p},\mathbf{S})$ to a graph signal $\mathbf{x}$ as in (2.36), is often called *graph convolution*, as it respects the scale-shift-sum principle of the classical convolution. To see this, it suffices to expand the terms of the sum in (2.38) to rewrite:

$$\mathbf{y} = \mathbf{H}(\mathbf{p},\mathbf{S})\mathbf{x} = p_0\mathbf{x} + p_1\mathbf{S}\mathbf{x} + \ldots + p_{L-1}\mathbf{S}^{L-1}\mathbf{x} \tag{2.39}$$

$$= p_0\mathbf{x} + p_1\mathbf{x}^{(1)} + \ldots + p_{L-1}\mathbf{x}^{(L-1)} \tag{2.40}$$

where we remind that $\mathbf{x}^{(l)}$ indicates the l-shifted version of the signal $\mathbf{x}$. See also Fig. 2.11 for a visualization of the graph convolution as a shift-register. This time, due to the sparsity pattern of $\mathbf{S}$, the computational complexity of the filtering operation reduces to $\mathcal{O}(L|\mathcal{E}|)$, differently from (2.36) which was $\mathcal{O}(N^2)$.

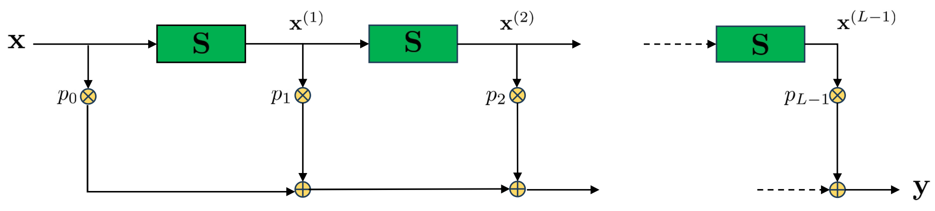


Figure 2.11: Graph convolution as a shift register: each shifted version $\mathbf{x}^{(l)}$ of the signal $\mathbf{x}$ is multiplied with a scalar filter coefficient p_l and then added together.

Under the parameterization (2.38), it is natural to ask: how is the graph filter frequency response $\hat{\mathbf{p}}$ related to $\mathbf{p}$? It easy to show:

$$\mathbf{H}(\mathbf{p},\mathbf{S}) = \sum_{l-0}^{L-1} p_l \mathbf{S}^l = \mathbf{V}(\sum_{l-0}^{L-1} p_l \mathbf{\Lambda}^l)\mathbf{V}^{-1}\mathbf{V}(\sum_{l-0}^{L-1} p_l \mathbf{\Lambda}^l)\mathbf{V}^{-1} \tag{2.41}$$

$$= \mathbf{V}\mathrm{Diag}(\sum_{l-0}^{L-1} p_l \boldsymbol{\lambda}^l))\mathbf{V}^{-1} \tag{2.42}$$

$$= \mathbf{V}\mathrm{Diag}(\mathbf{\Psi}\mathbf{p})\mathbf{V}^{-1} \tag{2.43}$$

where we defined $\mathbf{\Psi}$ as the $N \times L$ Vandermonde matrix:

$$\mathbf{\Psi} = \begin{bmatrix} 1 & \lambda_0 & \lambda_0^2 & \cdots & \lambda_0^{L-1} \\ 1 & \lambda_1 & \lambda_1^2 & \cdots & \lambda_1^{L-1} \\ \vdots & \vdots & \vdots & \vdots & \\ 1 & \lambda_{N-1} & \lambda_{N-1}^2 & \cdots & \lambda_{N-1}^{L-1} \end{bmatrix}. \tag{2.44}$$

In other words, the graph filter frequency response $\hat{\mathbf{p}}$ of a shift-invariant filter can be expressed as a polynomial in the eigenvalues $\boldsymbol{\lambda}$; explicitly:

$$\underbrace{\begin{bmatrix} \hat{p}(\lambda_0) \\ \hat{p}(\lambda_1) \\ \vdots \\ \hat{p}(\lambda_{N-1}) \end{bmatrix}}_{\hat{\mathbf{p}}} = \underbrace{\begin{bmatrix} 1 & \lambda_0 & \lambda_0^2 & \cdots & \lambda_0^{L-1} \\ 1 & \lambda_1 & \lambda_1^2 & \cdots & \lambda_1^{L-1} \\ \vdots & \vdots & \cdots & \vdots & \\ 1 & \lambda_{N-1} & \lambda_{N-1}^2 & \cdots & \lambda_{N-1}^{L-1} \end{bmatrix}}_{\mathbf{\Psi}} \underbrace{\begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ p_{L-1} \end{bmatrix}}_{\mathbf{p}} \tag{2.45}$$

Graph Filter Design. The system of equations in (2.45) is fundamental in the realm of graph filter design, as discussed in [12]. The primary objective of this design approach is to address the following inquiry: given a desired filter frequency response $\hat{\mathbf{p}}$ and a GSO $\mathbf{S}$ with eigenvalues $\boldsymbol{\lambda}$, can we identify a polynomial in λ that effectively interpolates the set of points $\{(\lambda_n, \hat{p}_n)\}_{n=0}^{N-1}$? Under the assumption that $\lambda_i \neq \lambda_j; \forall i, j \in \mathcal{V}$, implying that all GSO eigenvalues are distinct, a polynomial of degree at most $N-1$ can precisely interpolate these N points. Algebraically speaking, achieving a perfect fit is feasible as long as $\hat{\mathbf{p}} \in \mathscr{C}(\mathbf{\Psi})$, meaning that the graph frequency response $\hat{\mathbf{p}}$ lies within the column space of $\mathbf{\Psi}$. The dimension of this space determines the minimum filter order required for a perfect interpolation.

GRAPH CONVOLUTION THEOREM

The findings of the previous section are diagrammatically summarized in Fig. 2.12, representing the graph convolution theorem. It is the counterpart of the convolution theorem for time signals introduced in Section 2.1.3; also in this case the theorem emerges as a consequence of the joint diagonalization of the (graph) filter and the (graph) shift operator $\mathbf{S}$.

We can thus draw the following observations:

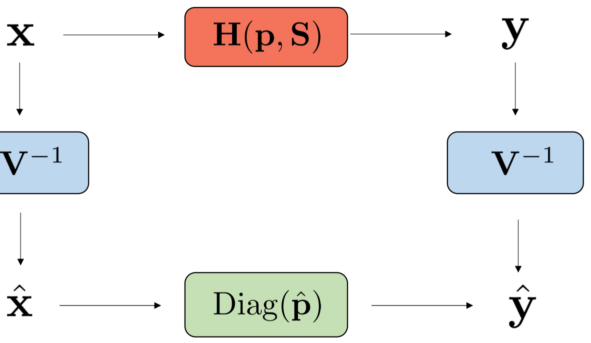


Figure 2.12: Graph Convolution Theorem: a graph convolution in the vertex domain is equivalent to a pointwise multiplication in the frequency domain. The filter coefficients in the frequency domain are expressed as a polynomial of the eigenvalues of the vertex domain.

- In the frequency domain, a graph convolution is given by a pointwise multiplication of the GF frequency response $\hat{\mathbf{p}}$ and the GFT $\hat{\mathbf{x}}$ of the graph signal $\mathbf{x}$;
- In the frequency domain, the filter $\mathbf{H}(\mathbf{p},\mathbf{S})$ is a diagonal operator which operates without mixing the signal frequency components; the elements of such diagonal matrix, which also are the eigenvalues of $\mathbf{H}(\mathbf{p},\mathbf{S})$, are given by the product of the Vandermonde matrix of eigenvalues $\boldsymbol{\Psi}$ and the graph filter taps $\mathbf{p}$. In the case of a circulant matrix, this boils down to the DFT of the filter taps $\mathbf{p}$.

Let us then compare these observations with the ones related to the standard convolution theorem at the end of Section 2.1.3. Although there are many similarities between the convolution theorem for time signals and graph signals, there are also striking differences. For time signals, the matrix responsible to transform the signal $\mathbf{x}$ and the filter taps $\mathbf{p}$ into their frequency domain representation, to obtain $\hat{\mathbf{x}}$ and $\hat{\mathbf{p}}$ respectively, is the same, namely the DFT matrix $\mathbf{F}$. This is what makes somehow the DFT matrix special: it is an eigenvector matrix made of the eigenvalues associated to such eigenvectors. This property does not hold anymore for a general graph domain. In this case, the matrix responsible to transform the signal $\mathbf{x}$ to the frequency domain representation is the GFT matrix $\mathbf{V}^{-1}$, given by the eigenvectors of the GSO $\mathbf{S}$, while the matrix responsible to transform $\mathbf{p}$ to the frequency domain representation $\hat{\mathbf{p}}$ is the Vandermonde matrix $\boldsymbol{\Psi}$ of eigenvalues [cf. (2.45)].

2.3.2. Node-Variant Graph Filtering

We wrap up this section by introducing a category of graph filters that will play a pivotal role in Chapter 6, specifically the category of shift-variant filters. This category enables a node-specific weighting scheme, thus often denoted as *node-variant* graph filters (NV-GFs) [13], representing an expansion of time-variant filters from classical signal processing to the realm of graphs. Within the realm of NV-GFs, we discern two variants, namely *type-I* and *type-II*, each defined in the context of a given graph $\mathbf{S}$ and a fixed order $L-1$, as follows:

$$\mathbf{H}_I(\mathbf{P},\mathbf{S}) = \sum_{l=0}^{L-1} \mathrm{Diag}(\mathbf{p}_l)\mathbf{S}^l, \tag{2.46}$$

$$\mathbf{H}_{II}(\mathbf{P},\mathbf{S}) = \sum_{l=0}^{L-1} \mathbf{S}^l\,\mathrm{Diag}(\mathbf{p}_l), \tag{2.47}$$

where $\mathbf{P} \in \mathbb{C}^{N\times L}$ collects the filter coefficients $\mathbf{P} = [\mathbf{p}_0,\ldots,\mathbf{p}_{L-1}]$ with $\mathbf{p}_l := [p_{l1},\ldots,p_{lN}]^\top \in \mathbb{C}^N$ the l-th hop filter tap vector. The application of a NV-GF on a signal $\mathbf{x}$ to obtain a new signal $\mathbf{y}$ will be referred to as *node-variant graph convolution*.

Even though NV-GFs can implement a broader class of linear operators, they have drawbacks. Perhaps the most important one, is that they do not commute with the GSO $\mathbf{S}$ (in general), and as such are not jointly diagonalizable. Thus, the spectral analysis of graph signals, and the graph convolution theorem, fundamental for node-invariant GFs, are lost. At least, in the form we have seen so far. Chapter 6 will spark new lights on it.

2.4. LEARNING GRAPHS FROM DATA

The remainder of this chapter serves as a gentle introduction to the problem of learning graphs from data, commonly referred to as *graph topology inference*, *network topology identification*, or *graph learning*; we will use these three terms interchangeably. While the roots of this problem can be traced back to statistical literature, particularly in the context of hypothesis testing, it has recently garnered renewed attention, due to the rich and explainable structure provided by network configurations, as well as the proliferation of graph architectures such as graph filters [14] and graph neural networks [15], which necessitate a graph for their operations.

For a traditional introduction to network topology identification from a statistical viewpoint, we recommend the book [16]; while from the perspective of graph signal processing, we suggest exploring the detailed overview papers [17–19]. Within this chapter, after illustrating and formally introducing the graph learning problem, we present three common learning problems to provide a concrete understanding of their implications, since they will also appear in Chapter 4.

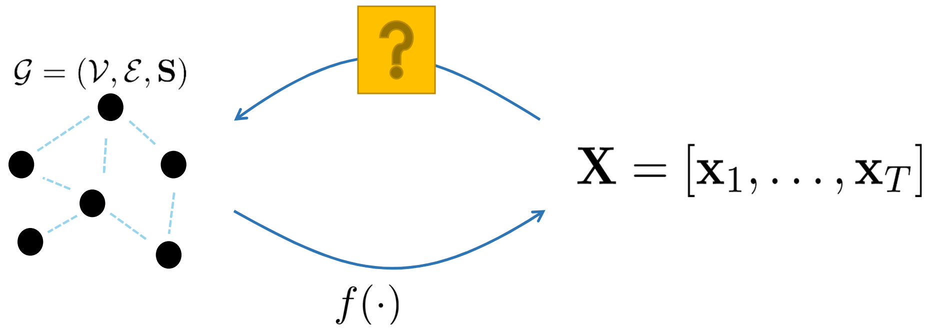

$\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{S})$
?
$\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_T]$
$f(\cdot)$

Figure 2.13: A network process $f(\cdot)$ generates a series of vectors $\mathbf{x}_1,\ldots,\mathbf{x}_T$. The goal is to learn the graph topology encoded by $\mathbf{S}$ through the knowledge of the data $\{\mathbf{x}_t\}$ and prior information on the generating process $f(\cdot)$.

GRAPH TOPOLOGY INFERENCE

Consider the following scenario: we have a collection of T N-dimensional vectors, denoted as $\mathbf{x}_1,\ldots,\mathbf{x}_T$, generated by a graph-dependent process $f(\mathcal{G})$, where the underlying graph $\mathcal{G} = (\mathcal{V},\mathcal{E},\mathbf{S})$ is *unknown*. Here, the term "unknown" signifies that the edge set $\mathcal{E}$ is not observed, while the vertex set $\mathcal{V}$ is known. In this context, each vector $\{\mathbf{x}_t\}_{t=1}^T$ can be regarded as a graph signal on the underlying latent graph $\mathcal{G}$.

Let us redirect our focus to the function $f(\cdot)$: what does it mean for it to be graph-dependent? It signifies that the network's structure, captured by the GSO $\mathbf{S}$, influences the potential interactions among the entities and, consequently, the observed values within each graph signal $\mathbf{x}_t$. For example, $f(\cdot)$ could represent the function "the value at a node equals the average value of its neighbors". Alternatively, in a more practical scenario, the function might model the dynamics of disease transmission within a social network, where individuals with close connections are more susceptible to infection, particularly if they share strong social ties.

The issue arises from the fact that while we may have some insight into how $f(\cdot)$ operates on the graph $\mathscr{G}$, the structure of $\mathscr{G}$ is unknown. The idea is to leverage the observed data to infer such structure. Thus, a loose formulation for the graph learning problem could be as follows:

> Given a collection of graph signals $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_T]$ and prior information on $f(\mathscr{G})$, learn a graph $\mathscr{G}$.

This statement prompts several questions. What criteria render one graph superior to another for interpreting the observed data? Also, should we learn *a* graph $\mathscr{G}$ or *the* graph $\mathscr{G}$? The statement lacks indications of optimality or suggestions for assessing quality. This is where mathematical modeling, the traditional approach in signal processing, together with an empirical risk minimization formulation, proves invaluable. The essence lies in making assumptions about the nature of $f(\cdot)$, typically employing a parametric approach, and the objective is to infer the underlying parameters. In this context, the primary parameter of interest is the graph $\mathscr{G}$, or equivalently, the shift operator $\mathbf{S}$, along with other parameters $\boldsymbol{\theta}$ that potentially govern the dynamics generating the data. That is, we make the (signal) modeling assumption:

$$\mathbf{X} \sim f(\mathbf{S}; \boldsymbol{\theta}) \tag{2.48}$$

where $\boldsymbol{\theta}$ captures any additional parameters of $f(\cdot)$ which are not encoded in the graph itself. Different functions $f(\cdot)$, and hence different hypotheses on the graph-data coupling, lead to different graph learning problems.

In this view, a more formal graph learning problem would read as:

> Given a collection of graph signals $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_T]$, infer a graph $\mathscr{G}$, and hence a GSO $\mathbf{S}$, such that the discrepancy between $f(\mathbf{S}; \boldsymbol{\theta})$ and $\mathbf{X}$ is minimized.

With some concrete examples, this formulation will be clearer. For this reason, we now specialize $f(\cdot)$ to three different models, namely the Gaussian graphical model (GGM) [20], the structural equation model (SEM), and the smoothness-based model (SBM), which will reappear in Chapter 5.

2.4.1. GAUSSIAN GRAPHICAL MODEL

Graphical models represent a class of statistical models which make use of graphs to encode the relationships between random variables [21, 22]. Consider a random vector $\mathbf{x} \in \mathbb{R}^N$ following a multivariate Gaussian distribution with mean $\boldsymbol{\mu} \in \mathbb{R}^N$ and covariance $\boldsymbol{\Sigma} \in \mathbb{S}^N_{++}$, i.e.,:

$$p(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^N |\boldsymbol{\Sigma}|}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\}. \tag{2.49}$$

It can be shown that the inverse covariance matrix $\boldsymbol{\Sigma}^{-1}$, called *precision matrix*, carries an important probabilistic interpretation. Specifically, it can be shown that [23]:

$$\boldsymbol{\Sigma}^{-1}_{ij} = 0 \Leftrightarrow p(x_i, x_j \mid x_{\mathcal{V}\setminus\{i,j\}}) = p(x_i \mid x_{\mathcal{V}\setminus\{i,j\}})p(x_j \mid x_{\mathcal{V}\setminus\{i,j\}}) \tag{2.50}$$

that is, a zero entry in the precision matrix encodes the *conditional independence* between the two random variables indexed by such entry, where the conditioning is with respect to all the other variables. How does this relates to graphs? Given an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{S})$, a random vector $\mathbf{x}$ is said to satisfy the undirected Gaussian graphical model with graph $\mathcal{G}$ if $\mathbf{x}$ is distributed according to (2.49), with:

$$\boldsymbol{\Sigma}^{-1}_{ij} = 0 \Leftrightarrow (i,j) \notin \mathcal{E}. \tag{2.51}$$

In other words, the sparsity pattern of the precision matrix $\boldsymbol{\Sigma}^{-1}$, which reveals the conditional independence between different variables, also captures the structure of the graph $\mathcal{G}$. Thus, function $f(\cdot)$ in (2.48) could be thought as the function that samples vectors from the distribution in (2.49), where its precision matrix $\boldsymbol{\Sigma}^{-1}$ encodes the sparsity pattern of the graph, that is $\mathbf{S} = \boldsymbol{\Sigma}^{-1}$. An interesting problem is then to learn $\mathcal{G}$ from vectors which are (assumed to be) sampled from such distribution, a GGM hypothesis.

Because we are into the realm of probability, when $\mathbf{x}$ in (2.49) is fixed, the probability density function $p(\cdot)$ only depends on the parameters of the distribution $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. We can then define the new function $l(\boldsymbol{\mu}, \boldsymbol{\Sigma}; \mathbf{x}) := p(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$ where $\mathbf{x}$ is known: under these lenses, function $l(\cdot)$ is called *likelihood function*[6] as it represents how likely it is for the observed $\mathbf{x}$ to have been drawn from a multivariate Gaussian distribution for different $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. Finding the best parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ which maximize such function, the so-called *maximum-likelihood estimation* (MLE), is a concept of paramount importance in signal processing and machine learning, as it enables to fit a probability distribution to observed data.

By maximizing the logarithm of the likelihood function[7] and acknowledging the one-to-one correspondence between the covariance matrix $\boldsymbol{\Sigma}$ and its inverse $\boldsymbol{\Sigma}^{-1}$, we can maximize the log-likelihood function in terms of Σ^{-1}, namely:

$$\boldsymbol{\Sigma}^{-1}_{MLE} = \underset{\boldsymbol{\Sigma}^{-1}}{\operatorname{argmax}} \log\left(|\boldsymbol{\Sigma}^{-1}|^{-1}\right) - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_{MLE})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_{MLE}), \tag{2.52}$$

where $\boldsymbol{\mu}_{MLE}$ is the maximum (log-)likelihood estimator of the mean $\boldsymbol{\mu}$.

[6]Notice that the likelihood function is not a probability density function

[7]The logarithm is a monotone increasing function and as such the maximizer of the function to which it is applied does not change.

Consider now the observation matrix $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_T]$, where each $\mathbf{x}_i$ is assumed to be distributed according to (2.49) while satisfying the GGM hypothesis (2.51). Under the assumptions that $\mathbf{x}_1, \cdots, \mathbf{x}_T$ are independent and identically distributed (i.i.d), we have:

$$l(\boldsymbol{\mu}, \boldsymbol{\Sigma}; \mathbf{X}) = p(\mathbf{x}_1 \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) \cdots p(\mathbf{x}_T \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}). \tag{2.53}$$

In this case $\boldsymbol{\mu}_{\text{MLE}} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{x}_t$, and because $\mathbf{S} := \boldsymbol{\Sigma}^{-1}$, we can cast the graph learning problem as:

$$\begin{aligned} \underset{\mathbf{S}}{\operatorname{argmin}} \quad & -\log\det(\mathbf{S}) + \operatorname{tr}(\mathbf{S}\hat{\boldsymbol{\Sigma}}) \\ \text{s. t.} \quad & \mathbf{S} \in \mathbb{S}^N_{++} \end{aligned} \tag{2.54}$$

where $\hat{\boldsymbol{\Sigma}} = \frac{1}{T}\sum_{t=1}^{T}(\mathbf{x}_t - \boldsymbol{\mu}_{\text{MLE}})(\mathbf{x}_t - \boldsymbol{\mu}_{\text{MLE}})^\top$ is the empirical covariance matrix. This problem is convex in $\mathbf{S}$ and thus yielding a unique minimum.

2.4.2. Structural Equation Model

Structural equation modeling (SEM) refers to a general statistical modeling technique for multivariate data which is widely adopted in several different disciplines, such as in social sciences [24], psychology [25] and genetics [26]. The popularity of SEM is mainly due to its ease of use and the ability to capture the influence of exogenous factors as well as causal effects for the variable of interest [27] (although we should point out that the latter interpretation has been a cause of concern [28]).

Given an observation vector $\mathbf{x} \in \mathbb{R}^N$, SEM poses a linear dependence between the signal value x_i and two other sets of variables: the signal values $\{x_j\}_{j\neq i}$, representing *endogenous variables*, and exogenous variables $\{u_i\}_{i=1}^N$. Formally:

$$x_i = \sum_{j\neq i} W_{ij} x_j + B_{ii} u_i + e_i, \tag{2.55}$$

where W_{ij} weights the influence that node j exerts on node i, B_{ii} encodes the influence of the exogenous variable for node i, and e_i captures the unmodeled disturbances.

Because $W_{ii} = 0$ and because W_{ij} captures the relationship between variables i and j, it seems natural to interpret the overall matrix of weights $\mathbf{W}$, with the (i, j)th entry equal to W_{ij}, as a weighted adjacency matrix of an interaction graph. Thus (2.55) endows a notion of “neighborhood regression”. By considering the matrix $\mathbf{W}$ to be the GSO $\mathbf{S}$, i.e., $\mathbf{S} = \mathbf{W}$, the matrix-vector form of (2.55) becomes:

$$\mathbf{x} = \mathbf{S}\mathbf{x} + \mathbf{B}\mathbf{u} + \mathbf{e} \tag{2.56}$$

where $\mathbf{B} = \operatorname{Diag}([B_{00}, \ldots, B_{(N-1)(N-1)}])$. This formulation corresponds to $f(\mathbf{S}; \boldsymbol{\theta})$ in (2.48) for SEM.

Upon availability of multiple observation vectors $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_T]$, viewed as graph signals residing over a latent graph described by the GSO $\mathbf{S}$ which obey a SEM model as in (2.56) yet without exogenous variables, a graph topology identification problem can be casted as the following least squares problem:

$$\begin{aligned} \underset{\mathbf{S}}{\operatorname{argmin}} \quad & \|\mathbf{X}-\mathbf{S}\mathbf{X}\|_F^2 + g(\mathbf{S}) \\ \text{s. t.} \quad & \mathbf{S} \in \mathscr{S} \end{aligned} \tag{2.57}$$

where $\mathscr{S} = \{\mathbf{S} | \operatorname{Diag}(\mathbf{S}) = \mathbf{0}, S_{ij} = S_{ji}, i \neq j\}$, i.e., it is the set of hollow symmetric matrices, and $g(\mathbf{S})$ is a regularizing term enforcing $\mathbf{S}$ to have desired properties, such as sparsity.

2.4.3. Smoothness-Based Model

In Chapter 2 we have seen what it means for a graph signal $\mathbf{x}$ to be smooth on top of a graph with a given Laplacian $\mathbf{L}$; namely the Laplacian quadratic form $\operatorname{LQ}(\mathbf{x},\mathbf{L})$ has a small value [cf. (2.30)]. Although this concept of variation can be explained in deterministic terms, it can also be casted as a probabilistic model under the lens of factor analysis. Given a vector $\mathbf{x} \in \mathbb{R}^N$, factor analysis expresses each variable x_i as the combination of $K < N$ unobserved *common factors* $\hat{x}_0, \ldots, \hat{x}_{K-1}$, which are the same for all $i = 0, \ldots, N$; in matrix-vector form:

$$\mathbf{x} = \mathbf{V}\hat{\mathbf{x}} + \boldsymbol{\mu} + \mathbf{e} \tag{2.58}$$

where $\mathbf{V} \in \mathbb{R}^{N \times K}$ is the matrix of *factor loadings*, $\boldsymbol{\mu} \in \mathbb{R}^N$ is the mean vector of $\mathbf{x}$, i.e., $\boldsymbol{\mu} = \mathbb{E}[\mathbf{x}]$, and $\mathbf{e}$ is the vector of so-called *specific factors*, which we assume to be white noise with isotropic variance σ^2, i.e., $\mathbf{e} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$. Essentially (2.58) can be seen as a collection of N multiple regression analysis for each variable x_i, each one having as regression coefficients the K values collected in the ith row of matrix $\mathbf{V}$, and the (common) predictors given by $\hat{\mathbf{x}}$. The term $\boldsymbol{\mu}$ can be considered to collect the intercept of each regression, and $\mathbf{e}$ to collect the associated noise. With modeling assumptions on the terms in (2.58), we now see how this can be interpreted in a GSP framework.

Assume to have a graph described by the Laplacian $\mathbf{L}$ and that the matrix $\mathbf{V}$ in (2.58) is the eigenvector matrix of $\mathbf{L}$, which admits the eigendecomposition $\mathbf{L} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^\top$. Moreover, suppose that we model the common factors $\hat{\mathbf{x}}$ to follow a zero-mean multivariate Gaussian distribution, with covariance matrix equal to the pseudoinverse of $\boldsymbol{\Lambda}$, i.e.,:

$$\hat{\mathbf{x}} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Lambda}^\dagger). \tag{2.59}$$

Then, it follows that:

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{L}^\dagger + \sigma^2\mathbf{I}), \tag{2.60}$$

$$\mathbf{x} \,|\, \hat{\mathbf{x}} \sim \mathcal{N}(\mathbf{V}\hat{\mathbf{x}} + \boldsymbol{\mu}, \sigma^2\mathbf{I}), \tag{2.61}$$

It is easy to see how the energy of the signal $\mathbf{x}$ is mainly gathered in the low-frequency components. Indeed, from (2.59), we observe that the factor $\hat{x}_i$ has a variance equal to $1/\lambda_i$: since for the Laplacian $\mathbf{L}$ holds $\lambda_i \leq \lambda_{i+1}$, we have that on average the energy of $\hat{x}_i$ is higher than the energy of $\hat{x}_{i+1}$; thus because lower eigenvalues are associated to smoother eigenvectors, such a signal model promotes the smoothness of the graph signal $\mathbf{x}$. An alternative view of seeing this is given by observing that a signal $\mathbf{x}$ obeying (2.60) can be generated as filtering white Gaussian noise $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ with a low-pass filter:

$$\mathbf{x} = (\mathbf{V}\boldsymbol{\Lambda}^{\dagger}\mathbf{V}^{\top})\mathbf{n} + \mathbf{e} \tag{2.62}$$

where the graph filter has non-increasing values along the diagonal, i.e., $\hat{h}(\lambda_i) = 1/\sqrt{\lambda_i}$. This represents the modeling formulation $f(\mathbf{S};\boldsymbol{\theta})$ of (2.48). Nevertheless, we have yet to observe any term indicative of the LQ, i.e., $\mathrm{LQ}(\mathbf{x}, \mathbf{L})$, which characterizes the inverse problem.

Given an observation $\mathbf{x}$ and the prior on the common factors $\hat{\mathbf{x}}$, the maximum a posteriori (MAP) estimation of $\hat{\mathbf{x}}$ reads as:

$$\begin{aligned} \hat{\mathbf{x}}_{\mathrm{MAP}}(\mathbf{x}) &:= \underset{\hat{\mathbf{x}}}{\operatorname{argmax}}\, p(\hat{\mathbf{x}} \mid \mathbf{x}) && (2.63)\\ &= \underset{\hat{\mathbf{x}}}{\operatorname{argmax}}\, p(\mathbf{x} \mid \hat{\mathbf{x}})\, p(\hat{\mathbf{x}}) && (2.64)\\ &= \underset{\hat{\mathbf{x}}}{\operatorname{argmin}} - \log p(\mathbf{x} \mid \hat{\mathbf{x}}) - \log p(\hat{\mathbf{x}}) && (2.65) \end{aligned}$$

By substituting (2.59) and (2.61) into (2.65), we reach:

$$\hat{\mathbf{x}}_{\mathrm{MAP}}(\mathbf{x}) = \underset{\hat{\mathbf{x}}}{\operatorname{argmin}}\, \|\mathbf{x} - \mathbf{V}\hat{\mathbf{x}}\|_2^2 + \kappa \hat{\mathbf{x}}^{\top} \boldsymbol{\Lambda} \hat{\mathbf{x}} \tag{2.66}$$

where κ endows a constant parameter for notational simplicity. While the first term in (2.66) is a data fidelity term trying to minimize the noise $\mathbf{e}$, the second encodes smoothness properties. To see this, we use the change of variable $\mathbf{y} = \mathbf{V}\hat{\mathbf{x}}$; then, solving (2.66) is equivalent to solving:

$$\mathbf{y}_{\mathrm{MAP}} = \underset{\mathbf{y}}{\operatorname{argmin}} \|\mathbf{x} - \mathbf{y}\|_2^2 + \kappa \mathbf{y}^{\top} \mathbf{L} \mathbf{y} \tag{2.67}$$

where we finally get the MAP estimate as $\hat{\mathbf{x}}_{\mathrm{MAP}}(\mathbf{x}) = \mathbf{V}^{\top}\mathbf{y}_{\mathrm{MAP}}$. Problem (2.67) can then be seen as the one which tries to find a noise-free version of the observed signal $\mathbf{x}$ which is also smooth on the graph $\mathbf{L}$; the smoothness measure is encoded by the second term of (2.67).

In the context of graph topology inference, we are given multiple observation vectors $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_T]$ and the goal is to learn a graph encoded by the GSO $\mathbf{S} = \mathbf{L}$ such that the observed signals are as smooth as possible. Thus, graph learning under a smoothness prior can be casted as:

$$\begin{aligned} \underset{\mathbf{S}}{\operatorname{argmin}} \quad & \frac{1}{T}\mathrm{tr}(\mathbf{X}^{\top}\mathbf{S}\mathbf{X}) + g(\mathbf{S}) \\ \text{s. t.} \quad & \mathbf{S} \in \mathcal{S} \end{aligned} \tag{2.68}$$

where the term $g(\mathbf{S})$ accommodates for additional topological properties (e.g., sparsity) and also helps avoiding the trivial solution $\mathbf{S} = \mathbf{0}$. The set $\mathcal{S}$ represents, in this case, the set of Laplacian matrices.

2.5. Conclusions

This chapter provided the foundational mathematical knowledge necessary for comprehending the subsequent chapters. Initially, we introduced fundamental concepts such as shift, convolution, and (shift-invariant) filters within the time-domain. Exploring the definition of the shift operator, we observed how the class of shift-invariant filters can be expressed as a polynomial of this operator, thereby demonstrating its commutative property with the shift. When considering the circulant shift (represented by the lower delay matrix), we noted that its eigenvalues correspond to the N complex roots of unity, defining the concept of frequency. Furthermore, we identified that the eigenvectors coincide with the inverse DFT matrix. Building upon this insight, we derived the convolution theorem, where the multiplication of a circulant matrix and a vector implements the DFT of the vectors, facilitating a pointwise multiplication in the frequency domain. Subsequently, we formally introduced graph concepts, including their representation matrices and the formal definition of a signal on a graph. Similar to the time domain, when we select a graph shift operator (GSO), we observe that the entire class of shift-invariant graph filters can be expressed as a polynomial of this GSO. In this case as well, the eigenvalue decomposition of the shift reveals a notion of frequency, leading to a graph Fourier transform and a graph convolution theorem. Viewing standard signal processing through these lenses, we can see it as a specific case of graph signal processing when the shift operator is a circulant matrix. Moving on to the second part of the chapter, we illustrated the problem of (static) graph topology inference. This problem aims to learn the edge connectivity of the graph from the available nodal data. We concretely discussed three different inference problems commonly encountered in the literature, namely the Gaussian graphical model, the structural equation model, and the smoothness-based model, which differ on the hypothesis behind the graph-dependent generative process of the data.

With this groundwork laid, we are now prepared to formally introduce the problems studied in this dissertation, along with the corresponding technical contributions aimed at addressing these challenges.

REFERENCES

[1] E. R. Dougherty. “The evolution of scientific knowledge: from certainty to uncertainty”. In: (2016).

[2] A. V. Oppenheim. *Discrete-time signal processing*. Pearson Education India, 1999.

[3] J. G. Proakis. *Digital signal processing: principles, algorithms, and applications, 4/E*. Pearson Education India, 2007.

[4] Y. LeCun, Y. Bengio, *et al.* “Convolutional networks for images, speech, and time series”. In: *The handbook of brain theory and neural networks* 3361.10 (1995), p. 1995.

[5] B. Bamieh. “Discovering transforms: A tutorial on circulant matrices, circular convolution, and the discrete fourier transform”. In: *arXiv preprint arXiv:1805.05533* (2018).

[6] J. Grolle, B. Donners, J. A. Annema, M. Duinkerken, and O. Cats. “Service design and frequency setting for the European high-speed rail network”. In: *Transportation Research Part A: Policy and Practice* 179 (2024), p. 103906. ISSN: 0965-8564. DOI: https://doi.org/10.1016/j.tra.2023.103906. URL: https://www.sciencedirect.com/science/article/pii/S0965856423003269.

[7] A. Sandryhaila and J. M. Moura. “Discrete signal processing on graphs”. In: *IEEE transactions on signal processing* 61.7 (2013), pp. 1644–1656.

[8] 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”. In: *IEEE signal processing magazine* 30.3 (2013), pp. 83–98.

[9] G. B. Folland. *Introduction to partial differential equations*. Vol. 102. Princeton university press, 1995.

[10] C. C. Pinter. *A book of abstract algebra*. Courier Corporation, 2010.

[11] M. Puschel and J. M. Moura. “Algebraic signal processing theory: Foundation and 1-D time”. In: *IEEE Transactions on Signal Processing* 56.8 (2008), pp. 3572–3585.

[12] E. Isufi, F. Gama, D. I. Shuman, and S. Segarra. “Graph filters for signal processing and machine learning on graphs”. In: *IEEE Transactions on Signal Processing* (2024).

[13] S. Segarra, A. G. Marques, and A. Ribeiro. “Optimal graph-filter design and applications to distributed linear network operators”. In: *IEEE Transactions on Signal Processing* 65.15 (2017), pp. 4117–4131.

[14] M. Coutino, E. Isufi, and G. Leus. “Advances in distributed graph filtering”. In: *IEEE Transactions on Signal Processing* 67.9 (2019), pp. 2320–2333.

[15] F. Gama, A. G. Marques, G. Leus, and A. Ribeiro. “Convolutional neural network architectures for signals supported on graphs”. In: *IEEE Transactions on Signal Processing* 67.4 (2018), pp. 1034–1049.

[16] E. D. Kolaczyk and G. Csárdi. *Statistical analysis of network data with R*. Vol. 65. Springer.

[17] G. Mateos, S. Segarra, A. G. Marques, and A. Ribeiro. “Connecting the dots: Identifying network structure via graph signal processing”. In: *IEEE Signal Processing Magazine* 36.3 (2019), pp. 16–43.

[18] X. Dong, D. Thanou, M. Rabbat, and P. Frossard. “Learning graphs from data: A signal representation perspective”. In: *IEEE Signal Processing Magazine* 36.3 (2019), pp. 44–63.

[19] G. B. Giannakis, Y. Shen, and G. V. Karanikolas. “Topology identification and learning over graphs: Accounting for nonlinearities and dynamics”. In: *Proceedings of the IEEE* 106.5 (2018), pp. 787–807.

[20] A. P. Dempster. “Covariance selection”. In: *Biometrics* (1972), pp. 157–175.

[21] S. L. Lauritzen. *Graphical models*. Vol. 17. Clarendon Press, 1996.

[22] J. Whittaker. *Graphical models in applied multivariate statistics*. Wiley Publishing, 2009.

[23] C. Uhler. “Gaussian graphical models: An algebraic and geometric perspective”. In: *arXiv preprint arXiv:1707.04345* (2017).

[24] A. S. Goldberger. “Structural equation methods in the social sciences”. In: *Econometrica: Journal of the Econometric Society* (1972), pp. 979–1001.

[25] R. C. MacCallum and J. T. Austin. “Applications of structural equation modeling in psychological research”. In: *Annual review of psychology* 51.1 (2000), pp. 201–226.

[26] X. Cai, J. Bazerque, and G. Giannakis. “Sparse structural equation modeling for inference of gene regulatory networks exploiting genetic perturbations”. In: *PLoS, Computational Biology* 9.5 (2013).

[27] J. Pearl. *Causality*. Cambridge university press, 2009.

[28] N. Cliff. “Some cautions concerning the application of causal modeling methods”. In: *Multivariate behavioral research* 18.1 (1983), pp. 115–126.

3
Advances in Graph Signal Processing

This chapter serves as a gentle introduction to the contributions offered by this Ph.D. dissertation. Rather than delving deeply into the mathematical intricacies, the objective of this chapter is to offer the reader a broad overview of the problem, along with a concise explanation of the approach or innovative ideas proposed to address it. Our aim is to ensure that each contribution is accessible to a diverse audience, irrespective of their background knowledge. Each contribution is further explored in its respective dedicated chapter, providing a more thorough examination. For the convenience of the reader, each section commences with an illustrative image capturing the essence of the research challenge addressed in the upcoming chapter, accompanied by a descriptive caption outlining the problem.

3.1. Graph Topology and Filter Tap Estimation

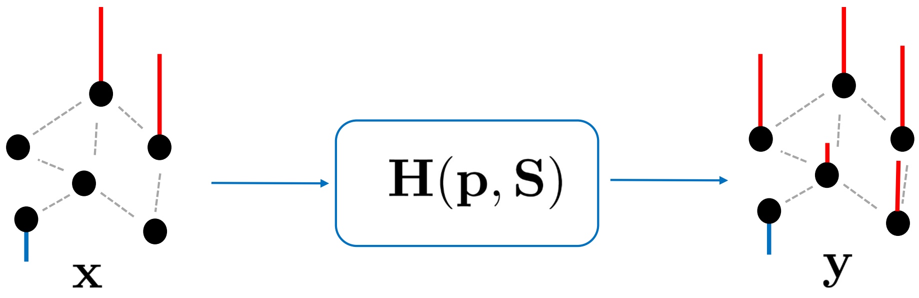


Figure 3.1: The learning problem: given input-output graph signal pairs $\{(\mathbf{x}_t, \mathbf{y}_t)\}_{t=1}^T$ learn the filter taps $\mathbf{p}$ and the GSO $\mathbf{S}$ which parameterize the graph filter $\mathbf{H}(\mathbf{p},\mathbf{S})$, with the prior knowledge of the sparsity pattern of $\mathbf{S}$.

In Chapter 2, we introduced the concept of graph convolution, implemented by a graph filter. Specifically, a graph filter of order $L-1$ is represented by the matrix polynomial:

$$\mathbf{H}(\mathbf{p},\mathbf{S}) = \sum_{l=0}^{L-1} p_l \mathbf{S}^l, \tag{3.1}$$

where $\mathbf{p} = [p_0,\dots,p_{L-1}]^\top$ denotes the filter taps. For a fixed $\mathbf{S}$, the number of free parameters in $\mathbf{H}(\mathbf{p},\mathbf{S})$ is equal to L, the length of the filter. This makes graph filters an attractive model to capture network dynamics in graph-based scenarios, especially when dealing with input-output data. Despite their limitation in terms of potential linear mappings compared to a general linear operator of dimension N^2, graph filters introduce an inductive bias, requiring less data to fit the model. Thus, in scenarios where the graph is known, individuals often adopt a supervised learning approach to learn the filter taps using such input and output data. Consequently, upon the arrival of new input data, predictions can be made for the associated output data.

However, there are instances where only the support of the graph is known, meaning only the binary adjacency matrix specifying which nodes are connected (and not their strength), is known. This is frequently encountered in brain networks where only a structural connectome is available, or in hydraulic networks where only the connections between pipes are known but not their capacities. Additionally, different GSOs possess distinct spectral properties and interpretations, presenting an intriguing challenge to determine the most suitable GSO that aligns with the available data.

Proposed Approach

Hence, our objective is as follows: given input-output graph signal pairs $\{(\mathbf{x}_t,\mathbf{y}_t)\}_{t=1}^{T}$, we aim to capture the network dynamics using a graph filter $\mathbf{H}(\mathbf{p},\mathbf{S})$; precisely, our aim is to simultaneously estimate the filter taps $\mathbf{p}$ and the GSO $\mathbf{S}$ that most accurately represent these network dynamics, with only the sparsity pattern of the graph at our disposal. A possible optimization problem would then read as:

$$\begin{aligned} \mathbf{p},\mathbf{S} = \underset{\mathbf{p},\mathbf{S}}{\operatorname{argmin}} \; & \sum_{t=1}^{T} l\left(\tilde{\mathbf{y}},\mathbf{y}_t\right) \\ \text{s.t. } & \mathbf{S} \in \mathscr{S} \\ & \operatorname{supp}(\mathbf{S}) \subseteq \mathscr{A} \end{aligned} \tag{3.2}$$

where $l(\cdot,\cdot)$ represents a loss function capturing the discrepancy between the predicted output signal $\tilde{\mathbf{y}} = \mathbf{H}(\mathbf{p},\mathbf{S})\mathbf{x}_t$ and the true output signal $\mathbf{y}_t$, $\mathscr{S}$ represents the set of valid GSOs and $\mathscr{A}$ denotes the set with the known support of $\mathbf{S}$.

In Chapter 4, after recognizing the non-convexity of the problem, we propose an iterative alternating minimization (AM) scheme, where the non-convex portion of the problem is solved through a sequential convex approximation (SCA) approach [1].

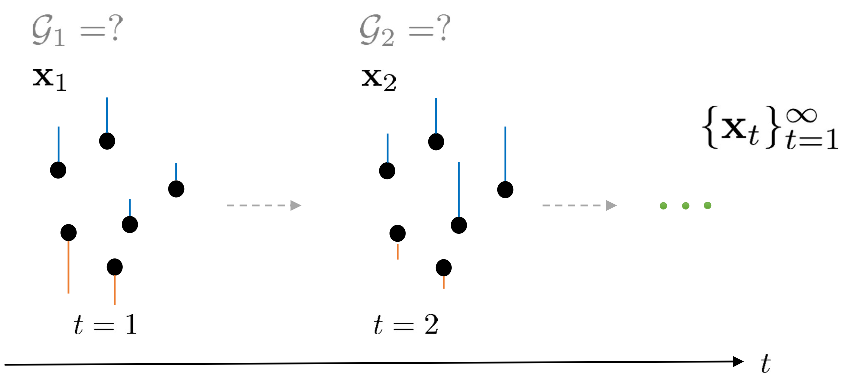


Figure 3.2: The time-varying graph topology identification problem: a network process $f(\cdot)$ generates a series of vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots$. The goal is to learn the time-varying graph topology encoded by the sequence $\mathbf{S}_1, \mathbf{S}_2, \ldots$ through the knowledge of the data $\{\mathbf{x}_i\}$ and prior information on the data generating process $f(\mathbf{S}_t, \boldsymbol{\theta})$ [cf. (2.48)].

3.2. TIME-VARYING GRAPH TOPOLOGY INFERENCE

In Section 2.4 of Chapter 2, we explored the significance of learning graphs in network science and how we can systematically address this challenge using mathematical optimization techniques. Specifically, we investigated how different assumptions about the data, such as Gaussianity or smoothness, give rise to distinct optimization problems, each tailored to specific applications. A common thread among these approaches is that, despite the possibility of observing different vectors $\mathbf{x}_1, \ldots, \mathbf{x}_T$ over a temporal horizon, the underlying graph $\mathbf{S}$ is presumed to remain constant. In other words, the graph topology is considered *static*, and the overall learning framework is referred to as *static* graph topology identification.

However there are plenty of real-world scenarios where such immutability property does not hold. Examples include:

- Functional connectivity networks: the graph models the functional connectivity between different brain regions, such as the prefrontal cortex, the amygdala or the hippocampus. This connectivity is time-varying, since the brain networks responds differently to different stimuli or tasks. If the time horizon is long enough, such changes are also due to brain development and aging, or due to neuropsychiatric disorders.

- Correlation networks, which represent random variables of interest as vertices, with connections between them determined by the correlation coefficient of those variables. This is exemplified in graphs depicting stock market correlations among various assets, which are notably impacted by geopolitical events and consequently evolve over time.

- Mobile networks, where each node is designated as either a base station or a mobile user, and the edges symbolize the signal strength between a user and their respective assigned base station. Due to user mobility, the signal strength fluctuates, and in certain instances, a user switches to a new base station, thus creating a new edge.

In all these examples the graph topology changes over time. The class of problems and the associated techniques concerning the identification of a time-varying network structure (from data) are known as time-varying graph topology identification.

One potential approach to tackle this problem involves considering each graph signal $\mathbf{x}_t$ as associated with an unknown graph $\mathcal{G}_t = \{\mathcal{V}, \mathcal{E}_t, \mathbf{S}_t\}$, , where $t \in \mathbb{N}_+$ serves as a discrete time index, as depicted in Fig. 3.2. This is akin to having a sequence of latent graphs $\{\mathcal{G}_t = \{\mathcal{V}, \mathcal{E}_t, \mathbf{S}_t\}\}_{t=1}^{\infty}$ generating the graph signals $\{\mathbf{x}_t\}_{t=1}^{\infty}$ according to some graph- and time-dependent process, i.e., :

$$\mathbf{x}_t = f(\mathbf{S}_t; \boldsymbol{\theta}) \tag{3.3}$$

with $\boldsymbol{\theta}$ capturing model-dependent parameters. This modeling approach also encompasses the scenario where the graph remains constant across multiple time instants.

We consider an additional challenge: the signals arrive in an *online* (or streaming) fashion. That is, we need to process the data on-the-fly, thus avoiding a batch-strategy. This requires an algorithm capable of digesting information in a timely manner.

Proposed Approach

Mathematically, our goal is to solve the sequence of time-invariant problems:

$$\mathbf{S}_t^{\star} := \underset{\mathbf{S}}{\operatorname{argmin}}\, F(\mathbf{S}; t) \quad t = 1, 2, \ldots \tag{3.4}$$

where function $F(\cdot; t)$ is a time-varying cost function that depends on the data model, i.e., the assumption that we make on the generative process of the data [cf. Section 2.4], and the index t makes the dependence on time explicit, which is due to the arrival of new data. In relation to the models explored in Chapter 2, the optimization problem (3.4) could be the time-varying counterpart of (2.54), (2.57) and (2.68), aiming to (informally) revert the data generating process $f(\cdot; \boldsymbol{\theta})$.

Solving (3.4) exactly for any t not only might be computationally expensive, since it involves running a solver multiple times, but it might also be unnecessary. Indeed, the sequence of solutions is time-varying and so we are more interested in following the general graph dynamics. Moreover, we are not taking into account that consecutive solutions are, in general, also similar. This is why we adopt an iterative scheme to solve (3.4) for each t which approximates the optimal solution, which is based on novel time-varying convex optimization tools taking into account the time-dependence of the cost function [2]. We provide an in-depth treatment of this problem and the relative solution in Chapter 5.

$$\begin{tikzcd}
\mathbf{x} \arrow[r] \arrow[d] & \boxed{\mathbf{H}_I(\mathbf{P},\mathbf{S})} \arrow[r] & \mathbf{y} \arrow[d] \\
\boxed{\mathbf{V}^H} \arrow[d] & & \boxed{\mathbf{V}^H} \arrow[d] \\
\hat{\mathbf{x}} \arrow[r] & \boxed{\mathbf{H}_{II}(\hat{\mathbf{P}},\mathbf{S}_f)} \arrow[r] & \hat{\mathbf{y}}
\end{tikzcd}$$

Figure 3.3: A generalization of the convolution theorem: a type-I NV-GF $\mathbf{H}_I(\mathbf{P},\mathbf{S})$ in the vertex domain is equivalent to a type-II NV-GF $\mathbf{H}_{II}(\hat{\mathbf{P}},\mathbf{S}_f)$ in the frequency domain.

3.3. A Generalization of the Convolution Theorem

In Chapter 2 we have seen the convolution theorem, which could be informally stated as:

> *“A convolution in one domain is equivalent to a pointwise multiplication in the other domain”*

This theorem applies in both classical signal processing and graph signal processing. Indeed, we have observed how classical signal processing can be viewed as a particular instance of graph signal processing when the graph shift operator $\mathbf{S}$ takes the form of a circular shift $\mathbf{S}_c$ (or any other circulant matrix).

However, under what conditions does the theorem hold? For a given shift operator $\mathbf{S}$, the convolution theorem holds true when the filter $\mathbf{H}$ and the shift $\mathbf{S}$ commute, meaning $\mathbf{HS} = \mathbf{SH}$, a property known as "shift-invariance". This condition is equivalent to having $\mathbf{S}$ and $\mathbf{H}$ jointly diagonalizable, indicating they share the same eigenvectors. Additionally, we have explored how $\mathbf{H}$ can be expressed as a polynomial of $\mathbf{S}$, with the coefficients of such polynomials representing the filter taps. These filters are considered "isotropic": in the temporal domain, this implies time-invariance where the filter remains constant when applied at different time instants; on a graph, this translates to node-invariance, meaning each node assigns weights to nodes in different hops using the same coefficient as any other node.

Although these modeling choices are appealing from an analytical perspective, due to an intuitive frequency interpretation, they pose restrictions or may even present infeasible solutions for certain real-world systems. This is particularly evident in scenarios such as underwater communications or high-mobility systems, where the channel between input and output signals exhibits significant time variation. Consequently, assuming a static filter response would result in suboptimal signal processing designs. In such cases, time-varying filters are often employed, representing a broader class of operators that also encompasses time-invariant filters.

The graph-based counterpart to time-varying filters are node-varying graph filters (NV-GFs) [see Section 2.3.2], wherein each node of the graph applies a distinct coefficient to weigh the values of its neighbors during the shifting operation. Due to the lack of joint diagonalizability with the GSO $\mathbf{S}$, the elegant spectral interpretation provided by the graph convolution is lost. Nevertheless, in Chapter 6 we are able to propose a new convolution theorem which generalizes the classical one, and that can be informally stated as:

> *"A node-variant graph convolution in one domain is equivalent to another node-variant graph convolution in the other domain."*

This statement should however raise a question: what does it mean to have a graph convolution in the frequency domain if we do not have a graph in frequency? Indeed remember from Section 2.3 that the support of the frequency domain is given by the eigenvalues $\boldsymbol{\lambda}$ of the GSO $\mathbf{S}$, which are a mere discretization of the real line $\mathbb{R}$ or complex plane $\mathbb{C}$; see Fig 3.4 for a GFT signal supported on $\boldsymbol{\lambda}$.

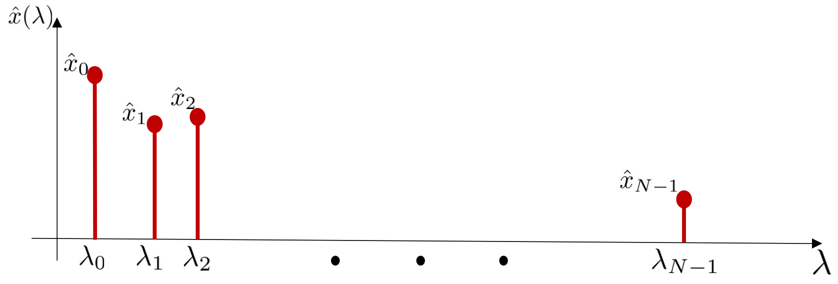


Figure 3.4: A GFT signal residing on a discretization of the real line.

A recent line of work [3, 4] posits the existence of a graph, termed dual graph and denoted with $\mathbf{S}_f$, capturing the structure of the frequency domain (see also Fig. 3.5). The motivation behind this line of research relies on the fact that classical signal processing tasks usually performed in the frequency domain, such as frequency-shifting, do not have their counterpart in GSP. Furthermore, given that a graph signal is inherently associated with a graph structure, it is desirable to establish a corresponding Fourier representation that is also inherently linked to a graph structure.

We make use of this notion to generalize the convolution theorem as we informally stated above, where a node-variant graph convolution in the primal domain with

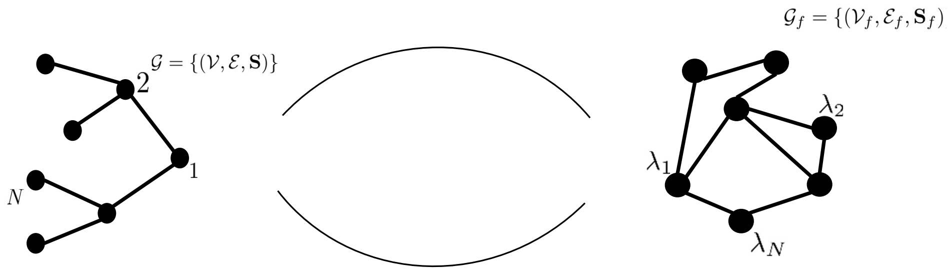

$\mathcal{G} = \{(\mathcal{V}, \mathcal{E}, \mathbf{S})\}$
2
1
N
$\mathcal{G}_f = \{(\mathcal{V}_f, \mathcal{E}_f, \mathbf{S}_f)$
λ_2
λ_1
λ_N

Figure 3.5: Illustrative example of primal GSO $\mathbf{S}$ and dual GSO $\mathbf{S}_f$, which capture the structure of the vertex and frequency domain, respectively.

a type-I NV-GF $\mathbf{H}_I(\mathbf{P}, \mathbf{S})$ is equivalent to a node-variant graph convolution in the dual domain with a type-II NV-GF $\mathbf{H}_{II}(\hat{\mathbf{P}}, \mathbf{S}_f)$, under a specific parametrization of the filter coefficients $\mathbf{P}$. We show how the classical convolution theorem in (graph) signal processing and the one related to time-varying filters are specific cases of this general convolution theorem. After discussing its implications in terms of non-stationarity, we propose a data-driven graph learning approach to learn a dual graph $\mathbf{S}_f$ from data such that it respects the introduced convolution theorem. We provide an in-depth treatment of this in Chapter 5.

3.4. CONCLUSIONS

In this chapter, we provided an introductory overview of the problems object of discussion in this dissertation and the proposed algorithmic solutions. Without further extending our discussion, in Chapter 4 we will devise a data-driven methodology to jointly estimate the filter taps and the GSO weights (the constituents of a graph filter), from input-output data. Following that, in Chapter 5, we will study the challenge of learning time-varying graphs from online data and propose an algorithmic approach to solve it. Lastly, in Chapter 6 we will study the convolution theorem and propose its extension to shift-variant filters.

REFERENCES

[1] S. Boyd. "Sequential convex programming". In: *Lecture Notes, Stanford University* (2008).

[2] A. Simonetto, A. Mokhtari, A. Koppel, G. Leus, and A. Ribeiro. "A class of prediction-correction methods for time-varying convex optimization". In: *IEEE Transactions on Signal Processing* 64.17 (2016), pp. 4576–4591.

[3] G. Leus, S. Segarra, A. Ribeiro, and A. G. Marques. "The dual graph shift operator: Identifying the support of the frequency domain". In: *Journal of Fourier Analysis and Applications* 27.3 (2021), p. 49.

[4] J. Shi and J. M. Moura. "Graph signal processing: Dualizing gsp sampling in the vertex and spectral domains". In: *IEEE Transactions on Signal Processing* 70 (2022), pp. 2883–2898.

4

Topology-Aware Joint Graph Filter and Edge Weight Identification for Network Processes

Data defined over a network have been successfully modelled by means of graph filters. However, although in many scenarios the connectivity of the network is known, e.g., smart grids, social networks, etc., the lack of well-defined interaction weights hinders the ability to model the observed networked data using graph filters. Therefore, in this paper, we focus on the joint identification of coefficients and graph weights defining the graph filter that best models the observed input/output network data. While these two problems have been mostly addressed separately, we here propose an iterative method that exploits the knowledge of the support of the graph for the joint identification of graph filter coefficients and edge weights. We further show that our iterative scheme guarantees a non-increasing cost at every iteration, ensuring a globally-convergent behavior. Numerical experiments confirm the applicability of our proposed approach.

4.1. Introduction

The increasing amount of *networked* data, also conceptualized as graph signals within the graph signal processing (GSP) field [2, 3], has gained a lot of attention in the scientific community. Due to this, many signal processing tasks have been adapted towards their networked counterpart, as extensively detailed in [4].

In the graph setting, it is common to parameterize network processes through graph filters, due to their versatility and their natural distributed implementation [5] [6]. They play an important role within GSP, with applications ranging from

Parts of this chapter have been published in the International Workshop on Machine Learning for Signal Processing (2020) [1].

reconstruction [7] [8] [9], denoising [10] and classification [11], to forecasting [12] [13] and (graph-)convolutional neural networks [14]. Notable recent advances in such structures are [15], which generalizes state-of-the-art graph filters to filters where every node weights the signal of its neighbors with different values, and [16], which extends the classical problem of blind system identification or blind de-convolution to the graph setting.

Given the structure of the graph, encoded by the so-called graph shift operator (GSO) [3], and assuming a process modelled by a graph filter, identifying an underlying network process from input/output networked data amounts to estimate the graph filter coefficients, thus alleviating the estimation workload [3] [17]. A key assumption in graph filtering is the *knowledge* of the GSO, which can be obtained from some other field of research or can be estimated from historical data. The latter relates to network topology inference or graph learning which, in recent years, has experienced an exponentially-increasing scientific interest, see, e.g., [18–20].

Related to the scenario we are going to consider, there are also works that model the observed signal as the output of an unknown graph filter over an unknown graph. In [21], a two-step GSO identification approach is taken, where first the GSO's eigenvectors are identified from the diffused (stationary) graph signals and then the GSO's eigenvalues are estimated based on some general properties of the GSO. In [22], the work of [21] is extended to non-stationary graph signals, entailing the solution of a system of quadratic matrix equations. Using the same approach, the problem of directed network topology identification is investigated in [23]. Note, though, that none of these above works focuses on estimating the related graph filter. More similar to our work, is the approach of [24], where not only the GSO but also the filter taps are learned. Although the context of [24] is different, in that work, a general linear filter operator is estimated from the data and then both the GSO and the filter taps are estimated from it.

All the previous approaches rely on a multi-step algorithm and only exploit some general properties of the GSO, e.g., sparsity. In addition, in many practical networks such as social and supply networks, the support of the graph is a priori known, that is, the connections between different entities of the network are already known, yet their importance might be unknown. And this information is not directly handled by the above algorithms.

Motivated by the above reasons, this work aims to jointly estimate the graph filter coefficients and the weights of the network topology. This joint approach leads to an optimization problem that is non-convex. We tackle the non-convexity of the problem by building on sequential convex programming (SCP), a local optimization tool for non-convex problems that leverages the convex optimization machinery. We show that an alternating minimization between the filter coefficients and the GSO guarantees that the objective function value at each iteration is non-increasing, obtaining a globally convergent method.

4.2. PRELIMINARIES

In this section, we introduce the GSP background material necessary for the rest of the paper, including the formal definition of graph signals and the core concepts of graph filtering and topology identification.

Graph Signal Processing We consider the case in which the data of interest live in a non-Euclidean domain, described by the undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{S})$, where $\mathcal{V} = \{1, \ldots, N\}$ is the set of nodes (or vertices), $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges, and $\mathbf{S}$ is a symmetric $N \times N$ matrix that represents the graph structure. The matrix $\mathbf{S}$ is called the graph shift operator (GSO) [3], whose entries $[\mathbf{S}]_{ij}$ for $i \neq j$ are different from zero only if nodes i and j are connected by an edge. Typical choices of the GSO include the (weighted) adjacency matrix $\mathbf{W}$ [3] and the graph Laplacian $\mathbf{L}$ [2].

This allows us to define a graph signal, denoted by the vector $\mathbf{x} \in \mathbb{R}^N$, as a mapping from the node set to the set of real vectors; that is, $\mathbf{x} : \mathcal{V} \to \mathbb{R}^N$. In this way, $x_i \in \mathbb{R}$ is a scalar that represents the signal value at node i. Because $\mathbf{S}$ reflects the local connectivity of $\mathcal{G}$, the operation $\mathbf{S}\mathbf{x}$ performs at each node a local computation enabling us to introduce the concept of filtering in the graph setting.

Graph Filters We can process a graph signal $\mathbf{x}$ by means of a so-called graph filter [3] as:

$$\mathbf{y} = \mathbf{H}(\mathbf{h}, \mathbf{S})\,\mathbf{x} = \sum_{k=0}^{K} h_k \mathbf{S}^k \mathbf{x}, \tag{4.1}$$

where K is the order of the filter, $\mathbf{H}(\mathbf{h}, \mathbf{S})$ is a polynomial matrix on $\mathbf{S}$ and $\mathbf{h} := [h_0, \ldots, h_K]^\top$ is the vector that contains the filter taps. Due the locality of $\mathbf{S}$, graph filters represent linear transformations that can be implemented in a distributed setting [21]. More formally, the output entry y_i of $\mathbf{y}$ at node i is a linear combination of $K+1$ terms: the first term is the signal value x_i of node i; the kth term ($k = 1, 2, \ldots, K$) combines signal values x_j from the k-hop neighbors of node i.

Topology Identification When the connections of the network cannot be directly observed or the network is just a conceptual model of pair-wise relationships among entities, a fundamental question is how to learn its structure from the graph signals. Formally, consider the matrix $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_T] \in \mathbb{R}^{N \times T}$ that stacks column-wise T graph signals $\mathbf{x}_t$ residing over the network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{S})$. The goal is to infer the *latent* underlying network topology encoded in the GSO $\mathbf{S}$ under some optimality criterion.

This problem has been addressed in the past by means of statistical approaches, mostly based on correlation analysis and its connections to covariance selection and high-dimensional regression for learning Gaussian graphical models. Only more recently, GSP postulated the network topology inference problem under the assumption that the observed signals exhibit certain properties over the graph, such as smoothness, stationarity or band-limitedness. The reader interested in this topic is referred to [18] [19] [20].

Differently from the traditional topology identification setting, instead of estimating $\mathbf{S}$ from $\mathbf{X}$, we rely on model (4.1) and focus on a problem where given input and output data, the values of the nonzero entries of $\mathbf{S}$, i.e., the edge weights, and the filter taps $\mathbf{h}$ of a graph filter $\mathbf{H}(\mathbf{h}, \mathbf{S})$ have to be jointly identified. In Section 4.3, we rigorously formulate this problem and, in Section 4.4, we propose a way to efficiently tackle it.

4.3. Joint graph filter and topology estimation

Suppose there is an unknown network process that can be accurately modelled by a graph filter $\mathbf{H}(\mathbf{h},\mathbf{S})$ where, in response to an input $\mathbf{x}_t$, we observe a corresponding output $\mathbf{y}_t$. Such dynamics can be found for instance in social networks, where as a result of an advertisement campaign, we may expect to observe a response of the network's users; or in epidemics, where the nodes of the network are cities and we monitor the evolution of a spreading disease from one time instant to the next.

Let us assume that there are T input-output pairs available, and that we stack them column-wise in the matrices $\mathbf{X} = [\mathbf{x}_1,\dots,\mathbf{x}_T]$ and $\mathbf{Y} = \left[\mathbf{y}_1,\dots,\mathbf{y}_T\right]$, respectively. Let the unknown filter $\mathbf{H}(\mathbf{h},\mathbf{S})$ be of the form in (4.1). At this point, we are ready to formally state the problem we are going to address.

Problem Statement *Given the input-output data $\{\boldsymbol{x}_t,\boldsymbol{y}_t\}_{t=1}^T$ and the support, $\mathscr{A}$, of the graph $\mathscr{G}$, the goal is to identify the filter coefficients* $\mathbf{h}$ *and the GSO* $\mathbf{S}$ *embodied in the graph filter* $\mathbf{H}(\boldsymbol{h},\mathbf{S})$*, that maps $\boldsymbol{x}_t$ into $\boldsymbol{y}_t$ as accurately as possible.*

The above problem can be mathematically defined with a least-squares formulation as:

$$\begin{array}{cl} \underset{\mathbf{h},\mathbf{S}}{\operatorname{argmin}} & \|\mathbf{Y}-\sum_{k=0}^{K} h_k\mathbf{S}^k\mathbf{X}\|_{\mathrm{F}}^2 \\ \text{s.t.} & \mathbf{S}\in\mathscr{S} \\ & \operatorname{supp}(\mathbf{S})\subseteq\mathscr{A} \end{array} \tag{4.2}$$

where $\mathscr{S}$ represents the set of valid GSOs, $\mathscr{A}$ denotes the set with the support of $\mathscr{G}$, and $\|\cdot\|_{\mathrm{F}}$ denotes the Frobenius matrix norm. Note the (relaxed) constraint on the support: as the sparsity pattern of the GSO might have been overestimated, we leave it to the algorithm to optimize it, eventually shrinking to zero some unnecessary edges. That is, we constrain only the entries of the GSO to be zero in correspondence to the zeros of the support, leaving the other entries unconstrained (both zero and non-zero values are admitted).

From (4.2), we can deduce that the problem is not convex. Indeed, the objective function is made up of cross-products between the entries of $\mathbf{S}$ and the filter coefficients h_k, and by the power terms $\mathbf{S}^k$. The overall optimization problem is hence not convex and traditional tools of convex optimization cannot be used.

Although not directly handling the fixed-support case, the works referenced in Section 4.1 address the estimation problem using multi-step approaches to find $\mathbf{S}$ and/or $\mathbf{h}$. For instance, in [24] each realization is modeled through a graph filter-based vector auto-regressive (VAR) model, and this structure is leveraged to first recover the graph filters $\mathbf{H}_i(\mathbf{h},\mathbf{S})$ representing the matrix filter taps of the VAR, and only then to recover the shift $\mathbf{S}$ and the coefficients $\mathbf{h}$ from them. Other approaches, such as [21] [22] are only interested in learning the shift $\mathbf{S}$, while others, such as [16], only in the filter coefficients $\mathbf{h}$.

Differently from the method in [24], in the following, we introduce a globally convergent SCP-based method to directly find both the filter taps $\mathbf{h}$ and the GSO $\mathbf{S}$.

To the best of our knowledge, this is the first work that jointly learns the filter taps and the graph topology from observations.

4.4. Alternating Minimization

To tackle the non-convexity of the problem and to bypass the limited flexibility of other methods, we resort to the alternating minimization (AM) approach, acting iteratively on **h** and **S**. The general AM pseudo-code, adapted to our case, is reported in Algorithm 3. Notice that due to steps 3 and 4 in Algorithm 3, the cost is guaranteed to be a non-increasing function of the iteration number. In the following, we show how to perform step 3 and 4 of the proposed algorithm.

Given the estimate of the GSO $\mathbf{S}$ at the $(n-1)$th iteration, i.e., $\mathbf{S}^{(n-1)}$, the estimation problem at the nth iteration for the filter taps vector $\mathbf{h}$, i.e., $\mathbf{h}^{(n)}$, reads as:

$$\mathbf{h}^{(n)} = \underset{\mathbf{h}}{\operatorname{argmin}} \|\mathbf{Y} - \sum_{k=0}^{K} h_k \big(\mathbf{S}^{(n-1)}\big)^k \mathbf{X}\|_{\mathrm{F}}^2. \tag{4.3}$$

Problem (4.3) is convex and boils down to the traditional linear least squares (LLS) problem.

The solution of (4.3) is then used in the next step, i.e., step 4, to minimize the function with respect to the constrained GSO $\mathbf{S}$; that is

$$\begin{aligned} \mathbf{S}^{(n)} \quad = \quad & \underset{\mathbf{S}}{\operatorname{argmin}} \quad \{f(\mathbf{S}) := \|\mathbf{Y} - \textstyle\sum_{k=0}^{K} h_k^{(n)} \mathbf{S}^k \mathbf{X}\|_{\mathrm{F}}^2\} \\ & \text{s.t.} \quad \mathbf{S} \in \mathscr{S} \\ & \qquad\;\; \operatorname{supp}(\mathbf{S}) \subseteq \mathscr{A} \end{aligned} \tag{4.4}$$

As problem (4.4) is not convex, we employ SCP [25], a heuristic and local optimization method for non-convex problems that leverages convex optimization, where the non-convex portion of the problem is modeled by convex functions that are (at least locally) accurate.

Given the non-convex function $f(\mathbf{S})$, the idea in SCP is to maintain a solution estimate $\mathbf{S}^{[l]}$ and a respective convex *trust region* $\mathscr{T}^{[l]} \subseteq \mathbb{R}^{N \times N}$ over which we “trust” our solution to reside[1]. Then, using a convex approximation $\widehat{f}$ of f, around $\mathbf{S}^{[l]}$, the next solution estimate, $\mathbf{S}^{[l+1]}$, is computed using the optimizer of $\widehat{f}$ in $\mathscr{T}^{[l]}$. Typical trust regions include ℓ_2-norm balls or bounded regions.

For our case, we define as trust region the box:

$$\mathscr{T}^{[l]} = \begin{cases} \mathbf{S} \in \mathscr{S}, \\ \operatorname{supp}(\mathbf{S}) \subseteq \mathscr{A} \\ |[\mathbf{S}]_{ij} - [\mathbf{S}^{[l]}]_{ij}| \leq \rho_{ij}(l), \text{ if } (i,j) \in \mathscr{E} \neq 0, \forall i,j \in \mathscr{V}, \end{cases} \tag{4.5}$$

where $\rho_{ij} : \mathbb{Z}_+ \to \mathbb{R}_{++}$ is a mapping from the iteration number to the breadth of the search for the (i,j)th entry.

[1]We use the superscript with square brackets to indicate the SCP iterations.

Algorithm 1 Joint GF & GSO Identification

```
Require: Feasible $\mathbf{S}^{(0)}$, $\varepsilon > 0$, $\mathscr{A}$, $\mathscr{S}$
1: $n \leftarrow 1$
2: while not converged do
3:     $\mathbf{h}^{(n)} \leftarrow \text{argmin}_{\mathbf{h}} f\left(\mathbf{h}, \mathbf{S}^{(n-1)}\right)$        [See Eq. (4.3)]
4:     $\mathbf{S}^{(n)} \leftarrow \text{argmin}_{\mathbf{S}} f\left(\mathbf{h}^{(n)}, \mathbf{S}\right)$        (SCP) [See Alg. 2]
5:     Check convergence $(\mathbf{h}^{(n)}, \mathbf{S}^{(n)}, \varepsilon)$
6:     $n \leftarrow n+1$
7: end while
8: return $\mathbf{S}^{(n)}, \mathbf{h}^{(n)}$
```

For the convex approximation of the function, we linearize the function $f(\mathbf{S})$ around the previous estimate $\mathbf{S}^{[l]}$ using its first-order Taylor approximation[2]:

$$\hat{f}^{[l]}(\mathbf{S}) := f(\mathbf{S}^{[l]}) + \text{tr}\left[\nabla_{\mathbf{S}} f(\mathbf{S}^{[l]})^{\top}(\mathbf{S} - \mathbf{S}^{[l]})\right]. \tag{4.6}$$

We then find a feasible intermediate iterate by solving the problem:

$$\hat{\mathbf{S}} = \underset{\mathbf{S} \in \mathscr{T}^{[l]}}{\arg\min}\, \hat{f}^{[l]}(\mathbf{S}). \tag{4.7}$$

Due to the non-convexity of the cost function $f(\mathbf{S})$, its value at the (feasible) point $\hat{\mathbf{S}}$ is not guaranteed to be lower than the one at $\mathbf{S}^{[l]}$. Hence, to find the "best" feasible solution $\mathbf{S}$ at the $(l+1)$th iteration, we first resort to a line search to find the optimal scaling step size parameter α_l toward the feasible descent direction $\Delta_l := \hat{\mathbf{S}} - \mathbf{S}^{[l]}$; that is,

$$\alpha_l^* = \arg\min_{\alpha_l \in [0,1]} f(\mathbf{S}^{[l]} + \alpha_l \Delta_l). \tag{4.8}$$

Then, we compute our next solution estimate $\mathbf{S}^{[l+1]}$ through

$$\mathbf{S}^{[l+1]} = \mathbf{S}^{[l]} + \alpha_l^* \Delta_l, \tag{4.9}$$

which is feasible for the original problem as long as the set $\mathscr{S}$ is convex, i.e., the update in (4.9) is a convex combination of feasible points. The specialized SCP procedure for our problem is summarized in Algorithm 2. Note that steps 7-9 guarantee, at each iteration, the feasibility of the iterate and a non-increasing cost function value, leading to the global convergence of Algorithm 3.

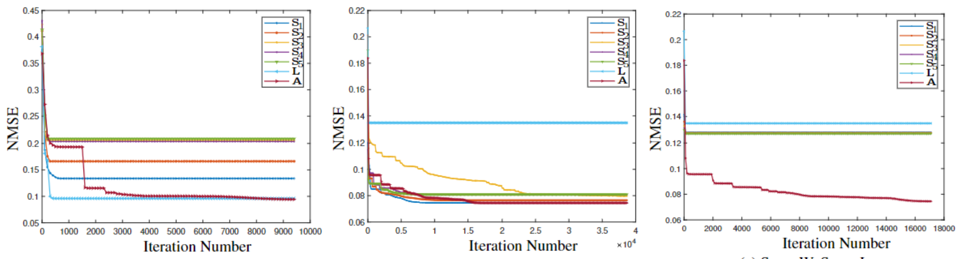
Due to the non-convexity of the cost function, the global optimality of the solution is not guaranteed, thus the results are dependent on the initial starting point(s) as they might lead to different local minima. Despite that in these cases multi-start is recommended, we have found in our numerical experiments that both the unweighted adjacency matrix, $\mathbf{A}$, and the respective combinatorial Laplacian matrix, $\mathbf{L}$, are good initial iterates, i.e., $\mathbf{S}^{(0)}$, for the proposed approach; they are straightforward choices and can be computed using the support of the graph. To

[2]The computation of $\nabla_{\mathbf{S}} f(\mathbf{S}^{[l]})$ is reported in the Appendix.

Algorithm 2 SCP

```
Require: $\mathbf{S}^{(n)}$, $\mathbf{h}^{(n)}$, $\{\rho_{ij}\}_{(i,j)\in\mathscr{E}}$ , $\varepsilon > 0$
 1: $l \leftarrow 1$
 2: $\mathbf{S}^{[0]} \leftarrow \mathbf{S}^{(n)}$
 3: while not converged do
 4:     Compute $\{\rho_{ij}(l-1)\}_{(i,j)\in\mathscr{E}}$
 5:     Construct $\hat{f}^{[l-1]}(\mathbf{S})$ as in (4.6)
 6:     Define $\mathscr{T}^{[l-1]}$ as in (4.5)
 7:     $\hat{\mathbf{S}} \leftarrow \arg\min_{\mathbf{S}\in\mathscr{T}^{[l-1]}} \hat{f}^{[l-1]}(\mathbf{S})$
 8:     $\alpha^*_{l-1} \leftarrow \arg\min_{\alpha_{l-1}\in[0,1]} f(\mathbf{S}^{[l-1]} + \alpha_{l-1}(\hat{\mathbf{S}} - \mathbf{S}^{[l-1]}))$
 9:     $\mathbf{S}^{[l]} \leftarrow \mathbf{S}^{[l-1]} + \alpha^*_{l-1}(\hat{\mathbf{S}} - \mathbf{S}^{[l-1]}))$
10:     Check convergence $(\mathbf{h}^{(n)}, \mathbf{S}^{[l]}, \varepsilon)$
11:     $l \leftarrow l + 1$
12: end while
13: return $\mathbf{S}^{[l]}$
```

validate this claim, in our experiments, we generate initial GSO iterates $\mathbf{S}_i^{(0)}$, through a method reported in the Appendix, and show their performance in the next section, along with those of **A** and **L**.



(a) $S_g = L, S_h = L$ (b) $S_g = W, S_h = W$ (c) $S_g = W, S_h = L$

Figure 4.1: NMSE for different settings of the true GSO type S_g and the hypothesis GSO type S_h. The legend in each plot contains the considered GSOs for initializing the algorithm.

4.5. Numerical Results

In this section, we show some numerical results obtained for identifying different graph filters and GSOs **S**. In these experiments, we consider cases where the GSOs to identify are the weighted adjacency matrix and the Laplacian. To evaluate the correctness of our method, we first generate a random graph composed of $N = 30$ nodes with the GSP Toolbox [26] and construct from the graph the respective GSO **S** involved in the graph filter that generates the output data. We then generate $T = 500$ input graph signals $\{\mathbf{x}_t\}_{t=1}^T$ drawn from a standard normal distribution. By fixing the order of the graph filter to $K = 5$, we generate graph filter taps **h** following a

Gaussian distribution with zero mean and $\sigma = 3$. Finally, the output graph signals $\{\mathbf{y}_t\}_{t=1}^{T}$ are generated following (4.1).

In our experiments, we analyze two main aspects of the proposed method: *i)* the convergence of the algorithm, regardless the initial starting point; and *ii)* the similarity in terms of edge weights between the groundtruth GSO $\mathbf{S}$ and the identified one $\widehat{\mathbf{S}}$. To provide a fair comparison, we assume we do not know in advance the type of GSO that generate the network process, i.e., $\mathscr{S}$ is not completely known a priori. For this, we provide a guess of GSO type as input to Algorithm 1, and hope that a proper guess leads to a good fitting. In the sequel, we denote with S_g the type of GSO used to *generate* the data, and with S_h the type of GSO *hypothesized*. Both types of GSOs can assume the values W and L, indicating respectively the (weighted) adjacency matrix and the Laplacian matrix[3].

As performance metric for the error evaluation, we consider the normalized MSE (NMSE), defined as:

$$\mathrm{NMSE} = \frac{\sum_{t=1}^{T} \|\widehat{\mathbf{y}}_t - \mathbf{y}_t\|_2^2}{\sum_{t=1}^{T} \|\mathbf{y}_t\|_2^2} \tag{4.10}$$

where $\widehat{\mathbf{y}}_t$ is the predicted graph signal relative to the input $\mathbf{x}_t$.

Figure 4.1(a) shows the NMSE as function of the "cumulative" iteration number[4], for $\mathrm{S}_g = \mathrm{S}_h =$ L. Regardless of the starting point, we observe the non-increasing behavior of the NMSE, corroborating the global convergence of the algorithm. For this particular $(\mathrm{S}_g, \mathrm{S}_h)$ combination, $\mathbf{L}$ and $\mathbf{A}$ are the best performing starting points in terms of final NMSE, with $\mathbf{L}$ reaching convergence in just a few iterations. The sharp steps downwards, especially noticeable in the case of $\mathbf{A}$ are due to the update of the graph filter coefficients $\mathbf{h}$. In this case, the other initial points are not better that the straightforward initial guesses. Similar observations can be made from Fig. 4.1(b). A case of GSO mismatch is shown in Fig. 4.1(c), where the data are generated using the weighted adjacency matrix, but the algorithm is running based on the Laplacian hypothesis. As expected, the $\mathbf{A}$ matrix is the best starting point. Comparing Fig. 4.1(b) and Fig. 4.1(c), where the curves starting at $\mathbf{A}$ and $\mathbf{L}$ achieve the same NMSE, we note how in case of matched hypotheses, the GSOs generated through the generation procedure yield a lower error with respect to the mismatched counterpart.

As a quantitative measure of similarity between the groundtruth and the inferred weights, we report their Spearman correlation coefficient r_s, which is a non parametric measure of rank correlation. In particular, it answers the following question: *do edges with higher weight in the groundtruth GSO tend to have a higher weight in the inferred one?* A perfect Spearman correlation of +1 or −1 occurs when each of the variables is a perfect monotone function of the other. In our setting, $r_s = 0.74$ thus confirming a strong positive correlation of the two vectors.

[3]Note how we don't use here the bold notation, because both S_g and S_h are (textual) parameters of the algorithm, in contrast to the considered GSOs starting points that are effectively matrices.

[4]We count all the iterations of the algorithm up to its convergence. We sum in a cumulative manner the outer and the inner iterations of Algorithm 1.

Moreover, as depicted in the Q-Q plot of Fig. 4.2, the quantiles of the two vectors lie almost entirely on the straight line, allowing us to state that the weights of the two GSOs come approximately from the same distribution. For a qualitative and

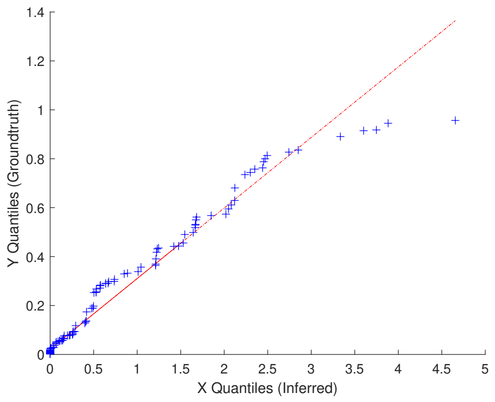
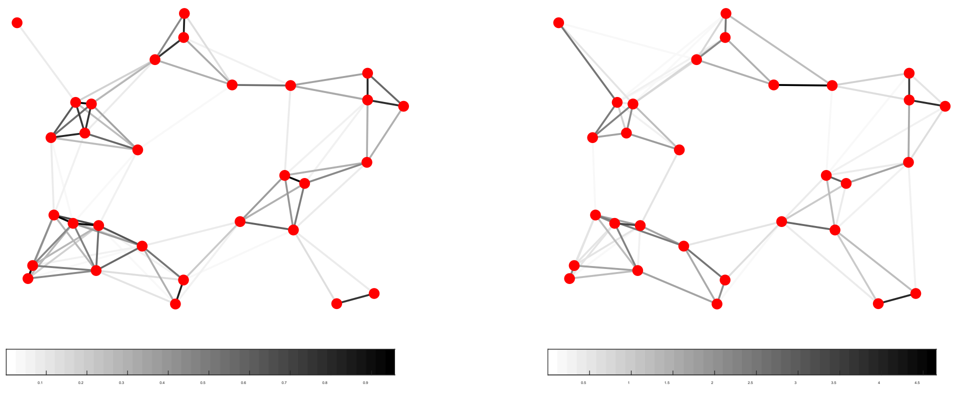


Figure 4.2: Q-Q plot of the weights of the groundtruth GSO and the weights of the inferred Laplacian for the case $S_g = L$, $S_h = L$

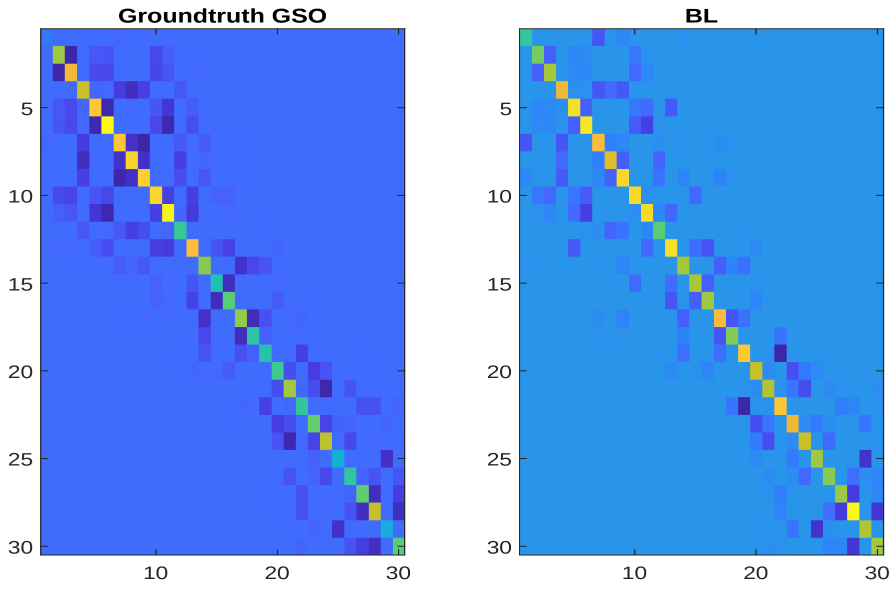
visual assessment of the method, in Fig. 4.3a and Fig. 4.3b we show, respectively, the graphs and the weighted sparsity pattern of the groundtruth and the learned Laplacian matrix (for $S_g = S_h = L$). We observe how, up to a scaling factor, the algorithm is able to give a larger weight to those edges that are also “important” in the original graph. All these considerations make us optimistic in the continuation of the development and the study of the proposed approach, driving us toward its application in more complex real-world scenarios.

4.6. CONCLUSION

In this work, we formulated and studied the problem of jointly estimating the filter coefficients and the graph shift operator (GSO) defining a graph filter that models the dynamics of signals defined over a network. In particular, motivated by practical scenarios, we exploited the a priori knowledge of the sparsity pattern of the network. We proposed an alternating-minimization approach, whose non-convex subproblem is handled through sequential convex programming methods. As shown in the numerical results, the proposed method is globally convergent and is able to identify the type of GSO used to generate the data. Quantitative statistical measures and qualitative graphics demonstrated the efficacy of the algorithm to assign higher values to those weights that are prominent in the real graph.



(a) Left: groundtruth graph. Right: inferred graph. The initial condition for the inferred graph was L. The darker the edge in the graph, the higher its value.



(b) Left: groundtruth Laplacian. Right: inferred Laplacian

Figure 4.3: (a) Graphs and (b) Laplacian matrix heatmaps for the case $S_g = S_h = L$

4.7. APPENDIX

To compute the derivative $\nabla_{\mathbf{S}} f(\mathbf{S})$, let us first expand the function $f(\mathbf{S})$[5]:

$$\begin{aligned} f(\mathbf{S}) &= \mathrm{tr}\left[(\mathbf{Y}-\mathbf{H}(\mathbf{h},\mathbf{S})\mathbf{X})(\mathbf{Y}-\mathbf{H}(\mathbf{h},\mathbf{S})\mathbf{X})^\top\right] \\ &= \mathrm{tr}\left(\mathbf{Y}\mathbf{Y}^\top\right) - 2\mathrm{tr}\left(\mathbf{H}\mathbf{X}\mathbf{Y}^\top\right) + \mathrm{tr}\left[\mathbf{H}^\top\mathbf{H}\mathbf{X}\mathbf{X}^\top\right] \\ &= \mathrm{tr}\left[\mathbf{Y}\mathbf{Y}^\top\right] - 2\sum_{k=0}^{K} h_k \mathrm{tr}\left[\mathbf{S}^k\mathbf{X}\mathbf{Y}^\top\right] \\ &\quad + \sum_{k_1}^{K}\sum_{k_2}^{K} h_{k_1}h_{k_2}\mathrm{tr}\left(\mathbf{S}^{k_1+k_2}\mathbf{X}\mathbf{X}^\top\right) \end{aligned}$$

Then

$$\nabla_{\mathbf{S}} f(\mathbf{S}) = -2\sum_{k=0}^{K} h_k \nabla_{\mathbf{S}}\mathrm{tr}\left[\mathbf{S}^k\mathbf{X}\mathbf{Y}^\top\right] + \sum_{k_1}^{K}\sum_{k_2}^{K} h_{k_1}h_{k_2}\nabla_{\mathbf{S}}\mathrm{tr}\left(\mathbf{S}^{k_1+k_2}\mathbf{X}\mathbf{X}^\top\right)$$

Because $\mathbf{S}$ is symmetric, we have to take into account its structure for the computation of the derivative of $f(\mathbf{S})$. Indeed, due to the matrix symmetry, the overall gradient can be decomposed in:

$$\nabla_{\mathbf{S}} f(\mathbf{S}) = \left[\frac{\partial f(\mathbf{S})}{\partial \mathbf{S}}\right] + \left[\frac{\partial f(\mathbf{S})}{\partial \mathbf{S}}\right]^\top - \mathrm{diag}\left[\frac{\partial f(\mathbf{S})}{\partial \mathbf{S}}\right].$$

Finally, because $\frac{\partial}{\partial \mathbf{S}}\mathrm{Tr}\left(\mathbf{S}^k\right) = k\left(\mathbf{S}^{k-1}\right)^\top$ and $\frac{\partial}{\partial \mathbf{S}}\mathrm{Tr}\left(\mathbf{B}\mathbf{S}^k\right) = \sum_{r=0}^{k-1}\left(\mathbf{S}^r\mathbf{B}\mathbf{S}^{k-r-1}\right)^\top$, we have that the component $\left[\partial f(\mathbf{S})/\partial\mathbf{S}\right]$ of the gradient is :

$$\begin{aligned} \frac{\partial f(\mathbf{S})}{\partial \mathbf{S}} &= -2\sum_{k=0}^{K} h_k \nabla_{\mathbf{S}}\mathrm{tr}\left[\mathbf{S}^k\mathbf{X}\mathbf{Y}^\top\right] + \sum_{k_1}^{K}\sum_{k_2}^{K} h_{k_1}h_{k_2}\nabla_{\mathbf{S}}\mathrm{tr}\left(\mathbf{S}^{k_1+k_2}\mathbf{X}\mathbf{X}^\top\right) \\ &= -2\sum_{k=1}^{K} h_k \left[\sum_{r=0}^{k-1}(\mathbf{S}^r\mathbf{X}\mathbf{Y}^\top\mathbf{S}^{k-r-1})^\top\right] + \sum_{k_1}^{K}\sum_{k_2}^{K} h_{k_1}h_{k_2} \sum_{r=0}^{k_1+k_2-1}(\mathbf{S}^r\mathbf{X}\mathbf{X}^\top\mathbf{S}^{k_1+k_2-r-1})^\top \end{aligned}$$

4.7.1. GSO CANDIDATE GENERATION

Let the model be $\mathbf{y} = \mathbf{H}(\mathbf{h},\mathbf{S})\mathbf{x}$ for some order K of the filter. Then, a $K=1$ approximation for the overall problem (4.2) is given by

$$\mathbf{y} \approx (\hat{h}_0^{(1)}\mathbf{I} + \hat{\mathbf{S}}_1)\mathbf{x}, \tag{4.11}$$

where $\hat{h}_0^{(1)}$ is the constant filter tap estimate related to the first order approximation, and $\hat{\mathbf{S}}_1 \in \mathscr{S}$ is the respective estimate for the GSO. We assume $\hat{h}_1^{(1)} = 1$ to avoid the scalar ambiguity that would otherwise arise in the term $\hat{h}_1\hat{\mathbf{S}}_1$. This also decouples the filter parameters from the GSO, both of which can be estimated by LLS. This way, a first GSO candidate $\mathbf{S}_1^{(0)}$ for the algorithm is found. Next, we consider a second order approximation of the model

[5]We set $\mathbf{h}^{(n)}$ to $\mathbf{h}$, and $\mathbf{H}(\mathbf{h},\mathbf{S})$ to $\mathbf{H}$, for the rest of the proof.

$$\mathbf{y} \approx (\hat{h}_0^{(2)}\mathbf{I} + \hat{\mathbf{S}}_2 + \hat{h}_2^{(2)}\mathbf{S}_1^{(0)2})\mathbf{x}, \tag{4.12}$$

where the variables are now $\hat{h}_0^{(2)}, \hat{h}_2^{(2)}$ and $\hat{\mathbf{S}}_2$, and we still assume the first filter tap $\hat{h}_1^{(2)}$ is equal to one. This again leads to a LLS problem which generates a second GSO candidate $\mathbf{S}_2^{(0)}$. We iterate this procedure by increasing the order of the filter at each step, and maintaining the term that is linear in the GSO variable $\mathbf{S}$. At the end, we have K initial GSO candidates $\mathbf{S}_1^{(0)}, \mathbf{S}_2^{(0)}, \ldots, \mathbf{S}_K^{(0)}$, which can be given as input to Algorithm 1.

REFERENCES

[1] A. Natali, M. Coutino, and G. Leus. “Topology-aware joint graph filter and edge weight identification for network processes”. In: *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE. 2020, pp. 1–6.

[2] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. “The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains”. In: *IEEE signal processing magazine* 30.3 (2013), pp. 83–98.

[3] A. Sandryhaila and J. M. Moura. “Discrete signal processing on graphs”. In: *IEEE transactions on signal processing* 61.7 (2013), pp. 1644–1656.

[4] A. Ortega, P. Frossard, J. Kovačević, J. M. Moura, and P. Vandergheynst. “Graph signal processing: Overview, challenges, and applications”. In: *Proceedings of the IEEE* 106.5 (2018), pp. 808–828.

[5] D. I. Shuman, P. Vandergheynst, D. Kressner, and P. Frossard. “Distributed signal processing via Chebyshev polynomial approximation”. In: *IEEE Transactions on Signal and Information Processing over Networks* 4.4 (2018), pp. 736–751.

[6] S. Segarra, A. G. Marques, and A. Ribeiro. “Optimal graph-filter design and applications to distributed linear network operators”. In: *IEEE Transactions on Signal Processing* 65.15 (2017), pp. 4117–4131.

[7] S. K. Narang, A. Gadde, and A. Ortega. “Signal processing techniques for interpolation in graph structured data”. In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE. 2013, pp. 5445–5449.

[8] 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: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2014, pp. 1115–1119.

[9] E. Isufi, P. D. Lorenzo, P. Banelli, and G. Leus. “Distributed Wiener-Based Reconstruction of Graph Signals”. In: *2018 IEEE Statistical Signal Processing Workshop (SSP)* (2018), pp. 21–25.

[10] M. Onuki, S. Ono, M. Yamagishi, and Y. Tanaka. “Graph signal denoising via trilateral filter on graph spectral domain”. In: *IEEE Transactions on Signal and Information Processing over Networks* 2.2 (2016), pp. 137–148.

[11] J. Ma, W. Huang, S. Segarra, and A. Ribeiro. “Diffusion filtering of graph signals and its use in recommendation systems”. In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Mar. 2016, pp. 4563–4567. DOI: 10.1109/ICASSP.2016.7472541.

[12] E. Isufi, A. Loukas, N. Perraudin, and G. Leus. “Forecasting time series with varma recursions on graphs”. In: *IEEE Transactions on Signal Processing* 67.18 (2019), pp. 4870–4885.

[13] A. Natali, E. Isufi, and G. Leus. “Forecasting Multi-Dimensional Processes Over Graphs”. In: *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2020, pp. 5575–5579.

[14] F. Gama, A. G. Marques, G. Leus, and A. Ribeiro. “Convolutional neural network architectures for signals supported on graphs”. In: *IEEE Transactions on Signal Processing* 67.4 (2018), pp. 1034–1049.

[15] M. Coutino, E. Isufi, and G. Leus. “Advances in distributed graph filtering”. In: *IEEE Transactions on Signal Processing* 67.9 (2019), pp. 2320–2333.

[16] S. Segarra, G. Mateos, A. G. Marques, and A. Ribeiro. “Blind identification of graph filters”. In: *IEEE Transactions on Signal Processing* 65.5 (2016), pp. 1146–1159.

[17] J. Liu, E. Isufi, and G. Leus. “Filter design for autoregressive moving average graph filters”. In: *IEEE Transactions on Signal and Information Processing over Networks* 5.1 (2018), pp. 47–60.

[18] G. Mateos, S. Segarra, A. G. Marques, and A. Ribeiro. “Connecting the dots: Identifying network structure via graph signal processing”. In: *IEEE Signal Processing Magazine* 36.3 (2019), pp. 16–43.

[19] X. Dong, D. Thanou, M. Rabbat, and P. Frossard. “Learning graphs from data: A signal representation perspective”. In: *IEEE Signal Processing Magazine* 36.3 (2019), pp. 44–63.

[20] G. B. Giannakis, Y. Shen, and G. V. Karanikolas. “Topology identification and learning over graphs: Accounting for nonlinearities and dynamics”. In: *Proceedings of the IEEE* 106.5 (2018), pp. 787–807.

[21] S. Segarra, A. G. Marques, G. Mateos, and A. Ribeiro. “Network topology inference from spectral templates”. In: *IEEE Transactions on Signal and Information Processing over Networks* 3.3 (2017), pp. 467–483.

[22] R. Shafipour, S. Segarra, A. G. Marques, and G. Mateos. “Identifying the topology of undirected networks from diffused non-stationary graph signals”. In: *arXiv preprint arXiv:1801.03862* (2018).

[23] R. Shafipour, S. Segarra, A. G. Marques, and G. Mateos. “DIRECTED NETWORK TOPOLOGY INFERENCE VIA GRAPH FILTER IDENTIFICATION”. In: *2018 IEEE Data Science Workshop (DSW)*. June 2018, pp. 210–214. DOI: 10.1109/DSW.2018.8439888.

[24] J. Mei and J. M. F. Moura. “Signal Processing on Graphs: Causal Modeling of Unstructured Data”. In: *IEEE Transactions on Signal Processing* 65.8 (Apr. 2017), pp. 2077–2092. ISSN: 1941-0476. DOI: 10.1109/TSP.2016.2634543.

[25] S. Boyd. “Sequential convex programming”. In: *Lecture Notes, Stanford University* (2008).

[26] N. Perraudin, J. Paratte, D. Shuman, L. Martin, V. Kalofolias, P. Vandergheynst, and D. K. Hammond. “GSPBOX: A toolbox for signal processing on graphs”. In: *ArXiv e-prints* (Aug. 2014). arXiv: 1408.5781 [cs.IT].

5
Learning Time-Varying Graphs from Online Data

This work proposes an algorithmic framework to learn time-varying graphs from online data. The generality offered by the framework renders it model-independent, i.e., it can be theoretically analyzed in its abstract formulation and then instantiated under a variety of model-dependent graph learning problems. This is possible by phrasing (time-varying) graph learning as a composite optimization problem, where different functions regulate different desiderata, e.g., data fidelity, sparsity or smoothness. Instrumental for the findings is recognizing that the dependence of the majority (if not all) data-driven graph learning algorithms on the data is exerted through the empirical covariance matrix, representing a sufficient statistic for the estimation problem. Its user-defined recursive update enables the framework to work in non-stationary environments, while iterative algorithms building on novel time-varying optimization tools explicitly take into account the temporal dynamics, speeding up convergence and implicitly including a temporal-regularization of the solution. We specialize the framework to three well-known graph learning models, namely, the Gaussian graphical model (GGM), the structural equation model (SEM), and the smoothness-based model (SBM), where we also introduce ad-hoc vectorization schemes for structured matrices (symmetric, hollows, etc.) which are crucial to perform correct gradient computations, other than enabling to work in low-dimensional vector spaces and hence easing storage requirements. After discussing the theoretical guarantees of the proposed framework, we corroborate it with extensive numerical tests in synthetic and real data.

5.1. Introduction

Learning network topologies from data is very appealing. On the *interpretable* side, the structure of a network reveals important descriptors of the network

Parts of this chapter have been published in the IEEE International Conference on Acoustics, Speech and Signal Processing (2021) [1] and in the IEEE Open Journal of Signal Processing (2022) [2].

itself, providing to humans a prompt and explainable decision support system; on the *operative* side, it is a requirement for processing and learning architectures operating on graph data, such as graph filters [3]. When this structure is not readily available from the application, a fundamental question is how to *learn* it from data. The class of problems and the associated techniques concerning the identification of a network structure (from data) are known as graph topology identification (GTI), graph learning, or network topology inference [4, 5].

While up to recent years the GTI problem has been focused on learning *static* networks, i.e., networks which do not change their structure over time, the pervasiveness of networks with a *time-varying* component has quickly demanded new learning paradigms. This is the case for biological networks [6], subject to changes due to genetic and environmental factors, or financial markets [7], subject to changes due to political factors, among others. In these scenarios, a static approach would fail in accounting for the temporal variability of the underlying structure, which is strategic to, e.g., detect anomalies or discover new emerging communities.

In addition, prior (full) data availability should not be considered as a given. In real time applications, data need to be processed on-the-fly with low latency to, e.g., identify and block cyber-attacks in a communication infrastructure, or fraudulent transactions in a financial network. Thus, another learning component to take into account, is the modality of data acquisition. Here, we consider the extreme case in which data are processed on-the-fly, i.e., a fully *online* scenario.

It is then clear how the necessity of having algorithms to learn time-varying topologies from online data is motivated by physical scenarios. For clarity, we elaborate on the three keywords - identification, time-varying and online - which constitute, other than the title of the present work, also its main pillars.

- *Identification/learning*: it refers to the (optimization) process of learning the graph topology.
- *Time-Varying/dynamic*: it refers to the temporal variability of the graph in its edges, in opposition to the static case.
- *Online/streaming*: it refers to the modality in which the data arrive and/or are processed, in opposition to a batch approach which makes use of the entire bulk of data.

This emphasis on the terminology is important to understand the differences between the different existing works, presented next.

5.1.1. Related Works

Static GTI has been originally addressed from a statistical viewpoint and only in the past decade under a graph signal processing (GSP) framework [8], in which different assumptions are made on how the data are coupled with the unknown topology; see [4, 5] for a tutorial. Only recently, dynamic versions of the static counterparts have been proposed. For instance, [9, 10] learn a sequence of graphs by enforcing a prior (smoothness or sparsity) on the edges of consecutive graphs; similarly, the work in

[11] extends the graphical Lasso [12] to account for the temporal variability, i.e., by estimating a sparse time-varying precision matrix. In addition to these works, the inference of causal relationships in the network structure, i.e., directed edges, has been considered in [13, 14]. See [15] for a review of dynamic topology inference approaches.

The mentioned approaches tackle the dynamic graph learning problem by means of a two-step approach: *i)* first, all the samples are collected and split into possibly overlapping windows; *ii)* only then the topology associated to each window is inferred from the data, possibly constrained to be similar to the adjacent ones. This modus-operandi fails to address the *online* (data-streaming) setting, where data have to be processed on-the-fly either due to architectural (memory, processing) limitations or (low latency) application requirements, such as real-time decision making.

This line of work has been freshly investigated by [16], which considers signals evolving according to a heat diffusion process, and by [17], which assumes the data are graph stationary [18]. In [19], the authors consider a vector autoregressive model to learn causality graphs by exploiting the temporal dependencies, while [20] proposes an online task-dependent (classification) graph learning algorithm, in which class-specific graphs are learned from labeled signals (training phase) and then used to classify new unseen data.

Differently from these works, our goal here is to provide a general (model-independent) algorithmic framework for time-varying GTI from online data that can be specialized to a variety of static graph learning problems. In particular, the generalization given by the framework enables us to render a static graph learning problem into its time-varying counterpart and to solve it via novel time-varying optimization techniques [21], providing a trade off between the solution accuracy and the velocity of execution. We introduce ad-hoc vectorization schemes for structured matrices to solve graph learning problems in the context of the Gaussian graphical model, the structural equation model, and the smoothness based model. All in all, a mature time-varying GTI framework for online data is yet to be conceived. This is our attempt to pave the way for a unified and general view of the problem, together with solutions to solve it.

5.1.2. Contributions

This paper proposes a general-purpose algorithmic blueprint which unifies the theory of learning time-varying graphs from online data. The specific contributions of this general framework are:

a) it is *model-independent*, i.e., it can be analyzed in its abstract form and then specialized under different graph learning models. We show how to instantiate three such models, namely, the Gaussian graphical model (GGM), the structural equation model (SEM) and the smoothness-based model (SBM);

b) it operates in *non-stationary* environments, i.e., when the data statistics change over time. This is possible by expressing the considered models in terms of the sample covariance matrix, which can be then updated recursively for each new streaming sample with a user-defined function, which discards past information.

c) it is *accelerated* through a prediction-correction strategy, which takes into account the time-dimension. Its iterative nature enables a trade-off between following the optimal solution (accuracy) and an approximate solution (velocity). It also exhibits an implicit regularization of the cost function due to the limited iteration budget at each time-instant, i.e., similar solutions at closed time instants are obtained.

Notation: we use $x(i)$ and $X(i,j)$ to denote the i-th entry of the column vector $\mathbf{x}$ and the ij-th entry of the matrix $\mathbf{X}$, respectively. Superscripts $^\top$ and † denote the transpose and the pseudoinverse of a matrix, respectively, while operators $\mathrm{tr}(\cdot)$ and $\mathrm{vec}(\cdot)$ denote the matrix trace and matrix vectorization, respectively. The vectors $\mathbf{0}$ and $\mathbf{1}$, and the matrix $\mathbf{I}$, denote the all-zeros vector, the all-ones vector, and the identity matrix, with dimension clarified in the context. The operators $\otimes$, $\odot$, $\oslash$ and $^\circ$ stand for Kronecker product, Hadamard (entry-wise) product, Hadamard (entry-wise) division and Hadamard (entry-wise) power, respectively. We have $[\,\cdot\,]_+ = \max(\mathbf{0},\cdot)$, where the maximum operates in an entry-wise fashion. Also, $\iota_{\mathcal{X}}(\cdot)$ is the indicator function for the convex set $\mathcal{X}$, for which holds $\iota_{\mathcal{X}}(\mathbf{x}) = 0$ if $\mathbf{x} \in \mathcal{X}$ and $+\infty$ otherwise. Given two functions $f(\cdot)$ and $g(\cdot)$, $f \circ g(\cdot)$ denotes their composition. A function $f(\cdot)$ with argument $\mathbf{x} \in \mathbb{R}^N$, which is parametrized by the time t, is denoted by $f(\mathbf{x};t)$. The gradient of the function $f(\mathbf{x};t)$ with respect to $\mathbf{x}$ at the point $(\mathbf{x};t)$ is denoted with $\nabla_{\mathbf{x}} f(\mathbf{x};t)$, while $\nabla_{\mathbf{xx}} f(\mathbf{x};t)$ denotes the Hessian evaluated at the same point. The time derivative of the gradient, denoted with $\nabla_{t\mathbf{x}} f(\mathbf{x};t)$, is the partial derivative of $\nabla_{\mathbf{x}} f(\mathbf{x};t)$ with respect to the time t, i.e., the mixed first-order partial derivative vector of the objective. Finally, $\|\cdot\|_p$ denotes the ℓ_p norm of a vector or, for a matrix, the ℓ_p norm of its vectorization. The Frobenius norm of a matrix is denoted with $\|\cdot\|_F$. Without any subscript, the norm $\|\cdot\|$ indicates the spectral norm.

5.2. Problem Formulation

In this section, we formalize the problem of learning graphs from data. In Section 5.2.1, we introduce the static graph topology inference problem, where we also recall three well-known models from the literature. Then, in Section 5.2.2 we formulate the (online) dynamic graph topology inference problem.

5.2.1. Graph Topology Identification

We consider data living in a non-Euclidean domain described by a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{S}\}$, where $\mathcal{V} = \{1, \ldots, N\}$ is the vertex set, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the edge set, and $\mathbf{S}$ is an $N \times N$ matrix encoding the topology of the graph. The matrix $\mathbf{S}$ is referred to as the graph shift operator (GSO) and typical instantiations include the (weighted) adjacency matrix $\mathbf{W}$ [8] and the graph Laplacian $\mathbf{L}$ [22]. By associating to each node $i \in \mathcal{V}$ a scalar value $x(i)$, we define $\mathbf{x} = [x(1), \ldots, x(N)]^\top \in \mathbb{R}^N$ as a *graph signal* mapping the node set to the set of real numbers.

Consider now the matrix $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_T]$ that stacks over the columns T graph signals generated from an unknown graph-dependent process $\mathscr{F}(\cdot)$; i.e., $\mathbf{X} = \mathscr{F}(\mathbf{S})$. Then, a GTI algorithm aims to learn the graph topology, i.e., to solve the "inverse" problem (not always well defined):

$$\mathbf{S} = \mathscr{F}^{-1}(\mathbf{X}). \tag{5.1}$$

The function $\mathscr{F}(\cdot)$ basically describes how the data are coupled with the graph and its knowledge is crucial. The data and the graph alone are insufficient to cast a meaningful graph learning problem. On one side, we need to know how the data depends on the graph from which they are generated. On the other side, we have to enforce some prior knowledge on the graph we want to learn.

Graph-data models. The choice of a data model is the forerunner of any GTI technique and, together with the graph-data coupling priors (e.g., smoothness, bandlimitedness) differentiates the different approaches. Due to their relevance for this work, we recall three widely used topology identification methods, namely the Gaussian graphical model [23], the structural equation model [24], and the smoothness-based model [25].

Gaussian graphical model (GGM) assumes each graph signal $\mathbf{x}_t$ is drawn from a multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ with mean $\boldsymbol{\mu}$ and positive-definite covariance matrix Σ. By setting the graph shift operator to be the precision matrix $\mathbf{S} = \Sigma^{-1}$, *graph learning in a GGM amounts to precision matrix estimation*, which in a maximum likelihood (MLE) sense can be formulated as:

$$\begin{aligned} &\underset{\mathbf{S}}{\text{minimize}} - \log\det(\mathbf{S}) + \operatorname{tr}(\mathbf{S}\hat{\Sigma}) \\ &\text{s. t.} \quad \mathbf{S} \in \mathbb{S}^N_{++} \end{aligned} \tag{5.2}$$

where $\hat{\boldsymbol{\Sigma}} = \frac{1}{T}\mathbf{X}\mathbf{X}^\top$ is the sample covariance matrix and $\mathbb{S}^N_{++}$ is the convex cone of positive-definite matrices. In this context, matrix $\mathbf{S}$ can be interpreted as the adjacency matrix (with self loops), although the problem can also be solved under some additional constraints forcing $\mathbf{S}$ to be a Laplacian [26].

Structural equation model (SEM) neglecting possible external inputs, and assuming an undirected graph, the SEM poses a linear dependence between the signal value $x_t(i)$ at node i and the signal values at some other nodes $\{x_t(j)\}_{j \neq i}$, representing the endogenous variables, i.e.,:

$$x_t(i) = \sum_{j \neq i} S(i,j) x_t(j) + e_t(i), \quad t = 1, \ldots, T \tag{5.3}$$

where $S(i,j)$ weights the influence that node j exerts on node i, and $e_t(i)$ represents unmodeled effects. In this view, with $\mathbf{S}$ encoding the graph connectivity, model (5.3) considers each node to be influenced only by its one-hop neighbors. In vector form, we can write (5.3) as:

$$\mathbf{x}_t = \mathbf{S}\mathbf{x}_t + \mathbf{e}_t, \quad t = 1, \ldots, T, \tag{5.4}$$

with $S(i,i)=0$, for $i=1,2,\dots,N$. Also, we consider $\mathbf{e}_t$ white noise with standard deviation σ_e. Graph learning under a SEM implies estimating matrix $\mathbf{S}$ by solving:

$$\begin{aligned} &\underset{\mathbf{S}}{\text{minimize}} \ \tfrac{1}{2T}\|\mathbf{X}-\mathbf{S}\mathbf{X}\|_F^2+g(\mathbf{S}),\\ &\text{s. t.} \quad \mathbf{S}\in\mathscr{S} \end{aligned} \tag{5.5}$$

where $\mathscr{S}=\{\mathbf{S}|\,\text{diag}(\mathbf{S})=\mathbf{0}, S(i,j)=S(j,i), i\neq j\}$, and $g(\mathbf{S})$ is a regularizer enforcing $\mathbf{S}$ to have specific properties; e.g., sparsity. In this context, matrix $\mathbf{S}$ is usually interpreted as the adjacency matrix of the network (without self loops). The first term of (5.5) can be equivalently rewritten as:

$$f(\mathbf{S})=\frac{1}{2T}\|\mathbf{X}-\mathbf{S}\mathbf{X}\|_F^2=\frac{1}{2}[\text{tr}(\mathbf{S}^2\hat{\mathbf{\Sigma}})-2\text{tr}(\mathbf{S}\hat{\mathbf{\Sigma}})+\text{tr}(\hat{\mathbf{\Sigma}})]. \tag{5.6}$$

which highlights its dependence on $\hat{\mathbf{\Sigma}}$.

Smoothness-based model (SBM) assumes each graph signal $\mathbf{x}_t$ to be smooth over the graph $\mathscr{G}$, where the notion of graph-smoothness is formally captured by the Laplacian quadratic form:

$$\text{LQ}_{\mathscr{G}}(\mathbf{x}_t):=\mathbf{x}_t^\top\mathbf{L}\mathbf{x}_t=\sum_{i\neq j}W(i,j)(x_t(i)-x_t(j))^2. \tag{5.7}$$

A low value of $\text{LQ}_{\mathscr{G}}(\mathbf{x}_t)$ suggests that adjacent nodes i and j have similar values $x_t(i)$ and $x_t(j)$ when the edge weight $W(i,j)$ is high.

Thus, the quantity:

$$\overline{\text{LQ}}_{\mathscr{G}}(\mathbf{X})=\frac{1}{T}\sum_{t=1}^{T}\text{LQ}_{\mathscr{G}}(\mathbf{x}_t)=\frac{1}{T}\text{tr}(\mathbf{X}^\top\mathbf{L}\mathbf{X})=\text{tr}(\mathbf{L}\hat{\mathbf{\Sigma}}) \tag{5.8}$$

represents the average signal smoothness on top of $\mathscr{G}$, which can be rewritten as the graph-dependent function:

$$f(\mathbf{S})=\text{tr}(\text{Diag}(\mathbf{S}\mathbf{1})\hat{\mathbf{\Sigma}})-\text{tr}(\mathbf{S}\hat{\mathbf{\Sigma}}) \tag{5.9}$$

with $\mathbf{S}=\mathbf{W}$. Building upon this quantity, graph learning under a graph smoothness prior can be casted as:

$$\begin{aligned} &\underset{\mathbf{S}}{\text{minimize}} \ f(\mathbf{S})+g(\mathbf{S})\\ &\text{s. t.} \ \mathbf{S}\in\mathscr{S} \end{aligned} \tag{5.10}$$

where the term $g(\mathbf{S})$ accommodates for additional topological properties (e.g., sparsity) and also helps avoiding the trivial solution $\mathbf{S}=\mathbf{0}$. The set $\mathscr{S}=\{\mathbf{S}|\,\text{diag}(\mathbf{S})=\mathbf{0}, S(i,j)=S(j,i)\geq 0, i\neq j\}$ encodes the topological structure, which coincides with the set of hollow symmetric matrices (i.e., with zeros on the diagonal) with positive entries.

Remark 1. *In [25], the authors express the smoothness quantity* (5.8) *in terms of the weighted adjacency matrix* $\mathbf{W}$ *and a matrix* $\mathbf{Z}\in\mathbb{R}_+^{N\times N}$ *representing the row-wise* *(squared) Euclidean distance matrix of* $\mathbf{X}$*; i.e.,* $\mathrm{tr}(\mathbf{X}^\top\mathbf{L}\mathbf{X}) = \frac{1}{2}\mathrm{tr}(\mathbf{W}\mathbf{Z}) = \frac{1}{2}\|\mathbf{W}\odot\mathbf{Z}\|_1$. *This formulation mainly brings the intuition that adding explicitly a sparsity term to the objective function would simply add a constant term to* $\mathbf{Z}$. *We favour* (5.9) *as a measure of graph signal smoothness since it fits within our framework, as will be clear soon. We emphasize however how the two formulations are equivalent, since* $\hat{\mathbf{\Sigma}}$ *can be directly expressed as a function of* $\mathbf{Z}$.

5.2.2. Online Time-Varying Topology Identification

When the graph topology changes over time, the changing interactions are represented by the sequence of graphs $\{\mathcal{G}_t = \{\mathcal{V}, \mathcal{E}_t, \mathbf{S}_t\}\}_{t=1}^{\infty}$, where $t \in \mathbb{N}_+$ is a discrete time index. This sequence of graphs, which is discrete in nature, can be interpreted as the sampling of some "virtual" continuous time-varying graph using the sampling period $h = 1$. To relate our expressions to existing literature, we will make the parameter h explicit in the formulas, yet it is important to remember that $h = 1$. Together with the graph sequence $\{\mathcal{G}_t\}_{t=1}^{\infty}$, we consider also streaming graph signals $\{\mathbf{x}_t\}_{t=1}^{\infty}$, such that signal $\mathbf{x}_t$ is associated to graph $\mathcal{G}_t$. At this point, we are ready to formalize the time-varying graph topology identification (TV-GTI) problem.

Problem statement. *Given an online sequence of graph signals* $\{\mathbf{x}_t\}_{t=1}^{\infty}$ *arising from an unknown time-varying network, the goal is to identify the time-varying graph topology* $\{\mathcal{G}_t\}_{t=1}^{\infty}$*; i.e., to learn the graph shift operator sequence* $\{\mathbf{S}_t\}_{t=1}^{\infty}$ *from* $\{\mathbf{x}_t\}_{t=1}^{\infty}$*. On top of this, to highlight the trade-off between accuracy and low-latency of the algorithm's solution.*

Mathematically, our goal is to solve the sequence of time-invariant problems:

$$\mathbf{S}_t^\star := \underset{\mathbf{S}}{\mathrm{argmin}}\, F(\mathbf{S}; t) \quad t = 1, 2, \ldots \tag{5.11}$$

where function $F(\cdot; t)$ is a time-varying cost function that depends on the data model [cf. Section 5.2.1], and the index t makes the dependence on time explicit, which is due to the arrival of new data. Although we can solve problem (5.11) for each t separately with (static) convex optimization tools, the need of a low-latency stream of solutions makes this strategy unappealing. This approach also fails to capture the inherent temporal structure of the problem, i.e, it does not exploit the prior time-dependent structure of the graph, which is necessary in time-critical applications.

To exploit also this temporal information, we build on recent advances of time-varying optimization [21, 27] and propose a general framework for TV-GTI suitable for non-stationary environments. The proposed approach operates on-the-fly and updates the solution as a new signal $\mathbf{x}_t$ becomes available. The generality of this formulation enables us to define a *template* for the TV-GTI problem, which can be specialized to a variety of static GTI methods. The only information required is the first-order (gradient) and possibly second-order (Hessian) terms of the function. In the next section, we lay down the mathematics of the proposed approach. The central idea is to follow the optimal time-varying solution of problem (5.11) with lightweight proximal operations [28], which can be additionally accelerated with a *prediction-correction* strategy. This strategy, differently from other adaptive optimization strategies such as least mean squares and recursive least squares, uses an evolution model to predict the solution, and observes new data to correct the predictions. The considerations of Section 5.3 will be then specialized to the different data models of Section 5.2.1 in Section 5.4, further analyzed theoretically in Section 5.5, and finally validated experimentally in Section 5.6.

5.3. Online Dynamic Graph Learning

To maintain our discussion general, we consider the *composite* time-varying function:

$$F(\mathbf{S};t) := f(\mathbf{S};t) + \lambda g(\mathbf{S};t) \tag{5.12}$$

where $f : \mathbb{R}^{N\times N} \times \mathbb{N}_+ \rightarrow \mathbb{R}$ is a smooth[1] strongly convex function [29] encoding a fidelity measure and $g : \mathbb{R}^{N\times N} \times \mathbb{N}_+ \rightarrow \mathbb{R}$ is a closed convex and proper function, potentially non differentiable, representing possible regularization terms. For instance, function $f(\cdot)$ can be the GGM objective function of (5.2), the SEM least-squares term of (5.5), or the SBM smoothness measure in (5.8).

Solving a time-varying optimization problem implies solving the *template* problem:

$$\mathbf{S}_t^\star := \underset{\mathbf{S}}{\operatorname{argmin}}\, f(\mathbf{S};t) + \lambda g(\mathbf{S};t) \quad \text{for } t = 1,2,\ldots \tag{5.13}$$

In other words, the goal is to find the sequence of optimal solutions $\{\mathbf{S}_t^\star\}_{t=1}^{\infty}$ of (5.13), which we will also call the *optimal trajectory*. However, solving exactly problem (5.13) in real time is infeasible because of the computational and time constraints. The exact solution may also be unnecessary since by itself it still approximates the true underlying time-varying graph. Under these considerations, an online algorithm that updates the approximate solution $\hat{\mathbf{S}}_{t+1}$ of (5.13) at time $t+1$, based on the former (approximate) solution $\hat{\mathbf{S}}_t$ is highly desirable for low complexity and fast execution[2].

5.3.1. Reduction

Instrumental for the upcoming analysis is to observe that the number of independent variables of the graph representation matrix plays an important role in terms of storage requirements, processing complexity and, most importantly, in the correct computations of function derivatives with respect to those variables. Thus, when considering structured matrices, such as symmetric, hollow or diagonal, we need to take into account their structure. We achieve this by ad-hoc vectorization schemes through duplication and elimination matrices, inspired by [30].

Consider a matrix $\mathbf{S} \in \mathbb{R}^{N\times N}$ and its corresponding "standard" vectorization $\text{vec}(\mathbf{S}) \in \mathbb{R}^{N^2}$. Depending on the specific structure of $\mathbf{S}$, different reduction and vectorization schemes can be adopted, leading to a lift from a matrix space to a vector space. The following spaces are of interest.

[1]We use the term smoothness for functions and the term graph-smoothness for graph signals.

[2]Problem (5.13) also endows the constrained case, in which the function $g(\cdot)$ comprises indicator functions associated to each constraint.

h-space. If $\mathbf{S}$ is symmetric, the number of independent variables is $k = N(N+1)/2$, i.e., the variables in its diagonal and its lower (equivalently, upper) triangular part. We can isolate these variables by representing matrix $\mathbf{S}$ with its *half-vectorization* form, which we denote as $\mathbf{s} = \text{vech}(\mathbf{S}) \in \mathbb{R}^k$. This isolation is possible by introducing the elimination matrix $\mathbf{E} \in \mathbb{R}^{k\times N^2}$ and the duplication matrix $\mathbf{D} \in \mathbb{R}^{N^2\times k}$ which respectively selects the independent entries of $\mathbf{S}$, i.e., $\mathbf{E}\text{vec}(\mathbf{S}) = \mathbf{s}$, and duplicates the entries of $\mathbf{s}$, i.e, $\mathbf{Ds} = \text{vec}(\mathbf{S})$. We call this vector space as the half-vectorization space (h-space).

hh-space. If $\mathbf{S}$ is symmetric and hollow, the number of independent variables is $l = N(N-1)/2$, i.e., the variables on its strictly lower (equivalently, upper) triangular part. In this case, we can represent matrix $\mathbf{S}$ in its *hollow half-vectorization* form, which we denote as $\mathbf{s} = \text{vechh}(\mathbf{S}) \in \mathbb{R}^l$. This reduction is achieved by applying the hollow elimination and duplication matrices $\mathbf{E}_h \in \mathbb{R}^{l\times N^2}$ and $\mathbf{D}_h \in \mathbb{R}^{N^2\times l}$, respectively, to the vectorization of $\mathbf{S}$. In particular, $\mathbf{E}_h$ extracts the variables of the strictly lower triangular part of the matrix, i.e., $\mathbf{s} = \mathbf{E}_h\text{vec}(\mathbf{S})$, while $\mathbf{D}_h$ duplicates the values and fills in zeros in the correct positions, i.e., $\text{vec}(\mathbf{S}) = \mathbf{D}_h\mathbf{s}$. We refer to the associated vector space as the hollow half-vectorization space (hh-space).

With the above discussion in place, we can now illustrate the general framework in terms of vector-dependent functions $f(\mathbf{s})$ for a vector $\mathbf{s}$, in contrast to matrix-dependent functions $f(\mathbf{S})$, simplifying exposition and notation. However, we underline that the information embodied in $\mathbf{S}$ and $\mathbf{s}$ is the same.

5.3.2. FRAMEWORK

We develop a prediction-correction strategy for problem (5.13) that starts from an estimate $\hat{\mathbf{s}}_t$ at time instant t, and *predicts* how this solution will change in the next time step $t+1$. This predicted topology is then *corrected* after a new datum $\mathbf{x}_{t+1}$ is available at time $t+1$. More specifically, the scheme has the following two steps:

(1) *Prediction*: at time t, an approximate function $\hat{F}(\mathbf{s}; t+1)$ of the true yet *unobserved* function $F(\mathbf{s}; t+1)$ is formed, using only information available at time t. Then, using this approximated cost, we derive an estimate $\mathbf{s}^\star_{t+1|t}$, of how the topology will be at time $t+1$, using only the information up to time t. This estimate is found by solving:

$$\mathbf{s}^\star_{t+1|t} := \underset{\mathbf{s}}{\text{argmin}}\, \hat{F}(\mathbf{s}; t+1). \tag{5.14}$$

To avoid solving (5.14) for each t, we find an estimate $\hat{\mathbf{s}}_{t+1|t}$ by applying P iterations of a problem-specific descent operator $\hat{\mathcal{T}}$ (e.g., gradient descent, proximal gradient) for which $\mathbf{s}^\star_{t+1|t} = \hat{\mathcal{T}}\mathbf{s}^\star_{t+1|t}$, i.e., $\mathbf{s}^\star_{t+1|t}$ is a fixed point of $\hat{\mathcal{T}}$. See Appendix 5.7 for possible instances of $\hat{\mathcal{T}}$.

In other words, problem (5.14) is solved recursively as:

$$\hat{\mathbf{s}}^{p+1} = \hat{\mathcal{T}}\hat{\mathbf{s}}^p, \quad p = 0, 1, \ldots, P-1 \tag{5.15}$$

with $\hat{\mathbf{s}}^0 = \hat{\mathbf{s}}_t$. Once P steps are performed, the predicted topology is set to $\hat{\mathbf{s}}_{t+1|t} = \hat{\mathbf{s}}^P$, which approximates the solution of (5.14) and, in turn, will be close to $\mathbf{s}^\star_{t+1}$ at time $t+1$.

For our framework, we consider a Taylor-expansion based prediction to approximate the first term of $F(\cdot; t+1)$, i.e., $f(\cdot; t+1)$ [cf. (5.12)], leading to the following quadratic function:

$$\hat{f}(\mathbf{s}; t+1) = \frac{1}{2}\mathbf{s}^\top \nabla_{\mathbf{ss}} f(\hat{\mathbf{s}}_t; t)\,\mathbf{s} + \big[\nabla_{\mathbf{s}} f(\hat{\mathbf{s}}_t; t) + \\ + h\nabla_{t\mathbf{s}} f(\hat{\mathbf{s}}_t; t) - \nabla_{\mathbf{ss}} f(\hat{\mathbf{s}}_t; t)\,\hat{\mathbf{s}}_t\big]^\top \mathbf{s} \tag{5.16}$$

where $\nabla_{\mathbf{ss}} f(\cdot) \in \mathbb{R}^{N\times N}$ is the Hessian matrix of $f(\cdot)$ with respect to $\mathbf{s}$ and $\nabla_{t\mathbf{s}} f(\cdot) \in \mathbb{R}^N$ is the partial derivative of the gradient of $f(\cdot)$ w.r.t. time t.

To approximate the second term of $F(\cdot; t+1)$, i.e., $g(\cdot; t+1)$ [cf. (5.12)], we use a one step-back prediction, i.e., $\hat{g}(\mathbf{s}; t+1) = g(\mathbf{s}; t)$. This implies that $\hat{g}(\cdot)$ does not depend on t, which in turn makes the constraint set and the regularization term independent of time, an assumption usually met in state-of-the-art topology identification [4]. Henceforth, we will omit this time dependency.

5

(2) *Correction*: at time $t+1$ the new data $\mathbf{x}_{t+1}$ and hence the cost function $F(\mathbf{s}; t+1)$ becomes available. Thus, we correct the prediction $\hat{\mathbf{s}}_{t+1|t}$ by solving the correction problem:

$$\mathbf{s}^\star_{t+1} := \underset{\mathbf{s}}{\operatorname{argmin}}\, F(\mathbf{s}; t+1). \tag{5.17}$$

Also in this case, we solve (5.17) with iterative methods to obtain an approximate solution $\hat{\mathbf{s}}_{t+1}$ by applying C iterations of an operator $\mathcal{T}$. In other words, the correction problem (5.17) is addressed through the recursion:

$$\hat{\mathbf{s}}^{c+1} = \mathcal{T}\hat{\mathbf{s}}^c, \quad c = 0, 1, \ldots, C-1 \tag{5.18}$$

with $\hat{\mathbf{s}}^0 = \hat{\mathbf{s}}_{t+1|t}$. Once the C steps are performed, the correction graph $\hat{\mathbf{s}}_{t+1}$ is set to $\hat{\mathbf{s}}_{t+1} = \hat{\mathbf{s}}^C$, which will approximate the solution $\mathbf{s}^\star_{t+1}$ of (5.17).

Algorithm 3 shows the pseudocode for the general online TV-GTI framework.

Remark 2. *We point out that the framework can adopt different approximation schemes, such as extrapolation-based techniques, and can also include time-varying constraint sets. The choice of approximation-scheme depends on the properties of the problem itself along with the required prediction accuracy. For an in-depth theoretical discussion regarding different prediction approaches and relative convergence results, refer to [31].*

5.4. Network Models and Algorithms

In this section, we specialize the proposed framework to the three static topology inference models discussed in Section 5.2.1. Notice that the data dependency of

Algorithm 3 Online Time-Varying Graph Topology Inference

```
Require: Feasible $\hat{\mathbf{S}}_0$, $f(\mathbf{S};t_0)$, $P$, $C$, operators $\hat{\mathscr{T}}$ and $\mathscr{T}$
 1: $\hat{\mathbf{s}}_0 \leftarrow$ ad-hoc vectorization of $\hat{\mathbf{S}}_0$
 2: for $t = 0,1,\dots$ do
 3:     // Prediction
 4:     Initialize the predicted variable $\hat{\mathbf{s}}^0 = \hat{\mathbf{s}}_t$
 5:     for $p = 0,1,\dots,P-1$ do
          Predict $\hat{\mathbf{s}}^{p+1}$ with (5.15)
 6:     end for
          Set the predicted variable $\hat{\mathbf{s}}_{t+1|t} = \hat{\mathbf{s}}^P$.
 7:     // Correction - time $t+1$: new data arrive
 8:     Initialize the corrected variable $\hat{\mathbf{s}}^0 = \hat{\mathbf{s}}_{t+1|t}$
 9:     for $c = 0,1,\dots,C-1$ do
          Predict $\hat{\mathbf{s}}^{c+1}$ with (5.18)
10:     end for
          Set the corrected variable $\hat{\mathbf{s}}_{t+1} = \hat{\mathbf{s}}^C$
11: end for
```

data-driven graph learning algorithms is exerted via the empirical covariance matrix $\hat{\mathbf{\Sigma}}$ of the graph signals; we have already shown this for the three considered models of Section 5.2.1. In other words, graph-dependent objective functions of the form $F(\mathbf{S})$ could be explicitly expressed through their parametrized version $F(\mathbf{S};\hat{\mathbf{\Sigma}})$. This rather intuitive, yet crucial observation, is central to render the proposed framework model-independent and adaptive, as explained next.

Non-stationarity. Relying on the explicit dependence of function $F(\cdot)$ on $\hat{\mathbf{\Sigma}}$ and envisioning non-stationary environments, we let the algorithm be adaptive by discarding past information. That is, function $F(\mathbf{S};t)$ in (5.12) can be written as $F(\mathbf{S};\hat{\mathbf{\Sigma}}_t)$, with $\hat{\mathbf{\Sigma}}_t$ the empirical covariance matrix, up to time t, with past data gradually discarded. This makes the framework adaptive and model-independent. The adaptive behavior can be shaped by, e.g., the exponentially-weighted moving average (EWMA) of the covariance matrix:

$$\hat{\mathbf{\Sigma}}_t = \gamma\hat{\mathbf{\Sigma}}_{t-1} + (1-\gamma)\mathbf{x}_t\mathbf{x}_t^\top \quad t = 1,2\dots \tag{5.19}$$

where the forgetting factor $\gamma \in (0,1)$ downweighs (for $\gamma \to 0$) or upweighs (for $\gamma \to 1$) past data contributions. For stationary environments, an option is the infinite-memory matrix covariance update $\hat{\mathbf{\Sigma}}_t = \frac{t-1}{t}\hat{\mathbf{\Sigma}}_{t-1} + \frac{1}{t}\mathbf{x}_t\mathbf{x}_t^\top$.

5.4.1. Time-Varying Gaussian Graphical Model

The GGM problem (5.2), adapted to a time-varying setting following template (5.13) leads to:

$$f(\mathbf{S};t) = -\log\det(\mathbf{S}) + \mathrm{tr}(\mathbf{S}\hat{\Sigma}_t) \tag{5.20a}$$

$$g(\mathbf{S};t) = \iota_{\mathscr{S}}(\mathbf{S}) \tag{5.20b}$$

where $\mathscr{S} = \mathbb{S}^N_{++}$. In this case $g(\cdot)$ encodes the constraint set of positive definite matrices and the regularization parameter is $\lambda = 1$.

Since $\mathbf{S}$ is symmetric, we use the half-vectorization $\mathbf{s} = \text{vech}(\mathbf{S}) \in \mathbb{R}^k$ to reduce the number of independent variables from N^2 to $k = N(N+1)/2$. Then, the gradient and the Hessian of the function $f(\cdot)$ in the h-space are respectively:

$$\nabla_{\mathbf{s}} f(\mathbf{s};t) = \mathbf{D}^\top \text{vec}(\hat{\Sigma}_t - \mathbf{S}^{-1}) \tag{5.21a}$$

$$\nabla_{\mathbf{ss}} f(\mathbf{s};t) = \mathbf{D}^\top (\mathbf{S} \otimes \mathbf{S})^{-1} \mathbf{D}. \tag{5.21b}$$

Likewise, the discrete-time derivative of the gradient is given by the partial mixed-order derivative [27]:

$$\nabla_{t\mathbf{s}} f(\mathbf{s};t) = \mathbf{D}^\top \text{vec}(\hat{\Sigma}_t - \hat{\Sigma}_{t-1}). \tag{5.22}$$

Note the Hessian term (5.21b) is time-independent, while the time-derivative of the gradient (5.22) is graph-independent.

Now, by defining $\hat{\mathbf{s}}_t := \text{vech}(\hat{\mathbf{S}}_\text{t}) \in \mathbb{R}^k$, we can particularize Algorithm 3 to:

5

- **Prediction:** with $\hat{\mathbf{s}}^0$ initialized as $\hat{\mathbf{s}}^0 = \hat{\mathbf{s}}_t$, the prediction update is :

$$\begin{aligned}\hat{\mathbf{s}}^{p+1} = \mathbb{P}_{\mathscr{S}}[\hat{\mathbf{s}}^p - 2\alpha_t(\nabla_{\mathbf{s}} f(\hat{\mathbf{s}}_t;t) + \\ + \nabla_{\mathbf{ss}} f(\hat{\mathbf{s}}_t;t)\left(\hat{\mathbf{s}}^p - \hat{\mathbf{s}}_t\right) + h\nabla_{t\mathbf{s}} f(\hat{\mathbf{s}}_t;t))]\end{aligned} \tag{5.23}$$

  for $p = 0,1,\ldots,P-1$, where α_t is a (time-varying) step size. Equation (5.23) entails a descent step along the approximate function $\hat{f}(\cdot;t+1)$ in (5.16), followed by the projection onto the convex set $\mathscr{S}$; see Appendix 5.7 for the definition of $\mathbb{P}_{\mathscr{S}}(\cdot)$. Then, the prediction $\hat{\mathbf{s}}_{t+1|t}$ is set to $\hat{\mathbf{s}}_{t+1|t} = \hat{\mathbf{s}}^P$.

- **Correction**: by setting $\hat{\mathbf{s}}^0 = \hat{\mathbf{s}}_{t+1|t}$, the correction update is:

$$\hat{\mathbf{s}}^{c+1} = \mathbb{P}_{\mathscr{S}}\left[\hat{\mathbf{s}}^c - \beta_t \nabla f(\hat{\mathbf{s}}^c;t+1)\right] \tag{5.24}$$

  for $c = 0,1,\ldots,C-1$, where β_t is a (time-varying) step size. Equation (5.24) entails a descent step along the true function $f(\cdot;t+1)$, followed by the projection onto the set $\mathscr{S}$. The correction $\hat{\mathbf{s}}_{t+1}$ is finally set to $\hat{\mathbf{s}}_{t+1} = \hat{\mathbf{s}}^C$.

The prediction step (5.23) instantiates (5.15) to $\hat{\mathscr{T}} = \mathbb{P}_{\mathscr{S}} \circ (I - \alpha_t \nabla_{\mathbf{s}} \hat{f})(\cdot)$, where $I(\cdot)$ is the identity function $I(\mathbf{s}) = \mathbf{s}$. Similarly, the correction step (5.24) instantiates (5.18) to $\mathscr{T} = \mathbb{P}_{\mathscr{S}} \circ (I - \beta_t \nabla_{\mathbf{s}} f)(\cdot)$. The overall computational complexity of one PC iteration is dominated by the matrix inversion and matrix multiplication, incurring a cost of $\mathcal{O}(N^3)$. A correction-only algorithm would also incur a cost of $\mathcal{O}(N^3)$ per iteration. See Appendix 5.9 for details.

5.4.2. Time-Varying Structural Equation Model

The SEM problem (5.5), adapted to a time-varying setting with sparsity-promoting regularizer, leads to [cf. (5.13)]:

$$f(\mathbf{S};t) = \frac{1}{2}[\text{tr}(\mathbf{S}^2\hat{\mathbf{\Sigma}}_t) - 2\text{tr}(\mathbf{S}\hat{\mathbf{\Sigma}}_t) + \text{tr}(\hat{\mathbf{\Sigma}}_t)] \tag{5.25a}$$

$$g(\mathbf{S};t) = \|\mathbf{S}\|_1 + \iota_{\mathscr{S}}(\mathbf{S}) \tag{5.25b}$$

where $\mathscr{S} = \{\mathbf{S} \in \mathbb{S}^N | \operatorname{diag}(\mathbf{S}) = \mathbf{0}, S(i,j) = S(j,i), i \neq j\}$ is the set of hollow symmetric matrices, and $\|\mathbf{S}\|_1 = \|\operatorname{vec}(\mathbf{S})\|_1$. Since $\mathbf{S}$ is symmetric and hollow, we operate on the hh-space to make the problem unconstrained and reduce the number of independent variables from N^2 to $l = N(N-1)/2$, through its hollow half-vectorization form $\mathbf{s} = \operatorname{vechh}(\mathbf{S}) \in \mathbb{R}^l$. In the hh-space, equations (5.25a) and (5.25b) become:

$$f(\mathbf{s};t) = \frac{1}{2}\mathbf{s}^\top \mathbf{Q}_t \mathbf{s} - 2\mathbf{s}^\top \hat{\boldsymbol{\sigma}}_t + \frac{1}{2}\hat{\sigma}_t \tag{5.26a}$$

$$g(\mathbf{s};t) = 2\|\mathbf{s}\|_1 \tag{5.26b}$$

where $\mathbf{Q}_t := \mathbf{D}_h^\top(\hat{\boldsymbol{\Sigma}}_t \otimes \mathbf{I})\mathbf{D}_h$ with $\otimes$ denoting the Kronecker product, $\hat{\boldsymbol{\sigma}}_t = \operatorname{vechh}(\hat{\boldsymbol{\Sigma}}_t)$, and $\hat{\sigma}_t = \operatorname{tr}(\hat{\boldsymbol{\Sigma}}_t)$. Since $\mathbf{Q}_t \succeq 0$, (5.26a) is convex.

To solve the time-varying SEM (TV-SEM) problem, we derive the gradient and the Hessian of function $f(\cdot)$ in the hh-space as:

$$\nabla_{\mathbf{s}} f(\mathbf{s};t) = \mathbf{Q}_t \mathbf{s} - 2\hat{\boldsymbol{\sigma}}_t \tag{5.27a}$$

$$\nabla_{\mathbf{ss}} f(\mathbf{s};t) = \mathbf{Q}_t \tag{5.27b}$$

Notice here how the Hessian is time-varying and independent on $\mathbf{s}$, differently from the GGM case. The time derivative of the gradient is given by the partial mixed-order derivative:

$$\nabla_{t\mathbf{s}} f(\mathbf{s};t) = \frac{1}{h}[(\mathbf{Q}_t - \mathbf{Q}_{t-1})\mathbf{s} - 2(\hat{\boldsymbol{\sigma}}_t - \hat{\boldsymbol{\sigma}}_{t-1})] \tag{5.28}$$

Now, by defining $\hat{\mathbf{s}}_t := \operatorname{vechh}(\hat{\mathbf{S}}_t) \in \mathbb{R}^l$, we can particularize Algorithm 3 to:

- **Prediction:** set $\hat{\mathbf{s}}^0 = \hat{\mathbf{s}}_t$. Then, the prediction is the proximal-gradient update:

$$\begin{aligned}\mathbf{u}^p = \hat{\mathbf{s}}^p - \alpha_t[&\nabla_{\mathbf{s}} f(\hat{\mathbf{s}}_t;t) + \\ &+ \nabla_{\mathbf{ss}} f(\hat{\mathbf{s}}_t;t)\left(\hat{\mathbf{s}}^p - \hat{\mathbf{s}}_t\right) + h\nabla_{t\mathbf{s}} f(\hat{\mathbf{s}}_t;t)]\end{aligned} \tag{5.29a}$$

$$\hat{\mathbf{s}}^{p+1} = \operatorname{sign}(\mathbf{u}^p) \odot [|\mathbf{u}^p| - 2\alpha_t\lambda\mathbf{1}]_+ \tag{5.29b}$$

  for $p = 0,\ldots,P$. Equation (5.29a) entails a descent step along the approximate function $\hat{f}(\cdot;t+1)$ in (5.16), followed by the non-negative soft-thresholding operator in (5.29b), which sets to zero all the (negative) edge weights of the graph obtained after the gradient descent in (5.29a). See Appendix 5.7 for the formal definition of proximal operator, leading to (5.29a) and (5.29b). The final prediction $\hat{\mathbf{s}}_{t+1|t}$ is set to $\hat{\mathbf{s}}_{t+1|t} = \hat{\mathbf{s}}^P$.

- **Correction:** set $\hat{\mathbf{s}}^0 = \hat{\mathbf{s}}_{t+1|t}$. Then, the correction is the proximal-gradient update:

$$\mathbf{u}^c = \hat{\mathbf{s}}^c - \beta_t \nabla f(\hat{\mathbf{s}}^c;t+1) \tag{5.30a}$$

$$\hat{\mathbf{s}}^{c+1} = \operatorname{sign}(\mathbf{u}^c) \odot [|\mathbf{u}^c| - 2\beta_t\lambda\mathbf{1}]_+ \tag{5.30b}$$

  for $c = 0,\ldots,C-1$. Equation (5.30a) entails a descent step along the true function $f(\cdot;t+1)$, followed by the non-negative soft-thresholding operator in (5.30b). Finally, $\hat{\mathbf{s}}_{t+1} = \hat{\mathbf{s}}^C$.

The prediction step (5.29) instantiates (5.15) to $\hat{\mathcal{T}} = \text{prox}_{\lambda g, \alpha_t} \circ (I - \alpha_t \nabla_{\mathbf{s}} \hat{f})(\cdot)$. Similarly, the correction step (5.30) instantiates (5.18) to $\mathcal{T} = \text{prox}_{\lambda g, \beta_t} \circ\ (I - \beta_t \nabla_{\mathbf{s}} f)(\cdot)$. The overall computational complexity of one PC iteration is dominated by the computation of matrix $\mathbf{Q}_t$, incurring a cost of $\mathcal{O}(N^3)$. A correction-only algorithm would also incur a cost of $\mathcal{O}(N^3)$ per iteration. See Appendix 5.9 for details.

5.4.3. Time-Varying Smoothness-based Model

The SBM model (5.10) adapted to a time-varying setting is:

$$f(\mathbf{S}; t) = \text{tr}(\text{Diag}(\mathbf{S1})\hat{\mathbf{\Sigma}}_t) - \text{tr}(\mathbf{S}\hat{\mathbf{\Sigma}}_t) \tag{5.31a}$$

$$g(\mathbf{S}; t) = \frac{\lambda_1}{4} \|\mathbf{S}\|_F^2 - \lambda_2 \mathbf{1}^\top \log(\mathbf{S1}) + \iota_{\mathcal{S}}(\mathbf{S}) \tag{5.31b}$$

where $\mathcal{S} = \{\mathbf{S} \in \mathbb{S}^N | \text{diag}(\mathbf{s}) = \mathbf{0}, S(i,j) = S(j,i) \geq 0, i \neq j\}$ is the set of hollow symmetric matrices. The log barrier term $\log(\mathbf{S1})$ is applied entry-wise and forces the nodes degree vector $\mathbf{d} = \mathbf{S1}$ to be positive while avoiding the trivial solution. The Frobenius norm term $\|\mathbf{S}\|_F^2$ controls the sparsity of the graph.

By operating in the hh-space, equations (5.31a) and (5.31b) become[3]:

$$f(\mathbf{s}; t) = \mathbf{s}^\top (\mathbf{K}^\top \hat{\boldsymbol{\sigma}}_d - 2\hat{\boldsymbol{\sigma}}_t) - \lambda_2 \mathbf{1}^\top \log(\mathbf{Ks}) + \frac{\lambda_1}{2} \|\mathbf{s}\|^2 \tag{5.32a}$$

$$g(\mathbf{s}; t) = \iota_{\mathbb{R}_+}(\mathbf{s}) \tag{5.32b}$$

where $\mathbf{K} \in \{0,1\}^{N \times l}$ is the binary matrix such that $\mathbf{d} = \mathbf{S1} = \mathbf{Ks}$, $\hat{\boldsymbol{\sigma}}_d = \text{diag}(\hat{\mathbf{\Sigma}}_t)$ and $\hat{\boldsymbol{\sigma}}_t = \text{vechh}(\hat{\mathbf{\Sigma}}_t)$.

To apply the proposed framework to solve the time-varying SBM (TV-SBM) problem, we derive the gradient and the Hessian of function $f(\cdot)$ in the hh-space as follows:

$$\nabla_{\mathbf{s}} f(\mathbf{s}; t) = \lambda_1 \mathbf{s} - \lambda_2 \mathbf{K}^\top (\mathbf{1} \oslash \mathbf{Ks}) + \mathbf{z}_t \tag{5.33a}$$

$$\nabla_{\mathbf{ss}} f(\mathbf{s}; t) = \lambda_1 \mathbf{I} + \lambda_2 \mathbf{K}^\top \text{Diag}(\mathbf{1} \oslash (\mathbf{Ks})^{\circ 2}) \mathbf{K} \tag{5.33b}$$

where $\oslash$ and $^\circ$ represent the Hadamard division and power, respectively. The time derivative of the gradient is given by the partial mixed-order derivative:

$$\nabla_{t\mathbf{s}} f(\mathbf{s}; t) = \frac{1}{h}(\mathbf{z}_t - \mathbf{z}_{t-1}) \tag{5.34}$$

where $\mathbf{z}_t = \mathbf{K}^\top \hat{\boldsymbol{\sigma}}_d - 2\hat{\boldsymbol{\sigma}}_t$. Now, by defining $\hat{\mathbf{s}}_t := \text{vechh}(\hat{\mathbf{S}}_t) \in \mathbb{R}^l$, we can particularize Algorithm 3 to:

- **Prediction:** with $\hat{\mathbf{s}}^0$ initialized as $\hat{\mathbf{s}}^0 = \hat{\mathbf{s}}_t$, the prediction update is:

$$\begin{aligned} \hat{\mathbf{s}}^{p+1} = \mathbb{P}_{\mathbf{s} \geq \mathbf{0}}[\hat{\mathbf{s}}^p - 2\alpha_t (\nabla_{\mathbf{s}} f(\hat{\mathbf{s}}_t; t) + \\ + \nabla_{\mathbf{ss}} f(\hat{\mathbf{s}}_t; t) \left(\hat{\mathbf{s}}^p - \hat{\mathbf{s}}_t\right) + h \nabla_{t\mathbf{s}} f(\hat{\mathbf{s}}_t; t))] \end{aligned} \tag{5.35}$$

[3]We move the log-barrier and Frobenius norm terms of $g(\cdot)$ function (5.31b) into the $f(\cdot)$ function to fit the structure of the general template.

for $p = 0, 1, \ldots, P-1$. Equation (5.35) entails a descent step along the approximate function $\hat{f}(\cdot; t+1)$ in (5.16), followed by the projection onto the non-negative orthant. Then, the prediction $\hat{\mathbf{s}}_{t+1|t}$ is set to $\hat{\mathbf{s}}_{t+1|t} = \hat{\mathbf{s}}^P$.

- **Correction**: by setting $\hat{\mathbf{s}}^0 = \hat{\mathbf{s}}_{t+1|t}$, the correction update is:

$$\hat{\mathbf{s}}^{c+1} = \mathbb{P}_{\mathbf{s}\geq\mathbf{0}}\left[\hat{\mathbf{s}}^c - \beta_t \nabla f(\hat{\mathbf{s}}^c; t+1)\right], \tag{5.36}$$

  for $c = 0, 1, \ldots, C-1$. Equation (5.36) entails a descent step along the true function $f(\cdot; t+1)$, followed by the projection onto the non-negative orthant. Finally, $\hat{\mathbf{s}}_{t+1} = \hat{\mathbf{s}}^C$.

The prediction step (5.35) instantiates (5.15) to $\hat{\mathscr{T}} = \mathbb{P}_{\mathbf{s}\geq\mathbf{0}} \circ (I - \alpha_t \nabla_{\mathbf{s}} \hat{f})(\cdot)$. Similarly, the correction step (5.36) instantiates (5.18) to $\mathscr{T} = \mathbb{P}_{\mathbf{s}\geq\mathbf{0}} \circ (I - \beta_t \nabla_{\mathbf{s}} f)(\cdot)$. The overall computational complexity per iteration is dominated by the computation of the gradient $\nabla_{\mathbf{s}} f(\mathbf{s}; t)$ (or the Hessian if $P > 1$), incurring a cost of $\mathcal{O}(N^2)$ (or $\mathcal{O}(N^3)$ if $P > 1$). See Appendix 5.9 for details.

5.5. Convergence Analysis

In this section, we first discuss the convergence of Algorithm 3 and the associated error bounds. As solver we consider the proximal gradient $\hat{\mathscr{T}} = \mathscr{T} = \text{prox}_{g,\rho} \circ (I - \rho \nabla_{\mathbf{s}} f)(\cdot)$ [32, 33]. Then, we show how the parameters of the three introduced models are involved in the bounds. To ease notation, we use $\mathbf{s} \in \mathbb{R}^p$ to indicate the vectorization of matrix variable $\mathbf{S} \in \mathbb{R}^{N \times N}$ [cf. Section 5.3.1].

For this analysis, we need the following mild assumptions.

***Assumption* 1.** *The function $f : \mathbb{R}^p \times \mathbb{N}_+ \to \mathbb{R}$ is m-strongly convex and L-smooth uniformly in t, i.e., $m\mathbf{I} \preceq \nabla_{\mathbf{ss}} f(\mathbf{s}; t) \preceq L\mathbf{I}$, $\forall\, \mathbf{s}, t$, while the function $g : \mathbb{R}^p \times \mathbb{N}_+ \to \mathbb{R} \cup \{+\infty\}$ is closed convex and proper, or $g(\cdot; t) = 0$, for all $t \in \mathbb{N}_+$.*

This guarantees that problem (5.13) admits a unique solution for each time instant, which in turn guarantees uniqueness of the solution trajectory $\{\mathbf{s}_t^\star\}_{t=1}^\infty$.

***Assumption* 2.** *The gradient of function $f(\cdot)$ has bounded time derivative, i.e. $\exists\, C_0 > 0$ such that $\|\nabla_{t\mathbf{s}} f(\mathbf{s}; t)\| \leq C_0\ \forall\, \mathbf{s} \in \mathbb{R}^p, t \in \mathbb{N}_+$.*

This guarantees that the solution trajectory is Lipschitz in time.

***Assumption* 3.** *The predicted function $\hat{f}(\cdot; t+1)$ is m-strongly convex and L-smooth uniformly in t; and $\hat{g}(\cdot; t+1)$ is closed, convex and proper.*

This implies that the prediction problem (5.14) belongs to the same class as the original problem, i.e., the functions of the two problems share the same strong convexity and Lipschitz constants m and L. Therefore, the same solver can be applied for the prediction and correction steps, i.e., $\hat{\mathscr{T}} = \mathscr{T}$.

***Assumption* 4.** *The matrix $\mathbf{S}$ of (5.12) has finite entries, i.e., $-\infty < S(i, j) < +\infty$, for all i, j.*

This guarantees $\|\mathbf{S}\| < +\infty$, i.e., $\mathbf{S}$ is a bounded operator, and it holds in practical scenarios. In particular, it is known that (finite) weighted graphs exhibit bounded eigenvalues, see [34][35]. Notably, if $\mathbf{S}$ is a normalized Laplacian, then $\|\mathbf{S}\| = 2$.

Similarly, assumptions 1-3 are mild and hold for the considered models, as we show next.

Proposition 1. *The three considered models of Section 5.4 can be m-strongly convex and L-smooth uniformly in t, for some scalar m and L, as supported by the following claims.*

Claim 1. *Denote with $\xi > 0$ and $0 < \chi < \infty$ the minimum and maximum admissible eigenvalues of the precision matrix $\mathbf{S}$, respectively; i.e., consider the set $\mathscr{S} = \{\mathbf{S} \in \mathbb{S}^N_{++} | \xi\mathbf{I} \preceq \mathbf{S} \preceq \chi\mathbf{I}\}$. Then, for the TV-GGM function $f(\cdot;t)$ in (5.20a), it holds:*

$$m = 1/\chi \qquad L = 2/\xi. \tag{5.37}$$

Claim 2. *Denote with λ_{min} and λ_{max} the smallest and highest eigenvalues for the set of empirical covariance matrices obtained with graph signals obeying (5.4). Then, for the TV-SEM function $f(\cdot;t)$ in (5.26a), it holds:*

$$m = \lambda_{min} \qquad L = 2\lambda_{max}. \tag{5.38}$$

Claim 3. *Consider the TV-SBM function $f(\cdot;t)$ in (5.32a), and recall that the log-barrier term avoids isolated vertices, i.e., $\mathbf{d} > \mathbf{0}$. Denote with $d_{min} > 0$ the minimum degree of the GSO search space. Under these assumptions, it holds:*

$$m = 2\lambda_1 \qquad L = 2\lambda_2(N-1)d_{min}^{-2}. \tag{5.39}$$

See Appendix 5.8 for a proof of Claim 1-3.

Thus, Assumption 1 holds since the Hessian of $f(\cdot;t)$ is bounded over time and $g(\cdot;t)$ is closed, convex and proper by problem construction; Assumption 2 holds since $\nabla_{t\mathbf{s}} f(\mathbf{s};t)$ is the difference between bounded vectors which involve covariance matrices not too different from each other (one is the rank-one update of the other), which is finite as long as the graph signals are bounded, see (5.19) and, e.g., (5.34). Assumption 3 holds since $\hat{f}(\cdot;t+1)$ is a quadratic approximation of $f(\cdot;t)$ [cf. (5.16)] and $\hat{g}(\cdot;t+1) = g(\cdot;t)$, thus inheriting the properties of $f(\cdot;t)$ and $g(\cdot;t)$, which satisfy Assumption 1.

With this in place, we are now ready to show two different error bounds incurred during the prediction and correction steps performed by Algorithm 3, describing its sub-optimality as function of the model and algorithm's parameters. First, we show the error bound between the optimal prediction solution $\mathbf{s}^\star_{t+1|t}$ and the associated optimal correction $\mathbf{s}^\star_{t+1}$, which solve problems (5.14) and (5.17), respectively.

Proposition 2. *Let Assumptions 1-3 hold. Consider also the Taylor expansion based prediction (5.16) for $f(\cdot;t)$ and the one-step back prediction for $g(\cdot;t)$. Then, the* *distance between the optimal prediction solution* $\mathbf{s}^\star_{t+1|t}$*, solving problem* (5.14)*, and the associated optimal correction* $\mathbf{s}^\star_{t+1}$*, solving problem* (5.17)*, is upper bounded by:*

$$\|\mathbf{s}^\star_{t+1|t} - \mathbf{s}^\star_{t+1}\| \le \frac{2L}{m}\|\hat{\mathbf{s}}_t - \mathbf{s}^\star_t\| + \frac{2C_0 h}{m}(1+\frac{L}{m}) \tag{5.40}$$

where $\hat{\mathbf{s}}_t$ *is the approximate solution of the correction problem* (5.17) *at time* t*.*

Proof. Follows from [31, Lemma 4.2] in which constant $D_0 = 0$ by considering a static function $g(\cdot)$. ■

This bound enables us to measure how far the prediction is from the true corrected topology at time $t+1$. It depends on the estimation error $\hat{\mathbf{s}}_t - \mathbf{s}^\star_t$ achieved at time t, the ratio L/m and the variability of the function gradient $\nabla_{t\mathbf{s}} f(\mathbf{s};t)$. The bound suggests that a small gap can be achieved if *i)* the ratio L/m is small, which for the three considered models translates in having a small condition number for the involved covariance matrices or GSOs; and *ii)* the time-gradient $\nabla_{t\mathbf{s}} f(\mathbf{s};t)$ at consecutive time steps does not change significantly, which holds when the considered models have similar covariance matrices at adjacent time instants, i.e., the data statistics do not change too rapidly (see e.g. (5.22) and (5.28)).

Finally, we bound the error sequence $\{\|\hat{\mathbf{s}}_t - \mathbf{s}^\star_t\|_2, t = 1,2,\ldots\}$ achieved by Algorithm 3 by means of the following non-asymptotic performance guarantee, which is an adaptation of [31, Proposition 5.1].

Theorem 4. *Let Assumptions 1 and 3 hold, and consider two scalars* $\{d_t, \phi_t\} \in \mathbb{R}_+$ *such that:*

$$\|\mathbf{s}^\star_{t+1} - \mathbf{s}^\star_t\| \le d_t \quad \textit{and} \quad \|\mathbf{s}^\star_{t+1|t} - \mathbf{s}^\star_{t+1}\| \le \phi_t \tag{5.41}$$

for any $t \in \mathbb{N}_+$*. Let also the prediction and correction steps use the same step-sizes* $\rho_t = \alpha_t = \beta_t$*. Then, by employing* P *prediction and* C *correction steps with the proximal gradient operator* $\mathcal{T} = \text{prox}_{g,\rho_t} \circ (I - \rho_t \nabla_{\mathbf{s}} f)(\cdot)$*, the sequence of iterates* $\{\hat{\mathbf{s}}_t\}$ *generated by Algorithm 3 satisfies:*

$$\|\hat{\mathbf{s}}_{t+1} - \mathbf{s}^\star_{t+1}\|_2 \le q_t^C (q_t^P \|\hat{\mathbf{s}}_t - \mathbf{s}^\star_t\| + q_t^P d_t + (1+q_t^P)\phi_t) \tag{5.42}$$

where $q_t = \max\{|1-\rho_t m_t|, |1-\rho_t L_t|\} \in (0,1)$ *is the contraction coefficient [36].*

Proof. Follows from [31, Proposition 5.1] and [31, Lemma 2.5], with variables $\lambda = q_t$ and $\chi = \beta = 1$. ■

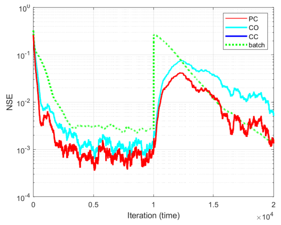
Theorem (5.42) states that the sequence of estimated graphs $\{\mathbf{s}_t\}_{t\in\mathbb{N}_+}$ hovers around the optimal trajectory $\{\mathbf{s}^\star_t\}_{t\in\mathbb{N}_+}$ with a distance depending on: *i)* the numbers P and C of iterations; *ii)* the estimation error achieved at the previous time instant $\|\hat{\mathbf{s}}_t - \mathbf{s}^\star_t\|$; and *iii)* the quantities d_t and ϕ_t. Moreover, (5.42) is a contraction (i.e., $q_t^{C+P} < 1$) when $\rho_t < 2/L_t$; in this case the initial starting point $\hat{\mathbf{s}}_0$ does not influence the error $\hat{\mathbf{s}}_{t+1} - \mathbf{s}^\star_{t+1}$ asymptotically, since the first term in (5.42) vanishes. However, the terms d_t and ϕ_t keep impacting the error also asymptotically, as long as the problem is time-varying; if the problem becomes static, i.e., the solution stops varying, then $d_t = \phi_t = 0$, and the overall error asymptotically goes to zero.

5.6. Numerical Results

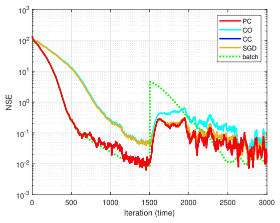
In this section, we show with numerical results how Algorithm 3, specialized to the three models (TV-GGM, TV-SEM, TV-SBM), can track the offline solution (5.13) obtained by the respective instantiations. For all the experiments, we initialize the empirical covariance matrix $\hat{\boldsymbol{\Sigma}}_0$ with some samples acquired prior to the analysis. We consider $P=1$ prediction steps and $C=1$ correction steps, which is the challenging setting of having the minimum iteration budget for streaming scenarios. We measure the convergence of Algorithm 3 via the normalized squared error (NSE) between the algorithm's estimate $\hat{\mathbf{s}}_t$ and the optimal (offline) solution $\mathbf{s}_t^\star$:

$$\mathrm{NSE}(\hat{\mathbf{s}}_t, \mathbf{s}_t^\star) = \frac{\|\hat{\mathbf{s}}_t - \mathbf{s}_t^\star\|_2^2}{\|\mathbf{s}_t^\star\|_2^2}. \tag{5.43}$$

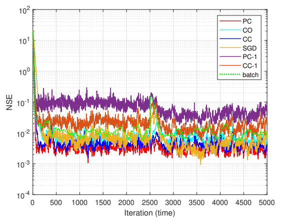
We use CVX [37] as solver for the offline computations, and report the required computational time in seconds achieved by Algorithm 3 and CVX.



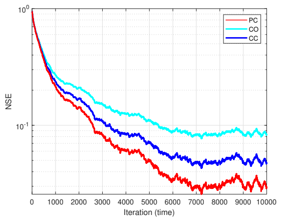
(a) **TV-GGM.** $N=18$, $\alpha=\beta=10^{-2}$, $\gamma=99.9\times10^{-2}$



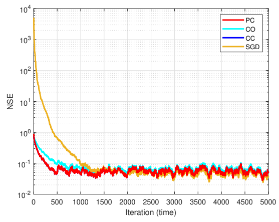
(b) **TV-SEM.** $N=28$, $\alpha=\beta=0.1\times10^{-2}$, $\lambda=0.5$, $\gamma=99\times10^{-2}$



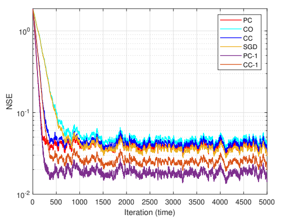
(c) **TV-SBM.** $N=28$, $\alpha=\beta=0.1\times10^{-2}$, $\lambda_1=10$, $\lambda_2=10$, $\gamma=99\times10^{-2}$



(d) **TV-GGM.** $N=18$, $\alpha=\beta=0.1\times10^{-2}$, $\gamma=99.9\times10^{-2}$



(e) **TV-SEM.** $N=28$, $\alpha=\beta=0.5\times10^{-2}$, $\lambda=5e-2$, $\gamma=99\times10^{-2}$



(f) **TV-SBM.** $N=28$, $\alpha=\beta=0.1\times10^{-2}$, $\lambda_1=1$, $\lambda_2=10$, $\gamma=99\times10^{-2}$

Figure 5.1: Normalized squared error (NSE) for the piecewise-constant (top row) and smooth (bottom row) synthetic scenarios between our online solution $\hat{\mathbf{s}}_t$ (or the other variants reported in the legend) with respect to the offline solution $\mathbf{s}_t^\star$ obtained with CVX. For the piecewise-constant scenario, it is also illustrated the NSE between the PC solution and the batch solution (green curve). Stochastic implementations are available for a subset of methods due to numerical instabilities caused by the rank-one matrix operations involved.

5.6.1. SYNTHETIC DATA

We generate a synthetic (seed) random graph $\mathbf{S}_0$ of N nodes using the GSP toolbox [38]. Then, edges abide two different temporal evolution patterns: *i) piecewise constant*; and *ii) smooth* temporal variation. Finally, we generate the stream of data according to the three considered models [cf. Section 5.4] for T time instants.

Piecewise. For the piecewise constant scenario, we randomly select $\lceil N/2 \rceil$ nodes of the initial graph $\mathbf{S}_0$ and double the weight of their edges, after $T/2$ samples. Then, for $t = \{1,\ldots,T\}$ we generate each graph signal $\mathbf{x}_t$ according to the three models: 1) for the TV-GGM, we use $\mathbf{x}_t \sim \mathcal{N}(\mathbf{0},\Sigma_t)$, where $\Sigma_t = \mathbf{S}_t^{-1}$; 2) for the TV-SEM we use $\mathbf{x}_t = (\mathbf{I}-\mathbf{S}_t)^{-1}\mathbf{e}_t$ [cf. (5.4)], with noise variance $\sigma_e^2 = 0.5$; and 3) for the TV-SBM we use $\mathbf{x}_t \sim \mathcal{N}(\mathbf{0},\mathbf{L}_t^{\dagger} + \sigma_e^2\mathbf{I}_N)$ as in [39] with $\sigma_e^2 = 0.5$.

Smooth. For the smooth scenario, starting from the initial graph $\mathbf{S}_0$, the evolution pattern follows an edge-dependent behavior, $S_t(i,j) = S_0(i,j)(1+e^{-0.01ijt})$ for $t = \{1,\ldots,T\}$. This means that each edge follows an exponential decaying behavior, with the decaying factor depending on the edge itself. The data are generated as in the piecewise constant scenario.

For the results, we will compare the following methods:

- **Prediction-correction (PC)** *red curve*: this is the proposed Algorithm 3 specialized to one of the three models, with $P = C = 1$.

- **Correction-only (CO)** *cyan curve*: this is a prediction-free algorithm which only considers the original problem (5.17) and applies $C = 1$ iteration of the recursion (5.18). It is equivalent to Algorithm 3 with $P = 0, C = 1$. We consider this algorithm to study the benefits of the prediction step performed by PC.

- **Correction-correction (CC)** *blue curve*: this is a prediction-free algorithm which only considers the original problem (5.17) and applies $C = 2$ iterations of the recursion (5.18). It is equivalent to Algorithm 3 with $P = 0, C = 2$. This is a more fair comparison than CO, since the number of iterations is the same as the one of PC.

- **Stochastic gradient descent (SGD)** *ochre curve*: this is a prediction-free and memory-less version of the algorithm which only considers the last acquired graph signal. That is, the empirical covariance matrix $\hat{\mathbf{\Sigma}}_t = \mathbf{x}_t\mathbf{x}_t^{\top}$ in (5.19) is just a rank-one update, achieved by setting $\gamma = 0$. We consider this to show how much the temporal variability of the function, captured by the time-derivative of the gradient in PC, affects the algorithm's convergence.

- **Prediction-correction rank-one (PC-1)** *purple curve*: this is a rank-one (stochastic) implementation of the PC algorithm; i.e., $\hat{\mathbf{\Sigma}}_t = \mathbf{x}_t\mathbf{x}_t^{\top}$ for the update in (5.19), and $P = C = 1$. Notice that, differently from SGD, it also uses the time-derivative of the gradient, which in this case is the difference between two rank-one covariance matrices (thus the length of the memory is equal to one). We consider this algorithm to check the impact of the prediction step in a stochastic implementation of PC;

- **Correction-correction rank-one (CC-1)** *orange curve*: this is a rank-one (stochastic) implementation of the CC algorithm; i.e., it considers $\hat{\mathbf{\Sigma}}_t = \mathbf{x}_t\mathbf{x}_t^\top$ for the update in (5.19), and $P = 0, C = 2$. It can be seen as a two-step SGD, and we consider it to study whether the prediction step of PC-1 is beneficial for stochastic implementations.

In addition, for the piecewise constant scenario, we also report (green curve) the NSE between the PC solution and the batch solution obtained having all the relevant data in advance, i.e., the solution that would be obtained with a static graph learning algorithm on the intervals where the graph remains constant. In general, a fair comparison can be made within the rank-one implementations (SGD, PC-1 and CC-1) and within the memory-aware ones (PC, CO, CC).

Results. The NSE achieved by Algorithm 3 for the three models is shown in Fig. 5.1, for both the piecewise constant (top row) and smooth (bottom row) scenarios. We use fixed step sizes for all the experiments. Notice that the only effect of the functions' hyperparameters is to shape the batch solution $\mathbf{s}_t^\star$ (and hence the time-varying trajectory $\hat{\mathbf{s}}_t$ at convergence). Thus, we run Algorithm 3 with different hyperparameters[4] and manually select them by ensuring that the trivial and complete graphs are excluded; the selected ones are displayed, together with the other algorithm's parameters, in the captions of Fig. 5.1.

GGM. Fig. 5.1a and Fig. 5.1d show the results for the piecewise constant and smooth scenarios, respectively. In both scenarios, the PC solution converges to the optimal offline counterpart and, for the piecewise constant, also to the batch solution(s). This demonstrates the adaptive nature of Algorithm 3 to react to changes in the data statistics. While for the piecewise constant scenario PC and CC offer the same convergence speed (which is expected, as explained in *"Does prediction help?"*), for the smooth scenario, the PC algorithm exhibits a faster convergence with respect to the prediction-free competitors CO and CC. This is because the temporal variability of the function (and of its gradient) is captured by the prediction step and exploited to fasten the convergence.

SEM. Similar considerations hold for the TV-SEM, whose results are illustrated in Fig. 5.1b and Fig. 5.1e. In both scenarios, PC and CC offer the same convergence rate (which also converge to the batch solution for the constant scenario), faster than a CO and SGD implementation. Interestingly, after the triggering event at $T/2$, SGD can track the optimal solution faster than CO with performances similar to PC and CC. A possible justification may be the memory-less nature of SGD, i.e., it only considers the last sample for the gradient evaluation, thus discarding past data. This renders the SGD more reactive to adapt to sudden changes of the data statistics compared to the memory-aware alternatives, which however exhibit similar performances thanks to the extra iteration they can benefit.

SBM. Finally, the TV-SBM results are shown in Fig. 5.1c and Fig. 5.1f. Also in this case, the PC solution converges to the offline counterpart for the two scenarios and faster

[4]The search space intervals for the hyperparameters are the following: $\alpha, \beta \in (0.01, 1) \times 10^{-2}$, $\lambda \in (0.005, 5)$, $\lambda_1, \lambda_2 \in (1, 10)$, $\gamma \in \{97, 99, 99.9\} \times 10^{-2}$.

than the prediction-free versions of the algorithm CC and CO. In particular, while in the piecewise constant scenario PC converges faster than CC and the rank-one implementations, in the smooth scenario the rank-one implementations exhibit faster convergent behavior with respect to the non-stochastic implementations. Similar to what has been said for the TV-SEM results, a possible reason can be the memory-aware characteristics of the non-stochastic methods; that is, while the information present in past data can be beneficial in the static scenario and thus help PC and CC to have a more reliable estimate of the true underlying (static) covariance matrix (and of the gradient), it may slow down the process in non-stationary environments with time-varying covariance matrices as in the smooth scenario.

Required time. An important metric to consider in time-sensitive applications is the average time per iteration. We report this information in Table 5.1, for the PC step and CVX, relative to the three considered models and settings in the top row of Fig. 5.1. Combining the information of the table and that of the plots in Fig. 5.1, it is clear how trading off the knowledge of the optimal solution for savings in terms of time seems an excellent compromise. Each prediction-correction step requires indeed around three orders of magnitude less time than the CVX counterpart, leading to a NSE at least smaller than $10e-1$.

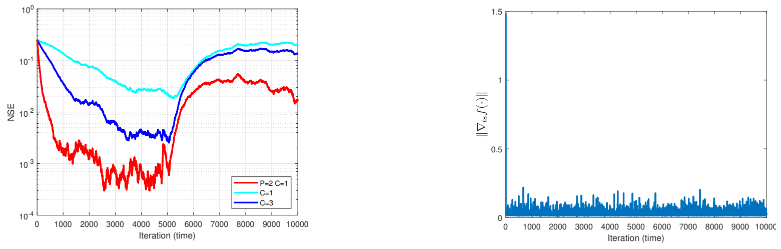
Does prediction help? Notice how in the piecewise constant scenario, the PC strategy does not seem to offer a major advantage with respect the CC strategy. Although this behavior could be hypothesized (since the setting is static), it is here empirically confirmed. To gain more insights we look at the structure of the prediction step (e.g., (5.23)), where the components playing a role in the descent direction are: the gradient $\nabla_{\mathbf{s}} f(\cdot)$; the Hessian $\nabla_{\mathbf{ss}} f(\cdot)$; and the time-derivative of the gradient $\nabla_{t\mathbf{s}} f(\cdot)$. Since we use $P = 1$, i.e., only one prediction step, the term $(\hat{\mathbf{s}}^p - \hat{\mathbf{s}}_t = \mathbf{0})$ that multiplies the Hessian does not contribute to the descent step. The added value of the prediction step with respect to a general (correction) descent method, in this case, would be only provided by the time-gradient $\nabla_{t\mathbf{s}} f(\cdot)$ (since the gradient $\nabla_{\mathbf{s}} f(\cdot)$ is common to either the prediction and the correction step). In the piecewise constant scenario, however, the underlying (true) covariance matrix is time-invariant within the two stationary intervals, leading to a zero time-derivative of the gradient (cf. (5.22)). This means that in static scenarios, with $P = 1$, the prediction step boils down to a correction step. Differently, for $P = 2$, the contribution of the second-order information may speed up the convergence, as illustrated in Fig. 5.2a for TV-GGM, with respect to a correction-only algorithm using $C = 3$.

In the smooth scenario, the temporal variability of the gradient captured by the time-derivative of the gradient $\nabla_{t\mathbf{s}} f(\cdot)$, plays a role in the prediction step, which can improve the convergence speed of the algorithm. The (bounded) norm of this vector over time is illustrated in Fig. 5.2b for the TV-GGM smooth scenario of Fig. 5.1d; this norm is linked to the constant C_0 introduced in Assumption 2 and the error in (5.40).

All in all, the results indicate the convergence of Algorithm 3 to the optimal offline counterpart and its capability to track it in non-stationary environments. The algorithm also converges to the batch solutions of the two stationary intervals,

Table 5.1: Average time (expressed in seconds) required to compute the PC and the CVX solution at each time instant.

	PC	CVX
TV-GGM	0.110×10^{-2}	3.6
TV-SEM	0.824×10^{-2}	2.0
TV-SBM	0.023×10^{-2}	3.6



(a) (b)

Figure 5.2: (a): NSE of PC with $P = 2$ and $C = 1$, CO with $C = 1$ and CC with $C = 3$ for the piecewise constant scenario; (b) Norm of the time-derivative of the gradient as a function of the iteration index for the smooth scenario.

obtained with all the relevant data. A defining characteristic of Algorithm 3 is its ability to naturally enforce similar solutions at each iteration, achieved with an early stopping of the descent steps, governed by the parameters P and C. That is, the algorithm adds an implicit *temporal* regularization to the problem which needs to be explicitly added when working with the entire batch of data.

Given these results and insights, we can outline a few principles that can be adopted when considering Algorithm 3 for learning problems:

- The prediction step with $P = 1$ can be beneficial when the underlying data statistics change over time, so that the time-variability of the gradient can be exploited. Otherwise, in a complete static scenario, it coincides with a correction step.

- Increasing P can improve the convergence speed when the approximated cost function is a good surrogate of the cost function in the next time instant.

- Memory-less (stochastic) variants of the algorithm can be suitable in fast-changing environments, due to their ability to discard past information and react quickly to changes in data statistics.

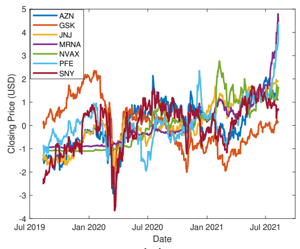
Being confident on the convergence of the algorithm, we now corroborate its performance with real data.

5.6.2. Real Data

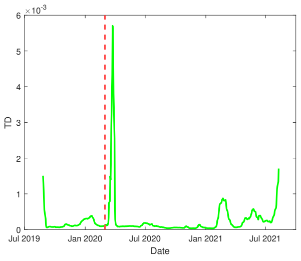
We now test the three considered algorithms on real data. Among other indicators employed in the simulations to assess the performance of the algorithm, we use the graph temporal deviation $\mathrm{TD}(t) := \|\hat{\mathbf{s}}_t - \hat{\mathbf{s}}_{t-1}\|_2$, which measures the global variability on the edges of the graph for different time instants. To gain further insights on the network evolution over time, we consider additional metrics (such as number of edges and temporal gradient norm) and visual analysis tools which will be introduced in the application-specific scenario at hand. In this case, the hyperparameters of each function are chosen in such a way that the inferred graphs are neither trivial nor complete, and interpretable patterns consistent with real events are visible from the plots of the employed metrics.

TV-GGM for Stock Price Data Analysis.

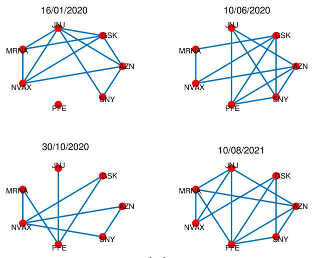
Data description: we collect historical stock (closing) prices relative to the S&P500 Index for seven pharmaceutical companies over the time period August 12th 2019 to August 10th 2021 using [40]. The collected data include the economic crisis related to the COVID-19 pandemic, followed by the vaccination campaign. The companies of interest are Pfizer (PFE), Astrazeneca (AZN), Johnson & Johnson (JNJ), GlaxoSmithKline (GSK), Moderna (MRNA), Novavax (NVAX) and Sanofi (SNY). Our goal is to leverage the TV-GGM in order to explore the relationships among these companies over time and observe the possible structural changes due to market instabilities.



(a)



(b)



(c)

Figure 5.3: (a) Standardized time series for the period August 12th 2019 - August 10th 2021; (b) graph temporal deviation for the stock market graph inferred with TV-GGM. The sharp peaks around March 2020 and after January 2021 happen consistently with real events; (c) inferred topologies at four different dates of interest. The absence of an edge between two nodes indicates their conditional independence.

Results: We consider $T = 504$ measurements (working days in August 2019 - August 2021) as graph signals $\{\mathbf{x}_t\}$ for the $N = 7$ quantities of interest, which are further standardized, i.e., each variable is centered and divided by its empirical standard deviation; see Fig. 5.3a for a plot of the standardized time series. We run the TV-GGM algorithm for different values of the forgetting factor γ, and monitor the evolution of the metrics earlier introduced. The value $\gamma = 0.75$ yielded results most consistent with the data behavior.

It is clear from Fig. 5.3a and the TD indicator in Fig. 5.3b that around March 2020 and after January 2021 the market has changed significantly, due to the instability generated by the pandemic and by the follow-up starting vaccination campaign. The sharp peaks in Fig. 5.3b around around the same period are a consequence of the dynamic inter-relationships among the companies; the inferred graph changes substantially in the two periods of interest and TD captures the market variability.

To really enjoy the visualization potential offered by graphs as a tool, we show in Fig. 5.3c snapshots of the inferred time-varying graph at four different dates of interest. Common among the four graphs is the presence of the edge connecting MRNA and NVAX, and the edge connecting AZN and SNY. The pharmaceutical companies associated to the endpoints of each of these two edges also show a similar trend in Fig. 5.3a. Notice moreover that since the sparsity pattern of the precision matrix reveals conditional independence among the variables indexed by its zero entries, these graphs enable us to visually inspect such independence over time. Although the information endowed in these graphs may carry a financial significance, we leave this possible knowledge-discovery task out of this manuscript, to avoid misleading or erroneous interpretations.

TV-SEM for Temperature Monitoring.

Data description: for this experiment we consider the publicly available weather dataset[5] provided by the Irish Meteorological Service, which contains hourly temperature (in °C) data from 25 stations across Ireland. We monitor the temperature evolution over the sensor network for the period January 2016 to May 2020, and leverage the TV-SEM to infer the time-varying features of the graph learned by the algorithm.

Results: for the analysis we consider $T = 38713$ measurements as graph signals $\{\mathbf{x}_t\}$ for the $N = 25$ stations under consideration, standardizing the data as done in the previous experiment; Fig. 5.4a depicts the standardized time series. It is interesting to notice the sinusoidal-like behavior of the aggregate time-series, due to higher (lower) temperature during the summer (winter) period, resulting in a smooth signal profile.

Fig. 5.4b illustrates the sparsity pattern of the time-varying graph and the importance of the weights at every time instant. This *learn-and-show* feature offered by Algorithm 3 gives us the ability to visualize the learning behavior of the algorithm on-the-fly, a strength of low-cost iterative algorithms w.r.t. batch counterparts. From the figure (and the observed almost zero graph temporal deviation, which is not illustrated here) a consistent temporal homogeneity is visible, i.e., the graph does not change significantly over adjacent time instants. In other words, nodes influencing each other in a particular time instant, are likely to influence each other in other time instants. A reasonable explanation is given by the smooth and regular pattern exhibited by the time-series of Fig. 5.4a, which is a consequence of the meteorological similarity over time, and by their high correlation coefficient.

An interesting trend arises when observing the number of edges of the graph inferred over time, shown in Fig. 5.5a. Although in adjacent time instants the number does not change abruptly, a pattern can be identified over a longer time span. In

[5]Data available at https://www.met.ie/climate/available-data/historical-data

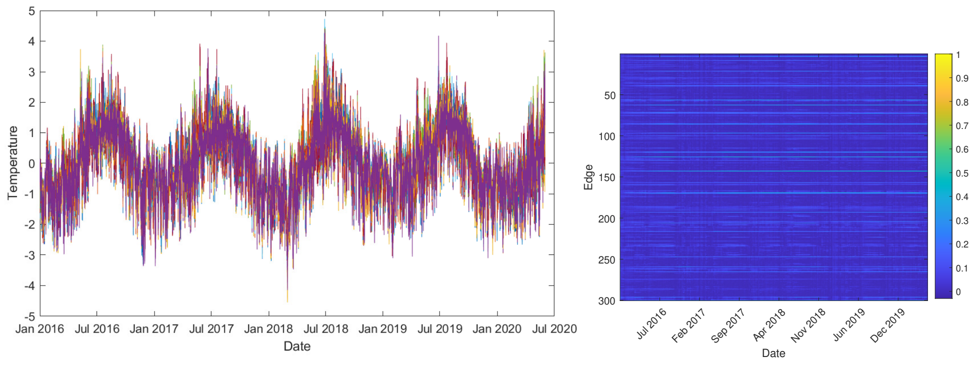

(a) (b)

Figure 5.4: (a) Standardized time series for the 25 Irish weather stations and (b) evolution of each edge weight over time.

particular, during winter and summer there is a sharp increment in the number of edges, with respect to autumn and spring where there is a significant reduction. To ease the visualization, the vertical red lines are placed in correspondence of the winter period of every year, while blue lines in correspondence of the autumn period. A possible reason for this phenomenon is given by the reduced variability of the temperature among the stations during summer and winter, and a higher variability during spring and autumn, leading to different graphs.

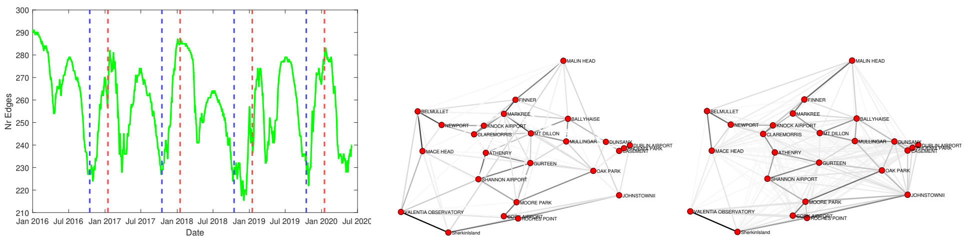

(a) (b) Oct 2016 (c) Jan 2017

Figure 5.5: Ireland temperature dataset. (a) Number of edges of the inferred graph over time. The red vertical lines correspond to January 15 of each year (winter), while the blue vertical line correspond to October 15 (autumn); snapshot of the inferred time-varying graph during (b) October 2016 (autumn graph) and (c) January 2017 (winter graph). Notice how stations close in space tend to be connected.

For the sake of visualization, we also report the inferred graphs for October 2016 (autumn) and January 2017 (winter). In line with our previous comments regarding Fig. 5.5a, a lower number of edges is visible in the autumn graph with respect to the winter graph; in particular, edges present in the autumn graph are also present in the winter one. Finally, notice how stations close in space tend to be connected, thus showing how stations close to each other have a greater influence with respect to stations farther away in space.

TV-SBM for Epileptic Seizure Analysis.

Data description: we use electrocorticography (ECoG) time series collected during an epilepsy study at the University of California, San Francisco (UCSF) Epilepsy Center, where an 8×8 grid of electrodes was implanted on the cortical brain's surface of a 39-year-old woman with medically refractory complex partial seizure [41]. The grid was supplemented by two strips of six electrodes: one deeper implanted over the left suborbital frontal lobe and the other over the left hippocampal region, thus forming a network of 76 electrodes, all measuring the voltage level in proximity of the electrode, which is an evidence of the local brain activity. The sampling rate is 400 Hz and the measured time series contains the 10 seconds interval preceding the seizure (pre-ictal interval) and the 10 seconds interval after the start of the seizure (ictal interval). Our goal is to leverage the TV-SBM in order to explore the dynamics among different brain areas at the seizure onset.

Results: for our analysis we consider $T = 3200$ time instants as graphs signals $\{\mathbf{x}_t\}$ for the $N = 76$ electrodes, which are further filtered (over the temporal dimension) at $\{60, 180\}$Hz to remove the spurious power line frequencies, and standardized as explained in the previous experiments.

Fig. 5.6 shows the graph temporal deviation, where we observe an increasing and protracted variability of the TD shortly after the seizure onset (red vertical line), proving TD to serve as an indicator of network alteration suitable for time-varying scenarios. To visualize the on-the-fly learning behavior of the algorithm, in Fig. 5.7a we show the evolution of (a fraction[6] of) the edge weights over time. In the first half of the time-horizon, we notice the presence of stronger edges with respect to the second half, where the graph is sparser. We show two snapshots of the time-varying graph in Fig. 5.7b, for the time instants 1500 (pre-ictal) and 1800 (ictal), where we also report the closeness centrality of each node, which expresses how "close" a node is to all other nodes in the network (calculated as the average of the shortest path length from the node to every other node in the network). During the ictal interval, the graph tends to be more disconnected and its nodes to have a lower closeness centrality value, especially in the lower part of the graph. In addition, we observe how the number of (strong) edges and the closeness centrality value drop in the ictal graph, especially in the lower part of the graph. This is consistent with the findings in [41] and indicates that, on average, signals in the pre-ictal interval behave more similar to each other as opposed to the signals in the ictal interval.

5.7. Conclusion

In this manuscript, we proposed an algorithmic template to learn time-varying graphs from streaming data. The abstract time-varying graph learning problem, where the data influence is expressed through the empirical covariance matrix, is

[6]For visualization, we show 500 random edges, since we recall that the number of total edges in an undirected graph of N nodes is $N(N-1)/2$.

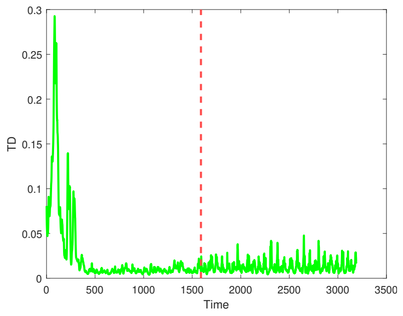


Figure 5.6: Graph temporal deviation for the epilepsy study. The red line indicates the seizure onset. During the ictal interval, a higher temporal deviation can be observed, indicating that the inferred graph is changing substantially.

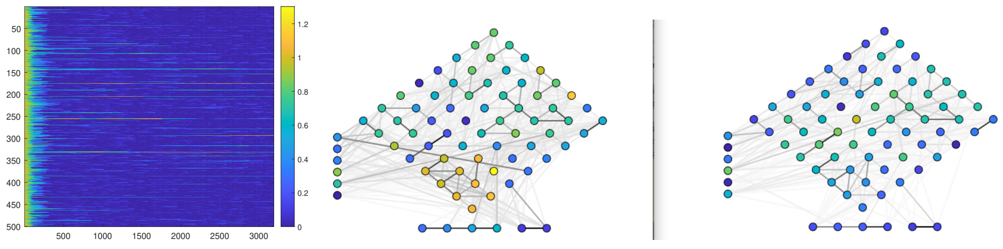


Figure 5.7: Epilepsy dataset. (a) evolution of each edge weight over time; (b) snapshots of the inferred time-varying graph at time instant 1500 and 1800. The color of an edge indicates its weight, with darker colors indicating higher weights, while the color of a node indicates the closeness centrality of such node, with brighter colors indicating higher values of closeness centrality.

casted as a composite optimization problem, with different terms regulating different desiderata. The framework, which works in non-stationary environments, lies upon novel iterative time-varying optimization algorithms, which on one side exhibit an implicit temporal regularization of the solution(s), and on the other side accelerate the convergence speed by taking into account the time variability. We specialize the framework to the Gaussian graphical model, the structural equation model, and the smoothness-based model, and we propose ad-hoc vectorization schemes for structured matrices central for the gradient computations which also ease storage requirements. The proposed approach is accompanied by theoretical performance guarantees to track the optimal time-varying solution, and is further validated with synthetic numerical results. Finally, we learn time-varying graphs in the context of stock market, temperature monitoring, and epileptic seizures analysis. The current line of work can be enriched by specializing the framework to other static graph learning methods present in literature, possibly considering directed graphs, by implementing distributed versions of the optimization algorithms, and by applying the developed models in other real-world applications.

Appendix

Consider the multi-valued function $\mathscr{T}:\mathbb{R}^N \to \mathbb{R}^N$, which we will refer to as *operator*. Here, we briefly review some operator theory concepts used in this manuscript; see [42].

Projection operator. Given a point $\mathbf{x} \in \mathbb{R}^N$, we define projection of $\mathbf{x}$ onto the convex set $\mathscr{C} \subseteq \mathbb{R}^N$ as:

$$\mathbb{P}_{\mathscr{C}}(\mathbf{x}) := \operatorname*{argmin}_{\mathbf{z}\in\mathscr{C}} \frac{1}{2}\|\mathbf{z}-\mathbf{x}\|_2 \tag{5.44}$$

Proximal operator. Consider the convex function $g:\mathbb{R}^N \to \mathbb{R}$. We define the proximal operator of $g(\cdot)$, with penalty parameter $\rho > 0$, as:

$$\operatorname{prox}_{g,\rho}(\mathbf{x}) := \operatorname*{argmin}_{\mathbf{z}}\{g(\mathbf{z}) + \frac{1}{2\rho}\|\mathbf{z}-\mathbf{x}\|_2^2\} \tag{5.45}$$

For some functions, the proximal operator admits a closed form solution [36, Ch. 6]. In particular:

- if $g(\mathbf{x}) = \iota_{\mathscr{C}}(\mathbf{x})$ then $\operatorname{prox}_g(\mathbf{x}) = \mathbb{P}_{\mathscr{C}}(\mathbf{x})$, i.e., it is the projection of $\mathbf{x}$ onto the convex set $\mathscr{C}$.
- if $g(\mathbf{x}) = \lambda\|\mathbf{x}\|_1$ then $\operatorname{prox}_g(\mathbf{x}) = \operatorname{sign}(\mathbf{x}) \odot [\mathbf{x} - \lambda\mathbf{1}]_+$, i.e., it is the soft-thresholding operator.

Consider the convex minimization problem:

$$\min_{\mathbf{x}} f(\mathbf{x}) + g(\mathbf{x}) \tag{5.46}$$

with $f, g:\mathbb{R}^N \to \mathbb{R}$ convex. It can be shown that problem (5.46) admits at least one solution [43], which can be found by the fixed point equation:

$$\mathbf{x} = \operatorname{prox}_{g,\rho}(\mathbf{x} - \rho\nabla f(\mathbf{x})) \tag{5.47}$$

5.8.

Proof of Claim 1: TV-GGM. Recall the expression of the Hessian in (5.21b), i.e., $\mathbf{H}(\mathbf{S}) = \mathbf{D}^\top(\mathbf{S}\otimes\mathbf{S})^{-1}\mathbf{D}$ and that matrix $\mathbf{S} \in \mathscr{S}$ is the precision matrix, with $\mathscr{S} = \{\mathbf{S} \in \mathbb{S}^N_{++} | \xi\mathbf{I} \preceq \mathbf{S} \preceq \chi\mathbf{I}\}$.

For the strong convexity, notice that since $\mathbf{S} \succ 0$, then also $\mathbf{H}(\mathbf{S}) \succ 0$. Indeed, by exploiting the semi-orthogonality of matrix $\mathbf{D}/\sqrt{2}$, we have:

$$\lambda_{\min}(\mathbf{H}(\mathbf{S})) = \min_{\|\mathbf{x}\|=1} \mathbf{x}^\top\mathbf{D}^\top(\mathbf{S}\otimes\mathbf{S})^{-1}\mathbf{D}\mathbf{x} \tag{5.48}$$

$$\geq \min_{\|\mathbf{x}\|=1} \mathbf{x}^\top\frac{\mathbf{D}^\top}{\sqrt{2}}(\mathbf{S}\otimes\mathbf{S})^{-1}\frac{\mathbf{D}}{\sqrt{2}}\mathbf{x} = \min_{\|\mathbf{y}\|=1} \mathbf{y}^\top(\mathbf{S}\otimes\mathbf{S})^{-1}\mathbf{y}$$

$$= \min_{\|\mathbf{z}\|=1} \sum_{i=1}^N\sum_{j=1}^N \frac{z_i z_j}{\lambda_i(\mathbf{S})\lambda_j(\mathbf{S})} \geq \frac{1}{\lambda^2_{\max}(\mathbf{S})} = 1/\chi^2$$

For the Lipschitz continuity of the gradient, we have

$$\begin{aligned}\|\mathbf{D}^\top(\mathbf{S}\otimes\mathbf{S})^{-1}\mathbf{D}\| &\le \|\mathbf{D}\|^2\|(\mathbf{S}\otimes\mathbf{S})^{-1}\| \\ &= 2\|(\mathbf{S}\otimes\mathbf{S})^{-1}\| = 2\|\mathbf{S}^{-1}\otimes\mathbf{S}^{-1}\| \\ &= 2\sqrt{\lambda_{\min}(\mathbf{S})^{-2}} = 2/\xi\end{aligned} \tag{5.49}$$

■

Proof of Claim 2: TV-SEM. Denote with $\lambda_{\min}$ and $\lambda_{\max}$ the smallest and highest eigenvalues for the set of empirical covariance matrices obeying the SEM model. Recall the expression of the Hessian in (5.27b), i.e. $\mathbf{H}(\mathbf{S};t)=\mathbf{Q}_t$, where $\mathbf{Q}_t := \mathbf{D}_h^\top(\hat{\boldsymbol{\Sigma}}_t\otimes\mathbf{I})\mathbf{D}_h$. Since $\mathbf{D}_h/\sqrt{2}$ is a semi-orthogonal matrix, we have:

$$\begin{aligned}\lambda_{\min}(\mathbf{H}(\mathbf{S})) &= \min_{\|\mathbf{x}\|=1}\mathbf{x}^\top\mathbf{D}_h^\top(\hat{\boldsymbol{\Sigma}}_t\otimes\mathbf{I})\mathbf{D}_h\mathbf{x} \\ &\ge \min_{\|\mathbf{y}\|=1}\mathbf{y}^\top(\hat{\boldsymbol{\Sigma}}_t\otimes\mathbf{I})\mathbf{y} = \min_{\|\mathbf{z}\|=1}\sum_{i=1}^{N}\lambda_i(\hat{\boldsymbol{\Sigma}}_t)z_i^2 \ge \lambda_{\min}\end{aligned} \tag{5.50}$$

where $\lambda_{\min}$ is the smallest eigenvalue of $\hat{\boldsymbol{\Sigma}}_t$.

For the Lipschitz continuity of the gradient, we have:

$$\|\mathbf{D}_h^\top(\hat{\boldsymbol{\Sigma}}_t\otimes\mathbf{I})\mathbf{D}_h\| \le 2\|\hat{\boldsymbol{\Sigma}}_t\otimes\mathbf{I}\| = 2\lambda_{\max} \tag{5.51}$$

■

Proof of Claim 3: TV-SBM. For the strong convexity it suffices to notice that for $m>0$, $f(\mathbf{s};t)-\frac{m}{2}\|s\|^2 = \mathbf{s}^\top\mathbf{z}_t - \lambda_2\mathbf{1}^\top\log(\mathbf{K}\mathbf{s}) + (\lambda_1-\frac{m}{2})\|\mathbf{s}\|^2$ is convex. In turn, this implies that strong convexity of $f(\cdot;t)$ is guaranteed for $0<m\le 2\lambda_1$.

For the Lipschitz continuity of the gradient, recall that nodal degree vector $\mathbf{d}>0$. Denote with $d_{\min}$ the minimum degree of the GSO search space. Also, recall the expression of the Hessian $\mathbf{H}=\mathbf{K}^\top\operatorname{Diag}(\mathbf{1}\oslash(\mathbf{K}\mathbf{s})^{\circ 2})\mathbf{K}$. Then:

$$\begin{aligned}\|\mathbf{K}^\top\operatorname{Diag}(\mathbf{1}\oslash(\mathbf{K}\mathbf{s})^{\circ 2})\mathbf{K}\| &\le \|\mathbf{K}\|^2\max(\mathbf{1}\oslash(\mathbf{K}\mathbf{s})^{\circ 2})) \\ &= \|\mathbf{K}\|^2 d_{\min}^{-2} = 2(N-1)d_{\min}^{-2},\end{aligned} \tag{5.52}$$

where we made use of [20, Lemma 1] for the bound of $\mathbf{K}$. ■

5.9.

The computational (arithmetic) complexity per iteration of Algorithm 3 is dominated by the rank-one covariance matrix update in $\mathcal{O}(N^2)$ and by the method-specific gradient computations involved in the prediction and correction steps (and eventually Hessian, if $P>1$ [cf. Section 5.6 "*Does prediction help?*"]). Such method-specific computational complexities are shown next, together with a discussion on the costs for the offline counterparts.

TV-GGM. The worst case scenario computational complexity of the gradient $\nabla_{\mathbf{s}} f(\mathbf{s};t)$ in (5.21a) is $\mathcal{O}(N^3)$, which is due to the matrix inversion. This cost might be lowered exploiting the sparsity pattern of the sparse triangular factor of $\mathbf{S}$ or, in our case, exploiting the fact that it is a small perturbation with respect to the previous iterate. The multiplication with matrix $\mathbf{D}^\top$ has a cost of $\mathcal{O}(N^2)$, since $\mathbf{D} \in \mathbb{R}^{N^2 \times N(N+1)/2}$ has at most two 1's in each column and exactly one 1 in each row.

The worst case scenario computational complexity of the Hessian $\nabla_{\mathbf{ss}} f(\mathbf{s};t)$ in (5.21b) would be $\mathcal{O}(N^3)$. However, because the Hessian is used in a matrix-vector multiplication [cf. (5.23)], its factorization leads to a cost for the prediction step of $\mathcal{O}(N^3)$. Indeed, exploiting the Kronecker product, the Hessian can be written as $\mathbf{D}^\top(\mathbf{S}^{-1} \otimes \mathbf{I}_N)(\mathbf{I}_N \otimes \mathbf{S}^{-1})\mathbf{D}$; then, the multiplication of the Hessian for a vector simply entails the succession of four sparse matrix-vector multiplications all with a cost of $\mathcal{O}(N^3)$.

The term $\nabla_{t\mathbf{s}} f(\mathbf{s};t)$ in (5.22) has a computational complexity of $\mathcal{O}(N^2)$. Thus the overall computational complexity per iteration is $\mathcal{O}(N^3)$.

TV-SEM. The overall cost is dominated by the computation of $\mathbf{Q}_t = \mathbf{D}_h^\top(\hat{\mathbf{\Sigma}}_t \otimes \mathbf{I})\mathbf{D}_h$, which is present in the gradients and the Hessian. The matrix-matrix multiplication(s) have a cost of $\mathcal{O}(N^3)$, since $\mathbf{D}_h \in \mathbb{R}^{N^2 \times N(N-1)/2}$ has at most two 1's in each column and exactly one 1 in each row. Thus the overall computational complexity per iteration is $\mathcal{O}(N^3)$.

TV-SBM. Each column of $\mathbf{K}$ has exactly two non-zero entries (and each row has $N-1$ non-zero entries), thus $\mathbf{Ks}$ has a computational cost of $2|\mathcal{E}|$, with $|\mathcal{E}|$ the number of edges of the graph represented by $\mathbf{S}$ (in other words, $\|\mathbf{s}\|_0$). The operation $\mathbf{K}^\top(\mathbf{1} \oslash \mathbf{Ks})$ has a cost of $\mathcal{O}(N^2)$. The computational complexity of the Hessian is $\mathcal{O}(N^3)$, since it is the weighted sum of N outer products of vectors which are $(N-1)$-sparse in the same positions.

Thus the overall computational complexity per iteration is $\mathcal{O}(N^2)$ if $P = 0, 1$ and $\mathcal{O}(N^3)$ if $P > 1$.

Offline. The computational complexity for each time instant t incurred by an offline solver to solve instances of problem (5.13) depends by its algorithm-specific implementation closely related to the problem structure. The three problems we consider are (converted into) semidefinite programs (SDPs) and solved, in our case, by SDPT3, a Matlab implementation of infeasible primal-dual path-following algorithms, which involves the computation of second-order information. Since these computations are continuously repeated, for a fixed time instant t, till the algorithm convergence (say I iterations), a trivial lower bound for computing the offline solution for the three considered problems is $\mathbf{\Omega}(IN^3)$. To this cost must be also added the cost of other solver-specific steps which we do not explicitly consider here.

REFERENCES

[1] A. Natali, M. Coutino, E. Isufi, and G. Leus. “Online time-varying topology identification via prediction-correction algorithms”. In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2021, pp. 5400–5404.

[2] A. Natali, E. Isufi, M. Coutino, and G. Leus. “Learning time-varying graphs from online data”. In: *IEEE Open Journal of Signal Processing* 3 (2022), pp. 212–228.

[3] M. Coutino, E. Isufi, and G. Leus. “Advances in distributed graph filtering”. In: *IEEE Transactions on Signal Processing* 67.9 (2019), pp. 2320–2333.

[4] G. Mateos, S. Segarra, A. G. Marques, and A. Ribeiro. “Connecting the dots: Identifying network structure via graph signal processing”. In: *IEEE Signal Processing Magazine* 36.3 (2019), pp. 16–43.

[5] X. Dong, D. Thanou, M. Rabbat, and P. Frossard. “Learning graphs from data: A signal representation perspective”. In: *IEEE Signal Processing Magazine* 36.3 (2019), pp. 44–63.

[6] Y. Kim, S. Han, S. Choi, and D. Hwang. “Inference of dynamic networks using time-course data”. In: *Briefings in bioinformatics* 15.2 (2014), pp. 212–228.

[7] R. N. Mantegna. “Hierarchical structure in financial markets”. In: *The European Physical Journal B-Condensed Matter and Complex Systems* 11.1 (1999), pp. 193–197.

[8] A. Sandryhaila and J. M. Moura. “Discrete signal processing on graphs”. In: *IEEE transactions on signal processing* 61.7 (2013), pp. 1644–1656.

[9] V. Kalofolias, A. Loukas, D. Thanou, and P. Frossard. “Learning time varying graphs”. In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2017, pp. 2826–2830.

[10] K. Yamada, Y. Tanaka, and A. Ortega. “Time-varying graph learning with constraints on graph temporal variation”. In: *arXiv preprint arXiv:2001.03346* (2020).

[11] D. Hallac, Y. Park, S. Boyd, and J. Leskovec. “Network inference via the time-varying graphical lasso”. In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2017, pp. 205–213.

[12] J. Friedman, T. Hastie, and R. Tibshirani. “Sparse inverse covariance estimation with the graphical lasso”. In: *Biostatistics* 9.3 (2008), pp. 432–441.

[13] B. Baingana and G. B. Giannakis. “Tracking Switched Dynamic Network Topologies From Information Cascades”. In: *IEEE Transactions on Signal Processing* 65.4 (2017), pp. 985–997.

[14] R. Money, J. Krishnan, and B. Beferull-Lozano. “Online Non-linear Topology Identification from Graph-connected Time Series”. In: *arXiv preprint arXiv:2104.00030* (2021).

[15] G. B. Giannakis, Y. Shen, and G. V. Karanikolas. “Topology identification and learning over graphs: Accounting for nonlinearities and dynamics”. In: *Proceedings of the IEEE* 106.5 (2018), pp. 787–807.

[16] S. Vlaski, H. P. Maretić, R. Nassif, P. Frossard, and A. H. Sayed. “ONLINE GRAPH LEARNING FROM SEQUENTIAL DATA”. In: *2018 IEEE Data Science Workshop (DSW)*. 2018, pp. 190–194.

[17] R. Shafipour and G. Mateos. “Online Topology Inference from Streaming Stationary Graph Signals with Partial Connectivity Information”. In: *Algorithms* 13.9 (2020). ISSN: 1999-4893. URL: https://www.mdpi.com/1999-4893/13/9/228.

[18] A. G. Marques, S. Segarra, G. Leus, and A. Ribeiro. “Stationary graph processes and spectral estimation”. In: *IEEE Transactions on Signal Processing* 65.22 (2017), pp. 5911–5926.

[19] B. Zaman, L. M. L. Ramos, D. Romero, and B. Beferull-Lozano. “Online Topology Identification From Vector Autoregressive Time Series”. In: *IEEE Transactions on Signal Processing* 69 (2021), pp. 210–225. DOI: 10.1109/TSP.2020.3042940.

[20] S. S. Saboksayr, G. Mateos, and M. Cetin. “Online discriminative graph learning from multi-class smooth signals”. In: *Signal Processing* 186 (2021), p. 108101.

[21] A. Simonetto, E. Dall’Anese, S. Paternain, G. Leus, and G. B. Giannakis. “Time-Varying Convex Optimization: Time-Structured Algorithms and Applications”. In: *Proceedings of the IEEE* (2020).

[22] 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”. In: *IEEE signal processing magazine* 30.3 (2013), pp. 83–98.

[23] A. P. Dempster. “Covariance selection”. In: *Biometrics* (1972), pp. 157–175.

[24] J. B. Ullman and P. M. Bentler. “Structural equation modeling”. In: *Handbook of psychology* (2003), pp. 607–634.

[25] V. Kalofolias. “How to learn a graph from smooth signals”. In: *Artificial Intelligence and Statistics*. PMLR. 2016, pp. 920–929.

[26] S. Kumar, J. Ying, J. V. de Miranda Cardoso, and D. P. Palomar. “A Unified Framework for Structured Graph Learning via Spectral Constraints.” In: *Journal of Machine Learning Research* 21.22 (2020), pp. 1–60.

[27] A. Simonetto, A. Mokhtari, A. Koppel, G. Leus, and A. Ribeiro. “A class of prediction-correction methods for time-varying convex optimization”. In: *IEEE Transactions on Signal Processing* 64.17 (2016), pp. 4576–4591.

[28] B. Martinet. “Régularisation d’inéquations variationnelles par approximations successives. Rev. Française Informat”. In: *Recherche Opérationnelle* 4 (1970), pp. 154–158.

[29] J.-P. Vial. “Strong and weak convexity of sets and functions”. In: *Mathematics of Operations Research* 8.2 (1983), pp. 231–259.

[30] J. R. Magnus and H. Neudecker. *Matrix differential calculus with applications in statistics and econometrics*. John Wiley & Sons, 2019.

[31] N. Bastianello, A. Simonetto, and R. Carli. *Primal and Dual Prediction-Correction Methods for Time-Varying Convex Optimization*. 2020. arXiv: 2004.11709 [math.OC].

[32] E. K. Ryu and S. Boyd. “Primer on monotone operator methods”. In: *Appl. Comput. Math* 15.1 (2016), pp. 3–43.

[33] P. L. Combettes and J.-C. Pesquet. “Proximal splitting methods in signal processing”. In: *Fixed-point algorithms for inverse problems in science and engineering*. Springer, 2011, pp. 185–212.

[34] X. Zhan. “Extremal eigenvalues of real symmetric matrices with entries in an interval”. In: *SIAM journal on matrix analysis and applications* 27.3 (2005), pp. 851–860.

[35] K. C. Das and R. Bapat. “A sharp upper bound on the spectral radius of weighted graphs”. In: *Discrete mathematics* 308.15 (2008), pp. 3180–3186.

[36] A. Beck. *First-order methods in optimization*. SIAM, 2017.

[37] M. Grant, S. Boyd, and Y. Ye. *CVX: Matlab software for disciplined convex programming*. 2008.

[38] N. Perraudin, J. Paratte, D. Shuman, L. Martin, V. Kalofolias, P. Vandergheynst, and D. K. Hammond. “GSPBOX: A toolbox for signal processing on graphs”. In: *ArXiv e-prints* (Aug. 2014). arXiv: 1408.5781 [cs.IT].

[39] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst. “Learning Laplacian matrix in smooth graph signal representations”. In: *IEEE Transactions on Signal Processing* 64.23 (2016), pp. 6160–6173.

[40] *Yahoo! Finance*. https://finance.yahoo.com/lookup?s=API. [Online; accessed July-2021].

[41] M. A. Kramer, E. D. Kolaczyk, and H. E. Kirsch. “Emergent network topology at seizure onset in humans”. In: *Epilepsy research* 79.2-3 (2008), pp. 173–186.

[42] H. H. Bauschke, P. L. Combettes, *et al. Convex analysis and monotone operator theory in Hilbert spaces*. Vol. 408. Springer, 2011.

[43] P. L. Combettes and V. R. Wajs. “Signal recovery by proximal forward-backward splitting”. In: *Multiscale Modeling & Simulation* 4.4 (2005), pp. 1168–1200.

6
A Generalization of the Convolution Theorem and its Connections to Non-Stationarity and the Graph Frequency Domain

In this paper, we present a novel convolution theorem which encompasses the well known convolution theorem in (graph) signal processing as well as the one related to time-varying filters. Specifically, we show how a node-wise convolution for signals supported on a graph can be expressed as another node-wise convolution in a frequency domain graph, different from the original graph. This is achieved through a parameterization of the filter coefficients following a basis expansion model. After showing how the presented theorem is consistent with the already existing body of literature, we discuss its implications in terms of non-stationarity. Finally, we propose a data-driven algorithm based on subspace fitting to learn the frequency domain graph, which is then corroborated by experimental results on synthetic and real data.

6.1. Introduction

Convolution is the core operation in signal processing and machine learning systems. Its use is at the heart of digital filters [4] for audio applications and time series prediction [5], as well as for convolutional neural networks in deep learning [6], enabling scalable architectures and endowing a notion of locality among samples, properties exploited in, e.g., object recognition[7].

Parts of this chapter have been published in the Asilomar Conference on Signals, Systems, and Computers (2022) [1], in the IEEE International Conference on Acoustics, Speech and Signal Processing (2023) [2] in the IEEE Transactions on Signal Processing (2024) [3].

The three key operations defining a convolution are the *shift*, the *scale* and the *sum*. The shift is responsible to capture the underlying signal domain and brings the notion of locality and proximity among samples: in time, for instance, successive applications of the shift (in that case corresponding to a delay) give previous time samples. The scale defines how the shifted samples are weighted before summing them, and different weighting schemes lead to different structural properties (such as invariants) of the architecture implementing the convolution. The notion of regularity in time and in space, which are two very well structured domains, is reflected in the definition of their frequency domain. Specifically, a signal in these domains can be decomposed into elementary building blocks (such as sine waves) which endow a physical interpretation with a well understood meaning of variability. In a less structured domain modeled by a graph, this definition is not tight and multiple interpretations are possible.

Graph signal processing (GSP) [8] extends the convolution principle to data residing on graphs by means of graph filters (GFs) [9, 10], architectures which are parametric on the mathematical structure defining the *shift* operation. While in temporal signal processing this shift operator is mathematically represented by the (lower) shift/delay matrix, in GSP the *graph shift operator* (GSO) depends on the underlying network domain. The eigendecomposition of the GSO reveals its respective frequencies: specifically, the eigenvalues of the shift are the frequencies. In temporal signal processing these frequencies are the well-known complex roots of unity which obey a natural ordering, and the associated (normalized) eigenvectors are the Fourier modes (remember that the delay matrix is a particular case of a circulant matrix). In GSP, however, different shifts have different eigenvalues and hence different frequencies. In this case, an ordering purely based on their numeric value might not be meaningful and a more structured domain, for instance captured by a graph, might convey more information. This is the recent line of work explored in [11, 12], which we exploit here.

By relying on the novel notion of *dual graph* [11], which models the support of the frequency domain as a graph, in this work we introduce a new convolution theorem which generalizes the (graph) convolution theorem and the one related to time-varying filters. To do this, we adopt so-called node-varying graph filters (NV-GFs) [9] and we show how a node-varying convolution in one domain (captured by the *primal* graph) can be expressed as a node-variant convolution in the other domain (captured by the *dual* graph), while remaining consistent with the pre-existing body of literature. Based on the proposed theorem, we outline models for non-stationary graph signals. Finally, we propose an algorithmic approach to learn the dual graph from data with a subspace fitting approach [13] resorting to sequential convex programming [14] to tackle the non-convexity of the problem. The validity of this approach is finally corroborated by simulations on synthetic and real data.

6.1.1. RELATED WORKS

Although this work is novel in its genre, many of the concepts it relies on have been recently introduced. Modeling the frequency domain through a graph has been proposed in [11, 12], and similar efforts to interpret such a domain differently from ordering the eigenvalues of the GSO is given by means of embeddings in [15, 16]. Node-variant graph filters as a way to extend classical time-varying filters to the graph setting have been introduced in [9], while graph signal stationarity arguments and their implications have been studied in [17, 18]. Non-stationarity of random graph signals has been exploited in [19] in the context of network topology identification.

6.1.2. CONTRIBUTIONS

We summarize the specific contributions of this work as follows:

1. We propose a convolution theorem which encompasses the (graph) convolution theorem and the one related to time-varying filters; we show how the latter are specific instantiations of the proposed theorem for particular choices of the GSO and the scaling scheme;

2. We introduce novel non-stationary graph signal models;

3. We devise an algorithmic procedure to learn the dual graph from input-output data. The problem formulation can be casted as a blind polynomial regression, as such also applicable to graph-agnostic tasks, such as polynomial interpolations and jitter correction in communication applications. The solution approach relies on a subspace fitting method and it is accompanied by a theoretical study of the ambiguities present in the problem.

4. We showcase the validity of our findings on synthetic and real data with numerical simulations.

6.2. PRELIMINARIES

Graphs and Graph Signals. We consider data residing on a non-Euclidean domain, which we formally model by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{S})$, where $\mathcal{V} = \{1, \ldots, N\}$ is the set of nodes (or vertices), $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges, and $\mathbf{S}$ is an $N \times N$ matrix that represents the graph structure. The matrix $\mathbf{S}$ is called the graph shift operator (GSO), since it plays a role akin to the delay operator in temporal signal processing. Specifically, its entries $[\mathbf{S}]_{ij} \in \mathbb{C}$ for $i \neq j$ are different from zero only if nodes i and j are connected by an edge; typical examples of such a matrix are the (weighted) adjacency matrix $\mathbf{W}$ [20] and the graph Laplacian $\mathbf{L}$ [8]. A graph signal is then the vector $\mathbf{x} \in \mathbb{C}^N$, where $x_i : \mathcal{V} \to \mathbb{C}$ is the value collected at node i.

In this manuscript, for the sake of simplicity, we consider the shift operator $\mathbf{S}$ to admit an eigenvalue decomposition (EVD) written as $\mathbf{S} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^{-1}$, with $\mathbf{V}$ an invertible matrix collecting the eigenvectors and $\boldsymbol{\Lambda} = \mathrm{Diag}(\boldsymbol{\lambda})$ a diagonal matrix collecting the eigenvalues $\boldsymbol{\lambda}$ of $\mathbf{S}$. A fundamental assumption in GSP is that the matrix $\mathbf{V}$ provides a basis for expressing signals living on $\mathbf{S}$, and with favorable discrete Fourier transform (DFT)-like properties providing a notion of frequency similar to the one in temporal signal processing. For this reason, the matrix $\mathbf{V}^{-1}$ is often referred to as the graph Fourier transform (GFT) and the projection of $\mathbf{x}$ onto this basis, i.e., $\hat{\mathbf{x}} = \mathbf{V}^{-1}\mathbf{x}$ as the GFT signal.

Filtering on Graphs. Given a graph $\mathbf{S}$, a classical graph filter (C-GF) of order $L-1$ is the matrix polynomial:

$$\mathbf{H}(\mathbf{p},\mathbf{S}) = \sum_{l=0}^{L-1} p_l \mathbf{S}^l, \tag{6.1}$$

where $\mathbf{p} = [p_0,\dots,p_{L-1}]^\top \in \mathbb{C}^L$ collects the graph filter coefficients (taps). The application of the filter $\mathbf{H}(\mathbf{p},\mathbf{S})$ on a signal $\mathbf{x}$ to obtain a new signal $\mathbf{y}$, i.e., $\mathbf{y} = \mathbf{H}(\mathbf{p},\mathbf{S})\mathbf{x}$, is often referred to as graph filtering or *graph convolution*, as it respects the scale-sum-shift principle of convolution. With a few simple calculations, it is easy to show that in the (graph) frequency domain, a graph convolution is expressed as a pointwise multiplication; this is the (graph) convolution theorem, which can be expressed as follows:

$$\mathbf{y} = \sum_{l=0}^{L-1} p_l \mathbf{S}^l \mathbf{x} \qquad \hat{\mathbf{y}} = \sum_{l=0}^{L-1} p_l \boldsymbol{\Lambda}^l \hat{\mathbf{x}} \tag{6.2}$$

with $\hat{\mathbf{y}} = \mathbf{V}^{-1}\mathbf{y}$ the GFT of $\mathbf{y}$. Notice that such filter is isotropic, meaning that for each $l = 0,\dots,L-1$, the filter coefficient p_l is shared among all the nodes of the shifted signal $\mathbf{S}^l\mathbf{x}$; for this reason a C-GF is an example of a *node-invariant* graph filter.

A more versatile and flexible version of (6.1) is the so-called *node-variant* graph filter [9], which allows a per-node weighting scheme of each shifted version of the input signal. Due to its relevance in this work, we distinguish among two flavours of a NV-GF, henceforth referred to as *type-I* and *type-II*, defined, for a given a graph $\mathbf{S}$ and fixed order $L-1$, respectively as:

$$\mathbf{H}_I(\mathbf{P},\mathbf{S}) = \sum_{l=0}^{L-1} \mathrm{Diag}(\mathbf{p}_l)\mathbf{S}^l, \tag{6.3}$$

$$\mathbf{H}_{II}(\mathbf{P},\mathbf{S}) = \sum_{l=0}^{L-1} \mathbf{S}^l\,\mathrm{Diag}(\mathbf{p}_l), \tag{6.4}$$

where $\mathbf{P} \in \mathbb{C}^{N\times L}$ collects the filter coefficients $\mathbf{P} = [\mathbf{p}_0,\dots,\mathbf{p}_{L-1}]$ with $\mathbf{p}_l := [p_{l1},\dots,p_{lN}]^\top \in \mathbb{C}^N$ the l-th hop filter tap vector. As a short-hand notation, we will use $\mathbf{H}_I$ and $\mathbf{H}_{II}$ to refer to the NV-GF in (6.3) and (6.4), respectively; when convenient for clarity of exposition, we will explicitly write $\mathbf{H}_I(\mathbf{P},\mathbf{S})$ or $\mathbf{H}_{II}(\mathbf{P},\mathbf{S})$ concordantly. The application of a NV-GF on a signal $\mathbf{x}$ to obtain a new signal $\mathbf{y}$ will be referred to as *node-variant graph convolution*. From a theoretical point of view, both NV-GF types have the same expressive behavior, yet the order of shifting and weighing is reversed. Specifically, in type-I, each node performs a linear combination of the (shifted) signal values of neighboring nodes, where the weights of the linear combination are neighbor-specific; in type-II, each node performs a linear combination of the (shifted) signal values of neighboring nodes, which have been already scaled by such nodes. Nonetheless, both can be implemented with the same complexity and in a distributed manner [9].

Dual Graph. Although often not explicitly stated in the academic literature, the support of the GFT signal $\hat{\mathbf{x}}$ is described by the eigenvalues $\boldsymbol{\lambda}$ of $\mathbf{S}$, which correspond to a discretization/sampling of a continuous domain, either the real line $\mathbb{R}$ or the complex plane $\mathbb{C}$. This is consistent with the discrete signal processing notion of frequency domain: when $\mathbf{S}$ represents a cycle graph, possibly capturing the time domain, its eigenvector matrix $\mathbf{V}^{-1}$ coincides with the (normalized) DFT matrix, and its eigenvalues $\boldsymbol{\lambda}$ with the complex frequencies on the unit circle, i.e., $\boldsymbol{\lambda} = [1, e^{-j2\pi/N}, \dots, e^{-j2\pi(N-1)/N}]$. However, a modern line of research attempts to model the (graph) frequency domain by means of a graph [11]. The motivation behind this line of research relies on the fact that classical signal processing tasks usually performed in the frequency domain, such as frequency-shifting, do not have their counterpart in GSP. Furthermore, given that a graph signal is inherently associated with a graph structure, it is desirable to establish a corresponding Fourier representation that is also inherently linked to a graph structure. This leads to the notion of a dual graph[1] $\mathbf{S}_f = \mathbf{V}_f \boldsymbol{\Lambda}_f \mathbf{V}_f^{-1}$, which represents the support for the GFT signal $\hat{\mathbf{x}}$. Because we want the GFT $\mathbf{V}_f^{-1}$ associated to the dual graph to map $\hat{\mathbf{x}}$ back to the signal $\mathbf{x}$, i.e., $\mathbf{x} = \mathbf{V}_f^{-1}\hat{\mathbf{x}}$, and we know that $\mathbf{x} = \mathbf{V}\hat{\mathbf{x}}$, we must have $\mathbf{V}_f = \mathbf{V}^{-1}$.

Thus, the primal graph provides *spectral templates* for the graph frequency domain, i.e., the eigenvectors $\mathbf{V}_f$ for the dual graph $\mathbf{S}_f$ are known once we know those of $\mathbf{S}$. The only unknown is then the eigenvalue matrix $\boldsymbol{\Lambda}_f := \mathrm{Diag}(\boldsymbol{\lambda}_f)$, which can be found, for instance, with an axiomatic or an optimization approach [11], or in a data-driven manner as we will show in Section 6.4. Although [12] proposes $\boldsymbol{\lambda}_f = \boldsymbol{\lambda}^\star$, we do not see this as a favorable definition, since in our view it only holds when specified to the "temporal" graph; in all the other cases, especially for undirected graphs, it would implies that primal and dual eigenvalues always coincide. Such an interpretation would be inconsistent with the desirable properties highlighted in [11, Axioms (1-3)].

6.3. An Encompassing Convolution Theorem

In this section we propose a convolution theorem which encompasses the graph convolution theorem [cf. (6.2)] introduced in Section 6.2 and the convolution theorem related to time-varying filters, which will be introduced later on to highlight similarities and differences. This generalization is made possible by using the node-variant graph filters (6.3) and (6.4) with an appropriate parametrization of the filter coefficients. Specifically, we show how a limited order NV-GF in the primal domain can be expressed as a limited order NV-GF in the dual domain. This is

[1]Not to be confused with the dual graph notion in graph theory, as the graph which has a vertex for each face of the original graph.

formally stated in the following theorem.

Theorem 5 (Node-variant convolution theorem)**.** *Consider a type-I NV-GF $\mathbf{H}_I$ defined over the graph $\mathbf{S}$ with filter taps $\{\mathbf{p}_l\}_{l=0}^{L-1}$, i.e., $\mathbf{H}_I(\mathbf{P},\mathbf{S})$, and assume that a dual graph $\mathbf{S}_f$ with dual graph frequencies $\boldsymbol{\lambda}_f$ describing the dual domain is given. Assume also that each filter tap vector $\mathbf{p}_l$ can be expressed as a polynomial of order $K-1$ in $\boldsymbol{\lambda}_f$. Then, there exists a set of coefficients $\{\hat{\mathbf{p}}_k\}_{k=0}^{K-1}$ for which the type-I NV-GF $\mathbf{H}_I(\mathbf{P},\mathbf{S})$ in the primal domain corresponds to a type-II NV-GF $\mathbf{H}_{II}$ on the dual graph $\mathbf{S}_f$ with filter taps $\{\hat{\mathbf{p}}_k\}_{k=0}^{K-1}$, i.e., $\mathbf{H}_{II}(\hat{\mathbf{P}},\mathbf{S}_f)$.*

Proof. By multiplying both sides of (6.3) with the GFT matrix $\mathbf{V}^{-1}$, we have:

$$\hat{\mathbf{y}} = \mathbf{V}^{-1}\sum_{l=0}^{L-1}\mathrm{Diag}(\mathbf{p}_l)\mathbf{S}^l\mathbf{x} = \mathbf{V}^{-1}\sum_{l=0}^{L-1}\mathrm{Diag}(\mathbf{p}_l)\mathbf{V}\boldsymbol{\Lambda}^l\hat{\mathbf{x}}. \tag{6.5}$$

Next, we use a basis expansion model (BEM) [21] to express the NV filter coefficients $\{\mathbf{p}_l\}_{l=0}^{L-1}$ as a linear combination of a dual graph dependent basis. Specifically, we express each $\mathbf{p}_l$ through powers of the dual eigenvalues $\boldsymbol{\lambda}_f$, representing our basis expansion; that is:

$$\mathbf{p}_l = \sum_{k=0}^{K-1} c_{lk}\boldsymbol{\lambda}_f^k = \boldsymbol{\Psi}_f\mathbf{c}_l \tag{6.6}$$

with $\boldsymbol{\Psi}_f := [\mathbf{1}, \boldsymbol{\lambda}_f, \ldots, \boldsymbol{\lambda}_f^{K-1}]$ the Vandermonde matrix of dual eigenvalues and $\mathbf{c}_l := [c_{l0}, \ldots, c_{l(K-1)}]^\top$ the expansion coefficients for the l-th primal filter tap vector $\mathbf{p}_l$. With this choice, substituting (6.6) in (6.5), we have:

$$\begin{aligned}\hat{\mathbf{y}} &= \mathbf{V}^{-1}\sum_{l=0}^{L-1}\mathrm{Diag}(\sum_{k=0}^{K-1} c_{lk}\boldsymbol{\lambda}_f^k)\mathbf{V}\boldsymbol{\Lambda}^l\hat{\mathbf{x}}\\ &= \sum_{k=0}^{K-1}\mathbf{V}^{-1}\mathrm{Diag}(\boldsymbol{\lambda}_f^k)\mathbf{V}\mathrm{Diag}(\sum_{l=0}^{L-1} c_{lk}\boldsymbol{\lambda}^l)\hat{\mathbf{x}}\\ &= \sum_{k=0}^{K-1}\mathbf{S}_f^k\mathrm{Diag}(\hat{\mathbf{p}}_k)\hat{\mathbf{x}}\end{aligned} \tag{6.7}$$

where $\hat{\mathbf{p}}_k := \sum_{l=0}^{L-1} c_{lk}\boldsymbol{\lambda}^l = \boldsymbol{\Psi}\hat{\mathbf{c}}_k$ is the k-th hop filter tap vector on the *dual* graph, with $\boldsymbol{\Psi} := [\mathbf{1}, \boldsymbol{\lambda}, \ldots, \boldsymbol{\lambda}^{L-1}]$ the Vandermonde matrix of primal eigenvalues, and $\hat{\mathbf{c}}_k := [c_{0k}, \ldots, c_{(L-1)k}]^\top$ the expansion coefficients for the k-th dual filter tap vector $\hat{\mathbf{p}}_k$. So in the frequency domain, we also obtain a NV-GF denoted as $\mathbf{H}_{II} = \sum_{k=0}^{K-1}\mathbf{S}_f^k\mathrm{diag}(\hat{\mathbf{p}}_k)$. Whenever the dependency on the dual filter coefficients and shift operator is necessary, we use $\mathbf{H}_{II}(\hat{\mathbf{P}},\mathbf{S}_f)$, where $\hat{\mathbf{P}}$ is the $N\times K$ matrix of coefficients $\hat{\mathbf{P}} = [\hat{\mathbf{p}}_0, \ldots, \hat{\mathbf{p}}_{K-1}]$. ■

This theorem allows us to delineate a general convolution theorem encompassing (6.2) as a special case, which relies on node-variant graph filtering and pictorially described in Fig. 6.1, as follows:

$$\mathbf{y} = \sum_{l=0}^{L-1} \mathrm{Diag}(\mathbf{p}_l)\mathbf{S}^l\mathbf{x} \qquad \hat{\mathbf{y}} = \sum_{k=0}^{K-1} \mathbf{S}_f^k \mathrm{Diag}(\hat{\mathbf{p}}_k)\hat{\mathbf{x}} \tag{6.8}$$

$$\mathbf{p}_l = \mathbf{\Psi}_f \mathbf{c}_l \qquad \hat{\mathbf{p}}_k = \mathbf{\Psi}\hat{\mathbf{c}}_k \tag{6.9}$$

The connection between the primal and the dual node-variant graph filters defined in (6.8) is given by the $K \times L$ expansion coefficients conveniently stored in the matrix $\mathbf{C} = [\mathbf{c}_0, \ldots, \mathbf{c}_{L-1}] = [\hat{\mathbf{c}}_0, \ldots, \hat{\mathbf{c}}_{K-1}]^\top$. This enables also to concisely express the node-variant coefficients in the primal and dual domain as $\mathbf{P} = \mathbf{\Psi}_f\mathbf{C}$ and $\hat{\mathbf{P}} = \mathbf{\Psi}\mathbf{C}^\top$, respectively.

Remark 3. *The same type of theorem construction can be obtained by reversing the types of filters adopted in the primal and dual domain; that is, applying a type-II NV-GF in the primal domain is equivalent to applying a type-I NV-GF in the dual domain, with the graph filter coefficients following the parametrization in* (6.9)*.*

Corollary 1. *Given a graph signal* $\mathbf{x}$*, the application of a node-variant graph filter* $\mathbf{H}_I(\mathbf{P}, \mathbf{S})$ *in the primal domain followed by the GFT* $\mathbf{V}^{-1}$ *is equivalent to the application of the GFT followed by a node-variant graph filter* $\mathbf{H}_{II}(\hat{\mathbf{P}}, \mathbf{S}_f)$ *in the dual domain. In other words, it holds (see also Fig. 6.1):*

$$\mathbf{V}^{-1}\mathbf{H}_I(\mathbf{P}, \mathbf{S}) = \mathbf{H}_{II}(\hat{\mathbf{P}}, \mathbf{S}_f)\mathbf{V}^{-1}. \tag{6.10}$$

In temporal signal processing, the frequency representation of windowing in the time domain is the convolution between the spectra of the signal and the window. Because a node-variant convolution of order $L-1$ is nothing else than the application of L windows on shifted versions of the input graph signal $\mathbf{x}$, a similar result can be derived in the graph setting; the following corollary expresses this.

Corollary 2. *Given an input graph signal* $\mathbf{x}$ *and a type-I NV-GF* $\mathbf{H}_I(\mathbf{P}, \mathbf{S})$*, with each* $\{\mathbf{p}_l\}_{l=0}^{L-1}$ *parametrized as in* (6.9)*, a node-variant graph convolution of order* $L-1$ *in the primal domain is equivalent to the sum of* L *classical graph convolutions in the dual domain, each one with as input a (modulated) version of* $\hat{\mathbf{x}}$*; that is:*

$$\hat{\mathbf{y}} = \sum_{l=0}^{L-1} \mathbf{H}(\mathbf{c}_l, \mathbf{S}_f)(\boldsymbol{\lambda}^l \odot \hat{\mathbf{x}}) \tag{6.11}$$

$$= \mathbf{H}(\mathbf{c}_0, \mathbf{S}_f)\hat{\mathbf{x}} + \ldots + \mathbf{H}(\mathbf{c}_{L-1}, \mathbf{S}_f)(\boldsymbol{\lambda}^{L-1} \odot \hat{\mathbf{x}}) \tag{6.12}$$

Proof. From the first equality of (6.7), we have:

$$\begin{aligned}\hat{\mathbf{y}} &= \sum_{l=0}^{L-1}\sum_{k=0}^{K-1} c_{lk}\mathbf{V}^{-1}\mathrm{Diag}(\boldsymbol{\lambda}_f^k)\mathbf{V}\mathbf{\Lambda}^l\hat{\mathbf{x}} \\ &= \sum_{l=0}^{L-1}\left(\sum_{k=0}^{K-1} c_{lk}\mathbf{S}_f^k\right)\mathbf{\Lambda}^l\hat{\mathbf{x}} \\ &= \sum_{l=0}^{L-1} \mathbf{H}(\mathbf{c}_l, \mathbf{S}_f)(\boldsymbol{\lambda}^l \odot \hat{\mathbf{x}}) \end{aligned} \tag{6.13}$$

$$\begin{tikzcd}
\mathbf{x} \arrow[r] \arrow[d] & \boxed{\mathbf{H}_I(\mathbf{P},\mathbf{S})} \arrow[r] & \mathbf{y} \arrow[d] \\
\boxed{\mathbf{V}^{-1}} \arrow[d] & & \boxed{\mathbf{V}^{-1}} \arrow[d] \\
\hat{\mathbf{x}} \arrow[r] & \boxed{\mathbf{H}_{II}(\hat{\mathbf{P}},\mathbf{S}_f)} \arrow[r] & \hat{\mathbf{y}}
\end{tikzcd}$$

Figure 6.1: General convolution theorem. A node-variant graph convolution in the primal domain is equivalent to a node-variant graph convolution in the dual domain.

Notice how the filter coefficients in (6.13) are the expansion coefficients $\mathbf{c}_l$ associated to the primal filter coefficients $\mathbf{p}_l$. ■

An important consequence of Corollary 2 is that windowing in the primal domain is equivalent to a C-GF in the dual domain; we will rely upon this when introducing non-stationary signal models in Section 6.3.3.

6.3.1. Consistency with the graph convolution theorem

Because a C-GF is a NV-GF with constant filter taps, we expect that our introduced theory encompasses the existing one. Indeed, we can formally show that the graph convolution theorem (6.2) falls within the introduced theory. To see this, consider $\mathbf{p}_l = p_l\mathbf{1}$, $\forall\, l$, i.e., the case in which each vector of filter taps $\mathbf{p}_l$ is constant over the nodes, thus corresponding to the C-GF as in (6.1). The column space of $\mathbf{P}$ is then one-dimensional, specifically spanned by the all-one vector. By construction, the first column of the Vandermonde matrix is the all-one vector. Thus any $\boldsymbol{\lambda}_f$ can be used as it will be zeroed-out by the matrix $\mathbf{C}$. As we will see later on, this also means that we cannot learn any dual graph from stationary graph signals, since any $\boldsymbol{\lambda}_f$ would suffice. Specifically, from (6.9), we have that $\mathbf{c}_l$ necessarily needs to be $\mathbf{c}_l = [p_l, \mathbf{0}^\top]^\top$, and overall:

$$[p_1\mathbf{1},\ldots,p_{L-1}\mathbf{1}] = \begin{bmatrix} 1 & \cdots & \lambda_{0,f}^{K-1} \\ \vdots & \ddots & \vdots \\ 1 & \cdots & \lambda_{f,N-1}^{(K-1)} \end{bmatrix} \begin{bmatrix} p_1 & \cdots & p_{L-1} \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix} \tag{6.14}$$

meaning that only the first row $\hat{\mathbf{c}}_0$ of $\mathbf{C}$ is different from zero. In particular, from the right equation in (6.9) this implies that:

$$\hat{\mathbf{y}} = \mathbf{S}_f^0 \operatorname{Diag}(\hat{\mathbf{p}}_0)\hat{\mathbf{x}} = \operatorname{Diag}(\boldsymbol{\Psi}\mathbf{p})\hat{\mathbf{x}}. \tag{6.15}$$

This shows how the proposed theory fits within the principle that a classical graph convolution (node-invariant GF) is a pointwise multiplication in the frequency domain.

Likewise, one can similarly show that if $\hat{\mathbf{p}}_k = \hat{p}_k\mathbf{1}$, $\forall\, k$, i.e., the case in which each vector of filter taps $\hat{\mathbf{p}}_k$ in the frequency domain is constant over the nodes (frequencies), then only $\mathbf{c}_0 = \hat{\mathbf{p}} := [\hat{p}_0, \ldots, \hat{p}_{K-1}]^\top$ is different from zero. This leads to a pointwise multiplication (windowing) [cf. (6.9) left] in the primal domain:

$$\mathbf{y} = \operatorname{Diag}(\mathbf{p}_0)\mathbf{S}^0\mathbf{x} = \operatorname{Diag}(\boldsymbol{\Psi}_f\hat{\mathbf{p}})\mathbf{x}, \tag{6.16}$$

which also complies with Corollary 2.

Another interesting relation arises when considering a NV-GF with $p_{li} = p_i$ for all l; i.e., the case in which each node i uses the same weight p_i (possibly different from p_j of node j) for the diffused sequence $\{\mathbf{x}, \mathbf{S}\mathbf{x}, \ldots, \mathbf{S}^{L-1}\mathbf{x}\}$. In this case, multiple $\boldsymbol{\lambda}_f$ and $\mathbf{C}$ satisfy the theorem; for instance we can choose $\boldsymbol{\lambda}_f = [p_1, \ldots, p_N]^\top$ and $\mathbf{c}_l = [0, 1, \mathbf{0}^\top]^\top$, i.e.,:

$$\begin{bmatrix} p_1\mathbf{1}^\top \\ \vdots \\ p_N\mathbf{1}^\top \end{bmatrix} = \begin{bmatrix} 1 & p_1 & \cdots & p_1^{K-1} \\ \vdots & \vdots & \vdots & \\ 1 & p_N & \cdots & p_N^{(K-1)} \end{bmatrix} \begin{bmatrix} 0 & \cdots & 0 \\ 1 & \cdots & 1 \\ \vdots & \cdots & \vdots \\ 0 & \cdots & 0 \end{bmatrix} \tag{6.17}$$

This implies that:

$$\hat{\mathbf{y}} = \mathbf{S}_f \operatorname{Diag}(\hat{\mathbf{p}}_1)\hat{\mathbf{x}} = \mathbf{S}_f \operatorname{Diag}(\boldsymbol{\Psi}\mathbf{1})\hat{\mathbf{x}}, \tag{6.18}$$

so that each $\hat{x}_i$ is multiplied with $\hat{p}_{1i} = [1, \lambda_i, \ldots, \lambda_i^{L-1}]\mathbf{1}$.

6.3.2. Relationship with time-varying channel propagation

The proposed theory also generalizes, to the graph setting, concepts which are familiar in the context of time-varying channel propagation [22], arising for instance in mobile communication scenarios. In that case, the received signal y at time n, i.e., $y[n]$[2], is modeled as:

$$y[n] = \sum_{l=0}^{L-1} p[n,l]\,x[n-l], \tag{6.19}$$

where $p[n,l]$ denotes the channel impulse response of the l-th path at the n-th time instant, and $x[n-l]$ is the transmitted signal at the $(n-l)$-th time instant. The gains

[2]We use square brackets to indicate that the argument is a time index and not a graph node.

associated to the different paths are assumed to be time-varying and approximated by a basis expansion model [21]; specifically:

$$\mathbf{p}_l = \sum_{k=0}^{K-1} c_{lk}\mathbf{b}_k, \tag{6.20}$$

where $\mathbf{p}_l = [p[0,l],\dots,p[N-1,l]]^\top$ stores the evolution of the filter impulse response over the N time instants, $\mathbf{b}_k \in \mathbb{R}^N$ is the k-th basis function, and c_{lk} is the coefficient associated to the l-th path and the k-th basis function. This alleviates the effort of having to deal with NL channel coefficients (usually a very high number), and converts the model into a simpler one with only LK BEM coefficients.

It is easy to show that we can write (6.19) in matrix-vector form, by taking into account (6.20), as:

$$\mathbf{y} = \sum_{k=0}^{K-1} \mathrm{Diag}(\mathbf{b}_k)\left(\sum_{l=0}^{L-1} c_{lk}\mathbf{D}^l\right)\mathbf{x} \tag{6.21}$$

where $\mathbf{x} = [x[0],\dots,x[N-1]]^\top$ and $\mathbf{D}$ is the $N\times N$ lower delay matrix; notice how the matrix $\sum_{l=0}^{L-1} c_{lk}\mathbf{D}^l$ implements a standard convolutional filter in time and observe its similarities with the left equation in (6.8). Next, denote with $\mathbf{F}\in\mathbb{C}^{N\times N}$ the normalized DFT matrix, and with $\mathbf{f}_k$ its kth column; the classical complex exponential BEM uses the Fourier basis as basis functions in (6.20), i.e., $\mathbf{b}_k = \sqrt{N}\mathbf{f}_k$. As such the gains $\mathbf{p}_l$ associated to the l-th path [cf. (6.20)] are modelled as smoothly-varying over time and hence expressed with a small number K of Fourier basis. Increasing K accommodates for faster changes.

With this choice, (6.21) can be expressed in the frequency domain as:

$$\begin{aligned}\hat{\mathbf{y}} = \mathbf{F}\mathbf{y} &= \sum_{k=0}^{K-1} \mathbf{F}\,\mathrm{Diag}(\sqrt{N}\mathbf{f}_k)\sum_{l=0}^{L-1} c_{lk}\mathbf{F}^H\mathrm{Diag}(\sqrt{N}\mathbf{f}_l)\mathbf{F}\mathbf{x}\\ &= \sum_{l=0}^{L-1}\left(\sum_{k=0}^{K-1} c_{lk}\mathbf{D}^k\right)\mathrm{Diag}(\sqrt{N}\mathbf{f}_l)\hat{\mathbf{x}}.\end{aligned} \tag{6.22}$$

While in (6.21) the matrix $\mathbf{D}$ shifts in the time domain, in (6.22) it shifts in the frequency domain; however, such shift matrix is the same in both domains. This is different when looking at the graph counterpart in (6.7), where the two shift matrices might be different.

Remark 4. *It is worthwhile to point out that the notion of smoothness for signals and filter coefficients is different in temporal and graph signal processing. In the time domain the basis to express smoothness for signals and filter coefficients is the same, both coinciding with the normalized DFT matrix. In GSP, the basis to express smoothness of graph signals and graph filter coefficients is different. Our theory shows that smoothness of graph signals is determined by the eigenvectors of the primal graph, while smoothness of the filter coefficients is determined by the Vandermonde matrix containing the dual graph frequencies.*

All in all (6.22) is the time domain counterpart of (6.7), by choosing the primal eigenvector matrix $\mathbf{V}$ to be $\mathbf{V}=\mathbf{F}^H$ and the basis functions $\boldsymbol{\lambda}_f^k$ to be $\boldsymbol{\lambda}_f^k = \sqrt{N}\mathbf{f}_k$.

6.3.3. NON-STATIONARITY

In this section we study how and where (non-)stationarity of graph signals stands within the introduced theory. From [17], a process $\mathbf{y}$ is said to be weakly stationary on a GSO $\mathbf{S}$ if the covariance matrix $\mathbf{C}_y := \mathbb{E}[\mathbf{y}\mathbf{y}^H]$ commutes with $\mathbf{S}$ or, equivalently, if $\mathbf{y}$ can be written as the output of a C-GF $\mathbf{H}$ [cf. (6.1)] when excited with a white input $\mathbf{x}$, i.e., $\mathbf{y} = \mathbf{H}\mathbf{x}$. As a consequence, the covariance matrix $\mathbf{C}_{\hat{y}}$ of the GFT process $\hat{\mathbf{y}}$ is diagonal, revealing the power spectral density of the process $\mathbf{y}$ on its diagonal.

While [17] offers conditions to identify and model stationary graph signals, it does not explore non-stationary signal models; this is our first attempt in that direction. Worth to mention is the work of [19], where a network topology identification approach is put forth to learn the GSO $\mathbf{S}$ given a set of realizations of a non-stationary graph signal $\mathbf{y}$ modelled with a classical graph convolution (6.2). There, however, non-stationarity is only considered with respect to node-invariant GFs, thus restricting the non-stationarity model taxonomy. Indeed, the (non-)stationarity of a random graph signal $\mathbf{y}$ obtained as $\mathbf{y} = \mathbf{H}\mathbf{x}$ for a generic graph filter $\mathbf{H}$ and an excitation input $\mathbf{x}$, depends either on the type of graph filter or the properties of the input $\mathbf{x}$. Precisely a non-stationary graph signal $\mathbf{y}$ can be either modelled as the output of a shift-invariant graph filter with a non-stationary input or as the output of a node-variant graph filter when excited with a white input. The following property and propositions formally describe these claims.

Property 1. Given input $\mathbf{x}$ and output $\mathbf{y} = \mathbf{H}_I(\mathbf{P},\mathbf{S})\mathbf{x}$, it holds:

$$\mathbf{C}_y = \mathbf{H}_I(\mathbf{P},\mathbf{S})\mathbf{C}_x\mathbf{H}_I(\mathbf{P},\mathbf{S})^H \tag{6.23}$$

with $\mathbf{C}_x := \mathbb{E}[\mathbf{x}\mathbf{x}^H]$. Moreover:

$$\mathbf{C}_{\hat{y}} = \mathbf{V}^{-1}\mathbf{C}_y\mathbf{V}. \tag{6.24}$$

Depending on the structure of the graph filter $\mathbf{H}_I(\mathbf{P},\mathbf{S})$ and the input signal $\mathbf{x}$, the ensuing propositions can be derived. Unless explicitly stated differently, we assume that the graph filter coefficient matrix $\mathbf{P}$ respects the parametrization in (6.9), i.e., $\mathbf{P} = \mathbf{\Psi}_f\mathbf{C}$, for some order L, K.

Proposition 1. *If $\mathbf{x}$ is a white graph signal, then [cf. Corollary 1]:*

$$\mathbf{C}_y = \mathbf{H}_I(\mathbf{P},\mathbf{S})\mathbf{H}_I(\mathbf{P},\mathbf{S})^H \tag{6.25}$$

$$\mathbf{C}_{\hat{y}} = \mathbf{H}_{II}(\hat{\mathbf{P}},\mathbf{S}_f)\mathbf{H}_{II}(\hat{\mathbf{P}},\mathbf{S}_f)^H. \tag{6.26}$$

In general, this means that $\mathbf{y}$ is non-stationary on $\mathbf{S}$ and $\hat{\mathbf{y}}$ is non-stationary on $\mathbf{S}_f$.

Proposition 2. *If $L = 1$ and $\mathbf{x}$ is a white signal (stationary by definition), then $\hat{\mathbf{y}}$ is stationary on $\mathbf{S}_f$.*

From (6.24), we can see that if $\mathbf{C}_y$ is diagonal, then $\mathbf{S}_f$ commutes with $\mathbf{C}_{\hat{y}}$ and as such $\hat{\mathbf{y}}$ is stationary on $\mathbf{S}_f$, with a power spectral density (in the primal domain) equal to the eigenvalues of $\mathbf{C}_{\hat{y}}$. This can happen only if $L = 1$ and $\mathbf{x}$ is a white signal, where the convolution simply becomes $\mathbf{y} = \mathrm{Diag}(\mathbf{p}_0)\mathbf{x}$, i.e., a windowing in the primal domain. The covariance matrix $\mathbf{C}_y$ is then $\mathbf{C}_y = \mathrm{Diag}(\mathbf{p}_0)^2$ and the cross correlation in $\mathbf{y}$ is zero. In this case the covariance matrix $\mathbf{C}_{\hat{y}}$ of the process $\hat{\mathbf{y}}$ in the dual domain is not diagonal in general, since it reads as:

$$\mathbf{C}_{\hat{y}} = \mathbf{V}^{-1}\,\mathrm{Diag}(\mathbf{p}_0)^2\mathbf{V}. \tag{6.27}$$

From (6.27), we can then conclude that $\mathbf{y}$ is non-stationary on $\mathbf{S}$ and, more importantly, $\mathbf{S}_f\mathbf{C}_{\hat{y}} = \mathbf{C}_{\hat{y}}\mathbf{S}_f$, i.e., $\mathbf{S}_f$ commutes with $\mathbf{C}_{\hat{y}}$. This implies that $\hat{\mathbf{y}}$ can be expressed as the output of a node-invariant graph filter in the dual domain when excited with white input, i.e., $\hat{\mathbf{y}} = \mathbf{H}(\hat{\mathbf{p}}, \mathbf{S}_f)\hat{\mathbf{x}}$ for some limited order filter coefficients $\hat{\mathbf{p}}$, rendering the estimation of the dual graph (see Section 6.4) a node-invariant graph filter estimation problem [9, 23]. This extends the classical notion of windowing in time domain (for instance used in power spectral density estimation), which corresponds to a frequency-invariant convolution in the frequency domain. A generalization of this is given for $L > 1$ and a general signal $\mathbf{x}$, for which Corollary 2 applies.

Proposition 3. *If $L = 1$ and $\mathbf{x}$ is a non-white yet stationary graph signal with covariance $\mathbf{C}_x = \mathbf{V}\mathbf{\Lambda}_x\mathbf{V}^H$, then $\hat{\mathbf{y}}$ is not stationary on $\mathbf{S}_f$ since:*

$$\mathbf{C}_{\hat{y}} = \mathbf{H}(\mathbf{c};\mathbf{S}_f)\mathbf{\Lambda}_x\mathbf{H}(\mathbf{c};\mathbf{S}_f)^H \tag{6.28}$$

for some coefficients $\mathbf{c} \in \mathbb{R}^K$, does not commute with $\mathbf{S}_f$.

Proposition 4. *If $\mathbf{p}_l = p_l\mathbf{1}$, $\forall\, l$, and $\mathbf{x}$ is not stationary on $\mathbf{S}$, then $\mathbf{y}$ is not stationary on $\mathbf{S}$ and $\hat{\mathbf{y}}$ is not stationary on $\mathbf{S}_f$. Indeed we have that the covariance matrix of $\mathbf{y}$:*

$$\mathbf{C}_y = \mathbf{H}(\mathbf{p};\mathbf{S})\mathbf{C}_x\mathbf{H}(\mathbf{p};\mathbf{S})^H \tag{6.29}$$

does not commute with $\mathbf{S}$; likewise $\mathbf{S}_f$ does not commute with $\mathbf{C}_{\hat{\mathbf{y}}} = \mathbf{V}^{-1}\mathbf{C}_{\mathbf{y}}\mathbf{V}$.

In other words, a node-invariant graph convolution of a non-stationary process $\mathbf{x}$ results in a non-stationary process $\mathbf{y}$ on $\mathbf{S}$ (and non-stationary GFT $\hat{\mathbf{y}}$ on $\mathbf{S}_f$). While this result is not novel, we include it here to ensure a comprehensive coverage.

The introduced propositions provide a way to artificially generate non-stationary graph signals on a given GSO $\mathbf{S}$ by filtering white noise, as explained next.

Generating non-stationary graph signals. Remember that $\mathbf{C}_{\hat{\mathbf{y}}} = \mathbf{V}^{-1}\mathbf{C}_{\mathbf{y}}\mathbf{V}$ has to be diagonal for $\mathbf{y}$ to be stationary on $\mathbf{S}$. Thus, non-stationary graph signals on $\mathbf{S}$ can be generated as long as $\mathbf{C}_{\mathbf{y}}$ is not diagonalizable by $\mathbf{V}$ (which would render $\mathbf{C}_{\hat{\mathbf{y}}}$ diagonal). In particular, if we want our GFT random process $\hat{\mathbf{y}}$ to have a specific covariance matrix equal to $\mathbf{C}_{\hat{\mathbf{y}}} = \mathbf{V}^{-1}\mathbf{C}_{\mathbf{y}}\mathbf{V}$, for some positive semidefinite (PSD) $\mathbf{C}_{\mathbf{y}}$ we can generate random samples $\hat{\mathbf{y}}$ as:

$$\hat{\mathbf{y}} = \mathbf{V}^{-1}\mathbf{R}\mathbf{V}\hat{\mathbf{x}} \tag{6.30}$$

where $\mathbf{R}$ is a matrix such that $\mathbf{C}_y = \mathbf{R}\mathbf{R}^H$ and $\hat{\mathbf{x}}$ is a white input; equivalently, in the primal domain:

$$\mathbf{y} = \mathbf{R}\mathbf{x}. \tag{6.31}$$

A simple example of such generation process is given by setting $\mathbf{C}_\mathbf{y} = \mathrm{Diag}(\mathbf{p})$ for some $\mathbf{p} \succcurlyeq \mathbf{0}$, so that (6.31) is a windowing operation in the primal domain, corresponding to a node-invariant graph convolution [cf. (6.30)] in the dual domain. With this choice, however, $\hat{\mathbf{y}}$ is stationary on $\mathbf{S}_f$ [cf. Proposition 2]. If this is not sought-after, a general non diagonal PSD $\mathbf{C}_\mathbf{y}$ should be used.

6.4. DUAL GRAPH IDENTIFICATION

So far we have assumed the knowledge of $\mathbf{S}_f$. In this section, we put forth a data-driven procedure to learn the dual graph eigenvalues $\boldsymbol{\lambda}_f$ in such a way that the resulting graph $\mathbf{S}_f$ respects the theory developed in Section 6.3. For simplicity, we restrict our attention to the case of an undirected primal graph with real-valued GSO $\mathbf{S}$ and real-valued graph signals and filter coefficients. The problem setting is the following: consider T graph signals $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_T]$ which can be modelled as non-stationary on the graph $\mathbf{S}$ as the result of filtering T (possibly unknown) input graph signals $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_T]$ with a NV-GF $\mathbf{H}_I(\mathbf{P}, \mathbf{S})$, i.e., $\mathbf{Y} = \mathbf{H}_I(\mathbf{P}, \mathbf{S})\mathbf{X}$. In particular, similar to temporal signal processing, we assume that the orders K and L are much smaller than N. Then the problem of identifying the dual graph can be formalized as follows:

Given $\mathbf{Y}$ (and possibly $\mathbf{X}$), find a dual graph $\mathbf{S}_f$ which is consistent with Theorem 5. In other words, find the dual eigenvalues $\boldsymbol{\lambda}_f$ such that a NV-GF $\mathbf{H}_I$ on $\mathbf{S}$ with $L \ll N$ can be expressed as a NV-GF $\mathbf{H}_{II}$ on $\mathbf{S}_f$ with $K \ll N$.

To tackle this problem, we adopt a two-step approach: in *step i)* we learn the filter taps $\mathbf{P}$ of the NV-GF $\mathbf{H}_I(\mathbf{P}, \mathbf{S})$ which best fit the available data, developing two distinct approaches for the input-output and output-only scenarios; in the second *step ii)* we find the dual eigenvalues $\boldsymbol{\lambda}_f$ by exploiting (6.9), i.e., we fit the model $\mathbf{P} = \boldsymbol{\Psi}_f\mathbf{C}$, which is a specific structured matrix factorization with a Vandermonde factor. We solve this problem by following the recently proposed approach in [2] relying on a subspace method followed by successive convex programming.

6.4.1. GRAPH FILTER ESTIMATION WITH INPUT-OUTPUT DATA

The goal of this section is to estimate the graph filter coefficients $\mathbf{P}$ from a set of data pairs $\mathscr{D} = \{(\mathbf{x}_t, \mathbf{y}_t) \mid t = 1, \ldots, T\}$, all obtained using the same node-varying graph filter. By vectorizing the expression $\mathbf{Y} = \mathbf{H}_I(\mathbf{P}, \mathbf{S})\mathbf{X}$ we obtain:

$$\begin{aligned}
\mathbf{y} &= \sum_{l=0}^{L-1} \mathrm{vec}(\mathrm{Diag}(\mathbf{p}_l)\mathbf{S}^l\mathbf{X}) \\
&= \sum_{l=0}^{L-1} ((\mathbf{S}^l\mathbf{X})^\top \circ \mathbf{I}_N)\mathbf{p}_l \\
&= [\mathbf{X}^\top \circ \mathbf{I}_N \cdots (\mathbf{S}^{L-1}\mathbf{X})^\top \circ \mathbf{I}_N]\,\mathrm{vec}(\mathbf{P}) \\
&= \mathbf{A}\,\mathrm{vec}(\mathbf{P})
\end{aligned} \tag{6.32}$$

where $\mathbf{y} = [\mathbf{y}_1, \ldots, \mathbf{y}_T]^\top$, $\mathbf{A} = [\mathbf{X}^\top \circ \mathbf{I}_N \cdots (\mathbf{S}^{L-1}\mathbf{X})^\top \circ \mathbf{I}_N]$ and $\circ$ denotes the Khatri-Rao product. An estimate of $\mathbf{P}$ can then be obtained as:

$$\mathbf{P} = \operatorname{unvec}(\mathbf{A}^\dagger \mathbf{y}), \tag{6.33}$$

where $\mathbf{A}^\dagger = (\mathbf{A}^H\mathbf{A})^{-1}\mathbf{A}^H$ is the pseudo-inverse of $\mathbf{A}$.

6.4.2. Graph Filter Estimation with Output-only Data

The goal of this section is to estimate the graph filter coefficients $\mathbf{P}$ with the only knowledge of the output graph signals $\mathbf{Y} \in \mathbb{R}^{N\times T}$. We assume that each $\mathbf{y}_t$ can be modelled as the output of a NV-GF when excited with a white input $\mathbf{x}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, which is not directly observable. A viable approach is then to fit the (empirical) second order information of the process which, together with the filter parametrization (6.3), leads to the following optimization problem:

$$\min_{\mathbf{P}} \|\hat{\mathbf{C}}_y - \mathbf{H}_I(\mathbf{P},\mathbf{S})\mathbf{H}_I(\mathbf{P},\mathbf{S})^\top\|_F^2 \tag{6.34}$$

where $\hat{\mathbf{C}}_y = (1/T)\mathbf{Y}\mathbf{Y}^\top$ is the sample covariance matrix ($\mathbf{Y}$ is already centered).

Problem (6.34) is non-convex in $\mathbf{P}$; thus, to alleviate the non-convexity, we consider instead the simpler (yet, again non-convex) problem:

$$\min_{\mathbf{P},\mathbf{U}} \|\mathbf{R} - \mathbf{H}_I(\mathbf{P},\mathbf{S})\mathbf{U}\|_F^2 \quad \text{s.t.}\ \mathbf{U}^\top\mathbf{U} = \mathbf{I} \tag{6.35}$$

where $\mathbf{R}$ is a square matrix such that $\hat{\mathbf{C}}_y = \mathbf{R}\mathbf{R}^\top$, and $\mathbf{U}$ is an $N \times N$ orthogonal matrix. By exploiting the SVD of the matrix $\mathbf{Y}$, i.e., $\mathbf{Y} = \mathbf{U}_y\mathbf{\Lambda}_y\mathbf{V}_y^\top$, possible choices for $\mathbf{R}$ are $\mathbf{R} = (1/\sqrt{T})\mathbf{U}_y\mathbf{\Lambda}_y\mathbf{U}_y^\top$ and $\mathbf{R} = (1/\sqrt{T})\mathbf{U}_y\mathbf{\Lambda}_y$. Notice that if (6.34) has a solution, then (6.35) has a solution.

Despite the fact that (6.35) is not jointly convex in $\mathbf{P}$ and $\mathbf{U}$, it is convex in $\mathbf{P}$ for a fixed $\mathbf{U}$; moreover, for a fixed $\mathbf{P}$, it reduces to the well-studied orthogonal Procrustes problem [24], for which a closed form solution exists albeit its non-convexity. Thus, an alternating minimization over $\mathbf{P}$ and $\mathbf{U}$ can be put forth, as follows: *a)* first, given the estimate of $\mathbf{U}$ at the $(n-1)$th iteration, i.e., $\mathbf{U}^{(n-1)}$, the estimation problem at the nth iteration for the filter taps matrix $\mathbf{P}$ reads as:

$$\mathbf{P}^{(n)} = \operatorname*{argmin}_{\mathbf{P}} \|\mathbf{R} - \mathbf{H}_I(\mathbf{P},\mathbf{S})\mathbf{U}^{(n-1)}\|_F^2, \tag{36a}$$

which can be solved in closed form by considering $\mathbf{Y} = \mathbf{R}$ and $\mathbf{X} = \mathbf{U}^{(n-1)}$ in (6.32). The solution of (36a) is then *b)* used in the next step to refine the estimate of the unitary matrix $\mathbf{U}$, i.e.,:

$$\mathbf{U}^{(n)} = \operatorname*{argmin}_{\mathbf{U}} \|\mathbf{R} - \mathbf{H}_I(\mathbf{P}^{(n)},\mathbf{S})\mathbf{U}\|_F^2$$
$$\text{s.t.}\ \mathbf{U}^\top\mathbf{U} = \mathbf{I} \tag{36b}$$

for which the closed form solution is $\mathbf{U}^{(n)} = \mathbf{V}_p\mathbf{U}_p^\top$, where $\mathbf{U}_p$ and $\mathbf{V}_p$ are the left and right singular vector matrices, respectively, of the matrix product $\mathbf{R}^\top\mathbf{H}_I(\mathbf{P}^{(n)},\mathbf{S})$, that is, $\mathbf{R}^\top\mathbf{H}_I(\mathbf{P}^{(n)},\mathbf{S}) = \mathbf{U}_p\mathbf{\Lambda}_p\mathbf{V}_p^\top$, with $\mathbf{\Lambda}_p$ the matrix of singular values. The solution of (36b) is then fed again into (36a), unless a predefined number of iterations or stopping criterion is reached.

6.4.3. DUAL GRAPH FREQUENCY ESTIMATION

The next step is to learn the dual graph frequencies $\boldsymbol{\lambda}_f$, which are the only unknowns of the dual GSO $\mathbf{S}_f$. In order to do this, consider again the matrix form of (6.6):

$$\mathbf{P} = \boldsymbol{\Psi}_f(\boldsymbol{\lambda}_f)\mathbf{C} \tag{6.36}$$

where we explicitly wrote $\boldsymbol{\Psi}_f$ as a function of $\boldsymbol{\lambda}_f$ to highlight the fact that the matrix is entirely determined by its second column $\boldsymbol{\lambda}_f$, representing our unknown. Then, the problem we aim to solve can be formally stated as follows:

Given the matrix $\mathbf{P} \in \mathbb{R}^{N\times L}$, recover the input vector $\boldsymbol{\lambda}_f \in \mathbb{R}^N$ and the coefficient matrix $\mathbf{C} \in \mathbb{R}^{K\times L}$ such that (6.36) holds as accurately as possible.

Although the problem can be approached from a pure algebraic point of view as a structured matrix factorization, a pleasing geometrical interpretation of (6.36) is given by interpreting the vectors $\mathbf{p}_0,\ldots,\mathbf{p}_{L-1}$ as function values obtained by sampling L distinct polynomials $p_0(\lambda_f),\ldots,p_{L-1}(\lambda_f)$, all with degree $K-1$, in the same N unknown locations $\lambda_{f,0},\ldots,\lambda_{f,N-1}$. The goal is to recover the original locations (and polynomial coefficients) from the available sampled function values. The only side information we have about these sampling points is *i)* which function they belong to and *ii)* that they are ordered in such a way that the related sampling points are aligned. See [2] and the Supplemental Material for an illustration.

Subspace Fitting. Assume we start by considering the following optimization problem to estimate $\boldsymbol{\lambda}_f$ and $\mathbf{C}$:

$$\min_{\boldsymbol{\lambda}_f,\mathbf{C}} \frac{1}{2}\|\mathbf{P} - \boldsymbol{\Psi}_f(\boldsymbol{\lambda}_f)\mathbf{C}\|_F^2, \tag{6.37}$$

which can be solved, for instance, with an alternating minimization approach, without guarantee of convergence to a global optimum.

An alternative subspace method can be devised when $L \geq K$. We then consider the economy-size SVD of matrix $\mathbf{P}$, i.e., $\mathbf{P} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{Z}^\top$, where $\mathbf{U} \in \mathbb{R}^{N\times K}$ and $\mathbf{Z} \in \mathbb{R}^{L\times K}$ are the left and right singular vectors, respectively, and $\boldsymbol{\Sigma} \in \mathbb{R}^{K\times K}$ is the diagonal matrix of singular values. Since both $\boldsymbol{\Psi}_f(\boldsymbol{\lambda}_f)$ and $\mathbf{U}$ represent a basis for the column space of $\mathbf{P}$, there exists a non-singular matrix $\mathbf{Q} \in \mathbb{R}^{K\times K}$ such that $\boldsymbol{\Psi}_f = \mathbf{U}\mathbf{Q}$. The subspace fitting [13] problem reads then as:

$$\min_{\boldsymbol{\lambda}_f,\mathbf{Q}} \frac{1}{2}\|\boldsymbol{\Psi}_f(\boldsymbol{\lambda}_f) - \mathbf{U}\mathbf{Q}\|_F^2, \tag{6.38}$$

which, upon substituting the pseudoinverse solution $\mathbf{Q} = \mathbf{U}^\dagger\boldsymbol{\Psi}_f$ into (6.38), can be casted as the following equivalent problem:

$$\min_{\boldsymbol{\lambda}_f} \{f(\boldsymbol{\lambda}_f) := \frac{1}{2}\|\boldsymbol{\Pi}\boldsymbol{\Psi}_f(\boldsymbol{\lambda}_f)\|_F^2\}, \tag{6.39}$$

with $\mathbf{\Pi} := \mathbf{I}_N - \mathbf{U}\mathbf{U}^\dagger$ the orthogonal projection matrix onto the orthogonal complement of $\mathbf{U}$. In other words, problem (6.39) aims to find a vector $\boldsymbol{\lambda}_f$ such that the Vandermonde matrix $\mathbf{\Psi}_f(\boldsymbol{\lambda}_f)$ is orthogonal to the subspace spanned by the orthogonal complement of $\mathbf{U}$. Notice that we require $L \geq K$, so that the matrix $\mathbf{P}$ can have rank K revealing the subspace of the Vandermonde matrix $\mathbf{\Psi}_f(\boldsymbol{\lambda}_f)$ we want to estimate.

Problem (6.38) is not convex in $\boldsymbol{\lambda}_f$ due to the polynomial degree K (unless $K = 2$, i.e., the model in (6.36) is linear in $\boldsymbol{\lambda}_f$). To tackle the non-convexity of the problem, we resort to sequential convex programming (SCP) [14], a local optimization method that leverages convex optimization, where the non-convex portion of the problem is modeled by convex functions that are (at least locally) accurate. As in any non-convex problem, the initial starting point plays a big role; thus, if no prior information on the variable is given, a multi-starting point approach is advisable. The SCP formulation can be found in [2] and in the Supplemental Material (and potentially in the final version of this manuscript).

Remark 5. *The nature of the problem and the formulation* (6.39) *share similarities with the MUSIC algorithm [25]. There, however, the problem considers $\mathbf{\Psi}_f^\top$ instead of $\mathbf{\Psi}_f$ and N independent 1-dimensional searches can be carried out to find $\boldsymbol{\lambda}_f$ (which would be contained in the second row of $\mathbf{\Psi}^\top$). In our case, each column contains all the N variables and an N-dimensional search is needed, rendering a "scanning" of the vector variable $\boldsymbol{\lambda}_f$ infeasible, unless N is very small.*

6.4.4. Ambiguities

Notice that (6.36) is not free of model ambiguities, since different pairs $(\boldsymbol{\lambda}_f, \mathbf{C})$ may lead to the same observation matrix $\mathbf{P}$. Because we assume that $\mathbf{P}$ has rank K, if $\mathbf{\Psi}_f$ and $\mathbf{C}$ are the true matrix factors satisfying (6.36), then for any $K \times K$ invertible matrix $\mathbf{T}$, it holds:

$$\mathbf{P} = \mathbf{\Psi}_f \mathbf{T}\mathbf{T}^{-1}\mathbf{C} = \mathbf{\Psi}'_f \mathbf{C}'. \tag{6.40}$$

However, $\mathbf{\Psi}'_f$ needs to be Vandermonde in order for (6.40) to respect the model structure in (6.36). As we proved in [2, Theorem 2], matrix $\mathbf{T}$ needs in general to be a Pascal matrix. The consequence of such theorem is that without any added constraint on $\mathbf{\Psi}_f$ or $\mathbf{C}$ in (6.36), every shifted and scaled version of the groundtruth parameter $\boldsymbol{\lambda}_f$ (for an appropriate $\mathbf{C}$), perfectly fits the observation model and is a valid solution for (6.39). This is satisfactory for our purpose: a shift and scale of the graph eigenvalues maintains the same topological structure of the original graph (removing the self loops caused by the shift); we will illustrate this in Section 6.5.

6.5. Numerical Experiments

In this section we perform numerical experiments relying on the theory outlined in Section 6.4. Specifically, we first test our learning algorithm on synthetic data to assess the validity of the approach and its drawbacks; finally, we use it on real data.

6.5.1. SYNTHETIC DATA

The goal of this section is to validate the approach outlined in Section 6.4, by assuming the knowledge of the dual eigenvalues $\boldsymbol{\lambda}_f$, which however is not used during the training phase but only used to compute the performance metrics. We have the following *generation* and *learning* phases (also depicted in the Supplemental Material for visual clarity):

1) Generation.

(a) **Dual eigenvalues $\boldsymbol{\lambda}_f$**: in order to have control over the spacing between different eigenvalues, we generate them with the following strategy. First, we uniformly sample the eigenvalue domain leading to the grid $\mathbf{u} := [u_0, u_1, \ldots, u_{N-1}]^\top$, where the distance among two samples $P := u_n - u_{n-1}$ is constant for all n. Then, the actual eigenvalues $\boldsymbol{\lambda}_f$ are generated as $\lambda_{f,n} = u_n + j_n$ where $j_n \sim \mathcal{TN}(0, (\delta P/2)^2)$, with $\mathcal{TN}(0, \sigma_j^2)$ a Gaussian distribution with zero mean and standard deviation σ_j, yet truncated at σ_j. The (jitter) parameter $\delta > 0$ specifies the randomness of the eigenvalues. If $\delta < 1$ the eigenvalues maintain the same ranking order as the uniform samples, while if $\delta \geq 1$ they can overlap to the adjacent ones. Notice that in this synthetic setup we are not interested in creating a "meaningful" dual graph, but rather assessing whether our algorithmic routine is able to identify such dual graph.

(b) **Filter taps P**: we generate the primal node-varying filter taps $\mathbf{P} \in \mathbb{R}^{N \times L}$ as $\mathbf{P} = \boldsymbol{\Psi}_f \mathbf{C}$, with $\boldsymbol{\Psi}_f$ the Vandermonde matrix associated to $\boldsymbol{\lambda}_f$ generated in step (a), and $\mathbf{C} \in \mathbb{C}^{K \times L}$ a random expansion coefficient matrix [cf. (6.9)] generated as $\text{vec}(\mathbf{C}) \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{KL})$. To increase the curvature of the considered polynomials (thus avoiding to have almost flat curves in the domain of interest), we perform an extra weighting scheme by increasing the weight of higher order monomials; that is, we multiply the matrix $\mathbf{C}$ with a mask matrix as $\mathbf{C} \leftarrow [\mathbf{1}, 2\mathbf{1}, \ldots, K\mathbf{1}]^\top \odot \mathbf{C}$, where the ith column of the mask matrix is the constant vector containing in all its entries the value i.

(c) **Input-Output Data X, Y**: we generate T input graph signals $\mathbf{x}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_N)$ and stack them in the matrix $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_T]$. Then, we filter $\mathbf{X}$ with the type-I NV-GF $\mathbf{H}_I(\mathbf{P}, \mathbf{S})$ to obtain T new (possibly noisy) graph signals $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_T] = \mathbf{H}_I(\mathbf{P}, \mathbf{S})\mathbf{X} + [\mathbf{n}_1, \ldots, \mathbf{n}_T]$, with $\mathbf{n}_t \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_N)$ the measurement noise.

2) Learning. The goal in this phase is to recover the original $\boldsymbol{\lambda}_f$ of step 1(a) from the input-output data $\{\mathbf{X}, \mathbf{Y}\}$. This is achieved with the algorithmic routine introduced in Section 6.4.1 and Section 6.4.3; namely:

(a) **Filter Taps $\tilde{\mathbf{P}}$**: we find an estimate $\tilde{\mathbf{P}}$ of the filter taps $\mathbf{P}$ through (6.33);

(b) **Dual eigenvalues $\tilde{\boldsymbol{\lambda}}_f$**: we find an estimate $\tilde{\boldsymbol{\lambda}}_f$ of $\boldsymbol{\lambda}_f$ with the procedure outlined in Section 6.4.3, namely, casting the problem as a subspace fitting problem and solving it with SCP. During this step, we also estimate the coefficient matrix $\mathbf{C}$ of the expansion model leading to $\tilde{\mathbf{C}}$.

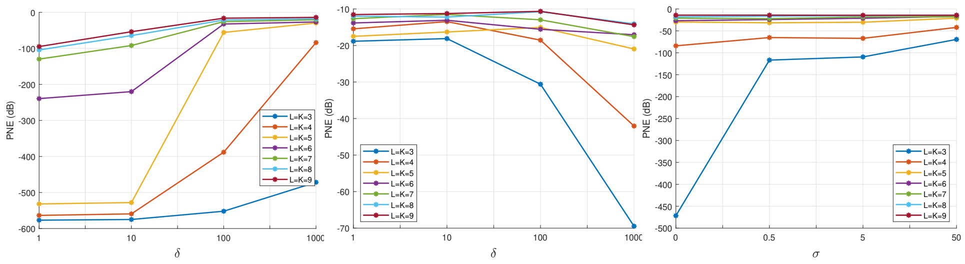


Figure 6.2: PNE (in dB) as a function of δ with $\sigma = 0$ (left) and $\sigma = 50$ (center), for different values of L, K; and PNE (in dB) as a function of σ for $\delta = 1000$ (right).

Metrics. To assess the validity of the proposed approach, we consider two different performance metrics, one relative to the estimation of the filter coefficients $\mathbf{P}$ and one relative to the estimation of the dual eigenvalues $\boldsymbol{\lambda}_f$. For the filter coefficients, we consider the normalized squared error (NSE), computed as:

$$\text{NSE}(\tilde{\mathbf{P}}, \mathbf{P}) = \frac{\|\tilde{\mathbf{P}} - \mathbf{P}\|_F^2}{\|\mathbf{P}\|_F^2}. \tag{6.41}$$

For the dual eigenvalues, recall that we can recover the solution of problem (6.39) up to a shift and scaling of the true positions [cf. Section 6.4.4]. Thus, as performance metric, we use the normalized error modulo Pascal (PNE), defined as:

$$\text{PNE}(\tilde{\boldsymbol{\lambda}}_f, \boldsymbol{\lambda}_f) = \min_{t_0, t_1} \frac{\|\boldsymbol{\lambda}_f - (t_0 \mathbf{1} + t_1 \tilde{\boldsymbol{\lambda}}_f)\|_2^2}{\|\boldsymbol{\lambda}_f\|_2^2} \tag{6.42}$$

which measures how far the true eigenvalues are from a linear transformation of the recovered estimates. Clearly (6.42) is zero whenever $\tilde{\boldsymbol{\lambda}}_f$ is a solution for (6.39).

Results. We generate our primal graph $\mathbf{S}$ as a random sensor network with $N = 40$ nodes, and set $u_0 = -1$ and $u_{N-1} = +1$ [cf. 1(a)]. We run the algorithm for different parameter configurations, specifically: the order of the filter in the primal domain $L \in \{2, \ldots, 9\}$, which is also the polynomial degree of the primal eigenvalues (cf. (6.7)); the order of the filter in the dual domain $K \in \{2, \ldots, 9\}$, with $K \leq L$, which is also the polynomial degree of the dual eigenvalues; the jitter parameter $\delta \in \{1, 10, 100, 1000\}$; and the noise standard deviation $\sigma \in \{0, 0.5, 5, 50\}$. The number of input (and output) graph signals is set to $T = 3000$. Due to the non-convexity of the cost function (6.39), we run the algorithm with 5 different starting points $\boldsymbol{\lambda}_f^0$, one of which is the uniform grid $\mathbf{u}$; this, together with the jitter parameter δ [cf. Generation (a)] helps us also understanding the "magnitude" of the objective function's non-convex landscape: if the objective function is highly non-convex, even an initial starting point $\boldsymbol{\lambda}_f^0$ close to the real $\boldsymbol{\lambda}_f$ (meaning a small δ) might incur a very high objective value and likely end-up in a local minimum. In such a case, a random starting point might be beneficial. The magnitude of non convexity of the objective function increases by increasing K. Finally, we compute the performance metrics relative to the solution associated to the different starting points.

In Fig. 6.2 we show the PNE (in dB) as a function of δ for different $L = K$[3], in the noiseless case (left figure) and noisy case (middle figure); in addition, we show (right) how the PNE varies as a function of the noise σ for fixed $\delta = 1000$. We can make the following observations: as expected, for low-degree polynomials, the algorithm performs better, since the non-convexity of the problem increases with increasing polynomial order. This is visible from the left and middle figure: for a small perturbation δ, also high orders yield a good performance; which degrades by increasing δ. However, when noise is present in the observations, a random initial starting points (i.e. higher δ) seems to be beneficial. Moreover, noise in the measurement (right figure) has obviously a negative impact in the learning performance, which however enables us to reconstruct the graph as we can see next.

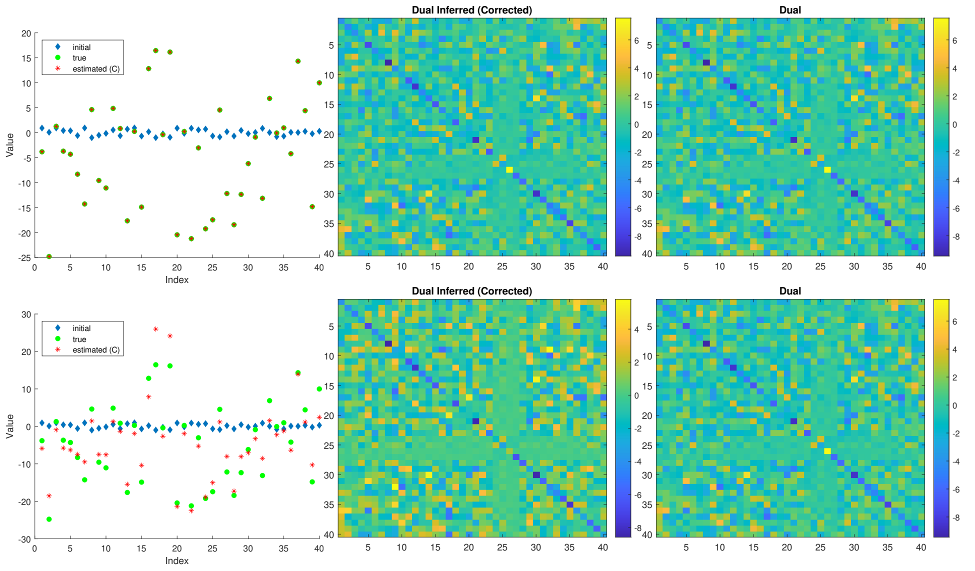


Figure 6.3: (Noiseless case $\sigma = 0$) Results for $L = K = 3$ (top row) and $L = K = 9$ (bottom row) with $\delta = 1 \times 10^4$. (Left Column) True eigenvalues (green circles), inferred eigenvalues (red crosses) and initial starting point of the algorithm (blue diamonds); (Center Column) Inferred dual GSO with ambiguity-correction; (Right Column) True dual GSO $\mathbf{S}_f$.

To visually assess the algorithm's performance, in Fig. 6.3 we show, for the noiseless case, the ambiguity-corrected estimated eigenvalues $\tilde{\boldsymbol{\lambda}}_f^c$ (red crosses), together with the original eigenvalues $\boldsymbol{\lambda}_f$ (green circles) and the initial starting point of the algorithm $\boldsymbol{\lambda}_f^0$ (blue diamonds) for $L = K = 3$ (top row) and $L = K = 9$ (bottom row),

[3]For all the plots we set $L = K$ for visualization clarity. To reproduce the experiments with different parameter settings, we make available the source code: https://github.com/albertonat/genConv

both with $\delta = 1 \times 10^4$, which corresponds to a completely random configuration of the eigenvalues. The ambiguity correction is explicitly performed as:

$$\tilde{\boldsymbol{\lambda}}_f^c = [\mathbf{1}\ \tilde{\boldsymbol{\lambda}}_f][\mathbf{1}\ \tilde{\boldsymbol{\lambda}}_f]^{\dagger}\boldsymbol{\lambda}_f, \tag{6.43}$$

which is the closest point to $\boldsymbol{\lambda}_f$ up to a linear transformation dictated by the optimal t_0 and t_1 minimizing (6.42).

In the $L = K = 3$ case, the NSE is $\text{NSE}(\tilde{\mathbf{P}}, \mathbf{P}) = 2.41 \times 10^{-29}$ and the PNE is $\text{PNE}(\tilde{\boldsymbol{\lambda}}_f, \boldsymbol{\lambda}_f) = 3.34 \times 10^{-25}$; in the $L = K = 9$ case the NSE is $\text{NSE}(\tilde{\mathbf{P}}, \mathbf{P}) = 9.20 \times 10^{-23}$ and the PNE is $\text{PNE}(\tilde{\boldsymbol{\lambda}}_f, \boldsymbol{\lambda}_f) = 1.03 \times 10^{-1}$. In both cases the inferred dual GSO, shown in the middle column of Fig. 6.3, correctly reveals the structure of the true dual GSO $\mathbf{S}_f$, shown in the right column; this even though the eigenvalue reconstruction (left column) is perfect only in the first case. In other words, even with an inexact (but not random) reconstruction of the eigenvalues, the algorithm seems to adequately capture the connections present in the true GSO $\mathbf{S}_f$. The reason behind the difference in error estimation among the two cases is mainly due to the high polynomial degree K, which on the analytic side renders the objective function (6.39) highly non-convex (and hence easier for the algorithm to end up in a local minimum); on the algebraic side it increases the numerical instability of performing the pseudoinverse of the matrix $\mathbf{A}$ required for a correct estimation of the filter parameter matrix $\mathbf{P}$. Hence, even a perfect dual graph frequency estimation step fitting perfectly the estimated $\mathbf{P}$, might fail to perfectly reconstruct the true eigenvalues $\boldsymbol{\lambda}_f$. For $\delta = 10$, the algorithm improves the PNE by two orders of magnitude and the inferred eigenvalues $\tilde{\boldsymbol{\lambda}}_f$ nearly overlap the true ones $\boldsymbol{\lambda}_f$.

For the noisy scenario, in Fig. 6.4 we show the results obtained by considering the two cases described above (i.e. $L = K = 3$ and $L = K = 9$ with $\delta = 1000$), but with a measurement noise having $\sigma = 50$. In this case the NSE is 8.88×10^{-6} and the PNE is 8.72×10^{-5}, while for the latter the NSE is 9.74×10^{-16} and the PNE is 2.00×10^{-1}. The eigenvalues and graph reconstruction is successful despite the fact that the reconstruction of the filter taps is not perfect as in the noiseless case, which gives us hope for the robustness of the algorithm when measuring noisy data.

Overall we can state that from an algorithmic point of view, the algorithm is robust in presence of a considerable eigenvalue perturbation δ and noise σ, especially for low polynomial degree. In instances where the polynomial degree is high, the inherent non-convex nature of the problem introduces substantial complexity into the optimization process, which however leads to a dual graph resembling the original one. the In addition, we observe that when the NSE is high, meaning that $\mathbf{P}$ has not been properly reconstructed, the PNE is also usually high, which is somehow expected since we rely on $\mathbf{P}$ to estimate the dual eigenvalues $\boldsymbol{\lambda}_f$.

6.5.2. Real Data

In this section we exploit the theory developed in Section 6.3 and Section 6.4 to infer a dual graph from real data. To evaluate the stationarity level of the given data $\mathbf{Y}$ for a given GSO $\mathbf{S}$, we use the proxy-measure $\rho = \|\text{diag}(\mathbf{V}^{-1}\mathbf{C}_y\mathbf{V})\|_2^2 / \|\mathbf{V}^{-1}\mathbf{C}_y\mathbf{V}\|_F^2$, which

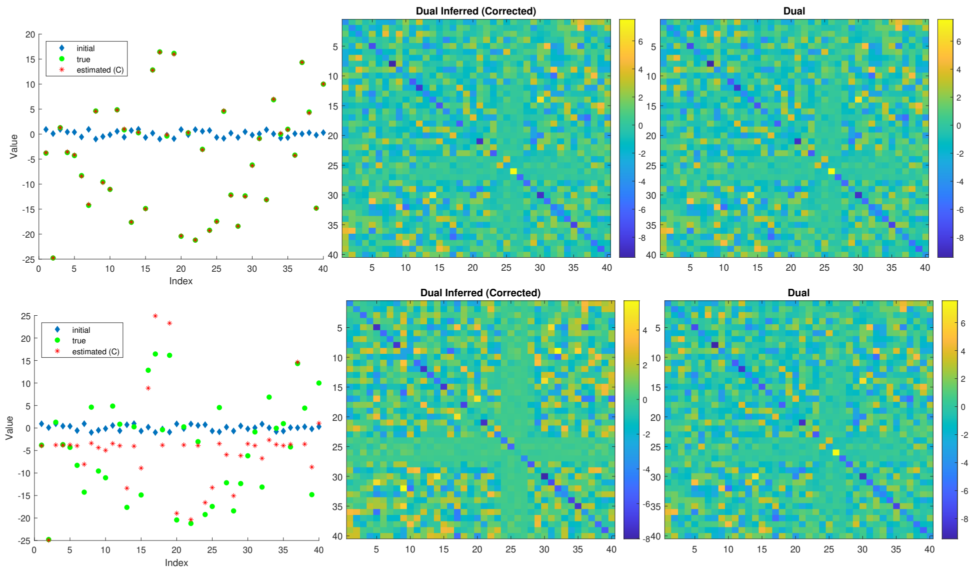


Figure 6.4: (Noisy case $\sigma = 50$) Results for $L = K = 3$ (top row) and $L = K = 9$ (bottom row) with $\delta = 1 \times 10^3$. (Left Column) True eigenvalues (green circles), inferred eigenvalues (red crosses) and initial starting point of the algorithm (blue diamonds; (Center Column) Inferred dual GSO with ambiguity-correction; (Right Column) True dual GSO $\mathbf{S}_f$.

measures the "diagonal dominance" of the spectral covariance matrix; a value of 1 indicates that the data are stationary.

As performance metrics we monitor two errors: *i)* the $\mathrm{NSE}(\mathbf{Y}, \mathbf{H}_I(\tilde{\mathbf{P}}, \mathbf{S})\mathbf{X})$ for the input-output case [cf. 6.4.1] or the $\mathrm{NSE}(\mathbf{R}, \mathbf{H}_I(\tilde{\mathbf{P}}, \mathbf{S})\tilde{\mathbf{U}})$ for the output-only case [cf. 6.4.2] (recall the definition of the NSE in (6.41)); and *ii)* the "corollary error" [cf. Corollary 1 in (6.10)]:

$$\varepsilon_c = \frac{\|\mathbf{V}^{-1}\mathbf{H}_I(\tilde{\mathbf{P}}, \mathbf{S}) - \mathbf{H}_{II}(\tilde{\hat{\mathbf{P}}}, \mathbf{S}_f)\mathbf{V}^{-1}\|_F^2}{\|\mathbf{V}^{-1}\mathbf{H}_I(\tilde{\mathbf{P}}, \mathbf{S})\|_F^2} \tag{6.44}$$

which assesses whether the inferred $\mathbf{S}_f$ is a "valid" dual graph consistent with our theorem, i.e., how much the upper and lower branches of Fig. 6.1 diverge from each other. To make things more clear, the $\tilde{\mathbf{P}}$ refers to the estimation of the primal filter tap matrix $\mathbf{P}$ either following (6.33) or (36a), while $\tilde{\hat{\mathbf{P}}}$ refers to the estimation of the dual filter tap matrix $\hat{\mathbf{P}}$, which is here computed as $\tilde{\hat{\mathbf{P}}} = \mathbf{\Psi}^{\dagger}\tilde{\mathbf{C}}^{\top}$, with $\tilde{\mathbf{C}} = \mathbf{\Psi}_f(\tilde{\boldsymbol{\lambda}}_f)^{\dagger}\tilde{\mathbf{P}}$, where $\tilde{\boldsymbol{\lambda}}_f$ is the estimation of the dual eigenvalues $\boldsymbol{\lambda}_f$ solving (6.39). All in all, error *i)* concerns the graph filter estimation, while *ii)* concerns the dual graph estimation.

Traffic Volume

we consider a subsampled version of the open dataset in [26] which contains $T = 1259$ traffic volume measurements at intervals of 15 minutes at $N = 13$ sensor locations along two major highways in Northern Virginia/Washington, D.C.; in addition, the physical (road) network is available, see Fig. 6.5(left). We denote with $\mathbf{S}$ the adjacency matrix representing the given road network, and with $\mathbf{Y} \in \mathbb{R}^{N\times T}$ the (centered) graph signals corresponding to the traffic volume measurements. These signals exhibit a non-stationary behavior captured by $\rho = 0.54$.

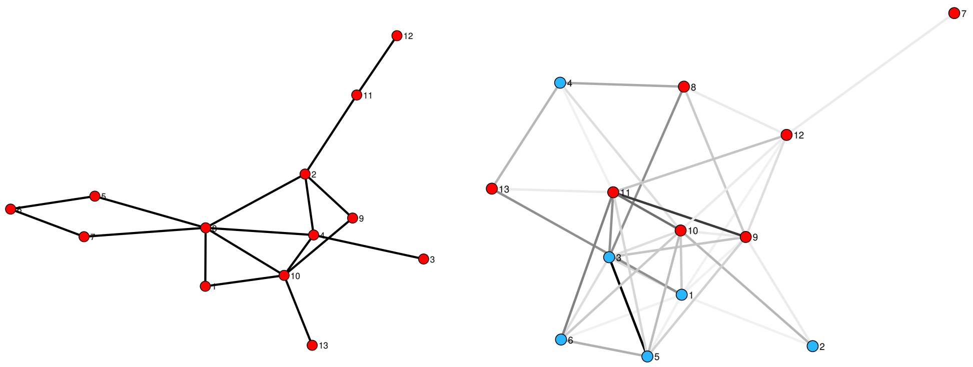


Figure 6.5: (Left): primal graph of the road network; (Right) dual graph of the road network, where each node represents a graph frequency. Red (blue) nodes are the standard low (high) frequencies.

In our experiment, we explore both input-output and output-only scenarios. In the former scenario, we define the input matrix $\mathbf{X}$ by aligning it with $\mathbf{Y}$, shifting each column two positions to the left; consequently, the learning task revolves around forecasting the traffic volume 30 minutes ahead. We initialize the parameters with $L = K = 3$ and execute the algorithm. Notice that our data was deliberately not partitioned into training, validation, and test sets, since our primary focus is optimizing the fitting of our graph filter to the provided data, rather than evaluating its forecasting performance.

First, we learn the filter coefficients $\mathbf{P}$ for both scenarios, yielding a NSE equal to 8×10^{-2} for both. Subsequently, we use the inferred $\tilde{\mathbf{P}}$ to learn the dual eigenvalues $\boldsymbol{\lambda}_f$ by solving (6.39). The associated dual GSO $\mathbf{S}_f = \mathbf{V}^{-1}\mathrm{Diag}(\tilde{\boldsymbol{\lambda}}_f)\mathbf{V}$, achieving a corollary error [cf. (6.44)] $\varepsilon_c = 1 \times 10^{-1}$ for scenario 1 and $\varepsilon_c = 3 \times 10^{-1}$ for scenario 2, is shown in Fig. 6.5(right), where we only display the 50% biggest edges (in absolute values) to ease the visualization. We color with blue the first half of the nodes, representing what are usually considered "high pass" frequencies, and with red the second half of the nodes, representing what are usually considered the "low pass" frequencies (remember that $\mathbf{S}$ is the adjacency matrix). Adjacent nodes in the graph are not necessarily among consecutive graph frequencies, as it is commonly assumed in GSP. This shows that the frequency ordering that is commonly assumed does not fit our theory and we might obtain a more expressive way to embed the different graph frequencies.

MNIST Dataset

We consider the MNIST dataset of gray-scale handwritten digits from 0 to 9, focusing on the digit 5[4] containing $T = 5949$ images of size 18×18. We model each pixel of the image as a node of the primal graph $\mathbf{S}$ and its pixel intensity as the graph signal value at that node. As a preprocessing step, we remove the mean from each pixel and vectorize the images, thus obtaining a graph signal matrix $\mathbf{Y} \in \mathbb{R}^{N \times T}$, with $N = 324$. As GSO $\mathbf{S}$, we consider the normalized Laplacian matrix of the 18×18 grid graph, for which the 5 exhibits a non-stationary behavior, since $\rho = 0.4$.

We assume that each $\mathbf{y}_t$ is the output of a NV-GF $\mathbf{H}_I(\mathbf{P}, \mathbf{S})$ when excited with a white input $\mathbf{x}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$; our goal is then to learn the filter taps $\mathbf{P}$ and subsequently the dual eigenvalues $\boldsymbol{\lambda}_f$ from the available data $\mathbf{Y}$. In this particular scenario, it is important to highlight that our access is limited to the output data $\mathbf{Y}$, since the corresponding noise input $\mathbf{X}$ is not available. Consequently, the sole viable approach becomes the output-only procedure [cf. Section 6.4.2]. Nevertheless, to navigate this limitation and to be also able to use the input-output approach of Section 6.4.1, we can employ the ensuing rationale to derive pairs $(\mathbf{x}_t, \mathbf{y}_t)$:

(a) compute the covariance matrix $\mathbf{C}_y = \mathbf{Y}\mathbf{Y}^\top / T$ and decompose it as $\mathbf{C}_y = \mathbf{R}\mathbf{R}^\top$;

(b) filter a white input signal $\mathbf{x}_t$ to generate a new signal $\mathbf{y}_t$ as $\mathbf{y}_t = \mathbf{R}\mathbf{x}_t$.

It follows that the $\mathbf{y}_t$ vector generated in this way follow the same distribution of the original $\mathbf{Y}$ and still represent the digit 5 (up to a sign ambiguity). This time, however, we have the associated input $\mathbf{X}$. We run the proposed algorithm for different orders L and K, with and without input $\mathbf{X}$.

In Fig. 6.6(left) we show the inferred filter taps $\tilde{\mathbf{P}}$, for $L = 4$, corresponding to the solution of problem (6.35) and yielding a $\text{NSE} = 3 \times 10^{-2}$; each filter tap $\{\mathbf{p}_l\}_{l=0}^{3}$ has been reshaped to have the same size of the input image and stacked in a row-wise fashion. It is interesting to notice how each $\mathbf{p}_l$ has a digit-shape look, with a decreasing pixel intensity for increasing filter tap order L; this indicates pixel-locality as an important factor for the creation of the final pixel intensity. Since each (reshaped) filter tap image in Fig 6.6 pointwise-multiplies a shifted white input (noise) image of same size (to be finally aggregated), it is visible how the most influential filter content gathers around the digit shape for the digit-formation, and decreases its importance for higher shifts, meaning that the process is local on the pixel and there are no long term-influences. The same NSE and $\mathbf{P}$-profile is obtained with the input-output approach.

Once we have the filter tap matrix $\tilde{\mathbf{P}}$, we then learn the dual eigenvalues $\boldsymbol{\lambda}_f$ by solving (6.39). The associated dual GSO $\mathbf{S}_f = \mathbf{V}^{-1}\text{Diag}(\tilde{\boldsymbol{\lambda}}_f)\mathbf{V}$, achieving a corollary error [cf. (6.44)] $\varepsilon_c = 9 \times 10^{-2}$, is shown in Fig. 6.6(right), where we only display 2% of the most significant edges in absolute values[5]. The node label i indicates the index

[4]Similar results can be obtained with the other digits, see https://github.com/albertonat/genConv

[5]For a graph with $N = 324$ nodes there would be more than 50k edges possible.

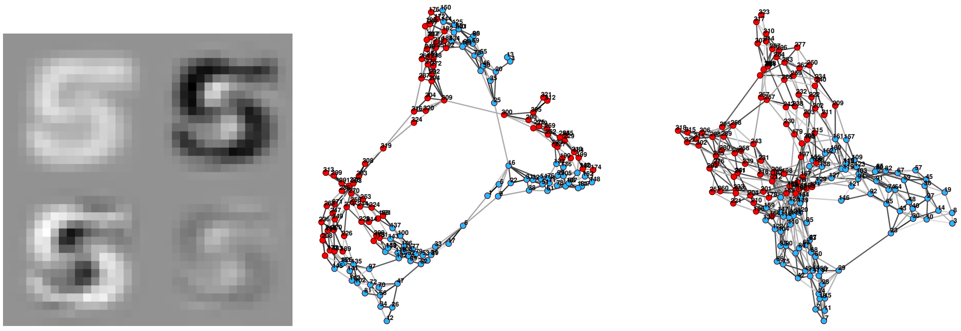


Figure 6.6: (Left) Illustration of each filter tap $\{\mathbf{p}_l\}_{l=0}^{3}$, following a row-wise stacking, reshaped to have the same shape of the input image. Increasing l renders the entries of $\mathbf{p}_l$ less influential for the construction of the final digit; (Right) Dual Graph of the 5-digit. The label indicates the index of the eigenvalues λ_i, and the node color indicates whether it is part of the first half (blue) or the second half (red), commonly associated to the low and high pass bands, respectively. Notice that the graph is connected but due to the edge thresholding (only for the visualization) it is split in two.

associated to λ_i. As in the previous experiment, we color the first half of the nodes (representing now the "low frequency" eigenvalues) with blue, and the other half with red. Similar results and conclusions can be made by following the input-output approach, where we obtain $\varepsilon_c = 3 \times 10^{-2}$ and the graph is similarly structured as the one in Fig. 6.6 (see Supplementary Material).

Together with the previous experiment, the following observations can be made:

- The connections between the eigenvalues do not follow the linear ordering as assumed by the traditional real-line interpretation (in that case, we would only have red-red and blue-blue connections without interactions); this has consequences when designing graph filters based solely on the value of the λ_i, since now the concept of "bands" needs also to account for the topology.

- Because the dual graph represents the support of the GFT signals, we can now inspect which neighborhood is influential for a particular frequency during a convolution on $\mathbf{S}_f$; this was not possible with the standard real-line interpretation, as the convolution operation was a simple pointwise multiplication.

All in all, these results confirm the commutative nature of the two branches of Fig. 6.1, thus rendering the dual convolution a preferred approach when $K < L$, or when $\mathbf{S}_f$ and/or the GFT signals exhibit sparsity, in addition to delivering an elegant theoretical framework.

6.6. CONCLUSIONS

In this work we proposed a convolution theorem which extends the classical convolution theorem in (graph) signal processing and the one related to time-varying filters. More precisely, we illustrate how a convolution in the primal graph domain can be redefined as a distinct convolution in the dual graph (frequency) domain, given a suitable filter parametrization. After illustrating the implications of such theorem in terms of non-stationarity of signals, and generative models thereof, we devise an algorithmic approach based on subspace fitting and non-convex programming techniques to learn the dual graph from data when this is not a priori known. We evaluated the proposed theory and algorithms on synthetic data, as well as on real data.

While our current theoretical framework holds promise for practical applications in the future, there are notable challenges that merit further exploration. A significant gap lies in the absence of a one-shot procedure to construct the dual GSO directly from a primal GSO, along with potential graph signals associated with it. This limitation curtails the broader applicability of the proposed theoretical insights. Nonetheless, multiple extensions of this work are possible. From an algebraic point of view, an interesting line of research would involve exploring the connections between Vandermonde and Hankel matrices, as well as with Krylov subspaces, potentially unveiling new algorithmic solutions to learn the dual eigenvalues. From a modeling point of view, an interesting extension of this work would include the node-varying graph filter coefficients also to be time-varying; holding promise for utilization in graph autoregressive models. In such cases, leveraging the basis expansion model technique across the temporal dimension becomes a potential avenue for further investigation. From an optimization point of view, the use of orthogonal polynomials might alleviate the ill-conditioning of the Vandermonde matrix.

Our hope is that in the coming years, further exploration and refinement of this research direction will reveal new insights and methodologies to process signals defined on graphs in a way previously unfeasible.

REFERENCES

[1] A. Natali and G. Leus. “A general convolution theorem for graph data”. In: *2022 56th Asilomar Conference on Signals, Systems, and Computers*. IEEE. 2022, pp. 48–52.

[2] A. Natali and G. Leus. “Blind polynomial regression”. In: *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2023, pp. 1–5.

[3] A. Natali and G. Leus. “A Generalization of the Convolution Theorem and its Connections to Non-Stationarity and the Graph Frequency Domain”. In: *IEEE Transactions on Signal Processing* 74 (2024), pp. 3424–3437.

[4] A. V. Oppenheim, J. R. Buck, and R. W. Schafer. *Discrete-time signal processing. Vol. 2.* Upper Saddle River, NJ: Prentice Hall, 2001.

[5] H. Lütkepohl. *Introduction to multiple time series analysis*. Springer Science & Business Media, 2013.

[6] Y. LeCun, Y. Bengio, and G. Hinton. “Deep learning”. In: *nature* 521.7553 (2015), pp. 436–444.

[7] Y. LeCun, Y. Bengio, *et al.* “Convolutional networks for images, speech, and time series”. In: *The handbook of brain theory and neural networks* 3361.10 (1995), p. 1995.

[8] 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”. In: *IEEE signal processing magazine* 30.3 (2013), pp. 83–98.

[9] S. Segarra, A. G. Marques, and A. Ribeiro. “Optimal graph-filter design and applications to distributed linear network operators”. In: *IEEE Transactions on Signal Processing* 65.15 (2017), pp. 4117–4131.

[10] D. I. Shuman, P. Vandergheynst, D. Kressner, and P. Frossard. “Distributed signal processing via Chebyshev polynomial approximation”. In: *IEEE Transactions on Signal and Information Processing over Networks* 4.4 (2018), pp. 736–751.

[11] G. Leus, S. Segarra, A. Ribeiro, and A. G. Marques. “The dual graph shift operator: Identifying the support of the frequency domain”. In: *Journal of Fourier Analysis and Applications* 27.3 (2021), p. 49.

[12] J. Shi and J. M. Moura. “Graph signal processing: Dualizing gsp sampling in the vertex and spectral domains”. In: *IEEE Transactions on Signal Processing* 70 (2022), pp. 2883–2898.

[13] M. Viberg and B. Ottersten. “Sensor array processing based on subspace fitting”. In: *IEEE Transactions on signal processing* 39.5 (1991), pp. 1110–1121.

[14] S. Boyd. “Sequential convex programming”. In: *Lecture Notes, Stanford University* (2008).

[15] N. Saito. “How can we naturally order and organize graph Laplacian eigenvectors?” In: *2018 IEEE Statistical Signal Processing Workshop (SSP)*. IEEE. 2018, pp. 483–487.

[16] A. Cloninger, H. Li, and N. Saito. “Natural graph wavelet packet dictionaries”. In: *Journal of Fourier Analysis and Applications* 27.3 (2021), pp. 1–33.

[17] A. G. Marques, S. Segarra, G. Leus, and A. Ribeiro. “Stationary graph processes and spectral estimation”. In: *IEEE Transactions on Signal Processing* 65.22 (2017), pp. 5911–5926.

[18] N. Perraudin and P. Vandergheynst. “Stationary signal processing on graphs”. In: *IEEE Transactions on Signal Processing* 65.13 (2017), pp. 3462–3477.

[19] R. Shafipour, S. Segarra, A. G. Marques, and G. Mateos. “Identifying the topology of undirected networks from diffused non-stationary graph signals”. In: *IEEE Open Journal of Signal Processing* 2 (2021), pp. 171–189.

[20] A. Sandryhaila and J. M. Moura. “Discrete signal processing on graphs”. In: *IEEE transactions on signal processing* 61.7 (2013), pp. 1644–1656.

[21] G. B. Giannakis and C. Tepedelenlioglu. “Basis expansion models and diversity techniques for blind identification and equalization of time-varying channels”. In: *Proceedings of the IEEE* 86.10 (1998), pp. 1969–1986.

[22] Z. Tang, R. C. Cannizzaro, G. Leus, and P. Banelli. “Pilot-assisted time-varying channel estimation for OFDM systems”. In: *IEEE Transactions on Signal Processing* 55.5 (2007), pp. 2226–2238.

[23] A. Natali, M. Coutino, and G. Leus. “Topology-aware joint graph filter and edge weight identification for network processes”. In: *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE. 2020, pp. 1–6.

[24] B. F. Green. “The orthogonal approximation of an oblique structure in factor analysis”. In: *Psychometrika* 17.4 (1952), pp. 429–440.

[25] R. Schmidt. “Multiple emitter location and signal parameter estimation”. In: *IEEE transactions on antennas and propagation* 34.3 (1986), pp. 276–280.

[26] L. Zhao, O. Gkountouna, and D. Pfoser. “Spatial auto-regressive dependency interpretable learning based on spatial topological constraints”. In: *ACM Transactions on Spatial Algorithms and Systems (TSAS)* 5.3 (2019), pp. 1–28.

7 Conclusion

In this concluding chapter, we summarize our contributions in the field of (graph) signal processing (GSP) and their potential impact in societal applications. We then outline research avenues which we consider have been under explored while holding a huge potential for the field.

7.1. Summary of the Contributions

In this manuscript, our focus has been on addressing two fundamental concepts in (graph) signal processing: graph topology identification and the study of shift-variant systems. Let us here briefly lay down the contributions of each chapter.

In Chapter 2, we delved into the fundamentals of GSP, beginning with the foundational principles. Specifically, we started by defining the concept of shift and convolution in the (discrete-)time domain. We then explored how we could capture the time domain (or more formally, its shifting operation) by a graph with a graph shift operator (GSO) $\mathbf{S}$ coinciding with the circular shift $\mathbf{S}_c$ [cf. (2.4)]. Building upon this insight, we found that (almost) all classical signal processing concepts, such as filters and spectral analysis, can be easily extended in a graph setting as long as the GSO $\mathbf{S}$ is diagonalizable, leading to new definitions of frequency, transforms, and shift-invariant filters. In the last part of the chapter, we introduced the problem of network topology identification from data and illustrated it through three examples: the Gaussian graphical model, the structural equation model and the smoothness based model.

Having delved into the world of graph filters and their capability to capture network dynamics like diffusive phenomena, we have come across a common type known as shift-invariant filters. These filters, prevalent in the literature, possess a unique property—they commute with the GSO $\mathbf{S}$. Now, traditionally, these filters are parametric in two variables: the GSO $\mathbf{S}$ and the filter coefficients $\mathbf{p}$. While much of the previous research has concentrated on learning the filter coefficients for a fixed GSO $\mathbf{S}$, in Chapter 3, we asked ourselves the following question::

RQ-1

"Given the availability of input-output networked data and partial information about the graph, is it possible to jointly learn the optimal filter coefficients $\mathbf{p}$ *and the GSO* $\mathbf{S}$ *that effectively capture the network dynamics?"*

We answered this question in Chapter 4. After formulating a constrained least-squares problem and recognizing its non-convexity, we proposed an algorithmic routine based on alternating minimization of $\mathbf{p}$ and $\mathbf{S}$. We tackled the non-convexity of the problem by building on sequential convex programming (SCP), a local optimization tool for non-convex problems that leverages the convex optimization machinery. We show that our approach guarantees that the objective function value at each iteration is non-increasing, obtaining a globally convergent method.

However, there are instances in which the graph's structure is entirely unknown, yet its understanding is vital. This could be either because the graph represents the primary object of interest or because it serves as an essential component of graph-based architectures, such as graph filters. Therefore, learning such structure, potentially from available data, often becomes a fundamental challenge. Additionally, networks frequently demonstrate temporal variability, with connections between entities evolving over time. Furthermore, the data we observe often arrive in an online streaming fashion, where storing them might be impractical. Consequently, in Chapter 3, we raised the question:

RQ-2

"How can we systematically learn time-varying graphs on-the-fly from online data?"

This question became the focal point of our exploration in Chapter 5. Initially, we established a premise: for meaningful exploration, it is imperative to impose constraints on the interplay between the graph topology and the observed data. Consequently, we provided a succinct overview of prevalent graph-data models and the static optimization problems that emerge when these models are assumed. Subsequently, we extended these algorithms into their time-evolving counterparts and integrated them within a coherent framework. This framework, agnostic to specific models, enabled analysis in its abstract form adaptable to various graph learning paradigms; we particularized it to the Gaussian graphical model, the structural equation model and the smoothness-based model, by showing a theoretical analysis and numerical experiments. Our approach harnessed innovative methodologies derived from time-varying convex optimization [1].

We then moved to another fundamental concept in (graph) signal processing, that of convolution. When the operator $\mathbf{H}$ implementing the convolution (a.k.a. the filter) commutes with the shift operator $\mathbf{S}$, implying their jointly diagonalizability, we say that the filter is *shift-invariant*. This property is often assumed in signal processing architectures due to its favourable spectral properties. However, many real-world systems are not shift-invariant. This topic has been researched for what concerns time-based systems, e.g. in underwater communications and mobility networks, but in the context of graphs it was missing. Thus, in Chapter 3 we asked the question:

RQ-3

> *"How can we extend the convolution theorem to encompass shift-variant systems operating on graphs, and is there a correlation with non-stationarity?"*

We addressed this question in Chapter 6, where we leveraged the concept of the dual graph $\mathbf{S}_f$ [2], which serves to depict the structure of the frequency domain. Through our exploration, we illustrated how a shift-variant system in one domain can be equated to another shift-variant system in the alternate domain. This novel theorem not only broadens the scope of the graph convolution theorem but also encompasses its counterpart for time-varying systems, which can be viewed as specific cases within this overarching theorem. Furthermore, we elucidated the interplay between shift-variant systems in both domains and their influence on the stationarity of graph signals within each domain.

The dual graph $\mathbf{S}_f$ remains somewhat enigmatic to date, typically uncovered through either an axiomatic or an optimization approach. However, with the introduction of the general convolution theorem, a connection emerged between the filter coefficients in the primary domain and the eigenvalues of the dual graph shift operator (GSO) $\mathbf{S}_f$. This revelation spurred yet another question:

RQ-4

> *"Is it possible to acquire knowledge of the dual graph $\mathbf{S}_f$ through a data-driven methodology exploiting the proposed convolution theorem?"*

To address this inquiry, once again in Chapter 6, we proposed a data-driven methodology aimed at learning the eigenvalues of the dual graph $\mathbf{S}_f$, given that they represent the sole unknowns when the primal graph shift operator (GSO) $\mathbf{S}$ is known. The resultant optimization problem leads to a structured matrix factorization, wherein one of the factors resembles a Vandermonde matrix, presenting similarities with a MUSIC-type approach, albeit more intricate to resolve. This problem was investigated in detail in [3] and solved with a subspace-based approach, revealing that despite encountering ambiguities related to scaling and translation, they do not significantly impede the process of graph learning.

7.2. OPEN RESEARCH AVENUES

In this concluding section, we highlight research paths that we consider crucial for the advancement of the signal processing on graphs community. While we acknowledge the existence of numerous potential minor research directions, we focus on avenues we consider of significant importance.

A fundamental research endeavor lies in comprehending the existence of an optimal GSO **S**. Currently, various GSOs yield entirely distinct spectral interpretations. It remains intriguing to investigate whether a universally optimal GSO exists, irrespective of specific applications. We foresee that theory on differential geometry and algebraic topology will be essential in addressing this inquiry. This would help in solving issues about the missing of bijectivity between the set of graphs and the set of matrix representations. We have seen indeed how in the context of the dual graph, the eigenvectors of the primal GSO **S** also constitute the eigenvectors of the dual **S**, and that having a Laplacian in one domain basically excludes having a Laplacian in the other domain. Thus formally:

ORQ-1

"Does there exist an optimal matrix representation to encode graph properties which is application-independent?"

Another important research avenue with great potential is the inference of time-varying directed graphs. Although research has been (and still is) carried out to process signals on directed graphs (see [4] for a tutorial on the subject), the time-varying topology inference problem has received little attention. Since many real world networks exhibit directionality in their associations, it is natural to ask whether a principled framework, similar to the one proposed in Chapter 5, can be offered for directed graphs. This prompts the research question:

ORQ-2

"Can we learn time-varying directed graphs in a principled way?"

In closing, we see enormous potential for a profound synergy between the signal processing and computer science communities. The practical, results-oriented mindset prevalent in computer science stands to gain immensely from the profound theoretical insights embedded within signal processing methodologies. Likewise, signal processing experts should occasionally break away from the confines of theoretical modeling assumptions and engage in practical coding. This approach allows for the scrutiny of theoretical assumptions in light of real-world scenarios, fostering a more robust and empirically grounded understanding of signal processing principles.

Bibliography

[1] A. Simonetto, E. Dall'Anese, S. Paternain, G. Leus, and G. B. Giannakis. "Time-Varying Convex Optimization: Time-Structured Algorithms and Applications". In: *Proceedings of the IEEE* (2020).

[2] G. Leus, S. Segarra, A. Ribeiro, and A. G. Marques. "The dual graph shift operator: Identifying the support of the frequency domain". In: *Journal of Fourier Analysis and Applications* 27.3 (2021), p. 49.

[3] A. Natali and G. Leus. "Blind polynomial regression". In: *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2023, pp. 1–5.

[4] A. G. Marques, S. Segarra, and G. Mateos. "Signal processing on directed graphs: The role of edge directionality when processing and learning from network data". In: *IEEE Signal Processing Magazine* 37.6 (2020), pp. 99–116.

CURRICULUM VITÆ

Alberto NATALI

07-02-1994	Born in Umbertide, Italy.

EDUCATION

2009-2013	Liceo Scientifico Tecnologico ITIS Leopoldo e Alice Franchetti, Città di Castello
2013–2016	B.Sc. in Computer Science & Electronic Engineering Università degli Studi di Perugia
2016-2019	M.Sc. in Computer Engineering & Robotics Università degli Studi di Perugia
2024	PhD. Electrical Engineering Delft University of Technology *Thesis:* Signal Processing and Optimization on Graphs: Learning Time-Varying Structures and Generalizing Convolution Principles *Promotor:* Prof. dr. ir. G.J.T. Leus