

Advances in graph signal processing
Graph filtering and network identification

Coutino, Mario

DOI

[10.4233/uuid:3654933b-8a8a-4a45-9a54-323e51641f5f](https://doi.org/10.4233/uuid:3654933b-8a8a-4a45-9a54-323e51641f5f)

Publication date

2021

Document Version

Final published version

Citation (APA)

Coutino, M. (2021). *Advances in graph signal processing: Graph filtering and network identification*.
<https://doi.org/10.4233/uuid:3654933b-8a8a-4a45-9a54-323e51641f5f>

Important note

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

Copyright

Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy

Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

ADVANCES IN GRAPH SIGNAL PROCESSING

GRAPH FILTERING AND NETWORK IDENTIFICATION

ADVANCES IN GRAPH SIGNAL PROCESSING
GRAPH FILTERING AND NETWORK IDENTIFICATION

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof. dr. ir. T.H.J.J van der Hagen,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op woensdag 21 april 2021 om 15:00 uur

door

Mario Alberto COUTIÑO MINGUEZ

Master of Science in Electrical Engineering,
Delft University of Technology, the Netherlands
geboren te La Paz, Baja California Sur, Mexico.

Dit proefschrift is goedgekeurd door de

promotor: prof. dr. G.J.T Leus

Samenstelling promotiecommissie:

Rector Magnificus,
Prof. dr. G.J.T Leus

voorzitter
Technische Universiteit Delft, promotor

Onafhankelijke leden:

Prof. dr. G.B. Giannakis
Prof. dr. ir. M.H.G. Verhaegen
Prof. dr. S. Barbarossa
Prof. dr. D.P. Palomar

University of Minnesota, United States
Technische Universiteit Delft
Sapienza Università di Roma, Italy
Hong Kong University of Science and Technology,
Hong Kong
Rice University, United States
Facebook, Inc., United Kingdom
Technische Universiteit Delft, reservelid

The work described in this thesis was financially supported by the ASPIRE project 14926 (within the STW OTP program) financed by the Netherlands Organization for Scientific Research (NWO). Mario Coutiño received financial support from CONACYT throughout his education.



Nederlandse Organisatie voor Wetenschappelijk Onderzoek



Keywords: distributed processing, graph filtering, graph theory, graph signal processing, topology identification

Front & Back: *The boy who saw a butterfly at the rim of the network pool.*
Art concept: Mario Coutiño. Design: Benoît Marcou.

Print: Ridderprint | www.ridderprint.nl

Copyright © 2021 by M. Coutino

ISBN 978-94-6416-560-9

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

*For them,
who as rivers,
even when flood,
nurture*
your son

*For them,
who though not here,
have never left*
your grandson

*For him,
who distant,
is always at my side*
your brother

CONTENTS

Preface	xi
Foreword	xiii
1 Introduction	1
1.1 Why Graph(s) [Signal Processing]?	2
1.2 Processing Data over Graphs	3
1.3 Summary of results and outline	4
1.3.1 Tackling the Network Data Deluge	5
1.3.2 Advances in Graph Filtering	5
1.3.3 Asynchronous Generalized Graph Filters	5
1.3.4 A Cascaded Structure of Graph Filters	5
1.3.5 State-Space based Network Topology Identification	6
1.3.6 Network Topology Identification from Partial Observations	6
1.4 List of contributions	7
1.4.1 Graph Signal Processing	7
1.4.2 Signal Processing	9
References	10
2 Tackling the Network Data Deluge	13
2.1 Graph Signal Processing	14
2.1.1 Representation of a graph	15
2.1.2 Signals over graphs.	16
2.1.3 Shifts on graphs	16
2.1.4 Frequency interpretation over graphs	17
2.2 Filtering in irregular domains	21
2.2.1 Graph Filters	22
2.2.2 Finite-impulse-response graph filters	22
2.2.3 Infinite-impulse-response graph filters	24
2.2.4 Graph Modal Response	24
2.3 Two challenges in networked data processing.	25
2.3.1 Enhanced distributed processing	25
2.3.2 Network topology identification from network processes	26
2.4 Chapter Summary	27
References	27

3	Advances in Graph Filtering	31
3.1	Introduction	31
3.1.1	Related Works	32
3.1.2	Chapter Contributions	33
3.1.3	Chapter Outline	34
3.2	Edge-variant graph filters	34
3.2.1	General form	34
3.2.2	Shift-invariant form	35
3.2.3	Filter Design	37
3.3	Constrained edge-variant graph filters	38
3.3.1	General Form	38
3.3.2	Shift-Invariant Constrained Edge-Variant Graph Filters	40
3.3.3	Filter Design	41
3.4	Edge-variant IIR graph filters	42
3.4.1	Edge-Variant ARMA ₁	42
3.4.2	Shift-Invariant EV ARMA ₁	43
3.4.3	Filter Design	43
3.5	Applications	44
3.5.1	Graph Filter Approximation	45
3.5.2	Distributed Linear Operator Approximation	46
3.5.3	Comparison with Distributed Optimization	50
3.5.4	Tikhonov-based denoising	51
3.6	Chapter Summary	52
	References	54
4	Asynchronous Generalized Graph Filters	57
4.1	Introduction	57
4.1.1	Chapter Contributions	58
4.1.2	Chapter Outline	58
4.2	Linear Operators as Graph Filters	59
4.3	Node Update Model	60
4.4	Asynchronous Updates	63
4.4.1	Classical Graph Filter: Asynchronous Case	64
4.5	Numerical Simulation	64
4.6	Chapter Summary	67
4.7	Appendix	67
4.7.1	Proof Lemma 1	67
4.7.2	Proof Lemma 2	68
4.7.3	Proof Lemma 3	68
	References	70
5	A Cascaded Structure of Graph Filters	71
5.1	Introduction	72
5.1.1	Chapter Contributions	73
5.1.2	Chapter Outline	74

5.2	Cascade Implementation of Graph Filters	74
5.2.1	Theoretical Study of Cascaded GF Error Surface	75
5.2.2	Relation to Graph Convolutional Networks	78
5.3	RELIEF Algorithm	79
5.3.1	linSparseSolve Routine	80
5.3.2	refitPair Routine	81
5.4	Data-driven RELIEF	81
5.4.1	Staggered RELIEF	83
5.4.2	Some Words About (Full) Backpropagation	84
5.5	Numerical Results	84
5.5.1	Error surfaces and root analysis	84
5.5.2	RELIEF Application: Distributed consensus	86
5.6	Chapter Summary	89
5.7	Appendix	89
5.7.1	Proof Lemma 4	89
5.7.2	Proof of Theorem 5.1	90
5.7.3	Proof of Theorem 5.2	90
5.7.4	LSMR Algorithm	93
	References	93
6	State-Space based Network Topology Identification	97
6.1	Introduction	97
6.1.1	Chapter contributions	98
6.1.2	Chapter Outline	99
6.2	The topology identification problem	99
6.3	First Order Differential Graph Model	100
6.4	State-Space Identification	101
6.4.1	Retrieving the state matrix A	102
6.4.2	Retrieving the input matrix B	104
6.4.3	Noisy setting	105
6.4.4	Continuous-time model identification	106
6.5	Identification of Network Connectivity	106
6.5.1	Known Scalar Mappings	107
6.5.2	Unknown Scalar Mappings	107
6.5.3	Single-Shot State Graph Estimation	107
6.6	Numerical Examples	108
6.6.1	Toy Synthetic Example	108
6.6.2	Discrete model validation	108
6.6.3	Network Topology Indentification: ETEX dataset	110
6.7	Chapter Summary	110
	References	112
7	Network Topology Identification from Partial Observations	115
7.1	Introduction	115
7.1.1	Chapter contributions	116
7.1.2	Chapter Outline	117

7.2	The Partially Observed Problem.	117
7.2.1	The graph inverse eigenvalue problem.	118
7.2.2	Ambiguous graphs: cospectral graphs	118
7.3	Graph construction by alternating projections	119
7.3.1	Alternating projections method	120
7.4	Inaccurate and partial eigendecomposition.	122
7.4.1	Uncertainty in the system matrices	122
7.4.2	Partial eigendecomposition	123
7.5	System consistency constraints	124
7.6	Numerical Results.	124
7.6.1	Network Topology Identification: Social graph.	124
7.6.2	Convergence of the alternating projections method	125
7.6.3	Partial observations	128
7.7	Chapter Summary	128
7.8	Appendix	129
7.8.1	Proof Theorem 7.2	129
7.8.2	Proof Theorem 7.3	129
7.8.3	Proof Proposition 7.1.	130
7.8.4	Proof Theorem 7.4	130
7.8.5	Enforcing Positive semidefiniteness under noise	130
7.8.6	Proof Theorem 7.5 (Sketch.)	131
7.8.7	Convergence of Alternating Projections	132
	References	132
8	Conclusion & Research Trends	135
8.1	Concluding remarks	135
8.2	Research Opportunities	137
8.2.1	Graph Filtering.	137
8.2.2	Network Topology Identification	138
	References	139
	Afterword	141
	Summary	145
	Samenvatting	147
	Curriculum Vitæ	149

PREFACE

*Het leven als mijn chocolade,
bitter maar verslavend*

Mario Coutino

This thesis was supposed to be thick. When this booklet was conceived, it was supposed to have the two main areas that my research covered: *submodular optimization and graph signal processing*. However, after some discussion with Geert (my promotor), we agreed that, perhaps, being consistent in the story was preferred over the page count. That decision led to the current version of this material, one of the faces of my research during my Ph.D.

Besides the content decision, I wanted to take some freedom and deviate from a traditional Ph.D. thesis. After reading (carefully) the guidelines for the Ph.D. dissertations, I realized that I could not do what I initially planned: a two-sided booklet. Unfortunately, there was no way for me to add some content that I wrote that I wanted to share with people reading this dissertation. So, I had to come up with something else, without compromising the technical quality nor the forms of a proper Ph.D dissertation. Besides all this, I think the current version manages to cut it, although it is not quite there. Anyways.

Though the thesis has several parts, not all of them are aimed for the technical reader. Several parts of this thesis are meant for those who are not so interested in my work but in me. Those to whom I owe an apology. An apology for not being there these years. For being far or for being absent. Thus, beyond any words of gratitude I could utter, I made all the features that are in this booklet, with the exception of Chapters 3 - 7, an apology. An apology for them. An apology to keep in their bookshelf or to enjoy reading, at least once. So, if you are a technical reader, please, indulge me. I will appreciate it.

I had a great time in my Ph.D., here you will find a bit of what I could not share with you. So, please enjoy it.

*Mario Coutiño
Delft, April 2021*

FOREWORD

*Very few people have that effect.
Very few people are tequila and champagne at the same time.*

F. Backman, *Beartown*

As the story of how the booklet came to be has been (briefly) shared, here, we tell a different story. Please, bear with us.

Like all journeys, this one also had to come to an end. And as in all good stories, there were great companions. These lines are the futile attempt to recount the impact of all of them on our journey and how these interactions came to be. If you find yourself being a non-technical reader, wanting to understand this Ph.D. beyond the piece you shared with us (if any), these words, the introduction, and the afterword are your way to the maze (peek at the quotes of each of the chapters at will).

We visited The Netherlands for the first time around two years before we started our studies here. In our visit, we discovered how a small-sized country, full of rules (and people mostly following them), was infused with life by one of the most “successful” programs of multiculturalism in the world (though currently, this is envisioned to change -not necessarily a bad thing). In our walks, alongside the canals and wet streets, we connected with vibrant people whose names we might not remember but that convinced us that this land was an interesting place to experience. We owe them, and this place, more than they could give credit for.

After deciding on moving to the country of the windmills and canals, we found ourselves alone. Although this was not strange for us, constantly moving from one place to another, we were lucky to cross paths with few individuals that enriched and made bearable our early years in The Netherlands. We are forever grateful. Among these personalities, we especially recall the edgy Yalin, the iron-clad Andrejs, the purposeful Kris, the story-teller Aryaman, and the cheerful Joanna and Mariana. We were all there. We somehow managed. And as all our close acquaintances know well, though we tend to disappear from people’s lives as fast as we arrive, no memory is not cherished.

In an inky and sometimes cold country like The Netherlands, we tend to spend long hours, if not days, in our homes. Housemates then, if any, become our bread and butter (even if we do not want it). From our brother Sasho and the always-ready-to-train Brian, passing by the there-is-always-something-to-eat Alex and full-of-lists Nasim, to the diligent Irene, the relentless Ludo, the vegan redditor André and the maker of my cover vision Benoit; for the chats, the parties, failed dishes and the endless topics to take our head away from work, we will be always thankful to all. Wherever life takes you, from the distance, we are there for you.

In our line of work, and at this point in life, offices, unfortunately (maybe), become our second home. And as in our houses, we had some sort of housemates. To our fortune, they sparked an infinite number of [add_adjective] conversations, shared an equal amount of pitas gyros as of fancy dinners, and, of course, made bearable the uncountable hours doing research and everything that it entails. In particular, we are proud of sharing time with Andreas, Sundeep, and Elvin. Somehow, they were not only friends and our seniors but also infused us with the drive that only truly ambitious persons radiate. As friends, we esteem their life advice, and as colleagues, we consider ourselves lucky for having their guidance. To Matt, Jamal, and Tarik, we just want to say thanks for letting us share a reflection of ourselves. You are a part of who we are: a mind craving for things to digest, a sensible soul always looking for words and a disciplined spirit shaped from knowing what sacrifice means. Thanks for that understanding of us. To Pim, perhaps one of the few that will be reading this far, we just wish you the best. By now, you must be either appearing in Linda or in het NOS Jeugdjournaal (goed werk broer). We just want to say thanks for showing us your culture, taking the time to think along with us and for the gezellige tijd we had in your land. To Krishna, whose destiny was far from us, we appreciate your always-present kind and well-mannered words. We will remember all our time together, even that one time you left us in the airport alone; we remember November (or December, the same thing). For the Germans (of course Manss and Manuel, you are also included in this bag!), oh! what can not be said about them? They were, and continue to be, a bit crazy and full of energy. Keep the love for sports, literature, and music. We are grateful for showing us these aspects of your culture, with a Dutch blend perhaps, that enriched our time here. For the guys in the big office -Aydin, Thom, Andreas, Jiani, and partners- besides being thankful for all the game nights that you took part in and the drinks and food we shared, we also have tons of apologies for constantly bothering you. Finally, we thank the new blood of the office -Alberto, Hanie, Metin, and Aybüke. Your young vibe and the continued flow of unanswerable questions keep us, in recent days, always at our toes. Thank you guys for these memories, and remember, the stronger trees are those who take their time to grow.

In an office, we are supposed to work, at least, most of the time. In our case, we had plenty of time and people for that, even if they came (or we crossed paths) for a short time. For these short encounters, we express our gratitude to Ehsan, Shubham, and our Latin, Danish, Japanese, Greek, and Chinese families (professors and students). Though time was short, we managed to get work done, run up and down Fuji in a day, and eat food as if there was no tomorrow. For these memories and the great times, thank you.

As Frodo had Doubledoor and Harry Potter had Grandelf, we had our share of wisdom threw here and there by our spiritual guides. For the leap of faith and unwavering trust, we are forever in debt with Radmila, a strong and well-intended lady from the Balkans that gave us the chance to learn and explore at our heart content. This thesis, by no means, could have not been possible without her fifty cents, almost 6 years ago. The other fifty cents to complete the euro were given by Richard. Due to his bet, we had the chance to meet our Danish family and to discover what it is to listen to music. For that experience, we will be always thankful. To Yorgos, our academic godfather, we can only say that we were delighted to see how much strength can be mustered when you pursue what you love. Your advice, your energy, and your dedication left a profound impression. We were so lucky to see all these things in action. We thank you for the support (and freshly brewed

coffee) that was unconditionally given to us. Finally, we would like to say some words about our promotor, who probably at this point is wondering why he has not appeared. His name is Geert and he is from Belgium. He has a mild character and is very passionate about research. He loves cycling and knows his beers. And though he started as our boss and quickly became our friend, he would never stop being our mentor. For the countless travels, from the visits to paradisiac islands to the tequila factories, and the countless arguments about our inability to pronounce phonemes properly, we can only be grateful. Being able to ride in your wheel made all the difference. Thank you.

Satellites (talking about planets), though not in the mind of many people, play a very important role, e.g., they change the tides and, therefore, also change the rotation of the planet. So, for those people that were not always in the front line of our work but that were crucial to get it done, we extend our thanks. Jorge, Alle-Jan, Richard, Minaskie, Irma, Brigitte, and Rosario, thanks for your support, help, occasional conversations, and for somehow allowing us to be us on the floor. We appreciate the latter as probably without this, this thesis would have never been finished.

Finally, to conclude our recount of companions, we come back home. We go back to the beginning. To our hermanos Ruy and Quique, the guys que always estuvieron y que siguen estando. Gracias for your company, su música, your talks acerca de la vida, our shared dreams, gaming sessions, and por mantener La Paz as a place que no es solo about our padres. We owe you. To the Guadalajara's gang that mantuvo the things moviendo and that have not lost their camino. Gracias for todo, we really apreciamos eso. Bárbara, muchas gracias por tu apoyo incondicional y compañía. Sin importar lo que cuentan nuestras nuevas historias, siempre te estaremos agradecidos por todo.

Emma, voor jou wie als een appeltje die van een boom in de schaduw valt in ons leven bent aangekomen, bedankt. En hoewel je gelukkig of ongelukkig niet in dit verhaal voorkwam, weet zeker dat in wat volgt, als je wil, je er zeker gaat zijn. Jouw meren hebben ons verdronken.

Estas ultimas palabras son para mi familia. La dedicatoria es más que suficiente. Ustedes no necesitan más palabras, siempre me tendrán a mi. Los amo.

This is how this thesis was made: with the help of many, through the hand of one.

*Alberto Minguez
Delft, April 2021*

1

INTRODUCTION

*So many introductions were for you,
let, at least, be one for them.*

Mario Coutiño

Forgetting the philosophical standoff, we can assert that *actions require interactions*. Therefore, it is naive to believe that it is possible to understand the inner workings of processes observed in our daily life, e.g., currency trading, friendship formation, oil pricing, without the understanding of the structure that defines (or supports) the interactions in such systems. For example, it is not possible to fully understand conflict without a proper assessment of how are the relationships among the involved parties. Similarly, we can not expect to produce high-quality predictions of users' consumption patterns, if we do not make use of the information available from users with the same characteristics, e.g., close friends, similar demographics, etc.

Although traditional signal processing has always made use of models and relations in data, e.g., the correlation between measurements, traditional tools are not sufficient to address the challenges that complex interactions, beyond time and space, bring into the table. As an answer, graph signal processing (GSP) has established itself as a balanced mix of the well-known mathematical rigor from signal processing and graph theory, with the empirical modeling seen in network theory. This blend has led to a powerful tool for analyzing data from network processes exploiting all available information about the existing interactions.

This thesis, using GSP as its foundation, aims to provide a further understanding of processes where the interactions between elements of a system are at their core. It presents advanced topics in areas related to how network data has to be processed, how we can implement these data processing pipelines, and how to discover relations between actors in a network process by observation of the network data itself.

In this chapter, we first motivate the use and study of graphs, as well as its combination with signal processing, for analyzing network data. We then provide the scope of the research in the field of GSP and the outline of this thesis. We conclude with our research contributions within GSP and other areas intersecting with signal processing. We do all this considering that the contributions of this thesis must reach an audience not versed in the area.

1.1. WHY GRAPH(S) [SIGNAL PROCESSING]?

1

We (used to) see people holding hands, gathering in bars, or playing board games in the park. We receive work emails addressed to more people than just us. We get involved (even if we do not want it) in annoying messaging or email chains about god-knows-which ways to improve (or protect) our lives. These, like many other signals in our lives, are an indication that we live in a connected world. That despite the fact that the boundaries of human activities are expanding, the fabric of our society is getting denser and denser. The threads (and threats -if you are in the paranoid side) that connect us, and bring us together, are increasing in numbers. Thus, it is foolish to just think about ourselves. We cannot believe anymore that our actions only affect us. When we vibrate, we all move, perhaps, at a different speed or with different amplitudes. The generated ripples affect us all in some way. Interactions are no longer negligible and must be understood.

Similar to societal matters, many more things in this world exhibit complex relations. Molecules bind together to form compounds, e.g., to make drugs that target a particular disease or suppress (excite) the production of certain substances, we combine several compounds. Ecological environments self-regulate due to the balanced interactions between producers, preys, and predators. The consumption and distribution of energy in the electrical grid is the result of the interplay of the different elements comprising it. It is then clear that systems, where interactions are fundamental, are everywhere. And (unfortunately) due to these interconnections, complexity arises. Hence, tools for dealing with such intrinsic complexity are much needed.

Throughout the years, we have rigorously and systematically recollected and analyzed data from *complex systems*, i.e., systems with many components interacting with each other. By using graphs, mathematical abstractions of networks, it has been recognized that we can achieve a better understanding of the collected data from this kind of system if the structure of the interconnections is understood first, or at least, exploited. This approach has led to breakthroughs in modeling, analysis, and forecasting of network data. For example, being able to model the complex relationships between supply and demand in a distribution system leads to improvements in the system's throughput, e.g., more efficient delivery of goods, traffic distribution, servicing, etc.

Despite all the successes achieved so far in analyzing data from networks (complex systems), there are still fundamental challenges to be addressed. First, complex systems, typically represented through networks, tend to be of considerable size, i.e., the number of elements of a system can easily exceed the tens of thousands. Second, most of the interactions observed in such systems are categorical. For instance, although we know if John and Maria are friends, establishing a quantitative value to such a friendship is not always possible. And third, many systems can not aggregate all the information in a single

point; that is, they operate in a distributed manner, thus only exchanging information locally. So, despite the process exhibiting some kind of global behavior, the locality of the exchanges makes the analysis and processing of information in such distributed systems difficult. It is then here where the benefits of cross-pollinating complex systems and graph theory with signal processing can be directly appreciated.

Signal processing has always understood that structure is paramount when performing any data analysis. Typical structures exploited are existing correlations in time, or space, and algebraic structures due to the fundamental laws of the physical process that generates (or acquires) the observations. Simplistically, we could say that signal processing provides a process-aware analysis of measured (observed) data. It combines statistics, linear algebra, and obscurer mathematics with meaningful models drawn from first-principles or domain knowledge to provide reliable and precise data analytics. Therefore, signal processing naturally arises as a tool to provide further insights for network data, complementing both network and graph theory. From this amalgamation, graph signal processing was established, providing structured and principled ways to process data over graphs [1, 2].

1.2. PROCESSING DATA OVER GRAPHS

Processing data has always been the first step towards obtaining meaningful information. Whenever we need to measure a certain parameter (e.g., temperature, political affiliation, flow), make a decision or a prediction (e.g., open a dike, launch a promotion, close a store), we first need to *crunch* the data to remove all irrelevant data (e.g., noise) and then highlight/identify the useful information.

Traditionally, most of the studied data was defined either as measurements taken at different points in space, e.g., the pressure at several locations of a beam; or at different time instants, e.g., the price of a company's stock throughout the year. The fact that the domains where these observations are defined can be *ordered*, i.e., there is a way to sort time or/and organize space, provides a structure to analyze the data. For instance, to predict what products are going to be bought more during holidays, a store could look at previous years' logs of sales to produce an estimate. Similarly, for predicting the temperature of an unobserved region in a room, it could be sufficient to extrapolate the room measurements considering the way that heat propagates in space.

As data is becoming more pervasive and available, data defined over completely new domains have started to arise. An example of this is the overwhelming amount of review data from user rating items in service providers such as Netflix or online retailers like Amazon. Though this data is geographically located (e.g., the users can be localized in a region of the globe) or can be time-stamped to provide a record of the moment in time it was created, it cannot be fully understood (or processed) considering only time or/and space. As a result, we need to accept that there might be other domains, perhaps not as regular as time or space, wherein the data is defined, that need to be leveraged to obtain further insights and thus increase our understanding of the observed data. Fortunately, most of the characteristics of these *irregular* domains, spawned from complex systems, can be captured by *graphs*.

As already mentioned, graphs are abstractions that can represent the interactions in

complex systems. They capture the interactions of different agents, such as Netflix users in our previous example, and provide a mathematical formalism to define models for data generated in these systems. Further, their abstract representation, typically using a matrix, provides a natural way to introduce *translations* in the “network” (graph) domain. Equipped with such notions, traditional signal processing tools, such as filters or detectors, can be readily extended to these irregular domains. The newly extended tools have allowed us to identify anomalies in network processes [3, 4], to clean noisy observations [5–7], predict missing values [8, 9], and design recommender systems [10, 11]. Moreover, the proposed models for describing complex systems have led to the possibility to perform the inverse problem: *identifying system interactions from data* [12, 13]. That is, modeling the information propagation in the network as a *filtering* operation, i.e., a process that removes irrelevant components of a signal, the underlying structure of the network can be retrieved considering that the basic building block of such filters is the mathematical representation of the network topology itself.

Due to the fundamental role that the network structure (topology) and the filtering operations play in processing data coming from complex systems, these two themes become the main object of our study. For filtering, we focus on developing more versatile and computationally attractive instances than “classical” solutions. And to deepen the understanding of the influence of the interactions in the observed data, we dive into the problem of dynamics-aware topology identification.

1.3. SUMMARY OF RESULTS AND OUTLINE

In this thesis, we dwell in recent theoretical and applied results concerning the filtering of signals defined over networks, which throughout this thesis, we refer to as *graph signals*. These results answer the general questions

- Q.1 How can state-of-the-art distributed processing of data over networks, particularly, employing so-called graph filters, be improved?
- Q.2 What are the conditions required, if any, to perform distributed graph filtering in networks without synchronized units?
- Q.3 How to implement and design, in a numerically stable manner, filtering operations on graphs that are efficient in terms of computations and communications?

Besides the above, this thesis also addresses the challenges of identifying the network topology from observations taken from processes occurring in networks and introduces methods to address this problem. This second part of the thesis focuses on answering

- Q.4 How can we discover the system interconnections from observing the evolution of a network process through nodal measurements?
- Q.5 In case we cannot obtain measurements from all the nodes in the network, is it possible to estimate the underlying connectivity of the network from the available nodal measurements?

To discuss our results within our research thrusts, the thesis is organized as follows. We first introduce the mathematical formalism that allows us to carry over notions from traditional signal processing to the network setting. We then provide our particular findings for the above questions. In the following, we provide a summary of the main parts of this thesis and their respective contributions to the state-of-the-art.

1.3.1. TACKLING THE NETWORK DATA DELUGE

This chapter motivates the processing of network data while covering the main concepts of graph signal processing: the mathematical representation of a network, graph signals and shifts on graphs, as well as graph frequencies and filters. Also, this chapter serves as a preamble, setting all necessary concepts, to the main contributions of this thesis and describes in more detail our research thrusts. The material covered in this chapter is classic. And it is presented by first establishing the connection between the matrix representation of a graph and the Laplacian operator.

1.3.2. ADVANCES IN GRAPH FILTERING

This chapter makes use of the graph signal processing formalism and puts forth generalizations of the typical structures employed for graph filters, the so-called *edge-varying graph filters*. To make the discussion complete, we also study the particularities of these graph filter constructions such as their graph frequency interpretation, and their implementation and design. We further present variations of the basic edge-varying graph filters. The proposed edge-varying filter structures, although not as expressive, generalize traditional graph filter structures and provide benefits in terms of computations and communications in the network. We illustrate the benefits of the proposed structures using typical signal processing applications such as distributed beamforming and consensus. Putting all these pieces together, from the theory to meaningful applications, we aim to answer Q.1.

1.3.3. ASYNCHRONOUS GENERALIZED GRAPH FILTERS

Focusing on practical aspects of graph filtering, this chapter discusses the distributed implementation of generalized graph filters in networks without synchronized units. Thus, this chapter focuses on addressing Q.2. To provide an answer to this question, we first introduce a computational model that allows nodes to implement asynchronous updates. Using this model, we then prove that we can perform distributed graph filtering in asynchronous networks as long as the rendition of the generalized graph filters meets certain conditions. We provide a characterization of the requirements of the involved graph filter coefficients and discuss the merits of combining graph filtering with the operator splitting technique.

1.3.4. A CASCADED STRUCTURE OF GRAPH FILTERS

This chapter further binds together theory with practice. Here, we study solutions for the well-known problems that large filter orders bring to filters, which, unfortunately, graph filters do not escape. In this chapter, we discuss the *cascaded implementation* of graph

filters. This implementation provides not only a modular definition but also improves the numerical stability of the filter design stage. Also, we show that, in most instances, it leads to better performance at reduced computational and communication costs. We introduce the cascaded structures by drawing analogs with adaptive filtering and discuss some theoretical properties of the corresponding design problem. Also, we present the RELIEF algorithm, an iterative method, which provides an efficient way to design cascaded graph filters in both model- and data-driven scenarios. Moreover, we discuss the relation of these cascaded structures with the so-called *graph neural networks* (structures that are gaining momentum in machine learning) and present applications that benefit from this type of graph filter implementation. All these results form an ensemble to provide an answer to Q.3.

1.3.5. STATE-SPACE BASED NETWORK TOPOLOGY IDENTIFICATION

Differently from the first half of this thesis, in this chapter, we focus on the inverse problem of *finding the network structure from observed data*. To do so, we first introduce a first-order differential graph model for network processes such as heat diffusion or linear chemical dynamics. Using this model as a starting point, the state-space formalism, borrowed from control theory, is used to relate the topology identification problem to that of *system identification*. Through this connection, we show how to leverage subspace-based techniques from system identification for estimating the underlying structure of the network, therefore addressing Q.4.

1.3.6. NETWORK TOPOLOGY IDENTIFICATION FROM PARTIAL OBSERVATIONS

In this chapter, we finally extend our results concerning network topology inference and study the consequences of *partial observability* of the network process, e.g., observing a reduced subset of actors in the network. This setting occurs in instances where we do not have access to all the nodes, due to budget constraints during measuring campaigns, or because there exist *hidden nodes*; that is, there are nodes in the network we are not aware of. This chapter then tackles Q.5, providing theoretical results concerning the limitations when estimating the topology from partial nodal measurements. Further, this chapter presents a set of practical algorithms to address this scenario that are robust to noisy and incomplete information.

1.4. LIST OF CONTRIBUTIONS

To finalize this introductory chapter, we provide a recount of the contributions made to the scientific literature during the Ph.D. period. We split this list into two. The first part, including thesis-related contributions, are peer-reviewed contributions within GSP, while the second part comprises peer-reviewed contributions in signal processing at large. We highlight the relevant articles for this thesis in [cyan](#).

1.4.1. GRAPH SIGNAL PROCESSING

Journal Papers

- J1 Q. Li, **M. Coutino**, G. Leus and M. Græsbøll, “Privacy-Preserving Distributed Processing by Graph Filtering”, *submitted IEEE Transactions on Signal and Information Processing over Networks*, 2020.
- J2 M. Yang, **M. Coutino**, E. Isufi, G. Leus, “Node-Adaptive Regularization for Graph Signal Reconstruction”, *submitted IEEE Transactions on Signal and Information Processing over Networks*, 2020.
- J3 **M. Coutino**, S. Chepuri, T. Maehara and G. Leus, “Fast Spectral Approximation of Structured Graphs with Applications to Graph Filtering”, *Algorithms*, 2020, 13, 214.
- J4 **M. Coutino** and G. Leus, “[A Cascaded Structure for Generalized Graph Filters](#)”, *submitted to IEEE Transactions on Signal and Information Processing over Networks*, 2020.
- J5 **M. Coutino**, E. Isufi, T. Maehara and G. Leus, “[State-Space Network Topology Identification from Partial Observations](#)”, *IEEE Transactions on Signal and Information Processing over Networks*, vol. 6, 211-225, 2020.
- J6 **M. Coutino**, E. Isufi, G. Leus, “[Advances in Distributed Graph Filtering](#)”, *IEEE Transactions on Signal Processing*, vol. 67 (9), 2320-2333, 2019.

Conference Papers

- C1 A. Natali, **M. Coutino** and G. Leus, “Topology-Aware Joint Graph Filter and Edge Weight Identification for Network Processes”, 30th IEEE International Workshop on Machine Learning for Signal Processing (MLSP) 2020.
- C2 **M. Coutino**, E. Isufi, T. Maehara and G. Leus, “[State-Space based Network Topology Identification](#)”, 28th European Signal Processing Conference (EUSIPCO), Amsterdam, The Netherlands, Sept. 2020.
- C3 Q. Li, **M. Coutino**, G. Leus, and M. Græsbøll Christensen, “Privacy-Preserving Distributed Graph Filtering”, 28th European Signal Processing Conference (EUSIPCO), Amsterdam, The Netherlands, Sept. 2020.

- C4 M. Yang, **M. Coutino**, E. Isufi, and G. Leus, “Node Varying Regularization for Graph Signals”, 28th European Signal Processing Conference (EUSIPCO), Amsterdam, The Netherlands, Sept. 2020.
- C5 **M. Coutino**, GV Karanikolas, G. Leus, and GB Giannakis, “Self-driven Graph Volterra Models for Higher-order Link Prediction”, IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Barcelona, Spain, May 2020.
- C6 Q. Yang, **M. Coutino**, G. Wang, GB Giannakis and G. Leus, “Learning Connectivity and Higher-order Interactions in Radial Distribution Grids”, IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Barcelona, Spain, May 2020.
- C7 **M. Coutino**, S. Chepuri, and G. Leus, “Learning Sparse Hypergraphs from Dyadic Relational Data”, IEEE 8th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), Le Gosier, Guadeloupe, Dec. 2019.
- C8 **M. Coutino**, E. Isufi, F. Gama, A. Ribeiro, and G. Leus, “Design Strategies for Sparse Control Of Random Time-Varying Networks”, 53rd Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, USA, Nov. 2019.
- C9 **M. Coutino** and G. Leus, “[On Distributed Consensus by a Cascade Of Generalized Graph Filters](#)”, 53rd Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, USA, Nov. 2019.
- C10 **M. Coutino** and G. Leus, “[Asynchronous Distributed Edge-Variant Graph Filters](#)”, IEEE Data Science Workshop (DSW), Minneapolis, USA, June 2019.
- C11 G. Ortiz-Jiménez, **M. Coutino**, S. Chepuri, and G. Leus, “Sampling and Reconstruction of Signals on Product Graphs”, IEEE Global Conference on Signal and Information Processing (GlobalSIP), Anaheim, USA, Nov. 2018.
- C12 **M. Coutino**, E. Isufi, T. Maehara and G. Leus, “On The Limits of Finite-Time Distributed Consensus Through Successive Local Linear Operations”, 52nd Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, USA, Oct. 2018.
- C13 **M. Coutino**, S. Chepuri, and G. Leus, “Sparsest Network Support Estimation: A Submodular Approach”, IEEE Data Science Workshop (DSW), Lausanne, Switzerland, June 2018.
- C14 S. Chepuri, **M. Coutino**, A. Marques and G. Leus, “Distributed Analytical Graph Identification”, IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), New Orleans, USA, April 2018.
- C15 **M. Coutino**, E. Isufi, and G. Leus, “[Distributed Edge-variant Graph Filters](#)”, IEEE 7th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), Curacao, Dutch Antilles, Dec. 2017. (**best paper award, ranked 3rd**).

International Tutorials

- P1 G. Leus, E. Isufi and **M. Coutino** “Graph Filters with Applications to Distributed Optimization and Neural Networks”, International Conference on Signal Processing and Communications, Bangalore, India, July 2020.
- P2 G. Leus, E. Isufi and **M. Coutino** “Graph Signal Processing: Connections to Distributed Optimization and Deep Learning”, IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain, May 2020.

1.4.2. SIGNAL PROCESSING

1

Journal papers

- J7 E. Tohidi, **M. Coutino** and D. Gesbert, “Revisiting Matching Pursuit: Beyond Approximate Submodularity”, *submitted to IEEE Transactions on Signal Processing*, 2020.
- J8 B. Li, **M. Coutino**, G. Leus and GB Giannakis, “A Momentum-Guided Frank-Wolfe Algorithm”, *submitted to IEEE Transactions on Signal Processing*, 2020.
- J9 S. Sharma, **M. Coutino**, S. Chepuri, KVS Hari and G. Leus, “Towards a General Framework for Fast and Feasible k-space Trajectories for MRI Based on Projection Methods”, *Magnetic Resonance Imaging*, 2020.
- J10 E. Tohidi, R. Amiri, **M. Coutino**, D. Gesbert, G. Leus and A. Karbasi, “Submodularity in Action: From Machine Learning to Signal Processing Applications”, *IEEE Signal Processing Magazine*, 2020.
- J11 G. Ortiz-Jiménez, **M. Coutino**, S. Chepuri and G. Leus, “Sparse Sampling for Inverse Problems with Tensors”, *IEEE Transactions on Signal Processing*, vol. 67 (12), 3272-3286, 2019.
- J12 E. Tohidi, **M. Coutino**, S. Chepuri, H. Behroozi, M. Nayebi and G. Leus, “Sparse Antenna and Pulse Placement for Colocated MIMO Radar”, *IEEE Transactions on Signal Processing*, vol. 67 (3), 579-593, 2019.
- J13 **M. Coutino**, S. Chepuri and G. Leus, “Submodular Sparse Sensing for Gaussian Detection With Correlated Observations”, *IEEE Transactions on Signal Processing*, vol. 66 (15), 4025-4039, 2018.

Conference papers

- C16 P. vd Meulen, **M. Coutino**, P. Kruizinga, JG Bosch and G. Leus, “Blind Calibration for Arrays with an Aberration Layer in Ultrasound Imaging”, 28th European Signal Processing Conference (EUSIPCO), Amsterdam, The Netherlands, Sept. 2020.
- C17 B. Li, **M. Coutino** and GB Giannakis, “Revisit of Estimate Sequence for Accelerated Gradient Methods”, IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Barcelona, Spain, May 2020.

- C18 T. Kazaz, **M. Coutino**, G. Janssen and A. vd Veen, “Joint blind calibration and time-delay estimation for multiband ranging”, IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Barcelona, Spain, May 2020.
- C19 T. Kazaz, **M. Coutino**, G. Janssen, G. Leus and A. vd Veen, “Joint Ranging and Clock Synchronization for Dense Heterogeneous IoT Networks”, 52nd Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, USA, Oct. 2018.
- C20 **M. Coutino**, S. Chepuri, and G. Leus, “Subset Selection for Kernel-Based Signal Reconstruction”, IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), New Orleans, USA, April 2018.
- C21 **M. Coutino**, S. Chepuri, and Geert Leus, “Sparse Sensing for Composite Matched Subspace Detection”, IEEE 7th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), Curacao, Dutch Antilles, Dec. 2017.
- C22 N. Krishnaprasad, **M. Coutino**, S. Prabhakar Chepuri, D. Fernández-Comesaña, and G. Leus, “DOA Estimation and Beamforming Using Spatially Under-Sampled AVS arrays”, IEEE 7th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), Curacao, Dutch Antilles, Dec. 2017.
- C23 **M. Coutino**, S. Chepuri, and G. Leus, “Near-Optimal Greedy Sensor Selection for MVDR Beamforming with Modular Budget Constraint”, 25th European Signal Processing Conference (EUSIPCO), Kos, Greece, Sept. 2017

Patents

- P1 T. Kazaz, **M. Coutino**, GJM Janssen, G. Leus and AJ vd Veen, “Phase-based Distance Determination For Wireless Sensor Networks”, *Patent, USPTO* 621 81 5,1 64, March 2019.

REFERENCES

- [1] A. Sandryhaila and J. M. Moura, *Discrete signal processing on graphs*, *IEEE Trans. Signal Process* **61**, 1644 (2013).
- [2] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, *The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains*, *IEEE Sig. Proc. Mag.* **30**, 83 (2013).
- [3] H. Wang, M. Tang, Y. Park, and C. E. Priebe, *Locality statistics for anomaly detection in time series of graphs*, *IEEE Transactions on Signal Processing* **62**, 703 (2013).
- [4] H. E. Egilmez and A. Ortega, *Spectral anomaly detection using graph-based filtering for wireless sensor networks*, in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (IEEE, 2014) pp. 1085–1089.
- [5] A. C. Yağan and M. T. Özgen, *A spectral graph wiener filter in graph fourier domain for improved image denoising*, in *Sig. and Inf. Proc. (GlobalSIP), 2016 IEEE Global Conference on* (IEEE, 2016) pp. 450–454.
- [6] M. Onuki, S. Ono, M. Yamagishi, and Y. Tanaka, *Graph signal denoising via trilateral filter on graph spectral domain*, *IEEE Trans. on Sig. and Inf. Proc. over Netw.* **2**, 137 (2016).

- [7] E. Isufi, A. Loukas, A. Simonetto, and G. Leus, *Autoregressive moving average graph filtering*, *IEEE Transactions on Signal Processing* **65**, 274 (2016).
- [8] S. K. Narang, A. Gadde, and A. Ortega, *Signal processing techniques for interpolation in graph structured data*, in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)* (IEEE, 2013) pp. 5445–5449.
- [9] Y. Mao, G. Cheung, and Y. Ji, *Image interpolation for dibr viewsynthesis using graph fourier transform*, in *3DTV-Conference: The True Vision-Capture, Transmission and Display of 3D Video (3DTV-CON), 2014* (IEEE, 2014) pp. 1–4.
- [10] J. Ma, W. Huang, S. Segarra, and A. Ribeiro, *Diffusion filtering of graph signals and its use in recommendation systems*, in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)* (IEEE, 2016) pp. 4563–4567.
- [11] W. Huang, A. G. Marques, and A. Ribeiro, *Collaborative filtering via graph signal processing*, in *2017 25th European Signal Processing Conference (EUSIPCO)* (IEEE, 2017) pp. 1094–1098.
- [12] G. B. Giannakis, Y. Shen, and G. V. Karanikolas, *Topology identification and learning over graphs: Accounting for nonlinearities and dynamics*, *Proceedings of the IEEE* **106**, 787 (2018).
- [13] G. Mateos, S. Segarra, A. G. Marques, and A. Ribeiro, *Connecting the dots: Identifying network structure via graph signal processing*, *IEEE Signal Processing Magazine* **36**, 16 (2019).

2

TACKLING THE NETWORK DATA DELUGE

*among the crowd i felt you;
there's no path, no consensus;
we dwell, in o'r thoughts,
together but alone*

@fftmod

Traditionally, time and space have been two physical *domains* that have allowed for defining a natural way to organize and explore data. For example, a series of stock prices during a year are considered as a *time series*; that is, a sequence of quantities that are ordered based on their recording time. Similarly, the concentration of a certain gas or the temperature in a geographical region are typical examples of *field measurements*. Here, the data is structured based on the (geographic) location where the measurements are taken. Naturally, the combination of such domains can be considered, i.e., spatio-temporal domain, thus a structure is naturally imposed on this data when it is examined over a window of time in a particular spatial region. Figure 2.1 provides examples of both temporal and spatial signals.

The inherent structure of the data, derived from its definition domain, provides ways to analyze it and highlights properties that are quantifiable and interpretable. For instance, the duality between the time domain and the so-called frequency domain [1] allows us to explore patterns within time-series such as periodicity or bursts of activity. Also, the existing ordered structure permits us to define a notion of *similarity* or *closeness* that commonly reveals properties in the data. For example, in the case of temperature measurements, it is expected that due to the spatial nature of the observations, data taken at spatially-close locations present some level of similarity.

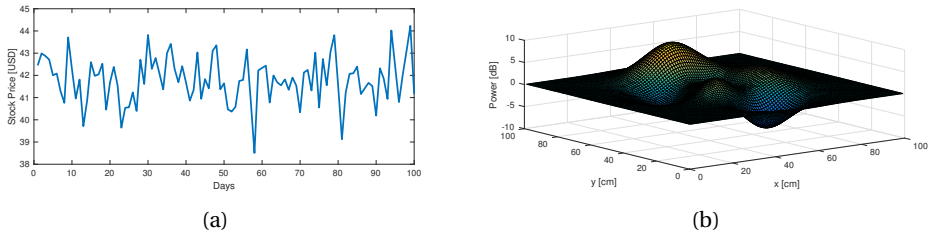


Figure 2.1: (a) Daily price of a company stock in US dollars. (b) Field measurement over a small spatial region.

2

We consider domains like space and time as *regular* domains; that is, they are domains in which an order is present or where a distance between their elements can be defined straightforwardly. However, for data describing the behavior of complex interactions between interconnected components such as the number of mutual likes given among users in social media or the exchanges of substances between elements of a biological network, neither time nor space capture the full complexity of it. This situation arises because the data lives in a different domain: the one determined by the interaction between elements. Unfortunately, networks are not, in general, “regular” because the definition of an order or distance between their elements is not, in all cases, possible to establish unambiguously. Thus, when dealing with this kind of data, the following questions arise naturally: How should we process and study data defined over *irregular* domains? What are the tools that provide an interpretable analysis of the data? Is there a framework that encompasses the analysis of both regular and irregular domains? The field of graph signal processing [2–4] has been put forth from within the signal processing community to address these questions. In this chapter, we present this field as a complete framework able to cope with the increasing amount of network-related data and introduce the problems within this area that comprise the two research thrusts that are contained in this thesis.

2.1. GRAPH SIGNAL PROCESSING

We consider a network to be represented by a *graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, \dots, v_N\}$ is the set of N vertices (nodes) and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the edge set of M tuples $e_{i,j} = (v_i, v_j)$ representing the connections between the nodes in the network. As a network, the connections in a graph can have *direction*; that is, a bidirectional exchange of information between nodes i and j does not necessarily have to exist. This behaviour is captured, and represented abstractly, through either directed or undirected graphs. A graph is said to be *undirected* if there is no orientation of the edges (information flow) for all tuples $(v_i, v_j) \in \mathcal{E}$, otherwise the graph is called *directed*. Figure 2.2 illustrates the difference between these two types of graphs. When there is information about the strength of the connection, i.e., $w_{i,j} \in \mathbb{R}_+$ is the weight of the edge $(v_j, v_i) \in \mathcal{E}$, we say that the graph is a *weighted graph*, otherwise we consider it as an *unweighted graph*.

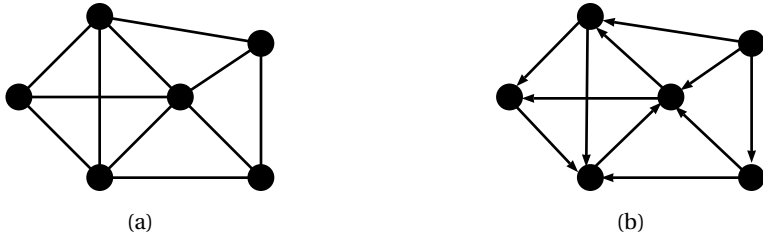


Figure 2.2: (a) Undirected graph. (b) Directed graph. The point of the arrows indicates the direction of the information “flow”.

2.1.1. REPRESENTATION OF A GRAPH

Although there are many ways to represent a graph, throughout this thesis, we consider its matrix representation. We assume that the connections in a graph are encoded through an $N \times N$ matrix, \mathbf{S} , where the (i, j) th entry, $[\mathbf{S}]_{i,j}$, represents the existence (and possibly the strength) of the connection between the j th and i th node in the network.

In the literature, there are several matrices that are commonly used to represent a graph such as the *adjacency matrix*, $\mathbf{W} \in \{0, 1\}^{N \times N}$, which captures only the structure of the graph, meaning that its nonzero entries represent the presence of a connection in the network, i.e., $[\mathbf{W}]_{i,j} = 1$ if $(v_j, v_i) \in \mathcal{E}$, otherwise it is set to zero. When instead of a link indicator the weight of the connection is used, the matrix representation, $\mathbf{W} \in \mathbb{R}^{N \times N}$, where $[\mathbf{W}]_{i,j} = w_{i,j}$, is referred to as the *weighted adjacency matrix*. If all nonzero weights are set to one, this matrix reduces to the adjacency matrix. In many instances, graphs are considered to be *simple*; that is, they do not have *self-loops*, i.e., connections from the nodes to themselves, implying that $(v_i, v_i) \notin \mathcal{E}, \forall i \in \mathcal{V}$. Other graph-related matrices of importance are the following

- Degree matrix: $\mathbf{D} = \text{diag}(\mathbf{W}\mathbf{1})$.
- Laplacian matrix: $\mathbf{L} = \mathbf{D} - \mathbf{W}$.
- Normalized Laplacian: $\mathbf{L}_N = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$.
- Scaled adjacency matrix: $\mathbf{W}_s = \frac{1}{|\lambda_{\max}(\mathbf{W})|}\mathbf{W}$.

Here, $\lambda_{\max}(\mathbf{W})$ denotes the eigenvalue of \mathbf{W} with the largest magnitude, i.e., $\lambda_{\max}(\mathbf{W}) := \max_{0 \leq i \leq N-1} |\lambda_i|$. For directed graphs, the degree matrix can either be defined as the *in-degree*, i.e., $\mathbf{D}_{\text{in}} = \mathbf{W}\mathbf{1}$, or the *out-degree*, i.e., $\mathbf{D}_{\text{out}} = \mathbf{W}^T\mathbf{1}$, matrix. The choice between them depends on the application.

Though several matrices are candidates to represent a graph, in this thesis, we do not advocate the usage of one over another. However, we emphasize that the support of the matrix representations, i.e., the nonzero entries, directly relates to the connections in the network.

Finally, we point out that although the values assigned to the network connections provide a natural way to establish similarity between elements in the domain, i.e., nodes, the resulting abstract object is a mathematical structure that is heavily dependent on the values assigned to it.



Figure 2.3: Illustration of an infinite directed chain graph.

2.1.2. SIGNALS OVER GRAPHS

When the domain of a signal in question is a network, we say the signal is “defined over a graph”. This definition means that there is a mapping from the nodes of a network (graph) to the set of, let us say, real numbers, i.e., $x_i : v_i \rightarrow \mathbb{R}$. For simplicity of notation, the node signals are collected in a vector, $\mathbf{x} \in \mathbb{R}^N$, where $x_i := [\mathbf{x}]_i$ denotes the signal of the i th node in the network.

2

2.1.3. SHIFTS ON GRAPHS

Having a regular structure facilitates the possibility of defining so-called *shift* operations due to the natural order of the domain. For example, in the time domain, shift operations can be easily defined because time can be “advanced” or “delayed”. However, in networks, there is no clear distinction, in terms of their order, between one node and another. So, shifts can not be established as intuitively as in the case of regular domains.

To define a notion of shift in networks, the matrix representation of the network, \mathbf{S} , plays a very important role. This is the reason why within graph signal processing, this matrix is commonly known as the *graph shift operator* (GSO). Using the so-called GSO, a shifted version of a graph signal \mathbf{x} is achieved by performing the following operation

$$\mathbf{x}^{(k)} = \mathbf{S}^k \mathbf{x} \quad (2.1)$$

where $\mathbf{x}^{(k)}$ stands for the k th shift of \mathbf{x} over \mathbf{S} . As any other shift operation, this can be computed recursively, i.e., $\mathbf{x}^{(k)} = \mathbf{S}^k \mathbf{x} = \mathbf{S} \mathbf{S}^{k-1} \mathbf{x} = \mathbf{S} \mathbf{x}^{(k-1)}$. Besides, as the support of the matrix \mathbf{S} is shared to that of the underlying network, the computation of the matrix-vector product $\mathbf{S} \mathbf{x}^{(k)}$, $\forall k \in \mathbb{N}$ can be computed *locally*. This means that the product of any vector with the GSO can be efficiently computed in a distributed manner using local exchanges of information, i.e., data transmission and aggregation between neighbors.

The main reason why (2.1) is considered to define a shift in the graph domain is due to the relation with the shift operator in the time domain. To illustrate the connection, let us consider an infinite long sequence of nodes connected in a chain; see Figure 2.3 for an example. Though the graph formed by this sequence is neither finite nor symmetric, it can still be represented by a matrix of infinite dimensions whose support is defined by the connections in the chain, i.e.,

$$\mathbf{S}_\infty = \begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 0 & \cdots \\ 0 & 0 & 1 & 0 & \cdots & 0 & \cdots \\ 0 & 0 & 0 & \ddots & \cdots & 0 & \cdots \\ \vdots & \vdots & \vdots & \cdots & 1 & \vdots & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \in \{0, 1\}^{\infty \times \infty}. \quad (2.2)$$

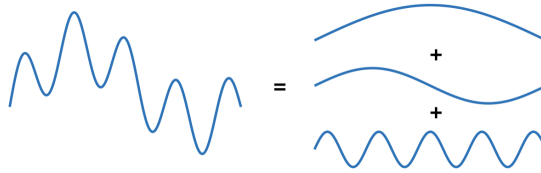


Figure 2.4: Illustration of the decomposition of a temporal signal into its different frequency components. Notice the increase in *variation* of the different components.

Assuming any quantization of time, i.e., a partition in intervals of finite length, a “forward” shift in time of an infinitely long signal, $\mathbf{x}_\infty \in \mathbb{R}^\infty$, is given by the product $\mathbf{S}_\infty \mathbf{x}_\infty$. As (2.1) can be seen as the generalization of this procedure, for any kind of graph, the definition of the shift in the graph domain is done through it.

2

2.1.4. FREQUENCY INTERPRETATION OVER GRAPHS

The concept of frequency provides a notion of *variation*. When dealing with a time series, its frequency-domain representation decomposes it in components with an increasing rate of variation, i.e., signals with a higher number of zero crossings per unit of time. Similarly, for spatial domains, a spatial frequency interpretation can be obtained by decomposing a spatial signal into different “spatial frequencies” also called *resonant modes*. An example of a time-domain signal decomposed in its frequency components is shown in Figure 2.4.

Although the frequency-domain representation of a signal is a concept that is always related to variation, it is an expansion of the original signal onto an orthogonal basis of the space wherein the signal is defined. However, there is a caveat, this basis is not any basis. It captures the inherent structure of the original domain. In particular, this representation respects the existing symmetries of the domain [5]. Hence, how can we identify what are the symmetries in the domain of interest? Or what object exhibits those symmetries such that a basis considering them is obtained? To answer all these questions, we rely on the *Laplace operator*.

As a network is similar to a two-dimensional Euclidean space, we first motivate the definition of frequencies in a graph using this regular domain. And then, we go back to establish the link with the frequency representation derived from the time domain. For a two-dimensional Euclidean space, the Laplace operator is given, in Cartesian coordinates, i.e., \mathbb{R}^2 , by

$$\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \quad (2.3)$$

where f is defined over \mathbb{R}^2 . This operator has the following properties:

- For ρ being an orthogonal transformation, e.g., rotation or reflection,

$$\Delta(f \circ \rho) = (\Delta f) \circ \rho.$$

- For τ being a translation or spatial shift,

$$\Delta(f \circ \tau) = (\Delta f) \circ \tau.$$

These properties show that the Laplace operator *commutes* with fundamental operations in \mathbb{R}^2 . Due to these commuting properties of the Laplacian, which show that it respects the fundamental symmetries in \mathbb{R}^2 , we commonly expand spatial signals defined in \mathbb{R}^2 in the eigenbasis of Δ . This basis consists of the so-called *double Fourier series*, where there is a Fourier cosine series in x and a Fourier sine series in y . Similarly, for the case of the spheric domain in \mathbb{R}^3 , the decomposition of signals usually is carried out through *spherical harmonics* that not only form a basis for the group of rotations in three dimensions but they are also the *eigenfunctions* of the corresponding Laplace operator.

As the Laplace operator “unveils” intrinsic properties of the structure of the domain, it is natural to ask if the Laplacian matrix of the graph provides a way to construct a basis respecting the properties of the underlying network domain. Fortunately, as the Laplacian matrix is the discrete counterpart of the Laplace operator, it acquires meaning on graphs (networks), extending discrete grids, i.e., uniform quantization of a finite space, to non-regular instances. Therefore, considering the Laplace matrix \mathbf{L} of a graph \mathcal{G} and assuming the existence of its eigenvalue decomposition, i.e.,

$$\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1} \quad (2.4)$$

where $\mathbf{U} = [\mathbf{u}_1, \dots, \mathbf{u}_N]$ is the $N \times N$ matrix containing the eigenvectors of \mathbf{L} and $\mathbf{\Lambda} = \text{diag}(\lambda_1, \dots, \lambda_N)$ is the $N \times N$ diagonal matrix whose i th diagonal element is the i th eigenvalue of \mathbf{L} , we can obtain the coefficients of the expansion of a graph signal, \mathbf{x} , in terms of the eigenbasis of \mathbf{L} , by

$$\hat{\mathbf{x}} = \mathbf{U}^{-1}\mathbf{x}. \quad (2.5)$$

From (2.5), it is clear that using these coefficients, the graph signal can be expressed through the eigenbasis of the Laplacian as

$$\mathbf{x} = \mathbf{U}\hat{\mathbf{x}} = \sum_{i=1}^N [\hat{\mathbf{x}}]_i \mathbf{u}_i. \quad (2.6)$$

Using this decomposition, we can analyze graph signals using the underlying structure of the irregular domain. In this analysis, the different eigenvectors, which are commonly referred to as (spatial) *modes*, capture the underlying domain’s structure.

The relation of the decomposition in terms of the eigenvectors of the Laplace operator with variation comes from the fact that in the majority of the cases the Laplace operator is *self-adjoint*, i.e., $\langle \Delta f, g \rangle = \langle f, \Delta g \rangle$, where $\langle \cdot, \cdot \rangle$ represents the corresponding inner product. For instance, in the case of finite-dimensional spaces, such as physical networks, this occurs when the Laplacian is *symmetric* (and hence the graph is undirected). In such a case, the Laplacian is diagonalizable by an orthonormal basis, i.e.,

$$\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^H \quad (2.7)$$

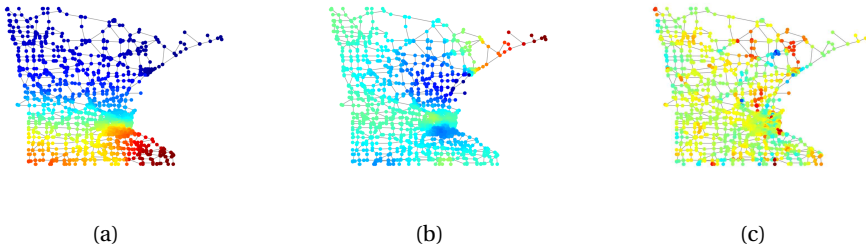


Figure 2.5: Comparison of three graph signals with different degrees of smoothness. (a) Smooth signal. (b) Mildly smooth signal. (c) Non-smooth signal.

and the eigenvectors of \mathbf{L} can be obtained recursively as the minimizers of the Rayleigh quotient, i.e.,

$$\begin{aligned} \mathbf{u}_i = \operatorname{argmin}_{\mathbf{z} \in \mathbb{R}^N} \quad & \mathbf{z}^\top \mathbf{L} \mathbf{z} \\ \text{s.t.} \quad & \|\mathbf{z}\|_2 = 1, \\ & \mathbf{z} \perp \operatorname{span}\{\mathbf{u}_1, \dots, \mathbf{u}_{i-1}\} \end{aligned} \quad (2.8)$$

where $\lambda_i = \mathbf{u}_i^\top \mathbf{L} \mathbf{u}_i \leq \lambda_{i+1} = \mathbf{u}_{i+1}^\top \mathbf{L} \mathbf{u}_{i+1}$.

Through the variational definition of the eigenvectors of the Laplacian [cf.(2.8)], eigenvectors related with smaller eigenvalues have a smaller norm induced by the geometry imposed by \mathbf{L} , i.e., they have a smaller quadratic form. That is, they are *smoother* with respect to the underlying structure of the domain. In the case of graphs, this property can be appreciated by expanding the quadratic form in terms of the node signals, i.e.,

$$\mathbf{x}^\top \mathbf{L} \mathbf{x} = \sum_{(v_i, v_j) \in \mathcal{E}} [\mathbf{W}]_{i,j} (x_i - x_j)^2. \quad (2.9)$$

When this quadratic form is small, and the weights are nonnegative, the node signals of neighboring nodes have to be close to each other; that is, the graph signal is smooth (varies slowly) over the connectivity of the network. Hence, the concept of variation on regular domains naturally extends to the network domain by the Laplace matrix. An example of signals with different levels of smoothness, over a given graph, is shown in Figure 2.5.

At this point, the relation of the eigenvectors of the Laplacian with variation on the graph has been explained through its relation with the Laplace operators of regular domains. However, to establish such relation, we have relied on the *symmetry* of the network representation, i.e., $\mathbf{L} = \mathbf{L}^\top$. That is, we have only considered the case of *undirected* graphs, thus failing to make a connection with the notion of frequency from the time domain, which arises from the *directed* nature of the domain, i.e., time progresses.

To elaborate on the connection with the time domain, we first need to recall our motivation to expand graph signals in the Laplacian eigenvectors. We argued that because the two-dimensional Laplace operator commutes with the fundamental operations in \mathbb{R}^2 ,

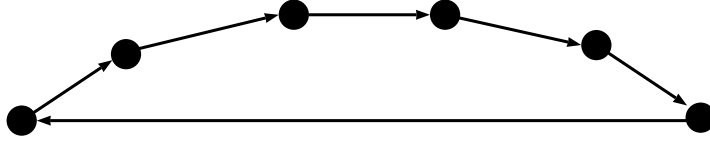


Figure 2.6: Example of a cyclic directed graph.

e.g., rotations, translations, etc., it respects the fundamental symmetries in \mathbb{R}^2 . Notice that when these operators are “symmetric” operators, the eigenfunctions of such operators are identical. Thus, this condition translates, in the finite-dimensional case, to the fact that the fundamental operations and the Laplacian matrix share their eigenvectors, i.e., the matrices are jointly diagonalizable with an orthonormal basis. As a result, we can deduce that the basis used for the expansion must be eigenfunctions (eigenvectors) of “shift” operations in the domain where the signal lives.

For the case of the time domain, it is well known that complex exponentials are eigenfunctions of shift operations, i.e., time translations. Hence, we can expand time series through complex exponential functions; that is, using a Fourier expansion. To illustrate this, let us focus on the periodic time domain. By considering the circulant shift matrix

$$\mathbf{S}_{\text{mod } N} = \begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 0 & \cdots \\ 0 & 0 & 1 & 0 & \cdots & 0 & \cdots \\ 0 & 0 & 0 & \ddots & \cdots & 0 & \cdots \\ \vdots & \vdots & \vdots & \cdots & 1 & \vdots & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \vdots & \vdots & \vdots & \vdots & \vdots & 0 \end{bmatrix} \in \{0, 1\}^{N \times N} \quad (2.10)$$

we can identify two things: (i) $\mathbf{S}_{\text{mod } N}$ is a circulant matrix and (ii) it represents the adjacency matrix of a cyclic directed graph; see Figure 2.6. Due to (i), the discrete Fourier transform (DFT) matrix diagonalizes $\mathbf{S}_{\text{mod } N}$. Hence, as the Laplacian matrix is constructed by subtracting $\mathbf{S}_{\text{mod } N}$ (adjacency matrix) from a diagonal matrix, the Laplacian matrix is also circulant. Thus, despite the lack of symmetry of \mathbf{L} on the periodic time domain, the eigenvectors of the respective Laplacian match those of $\mathbf{S}_{\text{mod } N}$ and are given by the DFT. The operation $\mathbf{U}^{-1}\mathbf{x}$ is then nothing else than the DFT of \mathbf{x} , the discretized frequency representation of the signal¹. Therefore, the operation in (2.5) used to obtain the expansion coefficients of the graph signal in terms of the *eigenbasis of the shift operator* is what within graph signal processing is known as the *graph Fourier transform* (GFT), while (2.6) is referred to as the *inverse graph Fourier transform* (IGFT). And the notion that the eigenvalues of the Laplacian are the *graph frequencies* is inherited from the time domain interpretation of these quantities.

Though so far we have made use of the Laplacian matrix to build insights into the selection of the basis for expanding graph signals, we remark two things: (i) this matrix is

¹The same situation occurs in the infinite time domain where instead of being circulant, the involved matrices are Toeplitz.

not the only matrix that can be used as graph shift operator, and (ii) the eigenbasis (eigenfunctions) of the adjacency and the Laplacian matrices, in general, do not coincide. As a result, in practice, *the selection of the eigenbasis to expand graph signals becomes application dependent* and hence also the selection of the matrix to represent the shift operator in the graph, i.e., which graph matrix to employ as \mathbf{S} . Moreover, while in the case of an undirected graph the “graph frequencies” are *ordered*, i.e., are real-valued, for arbitrary directed graphs, the eigenvalues of the selected graph shift operator, e.g., adjacency, Laplacian, etc., are complex-valued, lacking thus of a total order (the exception is the time domain where the graph frequencies are uniformly distributed over the complex unit circle). To provide a notion of “order” and therefore of *variation* (frequency) in the general case, within GSP, the graph frequencies are then proposed to be ordered by the *complexity* of their spectral components. A common way to measure complexity is by measuring the *total variation*, i.e.,

$$\text{TV}_{\mathcal{G}}(\mathbf{u}_k) := \|\mathbf{u}_k - \frac{1}{|\lambda_{\max}(\mathbf{S})|} \mathbf{S} \mathbf{u}_k\|_1, \quad (2.11)$$

where $\|\cdot\|_1$ is the ℓ_1 -norm and \mathbf{u}_k is the k th eigenvector of \mathbf{S} . Through this measure we can order the graph frequencies $\{\lambda_i\}$ and claim that $\lambda_i \geq \lambda_j$ if

$$\text{TV}_{\mathcal{G}}(\mathbf{u}_i) \geq \text{TV}_{\mathcal{G}}(\mathbf{u}_j). \quad (2.12)$$

Thus using this ordering the traditional notions of low- and high-pass filtering directly translate to directed graphs (not symmetric domains). To finalize, we point out that although the definition for $\text{TV}_{\mathcal{G}}(\cdot)$ in (2.11) makes use of the ℓ_1 -norm, other choices of norms are possible, e.g., see [2].

2.2. FILTERING IN IRREGULAR DOMAINS

When a signal is decomposed into its fundamental modes or in its *frequency components*, it is natural to desire to isolate or enhance some of its frequencies for further analysis or to perform signal processing tasks such as denoising. The workhorse of signal processing for this endeavor consists of *filters* [1]. Typically, filters retain components of a signal within the desired frequency band, e.g., low or high frequency bands, or shape its frequency content to obtain the desired response. These filters are commonly either implemented in the time or spatial domain depending on the type of signal, or in their related frequency domain. An example of time-domain filtering and its effect in the frequency domain is shown in Figure 2.7

Considering a signal decomposition $\hat{\mathbf{x}}$ [cf. (2.5)], the content of the signal can be altered by a point-wise multiplication operation, i.e., “spectrum shaping”, as

$$\hat{\mathbf{y}} = h(\Lambda) \hat{\mathbf{x}} \quad (2.13)$$

where $h(\Lambda) = \text{diag}(h(\lambda_1), \dots, h(\lambda_N))$ is the filtering operation in the representation domain. In case of the (graph) frequency domain, $h(\cdot)$ is the so-called (*graph*) *filter frequency response*.

Synthesising the signal back to the original domain [cf. (2.6)], i.e.,

$$\mathbf{y} = \mathbf{U} \hat{\mathbf{y}} = \mathbf{U} h(\Lambda) \hat{\mathbf{x}} \quad (2.14)$$

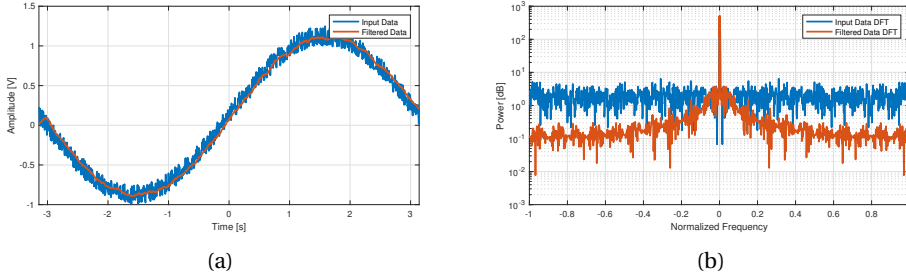


Figure 2.7: Illustration of time-domain filtering and its effect in the frequency domain. (a) Comparison of noisy signal (input) with filtered signal (output). (b) Spectra of both input and output signals. Higher frequencies are attenuated due to the filtering operation, reducing the jitter of the output signal.

2

exposes the fact that the filtering operation is representable by a linear operator diagonalizable by the decomposition basis, \mathbf{U} ; that is,

$$\mathbf{y} = \mathbf{U}h(\Lambda)\mathbf{U}^{-1}\mathbf{x} = \mathbf{H}\mathbf{x} \quad (2.15)$$

where $\mathbf{H} := \mathbf{U}h(\Lambda)\mathbf{U}^{-1}$ is the linear operator that implements the filtering operation on the signal \mathbf{x} .

The diagonalization property of the filtering operation implies that \mathbf{H} commutes with the graph shift matrix, \mathbf{S} , e.g.,

$$\mathbf{H}\mathbf{S} = \mathbf{S}\mathbf{H}. \quad (2.16)$$

Within graph signal processing, this property is known as the *shift-invariant* property as the matrix representation of the graph is considered as a shift in the graph domain.

2.2.1. GRAPH FILTERS

After establishing how to perform filtering over an irregular domain such as a graph, we here specialize this notion to instantiate particular types of graph filters introduced by the graph signal processing community.

2.2.2. FINITE-IMPULSE-RESPONSE GRAPH FILTERS

An important result of the joint diagonalizability of the filter operator \mathbf{H} and the shift operator \mathbf{S} is that \mathbf{H} can be expressed as a polynomial of \mathbf{S} , i.e.,

$$\mathbf{H} = p(\mathbf{S}) := \sum_{k=0}^K \phi_k \mathbf{S}^k = \mathbf{U} \sum_{k=0}^K \phi_k \Lambda^k \mathbf{U}^{-1} = \mathbf{U}p(\Lambda)\mathbf{U}^{-1} \quad (2.17)$$

where $K < N$ and $\{\phi_k \in \mathbb{R}\}_{k=1}^K$, as long as its spectrum is *simple*. That is, as long as the graph shift operator has eigenvalues with multiplicity one. This statement can be shown to hold by an interpolation argument. The pairs $\{\lambda_i, h(\lambda_i)\}_{i=1}^N$ form a set of N -ordered pairs. Thus,

a polynomial of *at most* order $N - 1$ is needed to *exactly* interpolate such points. This is equivalent to solving the linear system

$$\begin{bmatrix} 1 & \lambda_1 & \lambda_1^2 & \cdots & \lambda_1^{N-1} \\ 1 & \lambda_2 & \lambda_2^2 & \cdots & \lambda_2^{N-1} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & \lambda_N & \lambda_N^2 & \cdots & \lambda_N^{N-1} \end{bmatrix} \begin{bmatrix} \phi_0 \\ \phi_1 \\ \vdots \\ \phi_{N-1} \end{bmatrix} = \begin{bmatrix} h(\lambda_1) \\ h(\lambda_2) \\ \vdots \\ h(\lambda_N) \end{bmatrix} \quad (2.18)$$

which can be shown to be nonsingular if all eigenvalues have multiplicity one.

Looking at the structure of (2.17), we observe that the filtering operation is a weighted sum of $K + 1$ delayed versions of the original graph signal [cf. (2.1)] i.e.,

$$\mathbf{y} = \mathbf{H}\mathbf{x} = \sum_{k=0}^K \phi_k \mathbf{S}^k \mathbf{x} = \sum_{k=0}^K \phi_k \mathbf{x}^{(k)}. \quad (2.19)$$

The composition of this filtering operation is analogous to that of time domain filters, where the filter output is computed as a weighted sum of delayed versions of the signal. This is also the reason why graph filters constructed as

$$\mathbf{H}_c = \sum_{k=0}^K \phi_k \mathbf{S}^k \quad (2.20)$$

are known as (*classical*) *finite-impulse-response* (FIR) graph filters. An advantage of this kind of filters, in terms of computations, is that they can be run distributively due to the locality of \mathbf{S} [6, 7]. In particular, since $\mathbf{S}^k \mathbf{x} = \mathbf{S}(\mathbf{S}^{k-1} \mathbf{x})$ the nodes can compute locally the k th shift of \mathbf{x} from the former $(k - 1)$ th shift. Overall, an FIR filter of order K requires K local exchanges between neighbors and amounts to a computational and communication complexity of $\mathcal{O}(MK)$.

Although graph filters of the form (2.20) follow the theory of filtering in irregular domains, the set of operations that are achievable are restricted to operators that commute with \mathbf{S} . Although these operators must suffice for applications that require a shaping of the frequency domain, in many instances, we need operators that are not linked to the signal's graph frequency decomposition. To expand the possible set of operations that can be implemented distributively through sequential local aggregations, [7] proposed the node-variant (NV) FIR graph filter. These filters have the form

$$\mathbf{H}_{\text{nv}} \triangleq \sum_{k=0}^K \text{diag}(\boldsymbol{\phi}_k) \mathbf{S}^k \quad (2.21)$$

where the vector $\boldsymbol{\phi}_k = [\phi_{k,1}, \dots, \phi_{k,N}]^T$ contains the node dependent coefficients applied at the k th shift. Note that for $\boldsymbol{\phi}_k = \phi_k \mathbf{1}$, the NV FIR filter (2.21) reduces to the classical FIR filter (2.20). Furthermore, from (2.21), we can notice that the NV FIR filter preserves also the efficient implementation of (2.20) since it relies on the same distributed implementation of the shift operator \mathbf{S} . Thus this filter also has a computational and communication complexity of $\mathcal{O}(MK)$.

2.2.3. INFINITE-IMPULSE-RESPONSE GRAPH FILTERS

Similar to the traditional time-domain processing, we can also consider graph filters whose graph frequency response is given by a rational function instead of a polynomial, i.e.,

$$h(\lambda) = \frac{p_n(\lambda)}{p_d(\lambda)} = \frac{\sum_{k=0}^K b_k \lambda^k}{1 + \sum_{k=1}^K a_k \lambda^k}. \quad (2.22)$$

This kind of filters is typically used when an FIR filter requires too many taps (shifts) to achieve a desired frequency response. Building on this idea, in [8, 9], the authors introduced an autoregressive moving-average (ARMA) recursion on graphs to distributively implement a filtering operation characterized by a rational graph frequency response. Such filters are referred to as infinite-impulse-response (IIR) graph filters due their analogy with their time-domain processing counterparts.

The basic building block of IIR filters is an ARMA graph filter of order one (ARMA₁). This kind of filter is obtained as the steady-state of the first-order recursion

$$\mathbf{y}_t = \psi \mathbf{S} \mathbf{y}_{t-1} + \varphi \mathbf{x} \quad (2.23)$$

with arbitrary \mathbf{y}_0 and scalar coefficients ψ and φ . The operation (2.23) is a distributed recursion on graphs, where neighbors now exchange their former output \mathbf{y}_{t-1} rather than the input \mathbf{x} . The per-iteration complexity of such a recursion is $\mathcal{O}(M)$. Given that ψ satisfies the convergence conditions for (2.23) [9], the steady-state output of the ARMA₁ is

$$\mathbf{y} \triangleq \lim_{t \rightarrow \infty} \mathbf{y}_t = \varphi \sum_{\tau=0}^{\infty} (\psi \mathbf{S})^\tau \mathbf{x} = \varphi (\mathbf{I} - \psi \mathbf{S})^{-1} \mathbf{x} = \mathbf{H}_{\text{arma}_1} \mathbf{x} \quad (2.24)$$

where $\mathbf{H}_{\text{arma}_1} := \varphi (\mathbf{I} - \psi \mathbf{S})^{-1}$. Notice that $\mathbf{H}_{\text{arma}_1}$ commutes with the shift \mathbf{S} and despite that it has a different definition, it can still be represented as a polynomial of \mathbf{S} . However, by introducing the inverse in the definition of $\mathbf{H}_{\text{arma}_1}$, a different type of distributed implementation of the graph filter is available with its related benefits. Finally, we remark that this family of filters is directly linked to several GSP tasks including Tikhonov denoising, graph signal interpolation under a smoothness prior [9], and aggregate graph signal diffusion [10].

2.2.4. GRAPH MODAL RESPONSE

As discussed before, the filter frequency response definition [cf. 2.13] is akin to the one employed in traditional time-domain processing. However, differently from classical Fourier processing, for some particular graph shift operators, the graph frequencies might not be unique, i.e., the multiplicity of particular eigenvalues might be greater than one. Therefore, there is no one-to-one map between the so-called graph frequencies and the graph modes as in such cases the *modes* are not defined uniquely, i.e., only the associate eigensubspace is uniquely defined.

Although it is always possible to restrict the discussion to graphs with a simple spectrum, we here introduce a different kind of (graph) filter response by considering fixed

graph modes, i.e., a fixed representation basis. That is, instead of focusing on a mapping $h : \lambda \mapsto h(\lambda)$ which takes a graph frequency λ and applies the function $h(\cdot)$, we consider a filter response, referred to as *the (graph) modal response* which defines the gain/attenuation that a particular graph mode, i.e., basis vector, experiences after undergoing a graph filter operation. We define this formally as follows.

Definition 2.1. (Graph Modal Response)

The modal response of a graph filter

$$\mathbf{H} = \mathbf{U} \text{diag}\{h_1, \dots, h_N\} \mathbf{U}^{-1}, \quad (2.25)$$

is defined as the function

$$h : [N] \rightarrow \mathbb{C}, \quad i \mapsto h_i,$$

where h_i is the gain/attenuation experienced by the i th graph mode.

We note that this definition is equivalent to the graph frequency response definition when the shift operator has a simple spectrum. However, this definition is closer in meaning to that of the time-domain as every mode (basis vector) can be scaled or shifted independently. Due to this slight difference, the notion of graph modal response is later used in this thesis to provide insights into the transformations implemented by generalized graph filters [cf. Chapter 3].

2.3. TWO CHALLENGES IN NETWORKED DATA PROCESSING

In this section, we briefly discuss two main challenges when processing networked data. In particular, we introduce the problems of (i) enhancing distributed processing over networks through a generalized family of graph filters, and (ii) network topology identification when data from a network process is available. These two problems form the main research thrusts of this thesis.

2.3.1. ENHANCED DISTRIBUTED PROCESSING

Although graph filters, one of the core tools in graph signal processing, enjoy a direct distributed implementation, their performance is often traded off with distributed communication and computational savings. Thus, in cases where a given linear operator \mathbf{H} has to be implemented or approximated by a matrix polynomial as in (2.20) the filter order K might become large if high accuracy is required. As the computational complexity scales with K , implementing large-order graph filters incurs high costs, making them not attractive for many problem instances.

To improve this tradeoff, [7] introduced the NV graph filter [cf. (2.21)]. Despite the improved performance of NV graph filters compared to classical graph filters, they do not exploit all the flexibility available for distributed processing. Although NV graph filters provide different weights for different nodes in the network, the signals of the neighboring nodes of a particular node are weighted equally. Hence, there is still room to further improve the filtering capabilities by tuning the local aggregation of neighboring signals.

As a result, in this thesis, we present a generalization of the state-of-the-art distributed graph filters to filters where every node weighs the signal of its neighbors with different

values while keeping the aggregation operation linear. This new implementation, referred to as edge-variant graph filtering, yields a significant reduction in terms of communication rounds while preserving the approximation accuracy. We also formally characterize a subset of shift-invariant graph filters that can be described with edge-variant recursions and provide new insights into the approximation of linear operators. Moreover, the proposed generalization addresses a broader class of operators $\tilde{\mathbf{H}}$ that do not necessarily share the eigenvectors with \mathbf{S} , such as analog network coding [7] as well as other applications that we will detail later on in this thesis.

Beyond expanding the capabilities of graph filters to implement linear operators, in this thesis, we also focus on practical aspects concerning their implementation; that is, we identify a lack of understanding of the asynchronous operation of graph filters and of the numerical challenges that might arise when designing graph filters with large orders. Thus, this thesis studies the asynchronous execution of generalized graph filters and provides conditions that guarantee the implementability of such filters in networks without synchronized units. Further, we also present a cascaded implementation of graph filters which allows us to deal with the numerical challenges that occur during the filter design and implementation stage. We analyze the properties of this kind of structures and provide theoretical insights as well as algorithms for the design. The results obtained from studying these practical aspects enhance the distributed capabilities of graph filters and expand the state of the art within GSP.

2

2.3.2. NETWORK TOPOLOGY IDENTIFICATION FROM NETWORK PROCESSES

As modern signal processing techniques take into account the network structure to provide signal estimators [11–13], filters [2, 9, 14, 15], or detectors [16–18], appropriate knowledge of the interconnections of the network is required. In many instances, the knowledge of the network structure is given and can be used to enhance traditional signal processing algorithms. However, in other cases, the network information is unknown and needs to be estimated. Thus, retrieving the topology of the network has become a topic of extensive research [19–26].

Despite the extensive research done so far (for a comprehensive review the reader is referred to [20, 27] and references therein), most of the approaches do not lever a physical model beyond the one induced by *graph filters* [28] drawn from graph signal processing (GSP) [2, 4, 29]. Among the ones that propose a different interaction model e.g., [22, 26], none of them considers the network data (a.k.a. graph signals) as states of an underlying process nor that the inputs and the states of the network may evolve on different underlying networks. However, several physical systems of practical interest can be modeled through a state-space formulation with (probably) known dynamics, i.e., brain activity diffusion, finite element models, circuit/flow systems. For these processes, we thus need a more general approach to find the underlying connections. As a result, we focus on the problem of *retrieving the underlying network structure*, from input-output signals of a process that can be modeled through a *deterministic linear dynamical system* whose system matrices depend on the interconnections of the network.

2.4. CHAPTER SUMMARY

This chapter introduced the fundamentals of graph signal processing. We first discussed how a graph can represent a network and how this domain can define signals, i.e., graph signals. Later, we presented the concept of shifts on graphs and linked this concept with the notion of graph frequencies. We built the idea of graph frequencies on the Laplace operator of continuous and regular domains and its invariances. Further, using variational arguments, smoothness over networks was defined as the result of a quadratic form involving the Laplacian of the network representation. Using traditional concepts of time-domain and frequency-domain filtering, we presented the concept of filtering in irregular domains. Further, we discussed the different possible constructions for graph filters and their relation with traditional signal processing tools. We then introduced the two challenges in networked data processing that comprise the two research thrusts of this thesis. We showcased the shortcomings of graph filters and its variants in terms of degrees of freedom and communication complexity. Based on this, we briefly introduced the first contribution of this thesis which will be addressed in Chapters 3 - 5. Further, we discussed the problem of network topology identification and presented our second thesis contribution which will be covered in Chapters 6 and 7. Finally, we encourage the interested reader to consult [2, 4] for a more detailed treatment of graph signal processing and an overview of open challenges within the field.

REFERENCES

- [1] M. H. Hayes, *Statistical digital signal processing and modeling* (John Wiley & Sons, 2009).
- [2] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, *The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains*, *IEEE Sig. Proc. Mag.* **30**, 83 (2013).
- [3] J. Mei and J. M. Moura, *Signal processing on graphs: Causal modeling of unstructured data*, *IEEE Trans. Signal Process* **65**, 2077 (2017).
- [4] A. Ortega, P. Frossard, J. Kovačević, J. M. Moura, and P. Vandergheynst, *Graph signal processing: Overview, challenges, and applications*, *Proceedings of the IEEE* **106**, 808 (2018).
- [5] D. H. Sattinger and O. L. Weaver, *Lie groups and algebras with applications to physics, geometry, and mechanics*, Vol. 61 (Springer Science & Business Media, 2013).
- [6] D. I. Shuman, P. Vandergheynst, and P. Frossard, *Distributed signal processing via chebyshev polynomial approximation*, arXiv preprint arXiv:1111.5239 (2011).
- [7] S. Segarra, A. Marques, and A. Ribeiro, *Optimal graph-filter design and applications to distributed linear network operators*, *IEEE Trans. Signal Process* (2017).
- [8] A. Loukas, A. Simonetto, and G. Leus, *Distributed autoregressive moving average graph filters*, *IEEE Sig. Proc. Lett.* **22**, 1931 (2015).
- [9] E. Isufi, A. Loukas, A. Simonetto, and G. Leus, *Autoregressive moving average graph filtering*, *IEEE Trans. Signal Process* **65**, 274 (2017).
- [10] E. Isufi and G. Leus, *Distributed sparsified graph filters for denoising and diffusion*

- tasks, in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)* (IEEE, 2017) pp. 5865–5869.
- [11] S. K. Narang, A. Gadde, and A. Ortega, *Signal processing techniques for interpolation in graph structured data*, in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)* (IEEE, 2013) pp. 5445–5449.
- [12] P. Di Lorenzo, P. Banelli, E. Isufi, S. Barbarossa, and G. Leus, *Adaptive graph signal processing: Algorithms and optimal sampling strategies*, arXiv preprint arXiv:1709.03726 (2017).
- [13] S. Chen, R. Varma, A. Singh, and J. Kovačević, *Signal recovery on graphs: Fundamental limits of sampling strategies*, *IEEE Trans. Sig. and Inf. Proc. over Netw.* **2**, 539 (2016).
- [14] M. Coutino, E. Isufi, and G. Leus, *Distributed Edge-Variant Graph Filters*, in *Int. Workshop on Comp. Adv. in Multi-Sensor Adaptive Proc. (CAMSAP)* (IEEE, 2017).
- [15] M. Coutino, E. Isufi, and G. Leus, *Advances in distributed graph filtering*, arXiv preprint arXiv:1808.03004 (2018).
- [16] C. Hu, J. Sepulcre, K. A. Johnson, G. E. Fakhri, Y. M. Lu, and Q. Li, *Matched signal detection on graphs: Theory and application to brain imaging data classification*, *NeuroImage* **125**, 587 (2016).
- [17] S. P. Chepuri and G. Leus, *Subgraph detection using graph signals*, in *Proc. of the 50th Asilomar Conf. Sig., Syst. and Comp.* (2016) pp. 532–534.
- [18] E. Isufi, A. S. Mahabir, and G. Leus, *Blind graph topology change detection*, *Signal processing letters* **25**, 655 (2018).
- [19] V. Kalofolias, *How to learn a graph from smooth signals*, in *Artificial Intelligence and Statistics* (2016) pp. 920–929.
- [20] G. Mateos, S. Segarra, and A. G. Marques, *Inference of graph topology*, *Cooperative and Graph Signal Processing: Principles and Applications* (PM Djuric and C. Richard, eds.), Amsterdam, Netherlands: Elsevier (2018).
- [21] G. Karanikolas, G. B. Giannakis, K. Slavakis, and R. M. Leahy, *Multi-kernel based non-linear models for connectivity identification of brain networks*, in *Proc. of the IEEE Int. Conf. on Acoust. Speech and Sig. Process. (ICASSP)* (2016) pp. 6315–6319.
- [22] Y. Shen, B. Baingana, and G. B. Giannakis, *Kernel-based structural equation models for topology identification of directed networks*, *IEEE Trans. on Sig. Process.* **65**, 2503 (2017).
- [23] K.-S. Lu and A. Ortega, *Closed form solutions of combinatorial graph laplacian estimation under acyclic topology constraints*, arXiv preprint arXiv:1711.00213 (2017).
- [24] J. Zhou and J.-a. Lu, *Topology identification of weighted complex dynamical networks*, *Physica A: Statistical Mechanics and Its Applications* **386**, 481 (2007).
- [25] Y. Shen, B. Baingana, and G. B. Giannakis, *Tensor decompositions for identifying directed graph topologies and tracking dynamic networks*, *IEEE Trans. on Sig. Process.* **65**, 3675 (2017).
- [26] R. Shafipour, S. Segarra, A. G. Marques, and G. Mateos, *Network topology inference from non-stationary graph signals*, in *Proc. of the IEEE Int. Conf. on Acoust. Speech and Sig. Process. (ICASSP)* (2017) pp. 5870–5874.
- [27] G. B. Giannakis, Y. Shen, and G. V. Karanikolas, *Topology identification and learning*

- over graphs: Accounting for nonlinearities and dynamics*, Proceedings of the IEEE **106**, 787 (2018).
- [28] A. G. Marques, S. Segarra, G. Leus, and A. Ribeiro, *Stationary graph processes and spectral estimation*, *IEEE Trans. Signal Process.* (2017).
- [29] A. Sandryhaila and J. M. Moura, *Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure*, *IEEE Sig. Proc. Mag.* **31**, 80 (2014).

3

ADVANCES IN GRAPH FILTERING

*As the nights, there are no
two quotes alike.*

Alberto Minguez

Graph filters are one of the core tools in graph signal processing. A central aspect of them is their direct distributed implementation. However, the filtering performance is often traded with distributed communication and computational savings. To improve this tradeoff, this chapter provides a generalization of state-of-the-art distributed graph filters to filters where every node weights the signal of its neighbors with different values while keeping the aggregation operation linear. This new implementation, labeled as edge-variant graph filter, yields a significant reduction in terms of communication rounds while preserving the approximation accuracy. In addition, we characterize a subset of shift-invariant graph filters that can be described with edge-variant recursions. By using a low-dimensional parametrization, these shift-invariant filters provide new insights in approximating linear graph spectral operators through the succession and composition of local operators, i.e., fixed support matrices.

3.1. INTRODUCTION

Filtering is one of the core operations in signal processing. The necessity to process large amounts of data defined over non-traditional domains, characterized by a graph, triggers advanced signal processing of the complex data relations embedded in that graph. Examples of the latter include biological, social, and transportation network data. The

Parts of this chapter have been published in the Proceedings of the *IEEE* International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (2018) [1] and in the *IEEE* Transactions on Signal Processing (2019) [2]

field of graph signal processing (GSP) [3–5] has been established to incorporate the underlying structure in the processing techniques. Through a formal definition of the graph Fourier transform (GFT), harmonic analysis tools employed for filtering in traditional signal processing have been adapted to deal with signals defined over graphs [6–12]. Similarly to time-domain filtering, graph filters manipulate the signal by selectively amplifying/attenuating its graph Fourier coefficients. Graph filters have seen use in applications such as signal analysis [13, 14], classification [15, 16], reconstruction [8, 17, 18], denoising [9, 19–21] and clustering [22]. Furthermore, they are the central block in graph filter banks [23, 24], wavelets [25], and convolutional neural networks [26, 27].

Distributed implementations of graph filters emerged as a way to deal with the ubiquity of big data applications and to improve the scalability of computation. By allowing nodes to exchange only local information, finite impulse response (FIR) [5, 7, 10] and infinite impulse response (IIR) [11, 12, 28] architectures have been devised to implement a variety of responses. However, being inspired by time-domain filters, the above implementations do not fully exploit the structure in the graph data. The successive signal aggregations are locally weighted with similar weights. This procedure often leads to high orders when approximating the desired response. To overcome this challenge, this chapter proposes a generalization of the distributed graph filtering concept by applying edge-based weights to the information coming from different neighbors. While the detailed contributions are provided in Section 3.1.2, let us here highlight that the above twist yields graph filters that are flexible enough to capture complex responses with much lower complexity.

3.1.1. RELATED WORKS

Driven by the practical need to implement a linear function with only a few local operations, the works in [10, 29] propose to ease the communication and computational cost of graph filters (GF).

In [10], polynomial graph filters (i.e., the FIR structure) are extended to graph filters with node-dependent weights. This architecture, which we refer to as a node-variant (NV) FIR graph filter, assigns different weights to different nodes and yields the same implementation as the classical FIR graph filter [5, 7]. The NV FIR filter addresses a broader family of linear operators that goes beyond the class of shift-invariant graph filters. However, the NV FIR filter still uses the same weight for all signals arriving at a particular node, ignoring the affinity between neighbors. As we show next, this limits the ability of the NV FIR filter to approximate the desired linear operator with very low orders.

The work in [29] introduces stochastic sparsification to reduce the cost of a distributed implementation. This method considers random edge sampling in each aggregation step to implement the filter output with a lower complexity by sacrificing the accuracy of the graph filter. Although conceptually similar to the material presented here, the filter following [29] is stochastic, hence its performance guarantees only hold in expectation. Moreover, since this approach applies only to shift-invariant filters, such as the FIR [5, 7] and the IIR [11, 12], it cannot address linear operators that are not shift invariant.

Another related problem, which can be interpreted through graph filtering, is the multilayer sparse approximation of matrices [30]. Different from the previous two approaches, here, a dense linear transform (matrix) is approximated through a sequence of sparse ma-

trix multiplications to obtain a computational speedup. While this framework can be considered as a series of diffusion steps over a network, the support of such sparse matrices differs in each iteration. This procedure, in practice, can be a limitation since it often requires information from non-adjacent nodes within an iteration. Finally, in [31], the problem of optimal subspace projection by local interactions is studied. This paper designs the weights of a network to achieve the fastest convergence to this kind of linear projection operators. However, this method does not address the more general GSP setup of interest: the implementation of graph filters or more general linear operators.

3.1.2. CHAPTER CONTRIBUTIONS

The main contribution of this chapter is the extension of the state-of-the-art graph filters to edge-variant graph filters. Due to the increased degrees of freedom (DoF), these filters allow for a complexity reduction of the distributed implementation while maintaining the approximation accuracy of current approaches. The salient points that broaden the existing literature are listed below.

- We present edge-variant architectures to implement FIR and IIR graph filtering. This framework extends the state-of-the-art graph filters by finding optimal edge weights to perform local aggregation of data. These new edge weights can differ per communication round and are generally distinct from the original weights of the graph matrix, thereby effectively reducing the number of exchanges. As for classical graph filter implementations, only local exchanges are required in each communication round, thus yielding an efficient distributed implementation. Three forms are analyzed: First, the general class of linear edge-variant FIR filters is presented and its distributed implementation is discussed. Then, the constrained edge-variant FIR graph filter is introduced, which maintains a similar distributed implementation as the general form, yet allowing a simple least-squares design. Finally, the family of edge-variant autoregressive moving average graph filters of order one (ARMA₁) is treated. This new distributed IIR architecture allows for a better trade-off between approximation accuracy and convergence rate compared to current IIR approaches.
- Through the definition of the filter modal response, we give a Fourier interpretation to a particular family of edge-variant graph filters. This subfamily shows a shift-invariant nature and links the filtering operation with a scaling of the graph modes (e.g., the graph shift eigenvectors). Further, we demonstrate the connection between the constrained edge-variant graph filters and per-tone filtering in traditional time-domain systems.
- Considering the general problem of approximating any linear operator through local operations, we demonstrate the applicability of our methods to problems beyond graph signal processing. Besides outperforming state-of-the-art graph filters in GSP tasks, e.g., approximating a user-provided frequency response, distributed consensus, and Tikhonov denoising; we present two new applications that could be addressed distributively with the proposed edge-variant graph filters: a distributed solution to an inverse problem and distributed beamforming.

3.1.3. CHAPTER OUTLINE

This chapter is organized as follows. Section 3.2 generalizes the FIR graph filters to their edge-variant version. Here, we also introduce the shift-invariant edge-variant graph filter and characterize its graph modal response. Section 3.3 analyzes a particular subfamily of edge-variant FIR graph filters, which enjoys a similar distributed implementation and a least-squares design strategy. In Section 3.4, we generalize the idea of edge-variant filtering to the class of IIR graph filters. Section 3.5 corroborates our findings with numerical results and Section 7.7 discusses some concluding remarks.

3.2. EDGE-VARIANT GRAPH FILTERS

Let us assume a scenario in which each node *trusts* differently the information coming from different neighbors, e.g., a person is likely to value more the opinion of his/her partner than that of a colleague. So, it is reasonable to treat this case as a graph filter, where each node weights differently the information of its neighbors.

Here, we formalize the above intuition in terms of edge-variant (EV) FIR graph filters. First, we introduce the general form of these filters and then in Section 3.2.2 we focus on the class of shift-invariant edge-variant (SIEV) FIR graph filters. The filter design strategy is discussed in Section 3.2.3.

3.2.1. GENERAL FORM

Consider an extension of the above edge-dependent fusion to several diffusion steps (signal shifts) where in each shift a different set of weights is used. At the k th diffusion, node v_i weights its neighbouring node v_l with the weight $\phi_{i,l}^{(k)}$. Hence, in each shift $k \in [K]$, and for each node $v_i \in \mathcal{V}$, there is a set of coefficients $\{\phi_{i,l}^{(k)}\}$ for $l \in \mathcal{N}_{v_i}$. Here, \mathcal{N}_{v_i} denotes the set of nodes adjacent to v_i as well as the node v_i itself, and K is the number of shifts. Mathematically, the above behavior can be written through an order- K *general EV* FIR graph filter defined as

$$\mathbf{H}_{\text{ev}} \triangleq \Phi_1 + \Phi_2 \Phi_1 + \dots + \Phi_K \Phi_{K-1} \cdots \Phi_1 = \sum_{k=1}^K \Phi_{k:1}, \quad (3.1)$$

where $\Phi_{k:1} = \Phi_k \Phi_{k-1} \cdots \Phi_1$ and $\Phi_j \in \mathbb{C}^{N \times N}$ is an edge-weighting matrix constructed from the coefficient set $\{\phi_{i,l}^{(j)}\}$, more specifically $[\Phi_j]_{i,l} = \phi_{i,l}^{(j)}$.

The fact that $l \in \mathcal{N}_{v_i}$ can be formalized by the following assumption.

Assumption 3.1. *Each Φ_j , $j \in [K]$ shares the support with $\mathbf{S} + \mathbf{I}$.*

Notice that this assumption possibly allows each node to use also its own information when \mathbf{S} has zero entries on its diagonal, e.g., when $\mathbf{S} = \mathbf{W}$. Note that definition (3.1) does not impose any symmetry on the coefficient matrices Φ_j . In fact, depending on how adjacent nodes trust each other, the applied weights can be different. From this point on, Assumption 3.1 will extend throughout the chapter.

The above filter description can also be interpreted differently through time-varying shift operators [32, 33], where Φ_j is the weighted, possibly directed shift operator for the

j th diffusion step with the same support as $\mathbf{S} + \mathbf{I}$. Therefore, the general EV FIR filter accounts for signals that are generated through time-varying systems in directed subgraphs of the original graph. In this interpretation, the filter coefficient matrix only allows for *edge deletion* or a re-weighting of graph flows.

Note that recursion (3.1) is a distributed graph filter. For computing the output $\mathbf{y} = \mathbf{H}_{\text{ev}}\mathbf{x}$, each node is only required to track the following quantities:

- the shifted signal output $\mathbf{x}^{(k)} = \Phi_k \mathbf{x}^{(k-1)}$, $\mathbf{x}^{(0)} = \mathbf{x}$,
- the accumulator output $\mathbf{y}^{(k)} = \mathbf{y}^{(k-1)} + \mathbf{x}^{(k)}$, $\mathbf{y}^{(0)} = \mathbf{0}$.

From the locality of Φ_k , the output $\mathbf{x}^{(k)}$ can be computed locally in each node by combining only neighboring data. Hence, (3.1) preserves the efficient distributed implementation of the classical FIR graph filter (2.20). The final filter output is $\mathbf{y} = \mathbf{y}^{(K)}$ which yields a complexity of $\mathcal{O}(MK)$. This stems from the fact that at every step, similarly to the classical FIR, only a single sparse matrix - vector multiplication is performed with complexity $\mathcal{O}(M)$. Hence, the complexity scales linearly in K for a fixed number of edges M .

Before addressing the design strategy of the filter (3.1), in the sequel, we introduce a particular structure of EV FIR graph filters that enjoys a graph Fourier domain interpretation.

3

3.2.2. SHIFT-INVARIANT FORM

An important family of graph filters is that of *shift-invariant graph filters* [cf. (2.16)], i.e., filters that commute with the graph shift operator \mathbf{S} . That is, given the shift \mathbf{S} and the filter matrix \mathbf{H} , the following holds

$$\mathbf{S}\mathbf{H} = \mathbf{H}\mathbf{S}. \quad (3.2)$$

For a non-defective shift operator \mathbf{S} and filter \mathbf{H} , i.e., the matrices accept an eigenvalue decomposition, condition (3.2) is equivalent to saying that the matrices \mathbf{S} and \mathbf{H} are jointly diagonalizable, or that their eigenbases coincide.

There is no reason to believe that graph filters of the form (9) are shift invariant. However, it is possible to characterize a subset of edge-variant graph filters that satisfy this property. To do so, we consider Assumption 3.1 and the following assumption on the coefficient matrices of the EV FIR graph filter:

Assumption 3.2. *Each Φ_j , $j \in [K]$ is diagonalizable with the eigenbasis of \mathbf{S} .*

Given the above assumption holds, we can rewrite (3.1) as

$$\mathbf{H}_{\text{ev}} = \sum_{k=1}^K \Phi_{k:1} = \mathbf{U} \left[\sum_{k=1}^K \prod_{j=1}^k \Lambda_j \right] \mathbf{U}^{-1}, \quad (3.3)$$

where we substituted $\Phi_j = \mathbf{U}\Lambda_j\mathbf{U}^{-1}$. To provide a closed-form expression for the effect of such graph filters on the graph modes, we first describe the set of *fixed-support matrices that are diagonalizable with a particular eigenbasis* (i.e., matrices that meet Assumptions 3.1 and 3.2). Mathematically, this set is defined as

$$\mathcal{J}_{\mathbf{U}}^{\mathcal{A}} = \{\mathbf{A} : \mathbf{A} = \mathbf{U}\mathbf{\Omega}\mathbf{U}^{-1}, [\text{vec}(\mathbf{A})]_i = 0, \forall i \in \mathcal{A}\}, \quad (3.4)$$

where \mathcal{A} is the index set defining the zero entries of $\mathbf{S} + \mathbf{I}$ and $\mathbf{\Omega}$ is diagonal. The fixed-support condition in $\mathcal{G}_{\mathbf{U}}^{\mathcal{A}}$ can be expressed in the linear system form

$$\mathbf{\Phi}_{\mathcal{A}} \text{vec}(\mathbf{A}) = \mathbf{0}, \quad (3.5)$$

with $\mathbf{\Phi}_{\mathcal{A}} \in \{0, 1\}^{|\mathcal{A}| \times N^2}$ denoting the selection matrix whose rows are the rows of an $N^2 \times N^2$ identity matrix indexed by the set \mathcal{A} . By leveraging the vectorization operation properties and the knowledge of the eigenbasis of \mathbf{A} , we can rewrite (3.5) as

$$\mathbf{\Phi}_{\mathcal{A}} \text{vec}(\mathbf{A}) = \mathbf{\Phi}_{\mathcal{A}} (\mathbf{U}^{-\text{T}} * \mathbf{U}) \boldsymbol{\omega} = \mathbf{0}, \quad (3.6)$$

where “*” represents the Kathri-Rao product and $\boldsymbol{\omega} = [[\mathbf{\Omega}]_{1,1}, [\mathbf{\Omega}]_{2,2}, \dots, [\mathbf{\Omega}]_{N,N}]^{\text{T}}$ is the vector containing the eigenvalues of \mathbf{A} . From (3.6), we see that $\boldsymbol{\omega}$ characterizes the intersection of the nullspace of $\mathbf{\Phi}_{\mathcal{A}}$ and the range of $\mathbf{U}^{-\text{T}} * \mathbf{U}$. More formally,

$$\boldsymbol{\omega} \in \text{null}\{\mathbf{T}_{\mathbf{U}}^{\mathcal{A}}\}, \quad (3.7)$$

with $\mathbf{T}_{\mathbf{U}}^{\mathcal{A}} = \mathbf{\Phi}_{\mathcal{A}} (\mathbf{U}^{-\text{T}} * \mathbf{U})$.

Let us denote $d = \dim(\text{null}\{\mathbf{T}_{\mathbf{U}}^{\mathcal{A}}\})$. With this in place, the following proposition characterizes the matrices that belong to the set $\mathcal{G}_{\mathbf{U}}^{\mathcal{A}}$.

Proposition 3.1. (Graph shift nullspace property) *Given a basis \mathbf{U} and a sparsity pattern defined by the set \mathcal{A} , the matrices within the set $\mathcal{G}_{\mathbf{U}}^{\mathcal{A}}$ are of the form $\mathbf{A} = \mathbf{U}\mathbf{\Omega}\mathbf{U}^{-1}$ and have eigenvalues given by*

$$\mathbf{\Omega} = \text{diag}(\mathbf{B}_{\mathbf{U}}^{\mathcal{A}} \boldsymbol{\alpha}), \quad (3.8)$$

where the $N \times d$ matrix $\mathbf{B}_{\mathbf{U}}^{\mathcal{A}}$ is a basis for the nullspace of $\mathbf{T}_{\mathbf{U}}^{\mathcal{A}}$, i.e.,

$$\text{span}\{\mathbf{B}_{\mathbf{U}}^{\mathcal{A}}\} = \text{null}\{\mathbf{T}_{\mathbf{U}}^{\mathcal{A}}\},$$

and $\boldsymbol{\alpha}$ is the basis expansion coefficient vector.

Proof. The proof follows from (3.6)-(3.7). \square

The above result has been used for assessing the uniqueness of the graph shift operator in topology identification [34]. Here, we leverage Proposition 3.1 for interpreting the response of a particular class of SIEV FIR filters. Specifically, under Assumptions 3.1 and 3.2 we can express each $\mathbf{\Phi}_j$ of (3.1) as

$$\mathbf{\Phi}_j = \mathbf{U} \text{diag}(\mathbf{B}_{\mathbf{U}}^{\mathcal{A}} \boldsymbol{\alpha}_j) \mathbf{U}^{-1}, \quad (3.9)$$

and write our SIEV FIR filter as

$$\mathbf{H}_{\text{siev}} = \mathbf{U} \left[\sum_{k=1}^K \prod_{j=1}^k \text{diag}(\mathbf{B}_{\mathbf{U}}^{\mathcal{A}} \boldsymbol{\alpha}_j) \right] \mathbf{U}^{-1}. \quad (3.10)$$

The following proposition formally characterizes the frequency interpretation of such filters in terms of the modal response.

Proposition 3.2. (Modal Response of SIEV FIR) *An FIR graph filter of the form (3.1) satisfying Assumptions 3.1 and 3.2 has i th modal response*

$$h_i = \sum_{k=1}^K \prod_{j=1}^k (\mathbf{b}_{\mathbf{u}_i}^{(k)})^\top \boldsymbol{\alpha}_j \quad (3.11)$$

where $(\mathbf{b}_{\mathbf{u}_i}^{(k)})^\top$ is the i th row of $\mathbf{B}_{\mathbf{u}}^{(k)}$.

Proof. The proof follows directly from (3.10). \square

An interesting outcome of Proposition 3.2 is that the filter response is independent of the graph frequencies. This is clear from (3.11), where we see that the eigenvalue λ_i does not appear in the expression of h_i . Therefore, we can interpret SIEV FIR graph filters as *eigenvector filters*, since they act on the eigenmodes of the graph. That is, for each graph eigenmode (eigenvector) \mathbf{u}_i , \mathbf{H}_{siev} might apply a different gain given by (3.11) (independent of λ_i) to the component of the input signal \mathbf{x} in the direction of \mathbf{u}_i . This is in contrast to classical FIR graph filters which apply the same polynomial expression in λ_i to all modes $\{\mathbf{u}_i\}_{i \in [N]}$. As a result, the classical FIR graph filter will always filter different graph modes with the same graph frequency in the same way. The following section introduces methods for designing EV FIR graph filters in the node domain and SIEV FIR graph filters using the parametrization in (3.11).

3.2.3. FILTER DESIGN

General form. Given a desired operator $\tilde{\mathbf{H}}$, we design an EV FIR filter \mathbf{H}_{ev} [cf. (3.1)] that approximates $\tilde{\mathbf{H}}$ as the solution of the optimization problem

$$\begin{aligned} \underset{\{\Phi_k\}}{\text{minimize}} \quad & \|\tilde{\mathbf{H}} - \sum_{k=1}^K \Phi_{k:1}\| \\ \text{subject to} \quad & \text{supp}\{\Phi_k\} = \text{supp}\{\mathbf{S} + \mathbf{I}\}, \forall k \in [K], \end{aligned} \quad (3.12)$$

where $\|\cdot\|$ is an appropriate distance measure, e.g., the Frobenius norm ($\|\cdot\|_F$), or the spectral norm ($\|\cdot\|_2$).

Unfortunately, (3.12) is a high-dimensional nonconvex problem and hard to optimize. An approach to finding a local solution for it is through block coordinate descent methods, which provide local convergence guarantees when applied to such problems [35]. In fact, the cost in (3.12) is a *block multi-convex* function, i.e., the cost function is a convex function of Φ_i with all the other variables fixed.

Starting then with an initial set of matrices $\{\Phi_j^{(0)}\}_{j \in [K]}$ (potentially initialized with an order- K classical FIR filter), we solve a sequence of optimization problems where at the i th step, the matrix Φ_i is found. That is, at the i th iteration, we fix the matrices $\{\Phi_j^{(0)}\}_{j \in [K] \setminus \{i\}}$ and solve the convex problem

$$\begin{aligned} \underset{\Phi_i}{\text{minimize}} \quad & \|\tilde{\mathbf{H}} - \sum_{k=1}^K \Phi_{k:(i+1)}^{(0)} \Phi_i \Phi_{(i-1):1}^{(0)}\| \\ \text{subject to} \quad & \text{supp}\{\Phi_i\} = \text{supp}\{\mathbf{S} + \mathbf{I}\}, \end{aligned} \quad (3.13)$$

where $\Phi_{a:b}^{(0)} = \Phi_a^{(0)} \Phi_{a-1}^{(0)} \dots \Phi_{b+1}^{(0)} \Phi_b^{(0)}$ for $a \geq b$ and $\Phi_{a:b}^{(0)} = \mathbf{I}$, otherwise. Then, the matrix $\Phi_i^{(0)}$ is updated with its solution and the procedure is repeated for all $\{\Phi_j\}_{j \in [K]}$. If the final fitting error is large, the whole process can be repeated until the desired performance is reached, or until a local minimum is found.

Although filter (3.1) is the most general EV FIR filter form, the non-convexity encountered in the above design strategy may often lead to a local solution with an unacceptable performance. To tackle such issue, in Section 3.3, we introduce a constrained EV FIR filter which provides a higher flexibility than the state-of-the-art graph filters while accepting a simple least-squares design.

SIEV form. Besides enjoying the modal response interpretation, the SIEV FIR filter also has a simpler design than the general form (3.1). For $\{\tilde{h}_i\}_{i=1}^N$ being the desired graph modal response¹, the SIEV FIR filter design consists of solving the optimization problem

$$\underset{\{\alpha_j\}}{\text{minimize}} \quad \sum_{i=1}^N \left\| \tilde{h}_i - \sum_{k=1}^K \prod_{j=1}^k (\mathbf{b}_{\mathbf{U},i}^{\mathcal{A}})^T \alpha_j \right\|_2^2. \quad (3.14)$$

Similarly to (3.12), problem (3.14) is nonconvex and cannot in general be solved to global optimality with standard convex optimization methods. However, (3.14) is also a block multi-convex function in each α_i , $i \in [K]$ and, therefore, the block coordinate descent methods [35] can again be employed to find a local minimum, yet the number of unknowns is smaller than for the general EV form. Alternatively, the straightforward analytical expression of the gradient of the cost function allows the use of off-the-shelf solvers for global optimization, such as MATLAB's built-in `fmincon` function [36].

3.3. CONSTRAINED EDGE-VARIANT GRAPH FILTERS

To overcome the design issues of the general EV FIR filter, here we present a constrained version of it that retains both the distributed implementation and the edge-dependent weighting. This reduction of the DoF will, in fact, allow us to design the filter coefficients in a least-squares fashion. The structure of these filters along with their distributed implementation is presented in the next section. In Section 3.3.2 we provide a modal response interpretation of these filters, and finally in Section 3.3.3 we present the design strategy.

3.3.1. GENERAL FORM

The *constrained* EV (CEV) FIR graph filter is defined as

$$\mathbf{H}_{\text{cev}} = \Phi_1 + \Phi_2 \mathbf{S} + \dots + \Phi_K \mathbf{S}^{K-1} \triangleq \sum_{k=1}^K \Phi_k \mathbf{S}^{k-1}, \quad (3.15)$$

where the edge-weighting matrices $\{\Phi_k\}_{k \in [K]}$ share again the support with $\mathbf{S} + \mathbf{I}$. These filters enjoy the same distributed implementation of the general form (3.1). In fact, each node can compute locally the filter output by tracking the following quantities:

- the regular shift output $\mathbf{x}^{(k)} = \mathbf{S}\mathbf{x}^{(k-1)}$, $\mathbf{x}^{(0)} = \mathbf{x}$,

¹This can be for instance a low-pass form if we want to keep only the eigenvector contributions associated with the slowly varying graph modes.

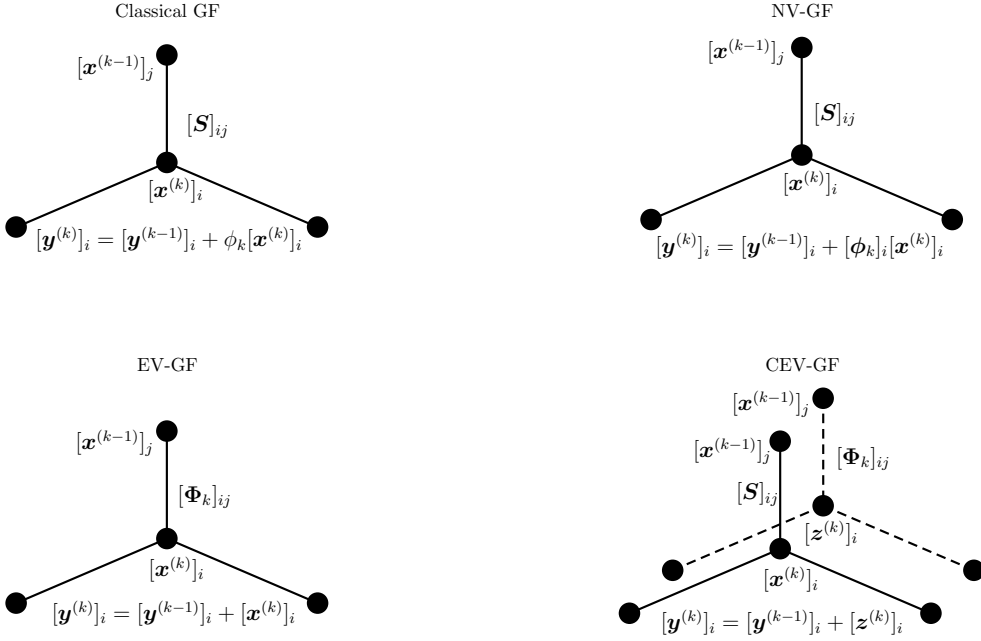


Figure 3.1: Illustration of the required communication, scaling, and recursion performed by the different graph filters.

- the weighted shift output $\mathbf{z}^{(k)} = \Phi_k \mathbf{x}^{(k-1)}$,
- the accumulator output $\mathbf{y}^{(k)} = \mathbf{y}^{(k-1)} + \mathbf{z}^{(k)}$, $\mathbf{y}^{(0)} = \mathbf{0}$.

From the locality of \mathbf{S} and Φ_k , both $\mathbf{x}^{(k)}$ and $\mathbf{z}^{(k)}$ require only neighboring information. The final filter output is $\mathbf{y} = \mathbf{y}^{(K)}$ and is obtained with the same computational complexity of $\mathcal{O}(MK)$. Figure 3.1 visually illustrates the differences between the processing steps for the different graph filters analyzed so far.

Note that the construction (3.15) still applies different weights to the signal coming from different edges. However, instead of adopting a different *diffusion* matrix at every step, the signal diffusion occurs through the graph shift \mathbf{S} . The additional extra step mixes locally $\mathbf{x}^{(k-1)}$ using edge-dependent weights, which are allowed to vary for each term k . We adopt the term *constrained* for this implementation since the first $k-1$ diffusion steps in every term k are constrained to be performed by the graph shift \mathbf{S} . Note though that the CEV FIR graph filter is *not* a special case of the EV FIR graph filter, see Fig. 3.2. We conclude this section with the following remark.

Remark 3.1. Note that the NV graph filter of order K from [10] [cf. (2.21)] is a particular case of the CEV graph filter of order K . The local matrices $\{\Phi_k\}_{k=1}^K$ are then in fact substituted as $\Phi_1 = \text{diag}(\phi_0) + \text{diag}(\phi_1)\mathbf{S}$ and $\Phi_k = \text{diag}(\phi_k)\mathbf{S}^k \forall k > 1$. Also, the classical graph filter of order K from [5–7] [cf. (2.20)] is a particular case of the CEV graph filter of order K . In that case, the local matrices $\{\Phi_k\}_{k=1}^K$ are just substituted as $\Phi_1 = \phi_0\mathbf{I} + \phi_1\mathbf{S}$ and $\Phi_k = \phi_k\mathbf{S}^k \forall k > 1$.

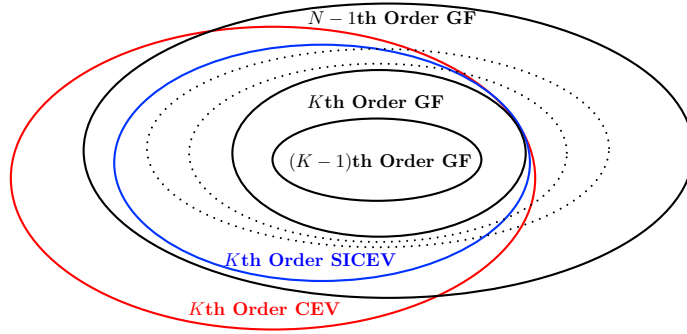


Figure 3.2: Relation between the classical and CEV FIR graph filters. This figure depicts the possibility of obtaining higher-order polynomial graph filters with reduced order CEV graph filters.

3

3.3.2. SHIFT-INVARIANT CONSTRAINED EDGE-VARIANT GRAPH FILTERS

Following the same lines of Section 3.2.2, we can use the set $\mathcal{G}_{\mathbf{U}}^{\mathcal{A}}$ (3.4) to characterize the graph modal response of the CEV FIR graph filter when the matrices $\{\Phi_k\}_{k=1}^K$ satisfy Assumptions 3.1 and 3.2.

This subset of CEV FIR graph filters, which we refer to as shift-invariant CEV (SICEV) FIR graph filters, can again be expressed in terms of $\mathbf{B}_{\mathbf{U}}^{\mathcal{A}}$ and $\{\alpha_k\}_{k=0}^K$ as

$$\mathbf{H}_{\text{sicev}} = \mathbf{U} \left[\sum_{k=1}^K \text{diag}(\mathbf{B}_{\mathbf{U}}^{\mathcal{A}} \alpha_k \odot \boldsymbol{\lambda}^{\odot(k-1)}) \right] \mathbf{U}^{-1}, \quad (3.16)$$

where $\boldsymbol{\lambda}^{\odot k}$ denotes the k th element-wise power of the eigenvalue vector of the shift operator \mathbf{S} . The subsequent proposition formalizes the modal response of these filters.

Proposition 3.3. (Modal Response of SICEV FIR) *An FIR graph filter of the form (3.15) satisfying Assumptions 3.1 and 3.2 has i th modal response*

$$h_i = \sum_{k=1}^K \gamma_{ik} \lambda_i^{k-1}, \quad (3.17)$$

where $\gamma_{ik} = (\mathbf{b}_{\mathbf{U},i}^{\mathcal{A}})^T \alpha_k$ is the k th polynomial coefficient for the i th graph frequency and $(\mathbf{b}_{\mathbf{U},i}^{\mathcal{A}})^T$ is the i th row of $\mathbf{B}_{\mathbf{U}}^{\mathcal{A}}$.

Proof. The proof follows directly from (3.16). \square

From (3.17), we see that there is a substantial difference between the SICEV FIR graph filters and the more general SIEV FIR graph filters. Here, the modal response is a polynomial in the graph frequencies. This is similar to the classical FIR filter (2.20), but now each frequency has a different set of coefficients. In other words, the modal response of the SICEV FIR filter is a *mode-dependent polynomial*. For readers more familiar with traditional discrete-time processing, this behavior can be interpreted as applying different polynomial filters to each frequency bin (see e.g., [37]).

Remark 3.2. *The particular form of the SICEV FIR filter allows it to match all shift-invariant polynomial responses of order K as well as a subset of higher-order polynomials of order up to $N - 1$ which includes all higher-order responses. The latter property follows from the observation that any shift-invariant graph filter is a polynomial of the graph shift operator [5] and from the filter response in (3.17). In fact, the SICEV FIR filter is still a polynomial of the shift \mathbf{S} , though with a different polynomial response per graph frequency. This additional freedom extends the set of functions that can be approximated by a SICEV FIR filter of order K . Figure 1(b) further illustrates the relation among different graph filters.*

3.3.3. FILTER DESIGN

General form. Following a similar approach as in Section 3.2.3, we can approximate a desired operator $\tilde{\mathbf{H}}$ with a CEV FIR filter by solving the problem

$$\begin{aligned} & \underset{\{\Phi_k\}}{\text{minimize}} && \|\tilde{\mathbf{H}} - \sum_{k=1}^K \Phi_k \mathbf{S}^{k-1}\|_F^2 \\ & \text{subject to} && \text{supp}\{\Phi_k\} = \text{supp}\{\mathbf{S} + \mathbf{I}\}, \forall k \in [K]. \end{aligned} \quad (3.18)$$

Exploiting then the properties of the vectorization operator and the Frobenius norm, we can transform (3.18) into

$$\begin{aligned} & \underset{\{\phi_k\}}{\text{minimize}} && \|\tilde{\mathbf{h}} - \sum_{k=1}^K (\mathbf{S}^{k-1} \otimes \mathbf{I}) \phi_k\|_2 \\ & \text{subject to} && \text{supp}\{\Phi_k\} = \text{supp}\{\mathbf{S} + \mathbf{I}\}, \forall k \in [K], \end{aligned} \quad (3.19)$$

where $\tilde{\mathbf{h}} \triangleq \text{vec}(\tilde{\mathbf{H}})$ and $\phi_k \triangleq \text{vec}(\Phi_k)$.

Since the support of the weighting matrices is known, problem (3.19) can be written in the reduced-size form

$$\underset{\{\phi_k\}}{\text{minimize}} \quad \|\tilde{\mathbf{h}} - \Psi \boldsymbol{\theta}\|_2^2 \quad (3.20)$$

where $\Psi = [\mathbf{I} \check{\mathbf{S}} \cdots \check{\mathbf{S}}_K]$, $\boldsymbol{\theta} = [\check{\boldsymbol{\phi}}_0^\top \check{\boldsymbol{\phi}}_1^\top \cdots \check{\boldsymbol{\phi}}_K^\top]^\top$, $\check{\boldsymbol{\phi}}_k$ is the vector ϕ_k with the zero entries removed and $\check{\mathbf{S}}_k$ is the matrix $(\mathbf{S}^k \otimes \mathbf{I})$ with the appropriate columns removed. In addition, if a regularized solution is desired, a natural penalization term might be the convex ℓ_1 -norm which induces sparsity in the solution yielding only few active coefficients. Problem (3.19) has a unique solution as long as Ψ is full column rank, i.e., $\text{rank}(\Psi) = \text{nnz}(\mathbf{S}) \cdot K + N$, where $\text{nnz}(\cdot)$ denotes the number of nonzero matrix elements. Otherwise, regularization must be used to obtain a unique solution.

Remark 3.3. *Besides leading to a simple least-squares problem, the design of the CEV FIR filter can also be computed distributively. Given that each node knows the desired filter response and the graph shift operator (i.e., the network structure), it can be shown that by re-ordering the columns of Ψ and the entries of $\boldsymbol{\theta}$ the framework of splitting-over-features [38] can be employed for a decentralized estimation of $\boldsymbol{\theta}$.*

SICEV form. Similar to the more general CEV FIR filter, the design of $\{\boldsymbol{\alpha}_k\}_{k=1}^K$ for the SICEV form can be performed in a least-squares fashion. First, for a set of vectors $\{\boldsymbol{\alpha}_k\}_{k=1}^K$

the modal response of the SICEV FIR filter reads as

$$\mathbf{h}_\lambda = \sum_{k=1}^K [\mathbf{B}_\mathbf{U}^{\mathcal{A}} \boldsymbol{\alpha}_k \odot \lambda^{\odot(k-1)}], \quad (3.21)$$

where \mathbf{h}_λ is obtained by stacking the modal responses, i.e., $\{h_i\}_{i=1}^N$, in a column vector. By using the properties of the Hadamard product, (3.21) can be written as

$$\mathbf{h}_\lambda = \sum_{k=1}^K \text{diag}(\lambda^{\odot(k-1)}) \mathbf{B}_\mathbf{U}^{\mathcal{A}} \boldsymbol{\alpha}_k = \sum_{k=1}^K \mathbf{M}_k \boldsymbol{\alpha}_k, \quad (3.22)$$

with $\mathbf{M}_k = \text{diag}(\lambda^{\odot(k-1)}) \mathbf{B}_\mathbf{U}^{\mathcal{A}}$. Defining then $\mathbf{M} = [\mathbf{M}_1, \dots, \mathbf{M}_K]$ and $\boldsymbol{\alpha} = [\boldsymbol{\alpha}_1^\top, \dots, \boldsymbol{\alpha}_K^\top]^\top$, we obtain the linear relation

$$\mathbf{h}_\lambda = \mathbf{M}\boldsymbol{\alpha}. \quad (3.23)$$

Therefore, the approximation of a desired response $\tilde{\mathbf{h}}_\lambda = [\tilde{h}_1, \dots, \tilde{h}_N]^\top$ consists of solving the least-squares problem

$$\underset{\boldsymbol{\alpha} \in \mathbb{R}^{d(K+1)}}{\text{minimize}} \quad \|\tilde{\mathbf{h}}_\lambda - \mathbf{M}\boldsymbol{\alpha}\|_2, \quad (3.24)$$

which has a unique solution when \mathbf{M} is full column rank, i.e., $\text{rank}(\mathbf{M}) = d(K+1) \leq N$.

Remark 3.4. *An interesting result from (3.23) is the fact that perfect graph spectrum approximation can be achieved when $d(K+1) \geq N$. Therefore, given a certain graph and its associate shift operator, \mathbf{S} , by checking the dimension of the nullspace of $\mathbf{T}_\mathbf{U}^{\mathcal{A}}$, d , a filter order K can be chosen such that $d(K+1) \geq N$, guaranteeing that the desired graph frequency response can be perfectly approximated.*

3.4. EDGE-VARIANT IIR GRAPH FILTERS

We now extend the edge-variant filtering concept to the class of IIR graph filters. Here, we focus on the basic building block of these filters, i.e., the ARMA₁ recursion (2.23). We follow the same organization of the former two sections, by introducing the edge-variant ARMA₁ structure in Section 3.4.1, the shift-invariant version in Section 3.4.2, and the design strategies in Section 3.4.3.

3.4.1. EDGE-VARIANT ARMA₁

We build an edge-variant ARMA₁ (EVA₁) recursion on graphs by modifying (2.23) as

$$\mathbf{y}_t = \boldsymbol{\Phi}_1 \mathbf{y}_{t-1} + \boldsymbol{\Phi}_0 \mathbf{x}, \quad (3.25)$$

where $\boldsymbol{\Phi}_0$ and $\boldsymbol{\Phi}_1$ are the edge-weighting matrices having the support of $\mathbf{S} + \mathbf{I}$ that respectively weight locally the entries of \mathbf{y}_{t-1} and \mathbf{x} . Proceeding similarly as in [12], for $\|\boldsymbol{\Phi}_1\|_2 < 1$, the steady-state output of (3.25) is

$$\mathbf{y} = \lim_{t \rightarrow \infty} \mathbf{y}_t = (\mathbf{I} - \boldsymbol{\Phi}_1)^{-1} \boldsymbol{\Phi}_0 \mathbf{x} \triangleq \mathbf{H}_{\text{eva}_1} \mathbf{x}, \quad (3.26)$$

where we notice the inverse relation w.r.t. the edge-weighting matrix Φ_1 . Recursion (3.25) converges to (3.26) linearly with a rate governed by $\|\Phi_1\|_2$. The classical form (2.23) can be obtained by substituting $\Phi_1 = \psi\mathbf{S}$ and $\Phi_0 = \phi\mathbf{I}$.

The EVA₁ filter presents the same frequency interpretation challenges as the FIR filter counterpart. Therefore, we next analyze the shift-invariant version of it and we will see a rational modal response.

3.4.2. SHIFT-INVARIANT EV ARMA₁

By limiting the choices of $\{\Phi_0, \Phi_1\}$ to the ones that satisfy Assumptions 3.1 and 3.2, we obtain the shift-invariant edge-variant ARMA₁ (SIEVA₁) graph filter

$$\mathbf{H}_{\text{sieva}_1} = \mathbf{U}[(\mathbf{I} - \text{diag}(\mathbf{B}_{\mathbf{U}}^{\mathcal{A}} \boldsymbol{\alpha}_1))^{-1} \text{diag}(\mathbf{B}_{\mathbf{U}}^{\mathcal{A}} \boldsymbol{\alpha}_0)] \mathbf{U}^{-1}, \quad (3.27)$$

where $\boldsymbol{\alpha}_0$ and $\boldsymbol{\alpha}_1$ are the respective basis expansion vectors of Φ_0 and Φ_1 onto the nullspace of $\mathbf{T}_{\mathbf{U}}^{\mathcal{A}}$ (see Proposition 3.1). From (3.27), we see that the inverse relation that appears in (3.26) indeed appears as a function affecting the graph eigenmodes. The following proposition concludes this section by stating this finding in a formal way.

Proposition 3.4. (Modal Response of SIEVA₁) *An ARMA₁ graph filter of the form (3.26) satisfying Assumptions 3.1 and 3.2 for $K = 1$ has i th modal response*

$$h_i = \frac{(\mathbf{b}_{\mathbf{U},i}^{\mathcal{A}})^{\top} \boldsymbol{\alpha}_0}{1 - (\mathbf{b}_{\mathbf{U},i}^{\mathcal{A}})^{\top} \boldsymbol{\alpha}_1} \quad (3.28)$$

where $(\mathbf{b}_{\mathbf{U},i}^{\mathcal{A}})^{\top}$ is the i th row of the matrix $\mathbf{B}_{\mathbf{U}}^{\mathcal{A}}$.

Proof. The proof follows directly from (3.27). \square

3.4.3. FILTER DESIGN

In the following, different designs are presented for EV-ARMA₁ graph filters. We first introduce the design for the most general form in the node domain. Then we specialize the design for the case of the SIEVA₁ in the spectral domain to leverage the reduced dimensionality of the resulting optimization problem.

EVA₁ form. Here, we extend the design approach of [39] and design $\{\Phi_0, \Phi_1\}$ by using Prony's method. For $\tilde{\mathbf{H}}$ being the desired operator, we can define the fitting error matrix

$$\mathbf{E} = \tilde{\mathbf{H}} - (\mathbf{I} - \Phi_1)^{-1} \Phi_0, \quad (3.29)$$

which presents nonlinearities in the denominator coefficients, i.e., in Φ_1 . To tackle this issue, we consider the modified fitting error matrix

$$\mathbf{E}' = \tilde{\mathbf{H}} - \Phi_1 \tilde{\mathbf{H}} - \Phi_0, \quad (3.30)$$

which is obtained by multiplying both sides of (3.29) by $\mathbf{I} - \Phi_1$. This way, the filter design problem is transformed to the convex optimization problem

$$\begin{aligned} & \underset{\Phi_0, \Phi_1}{\text{minimize}} && \|\tilde{\mathbf{H}} - \Phi_1 \tilde{\mathbf{H}} - \Phi_0\| \\ & \text{subject to} && \|\Phi_1\|_2 < \delta, \quad \delta < 1, \\ & && \text{supp}\{\Phi_0\} = \text{supp}\{\Phi_1\} = \text{supp}\{\mathbf{S} + \mathbf{I}\}. \end{aligned} \quad (3.31)$$

The objective function in (3.31) aims at reducing the modified error E' , while the first constraint trades the convergence rate of (3.25) with approximation accuracy.

Note that it is possible to improve the performance of the filter design (3.31) if a second step is performed after the matrices $\{\Phi_0, \Phi_1\}$ have been obtained. Using the matrix Φ_1 obtained by solving (3.31), we can fit again Φ_0 by using the error expression in (3.29). That is, we can obtain a new matrix Φ_0 by solving

$$\begin{aligned} & \underset{\Phi_0}{\text{minimize}} && \|\tilde{\mathbf{H}} - (\mathbf{I} - \Phi_1)^{-1} \Phi_0\| \\ & \text{subject to} && \text{supp}\{\Phi_0\} = \text{supp}\{\mathbf{S} + \mathbf{I}\}, \end{aligned} \quad (3.32)$$

where Φ_1 is the coefficient matrix obtained from (3.31). This procedure is known as Shanks' method [40] and has also been adopted in [12, 39].

SIEVA₁ form. Following the same idea as in (3.29)-(3.31), the modified fitting error of a SIEVA₁ graph filter is

$$e'_i = \tilde{h}_i - \tilde{h}_i (\mathbf{b}_{\mathbf{U},i}^{\mathcal{A}})^T \boldsymbol{\alpha}_1 - (\mathbf{b}_{\mathbf{U},i}^{\mathcal{A}})^T \boldsymbol{\alpha}_0, \quad (3.33)$$

with \tilde{h}_i , $(\mathbf{b}_{\mathbf{U},i}^{\mathcal{A}})^T \boldsymbol{\alpha}_0$, and $(\mathbf{b}_{\mathbf{U},i}^{\mathcal{A}})^T \boldsymbol{\alpha}_1$ denoting respectively the desired modal response and the eigenvalues of Φ_0 and Φ_1 w.r.t. the i th mode. In vector form, (3.33) can be written as

$$\mathbf{e}' = \tilde{\mathbf{h}}_\lambda - \tilde{\mathbf{M}} \tilde{\boldsymbol{\alpha}}, \quad (3.34)$$

with $\mathbf{e}' = [e'_1, \dots, e'_N]^T$, $\tilde{\mathbf{h}}_\lambda = [\tilde{h}_1, \dots, \tilde{h}_N]^T$, $\tilde{\mathbf{M}} = [\mathbf{B}_{\mathbf{U}}^{\mathcal{A}}, \text{diag}(\tilde{\mathbf{h}}_\lambda) \mathbf{B}_{\mathbf{U}}^{\mathcal{A}}]$, and $\tilde{\boldsymbol{\alpha}} = [\boldsymbol{\alpha}_0^T \ \boldsymbol{\alpha}_1^T]^T$. Then, $\{\boldsymbol{\alpha}_0, \boldsymbol{\alpha}_1\}$ can be estimated as the solution of the constrained least-squares problem

$$\begin{aligned} & \underset{\boldsymbol{\alpha}_0, \boldsymbol{\alpha}_1 \in \mathbb{R}^d}{\text{minimize}} && \|\tilde{\mathbf{h}}_\lambda - \tilde{\mathbf{M}} \tilde{\boldsymbol{\alpha}}\|_2^2 \\ & \text{subject to} && \|\mathbf{B}_{\mathbf{U}}^{\mathcal{A}} \boldsymbol{\alpha}_1\|_\infty < \delta, \ \delta < 1, \ \boldsymbol{\alpha} = [\boldsymbol{\alpha}_0^T \ \boldsymbol{\alpha}_1^T]^T. \end{aligned} \quad (3.35)$$

As (3.31), problem (3.35) again aims at minimizing the modified fitting error, while tuning the convergence rate through δ . Here again, Shanks' method can be considered.

Differently from the general EVA₁, here the number of unknowns is reduced to $2d$, as now only the vectors $\boldsymbol{\alpha}_0$ and $\boldsymbol{\alpha}_1$ need to be designed. Due to this low dimensionality, one can also opt for global optimization solvers to find an acceptable local minimum of the true error (i.e., the equivalent of (3.29)).

Remark 3.5. *The approximation accuracy of the EVA₁ and SIEVA₁ filters can be further improved by following the iterative least-squares approach proposed in [41]. This method has shown to improve the approximation accuracy of Prony's design by not only taking the modified fitting error into account but also the true one. However, as this idea does not add much to this work, interested readers are redirected to [41] for more details.*

3.5. APPLICATIONS

We now present a set of numerical examples to corroborate the applicability of the proposed filters for several distributed tasks. Table I presents a summary of the different graph filters mentioned in this chapter along with their specifications. In our simulations, we made use of the GSP toolbox [42].

Filter Type	Expression	Shift-Invariant	Design Strategy	Implementation Cost	Coefficients
Classical FIR [5]	$\mathbf{H}_c \triangleq \sum_{k=0}^K \phi_k \mathbf{S}^k$	always	LS [5], Chebyshev [6, 7]	$\mathcal{O}(MK)$	scalars: $\{\phi_k\}$
NV FIR [10]	$\mathbf{H}_{nv} \triangleq \sum_{k=0}^K \text{diag}(\phi_k) \mathbf{S}^k$	not in general	LS, convex program [10]	$\mathcal{O}(MK)$	vectors: $\{\phi_k\}$
EV FIR (*)	$\mathbf{H}_{ev} \triangleq \sum_{k=1}^K \Phi_k \dots \Phi_1$	not in general	iterative design [Sec. 3.2.3]	$\mathcal{O}(MK)$	matrices: $\{\Phi_k\}$
SIEV FIR (*)	(3.10)	always	iterative design [Sec. 3.2.3]	$\mathcal{O}(MK)$	vectors: $\{\alpha_k\}$
CEV FIR (*)	$\mathbf{H}_{cev} \triangleq \sum_{k=1}^K \Phi_k \mathbf{S}^{k-1}$	not in general	LS [Sec. 3.3.3]	$\mathcal{O}(MK)$	matrices: $\{\Phi_k\}$
SICEV FIR (*)	(3.15)	always	LS [Sec. 3.3.3]	$\mathcal{O}(MK)$	vectors: $\{\alpha_k\}$
Classical ARMA ₁ [12]	$\mathbf{H}_{arma_1} \triangleq \varphi(\mathbf{I} - \psi \mathbf{S})^{-1}$	always	closed-form, iterative design [12]	$\mathcal{O}(I \cdot M)$	scalars: $\{\varphi, \psi\}$
EVA ₁ (*)	$\mathbf{H}_{eva_1} \triangleq (\mathbf{I} - \Phi_1)^{-1} \Phi_0$	not in general	Shanks' Method [Sec. 3.4.3]	$\mathcal{O}(I \cdot M)$	matrices: $\{\Phi_0, \Phi_1\}$
SIEVA ₁ (*)	(3.27)	always	Shanks' Method [Sec. 3.4.3]	$\mathcal{O}(I \cdot M)$	vectors: $\{\alpha_0, \alpha_1\}$

Table 3.1: Summary of the different graph filters. (*) indicates a contribution of this work. Here, I stands for the maximum number of iterations.

3.5.1. GRAPH FILTER APPROXIMATION

We here test the proposed FIR graph filters in approximating a user-provided frequency response. We consider a random community graph of $N = 256$ nodes and shift operator $\mathbf{S} = \mathbf{L}$. The frequency responses of interest are two commonly used responses in the GSP community, i.e.,

(i) the exponential kernel

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

with γ and μ being the spectrum decaying factor and the central parameter, respectively;

(ii) the ideal low-pass filter

$$\tilde{h}(\lambda) = \begin{cases} 1 & 0 \leq \lambda \leq \lambda_c \\ 0 & \text{otherwise,} \end{cases}$$

with λ_c being the cut-off frequency.

The approximation accuracy of the different filters is evaluated in terms of the normalized squared error $\text{NSE} = \|\tilde{\mathbf{H}} - \mathbf{H}_{\text{fit}}\|_F^2 / \|\tilde{\mathbf{H}}\|_F^2$. \mathbf{H}_{fit} stands for the filter matrix of the fitted filters.

Figures 3.3 and 3.4 illustrate the performances of the different filters. In the exponential kernel scenario, we observe that the CEV FIR filter outperforms the other alternatives by showing a performance improvement of several orders of magnitude. A similar result is also seen in the low-pass example, where the CEV FIR filter achieves the error floor for $K = 8$, while the NV graph filter for $K = 13$ and the classical FIR filter for $K = 17$. Additionally, we observe that the SIEV FIR filter achieves a similar performance as the NV FIR filter. This result suggests that despite the additional DoF of the SIEV FIR filter, the nonconvex design strategy (3.14) yields a local minimum that does not exploit the full capabilities of the filter. This local minimality effect can be seen in the stagnation of the error of the SIEV FIR filter for the exponential kernel case after $K \geq 8$. Finally, we notice that the SICEV filter achieves a performance similar to the NV Filter of the same order, while having less DoF. This characteristic of the SICEV shows its benefits as the order increases. Having to

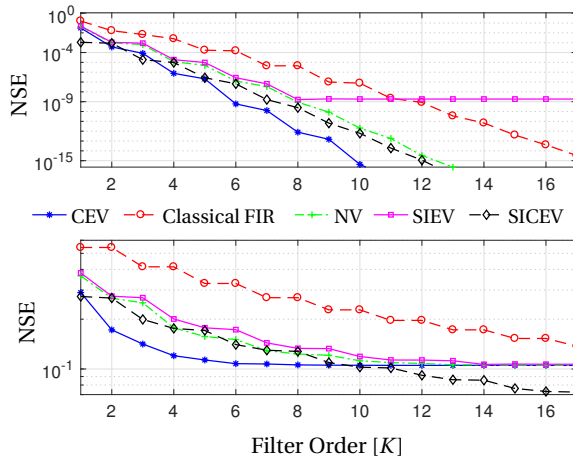


Figure 3.3: NSE versus filter order for different FIR graph filters. (Top) Results in approximating the exponential kernel response. (Bottom) Results in approximating a low-pass response.

estimate less parameters, the error stagnation for the step response is achieved at a higher filter order, hence a better approximation can be obtained.

The above observations further motivate the use of the CEV FIR filter, which trades better the simplicity of the design and the available DoF. In fact, even though the CEV FIR filter is conceptually simpler than the SIEV graph filter, it performs better than the latter. In addition, the larger DoF of the CEV FIR filter compared to the NV FIR filter (i.e., $\text{nnz}(\mathbf{S}) \cdot K + N$ vs $N \cdot (K + 1)$) allows the CEV FIR filter to better approximate the desired response. In a distributed setting, these benefits translate into communication and computational savings.

3.5.2. DISTRIBUTED LINEAR OPERATOR APPROXIMATION

Several distributed tasks of interest consist of performing a linear operation $\tilde{\mathbf{H}} \in \mathbb{R}^{N \times N}$ over a network. This can be for instance a beamforming matrix over a distributed array or a consensus matrix. In most of these cases, such linear operators cannot be straightforwardly distributed. In this section, we illustrate the capabilities of the developed graph filters in addressing this task.

Given a desired linear operator $\tilde{\mathbf{H}}$, we aim at implementing this linear operator distributively through the solution of the optimization problem

$$\begin{aligned} & \underset{\boldsymbol{\theta}}{\text{minimize}} && \|\tilde{\mathbf{H}} - \mathbf{H}(\mathbf{S}, \boldsymbol{\theta})\| \\ & \text{subject to} && \boldsymbol{\theta} \in \Theta, \end{aligned} \tag{3.36}$$

where $\mathbf{H}(\mathbf{S}, \boldsymbol{\theta})$ stands for the considered graph filter parametrized by the shift \mathbf{S} and a set of parameters $\boldsymbol{\theta}$ living in the domain Θ .

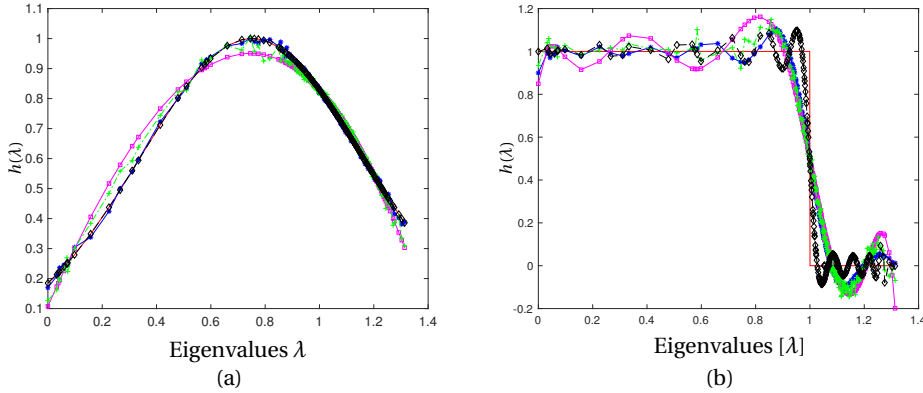


Figure 3.4: (a) Frequency response of the graph filters when approximating an exponential kernel with parameters $\mu = 0.75$ and $\gamma = 3$. (b) Frequency response of the graph filters when approximating a perfect low pass filter with $\lambda_c = 1$.

Distributed consensus. For distributed consensus, the operator $\tilde{\mathbf{H}}$ has the form $\tilde{\mathbf{H}} = \frac{1}{N} \mathbf{1}\mathbf{1}^T$, which for $\mathbf{S} = \mathbf{L}$ translates into a low-pass graph filter passing only the constant signal component. In this example, we consider a community network consisting of $N = 512$ nodes generated using the GSP Toolbox [42].

Figure 3.5 compares the fitting NSE = $\|\tilde{\mathbf{H}} - \mathbf{H}_{\text{fit}}\|_F^2 / \|\tilde{\mathbf{H}}\|_F^2$ for the different FIR graph filters. We notice once again that the CEV implementation offers the best approximation accuracy among the contenders achieving an NSE of order 10^{-4} in only 10 exchanges. These results yield also different insights about the SIEV and SICEV graph filters.

First, both the SIEV and the SICEV implementations fail to compare well with the CEV, though the linear operator $\tilde{\mathbf{A}}$ is shift invariant. We attribute this degradation of performance to Assumption 3.2, which is a sufficient (not necessary) condition for these filters to have a modal response interpretation. In fact, forcing each filter coefficient matrix to be shift invariant seems limiting the filter ability to match well the consensus operator.

Second, the different design strategies used in SIEV and SICEV filters further discriminate the two filters. We can see that the least-squares the design of the SICEV implementation is more beneficial, though the SIEV filter has more DoF. Unfortunately, this is the main drawback of the latter graph filter, which due to the nonconvexity of the design problem leads to suboptimal solutions. However, we remark that both these filters outperform (or compare equally with) the classical FIR filter. Further investigation in this direction is needed to understand if the SIEV and/or SICEV structures can be used to achieve finite-time consensus as carried out in [43, 44].

Wiener-based denoising. When the statistics of the graph signal and noise signal are available, a typical approach for performing denoising is the Wiener filter. This filter is obtained by minimizing the mean-squared error, i.e.,

$$\tilde{\mathbf{H}} = \underset{\mathbf{H} \in \mathbb{R}^{N \times N}}{\operatorname{argmin}} \mathbb{E}[\|\mathbf{H}\mathbf{z} - \mathbf{x}\|_2^2], \quad (3.37)$$

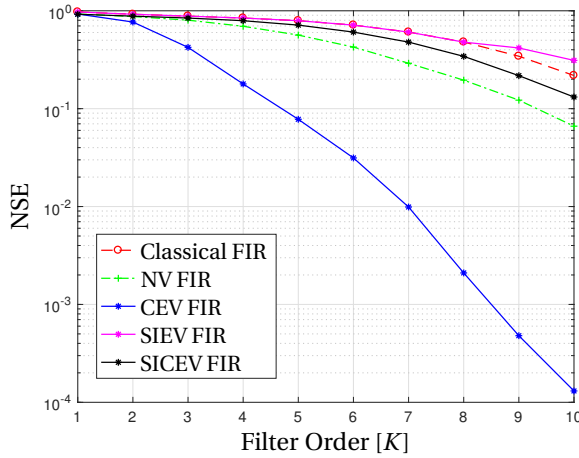


Figure 3.5: NSE versus filter order for different distributed FIR filter implementations when approximating the consensus operator $\hat{\mathbf{H}} = 1/N\mathbf{1}\mathbf{1}^T$.

where $\mathbf{z} = \mathbf{x} + \mathbf{n}$ is the graph signal corrupted with additive noise. For the case of zero-mean signals \mathbf{x} and \mathbf{n} with covariance matrices $\Sigma_{\mathbf{x}}$ and $\Sigma_{\mathbf{n}}$, respectively, the solution for (3.37) is

$$\hat{\mathbf{H}} = \Sigma_{\mathbf{x}}(\Sigma_{\mathbf{x}} + \Sigma_{\mathbf{n}})^{-1}, \quad (3.38)$$

given $\Sigma_{\mathbf{x}} + \Sigma_{\mathbf{n}}$ is not singular. When the covariance matrices $\Sigma_{\mathbf{x}}$ and $\Sigma_{\mathbf{n}}$ share the eigenvectors with the graph shift operator, the optimal filter $\hat{\mathbf{H}}$ can be approximated by classical graph filters. However, in many instances, this is not the case.

We illustrate an example where instead of approximating the Wiener filter through a classical FIR graph filter, we employ a CEV FIR filter. For this example we consider the Molenne dataset², where the temperature data of several cities in France has been recorded. The graph employed is taken from [45] and the graph signal has been corrupted with white Gaussian noise. The results in terms of NSE for the different fitted graph filters are shown in Figure 3.6. From this plot we observe that the CEV FIR filter outperforms all the other alternatives. This is due to the fact that the optimal Wiener filter is not jointly diagonalizable with the eigenbasis of the shift operator, i.e., the covariance matrix of data is not shift invariant, hence classical graph filters are not appropriate to approximate the filter.

Distributed Beamforming. We here consider the task of applying a beamforming matrix \mathbf{Z}^H to signals acquired via a distributed array. More specifically, we aim at obtaining the output

$$\mathbf{y} = \mathbf{Z}^H \mathbf{x}, \quad (3.39)$$

where \mathbf{x} is the data acquired in a distributed way. Since \mathbf{Z}^H might often be a dense matrix, e.g., in zero-forcing beamforming, the operation (3.39) cannot be readily distributed. To

²Access to the raw data through the link

donneespubliques.meteofrance.fr/donnees_libres/Hackathon/RADOMEH.tar.gz

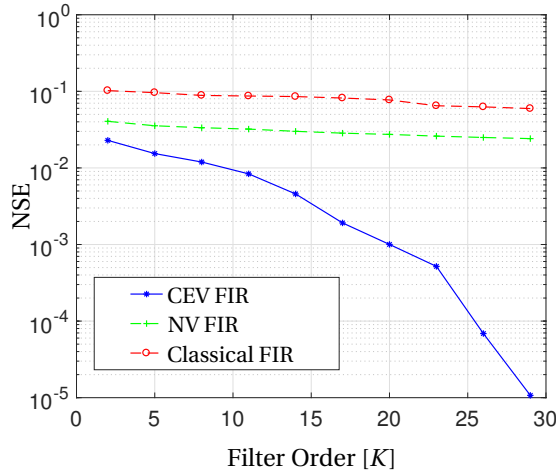


Figure 3.6: NSE versus filter order for different distributed FIR filter implementations when approximating the Wiener filter for the Molene temperature dataset.

obtain the output at each node, we approximate the beamforming matrix with different graph filters.

We illustrate this scenario in a distributed 2D sensor array. The network is generated using $N = 1000$ random locations on a 2D plane where the communication network is an $[N/5]$ -nearest neighbor graph. The beamforming matrix is the matched filter [46] matrix for a uniform angular grid of 1000 points in the range $(-180, 180)$. In other words, every node will see the information from a sector of approximately 0.36 degrees. Since in general \mathbf{Z}^H does not share the eigenbasis with \mathbf{S} , classical graph filters fail to address this task. Therefore, here we compare only the CEV FIR filter and the NV FIR filter. Figure 3.7 shows two output beampatterns obtained by solving (3.36) with $\tilde{\mathbf{H}} = \mathbf{Z}^H$ for the two considered filters with order $K = 5$. We notice that the CEV outperforms the NV FIR filter as it follows more closely the desired beampattern.

To demonstrate that our proposed designs can be directly extended to *asymmetric* shift matrices, we also plot, in Figure 3.7, the beampatterns obtained when we consider an asymmetric version of the previous communication network. This asymmetric network is generated by converting 323 random edges of the original nearest neighbor graph into directed edges. We refer to these beampatterns in the figure as CEV_{Asym} for the CEV filter and NV_{Asym} for the NV filter. From Figure 3.7, we observe that despite the performance changes in the sidelobe region, in the mainlobe region the CEV and CEV_{Asym} perform similarly both outperforming their NV graph filter counterparts.

In the discussed beamforming application, we assume that \mathbf{Z}^H is fixed and that it must be applied to the array data. This problem can be solved through distributed convex optimization tools by solving the least-squares problem

$$\underset{\mathbf{y}}{\text{minimize}} \quad \|\mathbf{x} - (\mathbf{Z}^H)^\dagger \mathbf{y}\|_2^2. \quad (3.40)$$

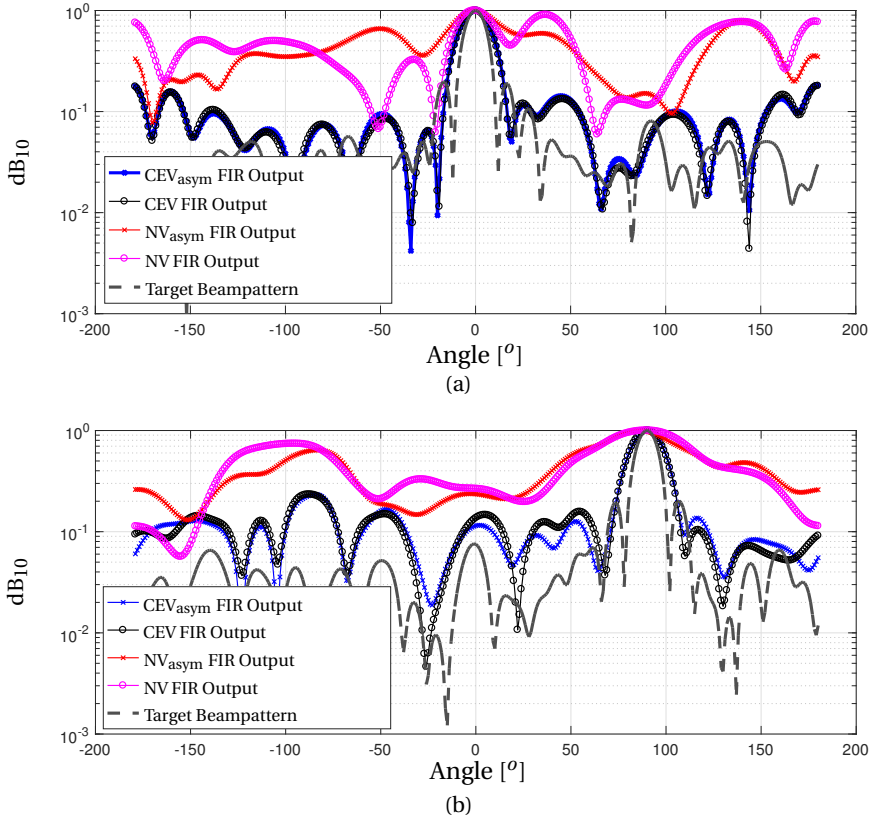


Figure 3.7: Comparison of beampatterns for the node with desired steering angle, θ_0 . (a) $\theta_0 = 0^\circ$ and (b) $\theta_0 = 90^\circ$.

Note that our formulation avoids the computation of the pseudo-inverse and the graph-filtering based approach requires only five iterations to compute the final output.

We next compare the CEV and the NV graph filters with distributed optimization tools for solving a general inverse problem.

3.5.3. COMPARISON WITH DISTRIBUTED OPTIMIZATION

We now compare the proposed graph filters with the primal-dual method of multipliers (PDMM)³ [47] to solve the following least-squares problem in a distributed fashion:

$$\underset{\mathbf{x}}{\text{minimize}} \quad \|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2. \quad (3.41)$$

Without loss of generality, we consider \mathbf{H} to be an $N \times N$ matrix. The baseline assumption for all distributed optimization methods is that v_i knows its own data, i.e., the

³PDMM is an alternative to the classical alternating direction method of multipliers (ADMM), and is often characterized by a faster convergence [47].

i th element of \mathbf{y} , y_i , and its own regressor, i.e., the i th row of \mathbf{H} , \mathbf{h}_i^\top . The task is then for each node to retrieve the full vector $\mathbf{x}_{\text{ls}} = \mathbf{H}^\dagger \mathbf{y}$ by means of local communications. Here, the communication graph is a community graph with $N = 512$ with $\lceil \sqrt{N}/2 \rceil$ communities generated using the GSP toolbox [42].

For the graph filter-based approaches, we approximate \mathbf{H}^\dagger through a set of rank-one matrices $\{\tilde{\mathbf{H}}_i \triangleq \mathbf{I} \tilde{\mathbf{h}}_i^\top\}_{i=1}^N$ with $\tilde{\mathbf{h}}_i^\top$ being the i th row of \mathbf{H}^\dagger . This means that in contrast to distributed optimization methods, here every node ν_i needs to know the full \mathbf{H} . Each $\tilde{\mathbf{H}}_i$ is then fitted with the NV and CEV recursions to approximate \mathbf{x}_{ls} as the output after filtering the graph signal \mathbf{y} . It must be noticed that the number of rounds between adjacent nodes does not scale with N . In fact, both the NV and the CEV will shift the signal only K times and the nodes can locally apply the respective coefficients to obtain the outputs.

To quantify the performance, we perform 100 Monte Carlo simulations with a randomly generated system matrix and solution vector. Figure 3.8 compares the graph filter approaches with the distributed optimization methods in terms of the NSE = $\|\mathbf{x} - \hat{\mathbf{x}}^{(k)}\|_2^2 / \|\mathbf{x}\|_2^2$. The graph filter methods achieve a faster decay compared to the distributed optimization method in the first hundred iterations. However, due the ill-conditioning of the system matrices, perfect approximation of the desired response is not achieved and both graph filters exhibit an error floor. PDMM, on the other hand, does not run into this issue and guarantees convergence to the true solution. Despite this difference in performance, the graph filter approaches can be employed for cases where the accuracy requirements are not strict, or as *warm starts* for distributed optimization methods. The above comparison, besides proposing graph filters as an alternative for solving distributed least-squares problems, raises the question *how graph filters relate to distributed convex optimization*. Further research is needed to relate the design and implementation of distributed EV graph filters with the well-established theory of distributed optimization.

3.5.4. TIKHONOV-BASED DENOISING

One of the central problems in GSP is that of recovering an unknown signal \mathbf{x} from a noisy realization $\mathbf{z} = \mathbf{x} + \mathbf{n}$ given that \mathbf{x} is smooth w.r.t. the underlying graph [4]. Differently known as the Tikhonov denoiser, the estimation of \mathbf{x} can be obtained by solving the regularized least-squares problem:

$$\mathbf{x}_{\text{tik}} = \arg \min_{\mathbf{x} \in \mathbb{R}^N} \|\mathbf{z} - \mathbf{x}\|_2^2 + \mu \mathbf{x}^\top \mathbf{S} \mathbf{x}, \quad (3.42)$$

for $\mathbf{S} = \mathbf{L}$ and where μ trades off the noise removal with the smoothness prior. Problem (3.42) has the well-known solution $\mathbf{x}_{\text{tik}} = (\mathbf{I} + \mu \mathbf{S})^{-1} \mathbf{z}$, which in terms of the terminology used in Chapter 2 is an ARMA₁ graph filter with $\varphi = 1$ and $\psi = -\mu$ (see also [12] for further analysis). While recursion (2.23) can implement this problem distributively, the convergence of the Neumann series in (2.24) cannot be controlled as the rate is fixed by $|\mu| \lambda_{\max}\{\mathbf{S}\}$.

Here, we show that through the EVA₁ in (3.25) it is possible to improve the convergence speed of the ARMA₁ graph filter by exploiting the additional DoF given by the edge-weighting matrices $\{\Phi_0, \Phi_1\}$. However, since now the design is not exact and involves the modified error [cf. (3.31)], this speed benefit will come at the expense of accuracy.

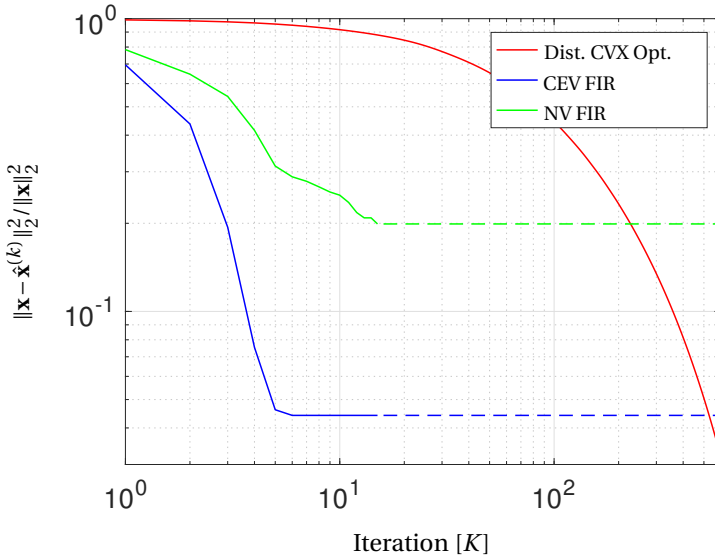


Figure 3.8: Convergence error versus the number of iterations for the NV and the CEV graph filters and for the PDMM solver [47]. Dashed lines indicate the saturation floor of the NV and CEV FIRs.

To illustrate this, we consider an example of problem (3.42) with $\mu = 0.8$ and $\mathbf{S} = \lambda_{\max}^{-1}(\mathbf{L})\mathbf{L}$, such that \mathbf{S} has unitary spectral norm. Here, the network is generated using $N = 300$ random locations on a 2D plane where the communication network is an $\lceil N/5 \rceil$ -nearest neighbor graph. Figure 3.9 shows the convergence error of the EVA_1 for different values of δ in (3.31) and compares it with the classical ARMA_1 .

We make the following observations. First, low values of δ are preferred to improve the convergence speed. However, values below 0.7 should in general be avoided since this restricts too much the feasible set of (3.31), hence leading to a worse approximation error. Second, values of $\delta \approx 0.7$ seem to give the best tradeoff, since the convergence speed is doubled w.r.t the ARMA_1 and the approximation error is close to machine precision.

Finally, we did not plot the classical FIR filter for solving this problem, since its performance is identical to the ARMA_1 for the same distributed cost [12].

3.6. CHAPTER SUMMARY

In this chapter, a generalization of distributed graph filters was proposed. These filters, which we referred to as edge-variant graph filters, can assign different weights to the information coming from a node's neighbors. Through the design of edge-weighting matrices, we have shown that it is possible to weight, possibly in an asymmetric fashion, the information propagated in the network and improve the performance of state-of-the-art graph filters.

Using the notion of filter modal response, presented in Chapter 2, we showed that a

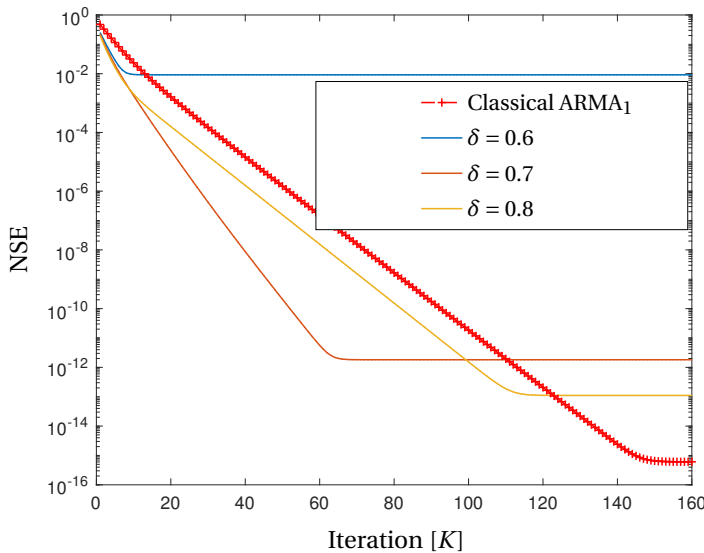


Figure 3.9: Convergence error versus the number of iterations for the Tikhonov denoising problem. The EVA₁ results are plotted for different values of δ in (3.31) to highlight the tradeoff between convergence speed and approximation accuracy.

subclass of the edge-variant graph filters has a graph Fourier interpretation that illustrates the filter action on the graph modes. Although the most general edge-variant graph filter encounters numerical challenges in the design phase, we introduced a constrained version of it to tackle this issue. This so-called constrained edge-variant graph filter enjoys a similar distributed implementation, generalizes the state-of-the-art approaches, and enjoys a simple least-squares design. For this graph filter rendition, we also showed that there exists a subclass that has a modal response interpretation.

Finally, we extended the edge-variant idea to the family of IIR graph filters, particularly to the ARMA₁ graph filter. We showed that by adopting the same approach, a distributed rational filter can be achieved yet with much faster convergence speed. Several numerical tests corroborate our findings and show the potential of the proposed filters to improve state-of-the-art techniques.

Despite that, in this chapter, we have shown that generalized graph filters offer tangible benefits for distributed processing, we did not discuss two characteristics of their implementation and design. First, it is not clear how to implement these distributed graph filters in an asynchronous network, i.e., lack of synchronization among nodes. And second, how should we deal with the numerical problems that appear when designing generalized GFs with large filters orders using direct methods. In the following two chapters, we discuss these two aspects of the implementation of GFs and present our results in these research directions.

REFERENCES

- [1] M. Coutino, E. Isufi, and G. Leus, *Distributed edge-variant graph filters*, in *IEEE 7th Int. Workshop Comp. Adv. in Multi-Sensor Adap. Proc.(CAMSAP)* (IEEE, 2017).
- [2] M. Coutino, E. Isufi, and G. Leus, *Advances in distributed graph filtering*, *IEEE Transactions on Signal Processing* **67**, 2320 (2019).
- [3] G. Taubin, *Geometric signal processing on polygonal meshes*, in *EUROGRAPHICS* (2000).
- [4] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, *The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains*, *IEEE Sig. Proc. Mag.* **30**, 83 (2013).
- [5] A. Sandryhaila and J. M. Moura, *Discrete signal processing on graphs*, *IEEE Trans. Signal Process* **61**, 1644 (2013).
- [6] G. Taubin, T. Zhang, and G. Golub, *Optimal surface smoothing as filter design*, in *European Conf. on Computer Vision* (Springer, 1996) pp. 283–292.
- [7] D. I. Shuman, P. Vandergheynst, and P. Frossard, *Distributed signal processing via chebyshev polynomial approximation*, arXiv preprint arXiv:1111.5239 (2011).
- [8] S. K. Narang, A. Gadde, and A. Ortega, *Signal processing techniques for interpolation in graph structured data*, in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)* (IEEE, 2013) pp. 5445–5449.
- [9] M. Onuki, S. Ono, M. Yamagishi, and Y. Tanaka, *Graph signal denoising via trilateral filter on graph spectral domain*, *IEEE Trans. on Sig. and Inf. Proc. over Netw.* **2**, 137 (2016).
- [10] S. Segarra, A. Marques, and A. Ribeiro, *Optimal graph-filter design and applications to distributed linear network operators*, *IEEE Trans. Signal Process* (2017).
- [11] A. Loukas, A. Simonetto, and G. Leus, *Distributed autoregressive moving average graph filters*, *IEEE Sig. Proc. Lett.* **22**, 1931 (2015).
- [12] E. Isufi, A. Loukas, A. Simonetto, and G. Leus, *Autoregressive moving average graph filtering*, *IEEE Trans. Signal Process* **65**, 274 (2017).
- [13] D. I. Shuman, B. Ricaud, and P. Vandergheynst, *Vertex-frequency analysis on graphs*, *Applied and Computational Harmonic Analysis* **40**, 260 (2016).
- [14] A. Sandryhaila and J. M. Moura, *Discrete signal processing on graphs: Frequency analysis*. *IEEE Trans. Signal Processing* **62**, 3042 (2014).
- [15] M. Belkin, P. Niyogi, and V. Sindhwani, *Manifold regularization: A geometric framework for learning from labeled and unlabeled examples*, *Journal of machine learning research* **7**, 2399 (2006).
- [16] J. Ma, W. Huang, S. Segarra, and A. Ribeiro, *Diffusion filtering of graph signals and its use in recommendation systems*, in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)* (IEEE, 2016) pp. 4563–4567.
- [17] 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 *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)* (IEEE, 2014) pp. 1115–1119.
- [18] E. Isufi, P. Di Lorenzo, P. Banelli, and G. Leus, *Distributed wiener-based reconstruction of graph signals*, in *IEEE Stat. Sig. Proc. (SSP) Workshop* (2018).

- [19] F. Zhang and E. R. Hancock, *Graph spectral image smoothing using the heat kernel*, *Pattern Recognition* **41**, 3328 (2008).
- [20] A. C. Yağın and M. T. Özgen, *A spectral graph wiener filter in graph fourier domain for improved image denoising*, in *Sig. and Inf. Proc. (GlobalSIP), 2016 IEEE Global Conference on* (IEEE, 2016) pp. 450–454.
- [21] E. Isufi and G. Leus, *Distributed sparsified graph filters for denoising and diffusion tasks*, in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)* (IEEE, 2017) pp. 5865–5869.
- [22] N. Tremblay, G. Puy, R. Gribonval, and P. Vandergheynst, *Compressive spectral clustering*, in *Int. Conf. on Machine Learning* (2016) pp. 1002–1011.
- [23] D. B. Tay and Z. Lin, *Design of near orthogonal graph filter banks*, *IEEE Sig. Proc. Lett.* **22**, 701 (2015).
- [24] O. Teke and P. P. Vaidyanathan, *Extending classical multirate signal processing theory to graphs part ii: M-channel filter banks*, *IEEE Trans. Signal Process* **65**, 423 (2017).
- [25] D. K. Hammond, P. Vandergheynst, and R. Gribonval, *Wavelets on graphs via spectral graph theory*, *Applied and Computational Harmonic Analysis* **30**, 129 (2011).
- [26] M. Defferrard, X. Bresson, and P. Vandergheynst, *Convolutional neural networks on graphs with fast localized spectral filtering*, in *Advances in Neural Information Processing Systems* (2016) pp. 3844–3852.
- [27] F. Gama, A. G. Marques, G. Leus, and A. Ribeiro, *Convolutional neural networks architectures for signals supported on graphs*, arXiv preprint arXiv:1805.00165 (2018).
- [28] X. Shi, H. Feng, M. Zhai, T. Yang, and B. Hu, *Infinite impulse response graph filters in wireless sensor networks*, *IEEE Sig. Proc. Lett.* **22**, 1113 (2015).
- [29] E. Isufi, A. Loukas, A. Simonetto, and G. Leus, *Filtering Random Graph Processes Over Random Time-Varying Graphs*, *IEEE Trans. Signal Process* (2017).
- [30] L. L. Magoarou and R. Gribonval, *Flexible multilayer sparse approximations of matrices and applications*, *IEEE Journal of Sel. Topics in Sig. Proc.* **10**, 688 (2016).
- [31] S. Barbarossa, G. Scutari, and T. Battisti, *Distributed signal subspace projection algorithms with maximum convergence rate for sensor networks with topological constraints*, in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)* (IEEE, 2009).
- [32] V. Kalofolias, A. Loukas, D. Thanou, and P. Frossard, *Learning time varying graphs*, in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)* (IEEE, 2017) pp. 2826–2830.
- [33] M. Kolar, L. Song, A. Ahmed, and E. P. Xing, *Estimating time-varying networks*, *The Annals of App. Stats.* , 94 (2010).
- [34] M. Coutino, S. Chepuri, and G. Leus, *Sparsest network support estimation: A sub-modular approach*, in *2018 IEEE Data Science Workshop* (2018).
- [35] Y. Xu and W. Yin, *A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion*, *SIAM Journal on imaging sciences* **6**, 1758 (2013).
- [36] *MATLAB Optimization Toolbox*, *The MathWorks, Inc., Natick, Massachusetts, United States*.
- [37] K. Van Acker, G. Leus, M. Moonen, O. Van de Wiel, and T. Pollet, *Per tone equalization for dmt-based systems*, *IEEE Trans. Commun.* **49**, 109 (2001).

- [38] C. Manss, D. Shutin, and G. Leus, *Distributed splitting-over-features sparse bayesian learning with alternating direction method of multipliers*, in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)* (2017).
- [39] E. Isufi, A. Loukas, and G. Leus, *Autoregressive moving average graph filters a stable distributed implementation*, in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)* (2017).
- [40] J. L. Shanks, *Recursion filters for digital processing*, *Geophysics* **32**, 33 (1967).
- [41] J. Liu, E. Isufi, and G. Leus, *Filter design for autoregressive moving average graph filters*, arXiv preprint arXiv:1711.09086 (2017).
- [42] N. Perraudin, J. Paratte, D. Shuman, L. Martin, V. Kalofolias, P. Vandergheynst, and D. K. Hammond, *GSPBOX: A toolbox for signal processing on graphs*, ArXiv e-prints (2014), [arXiv:1408.5781 \[cs.IT\]](https://arxiv.org/abs/1408.5781) .
- [43] L. Wang and F. Xiao, *Finite-time consensus problems for networks of dynamic agents*, *IEEE Trans. on Autom. Control* **55**, 950 (2010).
- [44] A. Sandryhaila, S. Kar, and J. M. Moura, *Finite-time distributed consensus through graph filters*, in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)* (IEEE, 2014) pp. 1080–1084.
- [45] N. Perraudin and P. Vandergheynst, *Stationary signal processing on graphs*, *IEEE Trans. Signal Process* **65**, 3462 (2017).
- [46] L. Harry and V. Trees, *Optimum array processing*, Part IV of Detection, Estimation and Modulation Theory (2002).
- [47] G. Zhang and R. Heusdens, *Distributed optimization using the primal-dual method of multipliers*, *IEEE Trans. on Sig. and Inf. Proc. over Netw.* **4**, 173 (2018).

4

ASYNCHRONOUS GENERALIZED GRAPH FILTERS

*And as the days, there are no
two equal quotes.*

Mario Coutiño

As the size of the sensor network grows, synchronization starts to become the main bottleneck for distributed computing. As a result, the straightforward implementation of the GFs presented in Chapter 3, assuming fully synchronized communication between nodes, is not feasible. In this Chapter, we aim to leverage results from parallel and asynchronous computing to provide guarantees for asynchronous graph filtering. Therefore, we establish what are the conditions and which kind of generalized graph filters that allow for asynchronous execution of the filtering operations. Moreover, to deal with the possible reduction of convergence speed due to asynchronous updates, we also show how a slight modification to the graph filter recursion, using the widely-known operator splitting technique, can be performed to obtain faster convergence.

4.1. INTRODUCTION

Although GFs are the workhorse for processing data living on irregular domains, as mentioned in previous chapters, there exist challenges concerning the practical implementation of some of their renditions. As motivated before, to improve the scalability of the filtering operations, the *locality* of the graph filter operations is exploited. That is, we can implement graph filters distributively to reduce the computational cost incurred by centralized computations. Unfortunately, even though the generalized GFs introduced in

Parts of this chapter have been published in the Proceedings of the IEEE Data Science Workshop (2019) [1]

Chapter 3 achieve a significant reduction in communication costs, i.e., local exchanges, while maintaining the distributed implementation, they operate under the same working assumption as that of classical GFs: *full synchronization of the nodes in the network*.

Typically, we can ensure synchronization when the GFs are implemented based on matrix-vector multiplications within a computational system. However, guaranteeing this operational mode presents a real challenge when the distribution of work is done across nodes (agents) in a physical network because most practical distributed systems are not synchronous. Hence, asynchronous graph filtering methods are of the utmost importance for overcoming the lack of synchronization of real-world distributed systems.

Fortunately, there exists extensive research concerning the asynchronous implementation of common processing tasks [2–5]. Surprisingly enough, results from distributed computation have shown, both analytically and experimentally, that asynchronous methods for solving linear problems *can be faster* than synchronous methods, and that for certain problems, they even *converge* when their synchronous counterparts do not [6]. Therefore, the goal of this chapter is to bridge the seminal results in these areas with the structures exhibited by generalized GFs to show that asynchronous filtering is possible under mild conditions.

4.1.1. CHAPTER CONTRIBUTIONS

4

The main contribution of this chapter is theoretical and concerns the feasibility of the asynchronous implementation of graph filtering in a distributed network. To do so, we provide conditions on the coefficients of the GFs that allow an asynchronous execution and show that the recent results of [7] are a particular case of the ones presented here whose foundations lay in the well-known companion matrix of a polynomial. Specifically, the main points that extend existing research are enumerated below.

- We provide conditions and analytical guarantees for the convergence of a family of asynchronous distributed CEV GFs. Differently from other works, we provide both strong and weak guarantees for the convergence in the asynchronous regime without invoking diagonal dominance or its relation with inexact multiplicative block relaxation; see, e.g., [6].
- We introduce our results through lemmas with sufficient conditions that facilitate the selection of filter coefficients during the design stage. And we show, using the companion-matrix formalism, that our results generalize those from [7] related to classical GFs. Also, our results hold for both exact and inexact, i.e., in a noisy setting, update regimes.
- We discuss the effect of using the widely-used operator splitting technique to improve the convergence speed of the asynchronous implementation of GFs.

4.1.2. CHAPTER OUTLINE

This chapter is organized as follows. Section 4.2 revisits the relation of linear operators with GFs and establishes a model linking the application of a generalized ARMA GF with solving a linear system of equations. Section 4.3 presents the computational model that

nodes employ in this chapter to update their values when implementing GFs. In Section 4.4, we use the introduced node update model to present our main results concerning the asynchronous implementation of generalized GFs. Section 4.5 demonstrates the validity of our theoretical results through a series of experiments and discusses the use of splitting methods to improve the convergence speed of the asynchronous implementation of GFs. Finally, Section 4.6 presents our concluding remarks.

4.2. LINEAR OPERATORS AS GRAPH FILTERS

To begin, let us first consider the following linear system

$$\mathbf{A}\mathbf{y} = \mathbf{x} \in \mathbb{R}^N, \quad (4.1)$$

and its solution $\mathbf{y}_{\text{sol}} = \mathbf{A}^{-1}\mathbf{x}$. Such systems arise in many kind of network processes and physical problems such as convection-diffusion systems [8], partial differential equations on graphs [9] or network inference problems [10].

In many instances, due to the high cost of matrix inversion or constraints in the availability/manipulation of the data, the solution of the linear system (4.1) is computed *iteratively* using the following recurrence relation [11]

$$\mathbf{y}_{l+1} = \mathbf{y}_l + (\mathbf{x} - \mathbf{A}\mathbf{y}_l), \quad (4.2)$$

for some initial vector \mathbf{y}_0 (typically the zero vector). For this particular kind of recurrence, under certain conditions on the matrix \mathbf{A} [11], we can guarantee that

$$\lim_{l \rightarrow \infty} \mathbf{y}_l = \mathbf{A}^{-1}\mathbf{x} = \mathbf{y}_{\text{sol}}. \quad (4.3)$$

Then, it is clear that this iterative procedure thus allows us to avoid the cost of performing the matrix (pseudo)inversion, to leverage the possible parallelizable matrix structure or to satisfy communication constraints.

In many network inference problems such as Tikhonov-based graph signal interpolation [10], \mathbf{A} has a structure that can be expressed by *local matrices*. That is, the linear operator \mathbf{A} can be represented through local operations. To illustrate this scenario, we consider the following filter operation $\mathbf{y}_{\text{sol}} = \mathbf{A}^{-1}\mathbf{x} = \mathbf{H}\mathbf{x}$, where \mathbf{H} is, for example, a CEV GF, i.e.,

$$\mathbf{H} = (\mathbf{I} - \sum_{k=1}^K \Phi_k \mathbf{S}^{k-1})^{-1}, \quad (4.4)$$

where we identify $\mathbf{S} \in \mathbb{R}^{N \times N}$ as the graph shift operator, i.e., the matrix representation of the network, and $\Phi_k \in \mathbb{R}^{N \times N}$, $\forall k$ as the edge-weighting matrices from the CEV GF. Here, we recall that the action of both \mathbf{S} and Φ_k can be implemented distributively. Thus, the filtering operation is done using only local exchanges.

Unfortunately, in general, \mathbf{A} does not exhibit parallelizable features over the communication graph of the distributed system. This situation arises naturally as the desired linear operator often is a dense operator that does not carry any relation to the network structure but to the underlying process under study, e.g., a set of beamformers for a distributed array. To deal with this issue, a way to fully distribute the operations, allowing us to meet

the communication constraints of the distributed system, is to approximate the system matrix inverse by a cascade of two CEV GFs, one with the form of (4.4) and the other with its inverse form. More specifically, we then compute the (approximate) solution of the system true local operations as

$$\mathbf{y}_{\text{sol}} = \mathbf{A}^{-1}\mathbf{x} = \mathbf{H}_B\mathbf{H}_A\mathbf{x}, \quad (4.5)$$

where \mathbf{H}_A has the form of the CEV GF (4.4) and \mathbf{H}_B has the inverse form of (4.4). Notice that this structure is the output of an autoregressive moving average (ARMA) CEV GF [cf. Section 3.4], matching the traditional ARMA GF definition if the matrices \mathbf{H}_A and \mathbf{H}_B commute.

Before continuing, we make clear the aim of this chapter. Although in this thesis we have not discussed the design of generalized ARMA GFs, we remark the following. In this chapter, we do not focus on designing ARMA GFs [cf. (4.5)], which has been studied in [12] for the case of classical GFs and whose design methods can be extended to generalized GFs. Instead, our aim is to provide guarantees and conditions in the GF coefficients for ensuring the convergence of iterative methods of the form (4.2) when there is a lack of synchronization between the processing units (nodes) for structured linear systems as (4.5). To do so, in this chapter, we first build lemmas around the asynchronous execution of the GF \mathbf{H}_A , and then, we consider the subsequent application of \mathbf{H}_B .

4

4.3. NODE UPDATE MODEL

In this section, before discussing results related to the application of \mathbf{H}_A in an asynchronous manner, we introduce a node update model that is amenable for asynchronous computations. We first argue why the straightforward model is not suitable for that kind of network operation mode, and then we introduce *nodal memory*, which is pivotal for the asynchronous implementation of GFs.

As we first focus on the application of \mathbf{H}_A , we start by considering the output of the filter operation

$$\mathbf{y}^* = \mathbf{H}_A\mathbf{x}. \quad (4.6)$$

Equivalently, \mathbf{y}^* can be defined as the solution of the linear system

$$\left(\mathbf{I} - \sum_{k=1}^K \Phi_k \mathbf{S}^{k-1}\right)\mathbf{y}^* = \mathbf{x}. \quad (4.7)$$

Using the structure of \mathbf{H}_A and its relation to the solution of a linear system [cf. (4.7)], we can substitute the equivalent system matrix in the recurrence relation (4.2) and obtain

$$\begin{aligned} \mathbf{y}_{l+1} &= \mathbf{x} + \sum_{k=1}^K \Phi_k \mathbf{S}^{k-1} \mathbf{y}_l \\ &= \mathbf{x} + \mathbf{B} \mathbf{y}_l, \end{aligned} \quad (4.8)$$

From (4.8), we observe that convergence is guaranteed when the *iteration matrix*, i.e., \mathbf{B} , has a spectral radius, $\rho(\mathbf{B})$, strictly smaller than one. This result can be obtained by observing that the error $\mathbf{e}_l = \mathbf{y}^* - \mathbf{y}_l$ at the l th iteration is given by

$$\mathbf{e}_{l+1} = \mathbf{B} \mathbf{e}_l. \quad (4.9)$$

Hence, the error is guaranteed to vanish (asymptotically) when $\rho(\mathbf{B}) < 1$.

In the recurrence expression (4.8), we observe that a graph filter of order K is required to update the current solution. This implies that a set of K communication rounds are required *before* recursion (4.8) can be updated. Hence, this model might not be suitable when disruptions in the communication among nodes occur. A possible way to deal with such a problem is to endow the nodes with *memory* as in the case of linear recursive sequences [13] where based on the characteristic polynomial of the recurrence, it is possible to generate the sequence using the so-called companion matrix [14] and the previous values in the series. Following this idea, we can consider that each diffusion step is stored and is available for transmission. More formally, by defining the k th shift of the recurrence vector as

$$\mathbf{y}_l^{(k)} = \mathbf{S}\mathbf{y}_l^{(k-1)}, \quad (4.10)$$

and stacking the vectors $\{\mathbf{y}_l^{(k)}\}_{k=0}^{K-1}$ in the column vector $\tilde{\mathbf{y}}_l \in \mathbb{R}^{NK}$, we can write an extended recurrence equation as follows

$$\begin{aligned} \tilde{\mathbf{y}}_{l+1} &= \begin{bmatrix} \mathbf{x} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} \Phi_1 & \Phi_2 & \cdots & \Phi_K \\ \mathbf{S} & & & \mathbf{0} \\ & \ddots & & \vdots \\ & & \mathbf{S} & \mathbf{0} \end{bmatrix} \tilde{\mathbf{y}}_{l+1} \\ &= \tilde{\mathbf{x}} + \tilde{\mathbf{B}}\tilde{\mathbf{y}}_l, \end{aligned} \quad (4.11)$$

for some initial vector $\tilde{\mathbf{y}}_0$ (typically the zero vector). Here, the first N elements of $\tilde{\mathbf{y}}_{l+1}$ and $\tilde{\mathbf{y}}_{l+1}$ are the vectors \mathbf{y}_{l+1} and \mathbf{y}_l [cf. (4.8)], respectively.

Despite that (4.11) provides a recurrence equation for the node variables, (4.11) does not directly reveal the locality of the operations. To show this, let us stack \mathbf{y}_l row-wise in the $K \times N$ matrix \mathbf{Y}_l , i.e.,

$$\mathbf{Y}_l := [\mathbf{y}_l^{(0)}, \dots, \mathbf{y}_l^{(K-1)}]^\top, \quad (4.12)$$

so that we can interpret the n th column of \mathbf{Y}_l as the n th node variable related to the l th iteration. Then, (4.11) can be rewritten as

$$\mathbf{Y}_{l+1} = \begin{bmatrix} \mathbf{x}^\top \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} \sum_{k=1}^K [\mathbf{Y}_l \Phi_k^\top]_{k,:} \\ \mathbf{Y}_l \mathbf{S}^\top \end{bmatrix}, \quad (4.13)$$

which shows that the n th node variable can be updated solely by combining the node variables of the neighbors of node n . As a result, by a mere exchange of local variables among neighboring nodes, convergence of (4.11) can be achieved in a distributed fashion. To simplify the further discussion, we will only work with (4.11) from now on and keep in mind that we can always distribute it following (4.13).

From the recurrence relation in (4.11), we note that the condition for convergence is now $\rho(\tilde{\mathbf{B}}) < 1$. Even though this condition on the spectrum of the iteration matrix guarantees the convergence of (4.11), we can derive a *weaker* sufficient condition for convergence, involving the matrices that comprise $\tilde{\mathbf{B}}$. The following lemma provides such a guarantee.

Lemma 1. Consider the recurrence equation

$$\bar{\mathbf{y}}_{l+1} = \bar{\mathbf{x}} + \bar{\mathbf{B}}\bar{\mathbf{y}}_l,$$

where $\bar{\mathbf{x}}$ and $\bar{\mathbf{B}}$ are defined as in (4.11). Then, for matrices \mathbf{S} and $\{\Phi_k\}_{k=1}^K$ such that

$$\rho \leq 1, \quad \gamma := a \frac{1 - \rho^K}{1 - \rho} < 1,$$

where $\rho := \|\mathbf{S}\|$ and $a := \max_{k \in \{1, \dots, K\}} \|\Phi_k\|$, as $l \rightarrow \infty$, the recurrence converges and the first N elements of $\bar{\mathbf{y}}_l$ converge to \mathbf{y}^* in the induced operator $\|\cdot\|$ -norm chosen for ρ and a .

Proof. See Appendix 4.7.1. □

This result guarantees the convergence of the recursion, in any norm, when the GF coefficients meet the conditions, and the GSO is scaled appropriately.

As we explained before, (4.11) requires the exchange of information from neighbors. In most instances, these exchanges are *inexact*. That is, errors in the communication between nodes corrupt the information transmitted in the network. Hence, the updates are not performed correctly. In the following lemma, we show weak sufficient conditions for the ϵ -approximate convergence of the iterates towards the optimal solution in the noisy setting.

Lemma 2. Consider the noisy recurrence equation

$$\bar{\mathbf{y}}_{l+1}^{\text{in}} = \bar{\mathbf{x}} + \bar{\mathbf{B}}\bar{\mathbf{y}}_l^{\text{in}} + \mathbf{v}_l,$$

where $\bar{\mathbf{x}}$ and $\bar{\mathbf{B}}$ are defined as in (4.11) and \mathbf{v}_l is a perturbation from the true update due to communication or fixed-precision errors satisfying

$$\|\mathbf{v}_l\| \leq \beta, \quad \forall l \in \mathbb{N}.$$

Then, for \mathbf{S} and $\{\Phi_k\}_{k=1}^K$ meeting the conditions of Lemma 1, as $l \rightarrow \infty$, the first N elements of $\bar{\mathbf{y}}_l^{\text{in}}$ eventually lie within a $\|\cdot\|$ -ball centered at \mathbf{y}^* of radius

$$\epsilon = \frac{\beta}{1 - \gamma},$$

with γ defined in Lemma 1.

Proof. See Appendix 4.7.2. □

From this result, we observe that if there is noise during the transmission, i.e., inexact updates, we gravitate around the desired output. Although, in general, this result might look discouraging, we show that if the perturbation is *structured*, as in the case of noiseless asynchronous updates, the recurrence asymptotically converges to the desired output.

4.4. ASYNCHRONOUS UPDATES

Though we introduced the notion of nodal memory and argued why it is central for the asynchronous implementation of GFs, we have not discussed asynchronous updates. In the following, we present how the previously node update model is adapted to allow for such type of updates.

To model asynchronous updates, we can recast the simple recurrence (4.11) into its asynchronous version

$$\bar{\mathbf{y}}_{l+1}^a = [\mathbf{D}_l \bar{\mathbf{x}} + \mathbf{D}_l \bar{\mathbf{B}} \bar{\mathbf{y}}_l^a] + [(\mathbf{I} - \mathbf{D}_l) \bar{\mathbf{y}}_l^a], \quad (4.14)$$

where the first term corresponds to the updated entries, and the second term denotes the entries that remain unchanged. In (4.14), \mathbf{I} is the $KN \times KN$ identity matrix and

$$\mathbf{D}_l := \text{diag}(\mathbf{w}_l) \in \{0, 1\}^{KN \times KN}.$$

Here, $\text{diag}(\cdot)$ constructs a diagonal matrix using its argument, $\mathbf{w}_l = [(\mathbf{w}_l^{(0)})^\top, \dots, (\mathbf{w}_l^{(K-1)})^\top]^\top$, where $\mathbf{w}_l^{(k)}$ is an N -binary vector with $[\mathbf{w}_l^{(k)}]_i = 1$ if the i th entry of $\mathbf{y}_l^{(k)}$ is updated or zero otherwise. Notice that no further structure is enforced on \mathbf{D}_l . This setting encompasses cases where not all memory entries are updated within a node, i.e., asynchronous updates within nodes.

Using the asynchronous update model (4.14), in the following lemma, we provide sufficient guarantees for the convergence of the recurrence using properties of the structured perturbation corresponding to the asynchronous update model.

Lemma 3. *Let each node update its local variable using the asynchronous recurrence (4.14). Further, let $\bar{\mathbf{x}}$ and $\bar{\mathbf{B}}$ be defined as in (4.11) and let the sequence matrices \mathbf{D}_l be sufficiently exciting, i.e., $\sum_l^\infty [\mathbf{D}_l]_{i,i} \gg 0, \forall i$. Then, for \mathbf{S} and $\{\Phi_k\}_{k=1}^K$ meeting the conditions of Lemma 1, $(\bar{\mathbf{y}}_l^a)^{(0)}$ converges to \mathbf{y}^* as $l \rightarrow \infty$.*

Proof. See Appendix 4.7.3. □

In Lemma 3, the condition on the sequence \mathbf{D}_l enforces a sufficient exchange of information over the network. That is, all the entries are seen sufficiently, i.e., updated, throughout the recurrences.

After these results, we are ready to prove the main result concerning the behavior of asynchronous constrained edge-variant graph filters.

Theorem 4.1. *Let a network perform the filtering operation*

$$\mathbf{y}_{\text{sol}} = \mathbf{H}\mathbf{x} = \mathbf{H}_B \mathbf{H}_A \mathbf{x},$$

where \mathbf{H}_A is a filter of order K of the form (4.4) and \mathbf{H}_B is a filter of order P of the inverse form of (4.4). Then, if the involved matrices \mathbf{S} and $\{\Phi_k\}_{k=1}^K$ in the graph filter \mathbf{H}_A and the sequence of matrices \mathbf{D}_l satisfy the conditions of Lemma 3, the asynchronous implementation of \mathbf{H} converges to \mathbf{y}_{sol} .

Proof. We can apply \mathbf{H}_A and use the results of Lemma 3 to show that the filtering operation converges to the desired output. Further, as each node is endowed with memory, as long as they can communicate with their neighbors, they can locally compute the action of \mathbf{H}_B when the filter order, P , of \mathbf{H}_B is lower than or equal to K . In the case that $P > K$, we can increase the memory of \mathbf{H}_A and set the corresponding matrices Φ_k to zero. □

Corollary 1. *The exchange of exact updates for inexact updates, as defined in Lemma 2, in the \mathbf{H}_A filter operation defined in Theorem 4.1 forces the sequence, as $l \rightarrow \infty$, to eventually lie within a $\|\cdot\|$ -ball centered at \mathbf{y}_{sol} of radius*

$$\epsilon_{\mathbf{H}_B} = \|\mathbf{H}_B\| \frac{\beta}{1-\gamma}.$$

The result from Theorem 4.1 provides insights into the sequential application of general asynchronous graph filters. That is, in the case of *classical* graph filters, where any pair of GFs commute, this result implies that we can implement any ARMA GF asynchronously. However, for generalized GFs, the ordering of the MA and AR part of the GF is *critical*.

4.4.1. CLASSICAL GRAPH FILTER: ASYNCHRONOUS CASE

Here, we particularize our results to the case of the simplest GF: the classical GF, connecting to the results of [7]. To do so, we replace each Φ_k by $\alpha_k \mathbf{S}$ in (4.11) and obtain the following relation

$$\bar{\mathbf{y}}_{l+1} = \bar{\mathbf{x}} + (\mathbf{C}_q^\top \otimes \mathbf{S}) \bar{\mathbf{y}}_l, \quad (4.15)$$

where

$$\mathbf{C}_q = \begin{bmatrix} \phi_1 & 1 & 0 & \cdots & 0 \\ \phi_2 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \phi_{K-1} & 0 & 0 & \cdots & 1 \\ \phi_K & 0 & 0 & \cdots & 0 \end{bmatrix} \quad (4.16)$$

is the companion matrix (with mirrored rows and columns w.r.t the standard definition) of the polynomial q defined as

$$q(t) = t^K - \sum_{i=1}^K \phi_i t^{K-i}. \quad (4.17)$$

Using (4.15), we can state that the recursion converges if

$$\rho(\mathbf{C}_q) \rho(\mathbf{S}) < 1. \quad (4.18)$$

For a normalized shift, i.e., $\rho(\mathbf{S}) = 1$, the condition reduces to $\rho(\mathbf{C}_q) < 1$. Therefore, by noticing that the eigenvalues of \mathbf{C}_q are the same as the roots of $q(t)$, we can perform *stable* filter design for the recursion (4.15) by enforcing constraints on the roots of $q(t)$. Notice that if (4.18) is satisfied, all the results from the lemmas follow.

In the following section, we present a series of numerical experiments demonstrating the behavior of the asynchronous edge-variant graph filters. Also, we discuss the role of the well-known operator splitting technique in the acceleration of the convergence of asynchronous GFs.

4.5. NUMERICAL SIMULATION

In this section, we illustrate the convergence results for general graph filters and briefly discuss the splitting of the system to provide an improvement in convergence speed.

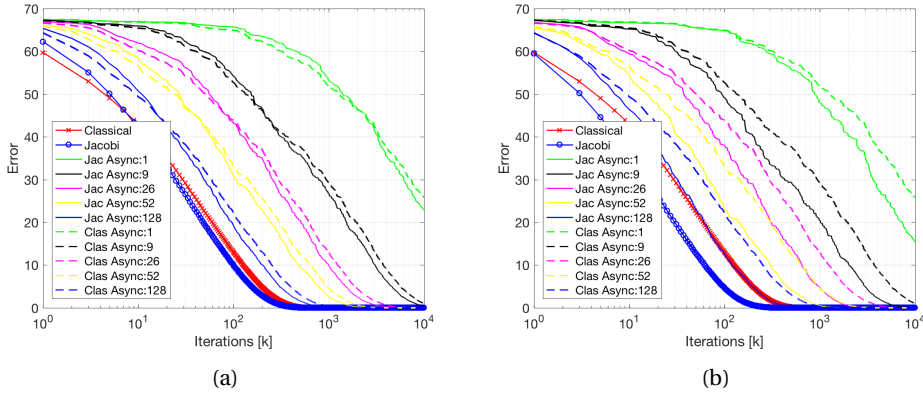


Figure 4.1: Convergence rate for different ARMA₁ implementations. (a) Jacobi updates with $\mathbf{M} = 4\text{diag}(\mathbf{H}^{-1})$. (b) Jacobi updates with $\mathbf{M} = 2.5\text{diag}(\mathbf{H}^{-1})$. Here, $\text{diag}(\cdot)$ denotes the diagonal part of the matrix argument and the number in the legend corresponds to the number of coordinated nodes.

In literature [11], recurrence equations like (4.2) are sometimes referred to as *splitting methods*. The term comes from the fact that (4.2) can be rewritten as

$$\mathbf{y}_{l+1} = (\mathbf{I} - \mathbf{M}^{-1}\mathbf{A})\mathbf{y}_l + \mathbf{M}^{-1}\mathbf{x}, \quad (4.19)$$

where $\mathbf{A} = \mathbf{M} - \mathbf{N}$. Notice that in this case, the iteration matrix \mathbf{B} is given by $\mathbf{M}^{-1}\mathbf{N}$. Hence, the (asymptotic) convergence rate is now controlled by $\rho(\mathbf{M}^{-1}\mathbf{N})$ instead of $\rho(\mathbf{I} - \mathbf{A})$.

Typical alternatives for \mathbf{M}^{-1} are the diagonal part and the lower triangular part of \mathbf{A} . The former and the latter choices lead to the so-called *Jacobi* and *Gauss-Seidel* methods, respectively. While it is well-known that Gauss-Seidel exhibits better convergence properties, the Jacobi method is, in general, the only one easy to compute in a distributed fashion, i.e., the inversion of a diagonal matrix is easily distributable.

In the following example, we show a comparison between the classical implementation and the Jacobi implementation for different numbers of randomly picked coordinated nodes, i.e., nodes that update at the same time. Here, we consider that the underlying graph is a community graph with $N = 256$ nodes (generated using the GSPToolbox [15]) and we assume a classical ARMA₁ [16]

$$\mathbf{A} = 1/\varphi(\mathbf{I} - \psi\mathbf{S}) \rightarrow \mathbf{H} = \varphi(\mathbf{I} - \psi\mathbf{S})^{-1}, \quad (4.20)$$

with $\varphi = 1$ and $\psi = 1/(\rho(\mathbf{S}) + \epsilon)$. In this case, $\rho(\mathbf{I} - \mathbf{H}) = 0.9903$. In Figs. 4.1, we show a comparison between two different splittings and the algorithm without splitting. We can see how the Jacobi implementation converges faster than the implementation without splitting, while maintaining its distributed nature. In addition, by a proper selection, i.e., optimizing $\rho(\mathbf{M}^{-1}\mathbf{N})$, we observe that the convergence speed can be increased. In this example, it is shown that the asynchronous implementation always converges independently of the number of nodes that are updated at each iteration as long as there is sufficient communication among the network.

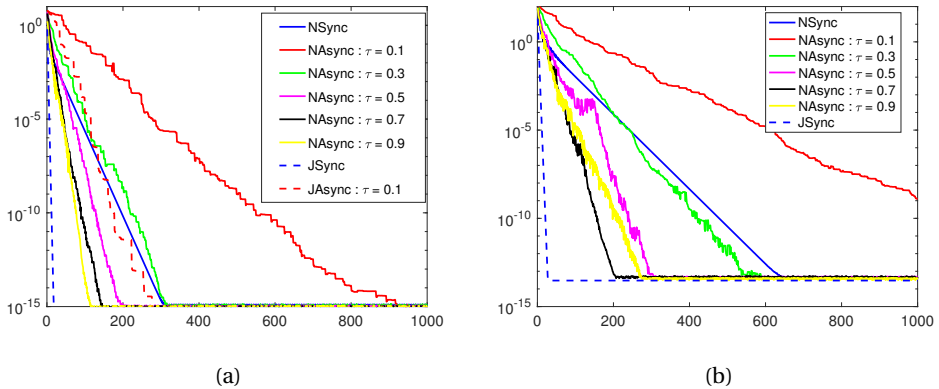


Figure 4.2: (a) Results for the time-varying Poisson equation. (b) Results for the ARMA graph filter. A community graph with $N = 64$ nodes has been used for the simulations.

In our analysis, we have provided guarantees for the behavior of asynchronous GFs for general norms. However, we want to remark that although global properties, such as bounded spectral norm, guarantee asymptotic convergence, they do not ensure monotonic convergence to the solution, i.e., to the desired filtered signal. Hence, if $\rho(\mathbf{B}) < 1$, then the recurrence converges, but the error at each iteration is not strictly decreasing.

In the following, we show this behavior for two different instances. First, we consider a Poisson equation with weighted feedback, i.e.,

$$\partial_{xx}\mathbf{y} = \mathbf{f} \rightarrow \mathbf{L}\mathbf{y} = \mathbf{g} + \mathbf{D}\mathbf{y}, \quad (4.21)$$

where \mathbf{g} is a constant field, \mathbf{L} the discretized Laplacian, \mathbf{D} a diagonal matrix with weights and $\mathbf{H} = \mathbf{L} - \mathbf{D}$ with $\rho(\mathbf{H}) = 1.6812$. Second, we consider an ARMA GF, i.e.,

$$\mathbf{H} = \mathbf{H}_B \mathbf{H}_A = [\sum_{l=0}^1 \phi_l \mathbf{S}^l] [\sum_{k=1}^3 \Phi_k \mathbf{S}^{k-1}]^{-1}, \quad (4.22)$$

with $\rho(\mathbf{H}_B) = 2.048$. For these experiments, a community graph of $N = 64$ nodes has been considered.

In Fig. 4.2a and 4.2b, we can observe that in both cases, the asynchronous versions of the GFs converge to the solution. We also notice that for different values of τ , i.e., the fraction of coordinated nodes, the convergence is faster than the synchronous version, i.e., $\tau = 1$. This behavior has been described in the parallel computing literature, where the variable shrinkage of the iteration matrix at every step is the attributed cause. That is, under certain circumstances, using the interlacing theorem, one can show that the eigenvalues of the asynchronous system are smaller than those of the synchronous one. Finally, we want to remark that there is indeed a loss in convergence speed for the *global* updating scheme [c.f. (4.8)], see, e.g., the fast convergence of the dotted lines in the figures. Those lines represent the Jacobi (J) and classical (N) updates in their synchronous (Sync) and asynchronous (Async) versions.

4.6. CHAPTER SUMMARY

In this chapter, we discussed the asynchronous implementation of generalized GFs. We first introduced a model for the node updates employing memory, making it amenable for asynchronous execution. The proposed model encompasses cases where there is miscommunication between nodes, errors within information packages, or noisy communication channels. Then, using the relation between the solution of linear inverse problems, we presented a family of generalized GFs that can be implemented asynchronously.

To guide the design for the presented family of GFs, we derived conditions for the GF coefficients guaranteeing the convergence of their asynchronous execution. Differently from other approaches, we use weaker conditions, more amenable for filter design, to provide convergence guarantees in both exact and inexact settings. Finally, we show that using our analysis tool, based on the companion matrix of a polynomial, previous results concerning the asynchronous implementation of classical GFs can be derived as a particular case of our results. To complement our theoretical results, we briefly discussed the role of the widely-used operator splitting method for speeding up the convergence of the GF implementation and showed that for different levels of asynchronicity, it is possible to converge faster than the fully asynchronous rendition.

4.7. APPENDIX

4.7.1. PROOF LEMMA 1

Let us consider the first block of the recurrences, i.e.

$$\bar{\mathbf{y}}_{l+1}^{(0)} = \mathbf{x} + \sum_{k=1}^K \Phi_k \bar{\mathbf{y}}_l^{(k-1)} = \mathbf{x} + \sum_{k=1}^K \Phi_k \mathbf{S}^{k-1} \bar{\mathbf{y}}_{l-k+1}^{(0)}.$$

As $(\mathbf{I} - \sum_{k=1}^K \Phi_k \mathbf{S}^{k-1})$ is not singular, i.e., there exists a solution \mathbf{y}^* for the system, we have that

$$\mathbf{y}^* = \mathbf{x} + \sum_{k=1}^K \Phi_k \mathbf{S}^{k-1} \mathbf{y}^*.$$

Then, by subtracting both expressions we obtain

$$(\bar{\mathbf{y}}_{l+1}^{(0)} - \mathbf{y}^*) = \sum_{k=1}^K \Phi_k \mathbf{S}^{k-1} (\bar{\mathbf{y}}_{l-k+1}^{(0)} - \mathbf{y}^*),$$

from where we can bound the norm of the error at each iteration as

$$\begin{aligned} \|(\bar{\mathbf{y}}_{l+1}^{(0)} - \mathbf{y}^*)\| &= \left\| \sum_{k=1}^K \Phi_k \mathbf{S}^{k-1} (\bar{\mathbf{y}}_{l-k+1}^{(0)} - \mathbf{y}^*) \right\| \leq \sum_{k=1}^K \|\Phi_k \mathbf{S}^{k-1} (\bar{\mathbf{y}}_{l-k+1}^{(0)} - \mathbf{y}^*)\| \\ &\leq \sum_{k=1}^K \|\Phi_k \mathbf{S}^{k-1}\| \|\bar{\mathbf{y}}_{l-k+1}^{(0)} - \mathbf{y}^*\| \leq \alpha_l \sum_{k=1}^K \rho^{k-1} \|\Phi_k\| \\ &\leq \alpha_l \sum_{k=1}^K \rho^{k-1} a = \alpha_l a \left(\frac{1 - \rho^K}{1 - \rho} \right), \end{aligned}$$

where we have defined $\alpha_l := \max_{0 \leq k < l} \|\bar{\mathbf{y}}_{l-k+1}^{(0)} - \mathbf{y}^*\|$. Choosing a as given in the lemma, provides

$$\|(\bar{\mathbf{y}}_{l+1}^{(0)} - \mathbf{y}^*)\| \leq \gamma \alpha_l,$$

where $\gamma < 1$. By recursive application of the inequality, we obtain

$$\|(\bar{\mathbf{y}}_{l+1}^{(0)} - \mathbf{y}^*)\| \leq \gamma^{l+1} \alpha_0, \quad (4.23)$$

which clearly vanishes for $l \rightarrow \infty$ and any α_0 .

As the first block converges as long as the conditions of the lemma are satisfied, the convergence of the rest of the block follows directly, e.g.,

$$\lim_{l \rightarrow \infty} \bar{\mathbf{y}}_{l+1}^{(1)} = \lim_{l \rightarrow \infty} \mathbf{S} \bar{\mathbf{y}}_l^{(0)} = \mathbf{S} \mathbf{y}^*.$$

4.7.2. PROOF LEMMA 2

As (4.13) is equivalent to (4.11), inexact updates in (4.13) can be expressed as inexact updates in the global system. Therefore, without loss of generality, we can build the proof for the global system as provided in the lemma.

Considering the inexact recurrence relation, and assuming that the exact system has a solution \mathbf{y}^* , the following inequality holds

$$\|((\bar{\mathbf{y}}_{l+1}^{\text{in}})^{(0)} - \mathbf{y}^*)\| \leq \left\| \sum_{k=1}^K \Phi_k \mathbf{S}^{k-1} ((\bar{\mathbf{y}}_{l-k+1}^{\text{in}})^{(0)} - \mathbf{y}^*) + \mathbf{v}_l^{(0)} \right\|$$

Using Lemma 1, expanding the recursion and recalling the definition of \mathbf{B} [cf. (4.8)], we obtain

$$\begin{aligned} \|((\bar{\mathbf{y}}_{l+1}^{\text{in}})^{(0)} - \mathbf{y}^*)\| &\leq \gamma^{l+1} \alpha_0 + \sum_{i=0}^l \|\mathbf{B}^i\| \|\mathbf{v}_{l-1}^{(0)}\| \\ &\leq \gamma^{l+1} \alpha_0 + \sum_{i=1}^l \omega^i \beta = \gamma^{l+1} \alpha_0 + \beta \frac{1 - \omega^{l+1}}{1 - \omega}, \end{aligned}$$

where $\omega := \|\mathbf{B}\|$, $\gamma := a(1 - \rho^K / (1 - \rho)) \leq 1$ and $\alpha_0 := \|\bar{\mathbf{y}}_0^{(0)} - \mathbf{y}^*\|$ as defined in Lemma 1. Taking the limit of the previous expression, we can obtain the asymptotic bound

$$\lim_{l \rightarrow \infty} \|((\bar{\mathbf{y}}_{l+1}^{\text{in}})^{(0)} - \mathbf{y}^*)\| \leq \frac{\beta}{1 - \omega} \leq \frac{\beta}{1 - \gamma}.$$

Finally, as γ is an upper bound of ω (see Lemma 1), the inequality follows.

4.7.3. PROOF LEMMA 3

To prove this result, we show that under the conditions of the lemma, the norm of the perturbation is subject to a non-expansive operation.

So, we first make use of the inexact update recurrence formulation to rewrite the update equation as

$$\bar{\mathbf{y}}_{l+1}^a = \bar{\mathbf{x}} + \bar{\mathbf{B}} \bar{\mathbf{y}}_l^a + \mathbf{n}_l,$$

where

$$\mathbf{n}_l := (\mathbf{D}_l - \mathbf{I})(\bar{\mathbf{x}} + \bar{\mathbf{B}}\bar{\mathbf{y}}_l^a - \bar{\mathbf{y}}_l^a).$$

Using the results of Lemmas 1 and 2, we can construct the expression (assuming \mathbf{y}^* exists)

$$\|(\bar{\mathbf{y}}_{l+1}^a)^{(0)} - \mathbf{y}^*\| \leq \gamma^{l+1}\alpha_0 + \sum_{i=0}^l \|\mathbf{B}^i\| \|\mathbf{n}_{l-i}^{(0)}\|.$$

At this point, it is clear that the first term vanishes. However, to show that the sequence converges to \mathbf{y}^* , we need to show that the second term vanishes as well for increasing l . To show this, first let us define

$$\begin{aligned} \mathbf{r}_l &:= \mathbf{x} + (\bar{\mathbf{B}} - \mathbf{I})\bar{\mathbf{y}}_l^a = \mathbf{x} + (\bar{\mathbf{B}} - \mathbf{I})(\mathbf{x} + \bar{\mathbf{B}}\bar{\mathbf{y}}_{l-1}^a + \mathbf{n}_{l-1}) \\ &= \bar{\mathbf{B}}\mathbf{r}_{l-1} + (\bar{\mathbf{B}} - \mathbf{I})\mathbf{n}_{l-1}. \end{aligned}$$

Using the definition for \mathbf{n}_l , we can rewrite \mathbf{r}_l as

$$\begin{aligned} \mathbf{r}_l &= \bar{\mathbf{B}}\mathbf{r}_{l-1} + (\bar{\mathbf{B}} - \mathbf{I})(\mathbf{D}_l - \mathbf{I})\mathbf{r}_{l-1} = ((\bar{\mathbf{B}} - \mathbf{I})\mathbf{D}_l + \mathbf{I})\mathbf{r}_{l-1} \\ &= \prod_{i=0}^l ((\bar{\mathbf{B}} - \mathbf{I})\mathbf{D}_i + \mathbf{I})\mathbf{r}_0. \end{aligned}$$

Now, let us split $\bar{\mathbf{B}}$ into two matrices, $\{\bar{\mathbf{B}}^+, \bar{\mathbf{B}}^-\}$, containing its positive and negative parts of the spectrum, respectively. Considering this partition, we can rewrite the expression for \mathbf{r}_l as

$$\mathbf{r}_l = \prod_{i=0}^l ((\bar{\mathbf{B}}^+ - \mathbf{I})\mathbf{D}_i + \mathbf{I} + \bar{\mathbf{B}}^-\mathbf{D}_i)\mathbf{r}_0.$$

From this expression, and the fact that $\rho(\bar{\mathbf{B}}) < 1$, we note the following

$$\begin{aligned} \text{eigs}\{\bar{\mathbf{B}}^+ - \mathbf{I}\} &\in [-1, 0) \\ \text{eigs}\{\mathbf{I} + \bar{\mathbf{B}}^-\mathbf{D}_i\} &\in (0, 1]. \end{aligned}$$

Hence,

$$\rho((\bar{\mathbf{B}} - \mathbf{I})\mathbf{D}_i + \mathbf{I}) \leq 1,$$

which leads to a non-expansive operation over the residual and in turn leads to a non-expansive operation over the perturbation, \mathbf{n}_l , i.e.,

$$\mathbf{n}_l = (\mathbf{D}_l - \mathbf{I})\mathbf{r}_l = (\mathbf{D}_l - \mathbf{I}) \prod_{i=0}^l \mathbf{M}_i \mathbf{r}_0,$$

where $\mathbf{M}_i = (\bar{\mathbf{B}} - \mathbf{I})\mathbf{D}_i + \mathbf{I}$. This implies

$$\|\mathbf{n}_l\| \leq \|(\mathbf{D}_l - \mathbf{I})\| \prod_{i=0}^l \|\mathbf{M}_i\| \|\mathbf{r}_0\|.$$

with $\rho(\mathbf{M}_i) \leq 1$. Hence, for $l \rightarrow \infty$ and a sufficiently exciting sequence i.e., all entries are sufficiently updated due to the lemma condition, the perturbation contracts, thus vanishes.

REFERENCES

- [1] M. Coutino and G. Leus, *Asynchronous distributed edge-variant graph filters*, in *2019 IEEE Data Science Workshop (DSW)* (2019) pp. 115–119.
- [2] D. P. Bertsekas, *Distributed asynchronous computation of fixed points*, *Mathematical Programming* **27**, 107 (1983).
- [3] D. Chazan and W. Miranker, *Chaotic relaxation*, *Linear algebra and its applications* **2**, 199 (1969).
- [4] A. Frommer and D. B. Szyld, *On asynchronous iterations*, *Journal of computational and applied mathematics* **123**, 201 (2000).
- [5] J. M. Bahi, S. Contassot-Vivier, and R. Couturier, *Parallel iterative algorithms: from sequential to grid computing* (Chapman and Hall/CRC, 2007).
- [6] E. Chow, *Convergence models and surprising results for the asynchronous jacobi method*, in *2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS)* (IEEE, 2018) pp. 940–949.
- [7] O. Teke and P. Vaidyanathan, *The asynchronous power iteration: A graph signal perspective*, in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (IEEE, 2018) pp. 4059–4063.
- [8] C. Kelley, *Iterative methods for linear and nonlinear equations*, *Frontiers in applied mathematics* **16**, 575 (1995).
- [9] A. I. Vol’pert, *Differential equations on graphs*, *Mathematics of the USSR-Sbornik* **17**, 571 (1972).
- [10] E. Isufi, A. Loukas, and G. Leus, *Autoregressive moving average graph filters a stable distributed implementation*, in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)* (2017).
- [11] Y. Saad, *Iterative methods for sparse linear systems*, Vol. 82 (siam, 2003).
- [12] J. Liu, E. Isufi, and G. Leus, *Filter design for autoregressive moving average graph filters*, *IEEE Transactions on Signal and Information Processing over Networks* **5**, 47 (2019).
- [13] A. Brousseau, *Linear Recursion and Fibonacci Sequences* (Fibonacci Assoc., 1971).
- [14] R. Bellman, *Introduction to matrix analysis*, Vol. 19 (Siam, 1997).
- [15] N. Perraudin, J. Paratte, D. Shuman, L. Martin, V. Kalofolias, P. Vandergheynst, and D. K. Hammond, *GSPBOX: A toolbox for signal processing on graphs*, *ArXiv e-prints* (2014), [arXiv:1408.5781 \[cs.IT\]](https://arxiv.org/abs/1408.5781) .
- [16] E. Isufi, A. Loukas, A. Simonetto, and G. Leus, *Autoregressive moving average graph filtering*, *IEEE Trans. Signal Process* **65**, 274 (2017).

5

A CASCADED STRUCTURE OF GRAPH FILTERS

7

*we could live as haikus:
three parts, some shorter than others;
in times, illogical, always beautiful*

fruzti, could we?

WHILE classical graph filters allow for a stable design using optimal polynomial approximation theory, generalized graph filters, as discussed in Chapter 3, require approaches based on least squares designs. Unfortunately, due to the involved design system matrices, finding the generalized graph filter coefficients tends to be severely ill-conditioned. This issue, accentuated for increasing graph filter orders, naturally leads to very large (small) filter coefficients or error saturation, casting a shadow on the benefits of these richer graph filter structures. In addition to this, data-driven design/learning of graph filters with large filter orders, even in the case of classical graph filters, suffers from the eigenvalue spread of the input-data covariance matrix and mode coupling, leading to convergence-related issues such as the ones observed when identifying time-domain filters with large orders. To alleviate these conditioning and convergence problems, and to reduce the overall design complexity, in this chapter, we introduce a cascade implementation for generalized graph filters and present an efficient algorithm for designing the graph filter coefficients in both model- and data-driven settings. Further, we establish the

Parts of this chapter have been published in the Proceedings of the Asilomar Conference on Signals, Systems, and Computers (2019) [1] and are submitted to the *IEEE Transactions on Information Processing over Networks* (2020).

connections of this implementation with the so-called graph convolutional neural networks and illustrate the performance of the proposed structure through the distributed consensus application.

5.1. INTRODUCTION

As mentioned in Chapter 2 and Chapter 3, similar to time-domain filters in traditional signal processing, GFs [2] have become the workhorse of GSP for solving inference problems such as interpolation/estimation [3, 4] and classification/detection [5, 6]. Unfortunately, similarly to their time-domain counterparts, higher-order graph filters may present stability issues in both their application and design, e.g., quantization of filter coefficients, ill-conditioning of system matrices, etc. To tackle these problems, several works have aimed at designing robust GFs, see, e.g., [7, 8] or to leverage polynomial approximation techniques to design graph filters in a stable fashion [9].

Although research has been carried out to obtain stable GF designs, most of these efforts have been focused on the so-called classical GFs, structures that share a one-to-one relation with time-domain filters and allow for spectrum-shaping designs. For the case of more complex GFs, capable of better approximating linear operators due to their increased degrees of freedom, such as the node-variant [10] and the constrained edge-variant GFs [11], a node-based design must be performed as they do not generally accept a spectrum-shaping design. Therefore, the design of such GFs relies on system matrices constructed with shifted versions, i.e., matrix powers, of the so-called graph shift operator (GSO), i.e., the matrix representation of the network. Due to the (possibly) large spread in the eigenvalues of the GSO, the resulting system matrices used for the design of these generalized GFs tend to have a poor numerical conditioning, especially for large filter orders. This leads to instabilities in their design and to the slow convergence of iterative methods employed for finding the respective coefficients [12]. Hence, there exists a need to develop filter design methods for these structures that are numerically stable and can cope with the convergence issues of the iterative methods involved in the design.

Besides the above issues, although classical GFs benefit from spectrum-shaping-type designs, their stable design is only possible when the GF response is known a priori. That is, this kind of design is only applicable when the shape of the (discrete) spectrum of the desired linear transform is known beforehand, i.e., model-driven design. However, in many cases, a data-driven design is desirable (or is the only option). Such situations arise when the only information available about the linear transform is given in terms of input-output data. This calls for methods that identify the underlying GF by mapping the available inputs to the respective outputs. Although for classical GFs, a straightforward deconvolution-type of approach can be devised for finding the input-output (and hence the spectrum-shaping function) mapping, see, e.g., [13] for blind graph filter identification, differently from the time-domain, this method requires the full eigendecomposition of the GSO which, in many instances, can be prohibitive. Further, for generalized GFs, these deconvolution-type of approaches are simply not possible as these structures are not guaranteed to be diagonalizable by the eigenbasis of the GSO. Therefore, a stable and data-efficient method for identifying generalized GFs from input-output data is much needed.

To deal with similar issues, time-domain filters have been designed/identified, in the data-driven setting, by means of iterative methods such as least mean squares (LMS) or recursive least squares (RLS) [14, 15], which due to their effectiveness, have already been adapted to the graph setting, see, e.g., [16]. However, it is well known that, for instance, LMS suffers from two main problems: the eigenvalue spread of the correlation matrix of the input data, and the coupling between modes of convergence [17]. Specifically, the former effect leads to a nonuniform convergence of the different filter coefficients and the latter to nonmonotonic trajectories toward convergence. Although RLS provides a way to decouple such convergence routes by means of a matrix inversion, it is known that RLS exhibits a large sensitivity to numerical accuracy, plus, the eigenvalue spread remains a problem [18]. Unfortunately, the latter problem gets accentuated when modelling autoregressive moving average (ARMA) processes, which requires higher-order finite impulse response (FIR) filters, as it has been shown that the eigenvalue spread is a nondecreasing function of the filter length. In addition to all these, learning long filters requires a small step size for both LMS or RLS, which in turn slows down the convergence and increases the number of parameter updates, i.e., times that the filter coefficients are updated. Therefore, further structure has to be imposed on the filters to counteract these issues.

In this chapter, our goal is two-fold. First, we aim to improve the conditioning of the system matrices involved in the design of generalized GFs [2] where classical time-domain techniques are not applicable. And second, to develop a stable and efficient updating scheme for the GF identification/learning problem in the data-driven setting. To tackle these tasks, inspired by constructions used in audio for linear prediction [19], we first introduce a cascaded implementation of generalized GFs, establishing the respective connections with so-called graph neural networks (GNNs) [20], and we then introduce an efficient algorithm for learning the coefficients of the generalized GFs that is applicable to both the model- and data-driven setting.

5.1.1. CHAPTER CONTRIBUTIONS

The main contribution of this chapter is the introduction of a framework based on the cascaded implementation of GFs which allows for a stable and efficient filter coefficient design/learning. More specifically, our contributions broadening the state-of-the-art are the following.

- We introduce a cascaded structure of distributed GFs to mitigate the effects of large GF orders on the conditioning of the GF coefficient design problem, allowing for a better numerical stability and approximation of general linear operators.
- We analyze the error surface for the least squares design of the proposed cascaded structure and show that only saddle points are introduced in the error surface of the direct implementation. This explains why stochastic gradient descent methods are ideal for designing/identifying cascaded GF coefficients.
- Exploiting the structure of the design matrices, we propose an iterative design method for cascaded GFs for the model-driven case, i.e., known data transformation. Further, under minor modifications, we adapt the proposed method to the data-driven setting. In addition, the proposed algorithms are shown to be amenable

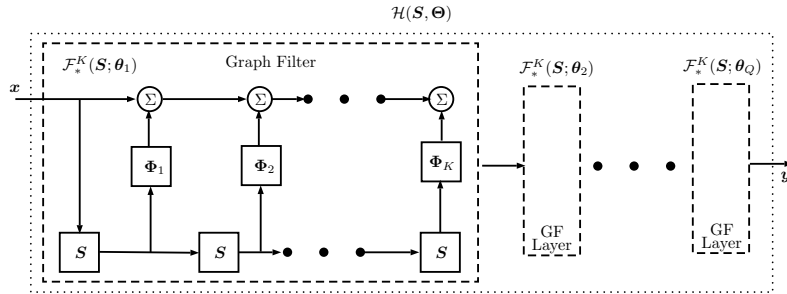


Figure 5.1: Schematic of the cascade GF implementation. Each GF can be considered as a layer of a GCN with a linear activation function. Here, we consider the CEV-GF [cf. (3.15)], however, the local matrices $\{\Phi_k\}_{k=1}^K$ can be specialized to $\{\phi_k \mathbf{I}\}_{k=1}^K$ and $\{\text{diag}(\phi_k)\}_{k=1}^K$ for C-GFs and NV-GFs, respectively.

to a large-scale implementation leveraging sparsity and deep learning frameworks such as TensorFlow [21] and Keras [22].

- Through common network applications, it is shown that the proposed cascaded implementation exhibits better numerical properties than the direct GF implementation achieving lower approximation errors while saving communication rounds in the distributed setting.

5

5.1.2. CHAPTER OUTLINE

This chapter is organized as follows. Section 5.2 introduces the proposed cascaded implementation for generalized GFs and analyzes the error surface of the corresponding least squares design problem. In addition, it illustrates the relation between cascaded GFs and graph convolutional neural networks (GCNNs). To deal with the nonconvex design of the cascaded structure, an iterative algorithm, referred to as RELAX, is proposed in Section 5.3 for the model-driven setting. Section 5.4 leverages the proposed RELAX algorithm to introduce its data-driven variant. Section 5.5 showcases the benefits of the proposed implementation by numerical experiments related to common network applications. Finally, Section 5.6 concludes the chapter.

5.2. CASCADE IMPLEMENTATION OF GRAPH FILTERS

One of the main problems of graph filter design is the numerical stability of the least squares problems involved. Despite that for classical GFs (C-GFs) polynomial fitting in the spectral domain, by means of the Chebyshev polynomial expansion [23], can be employed for large polynomial orders, other types of graph filters rely on a *node-domain* design, i.e., entry-wise fitting with respect to the target linear operator, \mathbf{H}^* ; that is,

$$\min_{\boldsymbol{\theta}} \|\mathcal{F}_*^K(\mathbf{S}; \boldsymbol{\theta}) - \mathbf{H}^*\|_F, \quad (5.1)$$

where $\mathcal{F}_*^K(\mathbf{S}; \boldsymbol{\theta})$ is a GF of order K with parameters $\boldsymbol{\theta} \in \mathbb{R}^{K \text{DoF}}$; in this notation, the asterisk is a space holder for the particular kind of GF employed, e.g., \mathcal{F}_c^K , $\mathcal{F}_{\text{nv}}^K$ or $\mathcal{F}_{\text{cev}}^K$; and DoF denotes the number of degrees of freedom of the particular kind of GF, e.g. DoF = 1 for the C-GF, DoF = N for the NV-GF and DoF = M for the CEV-GF.

Unfortunately, the LS problem in (5.1) quickly becomes ill-conditioned when the order increases. This issue leads to slow convergence of iterative methods for solving the least squares problem, memory issues due to low sparsity levels of the system matrices, or/and unstable solutions with very large (very small) filter coefficients. To deal with the conditioning issues of the system matrices in the node-domain design, and borrowing ideas from traditional cascaded implementations of finite- and infinite-impulse response filters [18, 24], we put forth a cascaded implementation of graph filters. That is, we propose to limit the maximum order of a given GF and employ this GF module as a building block of a larger system.

Mathematically, we introduce the graph operation, $\mathcal{H}(\mathbf{S})$, as

$$\mathcal{H}(\mathbf{S}; \boldsymbol{\Theta}) := \prod_{i=1}^Q \mathcal{F}_*^K(\mathbf{S}; \boldsymbol{\theta}_i) = \mathcal{F}_*^K(\mathbf{S}; \boldsymbol{\theta}_Q) \cdots \mathcal{F}_*^K(\mathbf{S}; \boldsymbol{\theta}_1), \quad (5.2)$$

where $\boldsymbol{\Theta} = [\boldsymbol{\theta}_1^\top, \dots, \boldsymbol{\theta}_Q^\top]^\top \in \mathbb{R}^{QK \text{DoF}}$. Notice that when the GF is a C-GF, (5.2) is another C-GF of order $\max\{KQ, N\}$. This is however not the case for both the NV-GF and CEV-GF unless the coefficient matrices have a particular structure, i.e., they enforce shift-invariance. The implementation of the proposed construction is shown in Fig. 5.1.

The adopted cascaded implementation is similar to the cascading of biquad filters [25] in time-domain processing for attenuating the effect of quantizing the filter coefficients. They are also similar to the structures employed for adaptive linear predictors [18] which are robust against conditioning of the input covariance matrix and accept larger step sizes in their updating steps as the coefficient sensitivities of a cascaded filter are much lower than that of the direct form [26]. Thus, similar to the motivations present in the time domain, by introducing such a filter implementation, we aim to: (i) improve the conditioning of the design problem, (ii) reduce the complexity of the optimization problems involved and (iii) achieve a better performance with a reduced order. However, all these advantages do not come for free: we need to give up the *convexity* of the overall filter design problem, i.e.,

$$\underset{\{\boldsymbol{\theta}_i\}_{i=1}^Q}{\text{argmin}} \quad \|\mathcal{H}(\mathbf{S}, \boldsymbol{\Theta}) - \mathbf{H}^*\|_{\text{F}}. \quad (5.3)$$

As the design of the direct implementation, i.e., a single GF with large order, is a convex problem [cf. (5.1)], it is theoretically guaranteed that an optimal configuration of coefficients can be efficiently found. However, due to the nonconvexity of the design problem for the cascaded problem [cf. (5.3)], this property cannot be guaranteed without understanding the error surface for this kind of problem. In the following, we present an analysis of the error surface for a particular family of problems of the form (5.3).

5.2.1. THEORETICAL STUDY OF CASCADED GF ERROR SURFACE

Surprisingly, similar to the results in system identification and adaptive filtering [26], it is possible to show that the error surface of the cascaded implementation of a particular

family of GFs exhibits critical points that are either global minimizers or saddle points. To show this, we first introduce some basic notions that characterize what happens to an error surface when a new parametrization is introduced. And then, we use these results to show the type of critical points that the error surface of the cascaded implementation of a particular family of GFs has.

CRITICAL POINTS AFTER REPARAMETRIZATION

Let us consider two *equivalent* implementations of a GF of order K , one in direct form with parameters $\boldsymbol{\theta}_d = [\theta_1, \dots, \theta_{K_d}]^\top$ and the other in cascaded form with parameters $\boldsymbol{\Theta}_c = [\boldsymbol{\theta}_{d_1}^\top, \dots, \boldsymbol{\theta}_{d_Q}^\top]$, where $\boldsymbol{\theta}_{d_i} \in \mathbb{R}^{K_c}$, $\forall i$ are the parameters of the i th cascaded module; that is,

$$\mathcal{F}_*^K(\mathbf{S}; \boldsymbol{\theta}_d) = \mathcal{H}(\mathbf{S}; \boldsymbol{\Theta}_c). \quad (5.4)$$

Further, let us denote with \mathcal{D}_d and \mathcal{D}_c the sets of feasible direct and cascaded GF coefficients of given order and kind, e.g., C-GF, NV-GF, etc., respectively. That is, if C-GFs of order K are selected, \mathcal{D}_d contains all the C-GFs of order at most K . Similarly, the equivalent set \mathcal{D}_c contains all the cascaded C-GFs with order at most K formed by modules of order K_c . Hence, $\boldsymbol{\theta}_d \in \mathcal{D}_d$ and $\boldsymbol{\Theta}_c \in \mathcal{D}_c$.

The above implies that the LS error with respect to a desired response \mathbf{H}^* is the same for both implementations (realizations). However, while for the case of the direct form the LS error function is convex with respect to the GF coefficients [cf. (5.1)], for the cascaded implementation, the LS error is clearly nonconvex [cf. (5.3)]. So, despite that the error surface for the direct implementation, due to its convexity, has only a global minimum, the cascaded implementation could have several local minima, changing the modality of the LS error surface.

To characterize the change in the modality of the cost function, we introduce the following adapted Lemma from [26].

Lemma 4. *Let a mapping $\varphi(\cdot) : \mathcal{D}_d \rightarrow \mathcal{D}_c$ exist and be continuous and surjective (onto), then all the newly formed stationary points (critical points) are saddle points.*

Proof. See Appendix 5.7.1. □

The previous Lemma guarantees that the original critical points are maintained and that all new critical points introduced are saddle points if a proper reparametrization is used. Here, proper refers to the properties that the mapping φ must exhibit. Unfortunately, as shown next, these properties cannot be guaranteed for all types of GFs. This fact is made formal in the next result.

Theorem 5.1. *The continuous and surjective map φ exists for cascaded GFs whose modules are first-order C-GFs.*

Proof. See Appendix 5.7.2. □

Corollary 1. *The map φ also exists if instead of C-GF modules, ARMA C-GF modules, see, e.g., [27], are used to implement the cascaded GF.*

SADDLE POINTS OF CASCADED C-GFs

Due to the previous results, only C-GFs allow for continuous and surjective mappings, and hence for the creation of critical points that are not global minima. Fortunately, these critical points are saddle points. The following result identifies where these saddle points, due to the cascaded reparametrization, appear in the parameter space.

Theorem 5.2. *Consider a direct implementation of a C-GF*

$$\mathbf{H} = \sum_{k=0}^K \phi_k \mathbf{S}^k, \text{ with } \phi_0 = 1,$$

and a cascaded GF $\mathcal{H}(\mathbf{S}; \boldsymbol{\theta})$, of the form (5.2) with $Q = K$ modules. Further, let the modules be given by

$$\mathcal{F}_c^1(\mathbf{S}; \boldsymbol{\theta}_q) = \mathbf{I} + \phi_1^{(q)} \mathbf{S}, \forall q \in [Q].$$

Then, the newly introduced saddle points in (5.3), with respect to a desired response \mathbf{H}^* , are found in a manifold where any two or more graph filter coefficients are the same.

Proof. See Appendix 5.7.3. □

Corollary 2. *The above result also holds in the case that the modules are ARMA₁ C-GFs, i.e.,*

$$\mathcal{F}_c^1(\mathbf{S}; \boldsymbol{\theta}_q) = (\mathbf{I} + \psi_1^{(q)} \mathbf{S})^{-1} (\mathbf{I} + \phi_1^{(q)} \mathbf{S}), \forall q \in [Q].$$

The result of Theorem 5.2 shows that saddle points only appear when two modules (stages) of C-GFs have the same coefficients. This is in line with the results of system identification stating that the modality of the error surface for cascaded IIR filters are affected by multiple poles or zeros [28]. In addition, although the result of Theorem 5.2 seems restrictive, i.e., it is stated for C-GFs of order one, the theorem can be extended to second- (or higher-) order modules. However, for the sake of exposition, we here consider first-order modules as they are the simplest GF units and they are the basis for graph neural networks (GNNs) and residual GNNs (RGNNs), see, e.g., [29]. Further, the restriction to $\phi_0 = 1$ is w.l.o.g. as for any other value different from unity, the same result can be obtained by proper scaling.

To conclude this section, we bring to attention the following. First, the fact that only saddle points are introduced when a C-GF is reparametrized as a cascade of C-GFs [cf. Theorem 5.2] indicates that *stochastic* descent methods, e.g., stochastic gradient descent (SGD) or LMS, are good candidates for minimizing the cost in (5.3). The stochastic nature of such algorithms provides a natural protection against saddle points. This is because the jitter present in these methods allows them to escape from saddle points. This key observation is the working assumption for the optimization methods used in state-of-the-art machine learning methods for optimizing nonconvex costs, see, e.g., [30]. Second, the negative result obtained for other kinds of GFs [cf. Theorem 5.1] suggests that although traditional descent methods might work “sufficiently well”, we should put efforts on devising methods that are not only *globally convergent* but that are *efficient*. This last aspect motivates the algorithm development presented in Section 5.3. There we propose an iterative method, amenable for sparse iterative solvers and deep learning optimization algorithms, to fit/learn the filter coefficients of (5.2).

Before introducing the proposed coefficient learning algorithm, in the following part, we present the relation between the proposed cascaded GF implementation and graph convolutional networks.

5.2.2. RELATION TO GRAPH CONVOLUTIONAL NETWORKS

In recent years, machine learning over graphs has drawn an increasing amount of attention [31–33]. However, despite that the graph structure provides a highly informative prior, the task of learning over graphs still remains highly complex due to this same structure. As a result, *graph convolutional neural networks* (GCNNs), which leverage the structural information of the graph, have been put forth [20]. GSP and more specifically graph filtering provide a formal understanding of the basic operations involved in GCNNs [34].

Formally, a GCNN is a neural network that operates on graph data and whose *hidden layers* can be represented as

$$\mathbf{X}^{(i)} = \rho(\mathbf{S}; \mathbf{X}^{(i-1)}), \quad (5.5)$$

where $\mathbf{X}^{(0)}$ is the original $N \times F^{(0)}$ input data (data matrix \mathbf{X}), \mathbf{S} the GSO and ρ a propagation rule [35]. Thus, the hidden layer $\mathbf{X}^{(i)}$ can be interpreted as an $N \times F^{(i)}$ feature matrix whose i th row represents the features of the i th node. Under this setting, by stacking Q different layers, i.e.,

$$\mathcal{H}_{\text{gcnn}}(\mathbf{S}; \mathbf{X}^{(0)}) := \rho(\mathbf{S}; \rho(\mathbf{S}; \rho(\mathbf{S}; \dots \rho(\mathbf{S}; \mathbf{X}^{(0)}) \dots))), \quad (5.6)$$

we obtain a so-called GCNN. Note that different flavors of GCNNs can be obtained by choosing different alternatives for the propagation rule. Typically, the propagation rule is selected as

$$\rho(\mathbf{S}; \mathbf{X}^{(i)}) := \sigma(\mathbf{S}\mathbf{X}^{(i)}\mathbf{W}^{(i)}), \quad (5.7)$$

where $\sigma(\cdot)$ is an element-wise non-linearity, e.g., a linear rectifier unit (ReLU), and $\mathbf{W}^{(i)}$ a weight matrix to combine the features of the nodes. From (5.7), we observe that the number of features at the $(i+1)$ th layer is defined by the number of columns of $\mathbf{W}^{(i)}$ and hence $\mathbf{W}^{(i)}$ is an $F^{(i)} \times F^{(i+1)}$ matrix.

Observing the expression involved in a GCNN, we can easily draw connections between the model (5.2) and a GCNN. For the filtering task, we are mostly concerned with one-dimensional signals at each node, hence the number of features at *all layers* can remain constant and equal to one. Thus, the weight matrices $\{\mathbf{W}^{(i)}\}$ are reduced to simple scalars. Further, despite that a single shift with respect to the GSO captures certain structural properties, a GF is able to highlight different (and possibly more complex) properties of the graph structure in a parsimonious manner, i.e., through a parametrized representation. Further, considering that we want to build a linear model-driven system, i.e., in most GF applications a desired linear operator is given to be approximated, the element-wise non-linearity $\sigma(\cdot)$ is not required and an identity function suffices, i.e., $\sigma(x) = x$. As a result, by using these considerations, we can see that (5.6) reduces to (5.2), where each layer is a first-order C-GF and the input is the graph signal \mathbf{x} .

5.3. RELIEF ALGORITHM

Let us consider again the model for the cascaded GF implementation

$$\mathcal{H}(\mathbf{S}; \Theta) = \prod_{i=1}^Q \mathcal{F}_*^K(\mathbf{S}; \theta_i), \quad (5.8)$$

where the parameter dependency has been stated explicitly, and $\Theta := [\theta_1^\top, \dots, \theta_Q^\top]^\top \in \mathbb{R}^{Q \text{DoF}}$ is the parameter vector of the cascaded GF implementation.

The *general filter design* task consists of the following optimization problem: given a linear operator $\mathbf{H}^* \in \mathbb{R}^{N \times N}$ find

$$\Theta^* = \underset{\{\theta_i \in \mathcal{C}\}_{i=1}^Q}{\operatorname{argmin}} \quad \|\mathcal{H}(\mathbf{S}, \Theta) - \mathbf{H}^*\|, \quad (5.9)$$

where \mathcal{C} is a desired feasible set, e.g., interval, normed ball, etc., and $\|\cdot\|$ a desired norm, i.e., spectral normal, Frobenious norm. Notice that because the operator to approximate is linear, and that there are no data-dependent non-linearities in \mathcal{H} , *input/output data* is not required for solving (5.9). This type of design is what we refer to as *model-based design*.

As discussed before, one of the main challenges for finding the parameter vector Θ^* , even for a simple (or convex) feasible set \mathcal{C} , is that its components interact in the cost function of (5.9) in a non-convex manner. Therefore, the cost function is generally a multi-modal function with several local minima and off-the-shelf convex solvers can not be employed directly.

An alternative to deal with the non-convexity of the cost function in (5.9) but still being able to use readily available efficient convex solvers, is to perform *sequential fitting* for each of the GF modules. That is, given a certain tolerance, i.e., error between the fitted GF and desired linear operator, we start by solving

$$\theta_1^* = \underset{\theta_1 \in \mathcal{C}}{\operatorname{argmin}} \quad \|\mathcal{F}_*^K(\mathbf{S}; \theta_1) - \mathbf{H}^*\|, \quad (5.10)$$

and proceed with the remaining filters by sequentially solving

$$\theta_q^* = \underset{\theta_q \in \mathcal{C}}{\operatorname{argmin}} \quad \|\mathcal{F}_*^K(\mathbf{S}; \theta_q) [\prod_{i=1}^{q-1} \mathcal{F}_*^K(\mathbf{S}; \theta_i^*)] - \mathbf{H}^*\|, \quad (5.11)$$

This way, a solution $\hat{\Theta}$ within the desired tolerance can be obtained.

Although this is a straightforward approach, it has several drawbacks. For instance, in this approach each filter is fitted once, hence if early stages result in a bad fit this cannot be corrected for in later stages.

To alleviate this issue, we propose to fit the filters using ideas similar to the ones employed in the RELAX method for mixed-spectrum estimation [36], where in every stage, after a set of parameters has been estimated, the old set of parameters are refitted to improve the cost function value. However, instead of refitting all GFs every time a new GF is added, as in the case of RELAX, we restrict ourselves to the left and right most filters. Hence, the name of the method: right (REchts) -left (LInks) itErative Fitting (RELIEF). In

addition, to avoid the explicit computation of the inverse filters, we use a sparse construction (which allows a matrix free implementation) for solving the LS design.

In the following, we first introduce the sparse construction for fitting individual GFs (`linSparseSolve` routine) and then the refitting routine to improve the cost function value after each stage of the algorithm (`refitPair` routine). A summary of the proposed method is shown in Algorithm 1.

5.3.1. `linSparseSolve` ROUTINE

Without loss of generality, let us focus on the cascaded GF implementation of order q given by

$$\mathcal{H}(\mathbf{S}; \Theta) = \mathbf{H}_1 \mathbf{M} \mathbf{H}_r, \quad (5.12)$$

where $\mathbf{H}_r := \mathcal{F}_*^K(\mathbf{S}; \theta_1)$, $\mathbf{H}_1 := \mathcal{F}_*^K(\mathbf{S}; \theta_q)$ and

$$\mathbf{M} := \prod_{i=2}^{q-1} \mathcal{F}_*^K(\mathbf{S}; \theta_i). \quad (5.13)$$

The minimizer of problem (5.11) for \mathbf{H}_1 with \mathbf{M} and \mathbf{H}_r fixed, assuming $\mathcal{C} = \mathbb{R}^N$, can be shown to be given by the solution of the least squares problem

$$\underset{\theta_q}{\operatorname{argmin}} \|\Omega_1 \theta_q - \operatorname{vec}(\mathbf{H}^*)\|_2, \quad (5.14)$$

where $\Omega_1 := (\mathbf{H}_r^\top \mathbf{M}^\top \otimes \mathbf{I}) \Psi$ with \otimes the Kronecker product, $\operatorname{vec}(\cdot)$ is the vectorization operation and

$$\Psi := [(\mathbf{I} \otimes \mathbf{I})\mathbf{J}, (\mathbf{S}^\top \otimes \mathbf{I})\mathbf{J}, \dots, ((\mathbf{S}^\top)^K \otimes \mathbf{I})\mathbf{J}] \in \mathbb{R}^{N^2 \times K \operatorname{DoF}}, \quad (5.15)$$

is the GF system matrix, with \mathbf{J} an $N^2 \times \operatorname{DoF}$ selection matrix that only preserves the nonzero entries of $\operatorname{vec}(\Phi_k)$. Notice that by construction, the Kronecker matrix is a *sparse matrix* and for low filter orders, i.e., below the diameter of the graph, the Ψ matrix is also sparse. Hence, it can be stored efficiently in memory if it is desired to construct Ω_1 explicitly. Least squares problems as (5.14) can be easily solved by methods such as LSMR [37]. In addition, if Ω_1 is explicitly computed, we can build a preconditioner to accelerate the convergence of LSMR by scaling the columns of Ω_1 such that every column has unit 2-norm. For very large systems, we may prefer to keep Ω_1 as an *operator*. That is, we avoid its construction explicitly and only provide a routine which computes the actions $\Omega_1 \mathbf{x}$ and $\Omega_1^\top \mathbf{x}$ for an arbitrary vector \mathbf{x} . Algorithms as LSMR can still be employed in this setup, however, the preconditioning of the system is not simple unless the norm of the columns of Ω_1 can be estimated accurately. Note that, as we assume that (5.14) is not necessarily consistent, i.e., large relative tolerance, LSMR is preferred over the popular LSQR method [38] due to its faster convergence. In Algorithm 1, the routine `linSparseSolve(A, b)` makes reference to the procedure of solving a least squares problem $\mathbf{A} \mathbf{x} \approx \mathbf{b}$ such as (5.14) by means of LSMR. For completeness, the LSMR algorithm, specialized for solving problems of the form (5.14), is provided in Appendix 5.7.4.

A similar system can be obtained for fitting \mathbf{H}_r for a fixed \mathbf{H}_1 and \mathbf{M} by making minor changes. Hence, for sake of space, we omit this derivation.

Algorithm 1: RELIEF Algorithm

Result: $\{\theta_i\}_{i \in [Q]}$: filter parameters
Input: \mathbf{H}^* , Ψ , Q , \maxIt , ϵ_{tol}
initialization: $\theta_i = \mathbf{0}$, $\forall i \in [Q]$, $q = 0$, $\mathbf{M} = \mathbf{H}_1 = \mathbf{I}$, $\mathbf{h}^* = \text{vec}(\mathbf{H}^*)$;
while ($\epsilon > \epsilon_{tol}$) & ($q < Q$) **do**
 $q = q + 1$;
 $\mathbf{H}_q \leftarrow \text{linSparseSolve}([\mathbf{H}_1^\top \mathbf{M}^\top \otimes \mathbf{I}] \Psi, \mathbf{h}^*)$;
 if $q > 1$ **then**
 $(\mathbf{H}_1, \mathbf{H}_q) \leftarrow \text{refitPair}(\mathbf{H}^*, \Psi, \mathbf{H}_q, \mathbf{H}_1, \mathbf{M}, \maxIt, \epsilon_{tol})$;
 $\mathbf{M} \leftarrow \mathbf{H}_q \mathbf{M}$
 end
 $\mathbf{H}_{total} \leftarrow \mathbf{M} \mathbf{H}_1$;
 $\epsilon \leftarrow \|\mathbf{H}_{total} - \mathbf{H}^*\|_F^2$;
end

5.3.2. refitPair ROUTINE

Similar to RELAX, we would like to *correct* for a possibly wrong fitting, without going to the extreme of doing a multi-block block-coordinate-descent (BCD) [39], where only a set of GFs are fitted at a time, while keeping the rest constant and then iterating until convergence (if achieved). Although such an approach is possible, it has some drawbacks. First, to the best of our knowledge, for multi-block BCD there are no guarantees on the behaviour of limit points of the generated sequences in the general case. Hence, to guarantee convergence, i.e., each subproblem is convex [39], each block has to be updated individually. Second, even if individual GF updates are considered, the fitting of that many GFs and the possibly slow convergence renders the approach unattractive. And third, if a direct solver with an explicit system matrix is used, the sparsity of the system matrix is lost.

Therefore, instead of iterating over all filters, at every step, we propose to refit the left- and right-most filters [cf. (5.12)]. This not only allows to have convergence guarantees, i.e., two-block BCD convergence is proved in [40], but also to have a reduced number of subproblems, which (i) effectively reduces the complexity of the overall approach and (ii) makes use of the sparsity of the involved system matrices. A summary of this routine is shown in Algorithm 2.

5.4. DATA-DRIVEN RELIEF

Though model-driven designs are appealing because they do not require input/output data, in many cases, the only information available of the underlying transform is given through input-output pairs, $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^P$. Examples of such scenarios appear in system network identification, e.g., [41, 42], or graph identification tasks, see, e.g., [13, 43, 44]. Therefore, there is a need to put forth a *data-driven method* for cascaded GF coefficient *learning*.

Formally, a data-driven design for the cascaded GF in (5.2) and in the unconstrained

Algorithm 2: refitPair Routine

Result: (θ_l, θ_r) : filter parameters
Input: $\mathbf{H}^*, \Psi, \mathbf{H}_l, \mathbf{H}_r, \mathbf{M}, \text{maxIt}, \epsilon_{\text{tol}}$
 initialization: $\text{numIt} = 0, \mathbf{h}^* = \text{vec}(\mathbf{H}^*)$;
while $(\epsilon > \epsilon_{\text{tol}}) \ \& \ (\text{numIt} < \text{maxIt})$ **do**
 $\text{numIt} = \text{numIt} + 1$;
 $\mathbf{H}_r \leftarrow \text{linSparseSolve}([\mathbf{I} \otimes \mathbf{H}_l \mathbf{M}] \Psi, \mathbf{h}^*)$;
 $\mathbf{H}_l \leftarrow \text{linSparseSolve}([\mathbf{H}_r^\top \mathbf{M}^\top \otimes \mathbf{I}] \Psi, \mathbf{h}^*)$;
 $\epsilon \leftarrow \|\mathbf{H}_l \mathbf{M} \mathbf{H}_r - \mathbf{H}^*\|_{\text{F}}^2$;
end

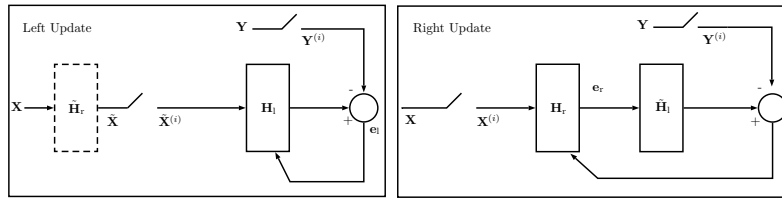


Figure 5.2: Schematic of proposed refitting update procedure for the proposed staggered RELIEF (left) Update cycle for the left coefficients. (right) Update cycle for the right coefficients. Here, $\mathbf{X}^{(i)}$ and $\mathbf{Y}^{(i)}$ are the i th input- and output-data batches, respectively; and \mathbf{e}_l and \mathbf{e}_r the error signals required for gradient computations.

5

case, solves the following optimization problem

$$\Theta^* = \underset{\{\theta_i\}_{i=1}^Q}{\text{argmin}} \sum_{i=1}^P \|\mathbf{y}_i - \mathcal{H}(\mathbf{S}; \Theta) \mathbf{x}_i\|_2^2 \quad (5.16)$$

Despite that (5.16) can be solved by deterministic variants of descent methods, or by substituting the cascaded GF by a higher-order GF, there are several issues that make these approaches not so attractive. First, it is well-known from the time-domain literature, see, e.g., [18, 26], that the sensitivities of coefficients of cascaded structures are much lower than those of direct forms; that is, the deviation of the magnitude response due to a small change in the coefficients of cascaded forms is much smaller than for direct forms. Second, deterministic approaches to solve (5.16) might lead to problems of high memory and computational cost as for large P , i.e., a large number of data pairs, computing the full gradient might be prohibitive. Plus, batch-processing benefits, e.g., parallelization, are not leveraged.

In addition to above issues, even a direct solution, i.e.,

$$\Theta^* = \underset{\{\theta_i\}_{i=1}^Q}{\text{argmin}} \|\mathbf{Y} \mathbf{X}^\dagger - \mathcal{H}(\mathbf{S}; \Theta)\|_{\text{F}}^2 \quad (5.17)$$

where $\mathbf{Y} = [\mathbf{y}_1, \dots, \mathbf{y}_P]$ and $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_P]$, is not desirable due to the initial effort of finding the pseudoinverse of the input data, i.e., \mathbf{X}^\dagger .

In addition, due to the result of Theorem 5.2, when the cascaded implementation and a stochastic (batch) variant of gradient descent is employed, such as LMS or SGD, there is no risk to be stuck in a saddle point due to the jitter present in the method. Hence, in the following, we introduce a batch version of RELIEF for the data-driven setting.

5.4.1. STAGGERED RELIEF

To adapt RELIEF to the data-driven setting, let us consider the original three-way factorization of the cascaded GF used by RELIEF in the matrix version of the cost in (5.16), i.e.,

$$\|\mathbf{Y} - \mathbf{H}_l \mathbf{M} \mathbf{H}_r \mathbf{X}\|_F^2. \quad (5.18)$$

From (5.18), we observe that now there is a matrix to the right of \mathbf{H}_r which means that the sparsity present in the system matrix exploited in RELIEF is lost. Hence, some computational benefits of the approach are lost. In addition, as discussed in (5.17), although directly taking the pseudoinverse of the input data matrix would allow for a direct application of RELIEF it requires a costly inversion. Therefore, we put forth a small variant of RELIEF to cope with these details and allow for batch updates.

To tackle the problems with a vanilla implementation of RELIEF in the data-driven setting, we propose the following updating scheme for the left and right GF coefficients. First, we replace the routine `linSparseSolve` in Algorithm 1 by the routine `descentMethod`. Here, the routine `descentMethod` is any stochastic variant of a descent method, e.g., SGD, RMSprop [45], Adam [46], etc., working under mini-batch assumptions and allowing a scheduler for step-size and batch-size. The latter requirement is to leverage the benefits of parallelization and a reduction of coefficient updates due to proper batch size scheduling, see, e.g., [47]. Second, the routine `refitPair` is substituted by a process summarized in Fig. 5.2. In this process, we refit and update, in two stages, the left and right coefficients, similar to the model-driven RELIEF. As seen in Fig. 5.2, both the left and right updates have a feedback loop of the error. Hence, they are also implemented as descent methods. However, due to the linearity of the operations, i.e., the transform is data independent, we can collapse, at every iteration, the right chain into a single transformation; that is, we define

$$\tilde{\mathbf{H}}_r = \mathbf{M} \mathbf{H}_r, \quad (5.19)$$

and process (possibly in parallel) the input to obtain the “filtered” version $\tilde{\mathbf{X}} := \tilde{\mathbf{H}}_r \mathbf{X}$. Though this step might be seen as trivial, it avoids multiplication-induced overheads that might appear if each GF is defined as a *layer* in a deep learning framework e.g., as in Tensorflow. This filtered version can then be used as input to the `descentMethod` routine to adjust the coefficients of the left-most GF for several epochs, i.e., complete passes over the data. Similarly, the left chain can be collapsed to form

$$\tilde{\mathbf{H}}_l = \mathbf{H}_l \mathbf{M}, \quad (5.20)$$

again, due to the linear nature of the model, and use the `descentMethod` routine to adjust now the coefficients of the right-most GF for several epochs. This refitting procedure, consisting of the left and right updates depicted in Fig. 5.2, is performed for several rounds or until a convergence criterium is met. Due to the collapsing and alternating updates, we refer to this method as staggered RELIEF.

Remark 5.1. *Although, in theory, the linear operators (5.19) and (5.20) should be no different than a “layered” implementation, i.e., an implementation where every batch sequentially undergoes the linear operators, we make the reader aware that, at implementation time, there might be differences due to rounding errors and the selected floating-point representation of the involved matrices. Hence, results might slightly vary but not considerable.*

5.4.2. SOME WORDS ABOUT (FULL) BACKPROPAGATION

A straightforward way to solve (5.16), including the previously discussed aspects, is *backpropagation* (BP) [48] equipped with a stochastic descent method. Within signal processing and neural networks (NNs), BP is a celebrated method for updating the coefficients of the different layers composing a NN. However, although BP can find the global optimum of (5.16) for cascaded structures employing C-GFs as modules, based on the result from Theorem 5.2, for other GF structures, it might incur a high computational cost due to the (possibly) large number of parameters per stage and the full backpropagation of the gradient. Moreover, due to the increased sensitivity of the coefficients, the learning has to be done with a small step size which directly affects the convergence of the method. In NNs, this last issue is usually tackled by the nonlinearities and the batch normalization layers present in the architectures, see, e.g., [49], which are not present in the cascaded structure. In addition, differently from the staggered RELIEF, the need to know beforehand the number of modules to use, i.e., Q is a hyper-parameter that needs to be set, and the large number of parameter updates, i.e., matrix-vector products, that have to be made when learning rates (descent steps) are small make this approach not as appealing as the proposed staggered RELIEF. However, despite that we advocate the use of a variant of RELIEF instead of full BP for finding the coefficients of the involved GFs, we do encourage to use full BP steps for fine-tuning after an initial coefficient configuration has been found using the proposed data-driven staggered RELIEF.

5.5. NUMERICAL RESULTS

To illustrate the performance of the introduced cascaded structure, in the following, we present a series of numerical experiments covering the theoretical aspects discussed in the chapter as well as the model- and data-driven application of RELIEF.

5.5.1. ERROR SURFACES AND ROOT ANALYSIS

First, we present an example that illustrates the results of Theorems 5.1 and 5.2. Here, we consider a simple example for a sensor network with $N = 32$ nodes constructed using the GSPToolbox [50]. The GSO considered in this example is the combinatorial Laplacian of the network. Here, we consider a 2nd-order C-GF with coefficients $\boldsymbol{\theta} = [0.369, -1.53, 1]^\top$. The roots of the corresponding polynomial are $[0.3, 1.23]$. In Fig. 5.3, we show the corresponding error surfaces for the different implementations, i.e., (left) direct and (right) cascaded. In this experiment, we generated 1000 Gaussian-distributed graph signals, i.e., $\mathbf{X} \in \mathbb{R}^{N \times 1000}$, and filtered them using the above mentioned C-GF. The filtered graph signals are then perturbed with additive white Gaussian noise whose standard deviation is $\sigma = 10^{-3}$, thus the observations follow the model $\mathbf{Y} = \mathbf{H}\mathbf{X} + \mathbf{N}$, where \mathbf{N} denotes the addi-

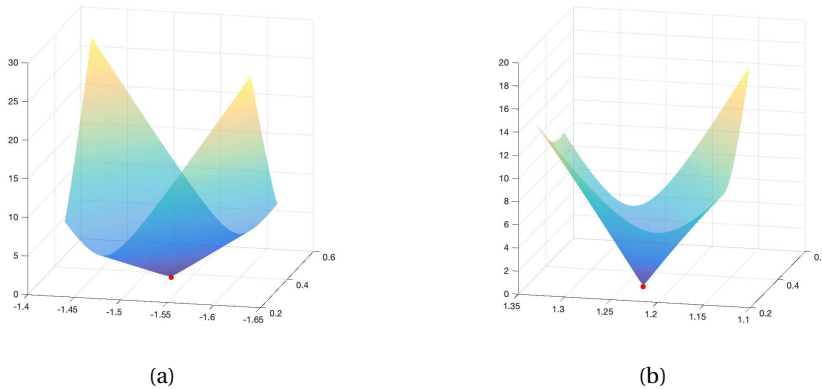


Figure 5.3: Comparison of the error surface profiles of the (a) direct, and (b) cascaded implementation for a C-GF with roots $[0.3, 1.23]$ for a sensor network graph with $N = 32$ nodes.

tive noise and \mathbf{H} the applied C-GF. The error surface shown in Fig. 5.3 is then the norm of the fitting error, e.g., $\|\mathcal{H}(\boldsymbol{\Theta})\mathbf{X} - \mathbf{Y}\|_F$. While for the direct implementation, the cost is known to be convex [cf. Fig. 5.3a], the minimizer has not changed for the cascaded implementation as discussed in Theorems 5.1 and 5.2. Further, the profile of the cost function is also different. That is, although no saddle points were introduced the gradient towards the minimum in Fig. 5.3b can be seen larger than that of Fig. 5.3a. In particular, in one of the dimensions (coefficients) of the direct implementation the cost function exhibits an elongated behavior, i.e., different sensitivity for each coefficient. This simple example illustrates the theoretical results derived in this chapter with respect to the cascaded implementation of C-GFs, which are consistent with well-known results within signal processing and adaptive filtering.

Now, we briefly discuss one of the problems of the cascaded implementation, namely, close and mirrored roots. As discussed before, the error surface of the cascaded implementation exhibits amenable properties when there are no repeated roots. However, when the roots are close to each other and mirrored, i.e., pair of complex conjugate roots, the chances that the roots move to the area where a singularity appears, i.e. repeated roots, increases. Thus, the cascaded implementation might exhibit problems finding the appropriate roots in these cases or a slower convergence. To show an example of this behaviour, let us consider a C-GF whose coefficients are $\boldsymbol{\theta} = [5.2, 0.23, 1.21, 0.45, 0.65]$. In this case, the roots of the corresponding 4th-order polynomial (up to 3 digits of precision) are $[0.336 + 0.543i, 0.336 - 0.543i, -0.358 + 0.421i, -0.358 - 0.421i]$. A comparison of the fitting error between the direct and cascaded implementation, for this case, is shown in Fig. 5.4. In Fig. 5.4a two things can be observed. First, the direct implementation obtains a better fit (loss) and with a much faster rate. Second, the cascaded implementation enters a *plateau* early on but escapes, i.e., it keeps descending. The plateau is reached when the root estimates collapse on the real axis and have similar values (see Fig. 5.4b). After the roots meet, they move away from the real axis and move towards the respective conjugate

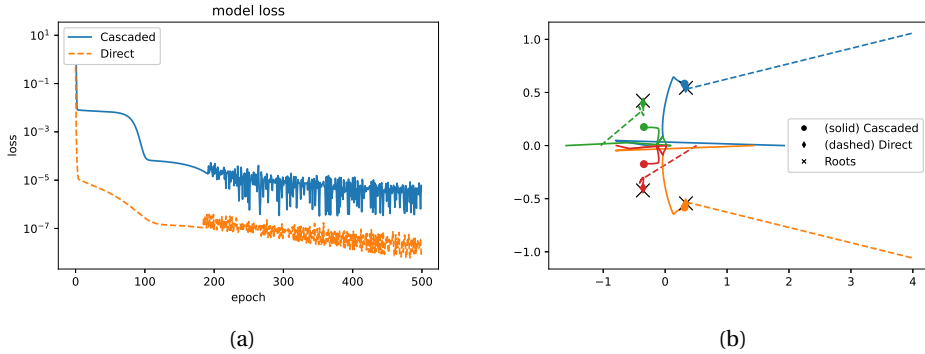


Figure 5.4: Comparison of the fitting performance for the direct and cascaded implementation of a C-GF with roots $[0.336 + 0.543i, 0.336 - 0.543i, -0.358 + 0.421i, -0.358 - 0.421i]$. (a) Evolution of fitting error per epoch. (b) Trajectory of the estimated filter polynomial roots.

pairs. As discussed before in this chapter, when a stochastic method is used to perform the optimization, the cascaded implementation is able to escape from saddle points without much trouble. Notice here that while at the 500th iteration, the direct implementation is really close to the original roots, the cascaded implementation still requires more iterations to reach them. In contrast with this result, we have the experiment shown in Fig. 5.5. In this case, another C-GF with roots $[-1, 0.4, 1, -0.4]$ is identified through data. Differently from the previous case, the cascaded implementation avoids the collapse of the roots as seen in Fig. 5.5b leading to a better fitting than the direct implementation and at a faster rate, see Fig. 5.5a. Thus, this illustrates that when areas near singularities are not reached by the root estimates, the cascaded implementation can outperform the fitting of the direct implementation even for relatively low GF orders. For a fair comparison, in these examples, we have employed a common rule for the step size of the stochastic gradient descent in both cases, particularly $\delta_k = 2/(k+3)$, where k denotes the epoch. A more careful selection of the step size rule could have reduced the jitter observed in Fig. 5.4, however, the trend is clear from the plot. Here, we have used a $\text{batchsize} = 2000$ and a sensor network with $N = 64$ nodes. The GSO for constructing these C-GFs was the normalized Laplacian of the network, and the number of samples used for fitting the models was 132000.

5.5.2. RELIEF APPLICATION: DISTRIBUTED CONSENSUS

To demonstrate the performance of the model-driven RELIEF method, we study the problem of consensus over networks [51]. That is, given a network (graph), we desire to compute the average of the signal over the network, i.e.,

$$\mathbf{y} = \frac{1}{N} \mathbf{1} \mathbf{1}^T \mathbf{x} = \mathbf{H}^* \mathbf{x}, \quad (5.21)$$

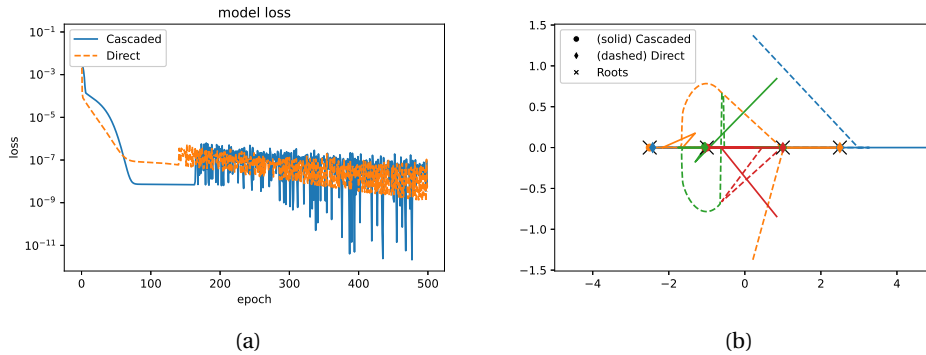


Figure 5.5: Comparison of the fitting performance for the direct and cascaded implementation of a C-GF with roots $[-1, 0.4, 1, -0.4]$. (a) Evolution of fitting error per epoch. (b) Trajectory of the estimated filter polynomial roots.

where $\mathbf{1}$ is the N -dimensional all-one vector. To do so, we explore the proposed implementations on different network sizes and base GFs, i.e., C-GF, NV-GF, and CEV-GF. In these experiments, for small problem sizes, i.e., $N < 300$, we employ direct solvers while for larger problem sizes LSMR based solvers. In particular, for $N = 400$ a solver with a non-explicit system matrix has been implemented. For $N = 500$ a standard LSMR solver, with explicit system matrix and diagonal preconditioning, has been used.

In Fig. 5.6a, Fig. 5.6c and Fig. 5.6e a comparison for respectively $N = 100$, $N = 400$ and $N = 500$ in terms of model error, i.e., $\|\mathcal{H}(\mathbf{S}, \Theta) - \mathbf{H}^*\|_{\mathbb{F}}^2$, is shown for different types of GFs and communication exchanges, i.e., data exchanges between neighbors. Here, *direct* and *cascaded* refer to a direct implementation, i.e., the order of the filter is equal to the number of communication rounds, and the proposed cascaded implementation, respectively. In all these figures, we observe that the cascaded implementation outperforms the direct implementation, and the best performance in terms of error is achieved by the CEV-GF.

To further test the proposed implementation, we compare the trained cascaded GFs with the primal-dual method of multipliers (PDMM)¹ [52] for network consensus. In Fig. 5.6b and Fig. 5.6d, we observe that the convergence speed of PDMM (in terms of communication rounds) is not comparable with the one obtained by the cascaded CEV-GF, despite that PDMM eventually guarantees consensus. This implies that by an adequate selection of the weight matrices, consensus can be achieved (up-to machine precision) in a low number of steps. Here, the reported error is the *action* error, i.e., $\|\mathcal{H}(\mathbf{S}, \Theta)\mathbf{x} - \mathbf{H}^*\mathbf{x}\|_2^2$.

Finally, we perform a test illustrating the performance of the data-driven staggered RELIEF using the GCNN framework for coefficient learning. In this example, we fit the same consensus operator on a community network with $N = 200$ nodes. For the GCNN-based solver, the number of filters (layers) has been set to $Q = 15$, and a set of $M = 10^3$ randomly generated data pairs has been generated. The batch size has been set to 3200 and the learning rate to 0.001. In Fig. 5.6f, the results for this experiment are shown. Similar as

¹PDMM is an alternative distributed optimization tool to the classical alternating direction method of multipliers (ADMM), which is often characterized by a faster convergence.

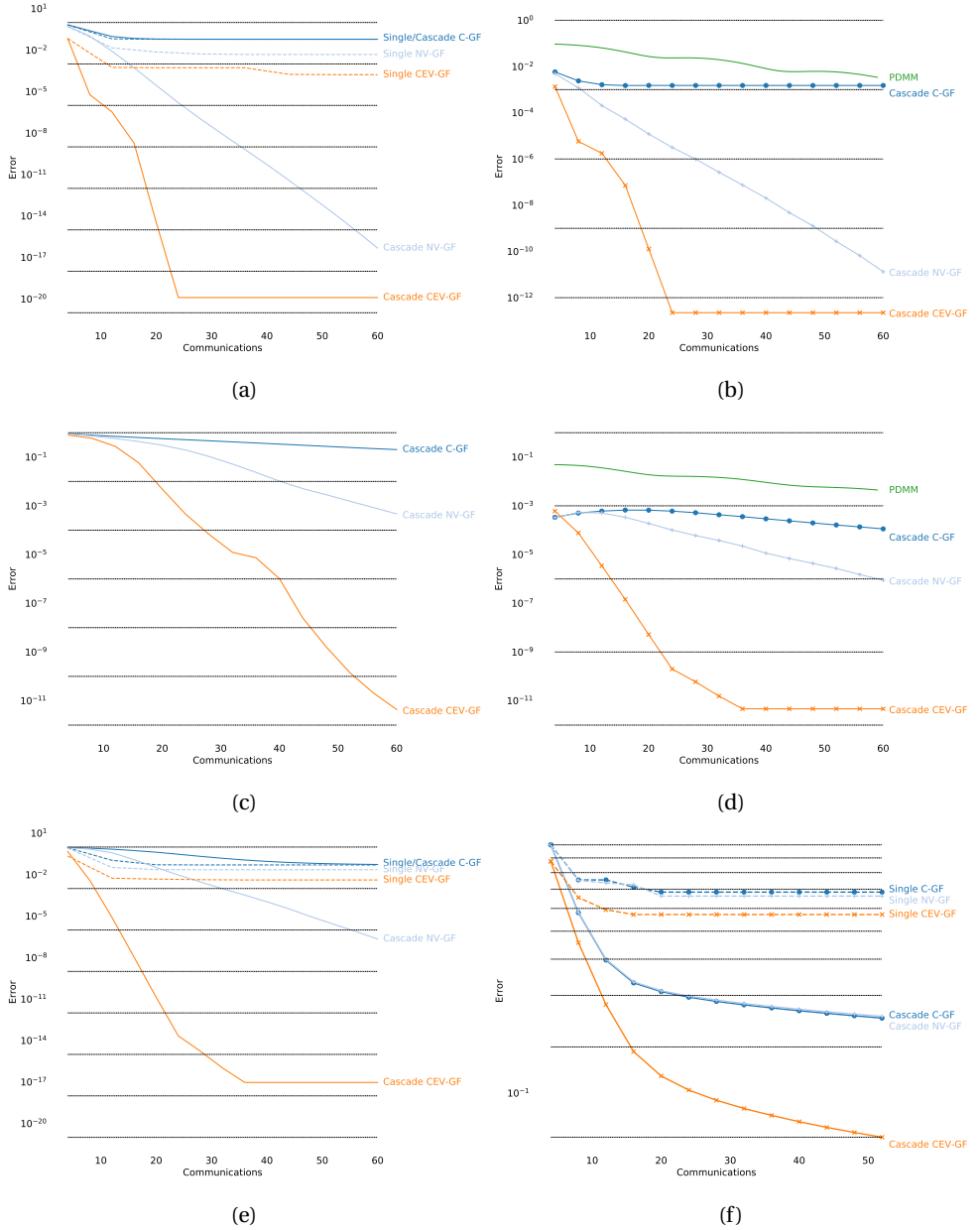


Figure 5.6: (Left) Model error comparison, i.e., $\|\mathcal{R}(\mathbf{S}, \Theta) - \mathbf{H}^* \|_{\mathbb{F}}^2$, of direct and cascaded GF implementations for (a) direct solver for $N = 100$, (c) operator-based (no-explicit system matrix) solver for $N = 400$ and (e) LSMR solver with explicit system matrix and diagonal preconditioning (Jacobi) for $N = 500$. (Right) Action error comparison, i.e., $\mathbb{E}\{\|\mathcal{R}(\mathbf{S}, \Theta)\mathbf{x} - \mathbf{H}^*\mathbf{x}\|_2^2\}$, for different GFs and PDMM for (b) $N = 100$ and (d) $N = 500$. (f) Error comparison for single and cascaded GF implementation optimized with GCNN framework.

in the other experiments, we observe that the cascaded implementation outperforms the direct implementation, and that among all the types of GFs, the CEV-GF obtains the best performance over a randomly generated test set. Here, as the GCNN framework is data dependent, the error measure reported is again the action error.

5.6. CHAPTER SUMMARY

In this chapter, we have introduced a cascaded implementation of generalized graph filters and we have drawn connections with neural network architectures for graph-supported data. Furthermore, we proposed an efficient stable algorithm for designing/learning the GF coefficients, which can be implemented leveraging state-of-the-art sparse solvers and preconditioning techniques in the model-driven case. In addition, by minor modifications, we showed that the proposed method can be applied in the data-driven setting as well and that the deep learning framework can be combined with the proposed algorithm to fit the involved GFs when data is available or easy to generate. Finally, we illustrated the applicability of the proposed implementations and design algorithms in a network consensus application, where we have shown that CEV-GF filters achieve machine-precision accuracy at a significantly lower communication costs than existing methods.

5.7. APPENDIX

5.7.1. PROOF LEMMA 4

To prove the result, we first recall that

$$\frac{\partial f(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \frac{\partial h(\boldsymbol{\Theta})}{\partial \boldsymbol{\theta}} \quad (5.22)$$

$$= \frac{\partial h(\boldsymbol{\Theta})}{\partial \boldsymbol{\Theta}} \frac{\partial \boldsymbol{\Theta}}{\partial \boldsymbol{\theta}}, \quad (5.23)$$

where $\partial f(\boldsymbol{\theta})/\partial \boldsymbol{\theta}$, $\partial h(\boldsymbol{\Theta})/\partial \boldsymbol{\theta}$ and $\partial \boldsymbol{\Theta}/\partial \boldsymbol{\theta}$ are the function gradients and the parameter Jacobian matrix, respectively, with respect to $\boldsymbol{\theta}$. Now, assume that $\boldsymbol{\Theta}^*$ is a critical point of $h(\boldsymbol{\Theta})$, i.e.,

$$\left. \frac{\partial h(\boldsymbol{\Theta})}{\partial \boldsymbol{\Theta}} \right|_{\boldsymbol{\Theta}^*} = 0. \quad (5.24)$$

By assumption, there exists a $\boldsymbol{\theta}^*$ such that $\varphi(\boldsymbol{\theta}^*) = \boldsymbol{\Theta}^*$ (due to the existence and surjectiveness of φ). Hence, by (5.22), $\boldsymbol{\theta}^*$ is a critical point of $f(\boldsymbol{\theta})$ and has the same nature as that of $\boldsymbol{\Theta}^*$ due to

$$\nabla_{\boldsymbol{\theta}}^2 f(\boldsymbol{\theta}) \Big|_{\boldsymbol{\theta}^*} = \left(\frac{\partial \boldsymbol{\Theta}}{\partial \boldsymbol{\theta}} \right)^H \Big|_{\boldsymbol{\theta}^*} \nabla_{\boldsymbol{\Theta}}^2 h(\boldsymbol{\Theta}) \Big|_{\boldsymbol{\Theta}^*} \left(\frac{\partial \boldsymbol{\Theta}}{\partial \boldsymbol{\theta}} \right) \Big|_{\boldsymbol{\theta}^*}. \quad (5.25)$$

In the case that there exists a critical point of $h(\boldsymbol{\Theta})$, $\varphi(\boldsymbol{\theta}^+) = \boldsymbol{\Theta}^+$, such that $\boldsymbol{\theta}^+$ is not a critical point itself, i.e., $\partial f(\boldsymbol{\theta}^+)/\partial \boldsymbol{\theta} \Big|_{\boldsymbol{\theta}^+} \neq 0$, then φ is not differentiable at $\boldsymbol{\theta}^+$ as the equivalence property (5.22) does not hold.

As the nature of a critical point as $\boldsymbol{\Theta}^*$ does not change with respect to that of $\boldsymbol{\theta}^*$, we now are left to show the nature of $\boldsymbol{\Theta}^+$. First, let us consider an open ball centered at $\boldsymbol{\theta}^+$ with

radio $r > 0$, i.e., $\mathcal{B}_{\theta^+, r}$. Since θ^+ is not a critical point, there exists a direction $\Delta\theta$ where $f(\theta^+ + \Delta\theta) < f(\theta^+)$ and $\theta^+ + \Delta\theta \in \mathcal{B}_{\theta^+, r}$. Further, by hypothesis, φ is continuous, then $\varphi(\theta^+ + \Delta\theta) \in \mathcal{B}'_{\theta^+, r}$, where $\mathcal{B}'_{\theta^+, r}$ is the image $\mathcal{B}_{\theta^+, r}$ under φ . Hence, $h(\varphi(\theta^+ + \Delta\theta)) < h(\varphi(\theta^+))$, which implies the result of the Lemma.

5.7.2. PROOF OF THEOREM 5.1

To show this, we build intuition for $Q = 2$ using modules with structure

$$\mathbf{H}_q = \mathbf{I} + \Phi_1^{(q)} \mathbf{S}. \quad (5.26)$$

They can be seen as the simplest modules to implement, i.e., root expansion.

The overall linear operator implemented using this construction is

$$\begin{aligned} \mathbf{H}_2 \mathbf{H}_1 &= (\mathbf{I} + \Phi_1^{(2)} \mathbf{S})(\mathbf{I} + \Phi_1^{(1)} \mathbf{S}) \\ &= \mathbf{I} + [\Phi_1^{(2)} + \Phi_1^{(1)}] \mathbf{S} + \Phi_1^{(2)} \mathbf{S} \Phi_1^{(1)} \mathbf{S}, \end{aligned} \quad (5.27)$$

which is not a CEV-GF. Hence, no surjective map is available.

Now, let us equip with the assumption that the GF coefficient matrices commute with the GSO, i.e., $\Phi_i^{(q)} \mathbf{S} = \mathbf{S} \Phi_i^{(q)} \forall i \in [K], q \in [Q]$. Then, the expression in (5.27) can be rewritten as

$$\mathbf{H}_2 \mathbf{H}_1 = \mathbf{I} + [\Phi_1^{(2)} + \Phi_1^{(1)}] \mathbf{S} + \tilde{\Phi} \mathbf{S}^2. \quad (5.28)$$

Unfortunately, $\tilde{\Phi}$, in general, does not share the support with \mathbf{S} , hence (5.28) is, again, not a CEV-GF. As these observations carry on for the NV-GFs and CEV-GFs with different module orders, we can ensure that the φ mapping does not exist for these structures.

However, when $\Phi_i^{(q)} = \phi_i^{(q)} \mathbf{I} \forall i \in [K], q \in [Q]$, it is possible to show that the mapping φ exists as in this case $\tilde{\Phi}$ in (5.28) is another scaled identity matrix, implying that the resulting operator is a C-GF, and thus that the map is onto (surjective). Finally, the map is continuous as the roots of a polynomial (cascaded form) depend continuously on the coefficients (direct form).

5.7.3. PROOF OF THEOREM 5.2

To show this result, we first recall

$$\frac{\partial h(\Theta)}{\partial \Theta} = \frac{\partial f(\theta)}{\partial \theta} \frac{\partial \theta}{\partial \Theta}. \quad (5.29)$$

Using this relation, a saddle point in (5.3) implies that the Jacobian $\partial \theta / \partial \Theta$ is singular. So, to provide the result of the proposition, in the following, we show the conditions in which the Jacobian has a zero determinant.

Due to the particular cascaded implementation, we introduce the notation $\alpha_q = \phi_1^{(q)}$ for simplicity. Therefore, the Jacobian $\mathbf{J} := \partial \theta / \partial \Theta$ has entries

$$[\mathbf{J}]_{i,j} = \partial \phi_i / \partial \alpha_j. \quad (5.30)$$

Taking the partial derivative w.r.t α_j of \mathcal{H} , we obtain

$$\frac{\partial \mathcal{H}}{\partial \alpha_j} = \mathbf{S} \prod_{q=1, q \neq j}^K (\mathbf{I} + \alpha_q \mathbf{S}). \quad (5.31)$$

Doing the same for \mathbf{H} , we obtain

$$\frac{\mathbf{H}}{\partial \alpha_j} = \sum_{k=1}^K \frac{\partial \phi_k}{\partial \alpha_j} \mathbf{S}^k. \quad (5.32)$$

As by hypothesis, \mathcal{H} and \mathbf{H} implements the same C-GF, we can equate the terms with the same order, i.e., power of \mathbf{S} , and obtain the following relations for the entries of the Jacobian, i.e.,

$$J_{1,j} := \mathbb{J}_{1,j} = \frac{\partial \phi_1}{\partial \alpha_j} = 1, \forall j \quad (5.33)$$

$$J_{2,j} := \mathbb{J}_{2,j} = \frac{\partial \phi_2}{\partial \alpha_j} = \sum_{q_1 \neq j}^K \alpha_{q_1} \quad (5.34)$$

$$\vdots \quad (5.35)$$

$$J_{i,j} := \mathbb{J}_{i,j} = \frac{\partial \phi_i}{\partial \alpha_j} = \sum_{\substack{q_1 < \dots < q_{i-1} \\ q_1, \dots, q_{i-1} \neq j}}^K \prod_{l=1}^{i-1} \alpha_{q_l}. \quad (5.36)$$

Now, the only thing left is to find the determinant of \mathbf{J} . First, let us consider the Jacobian matrix

$$\mathbf{J} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ J_{2,1} & J_{2,2} & \cdots & J_{2,K} \\ \vdots & \vdots & \vdots & \vdots \\ J_{K,1} & J_{K,2} & \cdots & J_{K,K} \end{bmatrix} \quad (5.37)$$

As row subtraction does not change the determinant of \mathbf{J} , we now reduce \mathbf{J} by a series of row operations to obtain a closed form for its determinant, similar to the one obtain for matrices derived for time-domain IIR filters in [].

Let us subtract the first column from every other column in \mathbf{J} , i.e.,

$$\begin{bmatrix} 1 & 0 & \cdots & 0 \\ J_{2,1} & (\alpha_1 - \alpha_2) & \cdots & (\alpha_1 - \alpha_K) \\ \vdots & \vdots & \vdots & \vdots \\ J_{k,1} & (\alpha_1 - \alpha_2) J_{k-1,1}^{(1)} & \cdots & (\alpha_1 - \alpha_K) J_{k-1,K-1}^{(1)} \\ \vdots & \vdots & \vdots & \vdots \\ J_{K,1} & (\alpha_1 - \alpha_2) J_{K-1,1}^{(1)} & \cdots & (\alpha_1 - \alpha_K) J_{K-1,K-1}^{(1)} \end{bmatrix} \quad (5.38)$$

where

$$\mathbb{J}^{(1)}]_{i,j} = J_{i,j}^{(1)} := \sum_{\substack{q_1 < \dots < q_{i-1} \\ q_1, \dots, q_{i-1} \neq j+1}}^K \prod_{l=1}^{i-1} \alpha_{q_l} \quad (5.39)$$

and $\mathbf{J}^{(1)}_{1,j} = 1 \forall j$.

Using the property of the determinant for block matrices, and the equivalence $\det(\mathbf{J}) = \det(\mathbf{J}_1)$, we factorize the determinant of \mathbf{J} as

$$\det(\mathbf{J}) = \prod_{k=2}^K (\alpha_1 - \alpha_k) \det(\mathbf{J}^{(1)}). \quad (5.40)$$

Applying this process recursively, i.e., row subtraction and determinant factorization, we can keep reducing the determinant expression until we get

$$\det(\mathbf{J}) = \prod_{\substack{i,j=1 \\ i < j}}^K (\alpha_i - \alpha_j). \quad (5.41)$$

Thus, the above expression implies the result of the theorem.

5.7.4. LSMR ALGORITHM

Algorithm 3: LSMR Algorithm for solving $\mathbf{Ax} \approx \mathbf{b}$

Result: \mathbf{x}_k : least squares solution

Input: $\mathbf{A}, \mathbf{b}, \epsilon_{\text{tol}}$

initialization: $\beta_1 \mathbf{u}_1 = \mathbf{b}, \alpha_1 \mathbf{v}_1 = \mathbf{A}^T \mathbf{u}_1, \bar{\alpha}_1 = \alpha_1,$

($\beta_1 \mathbf{u}_1 = \mathbf{b}$ is shorthand for $\beta_1 = \|\mathbf{b}\|, \mathbf{u}_1 = \mathbf{b}/\beta_1$)

$\bar{\zeta}_1 = \alpha_1 \beta_1, \rho_0 = 1, \bar{\rho}_0 = 1, \bar{c}_0 = 1, \bar{s}_0 = 0,$

$\mathbf{h}_1 = \mathbf{v}_1, \bar{\mathbf{h}}_0 = \mathbf{0}, \mathbf{x}_0 = \mathbf{0};$

for $k = 1, 2, \dots$, *until convergence* **do**

 :: Bidiagonalization ::

$\beta_{k+1} \mathbf{u}_{k+1} = \mathbf{A} \mathbf{v}_k - \alpha_k \mathbf{u}_k;$

$\alpha_{k+1} \mathbf{v}_{k+1} = \mathbf{A}^T \mathbf{u}_{k+1} - \beta_{k+1} \mathbf{v}_k;$

 :: Rotation for first QR factorization ::

$\rho_k = (\bar{\alpha}_k^2 + \beta_{k+1}^2)^{\frac{1}{2}};$

$c_k = \bar{\alpha}_k / \rho_k;$

$s_k = \beta_{k+1} / \rho_k;$

$\theta_{k+1} = s_k \alpha_{k+1};$

$\bar{\alpha}_{k+1} = c_k \alpha_{k+1};$

 :: Rotation for second QR factorization ::

$\bar{\theta}_k = \bar{s}_{k-1} \rho_k;$

$\bar{\rho}_k = ((\bar{c}_{k-1} \rho_k)^2 + \theta_{k+1}^2)^{\frac{1}{2}};$

$\bar{c}_k = \bar{c}_{k-1} \rho_k / \bar{\rho}_k;$

$\bar{s}_k = \theta_{k+1} / \bar{\rho}_k;$

$\bar{\zeta}_k = \bar{c}_k \bar{\zeta}_k;$

$\bar{\zeta}_{k+1} = -\bar{s}_k \bar{\zeta}_k;$

 :: Solution Update ::

$\bar{\mathbf{h}}_k = \mathbf{h}_k - (\bar{\theta}_k \rho_k / (\rho_{k-1} \bar{\rho}_{k-1})) \bar{\mathbf{h}}_{k-1};$

$\mathbf{x}_k = \mathbf{x}_{k-1} + (\bar{\zeta}_k / (\rho_k \bar{\rho}_k)) \bar{\mathbf{h}}_k;$

$\mathbf{h}_{k+1} = \mathbf{v}_{k+1} - (\theta_{k+1} / \rho_k) \mathbf{h}_k;$

 :: Convergence Check ::

if $|\bar{\zeta}_{k+1}| < \epsilon_{\text{tol}}$ **then**

 | stop;

end

end

REFERENCES

- [1] M. Coutino and G. Leus, *On distributed consensus by a cascade of generalized graph filters*, in *2019 53rd Asilomar Conference on Signals, Systems, and Computers* (IEEE, 2019) pp. 46–50.

- [2] M. Coutino, E. Isufi, and G. Leus, *Advances in distributed graph filtering*, arXiv preprint arXiv:1808.03004 (2018).
- [3] S. K. Narang, A. Gadde, and A. Ortega, *Signal processing techniques for interpolation in graph structured data*, in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)* (IEEE, 2013) pp. 5445–5449.
- [4] P. Di Lorenzo, S. Barbarossa, P. Banelli, and S. Sardellitti, *Adaptive least mean squares estimation of graph signals*, *IEEE Trans. Sig. and Inf. Proc. over Netw.* **2**, 555 (2016).
- [5] A. Loukas, M. Zuniga, I. Protonotarios, and J. Gao, *How to identify global trends from local decisions? event region detection on mobile networks*, in *INFOCOM, 2014 Proceedings IEEE* (IEEE, 2014) pp. 1177–1185.
- [6] C. Hu, J. Sepulcre, K. A. Johnson, G. E. Fakhri, Y. M. Lu, and Q. Li, *Matched signal detection on graphs: Theory and application to brain imaging data classification*, *NeuroImage* **125**, 587 (2016).
- [7] L. F. Chamon and A. Ribeiro, *Finite-precision effects on graph filters*, in *5th IEEE Global Conf. on Sig. and Inf. Proc. (GlobalSIP)* (2917).
- [8] T. Aittomäki and G. Leus, *Graph filter design using sum-of-squares representation*, in *2019 27th European Signal Processing Conference (EUSIPCO)* (IEEE, 2019) pp. 1–5.
- [9] D. I. Shuman, P. Vandergheynst, and P. Frossard, *Distributed signal processing via chebyshev polynomial approximation*, arXiv preprint arXiv:1111.5239 (2011).
- [10] S. Segarra, A. Marques, and A. Ribeiro, *Optimal graph-filter design and applications to distributed linear network operators*, *IEEE Trans. Signal Process.* (2017).
- [11] M. Coutino, E. Isufi, and G. Leus, *Distributed edge-variant graph filters*, in *IEEE 7th Int. Workshop Comp. Adv. in Multi-Sensor Adap. Proc. (CAMSAP)* (IEEE, 2017).
- [12] Y. Saad, *Iterative methods for sparse linear systems*, Vol. 82 (siam, 2003).
- [13] S. Segarra, G. Mateos, A. G. Marques, and A. Ribeiro, *Blind identification of graph filters*, *IEEE Transactions on Signal Processing* **65**, 1146 (2016).
- [14] M. H. Hayes, *Statistical digital signal processing and modeling* (John Wiley & Sons, 2009).
- [15] A. H. Sayed, *Fundamentals of adaptive filtering* (John Wiley & Sons, 2003).
- [16] P. Di Lorenzo, P. Banelli, E. Isufi, S. Barbarossa, and G. Leus, *Adaptive graph signal processing: Algorithms and optimal sampling strategies*, *IEEE Transactions on Signal Processing* **66**, 3584 (2018).
- [17] C. Cowan, *Performance comparisons of finite linear adaptive filters*, in *IEE Proceedings F (Communications, Radar and Signal Processing)*, Vol. 134 (IET, 1987) pp. 211–216.
- [18] J. Cioffi and T. Kailath, *Fast, recursive-least-squares transversal filters for adaptive filtering*, *IEEE Transactions on Acoustics, Speech, and Signal Processing* **32**, 304 (1984).
- [19] P. Prandoni and M. Vetterli, *An fir cascade structure for adaptive linear prediction*, *IEEE transactions on signal processing* **46**, 2566 (1998).
- [20] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, *Spectral networks and locally connected networks on graphs*, arXiv preprint arXiv:1312.6203 (2013).
- [21] M. Abadi et al., *TensorFlow: Large-scale machine learning on heterogeneous systems*, (2015), software available from tensorflow.org.
- [22] F. Chollet et al., *Keras*, <https://keras.io> (2015).
- [23] J. C. Mason and D. C. Handscomb, *Chebyshev polynomials* (CRC Press, 2002).

- [24] B. D. Rao, *Adaptive iir filtering using cascade structures*, in *Proceedings of 27th Asilomar Conference on Signals, Systems and Computers* (IEEE, 1993) pp. 194–198.
- [25] J. O. Smith, *Introduction to digital filters: with audio applications*, Vol. 2 (Julius Smith, 2007).
- [26] M. Nayeri and W. K. Jenkins, *Alternate realizations to adaptive iir filters and properties of their performance surfaces*, *IEEE Transactions on Circuits and Systems* **36**, 485 (1989).
- [27] E. Isufi, A. Loukas, A. Simonetto, and G. Leus, *Autoregressive moving average graph filtering*, *IEEE Trans. Signal Process* **65**, 274 (2017).
- [28] T. Söderström and P. Stoica, *Some properties of the output error method*, *Automatica* **18**, 93 (1982).
- [29] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, *Graph neural networks: A review of methods and applications*, arXiv preprint arXiv:1812.08434 (2018).
- [30] L. Bottou, *Large-scale machine learning with stochastic gradient descent*, in *Proceedings of COMPSTAT'2010* (Springer, 2010) pp. 177–186.
- [31] Y. LeCun, Y. Bengio, and G. Hinton, *Deep learning*, *nature* **521**, 436 (2015).
- [32] J. Schmidhuber, *Deep learning in neural networks: An overview*, *Neural networks* **61**, 85 (2015).
- [33] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning* (MIT press, 2016).
- [34] F. Gama, A. G. Marques, G. Leus, and A. Ribeiro, *Convolutional neural networks architectures for signals supported on graphs*, arXiv preprint arXiv:1805.00165 (2018).
- [35] T. N. Kipf and M. Welling, *Semi-supervised classification with graph convolutional networks*, arXiv preprint arXiv:1609.02907 (2016).
- [36] J. Li and P. Stoica, *Efficient mixed-spectrum estimation with applications to target feature extraction*, *IEEE transactions on signal processing* **44**, 281 (1996).
- [37] D. C.-L. Fong and M. Saunders, *Lsmr: An iterative algorithm for sparse least-squares problems*, *SIAM Journal on Scientific Computing* **33**, 2950 (2011).
- [38] C. C. Paige and M. A. Saunders, *Lsqr: An algorithm for sparse linear equations and sparse least squares*, *ACM Transactions on Mathematical Software (TOMS)* **8**, 43 (1982).
- [39] Y. Xu and W. Yin, *A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion*, *SIAM Journal on imaging sciences* **6**, 1758 (2013).
- [40] L. Grippo and M. Sciandrone, *On the convergence of the block nonlinear gauss–seidel method under convex constraints*, *Operations research letters* **26**, 127 (2000).
- [41] G. B. Giannakis, Y. Shen, and G. V. Karanikolas, *Topology identification and learning over graphs: Accounting for nonlinearities and dynamics*, *Proceedings of the IEEE* **106**, 787 (2018).
- [42] M. Coutino, E. Isufi, T. Maehara, and G. Leus, *State-space network topology identification from partial observations*, *IEEE Transactions on Signal and Information Processing over Networks* **6**, 211 (2020).
- [43] G. Mateos, S. Segarra, A. G. Marques, and A. Ribeiro, *Connecting the dots: Identifying network structure via graph signal processing*, *IEEE Signal Processing Magazine* **36**, 16 (2019).

- [44] H. E. Egilmez, E. Pavez, and A. Ortega, *Graph learning from filtered signals: Graph system and diffusion kernel identification*, IEEE Transactions on Signal and Information Processing over Networks **5**, 360 (2018).
- [45] T. Tieleman and G. Hinton, *Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude*, COURSERA: Neural networks for machine learning **4**, 26 (2012).
- [46] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, arXiv preprint arXiv:1412.6980 (2014).
- [47] S. L. Smith, P.-J. Kindermans, C. Ying, and Q. V. Le, *Don't decay the learning rate, increase the batch size*, arXiv preprint arXiv:1711.00489 (2017).
- [48] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, *Backpropagation applied to handwritten zip code recognition*, Neural computation **1**, 541 (1989).
- [49] S. Ioffe and C. Szegedy, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, arXiv preprint arXiv:1502.03167 (2015).
- [50] N. Perraudin, J. Paratte, D. Shuman, L. Martin, V. Kalofolias, P. Vandergheynst, and D. K. Hammond, *GSPBOX: A toolbox for signal processing on graphs*, ArXiv e-prints (2014), [arXiv:1408.5781 \[cs.IT\]](https://arxiv.org/abs/1408.5781) .
- [51] T. Li and J.-F. Zhang, *Consensus conditions of multi-agent systems with time-varying topologies and stochastic communication noises*, IEEE Trans. on Automatic Control **55**, 2043 (2010).
- [52] G. Zhang and R. Heusdens, *Distributed optimization using the primal-dual method of multipliers*, IEEE Trans. on Sig. and Inf. Proc. over Netw. **4**, 173 (2018).

6

STATE-SPACE BASED NETWORK TOPOLOGY IDENTIFICATION

water your flowers // forget the current season // spring is forever

-mc19

IN the previous three chapters, we focused on graph filtering and the particular implementation aspects of GFs. Here, we move to the second research thrust of this thesis: network topology identification. That is, we now study how to unveil the underlying connections of a network when we have to our avail nodal measurements and different kinds of prior information, e.g., the physical laws driving the network process.

Specifically, in this chapter, we explore the state-space formulation of a network process to tackle the problem of the network topology identification. To do so, we introduce subspace techniques borrowed from system identification literature to the graph signal processing community and extend them to the network topology identification problem. This framework provides a unified view of network control and signal processing on graphs. Further, we provide theoretical guarantees for the recovery of the underlying topological structure from input-output observations. Also, we study several methods, linking state-of-the-art techniques in GSP with the presented formalism and discuss their specific benefits and pitfalls.

6.1. INTRODUCTION

The topology of networks is fundamental to model interactions between entities and to improve our understanding of the processes evolving over them. We find examples of

Parts of this chapter have been presented in the Information Theory and Applications Workshop (2019) and published in the Proceedings of the European Signal Processing Conference (2020) [1]

such processes in transportation networks [2], brain activity [3], epidemic dynamics and gene regulatory networks [4].

While in several scenarios, the network structure is available, it is unknown and needs to be estimated in many others. The network topology is not only used for enhancing data processing tasks but also for data *interpretability*, i.e., the network topology provides an abstraction for the underlying data dependencies. Therefore, retrieving the network structure or the dependencies of the involved members (variables) has become a research topic of large interest [5–13].

In many instances, physical systems can be defined through a state-space formulation with known dynamics. An example is the diffusion of molecules in tissue [14]. This process is used to analyze brain functions by mapping the interaction of molecules with obstacles. The area of neural dynamics also considers the problem of network design in transport theory [15]. In such applications, the topology that provides a stable desired response needs to be found, following a differential equation. Finally, we recall the problem of finding the connections between reactants in chemical reaction networks [16, 17]. Here, molecules evolve in a solution according to the interaction between reactants present in it. Hence, to understand the underlying chemical process, the relation between reactants is required. Thus, it is clear that a more general approach, parting from first-principles, to find the underlying connections is preferred. So, we focus on the problem of *retrieving the network structure* of a process modeled through a *deterministic continuous-time dynamical system* whose system matrices depend on the underlying topology.

To address this problem, in this chapter, we first introduce a formalism based on the state-space representation of network processes. Then, we devise a framework, using well-established tools from system identification [18], to estimate the network topology from network observations while allowing for recovery guarantees and mathematical rigor.

6.1.1. CHAPTER CONTRIBUTIONS

Although there is a large body of literature addressing the problem of network topology identification from network observations, most of these works do not leverage the formalism provided by space-state models nor the subspace-based system identification techniques. This chapter, therefore, presents the following contributions broadening the state-of-the-art understanding of the topology inference problem from network measurements.

- We introduce a first-order differential graph model for describing the dynamics of a deterministic continuous-time linear network process.
- Using the state-space representation of the proposed model for network processes, we put forth a first-principles based network topology identification framework leveraging subspace-based system identification techniques. We also provide conditions to retrieve the network topology from sampled observations.
- We discuss how to combine the subspace-based framework with state-of-the-art graph signal processing techniques for recovering the network topology for the full network observation case. Further, we introduce a single-shot approach to estimate

the connectivity of the network related to the state of the systems while avoiding the estimation of the state-transition matrix.

6.1.2. CHAPTER OUTLINE

This chapter is organized as follows. Section 6.2 formulates the problem of network topology inference for continuous-time dynamical systems from sampled observations. Section 6.3 introduces a first-order differential graph model and its state-space description. The system identification framework for the proposed graph-based model is introduced in Section 6.4. Section 6.5 analyzes several instances of the network topology identification problem and introduces different methods for combining the system identification framework with tools available from GSP. Section 6.6 presents a series of numerical results to show the performance of the proposed method in both synthetic and real data. Finally, Section 6.7 concludes the chapter.

6.2. THE TOPOLOGY IDENTIFICATION PROBLEM

Consider a set of N nodes $\mathcal{V} = \{v_1, \dots, v_N\}$ representing cooperative agents such as sensors, individuals, and biological structures. On top of these agents, a process \mathcal{P} describes the evolution through time of the agent signals $\mathbf{x}(t) \in \mathbb{R}^N$. Signal $\mathbf{x}(t)$ is such that the i th component $x_i(t)$ represents the signal evolution of agent v_i . The agent interactions w.r.t. the evolution of signal $\mathbf{x}(t)$ are captured by a graph $\mathcal{G}_x = (\mathcal{V}, \mathcal{E}_x)$, where \mathcal{E}_x is the edge set of this graph. We consider process \mathcal{P} is represented by a first-order differential model

$$\partial_t \mathbf{x}(t) = h(\mathcal{V}, \mathcal{E}_x, \mathcal{E}_u, \mathbf{x}(t), \mathbf{u}(t)) \quad (6.1a)$$

$$\mathbf{y}(t) = c(\mathcal{V}, \mathcal{E}_x, \mathcal{E}_u, \mathbf{x}(t), \mathbf{u}(t)) \quad (6.1b)$$

where $\partial_t \mathbf{x}(t) := d\mathbf{x}(t)/dt$ and $h(\cdot)$ and $c(\cdot)$ are maps that describe respectively the dynamics of signal $\mathbf{x}(t)$ and observables $\mathbf{y}(t)$. In model (6.1), $\mathbf{u}(t)$ is the (known) system input and \mathcal{E}_u is the edge set of another graph $\mathcal{G}_u = (\mathcal{V}, \mathcal{E}_u)$ that captures the interactions between the elements of \mathcal{V} for input $\mathbf{u}(t)$. Put simply, process (6.1) describes the evolution of signal $\mathbf{x}(t)$ under the influence of maps $h(\cdot)$ and $c(\cdot)$ and the network topologies $\mathcal{G}_x = (\mathcal{V}, \mathcal{E}_x)$ and $\mathcal{G}_u = (\mathcal{V}, \mathcal{E}_u)$. For future reference, we will represent both graphs as $\mathcal{G}_* = \{\mathcal{V}, \mathcal{E}_*\}$, where “*” is a space holder for x and u . We remark that although (6.1) leads to a directed process, i.e., there is a directed field flow, the topology (or metric of the space) does not necessarily have to be directed, e.g., heat diffusion. That is, the matrix representation of \mathcal{G}_* can indeed be a symmetric matrix (undirected graph).

We can compactly define process \mathcal{P} through the set

$$\mathcal{P} := \{h(\cdot), c(\cdot)\} \quad (6.2)$$

which contains the interactions in the system. While process \mathcal{P} describes a continuous-time process, we usually have access to *sampled realizations of it*, i.e., the observables $\mathbf{y}(t)$ are collected on a finite set of time instances $\mathcal{T} := \{t_1, t_2, \dots, t_T\}$. As a result, we might also want to consider *discrete-time* approximations of (6.2), where we either have a discrete-time realization of \mathcal{P} and/or a finite number of observables $\mathcal{Y} := \{\mathbf{y}(t)\}_{t \in \mathcal{T}}$. Processes

belonging to this family of models include, for example, linearized dynamics of biochemical reactions, where the state variables, $\mathbf{x}(t)$, represent the concentration of mRNA and proteins in genes; the inputs, $\mathbf{u}(t)$, are external stimuli applied to the system such as heat or current, and the underlying network denotes the interactions (dependencies) between the genes. In addition to this, diffusion processes in networks modeled using the heat diffusion equation, also are included in this family of models. Here, the state variables represent nodal quantities that get diffused through the network by local aggregations. For further examples where these models are applicable, we refer the reader to the applications in [19].

With this in place, we ask: *how to retrieve the network topologies \mathcal{G}_x and \mathcal{G}_u for the agent signal $\mathbf{x}(t)$ and input signal $\mathbf{u}(t)$ given the process \mathcal{P} [cf. (6.2)], the input signal $\mathbf{u}(t)$, and the observables in \mathcal{Y} ?*

We answer the above question by employing results from Hankel matrices [20] and linear algebra whose foundations lie in system identification theory [21]. We consider subspace techniques that are by definition *cost function free*. This differs from the commonly used techniques in network topology identification [22] where the learned topology heavily depends on the cost function (e.g., smoothness or sparsity). If this prior knowledge is incorrect, it leads to structures not related to the physical interactions. The adopted techniques provide theoretical insights into when and how the underlying network structures can be identified. They also have two other benefits: first, they impose no parameterization on the dynamical model and, therefore, avoid solving nonlinear optimization problems as in prediction-error methods [23]; second, they allow identifying $\mathcal{G}_* = \{\mathcal{V}, \mathcal{E}_*\}$ from sampled data, i.e., by having access only to a subset of the observables $\mathbf{y}(t)$.

Finally, we remark that although more general models can be considered, e.g., higher-order differential models, we restrict ourselves to first-order models to ease exposition and provide a thorough and stand-alone work. Nevertheless, first-order differential models are of broad interest as they include diffusion processes –see [24] and references therein.

6

6.3. FIRST ORDER DIFFERENTIAL GRAPH MODEL

The graph $\mathcal{G}_* = \{\mathcal{V}, \mathcal{E}_*\}$ is mathematically represented by a matrix \mathbf{S}_* (sometimes referred to as a *graph shift operator* [25, 26]) that has as candidates the graph adjacency matrix, the graph Laplacian, or any other matrix that captures the relations of network elements. We then consider process \mathcal{P} is described through the linear continuous-time dynamical system

$$\partial_t \mathbf{x}(t) = f_x(\mathbf{S}_x) \mathbf{x}(t) + f_u(\mathbf{S}_u) \mathbf{u}(t) + \mathbf{w}(t) \in \mathbb{R}^N \quad (6.3a)$$

$$\mathbf{y}(t) = \mathbf{C} \mathbf{x}(t) + \mathbf{D} \mathbf{u}(t) + \mathbf{v}(t) \in \mathbb{R}^L \quad (6.3b)$$

where $\mathbf{C} \in \mathbb{R}^{L \times N}$ and $\mathbf{D} \in \mathbb{R}^{L \times N}$ are the system matrices related to the observables $\mathbf{y}(t)$. The variables, $\mathbf{w}(t)$ and $\mathbf{v}(t)$ represent perturbations in the states and additive noise in the observables, respectively. The matrix function $f_* : \mathbb{R}^{N \times N} \rightarrow \mathbb{R}^{N \times N}$ is defined via the Cauchy integral [27]

$$f_*(\mathbf{S}_*) := \frac{1}{2\pi i} \int_{\Gamma_{f_*}} f_{s,*}(z) R(z, \mathbf{S}_*) dz \quad (6.4)$$

where $f_{s,*}(\cdot)$ is the scalar version of $f_*(\cdot)$ and is analytic on and over the contour Γ_{f_*} . Here, $R(z, \mathbf{S}_*)$ is the resolvent of \mathbf{S}_* given by

$$R(z, \mathbf{S}_*) := (\mathbf{S}_* - z\mathbf{I})^{-1}. \quad (6.5)$$

From the dynamical system in (6.1), it can be seen that (6.3) represents first-order (linear) differential models, where the system matrices $f_x(\mathbf{S}_x)$ and $f_u(\mathbf{S}_u)$ are *matrix functions* of the matrices representing the graphs \mathcal{G}_x and \mathcal{G}_u .

Model (6.3) captures different settings of practical interest such as diffusion on networks [24], graph filtering operations [28, 29], random walks [30], and first-order autoregressive graph processes [31]. The corresponding discrete-time state-space system related to (6.3) is

$$\mathbf{x}(k+1) = \tilde{f}_x(\mathbf{S}_x)\mathbf{x}(k) + \tilde{f}_u(\mathbf{S}_u)\mathbf{u}(k) + \mathbf{w}(k) \quad (6.6a)$$

$$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) + \mathbf{D}\mathbf{u}(k) + \mathbf{v}(k) \quad (6.6b)$$

where $\tilde{f}_*(\cdot)$ is a matrix function (to be specified later) and $\mathbf{x}(k) \in \mathbb{R}^N$, $\mathbf{u}(k) \in \mathbb{R}^N$, and $\mathbf{y}(k) \in \mathbb{R}^L$ are the discrete counterparts of $\mathbf{x}(t)$, $\mathbf{u}(t)$, and $\mathbf{y}(t)$, respectively. The variables, $\mathbf{w}(k)$ and $\mathbf{v}(k)$ represent the discrete counterparts of perturbation $\mathbf{w}(t)$ and noise $\mathbf{v}(t)$, respectively. By defining then the matrices $\mathbf{A}(\mathbf{S}_x) := \tilde{f}_x(\mathbf{S}_x)$ and $\mathbf{B}(\mathbf{S}_u) := \tilde{f}_u(\mathbf{S}_u)$, the connection between the continuous-time (6.3) and the discrete-time representation (6.6) is given by

$$\mathbf{A}(\mathbf{S}_x) := \tilde{f}_x(\mathbf{S}_x) = e^{f_x(\mathbf{S}_x)\tau} \quad (6.7)$$

$$\mathbf{B}(\mathbf{S}_u) := \tilde{f}_u(\mathbf{S}_u) = \left(\int_0^\tau e^{f_x(\mathbf{S}_x)t} dt \right) f_u(\mathbf{S}_u) \quad (6.8)$$

where τ is the sampling period and $e^{\mathbf{X}} = \sum_{k=0}^{\infty} \frac{1}{k!} \mathbf{X}^k$ is the matrix exponential function [27]. Using then (6.7) and (6.8), we can compactly write model (6.6) as a linear discrete-time state-space model

$$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k) + \mathbf{w}(k) \quad (6.9a)$$

$$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) + \mathbf{D}\mathbf{u}(k) + \mathbf{v}(k), \quad (6.9b)$$

where we dropped the dependency of the system matrices \mathbf{A} and \mathbf{B} on \mathbf{S}_x and \mathbf{S}_u to simplify the notation. Throughout the chapter, we assume that the matrices \mathbf{C} and \mathbf{D} are known, i.e., we know how the observables are related to the states and inputs.

6.4. STATE-SPACE IDENTIFICATION

The family of subspace state-space system identification methods [32] relies on geometrical properties of the dynamical model (6.9). By collecting a batch of α different observables over time, $\mathbf{y}(t) \in \mathbb{R}^L$, into the αL -dimensional vector

$$\mathbf{y}_{k,\alpha} \triangleq [\mathbf{y}(k)^\top, \dots, \mathbf{y}(k+\alpha-1)^\top]^\top,$$

we get the input-output relationship

$$\mathbf{y}_{k,\alpha} = \mathcal{O}_\alpha \mathbf{x}(k) + \mathcal{T}_\alpha \mathbf{u}_{k,\alpha} + \mathbf{n}_{k,\alpha}, \quad (6.10)$$

where

$$\mathcal{O}_\alpha \triangleq \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \\ \vdots \\ \mathbf{CA}^{\alpha-1} \end{bmatrix}, \quad (6.11)$$

is the *extended observability* matrix of the system [33] and

$$\mathcal{T}_\alpha \triangleq \begin{bmatrix} \mathbf{D} & \mathbf{0} & \mathbf{0} & \cdots & \cdots & \mathbf{0} \\ \mathbf{CB} & \mathbf{D} & \ddots & & \cdots & \mathbf{0} \\ \vdots & \ddots & \ddots & & & \vdots \\ \vdots & & & & & \mathbf{0} \\ \mathbf{CA}^{\alpha-2}\mathbf{B} & \mathbf{CA}^{\alpha-3}\mathbf{B} & \cdots & \cdots & \mathbf{CB} & \mathbf{D} \end{bmatrix}, \quad (6.12)$$

is the matrix that relates the batch input vector

$$\mathbf{u}_{k,\alpha} \triangleq [\mathbf{u}(k)^\top, \dots, \mathbf{u}(k+\alpha-1)^\top]^\top,$$

with the batch observables $\mathbf{y}_{k,\alpha}$. The vector $\mathbf{n}_{k,\alpha}$ comprises the batch noise that depends on the system perturbation $\{\mathbf{w}(k)\}$ and on the observable noise $\{\mathbf{v}(k)\}$; the detailed structure of $\mathbf{n}_{k,\alpha}$ is unnecessary for our framework. The size of the batch α is user-specified but must be larger than the number of states. Assuming the number of nodes is the number of states, this implies $\alpha > N$.

Given then the input-output relation (6.10) and the structures for the \mathcal{O}_α in (6.11) and \mathcal{T}_α in (6.12), we proceed by estimating first the state matrix \mathbf{A} using the algebraic properties of (6.11) and subsequently the input matrix \mathbf{B} from the structure of (6.12) and a least squares problem.

6

6.4.1. RETRIEVING THE STATE MATRIX \mathbf{A}

A basic requirement for estimating \mathbf{A} is system *observability* [33]. Observability allows to infer the system state from the outputs for any initial state and sequence of input vectors. Put differently, we can estimate the entire system dynamics from input-output relations. System (6.9) is observable if the system matrices $\{\mathbf{A}, \mathbf{C}\}$ satisfy $\text{rank}(\mathcal{O}_N) = N$.¹

Consider now a set of $Q \triangleq T + \alpha - 1$ input-output pairs $\{\mathbf{y}(k), \mathbf{u}(k)\}_{k=1}^Q$. By stacking the discrete batch vectors $\mathbf{y}_{k,\alpha}$ for all observations into the matrix

$$\mathbf{Y} = [\mathbf{y}_{1,\alpha}, \dots, \mathbf{y}_{T,\alpha}],$$

and using expression (6.10), we can generate the Hankel-structured data equation [34]

$$\mathbf{Y} = \mathcal{O}_\alpha \mathbf{X} + \mathcal{T}_\alpha \mathbf{U} + \mathbf{N}, \quad (6.13)$$

¹Although this assumption might not hold for all instances in practice, for a full rank and well-conditioned system matrix \mathbf{A} , this holds. This condition is also required to provide mathematical rigor to the approach and obtain theoretical guarantees.

where \mathbf{X} is the matrix that contains the evolution of the states across the columns, i.e.,

$$\mathbf{X} = [\mathbf{x}(1), \dots, \mathbf{x}(T)],$$

and where the input \mathbf{U} and noise \mathbf{N} matrices are block Hankel matrices defined as

$$\mathbf{U} = [\mathbf{u}_{1,\alpha}, \dots, \mathbf{u}_{T,\alpha}] \quad \text{and} \quad \mathbf{N} = [\mathbf{n}_{1,\alpha}, \dots, \mathbf{n}_{T,\alpha}].$$

The structure in (6.13) is at the core of system identification methods [21] and this arrangement leads naturally to a subspace-based approach to find \mathbf{A} . To detail this, consider the noise-free case

$$\mathbf{Y} = \mathcal{O}_\alpha \mathbf{X} + \mathcal{F}_\alpha \mathbf{U}. \quad (6.14)$$

To continue, \mathbf{U} should allow us to remove the part of the output, \mathbf{Y} , that was not generated from the state signal, \mathbf{X} . That is, given \mathbf{U} , right-multiplying \mathbf{Y} with the projection matrix $\mathbf{\Pi}_{\mathbf{U}^T}^\perp$ should give us

$$\mathbf{Y}\mathbf{\Pi}_{\mathbf{U}^T}^\perp = \mathcal{O}_\alpha \mathbf{X}\mathbf{\Pi}_{\mathbf{U}^T}^\perp, \quad (6.15)$$

while ensuring the full row-rankness of $\mathbf{X}\mathbf{\Pi}_{\mathbf{U}^T}^\perp$. Given that \mathbf{U} is full row-rank, such a projection matrix, $\mathbf{\Pi}_{\mathbf{U}^T}^\perp$, can be constructed as

$$\mathbf{\Pi}_{\mathbf{U}^T}^\perp \triangleq \mathbf{I} - \mathbf{U}^T (\mathbf{U}\mathbf{U}^T)^{-1} \mathbf{U}. \quad (6.16)$$

Since $\mathbf{U}\mathbf{\Pi}_{\mathbf{U}^T}^\perp = \mathbf{0}$, this leads to (6.15). The condition of the inputs guaranteeing the invertibility of $\mathbf{U}\mathbf{U}^T$ and the full row-rankness of $\mathbf{X}\mathbf{\Pi}_{\mathbf{U}^T}^\perp$ is known as the *persistent excitation* condition [34]. This property requires the inputs to excite all the modes of the system. Despite that in many scenarios this property might be difficult to enforce, e.g., there is no total control on the inputs of the system, we consider the case where full control of the input signal is guaranteed; hence, as discussed in [34], we assume here that it is always possible to meet such condition with high probability.

Being able to project out the contribution of the input from the output, while ensuring the rank of the projected state signal matrix, leads to the following property:

$$\text{span}(\mathbf{Y}\mathbf{\Pi}_{\mathbf{U}^T}^\perp) = \text{span}(\mathcal{O}_\alpha). \quad (6.17)$$

That is, the signal subspace of the projected observables $\mathbf{Y}\mathbf{\Pi}_{\mathbf{U}^T}^\perp$ coincides with that of the extended observability matrix. Therefore, $\text{span}(\mathcal{O}_\alpha)$ can be estimated from $\mathbf{Y}\mathbf{\Pi}_{\mathbf{U}^T}^\perp$ and, subsequently, the system matrix \mathbf{A} by using the block structure of \mathcal{O}_α in (6.11). To detail this procedure, consider the *economy-size* singular value decomposition (SVD) of $\mathbf{Y}\mathbf{\Pi}_{\mathbf{U}^T}^\perp$

$$\mathbf{Y}\mathbf{\Pi}_{\mathbf{U}^T}^\perp = (\mathbf{Y}\mathbf{\Pi}_{\mathbf{U}^T}^\perp)_N = \mathbf{W}_{\alpha,N} \mathbf{\Sigma}_N \mathbf{V}_{\alpha,N}^T, \quad (6.18)$$

where $(\mathbf{Y}\mathbf{\Pi}_{\mathbf{U}^T}^\perp)_N$ is the N -rank approximation of $\mathbf{Y}\mathbf{\Pi}_{\mathbf{U}^T}^\perp$ and equality holds because of the full row-rank assumption of the projected input $\mathbf{X}\mathbf{\Pi}_{\mathbf{U}^T}^\perp$. Then, from condition (6.17), we have

$$\mathcal{O}_\alpha = \mathbf{W}_{\alpha,N} \mathbf{T}, \quad (6.19)$$

for some invertible similarity transform matrix $\mathbf{T} \in \mathbb{R}^{N \times N}$. Given \mathbf{C} known with full column-rank, we can estimate the transform matrix \mathbf{T} from the structure of \mathcal{O}_α [cf. (6.11)] as

$$\hat{\mathbf{T}} = (\mathbf{J}\mathbf{W}_{\alpha,N})^\dagger \mathbf{C}, \quad (6.20)$$

where $(\cdot)^\dagger$ denotes the Moore-Penrose pseudoinverse and $\mathbf{J}\mathbf{W}_{\alpha,N}$ denotes the first L rows of $\mathbf{W}_{\alpha,N}$. If \mathbf{C} is not full column-rank, the transform matrix \mathbf{T} is not unique. We will deal with the non-uniqueness of \mathbf{T} in Chapter 7.

Finally, to get \mathbf{A} , we exploit the *shift invariant* structure of \mathcal{O}_α w.r.t. \mathbf{A} , i.e.,

$$\mathbf{J}_u \mathcal{O}_\alpha \mathbf{A} = \mathbf{J}_l \mathcal{O}_\alpha \in \mathbb{R}^{(\alpha-1)L \times N}, \quad (6.21)$$

where $\mathbf{J}_u \mathcal{O}_\alpha$ and $\mathbf{J}_l \mathcal{O}_\alpha$ denote the upper and lower $(\alpha-1)L$ blocks of \mathcal{O}_α [cf. (6.11)], respectively. By substituting then the expression (6.19) for \mathcal{O}_α into (6.21) and by using the estimate $\hat{\mathbf{T}}$ (6.20) for the transform matrix \mathbf{T} , the least squares estimate for \mathbf{A} is given by

$$\hat{\mathbf{A}} = (\mathbf{J}_u \mathbf{W}_{\alpha,N} \hat{\mathbf{T}})^\dagger \mathbf{J}_l \mathbf{W}_{\alpha,N} \hat{\mathbf{T}}. \quad (6.22)$$

6.4.2. RETRIEVING THE INPUT MATRIX \mathbf{B}

While the input matrix \mathbf{B} can be obtained with a similar approach as \mathbf{A} [35] (yet with a more involved shift-invariant structure), we compute it together with the initial state $\mathbf{x}(0)$ by solving a least squares problem.

To do so, we expand system (6.9) to all its terms as

$$\begin{aligned} \mathbf{y}(k) - (\mathbf{u}(k)^\top \otimes \mathbf{I}_L) \text{vec}(\mathbf{D}) \\ = \mathbf{C}\mathbf{A}^k \mathbf{x}(0) + \left(\sum_{q=0}^{k-1} \mathbf{u}(q)^\top \otimes \mathbf{C}\mathbf{A}^{k-q-1} \right) \text{vec}(\mathbf{B}), \end{aligned} \quad (6.23)$$

where \mathbf{I}_L is the $L \times L$ identity matrix and $\text{vec}(\cdot)$ is the vectorization operator. We then collect the unknowns $\mathbf{x}(0)$ and $\text{vec}(\mathbf{B})$ into vector $\boldsymbol{\theta} = [\mathbf{x}(0)^\top \text{vec}(\mathbf{B})^\top]^\top$ and define the matrix

$$\hat{\Psi} \triangleq [\mathbf{C}\hat{\mathbf{A}}^k, \sum_{q=0}^{k-1} \mathbf{u}(q)^\top \otimes \mathbf{C}\hat{\mathbf{A}}^{k-q-1}],$$

where we substituted the state transition matrix \mathbf{A} with its estimate (6.22) while the other quantities \mathbf{C} , \mathbf{D} and $\mathbf{u}(k)$ are known. Finally, we get the input matrix \mathbf{B} by solving

$$\min_{\boldsymbol{\theta}} \frac{1}{Q} \sum_{k=1}^Q \|\mathbf{y}(k) - \hat{\Psi}\boldsymbol{\theta}\|_2^2. \quad (6.24)$$

Given the system matrices $\{\hat{\mathbf{A}}, \hat{\mathbf{B}}, \mathbf{C}, \mathbf{D}\}$, the state interaction graph $\mathcal{G}_x(\mathbf{S}_x)$ and input interaction graph $\mathcal{G}_u(\mathbf{S}_u)$ can be obtained by enforcing the constraints derived from the information of the physical process, i.e., the model dynamics. Hence, the network structure depends heavily on the estimate of the subspace span of \mathcal{O}_α .

6.4.3. NOISY SETTING

We discuss now a method for estimating \mathbf{A} and \mathbf{B} with perturbations in the state evolution $\mathbf{x}(t)$ and noise in the observables $\mathbf{y}(t)$. To tackle these challenges, we leverage *instrumental variables*.

Consider the following partition of the observables and input matrix

$$\mathbf{Y} = [\mathbf{Y}_1^\top, \mathbf{Y}_2^\top]^\top \quad \text{and} \quad \mathbf{U} = [\mathbf{U}_1^\top, \mathbf{U}_2^\top]^\top,$$

where \mathbf{Y}_1 (resp. \mathbf{U}_1) and \mathbf{Y}_2 (resp. \mathbf{U}_2) have respectively β and γ blocks of size L with $\gamma = \alpha - \beta$. The model for the observables \mathbf{Y}_2 is [cf.(6.13)]

$$\mathbf{Y}_2 = \mathcal{O}_\gamma \mathbf{X}_2 + \mathcal{T}_\gamma \mathbf{U}_2 + \mathbf{N}_2. \quad (6.25)$$

Following then the projection-based strategy to remove the dependency from \mathbf{U}_2 , we can write the projected noisy observables as [cf. (6.15)]

$$\mathbf{Y}_2 \mathbf{\Pi}_{\mathbf{U}_2^\top}^\perp = \mathcal{O}_\gamma \mathbf{X}_2 \mathbf{\Pi}_{\mathbf{U}_2^\top}^\perp + \mathbf{N}_2 \mathbf{\Pi}_{\mathbf{U}_2^\top}^\perp, \quad (6.26)$$

with \mathbf{X}_2 and \mathbf{N}_2 being the respective partitions of the state evolution and perturbation matrix.

Observe from (6.26) that the signal subspace is corrupted with the noise projection term $\mathbf{N}_2 \mathbf{\Pi}_{\mathbf{U}_2^\top}^\perp$. To remove the latter, we follow the instrumental variable approach. Since the noise is uncorrelated with the input \mathbf{U}_1 and since the noise present in the second block \mathbf{N}_2 is uncorrelated with the noise in the first block \mathbf{N}_1 , it holds that

$$\lim_{T \rightarrow \infty} \frac{1}{T} \mathbf{N}_2 [\mathbf{U}_1^\top, \mathbf{Y}_1^\top] = [\mathbf{0}, \mathbf{0}]. \quad (6.27)$$

Subsequently, we define $\mathbf{Z}_1 \triangleq [\mathbf{U}_1^\top, \mathbf{Y}_1^\top]$ and consider the matrix

$$\mathbf{G}_1 \triangleq \frac{1}{N} \mathbf{Y}_2 \mathbf{\Pi}_{\mathbf{U}_2^\top}^\perp \mathbf{Z}_1, \quad (6.28)$$

to estimate of the signal subspace. Matrix \mathbf{G}_1 is asymptotically “noise free” due to (6.27). From the economy-size SVD (N -rank approximation) of \mathbf{G}_1 , we get

$$\mathbf{G}_1 \approx \mathbf{W}_{\gamma, N} \mathbf{\Sigma}_N \mathbf{V}_{\gamma, N}^\top. \quad (6.29)$$

Finally, by using $\mathbf{W}_{\gamma, N}$, the system matrices \mathbf{A} and \mathbf{B} can be estimated from expressions (6.20), (6.22), and (6.24).

Note that the estimator for the signal subspace [cf. (6.29)] has intrinsic statistical properties. To control the variance and bias, different works have proposed to left and right weigh the matrices in (6.28) before the SVD [36]. As establishing optimal weighting matrices requires further analysis, we do not detail them here and refer interested readers to [37].

6.4.4. CONTINUOUS-TIME MODEL IDENTIFICATION

Given the discrete-time system matrices $\{\hat{\mathbf{A}}, \hat{\mathbf{B}}, \mathbf{C}, \mathbf{D}\}$, we can estimate the continuous-time transition matrices of (6.3). Observe from (6.7) that estimating $f_x(\mathbf{S}_x)$ from $\hat{\mathbf{A}}$ requires computing only the matrix logarithm of $\hat{\mathbf{A}}$. Nevertheless, as stated by the following proposition, there are conditions process \mathcal{P} should meet for this matrix logarithm to (i) exist and (ii) be unique. For clarity, the involved matrices are *real* since we have matrix functions that map real matrices onto real matrices.

Proposition 6.1. *Let the analytic function $f_{s,x}(\cdot)$ of $f_x(\mathbf{S}_x)$ in (6.7) satisfy*

$$(a) \quad e^{f_{s,x}(z)} \notin \mathbb{R}^-, \quad \forall z \in \text{eig}(\mathbf{S}_x)$$

$$(b) \quad f_{s,x}(z) > -\infty, \quad \forall z \in \text{eig}(\mathbf{S}_x)$$

where \mathbb{R}^- is the closed negative real axis. Then, process \mathcal{P} guarantees that (i) and (ii) are met.

If the conditions of Proposition 6.1 are met for $f_{s,x}(\cdot)$, then \mathbf{A} has no eigenvalues on \mathbb{R}^- . This implies that the principal logarithm² of \mathbf{A} , $\ln(\mathbf{A}) = \tau f_x(\mathbf{S}_x)$, exists and is unique. Note that if \mathbf{A} is real, its principal logarithm is also real. Hence, if $f_{s,x}(\cdot)$ satisfies the conditions of Proposition 6.1, the continuous-time transition matrices for nonsingular $\hat{\mathbf{A}} - \mathbf{I}$ are

$$\hat{f}_x(\mathbf{S}_x) = \frac{1}{\tau} \ln(\hat{\mathbf{A}}) \quad (6.30a)$$

$$\hat{f}_u(\mathbf{S}_u) = (\hat{\mathbf{A}} - \mathbf{I})^{-1} \hat{f}_x(\mathbf{S}_x) \hat{\mathbf{B}}. \quad (6.30b)$$

The expression for $\hat{f}_u(\mathbf{S}_u)$ is derived from

$$\int_0^{\tau} e^{f_x(\mathbf{S}_x)t} dt = f_x(\mathbf{S}_x)^{-1} (e^{f_x(\mathbf{S}_x)\tau} - \mathbf{I}). \quad (6.31)$$

Given $\hat{f}_x(\mathbf{S}_x)$ and $\hat{f}_u(\mathbf{S}_u)$, we are left to estimate the underlying topologies. Depending on the available prior information, the network topology can be estimated with methods promoting particular properties. Therefore, in the next section, we discuss how to retrieve the network topologies using the recovered system matrices.

Remark 6.1. *Although we focus principally on continuous-time models, all results hold also for purely discrete models with appropriate minor changes in the functional dependencies of the system matrices.*

6.5. IDENTIFICATION OF NETWORK CONNECTIVITY

At this point, the system matrices have been obtained. Now, we consider different scenarios for estimating the topology of the underlying networks.

²For two matrices \mathbf{X} and \mathbf{Y} , \mathbf{X} is said to be the matrix logarithm of \mathbf{Y} if $e^{\mathbf{X}} = \mathbf{Y}$. If a matrix is invertible and has no-negative real eigenvalues, there is a unique logarithm which is called principal logarithm [27].

6.5.1. KNOWN SCALAR MAPPINGS

In this case, we first obtain the eigenvalues of the graph matrices by applying the inverse mappings $f_{s,x}^{-1}$ and $f_{s,u}^{-1}$ to the spectra of the respective matrices. Therefore, for guaranteeing a unique set of eigenvalues for the graph matrices, the functions $\{f_{s,x}, f_{s,u}\}$ should be bijective, i.e., a one-to-one mapping, on an appropriate domain. For instance, if \mathbf{S}_* is the normalized Laplacian, the mappings should be bijective in the interval $[0, 2]$ as the spectrum of the normalized Laplacian lies there.

When the inverse mappings cannot be found analytically (e.g., due to computational reasons), the problem of finding the eigenvalues of the graph matrices boils down to a series of *root finding* problems. That is, consider $[\omega_*]_k$ as the k th eigenvalue of the matrix \mathbf{M}_* , where $\mathbf{M}_x := \hat{\mathbf{A}}$ and $\mathbf{M}_u := \hat{\mathbf{B}}$, and $f_{s,*}$ as the respective (known) scalar mapping. Then, the estimation of the eigenvalue vector λ_* for each of the matrices can be formulated as

$$[\hat{\lambda}_*]_k = \arg \min_{[\lambda_*]_k \in \mathbb{R}^+} \|f_{s,*}([\lambda_*]_k) - [\omega_*]_k\|_2^2 \quad (6.32)$$

for $*$ being a space holder for x and u . Fortunately, there exist efficient algorithms to obtain roots with a high accuracy even for non-linear functions [38]. As by definition $\hat{\mathbf{A}}$ and $\hat{\mathbf{B}}$ are matrix functions of \mathbf{S}_x and \mathbf{S}_u [cf. (6.4)], respectively, we can use the eigenbasis from these matrices to reconstruct the graph matrices as

$$\hat{\mathbf{S}}_* = \hat{\mathbf{U}}_* \text{diag}(\hat{\lambda}_*) \hat{\mathbf{U}}_*^{-1}, \text{ for } * \in \{x, u\} \quad (6.33)$$

with $\hat{\mathbf{U}}_x = \text{eigvecs}(\hat{\mathbf{A}})$ and $\hat{\mathbf{U}}_u = \text{eigvecs}(\hat{\mathbf{B}})$.

6.5.2. UNKNOWN SCALAR MAPPINGS.

When the scalar functions $\{f_{s,x}, f_{s,u}\}$ are unknown, we can opt to retrieve the *sparsest* graphs that are able to generate the estimated matrices, i.e.,

$$\begin{aligned} \hat{\mathbf{S}}_* = \underset{\omega_* \in \mathbb{R}^n}{\text{argmin}} \quad & \|\mathbf{S}_*\|_0 \\ \text{subject to} \quad & \mathbf{S}_* = \hat{\mathbf{U}}_* \text{diag}(\omega_*) \hat{\mathbf{U}}_*^{-1}, \mathbf{S} \in \mathcal{S} \end{aligned} \quad (6.34)$$

where $\text{diag}(\cdot)$ denotes a diagonal matrix with its argument on the main diagonal and \mathcal{S} is the set of desired graph matrices, e.g., adjacency matrices, combinatorial Laplacian matrices, etc. To do so, we can employ methods existing in the GSP literature that, given the graph matrix eigenbasis, retrieve a sparse matrix representation of the graph [22]. As these methods are treated in detail in their original references, we refer the reader to, e.g., [7, 39] and references therein for an in-detail explanation.

6.5.3. SINGLE-SHOT STATE GRAPH ESTIMATION.

As alternative to the previous two approaches, we can estimate the network topology related to the states by avoiding the computation of \mathbf{A} explicitly. That is, after obtaining an estimate of \mathbf{C}_T , and hence \mathbf{T} , we can notice that system (6.21) can be modified to include the graph matrix, i.e.,

$$\mathbf{J}_u \mathcal{O}_\alpha \mathbf{T}^{-1} \mathbf{S}_x \mathbf{A} = \mathbf{J}_l \mathcal{O}_\alpha \mathbf{T}^{-1} \mathbf{S}_x. \quad (6.35)$$

Notice that in (6.35), we not only exploit the shift invariance in the \mathcal{O}_α matrix but also the fact that \mathbf{S}_x and \mathbf{A} commute. We can check this relation holds by recalling that

$$\mathbf{J}_u \mathcal{O}_\alpha \mathbf{T}^{-1} \mathbf{S}_x \mathbf{A} = \begin{bmatrix} \mathbf{C} \mathbf{S}_x \mathbf{A} \\ \mathbf{C} \mathbf{A} \mathbf{S}_x \mathbf{A} \\ \vdots \\ \mathbf{C} \mathbf{A}^{\alpha-2} \mathbf{S}_x \mathbf{A} \end{bmatrix} = \begin{bmatrix} \mathbf{C} \mathbf{A} \mathbf{S}_x \\ \mathbf{C} \mathbf{A}^2 \mathbf{S}_x \\ \vdots \\ \mathbf{C} \mathbf{A}^{\alpha-1} \mathbf{S}_x \end{bmatrix} = \mathbf{J}_1 \mathcal{O}_\alpha \mathbf{T}^{-1} \mathbf{S}_x. \quad (6.36)$$

As a result, we can pose the following optimization problem

$$\min_{\mathbf{S}_x \in \mathcal{S}, \mathbf{M} \in \mathcal{M}} \|\mathbf{J}_u \mathcal{O}_\alpha \mathbf{T}^{-1} \mathbf{M} - \mathbf{J}_1 \mathcal{O}_\alpha \mathbf{T}^{-1} \mathbf{S}_x\|_{\mathbb{F}}^2 + \mu \|\mathbf{S}_x\|_1, \quad (6.37)$$

where we define $\mathbf{M} := \mathbf{S}_x \mathbf{A}$ to convexify the problem. Here, μ is a regularization parameter controlling the sparsity of \mathbf{S}_x and the optimization is carried out over the set of desired graph matrices, \mathcal{S} (as in (6.34)) and \mathcal{M} is a convex set of matrices meeting conditions derived by the matrix representation of the graph, e.g., if \mathbf{S}_x is restricted to a combinatorial Laplacian then $\mathbf{1}^\top \mathbf{M} = \mathbf{0}^\top$ must hold. Alternatively, we could solve for \mathbf{S}_x and \mathbf{A} by means of alternative minimization [40].

6.6. NUMERICAL EXAMPLES

To illustrate the performance of the proposed framework, in the following, we discuss a set of experiments employing both synthetic and real data.

6.6.1. TOY SYNTHETIC EXAMPLE

For this example, we consider a simple system where $\mathbf{S}_x \neq \mathbf{S}_u$ with $N = 15$ nodes. Here, $f_{s,x}$ is a scaled diffusion map (i.e., $f_{s,x}(z) = ae^{-z\tau}$, also known as heat kernel), and $f_{s,u}$ is the identity map, i.e., $f_{s,u}(z) = z$. It is assumed that all states are measured, i.e., $\mathbf{C} = \mathbf{I}$, and that there is a direct feedback from the input to the observations, i.e., $\mathbf{D} = \mathbf{I}$. As input, we considered a random piece-wise constant (during the sampling period) binary bipolar signal with 300 samples each. The reconstruction of the topology using the proposed framework is shown in Fig. 6.1. In Fig. 6.1a, the true and reconstructed adjacency matrices for the states and input are shown. As expected, when the data follows a practical model, the reconstruction of the matrices \mathbf{S}_x and \mathbf{S}_u is guaranteed to be exact. Here, since we have considered simple scalar mappings, we only perform root finding to retrieve the eigenvalues of the graph matrices. The eigenvalues comparison for both graphs is shown in Fig. 6.1b.

6.6.2. DISCRETE MODEL VALIDATION

In this experiment we corroborate the discrete model (6.6) in finding a graph from continuous-time data generated following the model (6.3). The underlying graphs are two fixed random regular graphs of $N = 50$ nodes with node degree $d = 3$. The data are generated by a continuous-time solver with system evolution matrix $f_x(\mathbf{S}_x) = -\mathbf{S}_x$ and input matrix $f_u(\mathbf{S}_u) = -(\mathbf{S}_u + \mathbf{I})$. The observable matrix \mathbf{C} is set to identity and \mathbf{D} is the zero matrix. The input signal is drawn from a standard normal distribution and we set the number of samples to N^3 with a sampling time of $\tau = 10^{-3}$.

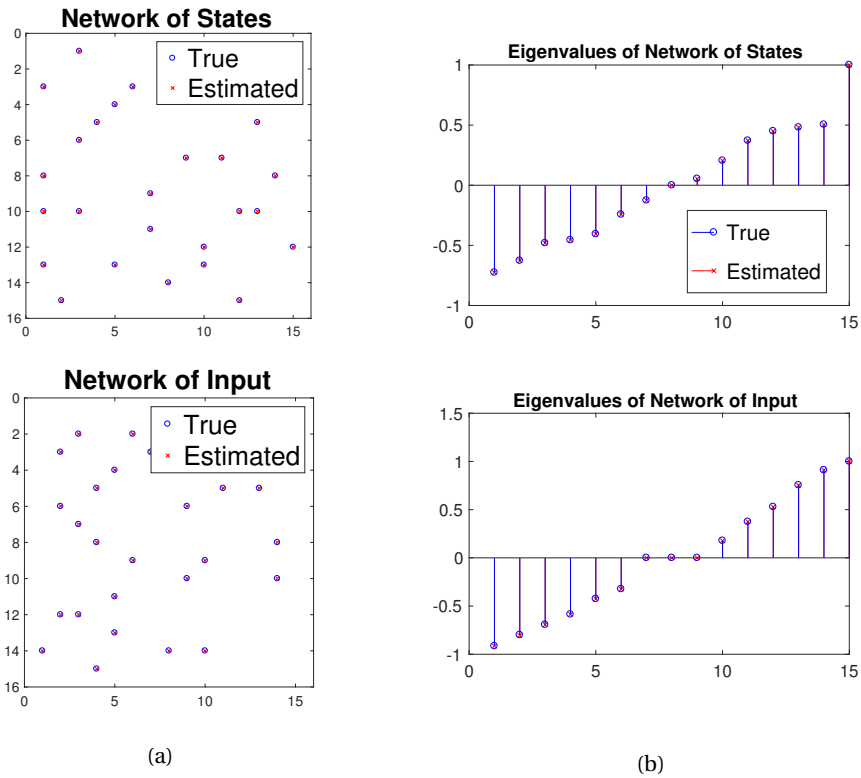


Figure 6.1: Comparison of the true graph (blue circles) and estimated graph (red crosses) for both the states and input using the method described in Section 6.5.1. (a) Adjacency matrices for the states graph (top) and the input signal graph (bottom). (b) Eigenvalues for both states and input graphs.

We compare the model-based approach (cf. Section 6.5.1) with the sparsity-promoting method from Section 6.5.2. For the latter, the system matrices \mathbf{A} and \mathbf{B} are first obtained from the continuously sampled data (cf. Section 6.3). Then, the eigenvectors of these matrices are used as spectral templates as in [7]. These results are shown in Figure 6.2. Figure 6.2a shows the estimated network topologies for a particular input signal realization, while Figure 6.2b compares the respective spectra. The spectral templates approach overestimates the number of edges and underestimates the graph eigenvalues. However, the graph obtained with spectral templates is a *matrix function* of the original graph, i.e., there is a function (polynomial) that maps the estimated graph to the original one. This is because \mathbf{S}_x has all eigenvalues with multiplicity one. The proposed technique relying on the discrete model (6.6) retrieves the eigenvalues and the graph structure perfectly. This result is not surprising since subspace-based system identification is a consistent estimator for the transition matrix and the model-aware method uses the knowledge of $f_{s,*}(\cdot)$ while the spectral templates approach does not. For this scenario, we also considered building the graph from the data covariance matrix, but this technique did not lead to satisfactory results. We attribute this misbehavior to the fact that the covariance matrix is not diagonalizable by the graph modes (i.e., eigenvectors of \mathbf{S}_x) due to the presence of an input signal.

6.6.3. NETWORK TOPOLOGY IDENTIFICATION: ETEX DATASET

We now consider data from the European Tracer Experiment (ETEX) [41]. In this experiment a tracer was released into the atmosphere and its evolution was sampled and stored from multiple stations in time. As it is unlikely that such a process has as many states as stations, we cluster the 168 measuring stations in 25 geographical regions and aggregate its measurements as a preprocessing step. This preprocessing is sustained by looking at the singular values of $\frac{1}{T}\mathbf{Y}$ in Fig. 6.3a. In this figure, it is observed that most of the dynamics can be described with a system of order 5, i.e., first knee in the plot. We selected 25 nodes as a trade off between complexity and graph interpretability (second knee). As the propagation of the tracer is considered a pure diffusion in an *autonomous* system, i.e., matrices \mathbf{B} and \mathbf{D} equal zero, we employ the proposed single-shot state graph estimation method [cf. (6.37)] to retrieve the underlying network structure. In this case, the observations are the system states, i.e., $\mathbf{C} = \mathbf{I}$. The estimated graph is shown in Fig. 6.3b. The size of the circle representing a vertex is proportional to the degree of the node. We can observe that the region of Berlin presents the highest degree which is consistent with the concentration results in [42]. The strong connectivity along the France–Germany region correlates with the spreading pattern of the agent. Despite that this graph has fewer nodes than the one obtained in [42] (see Fig. 6.3c), the estimated graph presents a better visual interpretability.

6.7. CHAPTER SUMMARY

In this chapter, we saw how to model a family of network processes by discretized first-order differential equations. We showed that under this framework, it is possible to relate the system matrices with the matrix representation of the underlying network. This formalism allowed us to bridge the state-space representation of dynamical network systems

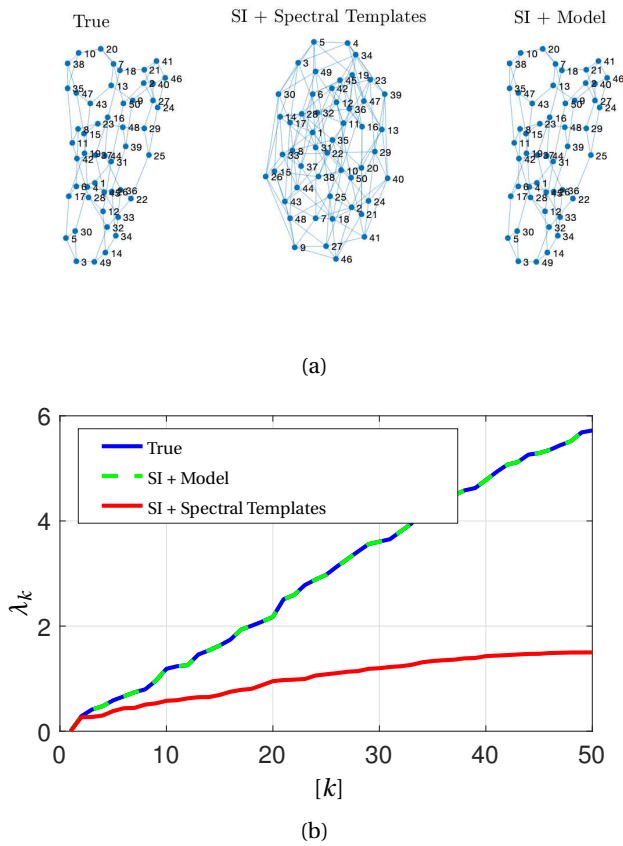


Figure 6.2: Comparison of the spectral template method using the system identification framework and the model-aware method using knowledge of the scalar function $f_{s,*}$. (a) Reconstructed state graph. (b) Comparison of eigenvalues of the estimated graphs.

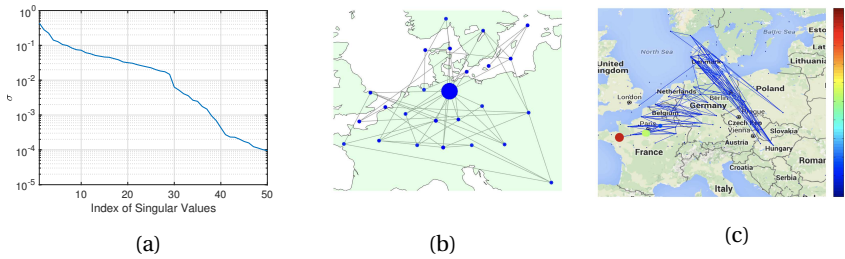


Figure 6.3: (a) First 50 singular values (σ) of $\frac{1}{T}\mathbf{Y}$ for $T = 3 \times 10^4$ and $\alpha = 169$. Here, the original dataset has been interpolated and filtered to increase the length of the recordings. The first knee, i.e., the drop in singular values (SV), occurs at the 5th SV, the second drop happens around the 20th SV. (b) Learned graph from dataset using the proposed single-shot state graph estimator. (c) Learned graph in [42] using the ETEX dataset.

with the problem of graph topology identification.

We showed that due to the established connection between state-space models and graph topology identification, it is possible to employ classical subspace-based techniques to unveil the network connectivity from measurements at the nodes of the network. Further, our introduced framework allowed us to consider cases where exogenous inputs, i.e., external excitations, evolve, or depend on a different network structure.

We also discussed general techniques, some of them leveraging state-of-the-art results from GSP, that can be employed together with subspace-based techniques to identify the graph topology. And we presented a single-shot method for topology identification that does not require the explicit computation of the system matrix related to the states.

REFERENCES

6

- [1] M. Coutino, E. Izufi, T. Maehara, and G. Leus, *State-space based network topology identification*, in *Sig. Proc. Conf. (EUSIPCO), 2020 28rd European* (IEEE, 2020).
- [2] J. A. Deri and J. M. Moura, *New york city taxi analysis with graph signal processing*, in *Proc. of the IEEE Conf. on Sig. and Inf. Process. (GLOBALSIP)* (2016) pp. 1275–1279.
- [3] O. Sporns, *Networks of the Brain* (MIT press, 2010).
- [4] F. Mittler, et al., *Reactive oxygen gene network of plants*, *Trends in Plant Science* **9**, 490 (2004).
- [5] V. Kalofolias, *How to learn a graph from smooth signals*, in *Artificial Intelligence and Statistics* (2016) pp. 920–929.
- [6] G. Mateos, S. Segarra, and A. G. Marques, *Inference of graph topology*, *Cooperative and Graph Signal Processing: Principles and Applications* (PM Djuric and C. Richard, eds.), Amsterdam, Netherlands: Elsevier (2018).
- [7] S. Segarra, A. G. Marques, G. Mateos, and A. Ribeiro, *Network topology inference from spectral templates*, *IEEE Transactions on Signal and Information Processing over Networks* (2017).
- [8] G. Karanikolas, G. B. Giannakis, K. Slavakis, and R. M. Leahy, *Multi-kernel based non-*

- linear models for connectivity identification of brain networks*, in *Proc. of the IEEE Int. Conf. on Acoust. Speech and Sig. Process. (ICASSP)* (2016) pp. 6315–6319.
- [9] Y. Shen, B. Baingana, and G. B. Giannakis, *Kernel-based structural equation models for topology identification of directed networks*, *IEEE Trans. on Sig. Process.* **65**, 2503 (2017).
- [10] K.-S. Lu and A. Ortega, *Closed form solutions of combinatorial graph laplacian estimation under acyclic topology constraints*, arXiv preprint arXiv:1711.00213 (2017).
- [11] J. Zhou and J.-a. Lu, *Topology identification of weighted complex dynamical networks*, *Physica A: Statistical Mechanics and Its Applications* **386**, 481 (2007).
- [12] Y. Shen, B. Baingana, and G. B. Giannakis, *Tensor decompositions for identifying directed graph topologies and tracking dynamic networks*, *IEEE Trans. on Sig. Process.* **65**, 3675 (2017).
- [13] R. Shafipour, S. Segarra, A. G. Marques, and G. Mateos, *Network topology inference from non-stationary graph signals*, in *Proc. of the IEEE Int. Conf. on Acoust. Speech and Sig. Process. (ICASSP)* (2017) pp. 5870–5874.
- [14] Y. Iturria-Medina, R. C. Sotero, E. J. Canales-Rodríguez, Y. Alemán-Gómez, and L. Melie-García, *Studying the human brain anatomical network via diffusion-weighted mri and graph theory*, *Neuroimage* **40**, 1064 (2008).
- [15] J. A. Hertz, *Introduction to the theory of neural computation* (CRC Press, 2018).
- [16] D. Angeli, *A tutorial on chemical reaction network dynamics*, *European journal of control* **15**, 398 (2009).
- [17] W. Pan, Y. Yuan, J. Gonçalves, and G.-B. Stan, *Reconstruction of arbitrary biochemical reaction networks: A compressive sensing approach*, in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)* (IEEE, 2012) pp. 2334–2339.
- [18] L. Ljung, *Perspectives on system identification*, *Annual Reviews in Control* **34**, 1 (2010).
- [19] Y. De Castro, T. Espinasse, and P. Rochet, *Reconstructing undirected graphs from eigenspaces*, *The Journal of Machine Learning Research* **18**, 1679 (2017).
- [20] F. Takens, *Detecting strange attractors in turbulence*, in *Dynamical systems and turbulence, Warwick 1980* (Springer, 1981) pp. 366–381.
- [21] M. Viberg, *Subspace-based methods for the identification of linear time-invariant systems*, *Automatica* **31**, 1835 (1995).
- [22] G. Mateos, S. Segarra, A. G. Marques, and A. Ribeiro, *Connecting the dots: Identifying network structure via graph signal processing*, *IEEE Signal Processing Magazine* **36**, 16 (2019).
- [23] L. Ljung, *Prediction error estimation methods*, *Circuits, Systems and Signal Processing* **21**, 11 (2002).
- [24] S.-Y. Chung, Y.-S. Chung, and J.-H. Kim, *Diffusion and elastic equations on networks*, *Publications of the Research Institute for Mathematical Sciences* **43**, 699 (2007).
- [25] A. Sandryhaila and J. M. Moura, *Discrete signal processing on graphs*, *IEEE Trans. Sig. Process* **61**, 1644 (2013).
- [26] A. Ortega, P. Frossard, J. Kovačević, J. M. Moura, and P. Vandergheynst, *Graph signal processing: Overview, challenges, and applications*, *Proceedings of the IEEE* **106**, 808 (2018).
- [27] N. J. Higham, *Functions of matrices: theory and computation*, Vol. 104 (Siam, 2008).

- [28] S. Segarra, A. Marques, and A. Ribeiro, *Optimal graph-filter design and applications to distributed linear network operators*, *IEEE Trans. Signal Process.* (2017).
- [29] M. Coutino, E. Isufi, and G. Leus, *Advances in distributed graph filtering*, *IEEE Transactions on Signal Processing* (2019).
- [30] L. Lovász *et al.*, *Random walks on graphs: A survey*, *Combinatorics, Paul erdos is eighty* **2**, 1 (1993).
- [31] E. Isufi, A. Loukas, N. Perraudin, and G. Leus, *Forecasting time series with varma recursions on graphs*, arXiv preprint arXiv:1810.08581 (2018).
- [32] M. Viberg, *Subspace methods in system identification*, *IFAC Proceedings Volumes* **27**, 1 (1994).
- [33] K. Ogata and Y. Yang, *Modern control engineering*, Vol. 4 (Prentice hall India, 2002).
- [34] M. Verhaegen and V. Verdult, *Filtering and system identification: a least squares approach* (Cambridge university press, 2007).
- [35] M. Verhaegen and P. Dewilde, *Subspace model identification. part i: The output-error state-space model identification class of algorithm*, *Int. J. Control* **56**, 1187 (1992).
- [36] P. Van Overschee and B. De Moor, *A unifying theorem for three subspace system identification algorithms*, *Automatica* **31**, 1853 (1995).
- [37] P. Van Overschee and B. De Moor, *Subspace identification for linear systems: Theory—Implementation—Applications* (Springer Science & Business Media, 2012).
- [38] J. Stoer and R. Bulirsch, *Introduction to numerical analysis*, Vol. 12 (Springer Science & Business Media, 2013).
- [39] M. Coutino, S. Chepuri, and G. Leus, *Sparsest network support estimation: A sub-modular approach*, in *2018 IEEE Data Science Workshop* (2018).
- [40] L. Grippo and M. Sciandrone, *On the convergence of the block nonlinear gauss–seidel method under convex constraints*, *Operations research letters* **26**, 127 (2000).
- [41] K. Nodop, R. Connolly, and F. Girardi, *The field campaigns of the european tracer experiment (etex): Overview and results*, *Atmospheric Environment* **32**, 4095 (1998).
- [42] D. Thanou, X. Dong, D. Kressner, and P. Frossard, *Learning heat diffusion graphs*, *IEEE Transactions on Signal and Information Processing over Networks* **3**, 484 (2017).

7

NETWORK TOPOLOGY IDENTIFICATION FROM PARTIAL OBSERVATIONS

*If for pleasure you look,
remember: quote yourself
somewhere in your book.*

Mario Contino

As seen in Chapter 6, first-order differential graph models can describe the dynamics of several complex network processes. Using their state-space representation, we showed that it is possible to identify the connectivity of the network from measurements taken at the nodes. However, in many cases, it is not possible to obtain observations from all the nodes due to system constraints or because there are hidden nodes. Thus, in this chapter, we focus on the problem of network topology inference when full access to nodal measurements is not possible. We present the difficulties that appear under this setting and introduce algorithms able to retrieve a representation of the connectivity of the network that is consistent with the observed network dynamics.

7.1. INTRODUCTION

Within system identification, e.g., see, [2–4], and compressing sensing and sparse recovery literature, e.g., see, [5–8], several techniques, different in flavor to those introduced

Parts of this chapter have been published in the *IEEE Transactions on Information Processing over Networks* (2020) [1]

in Chapter 6 and GSP-based techniques, have been proposed for estimating the interactions of networked systems. However, they need access to all system states and consider that the system states' rate of change is either measured directly or estimated accurately. Hence, when access to these quantities is not available, i.e., there are hidden nodes, these methods fail as pointed out in [5, 9].

To address the problem of hidden nodes in a network, i.e., partial observation of the system states, several works have advocated indirect methods to locate their position relative to the visible network structure [10, 11]. For instance, [10] showed that when a sparse recovery problem has to be solved to unveil the network structure, anomalously dense regressor vectors (non-sparse vectors) are correlated with the existence of a hidden node or with a node-dependent noise source. This observation leads to a series of attempts to differentiate the effect of a hidden node and a noise source in the network data. Most of these approaches rely on pair-wise comparisons using the so-called cancellation ratio [11]. Although it is possible to extend this kind of approach to unveil multiple entangled hidden nodes, we would need to apply the cancellation-coefficient based technique to candidate groups of nodes that are assumed connected to hidden nodes. Moreover, these methods are sensitive to basis expansion mismatch. Hence, setting an anomalous density threshold is not straightforward.

Therefore, in this chapter, we extend the results of Chapter 6 and focus our attention on the partially-observed network instance of the network topology identification problem. We identify the existing ambiguities present in the estimation procedure and develop techniques to retrieve a representation of the network structure that captures the interactions of the network elements and also the dynamics of the underlying network process.

7.1.1. CHAPTER CONTRIBUTIONS

In contrast to the case of full network observability, network topology inference from partial network observations is an ill-posed problem and it is still far from being completely understood. Thus, in this chapter, we present the following contributions broadening the state-of-the-art understanding of the topology inference problem from network measurements.

7

- We analyze the problem of network topology identification from partial-network observations and show that it is ill-posed. Further, similarly to [12], we also describe the ambiguities present when recovering the network topology from measurements that do not uniquely identify the underlying structure using cospectral graphs.
- We introduce an algorithm based on the alternating projections (AP) approach [13] that is provably globally convergent to estimate the network structure under the partial observation setting. We further prove that under mild conditions the proposed AP method converges locally with a linear rate to a feasible solution.
- Finally, we extend the topology inference problem from partial observations to instances where incomplete or inaccurate process dynamics are present. For these cases, we provide a mathematical analysis and conditions that guarantee the convergence of the proposed AP method when estimating a feasible network topology.

7.1.2. CHAPTER OUTLINE

This chapter is structured as follows. Section 7.2 first presents the challenges and ambiguities that arise when we try to estimate the network topology from partial network observations. Then, it introduces the graph inverse eigenvalue problem and its relation to the problem under study. Exploiting the link with the inverse eigenvalue problem, Section 7.3 provides an alternating-projections based method to find a feasible network structure and discusses the effect of noise and incomplete information on the estimation process. To complement the presented network topology estimation methods, Section 7.5 introduces system consistency constraints that can be enforced into the AP method to match the dynamics. Section 7.6 corroborates the theory with numerical results. Finally, Section 7.7 concludes the chapter.

7.2. THE PARTIALLY OBSERVED PROBLEM

In Chapter 6, we considered that the observables $\mathbf{y}(t)$ are available for the whole network, i.e., $\mathbf{C} = \mathbf{I}$, or more generally $\text{rank}(\mathbf{C}) = N$. This full-rank condition of the observation matrix allows for a unique estimate of the similarity matrix \mathbf{T} in (6.20). However, it is common to encounter, in practice, cases where this assumption is violated.

Instances where it not possible to observe the process on all nodes, i.e., $L \leq N$, or the observations are not informative enough, i.e., $\text{rank}(\mathbf{C}) < N$, appears in biological or chemical networks [5] where the trajectories of the genes or compounds are not directly measurable but only a few observables capturing mixtures of them. For example, in epidemics spreading, the original carrier of a virus may be hidden; in ecological networks, measurements cannot be retrieved from all ecological niches due to budget constraints. Finally, in social network diffusions [14], hidden users influencing the network dynamics have their data (information) hidden from third-party collectors.

In all these cases, we cannot find a unique transform matrix \mathbf{T} due to the rank deficiency of \mathbf{C} , therefore we only can estimate a set of equivalent matrices, i.e.,

$$\mathbf{A}_T \triangleq \mathbf{TAT}^{-1}, \quad \mathbf{B}_T \triangleq \mathbf{TB} \quad (7.1)$$

which also *realize* the system in (6.10). It follows from (7.1) that (although $\mathbf{A}_T \neq \mathbf{A}$ in general) if \mathbf{A} is diagonalizable as $\mathbf{A} = \mathbf{Q}_A \Lambda_A \mathbf{Q}_A$, then

$$\text{eig}(\mathbf{A}) = \text{eig}(\mathbf{A}_T) \quad (7.2)$$

holds, where $\text{eig}(\mathbf{A})$ are the eigenvalues of \mathbf{A} . The equality (7.2) yields since \mathbf{A} and \mathbf{A}_T are *similar* matrices¹.

In these situations, we cannot remove the ambiguity in the system matrices without additional information. In the sequel, we motivate next why this disambiguation problem is particularly hard. We further derive a method to estimate an *approximately feasible* realization of the *network topology* related to the signal subspace and to the knowledge of the (bijective) scalar mappings $\{f_{s,x}(\cdot), f_{s,u}(\cdot)\}$.

Despite that in the following we only describe how to recover the state network topology \mathcal{G}_x , we remark that an analogous approach can be carried out for retrieving the input network, \mathcal{G}_u , from the transformed matrix \mathbf{B}_T .

¹Two matrices \mathbf{X} and \mathbf{Y} are said to be similar if there exists an invertible matrix \mathbf{P} such that $\mathbf{X} = \mathbf{PYP}^{-1}$.

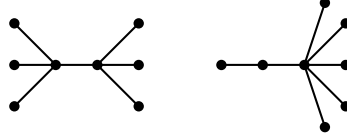


Figure 7.1: Two cospectral trees with the same number of edges. Both graphs have the same characteristic polynomial $t^4(t^4 - 7t^2 + 9)$, and hence are cospectral graphs.

7.2.1. THE GRAPH INVERSE EIGENVALUE PROBLEM

Let us consider that the shift operator \mathbf{S}_x belongs to a set \mathcal{S} containing all permissible matrices representing the state dynamics that lead to \mathbf{A}_T . That is, the set \mathcal{S} describes the properties of the graph representation matrix, e.g., zero diagonal (adjacency) $[\mathbf{S}_x]_{n,n} = 0 \forall n \in [N]$, unitary diagonal (normalized Laplacian) $[\mathbf{S}_x]_{n,n} = 1 \forall n \in [N]$, zero eigenvalue related to the constant eigenvector (combinatorial Laplacian) $\mathbf{S}_x \mathbf{1} = \mathbf{0}$, symmetry (undirected graphs) $\mathbf{S}_x = \mathbf{S}_x^\top$.

The ambiguity (7.2) introduced by the similarity transform matrix \mathbf{T} makes the problem of finding \mathbf{S}_x to become a feasibility problem, i.e.,

$$\begin{aligned} & \text{find} && \mathbf{S} \\ & \text{subject to} && \mathbf{S} \in \mathcal{S} \\ & && \text{eig}(\mathbf{S}) = \boldsymbol{\lambda} \end{aligned} \tag{7.3}$$

where $\boldsymbol{\lambda} = \text{eig}(\mathbf{S}_x)$ is the vector containing the eigenvalues of \mathbf{S}_x obtained by applying the inverse map to the eigenvalues of \mathbf{A}_T . Problem (7.3) recasts the network topology identification problem to that of finding a graph shift operator matrix \mathbf{S} that has a fixed spectrum. This problem belongs to the family of *inverse eigenvalue problems* [15]. In a way, problem (7.3) is the complement of the network spectral template approach [16]. Here, instead of having an eigenbasis and searching for a set of eigenvalues, we have a set of eigenvalues and search for an eigenbasis.

Problem (7.3) is ill-posed since its solution is non-unique in most cases. In what follows, we characterize these ambiguities in terms of equivalence classes between graphs and provide a method able to find a network topology satisfying the conditions of (7.3).

7.2.2. AMBIGUOUS GRAPHS: COSPECTRAL GRAPHS

We say there exist ambiguities in graphs when they belong to an equivalence class [17]. An example of equivalence classes, are graphs sharing their spectrum. A particular case of spectrally equivalent graphs are the so-called isomorphic graphs [18]. Two graphs \mathcal{G} and \mathcal{G}' , with respective graph shift operator matrices \mathbf{S} and \mathbf{S}' , are isomorphic if there exists a permutation matrix \mathbf{P} such that

$$\mathbf{S} = \mathbf{P}\mathbf{S}'\mathbf{P}^\top. \tag{7.4}$$

That is, the graph representation matrices are row- and column-permuted versions of each other. The permutation matrix \mathbf{P} implements the isomorphism. Therefore, if only the graph eigenvalues $\boldsymbol{\lambda}$ are available, the graphs are always indistinguishable up to node

reordering. This situation is not at all undesirable as the ordering of the nodes is often not important. However, isomorphic graphs are not the only ones sharing the spectrum.

Graphs that share the spectrum are called *cospectral* (or *isospectral*) graphs [19]. Note, however, that cospectral graphs are not necessarily isomorphic. Figure 7.1 illustrates an example of two cospectral graphs. Graph cospectrality renders the feasible set of (7.3) not a singleton and, therefore, we need to settle with any feasible graph satisfying the constraints. To put it simply, the identified topology from (7.3) will be a graph within the equivalence class of cospectral graphs regarding λ and \mathcal{S} . The following definition formalizes the latter.

Definition 7.1. (Equivalent cospectral graphs) *Two graphs \mathcal{G} and \mathcal{G}' of N nodes are cospectral equivalents with respect to the spectrum λ and the graph representation set \mathcal{S} , if they belong to the set*

$$\mathcal{C}_{\mathcal{S}}^{\lambda} := \{\mathcal{G} \mid \mathbf{S} \in \mathcal{S}, \text{eig}(\mathbf{S}) = \lambda\}.$$

As a result, the feasibility problem (7.3) reduces to a *graph construction* problem (graph inverse eigenvalue problem). So, the problem at hand can be rephrased as *given a spectrum λ and a set \mathcal{S} , construct a graph shift operator matrix $\mathbf{S} \in \mathcal{S}$ with spectrum λ* . We shall discuss next a method that addresses this construction problem.

7.3. GRAPH CONSTRUCTION BY ALTERNATING PROJECTIONS

Before detailing the graph construction method, we introduce the following assumptions.

Assumption 7.1. *The set \mathcal{S} is closed.*

Assumption 7.2. *For any $\mathbf{S} \in \mathbb{R}^{N \times N}$, the projection $P_{\mathcal{S}}(\mathbf{S})$ of \mathbf{S} onto the set \mathcal{S} is unique.*

The first assumption is technical and guarantees set \mathcal{S} includes all its limit points. The second assumption is slightly more restrictive and ensures the problem

$$P_{\mathcal{S}}(\mathbf{S}) := \underset{\hat{\mathbf{S}} \in \mathcal{S}}{\text{minimize}} \quad \|\mathbf{S} - \hat{\mathbf{S}}\|_{\text{F}} \quad (7.5)$$

has a unique solution. Although this assumption might seem restrictive, in most cases we only have access to a convex description of the feasible set \mathcal{S} which satisfies Assumption 7.2 (as the set is assumed closed) or only to the projection onto the convex approximation of the feasible set. Thus, it is fair to consider Assumption 7.2 holds in practice.

To ease exposition, we focus on the case of symmetric matrices, i.e., undirected graphs but remark that a similar approach can be followed for directed graphs². Further, denote by \mathcal{S}^N the set of symmetric $N \times N$ matrices, by \mathcal{S}_+^N the set of positive semidefinite matrices and by \mathcal{D}_N the set of $N \times N$ diagonal matrices. We then recall the following result from [21, Thm. 5.1].

Theorem 7.1 (adapted). *Given $\mathbf{S} \in \mathcal{S}^N$ and let $\mathbf{S} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$ be the spectral decomposition of \mathbf{S} with non-increasing eigenvalues $[\mathbf{\Lambda}]_{ii} \geq [\mathbf{\Lambda}]_{jj}$ for $i < j$. For a fixed $\mathbf{\Lambda}_o \in \mathcal{D}_N$ with non-increasing elements $[\mathbf{\Lambda}_o]_{ii} \geq [\mathbf{\Lambda}_o]_{jj}$ for $i < j$, a best approximant, in the Frobenius norm*

²This could be done by exchanging the spectral decomposition for the Schur decomposition [20] which decomposes a matrix into unitary matrices and an upper triangular matrix.

sense, of \mathbf{S} in the set of matrices with fixed eigenvalues

$$\mathcal{M} := \{\mathbf{M} \in \mathcal{S}^N \mid \mathbf{M} = \mathbf{V}\mathbf{\Lambda}_o\mathbf{V}^\top, \mathbf{V} \in \mathbf{O}(N)\}$$

is given by

$$P_{\mathcal{M}}(\mathbf{S}) := \mathbf{Q}\mathbf{\Lambda}_o\mathbf{Q}^\top$$

where $\mathbf{O}(N)$ denotes the set of the $N \times N$ orthogonal matrices.

Theorem 7.1 implies the projection of \mathbf{S} onto \mathcal{M} is not necessarily *unique*. As an example, consider the graph shift operator \mathbf{S} with repeated eigenvalues. Here, it is not possible to uniquely define a basis for the directions related to the eigenvalues with multiplicity larger than one. Hence, infinitely many eigendecompositions exist that lead to many projections of \mathbf{S} onto \mathcal{M} . Since every element of \mathcal{M} is uniquely determined by an element of $\mathbf{O}(N)$, the structure of \mathcal{M} is completely defined by the structure of $\mathbf{O}(N)$. Therefore, as $\mathbf{O}(N)$ is a *smooth manifold*, \mathcal{M} is one as well.

7.3.1. ALTERNATING PROJECTIONS METHOD

As it follows from Assumptions 7.1 and 7.2 and Theorem 7.1, we can project any graph shift operator matrix $\mathbf{S} \in \mathcal{S}^N$ onto \mathcal{S} and \mathcal{M} . Further, by noticing that the construction problem (7.3) is equivalent to finding a matrix in

$$\mathcal{S} \cap \mathcal{M} \tag{7.6}$$

we can consider the *alternating projections* (AP) [13] to find a point in (7.6). The AP method finds a point in the intersection of two *closed convex* sets by iteratively projecting a point onto the two sets. It performs the updates

$$\mathbf{S}_{k+1/2} = P_{\mathcal{S}}(\mathbf{S}_k) \tag{7.7a}$$

$$\mathbf{S}_{k+1} \in P_{\mathcal{M}}(\mathbf{S}_{k+1/2}) \tag{7.7b}$$

starting from a point $\mathbf{S}_0 \in \mathcal{M}$. The AP method has guaranteed convergence for convex sets and it does that linearly. However, for alternating projections on a combination of different types of sets (we have a set \mathcal{S} satisfying Assumptions 7.1 and 7.2, and a smooth manifold \mathcal{M}), additional conditions on both sets are necessary to guarantee convergence.

First, let us formalize the notion of a fixed point for the iterative procedure (7.7).

Definition 7.2 (Fixed point). *A matrix $\mathbf{S} \in \mathcal{S}^N$ is a fixed point of the alternating projections procedure in (7.7) if there exists an eigendecomposition of $P_{\mathcal{S}}(\mathbf{S})$,*

$$P_{\mathcal{S}}(\mathbf{S}) = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\top \in \mathcal{S}$$

with non-increasing elements $[\mathbf{\Lambda}]_{ii} \geq [\mathbf{\Lambda}]_{jj}$ if $i < j$ such that

$$\mathbf{S} = \mathbf{Q}\mathbf{\Lambda}_o\mathbf{Q}^\top \in \mathcal{M}.$$

This definition makes explicit two things. First, it defines \mathbf{S} as a fixed point if and only if

$$\mathbf{S} \in P_{\mathcal{M}}(P_{\mathcal{S}}(\mathbf{S})). \tag{7.8}$$

Second, when \mathbf{S} is a fixed point and $P_{\mathcal{S}}(\mathbf{S})$ has eigenvalues with multiplicity larger than one, progress can still be made towards a feasible solution when $P_{\mathcal{M}}(P_{\mathcal{S}}(\mathbf{S})) \notin \mathcal{S} \cap \mathcal{M}$. To see the latter, consider the case where an alternative eigendecomposition

$$P_{\mathcal{S}}(\mathbf{S}) = \tilde{\mathbf{Q}}\mathbf{\Lambda}\tilde{\mathbf{Q}}^{\top}, \quad (7.9)$$

is available for the fixed point \mathbf{S} . Assuming that

$$\tilde{\mathbf{S}} = \tilde{\mathbf{Q}}\mathbf{\Lambda}_o\tilde{\mathbf{Q}}^{\top} \neq \mathbf{Q}\mathbf{\Lambda}_o\mathbf{Q}^{\top} = \mathbf{S} \quad (7.10)$$

with \mathbf{Q} the eigenbasis that makes \mathbf{S} a fixed point, we can see that the new point $\tilde{\mathbf{S}}$ escapes from the fixed point. Further, since the successive projections between two closed sets is a nonincreasing function over the iterations [22, Thm. 2.3], we can prove that $\tilde{\mathbf{S}}$ presents a progress towards a feasible solution in $\mathcal{S} \cap \mathcal{M}$.³

With this in place, the following theorem shows that the AP method converges in the graph construction problem.

Theorem 7.2. *Let \mathcal{S} meet Assumptions 7.1 and 7.2 and consider the set \mathcal{M} defined in Theorem 7.1. Let also $\mathbf{S}_0, \mathbf{S}_1, \mathbf{S}_2, \dots$, be a sequence generated by the alternating projections method in (7.7). Then, there exists a limit point \mathbf{S} of this sequence that is a fixed point of (7.7) [cf. Definition 7.2] satisfying*

$$\|\mathbf{S} - P_{\mathcal{S}}(\mathbf{S})\| = \lim_{k \rightarrow \infty} \|\mathbf{S}_k - P_{\mathcal{S}}(\mathbf{S}_k)\|.$$

If the limit is zero, then $\mathbf{S} \in \mathcal{S} \cap \mathcal{M}$.

Proof. See Appendix 7.8.1 □

The above theorem proves the AP method retrieves a matrix $\mathbf{S} \in \mathcal{M}$ that realizes the original system, i.e., it preserves the underlying system modes and is an approximately *feasible network representation*. Nevertheless, Theorem 7.2 does not quantify the rate of convergence of such a method. By particularizing results for super-regular sets [23, Thm. 5.17], the following theorem shows that if the problem is feasible, locally, the proposed method converges linearly to a point in (7.6).

Theorem 7.3. *Let the set of all permissible matrices \mathcal{S} [cf. (7.3)] be convex and meet Assumptions 7.1 and 7.2. Let also the set \mathcal{M} be defined as in Theorem 7.1. Denote by $N_{\mathcal{S}}(\mathbf{S})$ the normal cone of the closed set \mathcal{S} at a point \mathbf{S} and, similarly, by $N_{\mathcal{M}}(\mathbf{S})$ the normal cone of the set \mathcal{M} at \mathbf{S} . Further, suppose that \mathcal{M} and \mathcal{S} have a strongly regular intersection at $\tilde{\mathbf{S}}$, i.e., the constant*

$$\tilde{c} = \max\{\langle u, v \rangle : u \in N_{\mathcal{M}}(\tilde{\mathbf{S}}) \cap B, v \in -N_{\mathcal{S}}(\tilde{\mathbf{S}}) \cap B\}$$

is strictly less than one with B being a closed unit Euclidean ball. Then, for any initial point $\mathbf{S}_0 \in \mathcal{M}$ close to $\tilde{\mathbf{S}}$, any sequence generated by the alternating projections method in (7.7) converges to a point in $\mathcal{M} \cap \mathcal{S}$ with R -linear rate

$$r \in (\tilde{c}, 1).$$

³ We considered the notion of fixed point to obtain a feasible set of the system matrices (matrices that realize the system) since, beyond their structure, the most important characteristic is their spectrum (set of eigenvalues).

Proof. See Appendix 7.8.2. □

These results guarantee that the AP method converges *globally* (at least) to a fixed point in \mathcal{M} and *locally*, i.e., within the neighborhood of the solution (if it exists), to a fixed point in $\mathcal{M} \cap \mathcal{S}$.

7.4. INACCURATE AND PARTIAL EIGENDECOMPOSITION

We now consider the case where the estimated eigenvalues are inexact because of noise or are incomplete because the full eigendecomposition of the system matrices is not feasible. To deal with such cases, we modify the structure of the set \mathcal{M} (in Theorem 7.1) to reflect the uncertainty and the partial eigendecomposition. The modified set has to be compatible with the structure used in Definition 7.2 and Theorems 7.2 and 7.3 to guarantee the convergence of the AP method. Thus, in the sequel, we focus on proving *compactness* for the modified versions of \mathcal{M} , which suffices to guarantee convergence of the AP method by the result of Theorem 7.2.

7.4.1. UNCERTAINTY IN THE SYSTEM MATRICES

The following proposition shows that the set \mathcal{M}_ϵ , which allows the estimated eigenvalues to lie within an ϵ -uncertainty ball, is compact.

Proposition 7.1. *Let $\Lambda_o \in \mathcal{D}_N$ with $[\Lambda_o]_{ii} \geq [\Lambda_o]_{jj}$ for $i < j$ be fixed. If $0 \leq \epsilon < \infty$ is a fixed scalar accounting for uncertainties on the elements of Λ_o , then the set*

$$\mathcal{M}_\epsilon := \{\mathbf{M} \in \mathcal{S}^N \mid \mathbf{M} = \mathbf{V}(\Lambda_o + \Lambda_\epsilon)\mathbf{V}^\top, \mathbf{V} \in \mathbf{O}(N), \quad (7.11)$$

$$\Lambda_\epsilon \in \mathcal{D}_N, \|\Lambda_\epsilon\|_2 \leq \epsilon\} \quad (7.12)$$

is compact.

Proof. See Appendix 7.8.3. □

The following result provides a best approximant of a matrix \mathbf{S} in the set \mathcal{M}_ϵ in the Frobenius norm.

Theorem 7.4. *Given $\mathbf{S} \in \mathcal{S}^N$ with eigendecomposition $\mathbf{S} = \mathbf{Q}\Lambda\mathbf{Q}^\top$ and non-increasing eigenvalues $[\Lambda]_{ii} \geq [\Lambda]_{jj}$ for $i < j$. For a fixed $\Lambda_o \in \mathcal{D}_N$ with $[\Lambda_o]_{ii} \geq [\Lambda_o]_{jj}$ for $i < j$, a best approximant of \mathbf{S} in \mathcal{M}_ϵ , in the Frobenius norm sense, is given by*

$$P_{\mathcal{M}_\epsilon}(\mathbf{S}) := \mathbf{Q}(\Lambda_o + \Lambda_\epsilon^*)\mathbf{Q}^\top$$

where

$$\Lambda_\epsilon^* := \operatorname{argmin}_{\Lambda_\epsilon \in \mathcal{D}_N} \|\Lambda - \Lambda_o - \Lambda_\epsilon\|_F, \text{ s.t. } \|\Lambda_\epsilon\|_2 \leq \epsilon.$$

Proof. See Appendix 7.8.4. □

Corollary 7.1. *The nonzero entries of Λ_ϵ^* are*

$$[\Lambda_\epsilon^*]_{ii} = \operatorname{sign}(\gamma_i) \cdot \min\{\epsilon, |\gamma_i|\}$$

with $\gamma_i := [\Lambda]_{ii} - [\Lambda_o]_{ii}$.

7.4.2. PARTIAL EIGENDECOMPOSITION

In physical systems, a discrete model of N degrees of freedom provides accurate information of about $N/3$ of the system natural frequencies [15, Ch. 5]. In other cases, the full eigendecomposition of the system matrix is not always possible. We, therefore, provide a projection onto a set that only considers a noisy part of the system matrix spectrum is available.

The following theorem provides the main result.

Theorem 7.5. *Let $\Lambda_m \in \mathcal{D}_m$ with $[\Lambda_m]_{ii} \geq [\Lambda_m]_{jj}$ for $i < j$ be fixed. If $0 \leq \epsilon < \infty$ is a fixed scalar accounting for uncertainties on the elements of Λ_m and $\rho := \max_{\mathbf{S} \in \mathcal{S}} \|\mathbf{S}\|_2$, then a best approximant, in the Frobenius norm sense, of $\mathbf{S} \in \mathcal{S}^N$, with $\|\mathbf{S}\|_2 \leq \rho$, in the set*

$$\begin{aligned} \mathcal{M}_\epsilon^m &:= \{\mathbf{M} \in \mathcal{S}^N \mid \mathbf{M} = \mathbf{V} \text{bdiag}(\Lambda_m + \Lambda_\epsilon, \bar{\Lambda}) \mathbf{V}^\top, \\ &\mathbf{V} \in \text{O}(N), \Lambda_\epsilon \in \mathcal{D}_m, \|\Lambda_\epsilon\|_2 \leq \epsilon, \bar{\Lambda} \in \mathcal{D}_{N-m}, \|\bar{\Lambda}\|_2 \leq \rho\} \end{aligned} \quad (7.13)$$

is given by

$$P_{\mathcal{M}_\epsilon^m}(\mathbf{S}) := \mathbf{Q} \text{bdiag}(\Lambda_m + \Lambda_\epsilon^*, \Lambda_{\bar{\sigma}}) \mathbf{Q}^\top.$$

Here, σ denotes the permutation of the subset of $[N]$ that solves the combinatorics problem

$$\min_{1 \leq [\sigma]_1 < \dots < [\sigma]_m \leq N} \sum_{i=1}^m ([\Lambda]_{[\sigma]_i [\sigma]_i} - [\Lambda_m]_{ii})^2 \quad (7.14)$$

where Λ is the diagonal matrix of eigenvalues of \mathbf{S} , and $\bar{\sigma}$ is the complementary set of σ . Matrix \mathbf{Q} is given by the (sorted) eigendecomposition of \mathbf{S} , i.e.,

$$\mathbf{S} = \mathbf{Q} \text{bdiag}(\Lambda_\sigma, \Lambda_{\bar{\sigma}}) \mathbf{Q}^\top$$

where Λ_σ is the permuted version of Λ and

$$\Lambda_\epsilon^* := \underset{\Lambda_\epsilon \in \mathcal{D}_m}{\text{argmin}} \|\Lambda_\sigma - \Lambda_m - \Lambda_\epsilon\|_F, \text{ s.t. } \|\Lambda_\epsilon\|_2 \leq \epsilon.$$

Furthermore, the set \mathcal{M}_ϵ^m is compact.

Proof. See Appendix 7.8.6. □

Corollary 7.2. *The optimal permutation σ of the indices $[N]$ that solves (7.14) can be found by solving a minimum-weight bipartite perfect matching problem.*

Put simply, Theorems 7.4 and 7.5 show the sets \mathcal{M}_ϵ and \mathcal{M}_ϵ^m are compact and provide a best Frobenius-norm approximant for \mathbf{S} in each case. Therefore, we can apply the AP method in (7.7) to these scenarios using the appropriate modifications. Finally, since sets \mathcal{M}_ϵ and \mathcal{M}_ϵ^m meet the conditions of Theorem 7.2, the convergence results for the AP method extend also to these scenarios.

7.5. SYSTEM CONSISTENCY CONSTRAINTS

The set \mathcal{S} [cf. (7.3)] plays an important role in the system topology that the AP method identifies. As such, it should be constrained such that the AP method yields a consistent system, i.e., the AP estimated state network should define an equivalent system to the original one. We briefly discuss here two constraints that can be added to \mathcal{S} to enforce system consistency.

By considering the system matrix [cf. (6.7)] is a bijective matrix function and by using the same construction as for the shift invariance property in (6.21), we can build the linear system

$$\begin{bmatrix} \mathbf{C}_T \\ \mathbf{C}_T f_x^{-1}(\mathbf{A}_T) \end{bmatrix} \mathbf{T} = \begin{bmatrix} \mathbf{C} \\ \mathbf{C} \mathbf{S}_x \end{bmatrix}. \quad (7.15)$$

Here, we leverage the invariance of the matrix function to nonsingular transforms, i.e.,

$$f(\mathbf{A}_T) = \mathbf{T} f(\mathbf{A}) \mathbf{T}^{-1}, \quad (7.16)$$

where $f(\cdot)$ is a matrix function, hence, can be applied to \mathbf{A}_T . In this way, we get a linear system that depends on \mathbf{S}_x and enforces the shift invariance condition. Nevertheless, the shift invariance condition does not change the optimality nor the uniqueness of the projection onto \mathcal{S} . This is because \mathbf{T} is a free optimization variable and does not affect the projection distance [cf. (7.5)].

7.6. NUMERICAL RESULTS

In this section, we present a series of numerical results to illustrate the performance of the proposed methods for different scenarios. We first illustrate how the model and the noise coloring influence the estimation performance of commonly used topology identification methods and the role of the discussed ambiguities. Then, we corroborate our theoretical results. Finally, we present results for the topology identification from partially-observed networks.

7.6.1. NETWORK TOPOLOGY IDENTIFICATION: SOCIAL GRAPH

We first make use of the instrumental variable approach of Section 6.4.3 on the Karate club graph [24] to illustrate the effect of the discussed ambiguities in different network topology estimation methods. The graph represents the connections of $N = 34$ members through 78 undirected edges. We consider the discrete model (6.6) with \mathbf{A} described by a continuous-time diffusion process $\mathbf{A} = e^{-\tau \mathbf{L}_x}$. The diffusion rate (or sampling time) is fixed to $\tau = 10^{-3}$. The input signal is randomly generated from a standard normal distribution and the power of both the state and the observation noise is $\sigma^2 = 10^{-3}$. We aim to recover the structure of the underlying graph by collecting the continuous-time diffused signals in the network at discrete times, thus $\mathbf{C} = \mathbf{I}$. The input to the network represents exogenous stimuli applied to the nodes. In social settings, this input can be interpreted as modifications to the ratings/preferences of the users which are being diffused in the network by local aggregations. The states represent the current values of the ratings/preferences of the nodes at the different sampled times. Here, we considered $\mathbf{D} = \mathbf{0}$.

We consider three different approaches to estimate the underlying network topology: *i*) a covariance-based approach, where the covariance matrix is estimated from the measurements at the nodes of the graph; *ii*) the instrumental variable approach combined with the spectral template method from [16]; and *iii*) the instrumental variable approach by enforcing the dynamics of the continuous system, i.e., model-based approach discussed in Chapter 6. These results are reported in Figure 7.2.

Figure 7.2a shows the fitting accuracy of the subspaces, while Figure 7.2b illustrates the fitting of the eigenvalues. In Figure 7.2c, we show the obtained graphs where the edges with absolute weight less than 10^{-3} are omitted. We observe that the system identification flow allows a better graph reconstruction and the proposed method offers the best alignment of the eigenbasis. Further, by leveraging the underlying physical model of the diffusion, we can reconstruct the graph spectrum with high fidelity. We also remark that despite both the basis and the spectrum are aligned, the retrieved graph looks different from the true one. This is because of the ambiguities discussed in Section 7.2.2. However, the obtained graph has the same eigenvalues as the original one and its basis diagonalizes the original network matrix. Notice that from the three methods, only the one leveraging the model information retrieves a connected graph after thresholding.

Finally, we remark that the task of estimating this topology based purely on a spectral decomposition is hard. This is because the combinatorial Laplacian of the Karate club graph has eigenvalues with a multiplicity larger than one. Thus, there is no unique basis for its eigendecomposition leading to difficulties in reconstructing the underlying topology.

7.6.2. CONVERGENCE OF THE ALTERNATING PROJECTIONS METHOD

We analyze here the convergence behavior of the alternating projections method (7.7). We present results using the sets \mathcal{M}_ϵ^m and $\mathcal{S} = \mathcal{L}_{\text{CVX}}$. The latter is the convex relaxation of the combinatorial Laplacian set; see [16]. These sets are chosen to illustrate the convergence results as \mathcal{L}_{CVX} encompasses the problem of finding Laplacian matrices with given eigenvalues and \mathcal{M}_ϵ^m is the most general set proposed in this work. For this scenario, we select a regular graph⁴ with $N = 30$ and node degree $d = 3$ and consider only half of its eigenvalues known, i.e, $m = N/2$. The AP method is analyzed for five different initial points.

These results are shown in Figure 7.3. Here, each solid line represents a different starting point. The (blue) dashed line shows the convergence behavior when the starting point is the (diagonal) eigenvalue matrix. These results show two main things. First, the predicted monotone behavior of the error $\|\mathbf{S}_k - P_{\mathcal{S}}(\mathbf{S}_k)\|_F$ holds and stagnates when a limit point is reached by the iterative sequence. Second, the error $\|\mathbf{S}_k - \mathbf{S}_{k+1}\|_F$ converges to the desired accuracy (order 10^{-6}), although not monotonically; the error convergence rate is generally linear and the starting point influences the slope. Finally, we emphasize that even when the set of known eigenvalues lies within an ϵ -ball of uncertainty, the alternating projections method converges. The convergence is guaranteed by the compactness of the set \mathcal{M}_ϵ^m .

⁴See the appendix of the chapter

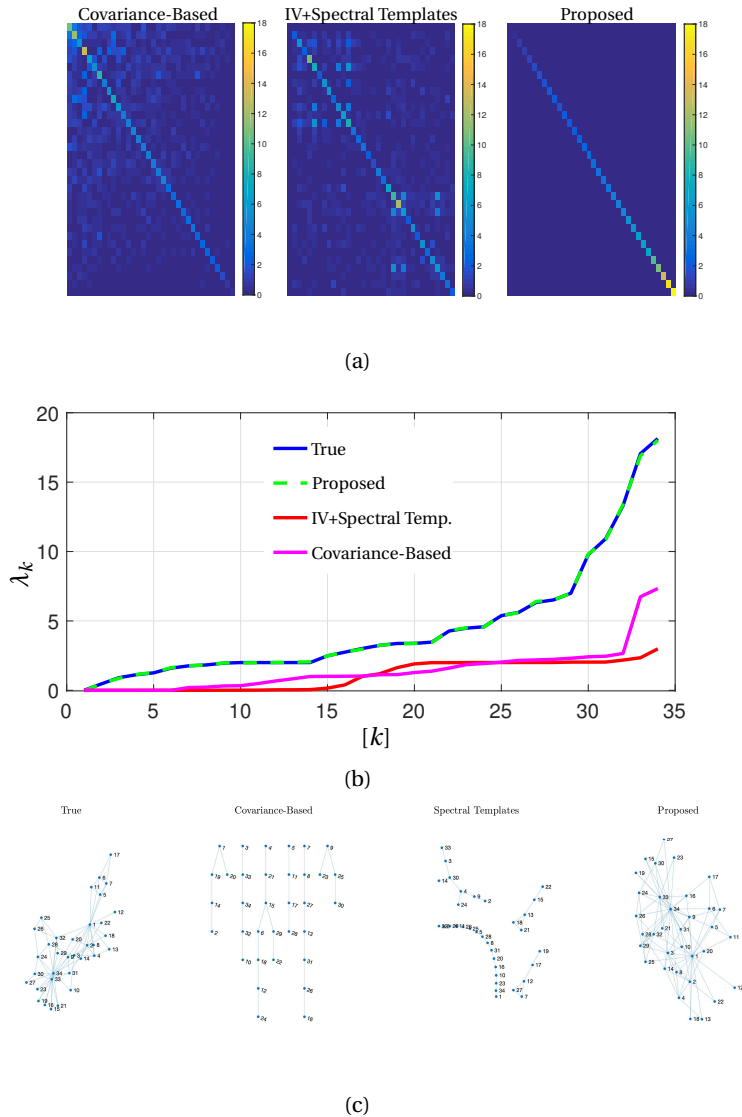


Figure 7.2: Comparison of several methods using and not using the instrumental variable approach. (a) Comparison of alignment of the eigenbasis of the estimated graphs with the ones of the true graph. (b) Comparison of eigenvalues of estimated graphs. (c) Comparison of estimated topologies.

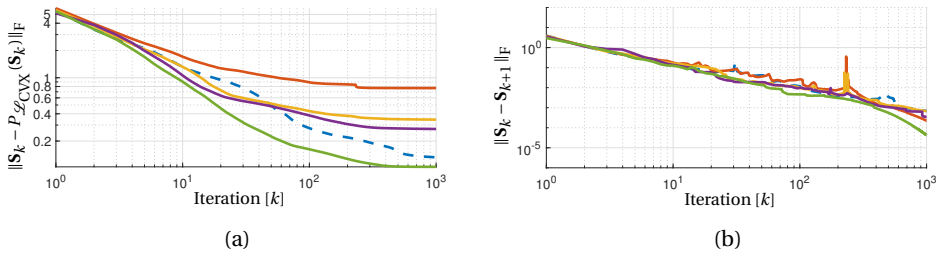


Figure 7.3: Convergence plots for the alternating projections method with M_ϵ^m and \mathcal{L}_{CVX} . (a) Error with respect to the projection, i.e., $\|\mathbf{S}_k - P_{\mathcal{L}_{CVX}}(\mathbf{S}_k)\|_F$. (b) Iterate error, i.e., $\|\mathbf{S}_k - \mathbf{S}_{k+1}\|_F$

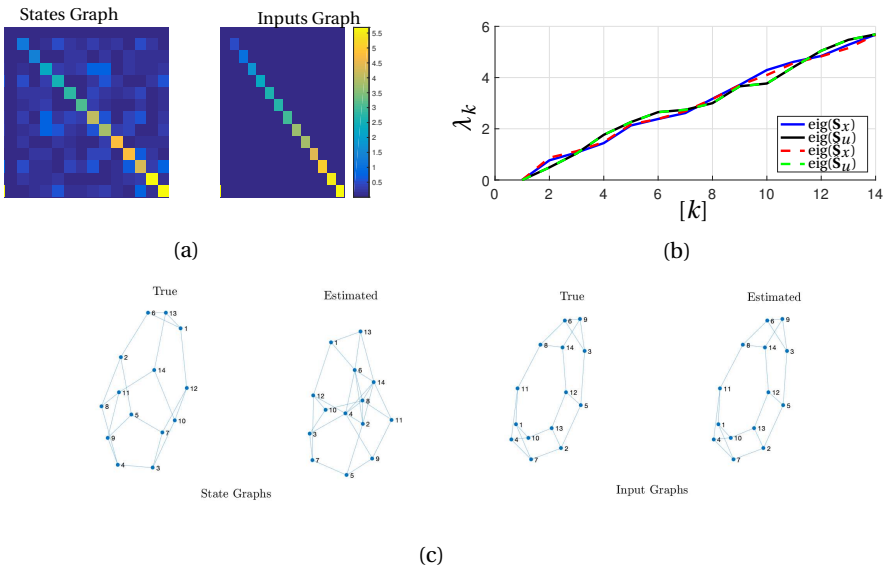


Figure 7.4: Results for the reconstruction of a graph from a dynamical system using partial observations and the alternating projections method. (a) Projection results on the modes of the estimated graphs, i.e., $|\hat{\mathbf{Q}}_*^T \mathbf{S}_* \hat{\mathbf{Q}}_*|$ with \mathbf{Q}_* being the eigenvectors of the estimated graph. (b) Comparison of the estimated graph eigenvalues. (c) Reconstructed graphs.

7.6.3. PARTIAL OBSERVATIONS

In this section, we consider the task of retrieving a graph that realizes a given system from partial observations. We consider two regular random graphs of $N = 14$ nodes and three edges per node. The data are generated from a continuous diffusion on the network and the input matrix is $\mathbf{B} = \mathbf{L}_u + \mathbf{I}$. The matrix \mathbf{C} is a Boolean matrix that selects half of the nodes (the odd labeled nodes from arbitrary labeling). Note that none of the previous methods can be employed to retrieve the network topology since $\mathbf{B} \neq \mathbf{I}$ and the network is not fully observed. Even if the covariance matrix is estimated from sampled data, its eigenstructure does not represent the eigenstructure of the state network topology.

We first estimate the system matrices using the system identification framework and then employ the AP method initialized with a random symmetric matrix that has as eigenvalues the estimated state network eigenvalues. The constraint set in (7.3) is the convex relaxation of the combinatorial Laplacian set [16]. We enrich this set with the system identification constraints to enforce the feasibility of the realization. Figure 7.4 reports the results after 30 iterations of the AP method.

From Figure 7.4a, we observe that the estimated state graph does not exactly share the eigenbasis with the original one, i.e., the graph mode projections do not form a diagonal matrix. However, we could perfectly match the input graph. This behavior is further seen in the eigenvalues, where those of the input graph are matched by the estimated eigenvalues. For the state graph, a perfect eigenvalue match is possible if a final projection onto \mathcal{M} is performed. These results are also reflected in the estimated topologies in Figure 7.4. Perfect reconstruction of the input graph support is achieved, while the state graph presents a different arrangement in the nodes and it is not regular.

Despite the differences in the state graphs, the estimated triple $\{\hat{\mathbf{S}}_x, \hat{\mathbf{S}}_u, \mathbf{C}\}$ realizes (approximately) the same system as the true triple $\{\mathbf{S}_x, \mathbf{S}_u, \mathbf{C}\}$. This is because the product of the involved system matrices is preserved, i.e., although the structure of the state graph is different, the observations can be reproduced with high confidence using the estimated system matrices. With the estimated graphs, we can predict the system output with an NRMSE fitness of $\approx 95\%$.

7

7.7. CHAPTER SUMMARY

In this chapter, we extend the framework introduced in Chapter 6 to deal with problems aiming to recover the connectivity of a network from incomplete measurements, e.g., there is no data available from all nodes. In particular, we discussed the challenges that arise when we try to retrieve the network topology from such partial observations.

We identify the problem under study as an ill-posed problem which connections with the so-called inverse-eigenvalue problem. Exploiting this link, we proposed an alternating projections method to recover a set of sparse matrices that realizes the system. Finally, we corroborated our theoretical results through numerical experiments and showed empirically the advantages that the proposed framework has compared to other alternatives that neglect the existing ambiguities.

7.8. APPENDIX

7.8.1. PROOF THEOREM 7.2

We construct the proof of the convergence of the method with arguments similar to [22].

Note that any matrix $\mathbf{S}_k \in \mathcal{M}$ can be represented by a matrix $\mathbf{Q} \in O(N)$. Then, since $O(N)$ is compact, yields the set \mathcal{M} is compact. The latter implies \mathcal{M} contains a convergent sequence and that \mathbf{S}_k has a limit point denoted by \mathbf{S}^* . Further, consider the implementation of (7.7)

$$P_{\mathcal{S}}(\mathbf{Q}_k) = \mathbf{Q}_k \mathbf{\Lambda}_k \mathbf{Q}_k^{\top},$$

and

$$\mathbf{S}_{k+1} = \mathbf{Q}_k \mathbf{\Lambda}_o \mathbf{Q}_k^{\top}.$$

From the compactness of $O(N)$, \mathbf{Q}_k converges to a point \mathbf{Q} that, in turn, makes \mathbf{S}_{k+1} converge to \mathbf{S} .

Further, since the successive projections between \mathcal{S} and \mathcal{M} is a nonincreasing function over the iterations [22, Thm. 2.3], the limit $\lim_{k \rightarrow \infty} \|\mathbf{S}_k - P_{\mathcal{S}}(\mathbf{S}_k)\|$ exists. Then, since $P_{\mathcal{S}}(\cdot)$ is a continuous operation, we have that

$$\begin{aligned} \|\mathbf{S}^* - P_{\mathcal{S}}(\mathbf{S}^*)\| &= \lim_{k \rightarrow \infty} \|\mathbf{S}_k - P_{\mathcal{S}}(\mathbf{S}_k)\| \\ &= \lim_{k \rightarrow \infty} \|\mathbf{S}_{k+1} - P_{\mathcal{S}}(\mathbf{S}_{k+1})\| \\ &= \|\mathbf{S} - P_{\mathcal{S}}(\mathbf{S})\|. \end{aligned} \tag{7.17}$$

Then, since $P_{\mathcal{S}}(\mathbf{S}_k)$ converges to $P_{\mathcal{S}}(\mathbf{S}^*)$ and the orthogonal matrix \mathbf{Q}_k converges to \mathbf{Q} , it follows that $P_{\mathcal{S}}(\mathbf{S}^*) = \mathbf{Q} \mathbf{\Lambda}^* \mathbf{Q}^{\top}$ for some $\mathbf{\Lambda}^*$. Further, the eigendecomposition $\mathbf{S} = \mathbf{Q} \mathbf{\Lambda}_o \mathbf{Q}^{\top}$ implies that $\mathbf{S} = P_{\mathcal{M}}(P_{\mathcal{S}}(\mathbf{S}^*))$.

From (7.17) and since \mathbf{S} is the projection of $P_{\mathcal{S}}(\mathbf{S}^*)$ onto \mathcal{M} , we have

$$\|\mathbf{S} - P_{\mathcal{S}}(\mathbf{S})\| \geq \|\mathbf{S} - P_{\mathcal{S}}(\mathbf{S}^*)\|.$$

By considering then that the projection of \mathbf{S} onto \mathcal{S} is unique (Assumption A.2), we have $P_{\mathcal{S}}(\mathbf{S}^*) = P_{\mathcal{S}}(\mathbf{S})$. Further, since \mathbf{S} is a projection of $P_{\mathcal{S}}(\mathbf{S}^*)$ onto \mathcal{M} , \mathbf{S} is a fixed point.

Finally, consider that the limit in (7.17) is zero. An arbitrary sequence generated by the AP method converges to some point $\tilde{\mathbf{S}} \in \mathcal{M}$, which is a limit point. From the inequality

$$\begin{aligned} \|\tilde{\mathbf{S}} - P_{\mathcal{S}}(\mathbf{S}_k)\| &= \|\tilde{\mathbf{S}} - P_{\mathcal{S}}(\mathbf{S}_k) + \mathbf{S}_k - \mathbf{S}_k\| \\ &\leq \|\mathbf{S}_k - P_{\mathcal{S}}(\mathbf{S}_k)\| + \|\tilde{\mathbf{S}} - \mathbf{S}_k\| \\ &= \|\tilde{\mathbf{S}} - \mathbf{S}_k\|, \end{aligned}$$

and by taking the limit for $k \rightarrow \infty$, we observe that $\tilde{\mathbf{S}}$ is also a limit of points in \mathcal{S} . As both sets are closed by assumption, the result follows.

7.8.2. PROOF THEOREM 7.3

Consider that \mathcal{S} and \mathcal{M} are a convex set and a smooth manifold, respectively. Therefore, they are both super-regular sets [23], which implies that \mathcal{S} and \mathcal{M} are super-regular sets at $\tilde{\mathbf{S}}$. By the results from [23, Thm. 5.15] and [23, Thm. 5.17] that guarantee the convergences of alternating projections in super-regular sets (under the same condition at the intersection as in this theorem), the convergence guarantee follows.

7.8.3. PROOF PROPOSITION 7.1

Consider that the set

$$\mathcal{D}_{N,\epsilon} := \{\mathbf{\Lambda}_\epsilon \mid \mathbf{\Lambda}_\epsilon \in \mathcal{D}_N, \|\mathbf{\Lambda}_\epsilon\|_2 \leq \epsilon\}$$

is compact. This holds since the set $\mathcal{D}_{N,\epsilon}$ is equivalent to the set of diagonal matrices whose diagonal entries lie in $[-\epsilon, \epsilon]$. Hence, the compactness.

Consider now the map $\phi: \text{O}(N) \times \mathcal{D}_{N,\epsilon} \rightarrow \mathcal{M}_\epsilon$ given by

$$\phi(\mathbf{V}, \mathbf{\Lambda}_\epsilon) = \mathbf{V}(\mathbf{\Lambda}_o + \mathbf{\Lambda}_\epsilon)\mathbf{V}^\top$$

which defines the set \mathcal{M}_ϵ and observe that $\text{O}(N)$ is compact. The map ϕ is continuous and surjective, thus the set \mathcal{M}_ϵ is compact.

7.8.4. PROOF THEOREM 7.4

The proof starts by expressing the set \mathcal{M}_ϵ as

$$\mathcal{M}_\epsilon := \bigcup_{\mathbf{\Lambda}_\epsilon \in \mathcal{D}_{N,\epsilon}} \mathcal{M}(\mathbf{\Lambda}_\epsilon), \quad (7.18)$$

where $\mathcal{M}(\mathbf{\Lambda}_\epsilon)$ is defined similarly to the set \mathcal{M} [cf. Theorem 7.1] by substituting $\mathbf{\Lambda}_o$ with $\mathbf{\Lambda}_o + \mathbf{\Lambda}_\epsilon$. Then, we can expand the projection problem for \mathbf{S} as

$$\begin{aligned} \min_{\mathbf{M} \in \mathcal{M}_\epsilon} \|\mathbf{S} - \mathbf{M}\|_F &= \min_{\mathbf{\Lambda}_\epsilon \in \mathcal{D}_{N,\epsilon}} \min_{\mathbf{M} \in \mathcal{M}(\mathbf{\Lambda}_\epsilon)} \|\mathbf{S} - \mathbf{M}\|_F \\ &= \min_{\mathbf{\Lambda}_\epsilon \in \mathcal{D}_{N,\epsilon}} \|\mathbf{S} - P_{\mathcal{M}(\mathbf{\Lambda}_\epsilon)}(\mathbf{S})\|_F \\ &= \operatorname{argmin}_{\mathbf{\Lambda}_\epsilon \in \mathcal{D}_{N,\epsilon}} \|\mathbf{S} - \mathbf{Q}(\mathbf{\Lambda}_o + \mathbf{\Lambda}_\epsilon)\mathbf{Q}^\top\|_F \\ &= \operatorname{argmin}_{\mathbf{\Lambda}_\epsilon \in \mathcal{D}_{N,\epsilon}} \|\mathbf{\Lambda} - \mathbf{\Lambda}_o - \mathbf{\Lambda}_\epsilon\|_F. \end{aligned}$$

To obtain the first equality, we used the union description of \mathcal{M}_ϵ in (7.18). The second and third equalities follow from Theorem 7.1. The last equality follows from the invariance of the Frobenius norm under unitary transforms. Finally, observe that the non-increasing ordering of eigenvalues for both $\mathbf{\Lambda}$ and $\mathbf{\Lambda}_o$ solves the last optimization problem, thus providing the best Frobenius-norm approximant for \mathbf{S} .

7.8.5. ENFORCING POSITIVE SEMIDEFINITNESS UNDER NOISE

Observe that the set \mathcal{M}_ϵ [cf. Prop. 7.1]

$$\begin{aligned} \mathcal{M}_\epsilon := \{\mathbf{M} \in \mathcal{S}^N \mid \mathbf{M} = \mathbf{V}(\mathbf{\Lambda}_o + \mathbf{\Lambda}_\epsilon)\mathbf{V}^\top, \mathbf{V} \in \text{O}(N), \\ \mathbf{\Lambda}_\epsilon \in \mathcal{D}_N, \|\mathbf{\Lambda}_\epsilon\|_2 \leq \epsilon\} \end{aligned}$$

specifies a neighborhood where the eigenvalues of \mathbf{M} must lie. However, if the desired operator \mathbf{M} is positive semidefinite (PSD), the constraint in the norm $\|\mathbf{\Lambda}_\epsilon\|_2$ might lead to matrices that are not PSD. The following proposition introduces an alternative set $\mathcal{M}_{\mathbf{a},\mathbf{b}}$ that guarantees that \mathbf{M} is PSD and shows that this new set is also compact.

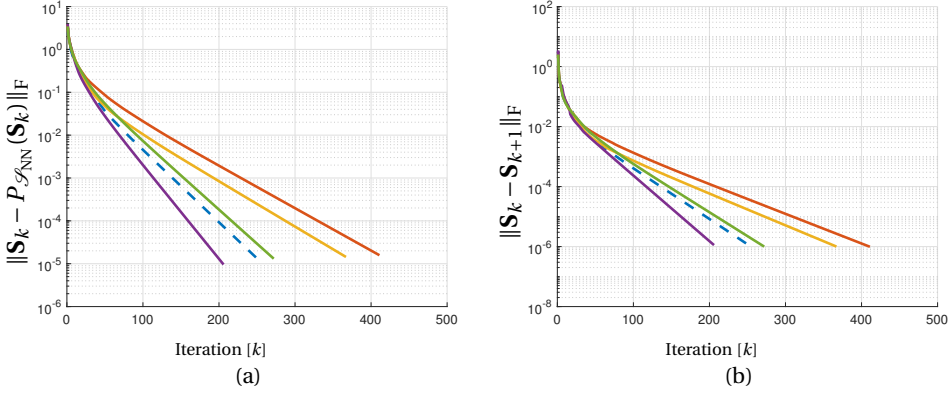


Figure 7.5: Convergence plots for the alternating projections method with \mathcal{M} and \mathcal{S}_{NN} . (a) Error with respect to the projection, i.e., $\|\mathbf{S}_k - P_{\mathcal{S}_{\text{NN}}}(\mathbf{S}_k)\|_{\text{F}}$. (b) Iterate error, i.e., $\|\mathbf{S}_k - \mathbf{S}_{k+1}\|_{\text{F}}$

Proposition 7.2. Consider the set of $N \times N$ diagonal matrices \mathcal{D}_N . Let $\Lambda_o \in \mathcal{D}_N$ be a fixed diagonal matrix with non-increasing eigenvalues $[\Lambda_o]_{ii} \geq [\Lambda_o]_{jj}$ for $i < j$. Let also $\Lambda_\epsilon \in \mathcal{D}_N$ be a diagonal matrix accounting for errors on the elements of Λ_o with bounded elements $[\Lambda_\epsilon]_{ii} \in [a_i, b_i]$.

Then, the set

$$\begin{aligned} \mathcal{M}_{\mathbf{a},\mathbf{b}} := \{ \mathbf{M} \in \mathcal{S}^N \mid \mathbf{M} = \mathbf{V}(\Lambda_o + \Lambda_\epsilon)\mathbf{V}^\top, \mathbf{V} \in \text{O}(N), \\ \Lambda_\epsilon \in \mathcal{D}_N, [\Lambda_\epsilon]_{ii} \in [a_i, b_i] \forall i \in [N] \}, \end{aligned} \quad (7.19)$$

is compact for vectors \mathbf{a} and \mathbf{b} such that $[a]_i = a_i$ and $[b]_i = b_i$.

Proof. The proof follows similarly to that of Proposition 7.1 for the set \mathcal{M}_ϵ .

Consider the map

$$\phi : \text{O}(N) \times \prod_{i=1}^N [a_i, b_i] \rightarrow \mathcal{M}_{\mathbf{a},\mathbf{b}},$$

which defines the set $\mathcal{M}_{\mathbf{a},\mathbf{b}}$ and observe that $\text{O}(N)$ and all intervals $[a_i, b_i]$ are compact.

Further, it can be shown that the map ϕ is continuous and surjective, thus the set $\mathcal{M}_{\mathbf{a},\mathbf{b}}$ is compact. Alternatively, consider that the Cartesian product of compact sets is compact (under the appropriate topology) leading to a compact set $\mathcal{M}_{\mathbf{a},\mathbf{b}}$. \square

7.8.6. PROOF THEOREM 7.5 (SKETCH.)

We first show that the set \mathcal{M}_ϵ^m is compact. Consider the map

$$\phi : \text{O}(N) \times \mathcal{D}_{m,\epsilon} \times \mathcal{D}_{N-m,\rho} \rightarrow \mathcal{M}_\epsilon^m.$$

which defines the set \mathcal{M}_ϵ^m and observe that $\text{O}(N)$, $\mathcal{D}_{m,\epsilon}$, and $\mathcal{D}_{N-m,\rho}$ are compact. Then, since the Cartesian product of compact sets is compact (under the appropriate topology), the set \mathcal{M}_ϵ^m is compact.

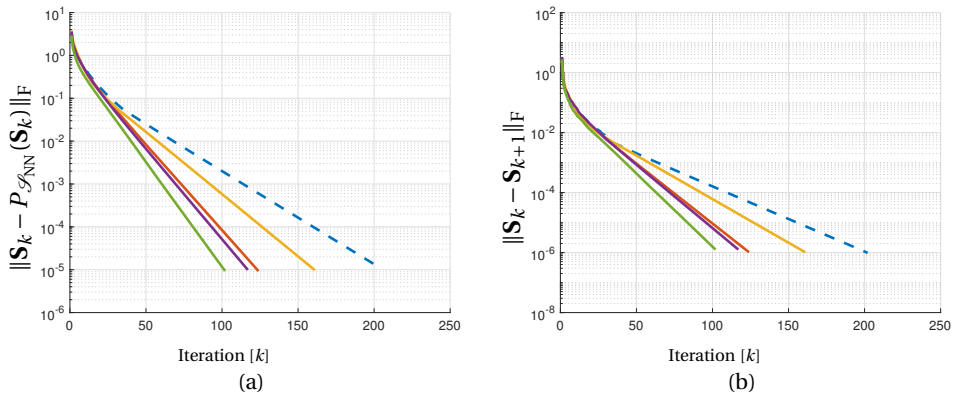


Figure 7.6: Convergence plots for the alternating projections method with \mathcal{M}_ϵ , with $\epsilon = 10^{-1}$, and \mathcal{S}_{NN} . (a) Error with respect to the projection, i.e., $\|\mathbf{S}_k - P_{\mathcal{S}_{\text{NN}}}(\mathbf{S}_k)\|_{\text{F}}$. (b) Iterate error, i.e., $\|\mathbf{S}_k - \mathbf{S}_{k+1}\|_{\text{F}}$

To prove the optimality of the provided best approximant, we proceed as follows. First, following the construction of [25, 26], it can be shown that for finding the shortest distance from \mathbf{S} to \mathcal{M}_ϵ^m , it suffices to find the shortest distance to a particular substructure of \mathcal{M}_ϵ^m defined by the permutation σ . Then, following the same construction of Theorem 7.4, it can be proven the structure of the optimal Λ_ϵ^* .

7.8.7. CONVERGENCE OF ALTERNATING PROJECTIONS

In the following, for completeness, we present further results on the convergence of the proposed alternating projections method for the discussed sets in the manuscript. Here, we fix the set \mathcal{S} as the set of nonnegative matrices, \mathcal{S}_{NN} . The selection of this feasible set is because this set encompasses the problem of finding stochastic matrices with fixed eigenvalues.

Similar results as the ones shown for the case of set \mathcal{M}_ϵ^m can be observed. The error with respect to the projection is always monotonic non-increasing, while the iterate error is not necessarily guaranteed to be monotonic. Despite this, in both cases (sets \mathcal{M} and \mathcal{M}_ϵ) the method converges to the desired accuracy of the iterates error.

For these experiments, the employed graph is a 3-regular graph with 30 nodes as shown in Figure 7.7.

REFERENCES

- [1] M. Coutino, E. Isufi, T. Maehara, and G. Leus, *State-space network topology identification from partial observations*, IEEE Transactions on Signal and Information Processing over Networks **6**, 211 (2020).
- [2] L. Ljung, *Perspectives on system identification*, Annual Reviews in Control **34**, 1 (2010).

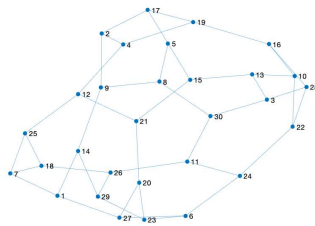


Figure 7.7: Graph employed for the numerical convergence tests.

- [3] M. Timme and J. Casadiego, *Revealing networks from dynamics: an introduction*, Journal of Physics A: Mathematical and Theoretical **47**, 343001 (2014).
- [4] W.-X. Wang, Y.-C. Lai, and C. Grebogi, *Data based identification and prediction of nonlinear and complex dynamical systems*, Physics Reports **644**, 1 (2016).
- [5] W. Pan, Y. Yuan, J. Gonçalves, and G.-B. Stan, *Reconstruction of arbitrary biochemical reaction networks: A compressive sensing approach*, in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)* (IEEE, 2012) pp. 2334–2339.
- [6] X. Han, Z. Shen, W.-X. Wang, and Z. Di, *Robust reconstruction of complex networks from sparse data*, Physical review letters **114**, 028701 (2015).
- [7] W. Pan, Y. Yuan, L. Ljung, J. Gonçalves, and G.-B. Stan, *Identification of nonlinear state-space systems from heterogeneous datasets*, IEEE Transactions on Control of Network Systems **5**, 737 (2017).
- [8] J. Casadiego, M. Nitzan, S. Hallerberg, and M. Timme, *Model-free inference of direct network interactions from nonlinear collective dynamics*, Nature communications **8**, 2192 (2017).
- [9] M. Timme, *Revealing network connectivity from response dynamics*, Physical review letters **98**, 224101 (2007).
- [10] R.-Q. Su, W.-X. Wang, and Y.-C. Lai, *Detecting hidden nodes in complex networks from time series*, Physical review E **85**, 065201 (2012).
- [11] R.-Q. Su, Y.-C. Lai, X. Wang, and Y. Do, *Uncovering hidden nodes in complex networks in the presence of noise*, Scientific reports **4**, 3944 (2014).
- [12] M. T. Angulo, J. A. Moreno, G. Lippner, A.-L. Barabási, and Y.-Y. Liu, *Fundamental limitations of network reconstruction from temporal data*, Journal of the Royal Society Interface **14**, 20160966 (2017).
- [13] H. H. Bauschke and J. M. Borwein, *On projection algorithms for solving convex feasibility problems*, SIAM review **38**, 367 (1996).
- [14] A. Guille, H. Hacid, C. Favre, and D. A. Zighed, *Information diffusion in online social networks: A survey*, ACM Sigmod Record **42**, 17 (2013).
- [15] M. Chu, G. Golub, and G. H. Golub, *Inverse eigenvalue problems: theory, algorithms, and applications*, Vol. 13 (Oxford University Press, 2005).

- [16] S. Segarra, A. G. Marques, G. Mateos, and A. Ribeiro, *Network topology inference from spectral templates*, IEEE Transactions on Signal and Information Processing over Networks (2017).
- [17] K. Devlin, *Sets, functions, and logic: an introduction to abstract mathematics* (Chapman and Hall/CRC, 2003).
- [18] F. R. Chung, *Spectral graph theory*, 92 (American Mathematical Soc., 1997).
- [19] A. E. Brouwer and W. H. Haemers, *Spectra of graphs* (Springer Science & Business Media, 2011).
- [20] G. H. Golub and C. F. Van Loan, *Matrix computations*, Vol. 3 (JHU press, 2012).
- [21] M. T. Chu and K. R. Driessel, *The projected gradient method for least squares matrix approximations with spectral constraints*, SIAM Journal on Numerical Analysis **27**, 1050 (1990).
- [22] R. Orsi, *Numerical methods for solving inverse eigenvalue problems for nonnegative matrices*, SIAM Journal on Matrix Analysis and Applications **28**, 190 (2006).
- [23] A. Lewis, R. Luke, and J. Malick, *Local convergence for alternating and averaged non-convex projections*, arXiv preprint arXiv:0709.0109 (2007).
- [24] W. W. Zachary, *An information flow model for conflict and fission in small groups*, Journal of anthropological research **33**, 452 (1977).
- [25] R. W. Brockett, *Dynamical systems that sort lists, diagonalize matrices, and solve linear programming problems*, Linear Algebra and its applications **146**, 79 (1991).
- [26] M. T. Chu, *Matrix differential equations: A continuous realization process for linear algebra problems*, Nonlinear analysis **18**, 1125 (1992).

8

CONCLUSION & RESEARCH TRENDS

*Write me a letter, sing me a song,
read me a poem; let us burn like the sun.*

Mario Coutiño

In this chapter, we summarize the contributions of this thesis within the field of graph signal processing. We provide concluding remarks in Section 8.1 and provide an overview of some possible research opportunities in Section 8.2.

8.1. CONCLUDING REMARKS

In this thesis, we defined extensions of the traditional tools of signal processing to instances involving data defined over irregular domains, e.g., graph signals. We built on these concepts and expanded the current state-of-the-art methods in graph signal processing. Specifically, we derived new results for the processing of networked data and the identification of hidden network interactions.

In Chapter 2, we presented the concept of the graph frequency domain. We leveraged the Laplace operator defined in continuous regular domains to motivate why the eigenbasis of the shift operator has useful properties to represent data. Using these properties and interpreting the matrix representation of a graph as a shift operator, we discussed filtering in irregular domains and related the variational interpretation of the Laplace operator to the notions of graph frequencies. Further, we discussed the relation between the graph frequencies and the Fourier frequencies from time domain. In addition, we discussed how graph frequencies can be ordered using total variation when the graph shift operator is not symmetric. Finally, we introduced the fundamentals of graph filtering such as the notion of graph filter response and graph modal response, and discussed open problems that were the basis for our research within the graph signal processing field.

In Chapter 3, we introduced the first contribution to networked data processing. Specifically, this contribution aimed to answer our first research question

Q.1 How can state-of-the-art distributed processing of data over networks, particularly, employing so-called graph filters, be improved?

There, we extended the traditional graph filters, in both their FIR and IIR versions, to allow edge-dependent weights. We characterized different subfamilies of these generalized graph filters that can be constructed with our proposed approach and provided design strategies, mostly based on least squares, for finding optimal parameters for the approximation of general linear operators. We showed that the introduced graph filters do not only outperform traditional graph filtering approaches in terms of communication complexity but also allow us to implement more linear operators than matrix polynomials of the graph shift operator. We also showed their competitiveness against the widely-used distributed optimization methods for practical tasks such as network averaging.

In Chapter 4, we revisited one of the main assumptions when implementing graph filters in a distributed manner: the synchronous operation of the nodes in the network. This chapter provides an answer to the second research question of this thesis

Q.2 What are the conditions required, if any, to perform distributed graph filtering in networks without synchronized units?

To address this question, we studied the setting of asynchronous graph filtering and provided a node update model useful for analyzing such an operation mode. We showed that under mild conditions an asynchronous implementation is possible and provided design guidelines for the GF coefficients to meet such requirements. Also, we connected our results with state-of-the-art results for asynchronous classical GFs and discussed the benefits of the operator splitting technique for GFs.

Building on generalized graph filters, Chapter 5 discussed the practical problems of graph filters with larger filter orders. Here, we aimed to provide an answer to our third research question

Q.3 How to implement and design, in a numerically stable manner, filtering operations on graphs that are efficient in terms of computations and communications?

Drawing parallels with time-domain filters, we proposed a cascaded implementation of graph filters that can deal with the numerical complications that appear in the design/identification stage. We then studied the behavior of the error surfaces when graph filters are reparametrized by the cascaded implementation, providing theoretical results concerning the critical points of the new design problem. Finally, we put forth the RELIEF algorithm and discussed its benefits, in terms of computations and memory, and applicability for model- and data-driven graph filter design or/and identification.

Chapter 6, digressing from graph filters, studied the problem of network topology identification from a system identification perspective, and, therefore, focused on our fourth research question

Q.4 How can we discover the system interconnection from observing the evolution of a network process through nodal measurements?

Differently from Chapter 3, in Chapter 6, we aimed to unveil the hidden structure in measured networked data. We first introduced the first-order differential graph models. We then leveraged the geometric properties of these models to establish a connection that allows us to port system identification techniques directly into the graph setting. We analyzed the fully-observed network case and showed that it is possible to apply GSP techniques for topology identification directly.

In Chapter 7, we aimed to extend the results of Chapter 6 and tackled our last research question

- Q.5 In case we cannot obtain measurements from all the nodes in the network, is it possible to estimate the underlying connectivity of the network from the available nodal measurements?

To address this question, we studied the setting of partial observations, e.g., hidden nodes, and argued that the lack of information leads to ambiguities in the connectivity of the network. After characterizing these ambiguities in the topology identification process, we introduced a global convergent procedure to obtain a network structure capable of implementing the observed process. Also, we analyzed the effect of noisy measurements or incomplete information in the proposed methods, e.g., only the leading eigenmodes of the system are available, and showed that, under minor modifications, we could employ the discussed algorithms to perform network topology identification.

8.2. RESEARCH OPPORTUNITIES

Although this thesis addressed several open questions within the field of graph signal processing, during the process of answering them, many others appeared due to its quick advance. Here, we briefly discuss some of these open questions and provide references to some initial works towards their solution.

8.2.1. GRAPH FILTERING

- (OQ1) *How to account for privacy issues arising in the processing of data over a network using graph filters?*

As nowadays people are becoming more aware of the problems that a lack of privacy brings, most of the modern applications are required to guarantee a given level of user privacy. However, although we developed distributed processing tools for analyzing networked data, in this thesis, the privacy aspect of these methods has not been thoroughly studied. Therefore, further analysis of these privacy aspects is required to provide secured data processing over networks using graph filters. We already carried out some initial research in this direction, see [1].

- (OQ2) *What are other possible design strategies for the edge-variant graph filters and their extensions?*

Although the general edge-variant graph filter does not enjoy a least-squares design, it allows for more flexibility in terms of implementable operators. By observing its structure,

parallels with graph neural networks can be drawn, see, e.g., [2]. Therefore, it is sensible to think that methods that have been proven useful for learning graph neural network parameters could work for designing generalized edge-variant graph filters and natural extensions.

(OQ3) *Is it possible to design nonlinear graph filters using decomposition techniques as the Adomian decomposition method?*

In this thesis and most of the works within GSP, graph filters are mostly linear operators. And although graph neural networks provide a framework for learning, from input-output data, topology-aware nonlinear data transforms, there is a lack of works of principled (or physically-meaningful) nonlinear graph filters. A possible direction to explore within this line of research points towards decomposition techniques used to solve partial differential equations. These techniques allow us to find the solutions of nonlinear systems without resorting to linearization. Within this body of literature, a promising method is the so-called Adomian decomposition method [3]. This approach makes use of *Adomian polynomials*, allowing for a solution method more flexible than those based on direct Taylor expansion. The expansion of operators in these recursive polynomials, and the respective reconstruction method, might allow for analytical nonlinear graph filtering while preserving the distributed spirit of GFs.

8.2.2. NETWORK TOPOLOGY IDENTIFICATION

(OQ4) *How to perform principled topology identification for time-varying first-order graph processes?*

In this thesis, we solely considered the case where the system matrices of the first-order graph process are time-invariant. This assumption translates into a fixed network topology through time. However, in many cases, the topology of the network changes through time, making the system matrices to change as well. Although there have been works addressing time-varying network topology identification, they have not considered yet first-order graph processes. Thus, further research on how to apply system identification techniques for time-varying scenarios to the network setting is needed.

(OQ5) *How to leverage the low-rank approximation of system identification frameworks to identify hypergraphs*

8

In many instances, it is not possible to guarantee that all the modes of a system can be retrieved accurately. Therefore, in many problems, a low-rank approximation of the system is the best thing we can obtain. This low-rank system realization can be interpreted as an object only retaining the per-group interactions in a network, as the granular interactions, i.e., node to node, might not be strong enough to be recovered. Therefore, despite that works as [4, 5] aimed to identify hypergraphs, which exhibit the group structures present in the data, the subspace-based techniques have not yet been explored in this venue.

(OQ6) *How to model and identify higher-order interactions in a principled manner?*

Most of the work done in this thesis aims to deal with pair-wise interactions in a graph either for processing node data or to discover pair-wise interactions. However, pair-wise interactions are not always sufficient to explain the behavior of several complex systems as biological or social networks. In these networks, small groups of interconnected nodes seem to play an important role. Therefore, there is a need for principled methods and models to identify such interactions for obtaining a further understanding of these interactions and processes. Initial research in this direction was done by us in [6, 7] recently.

REFERENCES

- [1] Q. Li, M. Coutino, G. Leus, and M. Christensen, *Privacy-Preserving Distributed Graph Filtering*, in *EURASIP European Sig. Proc. Conf. (EUSIPCO), Amsterdam, The Netherlands, 2020* (2020).
- [2] F. Gama, A. G. Marques, G. Leus, and A. Ribeiro, *Convolutional neural network architectures for signals supported on graphs*, *IEEE Transactions on Signal Processing* **67**, 1034 (2018).
- [3] G. Adomian, *Solving frontier problems of physics: the decomposition method*, Vol. 60 (Springer Science & Business Media, 2013).
- [4] D. Zhou, J. Huang, and B. Schölkopf, *Learning with hypergraphs: Clustering, classification, and embedding*, in *Advances in neural information processing systems* (2007) pp. 1601–1608.
- [5] M. Coutino, S. P. Chepuri, and G. Leus, *Learning sparse hypergraphs from dyadic relational data*, in *2019 IEEE 8th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)* (IEEE, 2019) pp. 216–220.
- [6] M. Coutino, G. Karanikolas, G. Leus, and G. B. Giannakis, *Self-driven graph volterra models for higher-order link prediction*, in *Proc. of the IEEE Int. Conf. on Acoust. Speech and Sig. Process. (ICASSP)* (2020).
- [7] Q. Yang, M. Coutino, G. Wang, G. B. Giannakis, and G. Leus, *Learning connectivity and higher-order interactions in radial distribution grids*, in *Proc. of the IEEE Int. Conf. on Acoust. Speech and Sig. Process. (ICASSP)* (2020).

AFTERWORD

*Beautiful coasts,
no matter the direction
another one awaits for you.*

Alberto Minguez

It seems this is the end. More than four years have passed, and things have ended for me. However, there is still more fabric to cut for graph signal processing. So, I will provide these closing words not as a strict conclusion but more as a recount of my experience throughout this research.

I was still working in embedded systems and computer vision when the field of graph signal processing made its first steps into the scene. Then, all was complicated. Researchers started making connections with “algebras” and other esoteric mathematical terms with network representations. The GSP field started taking shape after the notion of a “shift” in a graph was widely understood, and the idea of graph filtering was defined. Using these concepts as a foundation, around ten years ago, GSP got a pair of legs that have taken it to lengths.

With a couple of years in, GSP continued increasing its collection of tools, showing promising opportunities to model data defined over networks. The freshly-established field provided a principled way to include “prior” knowledge of the network topology to produce predictions about complex systems. Also, GSP was able to show that distributable and scalable methods, capable of coping with the data explosion from the last decade, were possible using the so-called graph filters. Moreover, it was quick in adapting tools from classical signal processing to apply them to problems in domains ranging from network sensors to healthcare systems.

Around that time, I had the opportunity to jump into GSP. And I was surprised to be bugged with the fact that, in my opinion, we were reinventing matrix polynomials (although I might have been wrong) known for decades now to numerical linear algebraists. Though we were on the right track, i.e., the connectivity was an important piece to solve the puzzle when predicting missing data or denoising network observations, at that point, more complex models, beyond polynomials, received little attention. We were comfortable, and there was a reason for that: we knew how to deal with filters that commute with our shift operations. Our de-facto tool, and closest ally in signal processing, the “shift-invariance” was limiting what we were doing with the structures we had at hand. However, quickly after a while, researchers started looking at this issue. The goal was simple: finding graph filters that use only local communications but can implement operations that are not only matrix polynomials. The first of these kinds of filters was the so-called node-varying graph filter, which weighted nodes differently at every communication round.

Following this structure, we put forth the natural generalization of this concept, the edge-varying graph filter, which increases the versatility of the structure by allowing per-edge weighting, differently than the per-node weighting from the node-varying graph filters.

The introduction of these more complex structures did not come without challenges. Questions such as: what do they represent? Is there a parallel with time-domain filters? Do they have a frequency interpretation? Or how should we design/implement them?; started to pop up. In this thesis, we answered these questions and with direct practical implications. In our other contributions, we tried to do the same for more intricate problems which have less direct practical implications. In particular, I found myself lingering on the importance/influence of eigenvectors and eigenvalues of the matrix representation of a network on the support of a graph. This question, I believe, is as asking what is more important, nature or nurture? To the best of my knowledge, this question can only be approached successfully by considering epigenetics. Genes/eigenvectors (nature) react to the environment/eigenvalues (nurture). Thus, there is a coupling. We have a recipe, and depending on how we mix the ingredients, our soufflé might or might not pop.

After some time of thinking about these, perhaps, not so transcendent problems, I got interested in the inverse eigenvalue problem literature. All the ambiguities appearing in these problems, and the parallels with the complex interplay of the support of a matrix and its spectrum led to connecting the inverse eigenvalue problem with the network topology identification problem. We first started by filling the gaps. We wanted to bridge control theory literature to network identification, as for a long time, this connection was not established, and much less exploited. Hence, we adapted well-established methods from system identification to the problem at hand and put together principled solutions tailored for the graph setting that exposed problems of some network topology identification pipelines. We then dove in the case of partial observations. This setting turned out to be challenging as the inherent lack of information resulted in the impossibility of resolving ambiguities when estimating connections in a hidden network through observations. This road led to a stop in esoteric math (for signal processing) such as Lie groups and orbits flows that finally converged nicely to a solution for the network topology identification problem, from partial observations, through less-esoteric means.

Once the snaky road of the inverse eigenvalue problem was left behind, we needed to pick up things from the beginning. An encore, let's say. Looking back, the more complex graph filtering structures, despite their benefits, have their drawbacks. They require more memory, i.e., all newly-added degrees of freedom must be saved in memory, incur a higher design cost, i.e., the least-squares designs are not linear in the number of nodes of the graph, to name a few. Some of these problems lead us to think about a cascaded implementation to deal with the design problems and numerical issues that graph filters exhibit because of the ill-conditioning that network matrices typically have. Around this time, graph convolutional networks started gaining momentum in machine learning and signal processing. These are structures with graph filters at their core. This fact, as well as their success in recommendation systems, reinforced the idea of having a cascaded implementation for implemented graph filtering operations. Though theoretically, different graph filter realizations might be equivalent, things can always differ numerically. Hence, we devote some of our time trying to understand how to devise methods that could deliver what we expect theoretically but that, due to numerical issues, we were not observing.

Fortunately, we were successful in this aspect despite some unexpected difficulties.

Finally, towards the end of my journey alongside graphs, I had the chance to look forward. Though in the last pages of this thesis, I discussed some possibilities and research directions within the field, I here want to share part of my personal view. First, I do think that privacy in network processes will continue gaining momentum. This fact is inevitable because we are becoming more and more aware of the risk of the so-called “loss of privacy” in this more and more interconnected world. Therefore, I do believe that it is needed to consider the “security” aspect of distributed algorithms from its conception. It should become a standard discussion in any work where dealing with information exchanges. Second, we need to look forward to “systematically” include nonlinear behaviors in our models. This approach, perhaps, complicates everything. And it is not, by far, a new remark. However, I believe this is the reason why signal processing lost some footing to machine learning in the last decade. We lacked a generalized interest in “non-elegant” or “brute-force-looking” kind of solutions to new problems. We, as signal processors, love structure, and we have always been on the design side or supported by laws, i.e., we could make the systems for which we design solutions, or worked with data from hard sciences. However, data now is arriving from “soft sciences” or from systems that we cannot control/design. This issue, though many may disagree, affected us as a community. The machine learning community, on the other hand, went back to the roots of many other sciences: experimentation. They have been using their sheer numbers and gazillion of frameworks to, somehow, develop brute force solutions (this for lack of a better word from my side). And only then, try to come up with explanations for what is happening. They mastered, in my view, the modern “inverse engineering”. I feel that we have put that aside and focus, perhaps, too much on models or solutions we can directly understand (or explain) and rejected that primal way of interacting with the world: just trying things out that seem ok. Although there are problems with this approach, sciences will part ways from the “exceptionalism”, like many things in society, and will be taken to greater heights by the numbers that rally behind it. In my view, science should stop being basketball and become more football. I think we will benefit a lot from having a weak-link-sport approach in science.

Putting all this in the context of graph signal processing, this means that we need to keep pushing the understanding that we have gained from our models, and connections with time-domain filtering, to the structures that computer scientists use in their take on the subject. And, at the same time, embrace what they have achieved in the practical sense, growing in return. It will be hard and full of frustration (I already see this from close friends that are making roads in this direction). However, it will be worthy. We, as a community, have lots still to offer, it is just a matter of removing some taboos that restrain us (and them). Perhaps, I failed to embrace this direction or decided to take it on too late, but it is never late for those coming behind.

SUMMARY

To the surprise of most of us, complexity in nature spawns from simplicity. No matter how simple a basic unit is, when many of them work together, the interactions among these units lead to complexity. This complexity is present in the spreading of diseases, where slightly different policies, or conditions, might lead to very different results; or in biological systems where the interactions between elements maintain the delicate balance that keep life running. Fortunately, despite their complexity, current advances in technology have allowed us to have more than just a sneak-peak at these systems. With new views on how to observe such systems and gather data, we aim to understand the complexity within.

One of these new views comes from the field of graph signal processing which provides models and tools to understand and process data coming from such complex systems. With a principled view, coming from its signal processing background, graph signal processing establishes the basis for addressing problems involving data defined over interconnected systems by combining knowledge from graph and network theory with signal processing tools. In this thesis, our goal is to advance the current state-of-the-art by studying the processing of network data using graph filters, the workhorse of graph signal processing, and by proposing methods for identifying the topology (interactions) of a network from network measurements.

To extend the capabilities of current graph filters, the network-domain counterparts of time-domain filters, we introduce a generalization of graph filters. This new family of filters does not only provide more flexibility in terms of processing networked data distributively but also reduces the communications in typical network applications, such as distributed consensus or beamforming. Furthermore, we theoretically characterize these generalized graph filters and also propose a practical and numerically-amenable cascaded implementation.

As all methods in graph signal processing make use of the structure of the network, we require to know the topology. Therefore, identifying the network interconnections from networked data is much needed for appropriately processing this data. In this thesis, we pose the network topology identification problem through the lens of system identification and study the effect of collecting information only from part of the elements of the network. We show that by using the state-space formalism, algebraic methods can be applied to the network identification problem successfully. Further, we demonstrate that for the partially-observable case, although ambiguities arise, we can still retrieve a coherent network topology leveraging state-of-the-art optimization techniques.

SAMENVATTING

Tot verbazing van velen komt complexiteit in de natuur voort uit eenvoud. Hoe simpel een basiseenheid ook is, als er veel samenwerken, leiden de interacties tussen deze eenheden tot complexiteit. Deze complexiteit is aanwezig in de verspreiding van ziekten, waarbij verschillende beleidsmaatregelen of condities tot hele andere resultaten zouden leiden; of in biologische systemen, waar de interacties tussen elementen het delicate evenwicht in stand houden dat leven mogelijk maakt. Ondanks deze complexiteit hebben technologische vooruitgangen het gelukkig voor ons mogelijk gemaakt om meer dan enkel een kijkje achter de schermen van deze systemen te nemen. Met de komst van nieuwe mogelijkheden om deze systemen te observeren en data te verzamelen, streven we ernaar de intrinsieke complexiteit te begrijpen.

Eén van deze nieuwe inzichten komt voort uit het gebied van signaalverwerking voor grafen of "graph signal processing," dat modellen en methoden aanbiedt om data van complexe systemen te begrijpen en te verwerken. Via een principiële aanpak gebaseerd op de klassieke signaalverwerking legt "graph signal processing" de basis voor de aanpak van problemen met data gedefinieerd door onderling verbonden systemen, door de kennis van grafen- en netwerktheorie te combineren met signaalverwerkingstechnieken. Het doel van deze thesis is om de state-of-the-art uit te breiden door de verwerking van netwerkdata te bestuderen met "graph filters," het werkpaard van "graph signal processing," en door methoden te ontwikkelen voor de identificatie van de structuur (interacties) van een netwerk op basis van netwerkmetingen.

Om de mogelijkheden van de huidige "graph filters," de netwerkdomein tegenhangers van filters in het tijdsdomein, uit te breiden, introduceren we een veralgemening van "graph filters". Deze nieuwe familie van filters biedt niet alleen meer flexibiliteit in de gedistribueerde verwerking van netwerkdata, maar vermindert ook de hoeveelheid communicatie in typische netwerktoepassingen, zoals gedistribueerde consensus of bundelvorming. Bovendien ontwikkelen we een theoretische karakterisatie van deze veralgemeende "graph filters" alsook een praktische en efficiënte trapsgewijze implementatie.

Aangezien alle methoden in "graph signal processing" de structuur van het netwerk gebruiken, dienen we de topologie te kennen. De identificatie van de netwerkkinterconnecties op basis van netwerkmetingen is daarom belangrijk voor de goede verwerking van die data. In deze thesis presenteren we het probleem van de identificatie van de netwerktopologie door de lens van systeemidentificatie en bestuderen we het effect als enkel een deel van het netwerk wordt geobserveerd. Via het formalisme van de systeemidentificatie, laten we zien dat algebraïsche methoden met succes toegepast kunnen worden voor de identificatie van de netwerktopologie. Verder demonstreren we dat voor het gedeeltelijk waarneembare scenario, hoewel dubbelzinnigheden optreden, we de netwerktopologie alsnog kunnen schatten door gebruik te maken van moderne optimalisatietechnieken.

CURRICULUM VITÆ

MARIO ALBERTO COUTIÑO MINGUEZ was born in an oasis known as La Paz (“The Peace”) in the coast of the Gulf of California in Mexico. He spent most of his young years surrounded by cactus, rocks, and empty beaches. He eventually moved to the land of the Mariachi (Guadalajara, Mexico) and obtained his Bachelor’s degree at the Universidad of Guadalajara in Communications and Electronics Engineering in 2013. Shortly after that, he opened a consultancy company with some of his friends and utterly failed after two successful clients. He then moved, sponsored by CONACYT, to The Netherlands where he obtained his Master’s degree in Electrical Engineering at the TUDelft in 2016. During his Master’s program, he worked in Thales Nederland and Bang & Olufsen. There, he was introduced to array signal processing, a topic which will be closed to his heart for the years to come. During that same time, he met Radmila Príbic, Richard Heusdens, and Geert Leus, the first people to put trust in him, academically, in the old continent. After his graduation, he joined Geert Leus in a four-year Ph.D. under the “ASPIRE” project supported by NWO and with the aid of CONACYT. During his Ph.D., Mario Coutiño collaborated with more than a dozen of great scientists, worked on graph signal processing and optimization, and spent two amazing seasons in AIP RIKEN (Tokyo, Japan) and the University of Minnesota (Minneapolis, MN, US). In CAMSAP 2017, he received a best paper award for his work on graph signal processing. His general scientific interest lies in numerical optimization, linear algebra, array signal processing, and graph signal processing. He is very passionate about education and taught “Applied Convex Optimization” together with Geert Leus. Mario listens approximately 10 hours of music daily, reads books as if they were water, and writes at least 12 words every night. Despite he is more Mexican than the cactus, he considers himself a European with no national flag (for better or worse). He goes by “Mario Coutino” or “Mario Contino” in scientific papers. For the latter, he has no clue why.

