Graph filter designs and implementations

Liu, J.

DOI
10.4233/uuid:09ce864a-18d8-496e-8ff9-e0144e26bba5

Publication date
2021

Document Version
Final published version

Citation (APA)
Liu, J. (2021). *Graph filter designs and implementations*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:09ce864a-18d8-496e-8ff9-e0144e26bba5

GRAPH FILTER DESIGNS AND IMPLEMENTATIONS

GRAPH FILTER DESIGNS AND IMPLEMENTATIONS

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus prof.dr.ir. T.H.J.J. van der Hagen
chair of the Board for Doctorates
to be defended publicly on
Friday 25 June 2021 at 15:00 o'clock

by

Jiani LIU

Master of Engineering in Underwater Acoustics Engineering,
Northwestern Polytechnical University, China
born in Xi'an, China.

This dissertation has been approved by the promotor.

promotor: Prof.dr.ir. G.J.T. Leus

Composition of the doctoral committee:

Rector Magnificus,	chairperson
Prof.dr.ir. G.J.T. Leus,	Delft University of Technology, promotor
Independent members:	
Prof. dr. ir. A. J. van der Veen	Delft University of Technology
Prof. dr. P. Borgnat	École Normale Supérieure de Lyon, France
Prof. dr. G. Mateos Buckstein	University of Rochester, USA
Prof. dr. A. G. Marques	King Juan Carlos University, Spain
Prof. dr. A. Hanjalic,	Technische Universiteit Delft (reserve member)

Keywords: Graph signal processing, graph filters, adjacency, Laplacian, FIR, ARMA, linear system on graphs, graph filter implementation.

ISBN 978-94-6423-321-6

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

To my family

CONTENTS

I

PRELIMINARIES

1

INTRODUCTION

WITH the development of new technologies, big data, and data processing are reflected in all aspects of our lives. Human life in the real world is being recorded all the time resulting in all kinds of different data: from personal data through our mobile devices, like messages and call histories, banking and financial activities, social networks, location information, to data from traffic networks, satellite communication, meteorological observation, and so on. The interactions of such high-dimensional data make data representation and processing become more and more irregular and complex.

Under these circumstances, the emergence of graph signal processing (GSP) has offered a brand new framework for analyzing such data and the connections among them. As shown in Fig. 1.1, the users from social media, e.g., Twitter, Facebook and so on, can be treated as vertices and their connections, e.g., the following and friending between each other, can automatically be seen as the edges connecting them. With the vertex-edge model, signals living on top of vertices can represent high-dimensional information and data, such as the times of cooperation in an academic network containing different types of authors from varying research fields, the average salary distribution in a network consisting of people who graduated from the same university, the gossip propagation among

1

Figure 1.1: Users and their connections in a social network, e.g., Twitter, Facebook, and so on (image courtesy: Google image).

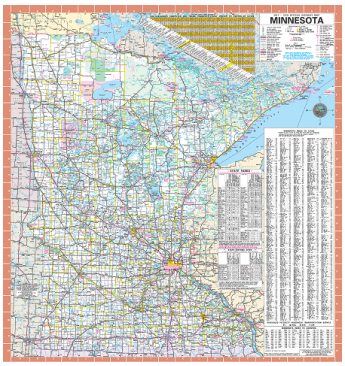
different connected girls in a social network, etc. Graph signal processing provides a meaningful representation which results in a potential improvement for some classical signal processing applications, e.g., estimation, prediction etc., which can not be fully exploited by traditional signal processing methods.

In this thesis, we contribute to the GSP field and develop fundamental signal processing techniques and algorithms. This chapter starts with a brief overview of the field of graph signal processing and in particular the related areas and applications. We investigate the background in this open field and introduce the motivation behind the current work and research. In the end, we point out the outline and contributions of this thesis.

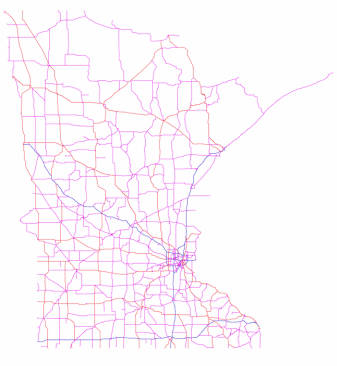
1.1. GRAPH SIGNAL PROCESSING: A BRIEF SURVEY

A graph is a fundamental mathematical structure used in various fields. Particularly in signal processing and related research areas, a graph is a data representation form that contains the structure of the signal and the topology information of the graph structure [1, 2]. Such complex graph structures can be brain networks [3], transportation networks [4], social and economic networks [5], and so on. Depending on the specific graph applications, the signal residing on the graph can be temperatures, electroencephalogram (EEG) signals, and so on. Common graphs that are utilized to describe real world data are the Erdős Rényi

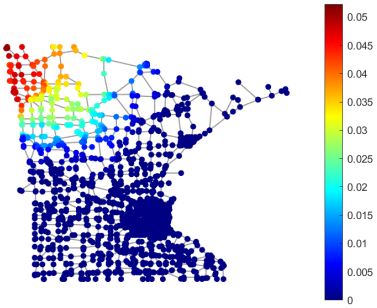
(ER) graph which is interconnected with random relationships, the small-world graph normally used by brain neurons, and so on. A useful example that illustrates the concept of signals on top of nodes is depicted in Figure 1.2. It shows the extracting process of the Minnesota traffic graph and the corresponding signal living on the graph is the transportation information on every node.



(a) Minnesota map.



(b) Highway system.



(c) Traffic graph.

Figure 1.2: Example of graph and graph signal extracting process: (a) The official Minnesota State Highway map (image courtesy: Google image). (b) The highway system of Minnesota State (image courtesy: Google image). (c) The highway Network represented by the traffic graph and the corresponding graph signals are the transportation information on every node. The image is plotted by GSPBox [6]. Note that the graph signals are computed by heat kernel filter [6].

Based on the concepts of a graph and graph signal, graph signal processing (GSP) extends some tools and definitions of classical signal processing, such as Fourier transform, filtering, and so on [1, 2], and applies them to graph data.

The main advantage of GSP is that it brings additional information of the graph structure and topology into the signal processing field. These extensions also offer us a new perspective to deal with some complex tasks in traditional signal processing.

Graph signal processing has been introduced into a number of research areas to address challenges and difficulties, such as image processing [7, 8], and network science [9, 10]. We now discuss a set of applications to show the interactions between GSP and other fields.

- *i) Sensor Networks.* A graph naturally represents the relationships and positions of sensors in a network [11, 12]. Similar data observations at neighboring nodes (sensors) lead to a smooth signal. With the smoothness information, we can build signal reconstruction methods with significant savings in cost and energy [13, 14]. Besides building sensor relationships, GSP can also be utilized for other applications in sensor networks, such as compression, data analysis, etc. Sensor networks offer a lot of opportunities for the development of GSP algorithms.

- *ii) Biological Analysis.* The biological analysis is a popular application for graph signal processing since the data observations often have a known network structure, such as the human brain. As an example, we can map the brain signals on a graph where each node (sensor) corresponds to a brain region, and the graph can provide the structural connectivity or the functional coherence between brain regions [15, 16]. Other potential GSP applications in this field can be the classification of biological signals [17], diseases analysis based on magnetic resonance imaging (MRI) [18], and so on.

- *iii) Signal Sampling.* The classical signal sampling theory shows that a bandlimited function can be perfectly recovered from its sampled sequence with high rate sampling [19]. With the graph model, some novel approaches are studied for sampling a graph by preserving the first-order difference of the original graph signal. Results show that random sampling leads to perfect recovery with a high probability for the same graph [20]. The main difference between sampling in classical signal processing and that in the graph domain is the irregular structure in the latter. The irregular topology provides us with multiple approaches to define the problem and the possibilities to make improvements in the field [21].

- *iv) Machine Learning and Deep Learning.* Since the graph model presents the natural structure and topology of a data set, graph methods play important roles in the machine learning and deep learning field. Graphs and GSP are usually the building blocks of neural network architectures that are able to deal with signals living on irregular structures. As an example, it has been studied that a multi-node version of aggregation graph neural network (GNNs) [22] can be seen as several regular convolutional neural networks (CNN) running at several designated nodes. With the property of the graph, the results are encouraging and show that the multi-node approach consistently outperforms the other architectures [22].

We only briefly discussed here some applications and examples to highlight the wide use of graphs and graph signal processing. For details and more information about GSP theory and applications, we recommend [1, 2, 21] as future reading. In this thesis, we mainly focus on one basic tool of GSP, i.e., the graph filter (GF). We propose a series of graph filter design methods and corresponding implementations. The upcoming sections quickly exploit the concepts of graph filtering and provide the thesis framework.

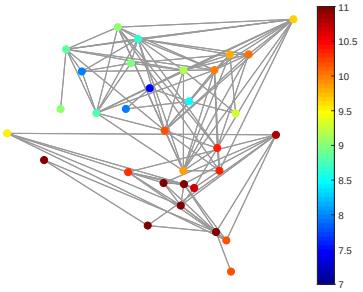
1.2. MOTIVATION

As the filters in classical signal processing, graph filters only aim at the useful spectral components of a graph signal. As such, it is important in graph signal processing to define a spectral domain. Once that is clear, the fundamental motivation for designing graph filters is to modify or extract spectral parts in terms of different objectives, e.g., using a low-pass graph filter to construct graph signals without noise.

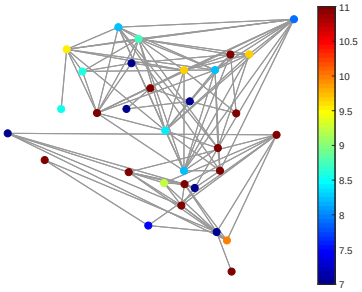
As an example, Fig.1.3 shows that a low-pass filter recovers useful signals from noisy data. In this case, the interactions and connections of the nodes (cities) are determined by their locations and can hence be described by a spatial correlation matrix. Based on that matrix, spatial denoising based on correlation information can be used but might be complex. However, we can describe this correlation matrix as a graph, as in Fig.1.3, and that can save computations because of the sparsity of the related graph. Moreover, for some more abstract cases, the nodes have no spatial location, or their spatial location is not related to their interaction. This situation normally occurs in social networks, brain networks, transportation networks, communication networks, and so on. In those cases, a graph is generally available to describe the interactions and relations be-

1

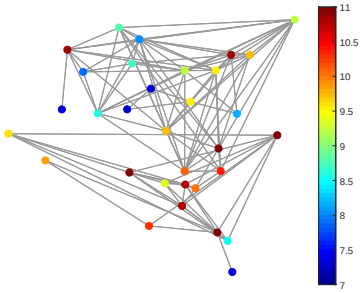
tween the nodes. Then, only a graph filter is useful and suitable for the related processing.



(a) Original Molene temperature data.



(b) Noisy Molene temperature data.



(c) Cleaned temperature data.

Figure 1.3: Illustration of the Molene temperature graph which includes different data realizations. The color on the vertex indicates the graph signal values (temperatures). The images are generated by GSPBox [6].

Some relevant applications of graph filters include graph signal reconstruction [23], denoising [24–26], smoothing [27], classification [28], recovery [29], graph clustering [30], and so on. Furthermore, a graph filter can also be seen as a basic building block for trilateral graph filters [26], graph filter banks [23, 31] and graph wavelets [32–35]. To further explain graph filter design and the implementation process, we introduce the graph Fourier transform (GFT) which is a direct analogue of the Fourier transform in classical signal processing. The GFT allows us to define graph filters in the graph frequency domain. It also shows us how a graph filter can be implemented in the vertex domain, which boils down

to exchanging messages between neighboring nodes. In Chapter 2, we will show details about the GFT and the mathematical relationship between the graph vertex and frequency domain. We will also give details for illustrating the filtering process in Fig.1.3. The description and importance of the graph frequency concept and graph filters are presented in [1, 2, 36, 37]. The challenges of defining frequencies and implementing filters are briefly discussed in [21]. Note that, in this thesis, we mainly design graph filters in the frequency domain and implement them in the graph vertex domain. Details are provided in the following chapters.

Similar to classical signal processing, we distinguish between two types of graph filters: finite impulse response (FIR) and infinite impulse response (IIR) graph filters. FIR graph filters, whose output can be computed in finite time, are common and well-studied in recent research. IIR graph filters need infinite time to compute their exact output (but approximations can be computed in finite time). They have several advantages, e.g., they can achieve better performance with smaller filter orders due to their rational structure. In Chapters 2 and 3, we will introduce details about FIR and IIR graph filters. With this background information, we show that the graph topology influences the set of graph frequencies and the resulting frequency responses, which are both important for designing graph filters.

Accordingly, the fundamental problem statement and the main motivation behind this thesis can be addressed step by step as follows:

- **Research Question 1.** *How to efficiently design graph filters without knowing the graph topology?*

  Some research has already been carried out on how to design graph filters when the graph topology, i.e., how the nodes are connected, is not known. This is often labeled as a universal graph filter design. However, some aspects still need to be investigated, e.g., how to perform universal graph filter design for directed graphs. This latter question is answered in Chapter 3. We classify graphs and graph filters into different categories and elaborate on the answers to the above question.

- **Research Question 2.** *How to efficiently exploit the potential of different graph filter structures? Also, how to obtain the best approximation accuracy through a filter with a given order?*

  Since IIR graph filters have some benefits as we mentioned, we will focus on autoregressive moving average (ARMA) graph filter design in this thesis

which is one type of IIR filters. Although distributed ARMA graph filters have been already investigated [38], there is still a research gap for the design of centralized ARMA graph filters. The detailed design methods to answer this question are provided in Chapters 4, 5, and 6. In Chapter 4, we propose the centralized ARMA filter design. In Chapter 5 and Chapter 6, we improve the filter design approaches from both a mathematical and practical perspective. Note that all the proposed design approaches can be computed without any knowledge of the graph topology. But if the graph is known, this information can also be exploited.

- **Research Question 3.** *With the designed ARMA graph filters, how to implement the filter in the vertex domain?*

  The proposed graph filter designs in this thesis mainly focus on approximating a desired response in the graph frequency domain. Using the filter coefficients, we implement filters in the vertex domain with different centralized and distributed approaches in Chapter 7. In addition, we will give a comparison of the different implementations to illustrate the performance of the proposed algorithms.

Starting from the next chapter, we will give more structural and comprehensible answers to the above-mentioned research questions. We mainly provide the answers from a graph signal processing and linear algebra perspective. The body of research and results presented in this thesis are funded by the China Scholarship Council and supported by the Circuits and Systems group, Delft University of Technology.

1.3. OUTLINE AND CONTRIBUTIONS OF THE THESIS

This thesis is divided into three main categories containing seven chapters. In this section, we show the details of each chapter.

Chapter 2.

This chapter introduces the fundamental concepts of graphs and graph signal processing. We review the mathematical descriptions and representations of the graph model and graph signal. From a linear algebra perspective, we develop the graph model called shift operator which is a matrix containing the graph connections. With these definitions and notations, we quickly interpret and formulate the vertex domain and frequency domain of a graph model, in particular, the

transformation between them, i.e., the Graph Fourier transform. The definition of the GFT can be utilized to introduce the graph filter. Then, we finalize this chapter by reviewing some characteristics of a graph filter.

Chapter 3.

In this chapter, we start with the definition of a graph filter based on the notation and graph model developed in Chapter 2. We group the graph filters into different types, i.e., finite impulse response (FIR) and infinite impulse response (IIR) graph filters, and discuss their advantages and disadvantages.

With the formulation of the directed and undirected graph models in Chapter 2, we extend the universal linear least squares (LLS) strategy of designing FIR graph filters from undirected to directed graphs. For either the normalized Laplacian (undirected graph) or normalized adjacency (directed graph) matrix, we sample the respective expected graph frequency area resulting in a number of frequency grid points. After the grid points have been determined, LLS is used to fit the response on these grid points.

Furthermore, we overview and summarize the state of the art of graph filters and filter design in both centralized and distributed settings. Then, we discuss the centralized autoregressive moving average (ARMA) model used in graph filter design and bring up the main design questions that will be tackled in the following chapters.

Chapter 4.

In this chapter, we focus on centralized ARMA filter design using a polynomial basis which is briefly introduced at the end of chapter 3. Based on this centralized setting, we propose two ARMA graph filter design methods, which can be adopted when the graph is known or in a universal fashion (unknown graph) by gridding the frequency domain (as done for the LLS FIR filter design). The proposed ARMA design methods work for undirected as well as directed graphs. The two methods developed in this chapter can be described as follows:

- *i) The first design approach is inspired by Prony's method [39], where a modified error between the modeled and the desired frequency response is minimized.* As for Prony's method [39], not the true error but a modified error that is linear in the unknown filter coefficients is minimized.

- *ii) The second approach minimizes the true error iteratively following the Steigliz-McBride idea [39].* As an initial condition, we can utilize the solu-

tion from the first method, thereby potentially improving the approximation accuracy of that solution.

The contributions of this chapter are published as

- J.Liu, E.Isufi and G.Leus, "Filter Design for Autoregressive Moving Average Graph Filters," in *IEEE Transactions on Signal and Information Processing over Networks*, vol.5, no.1, 2019, pp. 47-60.
- J.Liu; E. Isufi; G.Leus, "Autoregressive moving average graph filter design," In *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 2017, pp. 593-597.
- J.Liu; E.Isufi; G.Leus, "Autoregressive Moving Average Graph Filter Design," in *6th Joint WIC/IEEE Symposium on Information Theory and Signal Processing in the Benelux*, IEEE, 2016.

Chapter 5.

The proposed centralized ARMA methods in Chapter 4 suffer from the same numerical problems as the LLS method for FIR graph filters since they rely on the same polynomial basis functions. In this chapter, we mainly focus on improving the proposed design using orthogonal polynomials. Similar to the concept of orthogonal Chebyshev polynomials, this chapter aims to introduce the discrete orthogonal polynomial basis into graph filters. For both directed and undirected graphs, we will design FIR and ARMA graph filters based on orthogonal polynomial functions. As in Chapter 4, the design can be adopted when the graph is known or in a universal fashion (unknown graph) by gridding the frequency domain.

The proposed methods consider the discrete orthogonal polynomial basis and the contributions of this chapter are twofold:

- *i) We introduce the discrete orthogonal polynomial basis for the design of FIR graph filters for both undirected and directed graphs.* Since the continuous orthogonal polynomial basis (using Chebyshev polynomials) for FIR graph filters in undirected graphs is well studied, for either the normalized Laplacian (undirected graph) or the normalized adjacency (directed graph) matrix, we respectively formulate the discrete orthogonal polynomial basis for the FIR filter design.

- *ii) We introduce an efficient ARMA filter design method with the discrete orthogonal polynomial basis in both directed and undirected graphs.* For the ARMA model with a discrete orthogonal polynomial basis, we compute the orthogonal basis separately for the numerator and denominator parts. The solutions for undirected and directed graphs are formulated.

Chapter 6.

In this chapter, we propose a new filter design framework for both undirected and directed graphs. Instead of a polynomial basis (methods in Chapter 4 and Chapter 5), we focus on a partial fraction belonging to a rational polynomial basis.

Compared with polynomial basis functions, rational basis functions have a lot of numerical advantages [40–42], i.e., better interpolatory and extrapolatory performances [43]. Our approach is based on formulating the filter design as a least-squares problem and solving the error between the desired frequency response and the filter response recursively with the vector fitting method [44]. Throughout this chapter, we use FIR and ARMA filters as benchmarks to assess the performance of the proposed graph filters. Experimental results show that our algorithm can improve the performance of well-known graph filter designs.

Chapter 7.

In this chapter, we focus on the centralized and distributed implementations of the proposed ARMA graph filters. For the centralized design, the ARMA output can be simply found by solving a linear system of equations, which can be carried out efficiently with first-order methods [45] or conjugate gradient (CG) [46]. Moreover, to allow for easy distribution, we illustrate two iterative methods for solving the system which are easy to distribute and implement.

The proposed implementations can be characterized as:

- *i) We present an efficient centralized ARMA graph filter implementation for both directed and undirected graphs.* ARMA filtering of graph signals is written as a linear system of equations, which can be solved by efficient off-the-shelf algorithms, such as CG [46] for undirected graphs and BiCG[47] for directed graphs. We propose the details of these implementation algorithms and present some simulation results.

- *ii) We introduce two distributed implementations for ARMA graph filters, named the Richardson iteration and weighted Jacobi iteration.* Instead of

centralized implementations, we also propose distributed approaches [48–50] to solve the linear system. With the step size parameter, we can determine the convergence settings for both iterations.

The contribution of this chapter is submitted as

- J. Liu and G. Leus, "Implementations of the ARMA Graph Filters for a Directed Graph."(In preparation)

Chapter 8.
In this chapter, we draw some conclusions of our current work and summarize the key contributions of this thesis. Finally, we highlight some future research directions based on this thesis and graph signal processing theory.

The general notations used throughout this thesis are described as follows. We indicate by normal letters a or A a scalar variable; a bold lowercase letter $\boldsymbol{a}$ will represent a vector variable and a bold uppercase letter $\boldsymbol{A}$ a matrix variable. Furthermore, we indicate the absolute value of a by $|a|$ and the 2-norm of the vector $\boldsymbol{a}$ and matrix $\boldsymbol{A}$ by $\|\boldsymbol{a}\|_2$ and $\|\boldsymbol{A}\|_2$, respectively. a_i or $[\boldsymbol{a}]_i$ represents the i-th entry of $\boldsymbol{a}$, and similarly $A_{i,j}$ or $[\boldsymbol{A}]_{i,j}$ represents the (i,j)-th entry of $\boldsymbol{A}$. $\boldsymbol{a}^{(i)}$ will indicate the value of $\boldsymbol{a}$ after the i-th iteration. Also, $\boldsymbol{A}^{\dagger}$ represents the pseudo-inverse of matrix $\boldsymbol{A}$. We indicate the transpose and Hermitian of the matrix $\boldsymbol{A}$ by $\boldsymbol{A}^{\mathrm{T}}$ and $\boldsymbol{A}^{\mathrm{H}}$, respectively. The complex conjugate of a, $\boldsymbol{a}$, and $\boldsymbol{A}$ are represented as a^*, $\boldsymbol{a}^*$, and $\boldsymbol{A}^*$, respectively. Meanwhile, $\boldsymbol{A}\circ\boldsymbol{B}$ represents the element-wise Hadamard product. $\mathrm{diag}(\boldsymbol{A})$ represents the elements on the diagonal position of matrix $\boldsymbol{A}$ and $\mathrm{span}\{\boldsymbol{a}_1,\boldsymbol{a}_2,\ldots,\boldsymbol{a}_n\}$ represents the span of a set of vectors.

REFERENCES

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

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

[3] E. Bullmore and O. Sporns, *Complex brain networks: graph theoretical analysis of structural and functional systems,* Nature Reviews Neuroscience **10**, 186 (2009).

[4] R. Guimera, S. Mossa, A. Turtschi, and L. N. Amaral, *The worldwide air transportation network: Anomalous centrality, community structure, and cities' global roles,* Proceedings of the National Academy of Sciences **102**, 7794 (2005).

[5] M. O. Jackson, *Social and economic networks* (Princeton university press, 2010).

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

[7] Z. Wu and R. Leahy, *An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation,* IEEE Transactions on Pattern Analysis & Machine Intelligence , 1101 (1993).

[8] J. Shi and J. Malik, *Normalized cuts and image segmentation,* Departmental Papers (CIS) , 107 (2000).

[9] N. R. Council *et al.*, *Network science* (National Academies Press, 2006).

[10] K. Börner, S. Sanyal, and A. Vespignani, *Network science,* Annual review of information science and technology **41**, 537 (2007).

[11] R. Wagner, H. Choi, R. Baraniuk, and V. Delouille, *Distributed wavelet transform for irregular sensor network grids,* in *IEEE/SP 13th Workshop on Statistical Signal Processing, 2005* (IEEE, 2005) pp. 1196–1201.

[12] R. S. Wagner, R. G. Baraniuk, S. Du, D. B. Johnson, and A. Cohen, *An architecture for distributed wavelet analysis and processing in sensor networks,* in *Proceedings of the 5th international conference on Information processing in sensor networks* (ACM, 2006) pp. 243–250.

[13] M. Kaneko, G. Cheung, W.-t. Su, and C.-W. Lin, *Graph-based joint signal/power restoration for energy harvesting wireless sensor networks,* in *GLOBECOM 2017-2017 IEEE Global Communications Conference* (IEEE, 2017) pp. 1–6.

[14] A. Sakiyama, Y. Tanaka, T. Tanaka, and A. Ortega, *Efficient sensor position selection using graph signal sampling theory,* in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (IEEE, 2016) pp. 6225–6229.

[15] E. Bullmore and O. Sporns, *The economy of brain network organization,* Nature Reviews Neuroscience **13**, 336 (2012).

[16] O. Sporns, *Networks of the Brain* (MIT press, 2010).

[17] M. Ménoret, N. Farrugia, B. Pasdeloup, and V. Gripon, *Evaluating graph signal processing for neuroimaging through classification and dimensionality reduction,* in *2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)* (IEEE, 2017) pp. 618–622.

[18] C. Hu, X. Hua, J. Ying, P. M. Thompson, G. E. Fakhri, and Q. Li, *Localizing sources of brain disease progression with network diffusion model,* IEEE journal of selected topics in signal processing **10**, 1214 (2016).

[19] M. Unser, *Sampling-50 years after shannon,* Proceedings of the IEEE **88**, 569 (2000).

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

[21] 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).

[22] 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 (2019).

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

[24] S. Chen, A. Sandryhaila, J. M. Moura, and J. Kovacevic, *Signal denoising on graphs via graph filtering,* in *Signal and Information Processing (GlobalSIP), 2014 IEEE Global Conference on* (IEEE, 2014) pp. 872–876.

[25] S. Deutsch, A. Ortega, and G. Medioni, *Manifold denoising based on spectral graph wavelets,* in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on* (IEEE, 2016) pp. 4673–4677.

[26] M. Onuki, S. Ono, M. Yamagishi, and Y. Tanaka, *Graph signal denoising via trilateral filter on graph spectral domain,* IEEE Transactions on Signal and Information Processing over Networks **2**, 137 (2016).

[27] F. Zhang and E. R. Hancock, *Graph spectral image smoothing using the heat kernel,* Pattern Recognition **41**, 3328 (2008).

[28] A. Sandryhaila and J. M. Moura, *Classification via regularization on graphs.* in *GlobalSIP* (2013) pp. 495–498.

[29] S. Chen, A. Sandryhaila, J. M. Moura, and J. Kovačević, *Signal recovery on graphs: Variation minimization,* IEEE Transactions on Signal Processing **63**, 4609 (2015).

[30] N. Tremblay, G. Puy, R. Gribonval, and P. Vandergheynst, *Compressive spectral clustering,* in *Machine Learning, Proceedings of the Thirty-third International Conference (ICML 2016), June* (2016) pp. 20–22.

[31] D. B. Tay and Z. Lin, *Design of near orthogonal graph filter banks,* IEEE Signal Processing Letters **22**, 701 (2015).

[32] S. K. Narang and A. Ortega, *Compact support biorthogonal wavelet filterbanks for arbitrary undirected graphs,* IEEE transactions on signal processing **61**, 4673 (2013).

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

[34] A. Sakiyama, K. Watanabe, and Y. Tanaka, *Spectral graph wavelets and filter banks with low approximation error,* IEEE Transactions on Signal and Information Processing over Networks **2**, 230 (2016).

[35] D. I. Shuman, C. Wiesmeyr, N. Holighaus, and P. Vandergheynst, *Spectrum-adapted tight graph wavelet and vertex-frequency frames,* IEEE Transactions on Signal Processing **63**, 4223 (2015).

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

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

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

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

[40] D. Deschrijver, B. Haegeman, and T. Dhaene, *Orthonormal vector fitting: A robust macromodeling tool for rational approximation of frequency domain responses,* IEEE Transactions on advanced packaging **30**, 216 (2007).

[41] D. Deschrijver, B. Gustavsen, and T. Dhaene, *Advancements in iterative methods for rational approximation in the frequency domain,* IEEE Transactions on Power Delivery **22**, 1633 (2007).

[42] S. Grivet-Talocia and B. Gustavsen, *Passive macromodeling: Theory and applications*, Vol. 239 (John Wiley & Sons, 2015).

[43] L. N. Trefethen, *Approximation theory and approximation practice* (Siam, 2013).

[44] B. Gustavsen and A. Semlyen, *Rational approximation of frequency domain responses by vector fitting,* IEEE Transactions on power delivery **14**, 1052 (1999).

[45] D. P. Bertsekas, *Convex optimization theory* (Athena Scientific Belmont, 2009).

[46] J. R. Shewchuk *et al.*, *An introduction to the conjugate gradient method without the agonizing pain,* (1994).

[47] V. Faber and T. Manteuffel, *Necessary and sufficient conditions for the existence of a conjugate gradient method,* SIAM Journal on Numerical Analysis **21**, 352 (1984).

[48] G. Opfer and G. Schober, *Richardson's iteration for nonsymmetric matrices,* Linear algebra and its applications **58**, 343 (1984).

[49] L. Lei, *Convergence of asynchronous iteration with arbitrary splitting form,* Linear Algebra and its Applications **113**, 119 (1989).

[50] J. M. Bull and T. Freeman, *Numerical performance of an asynchronous jacobi iteration,* in *Parallel Processing: CONPAR 92—VAPP V* (Springer, 1992) pp. 361–366.

2

GRAPH SIGNAL PROCESSING

As we have already discussed in Chapter 1, graphs are mathematical structures that encode a relationship between different nodes. In this chapter, we will provide the necessary background information about *graph signal processing* (GSP). These fundamental theories will be called throughout the thesis. Thus, the main goal of this chapter is twofold:

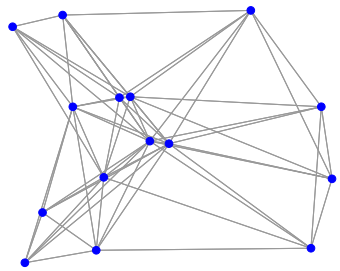
- Use mathematical tools to formulate the graph model mentioned in Chapter 1 and provide structural details about GSP.
- Formulate the basic principles of GSP as prior knowledge for the succeeding chapters.

This chapter is organized as follows. Section 2.1 briefly introduces different types of graphs and graph signal processing approaches. Then, Section 2.2 considers the graph model as a mathematical representation to organize the data (graph signal) that resides on top of networks. Section 2.3 introduces the spectral analysis of graph signals where the graph Fourier transform (GFT) is expressed as a useful tool for different graph operators, i.e., an adjacency-based operator for

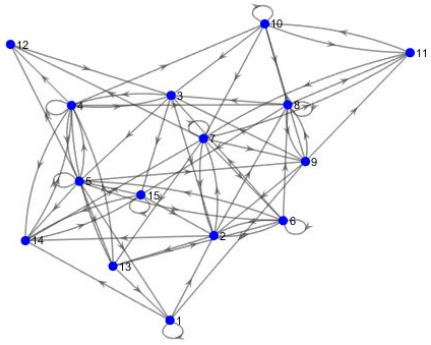
a directed graph and a Laplacian-based matrix for an undirected graph. In Section 2.4, we introduce graph filters and briefly discuss filter design. In the end, Section 2.5 concludes the chapter.

2.1. INTRODUCTION

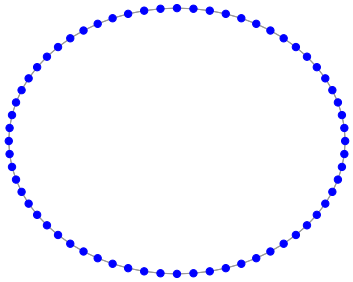
As data representation form, graphs can be classified into directed and undirected graphs depending on whether the edges have directions. Also, according to the types of connections between nodes, graphs can be sorted as ring graphs, random geometric graphs, small-world graphs, etc. We illustrate some examples in Fig.2.1.



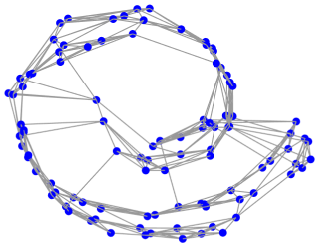
(a) Undirected graph with $N = 15$ nodes.



(b) Directed graph with $N = 15$ nodes.



(c) Ring graph with $N = 64$ nodes.



(d) Swiss roll graph with $N = 100$ nodes.

Figure 2.1: Illustration of some different types of graphs. The images are generated by GSPBox [1].

Graph signal processing, which is introduced in recent years, can be mainly separated into two categories:

- *Laplacian-based Approach.* Following spectral graph theory [2], this approach emphasizes the Laplacian matrix as a shift operator on the graph. This Laplacian also defines frequency spectra and other expansion bases for a graph model [3–5].

- *Adjacency-based Approach.* Following the linear discrete signal processing (DSP) framework, this approach depends on the adjacency matrix as a shift operator and builds up *DSP on graphs* (DSP_G) [6] tools including filtering, convolution, frequency transformation, and so on [7–9].

Both approaches enjoy an analogy for analyzing graphs through the *graph Fourier transform* (GFT). The GFT is utilized as a projection operator of a graph signal into the spectral (frequency) domain of the selected graph shift operator. Although the interpretations in the graph vertex domain are different, the GFTs for the two approaches are consistent as a decomposition of a graph signal into different frequency components which are related to the topology and the considered shift operator matrix.

Since the fundamental objectives of the two approaches are close to each other, in this thesis, we will develop our filter algorithms following both philosophies and intend to design the methods with generality. The next section recalls some background information that will be used throughout this thesis.

2.2. GRAPH MODEL

Consider a dataset with N elements and the connections between data elements are known. This model can be represented by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $\mathcal{V}$ the set of N nodes (vertices) and $\mathcal{E}$ the set of E edges.

2.2.1. GRAPH SHIFT OPERATOR

The local structure of $\mathcal{G}$ is captured by the adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{N\times N}$, where $[\boldsymbol{A}]_{j,i} \neq 0$ if there exists an edge between the nodes v_i and v_j which reflects the relation between the nodes v_i and v_j. Otherwise, $[\boldsymbol{A}]_{j,i} = 0$ indicates that there is no connection between nodes v_i and v_j. Since the relationship between nodes (data elements) can be different, in general, the non-zero edge weights $[\boldsymbol{A}]_{j,i}$ can also be different. Note that for an undirected graph $\mathcal{G}$, every edge between v_i and v_j leads to a similar edge between v_j and v_i, and thus $\boldsymbol{A}$ is symmetric, i.e., $[\boldsymbol{A}]_{i,j} = [\boldsymbol{A}]_{j,i}$. For directed graphs $\mathcal{G}$, such properties do not hold. Remark that a graph without weights on the edges is a graph for which all non-zero weights are selected as 1. In general, we will assume the weights on the edges are positive throughout this thesis.

The node degrees are characterized by the diagonal degree matrix $\boldsymbol{D}$ with

diagonal entries defined as

$$[\boldsymbol{D}]_{i,i} = \sum_{j=1}^{N} [\boldsymbol{A}]_{i,j}, \quad \text{or} \quad [\boldsymbol{D}]_{i,i} = \sum_{j=1}^{N} [\boldsymbol{A}]_{j,i} \tag{2.1}$$

which are the in-degree or out-degree matrices, and $[\boldsymbol{D}]_{i,i}$ represents the sum of all edge weights related to node v_i. Then, the discrete graph Laplacian, following spectral graph theory [2], is defined as

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}. \tag{2.2}$$

For directed graphs, some recent works introduce the use of the graph Laplacian matrix through the in-degree and out-degree matrices [10–12]. Throughout this thesis, we will use the adjacency matrix $\boldsymbol{A}$ as a representative for directed graphs, while for undirected graphs we use as an alternative the discrete graph Laplacian. In this context, the discrete graph Laplacian $\boldsymbol{L}$ for undirected graphs, which has edges without orientations, is also symmetric. We further indicate their normalized counterparts, i.e., the normalized adjacency matrix for directed graphs

$$\boldsymbol{A}_{\mathrm{n}} = \boldsymbol{A}/\|\boldsymbol{A}\|_2 \tag{2.3}$$

and the normalized Laplacian matrix for undirected graphs

$$\boldsymbol{L}_{\mathrm{n}} = \boldsymbol{D}^{-1/2}\boldsymbol{L}\boldsymbol{D}^{-1/2}. \tag{2.4}$$

Note that other alternatives can also be used for representing a graph.

As a short conclusion, every one of these graph representations can be referred to as a so-called *graph shift operator* $\boldsymbol{S}$, an operator that forms the basis for processing graph signals. Throughout this thesis, we generally consider two kinds of graph representations as graph shift operator $\boldsymbol{S}$, i.e., the adjacency matrix $\boldsymbol{A}$, and the graph Laplacian $\boldsymbol{L}$. However, sometimes other modifications are used [13, 14].

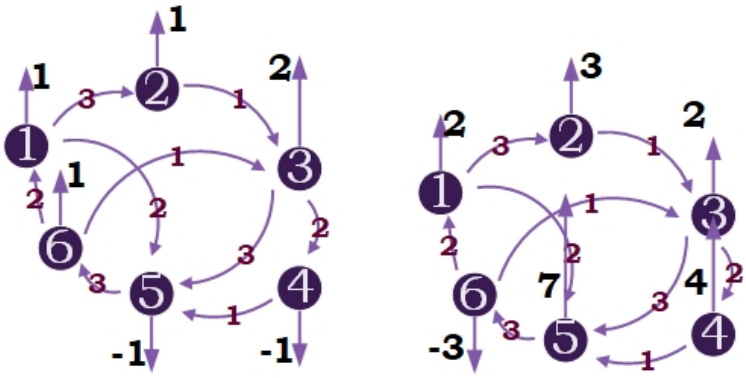
2.2.2. GRAPH SIGNAL

We will indicate with the vector $\boldsymbol{x} = [x_1, x_2, \cdots, x_N]^{\mathrm{T}} \in \mathbb{R}^{N\times 1}$ the *graph signal*, i.e., a signal living on the nodes of the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where each value x_i is associated to the corresponding node v_i.

In GSP, with the graph shift operator $\boldsymbol{S}$, the shifting of a graph signal x_n over the graph at the node v_n is defined as

$$y_n = \sum_{m=1}^{N} [\boldsymbol{S}]_{n,m} x_m, \tag{2.5}$$

which is a linear combination of the signal samples at its neighbors. Graph signal shifting is considered as a local communication between direct neighbors (nodes) and the signal y_n can be computed without any global information of the signal or graph.



(a) Directed graph with the original signal.

(b) The same graph with shifted signal.

Figure 2.2: A directed graph and the corresponding signal shifting. (a) The directed graph with the original signal. (b) The same graph with the shifted signal.

The complete shift of $\boldsymbol{x}$ by $\boldsymbol{S}$ can be written in a matrix-vector form as

$$\boldsymbol{y} = \boldsymbol{S}\boldsymbol{x}. \tag{2.6}$$

As an example, Fig. 2.2(a) shows a weighted directed graph with a graph signal. The corresponding weights are plotted on the edges. The graph signal is $\boldsymbol{x} = [1 \quad 1 \quad 2 \quad -1 \quad -1 \quad 1]^{\mathrm{T}}$ and the shift operator is $\boldsymbol{S} = \boldsymbol{A}$. The shifted signal can

be written as

$$\boldsymbol{y} = \begin{bmatrix} 2 \\ 3 \\ 2 \\ 4 \\ 7 \\ -3 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 2 \\ 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 2 & 0 & 3 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 2 \\ -1 \\ -1 \\ 1 \end{bmatrix} = \boldsymbol{A}\boldsymbol{x}.$$

Then, Fig. 2.2(b) shows the shifted version of the signal on the graph, which only needs the information from the direct neighbors.

The selection of the graph shift operator depends on the type of graph and the characteristics of the shift matrices. Different shift operators present different trade-offs. For instance, if $\boldsymbol{S}$ is the graph adjacency matrix $\boldsymbol{A}$, the shift operator can work on both directed and undirected graphs. The $\boldsymbol{A}$ matrix can also specialize the shifting process to the classical temporal DSP [15]. Meanwhile, the symmetric graph Laplacian $\boldsymbol{L}$ is normally restricted to undirected graphs. Since the matrix $\boldsymbol{L}$ is positive semi-definite and diagonalizable, the shift operator can avoid some analytical and numerical problems raised by the matrix $\boldsymbol{A}$.

For some applications, the choice of the operator should depend on the specific situation and the best trade-offs for the problem should be considered. In general, the most important factor for the selection is the difference between the *graph spectra* related to the different operators. Starting from the next section, we will introduce graph spectra and graph frequencies.

2.3. GRAPH SPECTRAL ANALYSIS

For both the graph Laplacian and adjacency approaches,the *graph Fourier transform* (GFT) can be defined as moving from the graph vertex domain to the graph frequency domain. The notion of graph frequency that extends from conventional signal processing presents a mathematical description for the frequency components of a graph signal. In this section, we mainly discuss the graph model in the frequency domain and try to understand these elementary frequencies from a theoretical perspective.

2.3.1. THE GRAPH FOURIER TRANSFORM

A graph Fourier transform is defined through the selection of a graph operator admitting a spectral decomposition.

Consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, and assume the selected operator enjoys an eigenvalue decomposition as

$$\boldsymbol{S} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{-1}, \tag{2.7}$$

where $\boldsymbol{U}$ is the eigenvector matrix with as columns the so-called graph modes $\boldsymbol{u}_1$ up to $\boldsymbol{u}_N$ (we assume the graph modes are always normalized to have a unit norm). Meanwhile, $\boldsymbol{\Lambda}$ is the diagonal eigenvalue matrix containing as diagonal entries the so-called graph frequencies $\boldsymbol{\Lambda} = \text{diag}(\lambda_1, \cdots, \lambda_N)$ (note that in this context we have $\|\boldsymbol{S}\|_2 = \max_n |\lambda_n| = |\lambda_{\max}|$).

In this thesis, we restrict ourselves to graphs for which $\boldsymbol{S}$ is real-valued and diagonalizable, meaning it enjoys an eigenvalue decomposition. If $\boldsymbol{S}$ is not diagonalizable, the eigenvalue decomposition needs to be reduced to a Jordan decomposition. We refer the reader to [6, 7] for details of dealing with this case. In practice, some inherent noise may appear in the measured graphs, e.g., social and sensor networks, leading to a non-diagonalizable matrix. For this instance, the shift operator could be modified to a diagonalizable matrix by a small perturbation within the considerable noise level [16].

To obtain the graph frequency representation of the graph signal $\boldsymbol{x}$, the eigenvector matrix $\boldsymbol{U}$ is used to transform the signal into the graph Fourier domain. Specifically, the GFT $\hat{\boldsymbol{x}}$ of $\boldsymbol{x}$ is defined as

$$\hat{\boldsymbol{x}} = \boldsymbol{U}^{-1}\boldsymbol{x} \tag{2.8}$$

and the corresponding inverse is

$$\boldsymbol{x} = \boldsymbol{U}\hat{\boldsymbol{x}}. \tag{2.9}$$

Note that, in general, when the graph operator $\boldsymbol{S}$ is real-valued but asymmetric, e.g., considering $\boldsymbol{S}$ as the adjacency matrix $\boldsymbol{A}$ for directed graphs, the corresponding graph modes $\boldsymbol{u}_n$ contain complex values and are not necessarily orthogonal to each other. Moreover, for a real-valued symmetric operator $\boldsymbol{S}$, $\boldsymbol{U}$ and $\boldsymbol{\Lambda}$ can always be selected as real-valued matrices. In this case, we have $\boldsymbol{U}^{-1} = \boldsymbol{U}^{\mathrm{T}}$ and $\boldsymbol{U}$ is orthonormal. Since we will deal with both directed and undirected graphs, in this thesis, we follow the general case (2.8).

In the following, we separate the graph into two categories, i.e., directed and undirected graphs, and list the possible choices of operators $\boldsymbol{S}$. According to the different types of shift matrices, we provide the details for the GFT and graph frequencies inside $\boldsymbol{\Lambda}$.

2.3.2. GRAPH FREQUENCY ANALYSIS WITH GFT

To fully understand the spectrum of the graph, some analysis of the graph frequencies and modes are introduced in this section. We first interpret the details for undirected graphs, and then move to the directed case.

Undirected graphs. For undirected graphs, the connections between different nodes are characterized by a symmetric adjacency matrix. Thus, candidates for $\boldsymbol{S}$ are $\boldsymbol{A}$, $\boldsymbol{L}$, and other modifications of them which are all symmetric matrices. Since the graph Laplacian is positive-semidefinite, we mainly prefer the Laplacian approach for undirected graphs.

With the decomposition (2.7), we can now rewrite the shift $\boldsymbol{S}$ for the undirected case as

$$\boldsymbol{S} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{\mathrm{T}}, \tag{2.10}$$

where all the eigenvalues λ_n are real-valued and non-negative. Then, the GFT $\hat{\boldsymbol{x}}$ of $\boldsymbol{x}$ is simplified as

$$\hat{\boldsymbol{x}} = \boldsymbol{U}^{\mathrm{T}}\boldsymbol{x}. \tag{2.11}$$

Since $\boldsymbol{S} = \boldsymbol{L}$ is real-valued, the graph modes $\boldsymbol{u}_n$ are assumed to be real-valued as well. Note that in some cases, they can be chosen to be complex-valued, e.g., for undirected circulant graphs, but that special case is not assumed in this thesis. The graph frequencies can be automatically ordered from small to large in the range of $[0, \lambda_{\max}]$, where a smaller value indicates a lower frequency [3]. For an undirected graph with $\boldsymbol{S} = \boldsymbol{L}_{\mathrm{n}}$, the graph frequencies are in the real interval from zero to two.

From the Courant-Fischer Theorem [17], the pairs of eigenvalues and eigenvectors for the symmetric Laplacian approach can also be demonstrated to be the iterative solution of the Rayleigh quotient [3, 15]

$$\lambda_1 = \min_{\substack{\boldsymbol{x} \in \mathbb{R}^N \\ \|\boldsymbol{x}\|_2 = 1}} \boldsymbol{x}^{\mathrm{T}}\boldsymbol{L}\boldsymbol{x}, \tag{2.12}$$

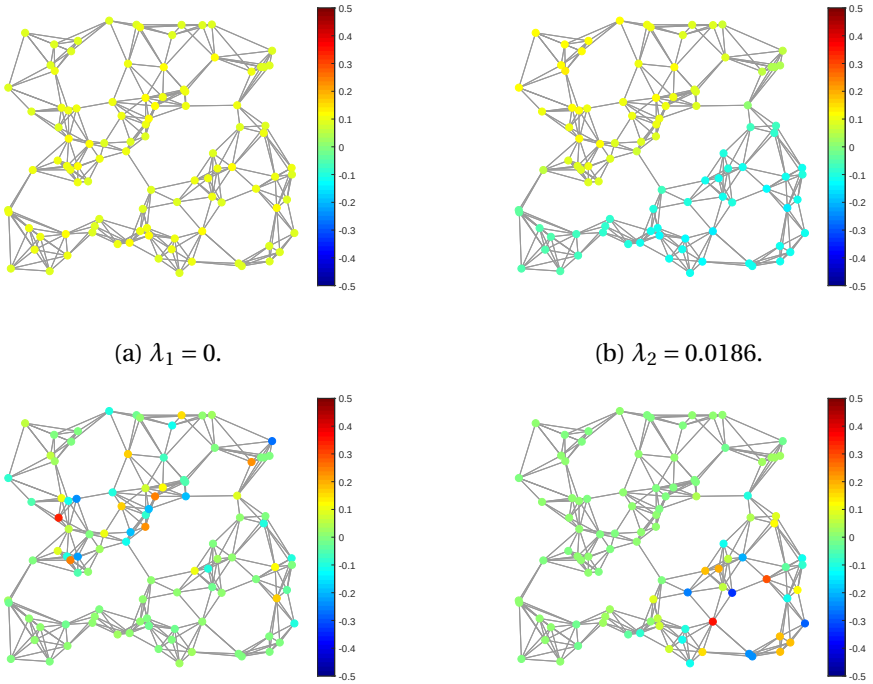
and

$$\begin{aligned} \lambda_n = & \min_{\substack{\boldsymbol{x} \in \mathbb{R}^N \\ \|\boldsymbol{x}\|_2 = 1}} \boldsymbol{x}^{\mathrm{T}}\boldsymbol{L}\boldsymbol{x},\ n = 2, 3, \cdots, N, \\ s.t. \quad & \boldsymbol{x} \perp \mathrm{span}\{\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_{n-1}\} \end{aligned} \tag{2.13}$$

where the eigenvector $\boldsymbol{u}_n$ is the minimizer of the nth problem. This quadratic form $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{L}\boldsymbol{x}$ can be used as a measure of the signal smoothness, which is

$$\boldsymbol{x}^{\mathrm{T}}\boldsymbol{L}\boldsymbol{x} = \frac{1}{2}\sum_{v_i \in \mathcal{V}} \sum_{v_j \in \mathcal{N}_i} A_{i,j}(x_i - x_j)^2 = \sum_{(v_i, v_j) \in \mathcal{E}} A_{i,j}(x_i - x_j)^2. \tag{2.14}$$

From the Laplacian quadratic problem (2.12) (2.13), as well as (2.14) we can notice that the GFT provides an orthogonal basis with an increased variation. Every additional orthogonal eigenvector minimizes the increase of variation.



(a) $\lambda_1 = 0$.

(b) $\lambda_2 = 0.0186$.

(c) $\lambda_{50} = 1.1369$.

(d) $\lambda_{100} = 1.4494$.

Figure 2.3: Example of elementary frequencies obtained from the normalized Laplacian $\boldsymbol{L}_{\mathrm{n}}$ of an undirected graph with $N = 100$ nodes. In this case, four different frequencies are shown, corresponding to different eigenvalues, ranging from the lowest frequency to the highest frequency. The entries of the related eigenvector $\mathbf{u}_i$ are shown on the nodes. The images are generated by GSPBox [1].

As illustrated by Fig. 2.3, the example shows that the eigenvectors (graph modes) $\boldsymbol{u}_n$ vibrate over the vertex set of the graph. We consider the normalized

Laplacian $\boldsymbol{L}_{\mathrm{n}}$ as the shift operator $\boldsymbol{S}$ and the index n of the eigenvalues increases with an increasing variation. In this specific case, the lowest frequency is $\lambda_1 = 0$ in Fig. 2.3(a), representing a constant value throughout the graph, and the highest frequency is $\lambda_{100} = 1.45$ in Fig. 2.3(d), where we can notice a large number of signal changes over the graph edges.

A number of previous researches have shown the relationship between the GFT and the classical Fourier transform in the signal processing field. To illustrate that, we first formulate the Fourier transform as

$$\hat{x}(\xi) := \left\langle x, e^{2\pi i \xi t} \right\rangle = \int_{\mathbb{R}} x(t) e^{-2\pi i \xi t} dt, \tag{2.15}$$

which is the expansion of a function (temporal signal) $x(t)$ based on complex exponentials. All these complex exponentials can be seen as the eigenfunctions of the one-dimensional Laplace operator [3, 15, 18]:

$$-\Delta(e^{2\pi i \xi t}) = -\frac{\partial^2}{\partial t^2} e^{2\pi i \xi t} = (2\pi\xi)^2 e^{2\pi i \xi t}. \tag{2.16}$$

Similar to the classical Fourier transform, we can also formulate the graph Fourier transform $\hat{\boldsymbol{x}}$ in (2.11) as the expansion of a graph signal $\boldsymbol{x}$ depending on the eigenvectors of the graph Laplacian:

$$\hat{x}_n := \langle \boldsymbol{x}, \boldsymbol{u}_n \rangle = \sum_{i=1}^{N} x_i u_{n,i}^*. \tag{2.17}$$

From this perspective, we can notice that the graph eigenvalues λ_n and eigenvectors $\boldsymbol{u}_n$ automatically provide a similar notion to the classical Fourier analysis. In (2.16), for a low frequency ξ, i.e., close to zero, the corresponding exponential is a slowly oscillating function. Analogously, the graph Laplacian eigenvectors associated with small values (low frequencies) λ_n change slowly over the graph.

Directed graphs. For directed graphs, the adjacency matrix $\boldsymbol{A}$ is no longer symmetric, i.e., $[\boldsymbol{A}]_{i,j}$ is not necessarily equal to $[\boldsymbol{A}]_{j,i}$, and the corresponding candidates for the graph shift operator $\boldsymbol{S}$ are not symmetric. Since the degree of node i is automatically separated into in-degree and out-degree, the graph Laplacian also contains two different realizations. In this thesis, we directly consider the adjacency matrix $\boldsymbol{A}$ (or a modification of $\boldsymbol{A}$) as the graph shift operator for this case.

With the decomposition (2.7), the graph frequencies λ_n are automatically complex-valued. To be specific, since $\boldsymbol{S}$ is real-valued, frequencies either appear in complex conjugate pairs or are purely real-valued. Moreover, the related graph modes also appear in complex conjugate pairs or are purely real-valued. As an example, for the shift operator $\boldsymbol{S} = \boldsymbol{A}_{\mathrm{n}}$, the graph frequencies are in the complex unit disc. See Fig. 2.4 for an example of a directed graph and its complex-valued graph frequencies.

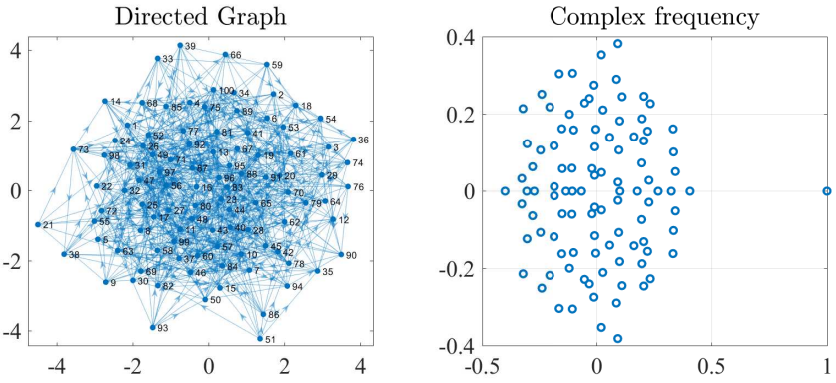


Figure 2.4: Directed graph of $N = 100$ nodes with $E = 752$ edges having different weights in the interval $[0,3]$. Complex-valued frequencies are generated by the eigenvalue decomposition of the normalized adjacency matrix $\boldsymbol{A}_{\mathrm{n}}$. The "largest" frequency has magnitude one. Some frequencies live on the real axis while the remaining frequencies appear as conjugate pairs in the complex plane.

Since the frequencies are directly related to the degree of variation of the spectral components, we can order them by relating frequencies to the complexity of the components [7]. This can be measured by the *graph total variation* of the related graph modes $\boldsymbol{u}_n$, which for a graph signal $\boldsymbol{x}$ is defined as

$$\mathrm{TV}_{\mathscr{G}}(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{A}_{\mathrm{n}}\boldsymbol{x}\|_2^2. \tag{2.18}$$

The graph total variation can be viewed as the distance between a graph signal and its shifted version. Here we highlight the use of $\boldsymbol{A}_n$ in (2.18). When the specific graph signal is a corresponding eigenvector $\boldsymbol{u}_n$ of the adjacency matrix $\boldsymbol{A}_{\mathrm{n}}$, the graph total variation $\mathrm{TV}_{\mathscr{G}}(\boldsymbol{u}_n)$ depends on the related eigenvalue λ_n as

$$\mathrm{TV}_{\mathscr{G}}(\boldsymbol{u}_n) = \left\| \mathbf{u}_n - \frac{\lambda_n}{|\lambda_{\max}|}\mathbf{u}_n \right\|_2^2 = \left| 1 - \frac{\lambda_n}{|\lambda_{\max}|} \right|, \tag{2.19}$$

where $\lambda_{\max}$ is the eigenvalue with the largest absolute value. With the definition of $\mathrm{TV}_{\mathcal{G}}(\boldsymbol{u})$, we can notice that all the eigenvectors, relating to the same eigenvalue, have the same graph total variation.

For a directed graph, the frequencies are ordered according to the similarity between the nth graph mode and its graph shifted version. In other words, the frequencies of a directed graph are ordered by their distance from $|\lambda_{\max}|$. Note that the order from the lowest to the highest frequency is not unique, due to the fact that the same distance can yield the same total variation for the corresponding components, e.g., the conjugate frequencies share the same graph total variation $\mathrm{TV}_{\mathcal{G}}(\boldsymbol{u})$. For example, in Fig. 2.4, graph frequencies closer to the point $(1,0)$ in the complex plane will represent lower frequencies in this context [7] ($\lambda_{\max} = 1$ in that case). Also, for an undirected graph, the signal total variation is related to (2.12) and (2.13).

The adjacency-based approach can also represent classical linear discrete signal processing (DSP). Finite (or periodic) time can be represented by the directed cycle graph, see Fig. 2.5 [7, 8, 19].

Figure 2.5: Using a graph representation for a finite discrete periodic time of length N.

The direction of the edges between nodes provides the time flow from the past to the future, and the last node with index N has the direction to the first node representing the periodic signal extension $x_{N+1} = x_1$. The corresponding adjacency matrix of this graph is the cyclic shift matrix given by

$$\boldsymbol{A} = \begin{bmatrix} & & & 1 \\ 1 & & & \\ & \ddots & & \\ & & 1 & \end{bmatrix}.$$

Using the eigenvalue decomposition, we can notice that the graph Fourier transform matrix is similar to the discrete Fourier transform, and the corresponding frequencies are

$$\lambda_n = e^{-j\frac{2\pi}{N}(n-1)}.$$

For either the directed or undirected graphs, computing the GFT (eigenvalue decomposition) requires $O(N^3)$ operations in general. The memory for storing the matrix $\boldsymbol{U}$ is $O(N^2)$ and applying $\boldsymbol{U}^{-1}$ to compute $\hat{\boldsymbol{x}}$ of the graph signal $\boldsymbol{x}$ costs $O(N^2)$ operations. These costs are expensive for a large graph, e.g., $N = 10^3$. One way to reduce the implementation cost is by tolerating an approximation of $\hat{\boldsymbol{x}}$ [20, 21]. Recent works show that the approximation of the GFT can be obtained with a small cost by a product method. Alternative ways to efficiently and accurately approximate the desired $\hat{\boldsymbol{x}}$ is by using filtering operations which we will introduce in the next section.

2.3.3. ILLUSTRATION OF GFT

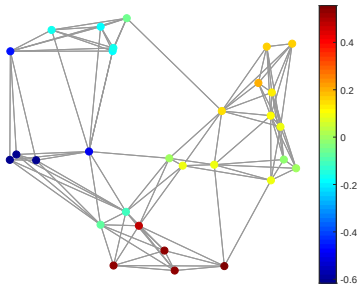
To conclude the relationship between the graph frequency coefficients and eigenvectors, the following property can now be stated:

Property 1. *For either an undirected or directed graph $\mathcal{G}$, let us denote $\hat{x}_n$ as the nth frequency coefficient of the graph signal $\boldsymbol{x}$. Then, the frequency coefficient $\hat{x}_n$ related to the real-valued graph frequency (mode) λ_n ($\boldsymbol{u}_n$) is real-valued as well. Meanwhile, the frequency coefficients $\hat{x}_n$ and $\hat{x}_{n'}$ related to the complex conjugate pair of graph frequencies (modes) λ_n and $\lambda_{n'}$ ($\boldsymbol{u}_n$ and $\boldsymbol{u}_{n'}$) form a complex conjugate pair as well.*

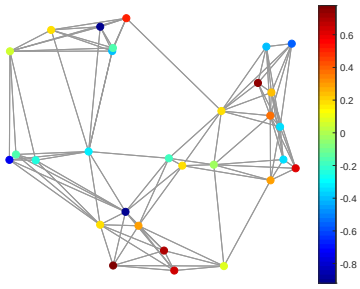
This property is built on the fact that for a real-valued matrix $\boldsymbol{S}$ for both directed and undirected graphs, eigenvalues, and eigenvectors appear in complex conjugate pairs [22, 23]. This also means that if the columns $\boldsymbol{u}_n$ and $\boldsymbol{u}_{n'}$ in the matrix $\boldsymbol{U}$ form a complex conjugate pair, the related rows in the matrix $\boldsymbol{U}^{-1}$ form a complex conjugate pair. Thus, with $\boldsymbol{U}^{-1}\boldsymbol{x}$, the frequency coefficients $\hat{x}_n$ and $\hat{x}_{n'}$ appear as a complex conjugate pair.

To illustrate the effect of the GFT on a graph signal, in Fig. 2.6(a) and (b), we take the GFT of two different graph signals on the same undirected graph. We also show the corresponding frequency coefficients in Fig. 2.6(c). Note that, in Fig. 2.6(a), the energy of the graph and graph signal concentrate on the frequencies related to the slowly varying components. On the other hand, the energy of the graph signal in Fig. 2.6(b) is associated with both high and low frequencies. Since we consider as shift operator $\boldsymbol{S} = \boldsymbol{L}_n$, the frequencies and frequency coefficients are real-valued in this case.

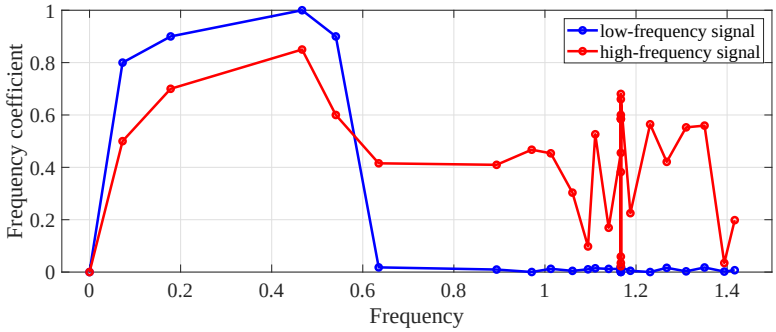
Since the GFT process depends on the matrix $\boldsymbol{U}$, it is also sensitive to the underlying graph structure which is related to the shift operator. Let us for instance consider two different graphs $\mathcal{G}_1 = (\mathcal{V}, \mathcal{E}_1)$ and $\mathcal{G}_2 = (\mathcal{V}, \mathcal{E}_2)$ with $\mathcal{V}$ the same set of



(a) Low-frequency graph signal on the graph with $N = 30$.



(b) High-frequency graph signal on the graph with $N = 30$.



(c) Corresponding GFTs (frequency coefficients) computed for the operator $\boldsymbol{S} = \boldsymbol{L}_n$.

Figure 2.6: The graph ($N = 30$) with two different types of graph signals and their corresponding GFTs. (a) and (b) represent, respectively, low- and high-frequency graph signals on the same graph. (c) Their corresponding frequency coefficients with graph shift operator $\boldsymbol{S} = \boldsymbol{L}_n$. The GFTs are normalized.

N nodes (vertices) and different edge sets $\mathscr{E}_1$ and $\mathscr{E}_2$. The different connections build up two different shift operators as $\boldsymbol{S}_1 = \boldsymbol{U}_1\boldsymbol{\Lambda}_1{\boldsymbol{U}_1}^{-1}$ and $\boldsymbol{S}_2 = \boldsymbol{U}_2\boldsymbol{\Lambda}_2{\boldsymbol{U}_2}^{-1}$. Then, with the same graph signal $\boldsymbol{x}$, we have the GFTs for $\mathscr{G}_1$ and $\mathscr{G}_2$ as $\hat{\boldsymbol{x}}_1 = {\boldsymbol{U}_1}^{-1}\boldsymbol{x}$ and $\hat{\boldsymbol{x}}_2 = {\boldsymbol{U}_2}^{-1}\boldsymbol{x}$. Note that the signal $\boldsymbol{x}$ is expanded into two different frequency bases $\boldsymbol{U}_1$ and $\boldsymbol{U}_2$. In general, changing the edges, including the directions and the weights, generates an alternative GFT even if the graph signal $\boldsymbol{x}$ stays unchanged.

2.4. GRAPH FILTERING

Together with the GFT, graph filters are a key tool to process the graph signal spectrum, i.e., to amplify or attenuate different graph frequencies. Graph filters find applications in graph signal denoising [24–26], smoothing [27], classification [28], sampling [29], recovery [30], and graph clustering [31]. Further, they serve as a basic building block for trilateral graph filters [26], graph filter banks [5, 32] and graph wavelets [4, 33–35]. In this section, we briefly introduce the graph filter concept based on the selected operator $\boldsymbol{S}$.

2.4.1. DEFINITION OF GRAPH FILTERS

We first assume that the selected graph operator $\boldsymbol{S}$, on which graph filter design is based, can be diagonalizable as mentioned in the previous section. This allows us to define a graph filter as follows.

Definition 1. *A graph filter $\boldsymbol{G}$ is a function $g(\cdot)$ applied to the graph shift operator $\boldsymbol{S}$, i.e., $\boldsymbol{G} = g(\boldsymbol{S})$, that allows for an eigen-decomposition of $\boldsymbol{G}$ in the form*

$$\boldsymbol{G} = \boldsymbol{U} g(\boldsymbol{\Lambda}) \boldsymbol{U}^{-1}, \tag{2.20}$$

where $g(\boldsymbol{\Lambda})$ is a diagonal matrix that highlights the filter impact on the graph frequencies $\boldsymbol{\Lambda}$.

Note that the decomposition of the filter on every eigenmode $\boldsymbol{u}_n$ of operator $\boldsymbol{S}$ with the corresponding frequency λ_n is related to the filter coefficient $g(\lambda_n)$. Hence, $g(\boldsymbol{\Lambda})$ has on the diagonal the *frequency response* of the graph filter, which at frequency λ_n we denote as

$$[g(\boldsymbol{\Lambda})]_{n,n} = g(\lambda_n) = \hat{g}_n.$$

In the graph vertex domain, the filter output $\boldsymbol{y}$ for a filter input $\boldsymbol{x}$ can be written as

$$\boldsymbol{y} = \boldsymbol{G}\boldsymbol{x}, \tag{2.21}$$

which in the graph frequency domain can be translated into

$$\hat{\boldsymbol{y}} = g(\boldsymbol{\Lambda})\hat{\boldsymbol{x}}, \tag{2.22}$$

where $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{y}}$ represent the input and output signals of the GFT in the frequency domain, respectively.

The graph filter $\boldsymbol{G} = g(\boldsymbol{S})$ is the operator that weights the information of the graph signal [6]. We now give some simple examples of different graph filters. For simplicity, we assume the graph frequencies are real-valued in these examples.

- *Constant filter.* The filter response of a constant filter can be formed as

$$\boldsymbol{G} = g(\boldsymbol{\Lambda}) = c\boldsymbol{I}, \tag{2.23}$$

  where c represents the constant value. For this kind of filter, all frequencies are allowed to pass with the weight c, and no frequency component is filtered out.

- *Ideal low-pass filter.* The ideal low-pass graph filter is given by

$$g(\lambda_n) = 1,\ \lambda_n < \lambda_c \ \text{ and } \ 0, \text{otherwise} \tag{2.24}$$

  where λ_c is the cut-off frequency. For this situation, only the frequencies up to cut-off frequency λ_c are allowed to pass.

- *The heat kernel.* The heat kernel is widely used and given by

$$g(\lambda_n) = e^{-c\lambda_n}, \tag{2.25}$$

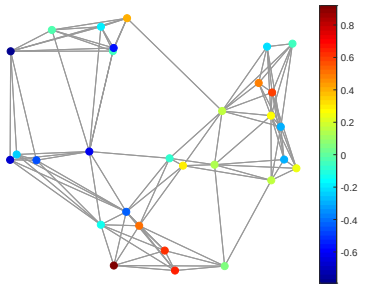
  where c is the weight. The function is exponentially decreasing with the frequency λ_n.

In Fig. 2.7, we use a graph ($N = 30$) with a graph signal to illustrate the effect of an ideal low-pass graph filter. The filtering operation in the graph Fourier domain is shown in Fig. 2.7(c) and the resulting signal is shown in Fig. 2.7(b). We can notice that the high-frequency content is removed with the ideal low-pass filter. Using the IGFT to go back to the vertex domain, the graph signal in Fig. 2.7(b) is less noisy and more smooth.

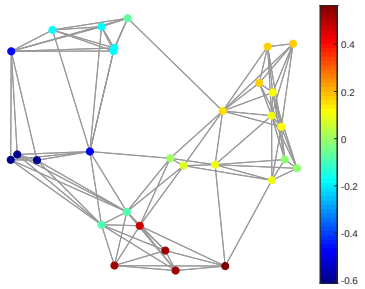
2.4.2. DESIGN OF GRAPH FILTERS

In this thesis, we mainly contribute to the graph filter design area. Thus, we now briefly introduce and discuss the filter design philosophy. Details can be found in the next chapters.

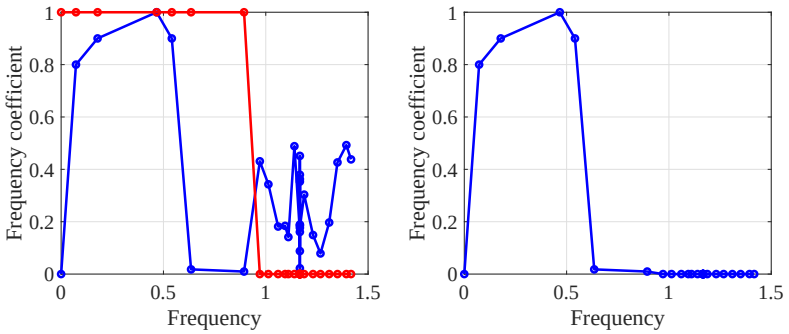
Throughout this thesis, we will consider different parametrizations of the graph filter function $g(\cdot)$, and thus we will often explicitly write this function as



(a) Noisy graph signal on the graph with $N = 30$.



(b) Denoised graph signal on the same graph.



(c) Corresponding GFTs and the low-pass filtering operation in the graph Fourier domain. Red points represent the low-pass filter response. Blue points are the frequency response of the graph signals.

Figure 2.7: The graph ($N = 30$) and graph signal to illustrate the effect of a low-pass graph filter. (a) and (b) represent, respectively, the noisy graph signals and the denoised signal on the same graph. (c) Their corresponding frequency coefficients with the graph shift operator $\boldsymbol{S} = \boldsymbol{L}_n$. We also show here the filtering operation in the graph Fourier domain.

$g(\cdot;\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is a vector that contains the graph filter parameters, i.e., filter coefficients, zeros and poles, or any other set of filter parameters. Correspondingly, we can also write $\hat{g}_n$ explicitly as $\hat{g}_n(\boldsymbol{\theta})$.

Assuming now that the desired frequency response at frequency λ_n is given by $\hat{h}_n$, the filter parameters $\boldsymbol{\theta}$ can be found by solving

$$\min_{\boldsymbol{\theta}} \sum_{n=1}^{N} |\hat{h}_n - \hat{g}_n(\boldsymbol{\theta})|^2. \tag{2.26}$$

We restrict the desired frequency response $\hat{h}$ to $\hat{h}_n = \hat{h}_{n'}$, if the corresponding eigenvalues (frequencies) satisfy $\lambda_n = \lambda_{n'}$. The desired response $\hat{h}_n$ can originate from different scenarios according to different demands:

- First of all, when we focus on graph filter design, i.e., when we want to design a low pass filter to smoothen or denoise a graph signal, the desired frequency response $\hat{h}_n$ basically indicates how much we want to attenuate a specific graph mode and thus it will generally be real-valued and symmetric w.r.t. the real axis (for both undirected and directed graphs).
- Also, when we want to do graph signal prediction, as done in [6], we want to design an all-pass filter and set $\hat{h}_n$ to be one (and thus real-valued) everywhere. In this case, the cost function (2.26) will also be weighted, but the filter design methods can easily be adapted to this desired weighting function [36].
- For some GSP applications, such as compression, the desired frequency response $\hat{h}_n$ will directly be the GFT of the signal, for which Property 1 holds.

In any case, whatever the scenario (filter design, prediction, smoothing, denoising, or compression) or type of graph (undirected or directed), the following property holds.

Property 2. *As mentioned above, $\hat{h}_n$ is selected either as real-valued and symmetric w.r.t. the real frequency axis, or as the GFT of a signal. The latter means that $\hat{h}_n$ is real-valued if λ_n is real-valued while $\hat{h}_n$ and $\hat{h}_{n'}$ form a complex conjugate pair if λ_n and $\lambda_{n'}$ form a complex conjugate pair (this is due to Property 1). Put differently, either way we select $\hat{h}_n$, if λ_n is real-valued, then $\hat{h}_n$ is real-valued whereas if λ_n and $\lambda_{n'}$ form a complex conjugate pair, then $\hat{h}_n$ and $\hat{h}_{n'}$ form a complex conjugate pair as well.*

2.5. CONCLUSION

In this chapter, we first briefly introduced some important types of graphs and GSP approaches. Then, we formulated the graph model as a mathematical representation to express the data (graph signal). After discussing the fundamental concepts of GSP, we introduced the spectral analysis of a graph signal where the GFT is utilized as a useful tool for different graph operators, e.g., the Laplacian

matrix for undirected graphs, and the adjacency matrix for directed graphs. This will be the basic approach throughout the following chapters. Finally, we showed the definition and properties of a graph filter and shortly discussed the filter design. This allows us to give more insights into the design methods in the next chapter.

REFERENCES

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

[2] F. R. Chung and F. C. Graham, *Spectral graph theory*, 92 (American Mathematical Soc., 1997).

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

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

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

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

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

[8] A. Sandryhaila and J. M. Moura, *Discrete signal processing on graphs: Graph filters.* in *ICASSP* (2013) pp. 6163–6166.

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

[10] F. Chung, *Laplacians and the cheeger inequality for directed graphs,* Annals of Combinatorics **9**, 1 (2005).

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

[12] F. Chung, *The diameter and laplacian eigenvalues of directed graphs,* the electronic journal of combinatorics **13**, 4 (2006).

[13] B. Girault, P. Gonçalves, and É. Fleury, *Translation on graphs: An isometric shift operator,* IEEE Signal Processing Letters **22**, 2416 (2015).

[14] A. Gavili and X.-P. Zhang, *On the shift operator, graph frequency, and optimal filtering in graph signal processing,* IEEE Transactions on Signal Processing **65**, 6303 (2017).

[15] 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).

[16] N. Tremblay, P. Gonçalves, and P. Borgnat, *Design of graph filters and filterbanks,* in *Cooperative and Graph Signal Processing* (Elsevier, 2018) pp. 299–324.

[17] R. A. Horn and C. R. Johnson, *Matrix analysis* (Cambridge university press, 1990).

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

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

[20] L. Le Magoarou, R. Gribonval, and N. Tremblay, *Approximate fast graph fourier transforms via multilayer sparse approximations,* IEEE transactions on Signal and Information Processing over Networks **4**, 407 (2018).

[21] L. Le Magoarou, N. Tremblay, and R. Gribonval, *Analyzing the approximation error of the fast graph fourier transform,* in *2017 51st Asilomar Conference on Signals, Systems, and Computers* (IEEE, 2017) pp. 45–49.

[22] C. H. Edwards, D. E. Penney, and D. T. Calvis, *Differential equations and boundary value problems* (Tsinghua University Press, 2004).

[23] V. Sinswat and F. Fallside, *Eigenvalue/eigenvector assignment by state-feedback,* International Journal of Control **26**, 389 (1977).

[24] S. Chen, A. Sandryhaila, J. M. Moura, and J. Kovacevic, *Signal denoising on graphs via graph filtering,* in *Signal and Information Processing (GlobalSIP), 2014 IEEE Global Conference on* (IEEE, 2014) pp. 872–876.

[25] S. Deutsch, A. Ortega, and G. Medioni, *Manifold denoising based on spectral graph wavelets,* in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on* (IEEE, 2016) pp. 4673–4677.

[26] M. Onuki, S. Ono, M. Yamagishi, and Y. Tanaka, *Graph signal denoising via trilateral filter on graph spectral domain,* IEEE Transactions on Signal and Information Processing over Networks **2**, 137 (2016).

[27] F. Zhang and E. R. Hancock, *Graph spectral image smoothing using the heat kernel,* Pattern Recognition **41**, 3328 (2008).

[28] A. Sandryhaila and J. M. Moura, *Classification via regularization on graphs.* in *GlobalSIP* (2013) pp. 495–498.

[29] A. Anis, A. Gadde, and A. Ortega, *Efficient sampling set selection for bandlimited graph signals using graph spectral proxies,* IEEE Transactions on Signal Processing **64**, 3775 (2016).

[30] S. Chen, A. Sandryhaila, J. M. Moura, and J. Kovačević, *Signal recovery on graphs: Variation minimization,* IEEE Transactions on Signal Processing **63**, 4609 (2015).

[31] N. Tremblay, G. Puy, R. Gribonval, and P. Vandergheynst, *Compressive spectral clustering,* in *Machine Learning, Proceedings of the Thirty-third International Conference (ICML 2016), June* (2016) pp. 20–22.

[32] D. B. Tay and Z. Lin, *Design of near orthogonal graph filter banks,* IEEE Signal Processing Letters **22**, 701 (2015).

[33] S. K. Narang and A. Ortega, *Compact support biorthogonal wavelet filterbanks for arbitrary undirected graphs,* IEEE transactions on signal processing **61**, 4673 (2013).

[34] A. Sakiyama, K. Watanabe, and Y. Tanaka, *Spectral graph wavelets and filter banks with low approximation error,* IEEE Transactions on Signal and Information Processing over Networks **2**, 230 (2016).

[35] D. I. Shuman, C. Wiesmeyr, N. Holighaus, and P. Vandergheynst, *Spectrum-adapted tight graph wavelet and vertex-frequency frames,* IEEE Transactions on Signal Processing **63**, 4223 (2015).

[36] 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).

3

GRAPH FILTERS

THIS chapter introduces some particular types of graph filters, e.g., *finite impulse response* (FIR) and *infinite impulse response* (IIR) graph filters. These different classes of graph filters (GFs) are utilized as basic tools for graph signal processing and related applications. Similar to temporal filters, these classes of GFs parameterize the graph signal spectra in different ways. In this chapter, we will show efficient implementations of these filter classes and well-studied design methods for GFs. These two aspects can be provided, respectively, in the vertex and frequency domains. Next to the standard form of these graph filters, we will also summarize some alternative forms, such as the distributed Chebyshev graph filter [2].

This chapter is organized as follows. Section 3.1 briefly introduces the design of GFs and covers some GF applications. Then, Section 3.2 considers the *universal design* concept which can avoid the computational cost of an eigenvalue decomposition during the filter design. Sections 3.3 and 3.4 then discuss in detail the FIR and IIR classes of graph filters. In the end, Section 3.5 concludes the chapter.

Part of this chapter has been published in the IEEE Transactions on Signal and Information Processing over Network [1] (2019).

3.1. INTRODUCTION

As we mentioned in Chapter 2, a *graph filter* (GF) can be represented as $\boldsymbol{G} = g(\boldsymbol{S})$ in the vertex domain, while the corresponding form is $[g(\boldsymbol{\Lambda})]_{n,n} = \hat{g}_n$ in the frequency domain. The design of a graph filter is often completed in the frequency domain, while the processing of graph signals is generally done in the vertex domain. Furthermore, the *universal design* approach can reduce the computational cost by avoiding an eigenvalue decomposition. Also, considering the local structure of the shift operator, the GF implementation can be carried out in the vertex domain without requiring any eigenvalue decomposition.

In this chapter, we will first introduce the universal filter design concept, and then use the basic classes of GFs to illustrate GF design methods and implementations. But before that, we highlight some applications of graph filters to demonstrate that GFs are useful tools.

- *i) Data Classification.* Data classification is an important task in traditional signal processing. This problem has also been studied in machine learning [3]. Using a graph representation of data, some novel approaches to this problem are proposed [4–6]. One of them utilizes the graph filter as a classification tool by designing the GF as an adaptive classifier [7]. This approach is based on some known labels and adaptively constructs the filter using those labels. The whole process trains the classifier and modifies the filter coefficients. Then, the constructed graph filter is utilized to classify all nodes by analyzing the filter output. This GF-based classifier approach can also perform well in the multiple classes problem.

- *ii) Image Processing.* Recently, some graph signal processing tools are utilized for classical image processing tasks [8–13]. As an example, in [14], the image interpolation problem can be formulated as a low-pass graph filtering problem to remove high-frequency noise. The numerical performance shows that the approach is not limited to 2D image interpolation and can be useful for improving other kinds of images [14].

- *iii) Filterbanks and Wavelets.* Filterbanks and wavelets are very important techniques for signal processing. There are several recent works to extend wavelets and filter banks to the GSP field [15–19]. Most of them are related to the two-channel critically sampled perfect reconstruction filter bank which is based on the design of a graph filter. Also, in [17, 20], the design

methods are nearly orthogonal and have the advantage of energy preservation and symmetry between the low-pass and high-pass response. Several numerical results are shown to illustrate these techniques. As applications, the filterbanks can be used as a useful tool for image-analysis [20] and analyzing/compressing arbitrarily linked irregular graphs [17].

- *iv) Signal Denoising and Recovery.* As a traditional problem in the signal processing area, the denoising and recovery of graph signals extend the problems from signals with a regular structure to a complex, irregular structure. These methods are often based on signal smoothing with filtering [21, 22], multiresolution analysis [23], and so on. For instance, a novel method to recover the true graph signal from a noisy measurement is based on total variation regularization [24]. The approach leads to a closed-form solution that can be represented by a graph filter [24]. With measurements from temperature sensors, the approach works well.

We only shortly discussed some applications and examples to highlight the wide use of graph filters. For details and more information about graph filter applications, we recommend [8, 25, 26] for further reading. The main contributions of this chapter are:

- *i) We extend the concept of universal filter design from an undirected graph to a directed graph.* Since universal filter design is studied for the undirected case $\boldsymbol{S} = \boldsymbol{L}_{\mathrm{n}}$, we present it for the directed case $\boldsymbol{S} = \boldsymbol{A}_{\mathrm{n}}$. Since the frequencies of a directed graph are shown as real-values and complex conjugate pairs, we consider grid points lying in the complex unit disc.
- *ii) We analyze the numerical results for FIR graph filters designed using the universal design concept.* We prove that the FIR coefficients resulting from the universal design are real-valued for both directed and undirected graphs.

The next section will introduce the universal GF design concept that will be used throughout this thesis.

3.2. UNIVERSAL DESIGN

Since estimating the graph frequencies entails some additional complexity, where the computational cost of an eigendecomposition is $O(N^3)$, graph filters are often designed with no explicit knowledge of the graph or the graph frequencies.

The desired frequency response is assumed to be a function over a continuous range of frequencies (the real line for undirected graphs or the complex plane for directed graphs). Solving the filter design problem for such a scenario is referred to as *universal filter design*.

The concept of GF design without explicit knowledge of the graph frequencies is first introduced, in [2, 12], based on a parameterization using Chebyshev polynomials. These approaches complete the design in a specific continuous range but they are limited to undirected graphs. In this thesis, we extend this concept to emphasize universal design for both directed and undirected graphs.

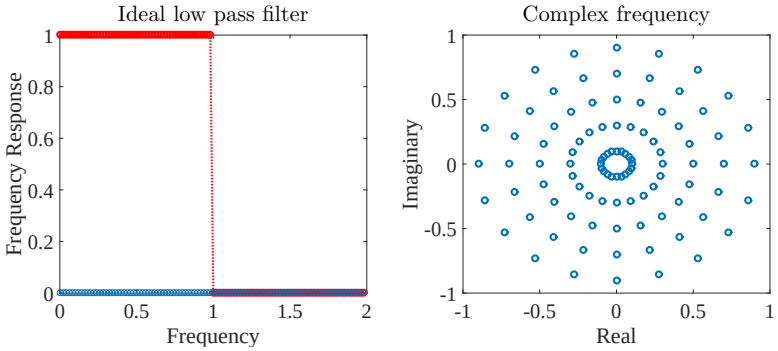
To illustrate the discrete universal design, we first repeat the design problem (2.26) mentioned in the previous chapter:

$$\min_{\boldsymbol{\theta}} \sum_{n=1}^{N} |\hat{h}_n - \hat{g}_n(\boldsymbol{\theta})|^2.$$

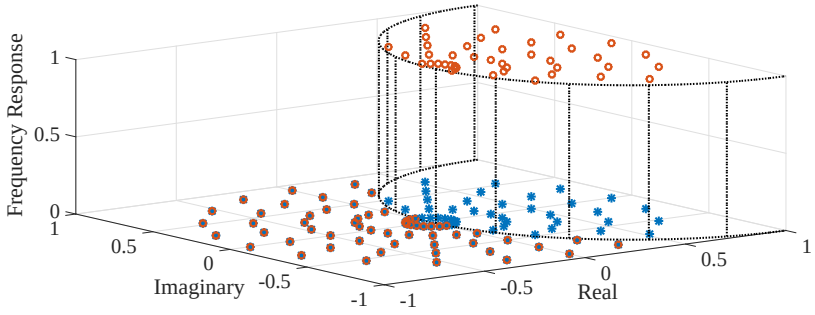
Following (2.26), the universal filter design problem can be tackled by discretizing the related continuous frequency range into a finite set of N potential graph frequencies. Then the design problem can be solved for this finite set of potential graph frequencies instead of for the true graph frequencies. In that case, the desired filter response $\hat{h}$ is a function of some potential graph frequencies that are spread out over the whole frequency range, and the set of filter coefficients $\boldsymbol{\theta}$ is useful for all graphs. This is referred to as the discrete universal design.

Now, we will introduce some examples of a discrete universal grid for both directed and undirected graphs as required background for the GF design in this thesis.

- *Undirected Case.* For undirected graphs, we take $\boldsymbol{S} = \boldsymbol{L}_{\text{n}}$ as the graph shift operator as we mentioned in the previous chapter. Thus, we can consider for instance N different grid points in the interval $[0, 2]$. Note that depending on the graph, we obtain a different eigenvalue spread, e.g., the eigenvalues of an Erdős Rényi graph [27] are in general closely spread around 0 and more widely spread around p, the link probability of the graph. However, since we want to be independent of the graph topology, we consider a uniformly-spaced grid in our design. As an example, we show the graph spectrum for an ideal low pass graph filter with cutoff frequency $\lambda_c = 1$ in Fig. 3.1(a) left. With another shift operator, e.g., $\boldsymbol{S} = \boldsymbol{L}$, the discrete universal design aims at gridding the whole interval $[\lambda_{\min}, \lambda_{\max}]$, where estimating $\lambda_{\min}$ and $\lambda_{\max}$ costs much less than an eigenvalue decomposition.



(a) Low pass filter for undirected graph and universal grid for directed graph based on the normalized adjacency matrix.



(b) Ideal low pass filter response of universal design for directed graph based on the normalized adjacency matrix.

Figure 3.1: (a) (left) Ideal low pass filter response of universal design for undirected graph ($N = 100$). (a) (right) Universal grid for directed graph based on the normalized adjacency matrix $\boldsymbol{A}_{\mathrm{n}}$ ($N = 100$). (b) Ideal low pass filter response of universal design for a directed graph with $N = 100$. The complex frequencies lying inside the circle with radius 1 centered at $(1,0)$ are "small" frequencies.

- *Directed Case.* Alternatively, for directed graphs with $\boldsymbol{S} = \boldsymbol{A}_{\mathrm{n}}$, the graph frequencies lie in the complex unit disc. Again trying to avoid any dependence on the graph, we suggest gridding this disc by N complex conjugate pairs of points, as shown in Fig. 3.1(a) right. Fig. 3.1(b) again shows an example of an ideal low pass filter in this context. The cutoff frequency λ_c for the corresponding ideal low pass filter is here defined as the distance from the point $(1,0)$ in the complex plane, and it is set as $\lambda_c = 1$ in Fig. 3.1(b). All graph frequencies with a distance to $(1,0)$ that is smaller than λ_c will

be part of the passband since they yield the "smaller" frequencies. For another shift operator of a directed graph, e.g., $\boldsymbol{S} = \boldsymbol{A}$, the complex disc will change to a radius $|\lambda_{\max}|$.

The universal design can be beneficial in a situation where the structure of the graph is unknown. Then, the filter coefficients are independent of the specific graph. Note that the graph filter design based on Chebyshev polynomials [2, 12] is an alternative approach to the discrete universal design.

Also, using random matrix theory [28–30], the information and distribution of the eigenvalues (frequencies) of a large matrix can be obtained through their asymptotic behavior [31]. Thus, besides the universal design we discussed, approximating the graph empirical spectral statistics is another way for designing the GF without explicit knowledge of the frequencies.

In the upcoming sections, we will focus on some specific graph filter classes and discuss the technical details.

3.3. FINITE IMPULSE RESPONSE GRAPH FILTER

From [32], an FIR graph filter $\boldsymbol{G}$ of order K can be expressed as a K-th order polynomial in the graph shift operator

$$\boldsymbol{G} = g(\boldsymbol{S};\boldsymbol{\theta}) = \sum_{k=0}^{K} g_k \boldsymbol{S}^k, \tag{3.1}$$

with $\boldsymbol{\theta} = [g_0, \ldots, g_K]^T$ collecting the FIR filter coefficients. If we use the vector $\boldsymbol{x}$ as the input of graph filter $\boldsymbol{G}$, then the output $\boldsymbol{y}$ can be formulated as

$$\boldsymbol{y} = \boldsymbol{G}\boldsymbol{x} = \sum_{k=0}^{K} g_k \boldsymbol{S}^k \boldsymbol{x}. \tag{3.2}$$

3.3.1. IMPLEMENTATION AND COST

In order to illustrate the implementation process of (3.2) with the shift operator $\boldsymbol{S}$ in the vertex domain, we first expand (3.2) as

$$\boldsymbol{y} = \boldsymbol{G}\boldsymbol{x} = g_0 \boldsymbol{S}^0 \boldsymbol{x} + g_1 \boldsymbol{S}^1 \boldsymbol{x} + \cdots + g_K \boldsymbol{S}^K \boldsymbol{x}. \tag{3.3}$$

As we mentioned in the previous chapter, the output signal y_i of every node v_i can be computed locally by exchanging previously shifted versions of the input

signal with its direct neighbors. Thus, the multiplications with the shift operator $\boldsymbol{S}$ can be computed as

$$\boldsymbol{S}^k\boldsymbol{x} = \boldsymbol{S}(\boldsymbol{S}^{k-1}\boldsymbol{x})$$

leading to an overall complexity of $O(KE)$ for the whole FIR filtering process, where E is the number of edges.

Since the graph shift operator is usually a sparse matrix, the implementation cost significantly reduces compared to the regular cost $O(N^2)$ of a matrix-vector multiplication since $E << N$, where N is the number of nodes. The distributed computing over the network also suggests that the FIR graph filter is a localized linear operation in the graph vertex domain.

3.3.2. FILTER DESIGN

We now focus on finding the filter coefficients g_k which are useful for approximating a desired (user-provided) frequency response.

We use $\boldsymbol{H}$ to represent the desired filtering operation in the vertex domain and $\hat{\boldsymbol{h}}$ as the desired frequency response in the frequency domain. Then, we can formulate the design problem as

$$\min_{g_0,\dots,g_K} \left\| \boldsymbol{H} - \sum_{k=0}^{K} g_k \boldsymbol{S}^k \right\|_2^2. \tag{3.4}$$

Since this is a linear least squares (LLS) problem, we can solve it efficiently with off-the-shelf algorithms [33]. In this thesis, we mainly focus on the design in the frequency domain and discuss the universal design for this case.

We first move the problem from the vertex domain to the graph frequency domain. Since the GFT and IGFT are linear processes, for problem (3.2), we then have

$$\boldsymbol{y} = \left(\sum_{k=0}^{K} g_k \boldsymbol{U}\boldsymbol{\Lambda}^k\boldsymbol{U}^{-1} \right)\boldsymbol{x} = \boldsymbol{U}\left(\sum_{k=0}^{K} g_k \boldsymbol{\Lambda}^k \right)\boldsymbol{U}^{-1}\boldsymbol{x}, \tag{3.5}$$

and the output of the FIR filter in the graph frequency domain is

$$\hat{\boldsymbol{y}} = \sum_{k=0}^{K} g_k \boldsymbol{\Lambda}^k \hat{\boldsymbol{x}}. \tag{3.6}$$

Under this circumstance, the filter frequency response has the polynomial form

$$g(\boldsymbol{\Lambda}) = \sum_{k=0}^{K} g_k \boldsymbol{\Lambda}^k. \tag{3.7}$$

Also, the element $\hat{g}_n$ of $g(\boldsymbol{\Lambda})$ at a specific λ_n in the spectral domain can be expressed as

$$\hat{g}_n = \sum_{k=0}^{K} g_k \lambda_n^k, \tag{3.8}$$

where λ_n can be the true graph frequencies or the potential graph frequencies represented by grid points for the universal design. When we take λ_n as grid points in the frequency range $[\lambda_{\min}, \lambda_{\max}]$, the number N is not necessarily the same as the number of nodes.

The filter coefficients in time and frequency are thus related as

$$\begin{pmatrix} \hat{g}_1 \\ \hat{g}_2 \\ \vdots \\ \hat{g}_N \end{pmatrix} = \begin{pmatrix} 1 & \lambda_1 & \cdots & \lambda_1^K \\ 1 & \lambda_2 & \cdots & \lambda_2^K \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \lambda_N & \cdots & \lambda_N^K \end{pmatrix} \begin{pmatrix} g_0 \\ g_1 \\ \vdots \\ g_K \end{pmatrix}. \tag{3.9}$$

By stacking the filter frequency response in $\hat{\boldsymbol{g}} = [\hat{g}_1, \cdots, \hat{g}_N]^{\mathrm{T}}$, we obtain the relation

$$\hat{\boldsymbol{g}} = \boldsymbol{\Psi}_{K+1} \boldsymbol{g}, \tag{3.10}$$

where $\boldsymbol{\Psi}_{K+1}$ is the $N \times (K+1)$ Vandermonde matrix representing the system with entries $[\boldsymbol{\Psi}_{K+1}]_{n,k} = \lambda_n^{k-1}$.

Assuming the desired frequency response is given by the vector $\hat{\boldsymbol{h}} = [\hat{h}_1, \cdots, \hat{h}_N]^{\mathrm{T}}$, design (2.26) can now be rewritten as the following LLS problem

$$\min_{\boldsymbol{g}} \left\| \hat{\boldsymbol{h}} - \boldsymbol{\Psi}_{K+1} \boldsymbol{g} \right\|^2. \tag{3.11}$$

The solution to this LLS problem is given by

$$\boldsymbol{g} = \boldsymbol{\Psi}_{K+1}^{\dagger} \hat{\boldsymbol{h}}, \tag{3.12}$$

where $\boldsymbol{\Psi}_{K+1}^{\dagger}$ is the pseudo-inverse of $\boldsymbol{\Psi}_{K+1}$.

For (3.8) to make sense as a graph filter that will be applied to a real-valued graph signal $\boldsymbol{x}$, we want the FIR filter coefficients $\boldsymbol{g}$ to be real-valued. The next proposition shows that this is the case.

Proposition 1. *Under Property 2 [cf. Chapter 2], the FIR filter coefficients $\boldsymbol{g}$ obtained by solving (3.11) are real-valued.*

Proof. The proof can be found in Appendix A. □

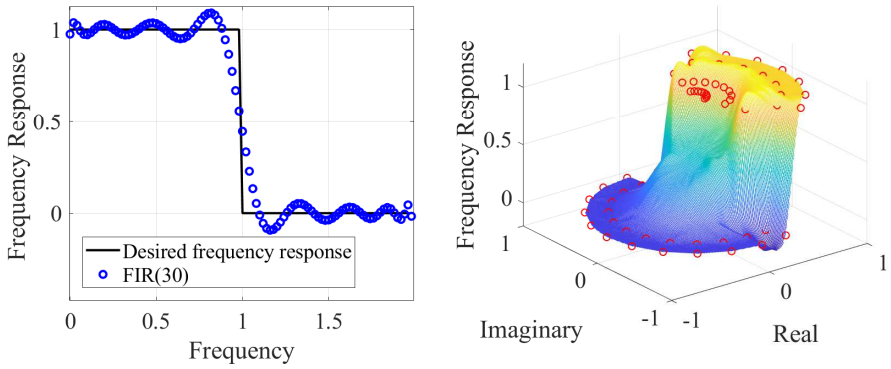


Figure 3.2: Universal design fashion $N = 100$ with ideal low pass FIR graph filter for both directed and undirected graphs. For the undirected graph, we take filter order $K = 30$, while for the directed case, the filter order is utilized as $K = 60$.

For example, in Fig. 3.2, we use $N = 100$ grid points (discrete universal design) to evaluate the design solutions of FIR graph filters for both directed and undirected graphs. We observe that, with the LLS discrete universal design, the approximation accuracy is worse on the points around the cut-off frequency for both cases.

We summarize the following statements for the FIR graph filter and the corresponding universal design:

- For the FIR graph filter, as shown in [34], [35], $\boldsymbol{\Psi}_{K+1}$ needs to be well-conditioned for this approach to work well. This will only be the case for small graph sizes N and/or small filter orders K, i.e., the FIR graph filter order K is much smaller than the number of frequencies N.

- Note that to improve the conditioning of the matrix $\boldsymbol{\Psi}_{K+1}$, close/equal eigenvalues, e.g., $\lambda_i = \lambda_j$, could be grouped together under the assumption that the desired filter response on those eigenvalues is equal, e.g., $\hat{h}_i = \hat{h}_j$. In any case, the FIR filter order K needs to be small and because of the nature of the polynomial fitting problem, this will lead to the limited accuracy of the FIR filter.

- Using the graph frequencies to design the filter, we can minimize the approximation error (3.11) for the specific graph. However, the computational cost of an eigenvalue decomposition is $O(N^3)$ which is quite high

for a large N, e.g., $N > 1000$. Using grid points, the universal design can avoid this high cost. However, since the coefficients are calculated for the grid points, the approximation error for the specific set of frequencies may be higher than we expect. Thus, the universal design makes a trade-off between the decomposition cost and the approximation accuracy.

- Normally, in the LLS universal design, the number of grid points N can influence the approximation performance of the design. For a dense grid with points close to each other, the LLS may be complex and ill-conditioned. Meanwhile, with a sparse grid with a small N, the resulting filter coefficients may cause a bad approximation accuracy on some frequencies between two grid points.

We have considered the implementation and design of the well-studied FIR graph filter. In the next section, we will introduce some modifications to the FIR graph filter and their implementations.

3.3.3. RELATED FIR GRAPH FILTERS

In this section, we discuss three kinds of related FIR graph filters:

- *i) Chebyshev FIR Graph Filter* where the polynomials in (3.1) can be replaced by Chebyshev polynomials of the second kind ;
- *ii) Node-variant FIR Graph Filter* which allows for the simultaneous implementation of multiple graph filters in different nodes of the graph;
- *iii) Edge-variant FIR Graph Filter* where every node weights the signals from its neighbors with different values.

As we will see next, all these graph filters improve the FIR graph filter from different perspectives.

Chebyshev FIR Graph Filter.

In [2], the authors provide some distributed signal processing applications of graph multiplier operators and introduce an efficient approximation method via shifted Chebyshev polynomials which can be viewed as FIR-Chebyshev graph filters [12],[2]. Note that this method is limited to undirected graphs with real-valued graph frequencies.

Using Chebyshev polynomials to approximate the frequency response, the main idea is truncating a shifted Chebyshev series expansion of the frequency response on the interval $[0, \lambda_{\max}]$. We first consider the specific frequency range $x \in [-1, 1]$, where the initial Chebyshev polynomials are formulated as

$$T_0(x) = 1 \ , \ T_1(x) = x. \tag{3.13}$$

For $k \geq 2$, the Chebyshev polynomials enjoy the generating procedure

$$T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x). \tag{3.14}$$

These Chebyshev polynomials form an orthogonal basis considering an appropriate weighting function $w(x) = (1-x^2)^{-1/2}$. By shifting the frequency range from $x \in [-1, 1]$ to $\lambda \in [0, \lambda_{max}]$, the Chebyshev graph filter design only requires the maximum frequency and not the full eigenvalue decomposition of the graph shift operator $\boldsymbol{S}$.

In the frequency domain, the filter frequency response can be expressed as

$$h(x) = \sum_{k=0}^{\infty} c_k T_k(x), \tag{3.15}$$

where c_k represents the coefficients of the Chebyshev polynomials and $h(x)$ is the desired frequency response. For the frequency range $x \in [-1, 1]$, the coefficients of the Chebyshev graph filter can easily be computed in closed form [36]. Then, we have

$$h(x) = \frac{1}{2}c_0 + \sum_{k=1}^{\infty} c_k T_k(x) \ , \ x \in [-1, 1]. \tag{3.16}$$

By shifting the domain of the Chebyshev polynomials to the range $[0, \lambda_{\max}]$ via $\lambda = \lambda_{\max}/2(x+1)$, we then have

$$h(\lambda) = \frac{1}{2}c_0 + \sum_{k=1}^{\infty} c_k \overline{T}_k(\lambda) \ , \ \lambda \in [0, \lambda_{\max}], \tag{3.17}$$

where, for $k \geq 2$, the shifted Chebyshev polynomials $\overline{T}_k(\lambda) = T_k(\frac{x-\alpha}{\alpha})$ satisfy

$$\overline{T}_k(\lambda) = \frac{1}{2}(\lambda - \alpha)\overline{T}_{k-1}(\lambda) - \overline{T}_{k-2}(\lambda). \tag{3.18}$$

with $\alpha = \lambda_{\max}/2$. In the vertex domain, the corresponding polynomial with the shift operator $\boldsymbol{S}$ is given by

$$\overline{T}_k(\boldsymbol{S}) = \frac{1}{2}(\boldsymbol{S} - \alpha \boldsymbol{I})\overline{T}_{k-1}(\boldsymbol{S}) - \overline{T}_{k-2}(\boldsymbol{S}). \tag{3.19}$$

Then, the FIR-Chebyshev graph filter up to order K can be expressed as

$$\boldsymbol{G} = \frac{1}{2} c_0 \boldsymbol{I} + \sum_{k=1}^{K} c_k \overline{T}_k(\boldsymbol{S}), \tag{3.20}$$

and the output of the filter in the vertex domain is

$$\boldsymbol{y} = \boldsymbol{G}\boldsymbol{x} = \frac{1}{2} c_0 \boldsymbol{I}\boldsymbol{x} + \sum_{k=1}^{K} c_k \overline{T}_k(\boldsymbol{S})\boldsymbol{x}. \tag{3.21}$$

Consider now the implementation of the FIR-Chebyshev graph filter. For any input vector $\boldsymbol{x}$, we have

$$\overline{T}_k(\boldsymbol{S})\boldsymbol{x} = \frac{1}{2}(\boldsymbol{S} - \alpha \boldsymbol{I})(\overline{T}_{k-1}(\boldsymbol{S})\boldsymbol{x}) - (\overline{T}_{k-2}(\boldsymbol{S})\boldsymbol{x}). \tag{3.22}$$

From (3.22), the benefit from a computational perspective of the Chebyshev polynomial is that the term $\overline{T}_k(\boldsymbol{S})\boldsymbol{x}$ can be computed recursively from $\overline{T}_{k-1}(\boldsymbol{S})\boldsymbol{x}$ and $\overline{T}_{k-2}(\boldsymbol{S})\boldsymbol{x}$. Then, the total computational cost of implementing the filter $\boldsymbol{G}$ is $O(EK)$, where E is the number of edges. Since the coefficients of the FIR - Chebyshev graph filter are computed independently without the knowledge of the graph frequencies, the design can also be called a universal design since an eigenvalue decomposition is avoided.

For the Chebyshev polynomial definition and its benefits, see [36], and for technical design approaches of the FIR-Chebyshev graph filter see [2, 12]. In addition to the well-studied FIR-Chebyshev graph filter, the filter design with Chebyshev polynomials can also be completed by using the estimated distribution of eigenvalues. Details for the design and estimation of the spectral distribution can be found in [31]

Node-variant FIR Graph Filter.

The node-variant FIR graph filter allows for the simultaneous implementation of multiple (regular) GFs in different nodes of the graph [35]. This kind of graph filter extends the FIR (3.1) and can be formulated as

$$\boldsymbol{G}_{\text{nv}} = \sum_{k=0}^{K} \text{diag}(\boldsymbol{c}_k)\boldsymbol{S}^k. \tag{3.23}$$

The $N \times 1$ vector $\boldsymbol{c}_k$ are constant and contain the filter coefficients that are applied at each node and its neighbors at the kth shift. For the case $\boldsymbol{c}_k = c_k \mathbf{1}_N$,

the node-variant FIR graph filter reduces to the FIR (3.1). Generally, when $\boldsymbol{G}_{\text{nv}}$ is applied to a graph signal, every node applies different weights to the shifted signals [35].

In [35], the authors mention that an alternative definition for the node-variant FIR graph filter is given by

$$\boldsymbol{G}'_{\text{nv}} = \sum_{k=0}^{K} \boldsymbol{S}^k \text{diag}(\boldsymbol{c}_k). \tag{3.24}$$

For every term in (3.24), the filter first modulates the input signal $\boldsymbol{x}$ with the vectors $\boldsymbol{c}_k$ and then applies the graph shift operator $\boldsymbol{S}^k$. The output of the filter in the vertex domain is

$$\boldsymbol{y} = \boldsymbol{G}_{\text{nv}}\boldsymbol{x} \text{ or } \boldsymbol{y} = \boldsymbol{G}'_{\text{nv}}\boldsymbol{x}. \tag{3.25}$$

For more details related to implementations, frequency interpretations, optimal designs, as well as other technical applications of node-variant FIRs such as (3.23) and (3.24), please see [35].

The most important benefit of the node-variant FIR graph filters, related to (3.23) and (3.24), is the additional flexibility generated by a large number of filter coefficients. Without undermining the local implementation, the node-variant FIR can be suitable for more general operators on graphs in the filter design phase.

We now briefly mention one of the node-variant FIR applications. Since the output of a GF can be seen as the outcome of a diffusion process, the corresponding filter coefficients are viewed as the rate of diffusion. With a connectivity graph, the node-variant FIR can apply to network processes to approximate the dynamics. The initial network state is the input of the GF, and the output is the final network state. With the shift operator, the objective can be to design the filter coefficients to approximate the desired linear transformation.

Edge-variant FIR Graph Filter.

Although the filter $\boldsymbol{G}_{\text{nv}}$ applies different weighting coefficients for each node after carrying out the filter shift, the weights to the neighboring signals for each filter shift are the same. However, they can also be different leading to an edge-variant FIR graph filter [37]. This improvement of the degrees of freedom allows for a potential reduction of the filter order, and the corresponding implementation cost.

The most general edge-variant FIR graph filter is defined in the vertex domain as

$$\begin{aligned} \boldsymbol{G}_{\text{ev}} &= \boldsymbol{\Phi}_0 + \boldsymbol{\Phi}_1 \circ \boldsymbol{S} + (\boldsymbol{\Phi}_2 \circ \boldsymbol{S})(\boldsymbol{\Phi}_1 \circ \boldsymbol{S}) + \cdots \\ &\quad + (\boldsymbol{\Phi}_K \circ \boldsymbol{S})(\boldsymbol{\Phi}_{K-1} \circ \boldsymbol{S}) \cdots (\boldsymbol{\Phi}_1 \circ \boldsymbol{S}) \\ &= \sum_{k=1}^{K} \prod_{j=1}^{K} (\boldsymbol{\Phi}_j \circ \boldsymbol{S}) + \boldsymbol{\Phi}_0 \end{aligned} \tag{3.26}$$

where $\boldsymbol{\Phi}_j \in \mathbb{R}^{N \times N}$ are edge-weighting matrices that apply different weights to the elements of the shift operator $\boldsymbol{S}$ using the Hadamard product $\circ$. The support of $\boldsymbol{\Phi}_j$ is always reduced to the support of $\mathbf{S}$. For this filter design, there is no symmetry assumption on the matrices $\boldsymbol{\Phi}_j$ and $\boldsymbol{\Phi}_0$ is considered to be a diagonal matrix [37].

Similar to the previous section, we use $\boldsymbol{H}$ to represent the desired filtering operation in the vertex domain and the design problem is now to solve the following optimization problem

$$\min_{\{\boldsymbol{\Phi}_j\}} \left\| \boldsymbol{H} - \sum_{k=1}^{K} \prod_{j=1}^{K} (\boldsymbol{\Phi}_j \circ \boldsymbol{S}) + \boldsymbol{\Phi}_0 \right\|_2^2. \tag{3.27}$$

Note that the spectral norm $\|\cdot\|_2$ can also be replaced by other appropriate distance measures, i.e., the Frobenius norm $\|\cdot\|_F$.

We can notice that the problem (3.27) is a high-dimensional non-convex problem which might lead to a suboptimal result. With a two-step approach to design the coefficients, this problem can be addressed. For the technical details of the design process, we recommend [37].

In [37], the authors also introduce an alternative description for a ready-to-distribute constrained edge-variant FIR graph filter as

$$\boldsymbol{G}_{\text{c - ev}} = \sum_{k=1}^{K} (\boldsymbol{\Phi}_k \circ \boldsymbol{S}) \boldsymbol{S}^{k-1} + \boldsymbol{\Phi}_0. \tag{3.28}$$

where, as for the previous case, the support of $\boldsymbol{\Phi}_j$ is reduced to that of $\mathbf{S}$ and $\boldsymbol{\Phi}_0$ is assumed to be a diagonal matrix.

Note that the filter $\boldsymbol{G}_{\text{c - ev}}$ has a distributed implementation with the intermediate result $\boldsymbol{x}^{(k)} = \boldsymbol{S}^{k-1}\boldsymbol{x}^{(k-1)}$, where $\boldsymbol{x}^{(0)} = \boldsymbol{x}$. Then, combining those intermediate results at each node, we have the output

$$\boldsymbol{y} = \sum_{k=0}^{K} \boldsymbol{y}^{(k)}, \tag{3.29}$$

with $\boldsymbol{y}^{(k)} = (\boldsymbol{\Phi}_k \circ \boldsymbol{S})\boldsymbol{x}^{(k)}$ and $\boldsymbol{y}^{(0)} = \boldsymbol{\Phi}_0 \boldsymbol{x}^{(0)}$. For details of the distributed implementation of $\boldsymbol{G}_{\text{c - ev}}$, the reader is referred to the original works [37] and [38].

3.3.4. DISCUSSION

In this section, we carefully introduced the first type of graph filter, the finite impulse response graph filter, and its modifications. We discussed the design and implementation of the FIR graph filter and summarized improved versions of the FIR filter. The architecture of the FIR filter enjoys a distributed implementation in the vertex domain. Also, with the perspectives of node-variant and edge-variant FIRs, more freedom is allowed in the design process.

In the next section, we will introduce an alternative type of graph filter, an infinite impulse response graph filter. To be specific, we will start with the well-known IIR graph filters, and then introduce the ARMA graph filters. We will introduce different filter forms and implementations, and finally dress out the problems discussed in this thesis.

3.4. INFINITE IMPULSE RESPONSE GRAPH FILTER

Although the FIR graph filter has some implementation advantages, the polynomial fitting solution has some inherent inaccuracies. Hence, other parameterizations have been studied. The infinite impulse response (IIR) graph filter is usually characterized by rational frequency response. Compared with the FIR graph filter, IIR filters have the potential to achieve lower approximation errors for the same number of filter coefficients due to the specific parameterization.

In [39], an IIR graph filter is introduced with $\boldsymbol{S} = \boldsymbol{L}$ for undirected graphs as

$$\boldsymbol{G}_{\text{IIR}} = g(\boldsymbol{L})^{-1} f(\boldsymbol{L}), \tag{3.30}$$

where $g(\cdot)$ and $f(\cdot)$ are both polynomials. With input $\boldsymbol{x}$ and output $\boldsymbol{y}$, the IIR graph filter operation in the vertex domain can be written as

$$\boldsymbol{y} = \boldsymbol{G}_{\text{IIR}} \boldsymbol{x} = g(\boldsymbol{L})^{-1} f(\boldsymbol{L}) \boldsymbol{x}. \tag{3.31}$$

Then, in the graph frequency domain, the IIR filter spectral response is a rational function in λ expressed as

$$h(\lambda) = g(\lambda)^{-1} f(\lambda). \tag{3.32}$$

IIR graph filters are linear and commutative [39]. The direct form of an IIR graph filter is shown in Fig. 3.3 which is a combination of two parts, i.e., the FIR model $f(\boldsymbol{L})$ and the inverse FIR model $g(\boldsymbol{L})^{-1}$.

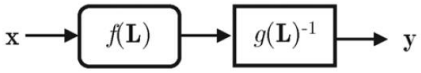


Figure 3.3: The direct form of an IIR graph filter with input $\boldsymbol{x}$ and output $\boldsymbol{y}$.

By factorizing $g(\cdot)$ and $f(\cdot)$ into p and q factors, the rational function of an IIR graph filter can also be represented using alternative structures, i.e., the cascade and parallel forms. The three types of rational functions are theoretically equivalent, but the corresponding results may differ due to numerical errors. For details, we recommend [39].

3.4.1. IMPLEMENTATION OF IIR GRAPH FILTER

Since the direct form of an IIR graph filter is built with an FIR model and an inverse FIR model as $\boldsymbol{G}_{\text{IIR}} = g(\boldsymbol{L})^{-1} f(\boldsymbol{L})$, the implementation of an IIR graph filter considers respectively two steps.

First, for the FIR model with filter order K, the implementation process is the same as in the previous section, and the output on each node is obtained by combining the input data in the neighborhood with different weights. Using the distributed implementation of Section 3.3.1, we can write the FIR module as

$$\boldsymbol{x}_{\text{K}} = f(\boldsymbol{L})\boldsymbol{x}. \tag{3.33}$$

Then, we consider the inverse FIR model as $\boldsymbol{y} = g(\boldsymbol{L})^{-1}\boldsymbol{x}_{\text{K}}$. By defining the matrix $\boldsymbol{B} = g(\boldsymbol{L})$, the output $\boldsymbol{y}$ of the IIR graph filter can be obtained by solving the following optimization problem

$$\min_{\boldsymbol{y}} \left\| \boldsymbol{B}\boldsymbol{y} - \boldsymbol{x}_{\text{K}} \right\|^2. \tag{3.34}$$

In [39], using the gradient descent method, the proposed solution of this problem can be formulated by the iteration

$$\boldsymbol{y}^{(t+1)} = \boldsymbol{y}^{(t)} - \gamma \boldsymbol{D}\boldsymbol{B}(\boldsymbol{B}\boldsymbol{y}^{(t)} - \boldsymbol{x}_{\text{K}}), \tag{3.35}$$

where t is the iteration index, $\boldsymbol{D}$ is a positive-definite matrix and γ is the step size. By setting $\boldsymbol{D} = \boldsymbol{I}$ and $\boldsymbol{D} = \boldsymbol{B}^{-1}$, two different algorithms are introduced, i.e., IDIIR and FastIDIIR [39].

The coefficients of the IIR graph filter can be computed using any rational approximation technique according to the specific form of function $h(\lambda)$. Then,

results from [39] show that the IIR graph filter has a better performance than FIR and FIR-Chebyshev graph filters. More technical details and results can be found in [39].

3.4.2. AUTOREGRESSIVE MOVING AVERAGE GRAPH FILTER

In this section, we briefly introduce the autoregressive moving average (ARMA) graph filter which is a special class of IIR graph filters. We first illustrate the widely used form of the ARMA graph filter proposed in [40]. Then, some modifications are introduced.

An ARMA graph filter is represented with a rational polynomial in the graph frequency λ [40], i.e.,

$$g(\lambda) = \frac{\sum_{q=0}^{Q} b_q \lambda^q}{\sum_{p=0}^{P} a_p \lambda^p}, \tag{3.36}$$

with denominator coefficients a_p and numerator coefficients b_q, leading to the following graph filter in the shift operator $\boldsymbol{S}$:

$$\boldsymbol{G} = \left(\sum_{p=0}^{P} a_p \boldsymbol{S}^p \right)^{-1} \sum_{q=0}^{Q} b_q \boldsymbol{S}^q, \tag{3.37}$$

where $\boldsymbol{a} = [a_0, \ldots, a_P]^T$ and $\boldsymbol{b} = [b_0, \ldots, b_Q]^T$ collect the ARMA filter coefficients. This type of graph filter is labelled as an ARMA graph filter of denominator order P and numerator order Q, expressed as ARMA(P,Q) in short.

We shortly summarize two characteristics of the ARMA graph filter introduced in this thesis:

- Stable ARMA filters are obtained when $\sum_{p=0}^{P} a_p \boldsymbol{S}^p$ is invertible, or equivalently, when $\sum_{p=0}^{P} a_p \lambda_n^p$ is different from zero for all $n = 1, 2, \ldots, N$. This stability condition is less critical as in the time domain, which is mainly because a graph signal is by default finite-length whereas a temporal signal is infinite-length. Hence, there is no big risk of the filter output signal growing unbounded.

- Note that for simplicity reasons we define the ARMA filter coefficients $\boldsymbol{a}$ and $\boldsymbol{b}$ in an ambiguous way since multiplying both $\boldsymbol{a}$ and $\boldsymbol{b}$ with the same

constant will not change the ARMA graph filter. Hence, whenever we design $\boldsymbol{a}$ and $\boldsymbol{b}$ in the following chapters, we will remove this ambiguity by constraining the first AR coefficient to be one, i.e., $a_0 = 1$, which is rather standard.

Pole-zero form. Besides (3.36) , we can also represent ARMA graph filters in other forms. We can for instance consider the pole-zero form, i.e.,

$$g(\lambda) = c\frac{\prod_{q=1}^{Q}(\lambda+\alpha_q)}{\prod_{p=1}^{P}(\lambda+\varphi_p)}, \tag{3.38}$$

where c is a constant number, $-\alpha_q$ are the zeros, and $-\varphi_p$ yields a set of poles of the filter response. In the vertex domain, with the graph shift operator $\boldsymbol{S}$, we obtain

$$\boldsymbol{G} = c\prod_{p=1}^{P}(\boldsymbol{S}+\varphi_p\boldsymbol{I})^{-1}\prod_{q=1}^{Q}(\boldsymbol{S}+\alpha_q\boldsymbol{S}). \tag{3.39}$$

Partial fraction form. Another option is to decompose the rational polynomial using partial fraction expansion which results in

$$g(\lambda) = \sum_{l=0}^{Q-P} r_l\lambda^l + \sum_{p=1}^{P}\frac{\tilde{\omega}_p}{\lambda+\varphi_p}, \tag{3.40}$$

with filter coefficients r_l and $\tilde{\omega}_k$. The related graph filter matrix can then be written as

$$\boldsymbol{G} = \sum_{l=0}^{Q-P} r_l\boldsymbol{S}^l + \sum_{p=1}^{P}\tilde{\omega}_p(\boldsymbol{S}+\varphi_p\boldsymbol{I})^{-1}. \tag{3.41}$$

Note that the pole-zero form and partial fraction expansion can be easily related to each other. These structures also have different implementation forms and graph filter matrix forms in the vertex domain. We will come back to the mentioned forms and give details when we propose the corresponding design methods in later chapters.

3.4.3. IMPLEMENTATION OF ARMA GRAPH FILTER

For the implementation, it is clear that the relation between the output $\boldsymbol{y}$ and the input $\boldsymbol{x}$ of an ARMA graph filter is given by $\boldsymbol{y} = \boldsymbol{G}\boldsymbol{x}$ which can be written as

$$\left(\sum_{p=0}^{P} a_p \boldsymbol{S}^p\right)\boldsymbol{y} = \left(\sum_{q=0}^{Q} b_q \boldsymbol{S}^q\right)\boldsymbol{x}. \tag{3.42}$$

Hence, by defining the matrices

$$\boldsymbol{P} = \sum_{p=0}^{P} a_p \boldsymbol{S}^p, \qquad \boldsymbol{Q} = \sum_{q=0}^{Q} b_q \boldsymbol{S}^q, \tag{3.43}$$

we can express (3.42) in the compact form

$$\boldsymbol{P}\boldsymbol{y} = \boldsymbol{Q}\boldsymbol{x}. \tag{3.44}$$

Following the IIR implementation in Section 3.4.1, the procedure of solving (3.44) can be separated into two steps. To compute the filter output $\boldsymbol{y}$ in (3.44), we first calculate the right-hand side denoted as $\boldsymbol{z} = \boldsymbol{Q}\boldsymbol{x}$ (which corresponds to pre-filtering $\boldsymbol{x}$ with an FIR module), and then $\boldsymbol{y}$ is found by simply solving the linear system

$$\boldsymbol{P}\boldsymbol{y} = \boldsymbol{z}. \tag{3.45}$$

Then, similar to the previous section, the output $\boldsymbol{y}$ can be solved by the optimization problem

$$\min_{\boldsymbol{y}} \left\|\boldsymbol{P}\boldsymbol{y} - \boldsymbol{z}\right\|^2. \tag{3.46}$$

One way to solve this problem is using the iteration (3.35) to formulate the solution of $\boldsymbol{y}$, where $\boldsymbol{x}_{\mathrm{K}} = \boldsymbol{z}$ and $\boldsymbol{B} = \boldsymbol{P}$. Also, there are several other methods to solve the linear system efficiently, like first order methods [33], conjugate gradient (CG) [1], and some distributed iterative methods [41–46]. We will present details of solving the linear system when we introduce several centralized and distributed implementation methods in Chapter 7.

Considering the pole-zero form (3.38) or partial fraction form (3.40) of the ARMA graph filter, we can obtain different implementations of the ARMA filter. We can summarize the differences of these implementations as follow:

- The pole-zero form (3.38) leads to a serial implementation of Q FIR(1) filters and P ARMA(1,0) filters. The partial fraction form (3.40), on the other

hand, leads to a parallel implementation of one FIR($Q-P$) filter and P ARMA(1,0) filters. Note that the design method, related to these ARMA forms, is proposed in chapter 6, and we will formulate the implementation more clearly in Section 6.2.

- In both cases, the ARMA(1,0) graph filters can potentially be implemented in a distributed manner following (3.35). For more details, we refer to [38]. Also, the centralized and distributed implementation methods in Chapter 7 can be utilized as alternative choices to (3.35).

3.4.4. DISCUSSIONS

In this section, we analyzed the second type of graph filter, the infinite impulse response graph filter, and its ARMA form. We have discussed distributed implementations and summarized the different forms of IIR graph filters. In the next section, we will conclude the whole chapter and introduce the main problems that will be discussed in this thesis which are related to the ARMA graph filter design and its corresponding implementations.

3.5. CONCLUSION

In this chapter, we first briefly introduced the universal design for graph filters. Then, we discussed some particular types of graph filters, e.g., finite impulse response (FIR), infinite impulse response (IIR) graph filters, and their modifications. We mainly introduced graph filters from the design and implementation perspectives.

In this thesis, we will focus on the centralized design of graph filters with the ARMA model which are a special class of IIR graph filters. In Chapter 4, we will solve the design problem of the ARMA graph filter of the form (3.36) in a universal way. In Chapter 5, to improve the design in Chapter 4, we introduce an orthogonal polynomial basis into the graph filter design for both directed and undirected graphs. In Chapter 6, we propose a new filter design framework that corresponds to the special cases of the pole-zero form (3.38) and the partial fraction form (3.40) of ARMA filters. In Chapter 7, we demonstrate several centralized and distributed implementation methods for the ARMA graph filter in the vertex domain.

REFERENCES

[1] 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).

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

[3] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification* (John Wiley & Sons, 2012).

[4] A. Blum and S. Chawla, *Learning from labeled and unlabeled data using graph mincuts,* (2001).

[5] X. Zhu, J. Lafferty, and Z. Ghahramani, *Combining active learning and semi-supervised learning using gaussian fields and harmonic functions,* in *ICML 2003 workshop on the continuum from labeled to unlabeled data in machine learning and data mining*, Vol. 3 (2003).

[6] F. Wang and C. Zhang, *Label propagation through linear neighborhoods,* IEEE Transactions on Knowledge and Data Engineering **20**, 55 (2008).

[7] A. Sandryhaila and J. M. Moura, *Discrete signal processing on graphs: Graph filters.* in *ICASSP* (2013) pp. 6163–6166.

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

[9] S. K. Narang, A. Gadde, E. Sanou, and A. Ortega, *Localized iterative methods for interpolation in graph structured data,* in *2013 IEEE Global Conference on Signal and Information Processing* (IEEE, 2013) pp. 491–494.

[10] Y. Wang, A. Ortega, D. Tian, and A. Vetro, *A graph-based joint bilateral approach for depth enhancement,* in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (IEEE, 2014) pp. 885–889.

[11] F. Zhang and E. R. Hancock, *Graph spectral image smoothing using the heat kernel,* Pattern Recognition **41**, 3328 (2008).

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

[13] A. Gadde, S. K. Narang, and A. Ortega, *Bilateral filter: Graph spectral interpretation and extensions,* in *2013 IEEE International Conference on Image Processing* (IEEE, 2013) pp. 1222–1226.

[14] D. Tian, H. Mansour, A. Knyazev, and A. Vetro, *Chebyshev and conjugate gradient filters for graph image denoising,* in *2014 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)* (IEEE, 2014) pp. 1–6.

[15] A. Sakiyama, K. Watanabe, and Y. Tanaka, *Spectral graph wavelets and filter banks with low approximation error,* IEEE Transactions on Signal and Information Processing over Networks **2**, 230 (2016).

[16] G. Shen and A. Ortega, *Optimized distributed 2d transforms for irregularly sampled sensor network grids using wavelet lifting,* in *2008 IEEE International Conference on Acoustics, Speech and Signal Processing* (IEEE, 2008) pp. 2513–2516.

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

[18] Y. Tanaka and A. Sakiyama, *m-channel oversampled graph filter banks,* IEEE Transactions on Signal Processing **62**, 3578 (2014).

[19] D. B. Tay and Z. Lin, *Design of near orthogonal graph filter banks,* IEEE Signal Processing Letters **22**, 701 (2015).

[20] S. K. Narang and A. Ortega, *Compact support biorthogonal wavelet filterbanks for arbitrary undirected graphs,* IEEE transactions on signal processing **61**, 4673 (2013).

[21] S. Mallat, *A wavelet tour of signal processing* (Elsevier, 1999).

[22] S. Chen, R. Varma, A. Singh, and J. Kovačević, *Signal recovery on graphs: Fundamental limits of sampling strategies,* IEEE Transactions on Signal and Information Processing over Networks **2**, 539 (2016).

[23] A. K. Fletcher, V. K. Goyal, and K. Ramchandran, *Iterative projective wavelet methods for denoising,* in *Wavelets: Applications in Signal and Image Processing X*, Vol. 5207 (International Society for Optics and Photonics, 2003) pp. 9–16.

[24] S. Chen, A. Sandryhaila, J. M. Moura, and J. Kovacevic, *Signal denoising on graphs via graph filtering,* in *Signal and Information Processing (GlobalSIP), 2014 IEEE Global Conference on* (IEEE, 2014) pp. 872–876.

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

[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] P. Erdos and A. Rényi, *On the evolution of random graphs,* Publ. Math. Inst. Hung. Acad. Sci **5**, 17 (1960).

[28] E. P. Wigner, *On the distribution of the roots of certain symmetric matrices,* Ann. Math **67**, 325 (1958).

[29] R. Couillet and M. Debbah, *Random matrix methods for wireless communications* (Cambridge University Press, 2011).

[30] A. M. Tulino, S. Verdú, *et al.*, *Foundations and trends® in communications and information theory,* Foundations and Trends® in Communications and Information Theory **1**, 1 (2004).

[31] S. Kruzick and J. M. Moura, *Graph signal processing: Filter design and spectral statistics,* in *2017 IEEE 7th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)* (IEEE, 2017) pp. 1–5.

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

[33] D. P. Bertsekas, *Convex optimization theory* (Athena Scientific Belmont, 2009).

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

[35] S. Segarra, A. G. Marques, and A. Ribeiro, *Optimal graph-filter design and applications to distributed linear network operators,* IEEE Transactions on Signal Processing **65**, 4117 (2017).

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

[37] M. Contino, E. Isufi, and G. Leus, *Distributed edge-variant graph filters,* in *2017 IEEE 7th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)* (IEEE, 2017) pp. 1–5.

[38] E. Isufi, *Graph-time signal processing: Filtering and sampling strategies,* Ph.D Thesis, Delft University of Technology (2019).

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

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

[41] L. F. Richardson, *Ix. the approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam,* Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character **210**, 307 (1911).

[42] G. Opfer and G. Schober, *Richardson's iteration for nonsymmetric matrices,* Linear algebra and its applications **58**, 343 (1984).

[43] Y. Saad, *Iterative methods for sparse linear systems*, Vol. 82 (siam, 2003).

[44] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: numerical methods*, Vol. 23 (Prentice hall Englewood Cliffs, NJ, 1989).

[45] L. Lei, *Convergence of asynchronous iteration with arbitrary splitting form,* Linear Algebra and its Applications **113**, 119 (1989).

[46] J. M. Bull and T. Freeman, *Numerical performance of an asynchronous jacobi iteration,* in *Parallel Processing: CONPAR 92—VAPP V* (Springer, 1992) pp. 361–366.

APPENDIX-A

Since we assume that the shift operator $\mathbf{S}$ is real-valued and diagonalizable, the graph frequencies λ_n (eigenvalues) can be grouped into three sets: $1 \le n \le M$, $M+1 \le n \le 2M$ and $2M+1 \le n \le N$.

The first and second groups are complex conjugate pairs while the last group consists of the real-valued frequencies. Note that this classification only changes the order of the frequencies, and has no influence on the results of the filter coefficients g_k.

Thus, we can split the Vandermonde matrix $\boldsymbol{\Psi}_{K+1}$ and write (3.11) as

$$\begin{aligned}&\min_{\mathbf{g}} \|\hat{\mathbf{h}} - \boldsymbol{\Psi}_{K+1}\mathbf{g}\|^2 \\ &= \min_{\mathbf{g}} \|[\hat{\mathbf{h}}_1^H, \hat{\mathbf{h}}_2^H, \hat{\mathbf{h}}_3^T]^T - [\boldsymbol{\Psi}_1^H, \boldsymbol{\Psi}_2^H, \boldsymbol{\Psi}_3^T]^T\mathbf{g}\|^2\end{aligned}$$

where the three blocks of matrices and vectors belong to the three different groups.

With $1 \le n \le M$, we use the nth and $(M+n)$th frequencies to represent a conjugate pair for the first and second groups of frequencies. Since $\lambda_n = \lambda_{M+n}^*$, the corresponding elements inside the Vandermonde matrix satisfy $[\boldsymbol{\Psi}_1]_{n,k} = [\boldsymbol{\Psi}_2]_{M+n,k}^*$, and thus we have $\boldsymbol{\Psi}_1 = \boldsymbol{\Psi}_2^*$.

According to Property 2, for the frequency pair $\lambda_n = \lambda_{M+n}^*$, the corresponding desired frequency response satisfies $\hat{h}_n = \hat{h}_{M+n}^*$ and thus, we also have $\hat{\mathbf{h}}_1 = \hat{\mathbf{h}}_2^*$. Meanwhile, for $2M+1 \le n \le N$, we have a real-valued $\boldsymbol{\Psi}_3$ and a real-valued $\hat{\mathbf{h}}_3$ since the corresponding frequencies λ_n inside this range are real-valued.

Now, we can rewrite the solution of (3.11) as

$$\begin{aligned}&\hat{\mathbf{g}} = \boldsymbol{\Psi}_{K+1}^{\dagger}\hat{\mathbf{h}} \\ &= (\boldsymbol{\Psi}_1^H\boldsymbol{\Psi}_1 + \boldsymbol{\Psi}_2^H\boldsymbol{\Psi}_2 + \boldsymbol{\Psi}_3^T\boldsymbol{\Psi}_3)^{-1}(\boldsymbol{\Psi}_1^H\hat{\mathbf{h}}_1 \\ &+\boldsymbol{\Psi}_2^H\hat{\mathbf{h}}_2 + \boldsymbol{\Psi}_3^T\hat{\mathbf{h}}_3) \\ &= (\boldsymbol{\Psi}_1^H\boldsymbol{\Psi}_1 + \boldsymbol{\Psi}_1^T\boldsymbol{\Psi}_1^* + \boldsymbol{\Psi}_3^T\boldsymbol{\Psi}_3)^{-1}(\boldsymbol{\Psi}_1^H\hat{\mathbf{h}}_1 \\ &+\boldsymbol{\Psi}_1^T\hat{\mathbf{h}}_1^* + \boldsymbol{\Psi}_3^T\hat{\mathbf{h}}_3).\end{aligned}$$

It is obvious that $\boldsymbol{\Psi}_1^H\boldsymbol{\Psi}_1 + \boldsymbol{\Psi}_1^T\boldsymbol{\Psi}_1^*$ and $\boldsymbol{\Psi}_1^H\hat{\mathbf{h}}_1 + \boldsymbol{\Psi}_1^T\hat{\mathbf{h}}_1^*$ are real-valued. Hence, solving (3.11) leads to a real-valued solution.

II

FILTER DESIGN

4

Filter Design for Autoregressive Moving Average Graph Filters

This chapter proposes two different strategies for designing autoregressive moving average (ARMA) graph filters on both directed and undirected graphs. The first approach is inspired by Prony's method, which considers a modified error between the modeled and the desired frequency response. The second technique is based on an iterative approach, which finds the filter coefficients by iteratively minimizing the true error (instead of the modified error) between the modeled and the desired frequency response. The performance of the proposed algorithms is evaluated and compared with finite impulse response (FIR) and infinite impulse response (IIR) graph filters, on both synthetic and real data. The obtained results show that ARMA filters outperform FIR filters in terms of approximation accuracy and they are suitable for graph signal compression, and prediction.

The remainder of this chapter is organized as follows. Section 4.1 briefly introduces the proposed design methods of *ARMA graph filters* and the main con-

tributions of this chapter. Then, Section 4.2 contains the filter design problem and the proposed design strategies. A few examples and the simulation results to illustrate the proposed framework are shown in Section 4.3. In the end, Section 4.4 concludes the chapter.

Part of this chapter has been published in the IEEE Transactions on Signal and Information Processing over Network [1] (2019).

4.1. INTRODUCTION

In the field of signal processing on graphs, graph filters play a crucial role in processing the spectrum of graph signals. We mainly separate graph filters into two classes, i.e., FIR and IIR graph filters. As we mentioned in the previous chapter, the FIR filter design is already a well-established theory. One of the most popular approaches to fit the graph frequency response of the FIR filter to the desired spectrum is through solving a linear least squares (LLS) fitting problem [2], which can be carried out for undirected as well as directed graphs. An alternative to FIR graph filters are IIR graph filters, such as the autoregressive moving average (ARMA) graph filters [3], or the gradient descent IIR graph filters [4]. These filters are characterized by a rational graph frequency response, which brings more degrees of freedom to the design.

To fully exploit the benefits of the rational frequency response, in this chapter, we focus on a centralized ARMA filter design. In a centralized fashion, we propose new ARMA graph filter design methods, which can be adopted when the graph is known or in a universal fashion by gridding the frequency domain (as done for the LLS FIR filter design with an unknown graph). The proposed ARMA design methods work for undirected as well as directed graphs.

Throughout this chapter, we will mainly use FIR filters and the IIR filters from [4] as benchmarks to assess the performance of the proposed ARMA filters, being their direct competitors, and propose ARMA filters as a useful alternative for potential applications.

The chapter's contribution is threefold:

- *i) We extend the universal design strategy from FIR graph filters to ARMA graph filters for both undirected and directed graphs.* For either the normalized Laplacian (undirected graph) or normalized adjacency (directed graph) matrix, we follow the design concept of Chapter 2 and respectively sample the real interval from zero to two or the complex unit disc. After the grid points have been determined, the main aim of the ARMA graph filter design is to fit the response on these grid points.

- *ii) We propose two ARMA graph filter design strategies, which can be applied to both directed and undirected graphs.* The first one is inspired by Prony's method [5], where a modified error between the modeled and the desired frequency response is minimized. Meanwhile, the second approach minimizes the true error iteratively following the Steigliz-McBride idea [5]. As

an initial condition, we can use the solution from the first method, thereby potentially improving the approximation accuracy of that solution. The two proposed methods can also be extended to a universal design by gridding the graph frequency domain as mentioned earlier.

- *iii) We use several numerical tests to validate our findings with both synthetic and real data.* We show that the ARMA filters outperform FIR filters and the IIR filters from [4] filters in terms of approximation accuracy, even with fewer filter coefficients. In our tests with the real Molene temperature dataset, ARMA graph filters are used for data compression (on a directed graph) and prediction (on both a directed and undirected graph) of the graph signal. The results show that the error resulting from our ARMA filter design is lower than that resulting from an FIR filter with the same number of filter coefficients.

4.2. ARMA GRAPH FILTER DESIGN

This section contains the proposed ARMA filter design methods. We start with a discussion of the ARMA design problem, followed by two approaches inspired by Prony's method, and finally an iterative approach.

4.2.1. ARMA DESIGN PROBLEM

Here, we first recall the definition of an ARMA graph filter, mentioned in Chapter 3. An autoregressive moving average graph filter is represented with a rational polynomial in the graph frequency λ [6], i.e.,

$$g(\lambda) = \frac{\sum_{q=0}^{Q} b_q \lambda^q}{\sum_{p=0}^{P} a_p \lambda^p},$$

with denominator coefficients a_p and numerator coefficients b_q, leading to the graph filter in the shift operator $\boldsymbol{S}$ given by

$$\boldsymbol{G} = \left(\sum_{p=0}^{P} a_p \boldsymbol{S}^p \right)^{-1} \sum_{q=0}^{Q} b_q \boldsymbol{S}^q,$$

where $\boldsymbol{a} = [a_0, \ldots, a_P]^T$ and $\boldsymbol{b} = [b_0, \ldots, b_Q]^T$ collect the ARMA filter coefficients. This type of graph filter is labeled as an ARMA graph filter of denominator order P and numerator order Q, expressed as ARMA(P,Q) in short.

As discussed in Chapter 3, we would like to find the ARMA filter coefficients $\boldsymbol{a}$ and $\boldsymbol{b}$ such that a desired frequency response $\hat{h}_n$ is matched, where the latter can be a desired filter shape (for filter design, smoothing, or denoising) or the GFT of a graph signal (for compression or prediction). In this context, note that many desired responses $\hat{h}_n$ already have the shape of an ARMA filter, e.g., for Tikhonov denoising or interpolation, which means no explicit fitting is required in that case.

More specifically, adapting (2.26) to our ARMA filter design problem, we want to minimize the following error

$$e_n = \hat{h}_n - \hat{g}_n = \hat{h}_n - \frac{\sum_{q=0}^{Q} b_q \lambda_n^q}{\sum_{p=0}^{P} a_p \lambda_n^p}. \tag{4.1}$$

Since (4.1) is nonlinear in $\boldsymbol{a}$ and $\boldsymbol{b}$, classical approaches like Prony's method [5] consider minimizing the following modified error

$$e'_n = \hat{h}_n \left(\sum_{p=0}^{P} a_p \lambda_n^p \right) - \sum_{q=0}^{Q} b_q \lambda_n^q. \tag{4.2}$$

The latter is clearly not equivalent to (4.1) but it is linear in $\boldsymbol{a}$ and $\boldsymbol{b}$.

In the sequel, our goal will be to find $\boldsymbol{a}$ and $\boldsymbol{b}$ that minimize (4.1) or (4.2) in the mean square sense, subject to $a_0 = 1$ as mentioned in Chapter 3. Similar to the FIR filter, if we want the ARMA filter to make sense as a graph filter that will be applied to a real-valued graph signal $\boldsymbol{x}$, we want the ARMA filter coefficients $\boldsymbol{a}$ and $\boldsymbol{b}$ to be real-valued. We will show that this is the case for the different proposed approaches.

Finally, note that, similar to Prony's method [5], the non-convex stability constraint $\sum_{p=0}^{P} a_p \lambda_n^p \neq 0$ will be ignored in the rest of the chapter, but it can easily be checked after the design.

4.2.2. METHODS INSPIRED BY PRONY

In this section, we propose the Prony inspired design methods, i.e., Prony's LS and Prony's projection.

Prony's LS.

To start, let us first stack e_n from (4.1) in the vector $\boldsymbol{e} = [e_1, \ldots, e_N]^{\mathrm{T}}$, which can be expressed as

$$\boldsymbol{e} = \hat{\boldsymbol{h}} - \mathrm{diag}(\boldsymbol{\Psi}_{P+1}\boldsymbol{a})^{-1}\boldsymbol{\Psi}_{Q+1}\boldsymbol{b}. \tag{4.3}$$

As we mentioned before, this nonlinear function is hard to handle and thus we focus on the modified error. Stacking e'_n from (4.2) in the vector $\boldsymbol{e}' = [e'_1, \ldots, e'_N]^{\mathrm{T}}$, we obtain the simpler linear expression

$$\boldsymbol{e}' = \hat{\boldsymbol{h}} \circ (\boldsymbol{\Psi}_{P+1}\boldsymbol{a}) - \boldsymbol{\Psi}_{Q+1}\boldsymbol{b} \tag{4.4}$$

$$= [\boldsymbol{\Psi}_{P+1} \circ (\hat{\boldsymbol{h}}\mathbf{1}_{P+1}^{\mathrm{T}})]\boldsymbol{a} - \boldsymbol{\Psi}_{Q+1}\boldsymbol{b}, \tag{4.5}$$

4

where "$\circ$" represents the element-wise Hadamard product and $\mathbf{1}_{P+1}$ is the $(P+1)\times 1$ all-one vector.

Minimizing $\|\boldsymbol{e}'\|^2$ over $\boldsymbol{a}$ and $\boldsymbol{b}$ leads to the following LLS problem

$$\min_{\boldsymbol{a},\boldsymbol{b}} \left\| [\boldsymbol{\Psi}_{P+1} \circ (\hat{\boldsymbol{h}}\mathbf{1}_{P+1}^{\mathrm{T}}), -\boldsymbol{\Psi}_{Q+1}] \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{b} \end{bmatrix} \right\|^2, \text{ s.t. } a_0 = 1, \tag{4.6}$$

which can be solved efficiently. The next proposition shows that the obtained $\boldsymbol{a}$ and $\boldsymbol{b}$ vectors are real-valued.

Proposition 2. *Under Property 2, the ARMA filter coefficients $\boldsymbol{a}$ and $\boldsymbol{b}$ obtained by solving (4.6) are real-valued.*

Proof. The proof is similar to the proof of Proposition 1. □

Since Prony's LS approach addresses the modified error (4.2) and not the desired error (4.1), we here consider a way to partly overcome this limitation, and potentially improve the approximation accuracy of (4.6).

Prony's projection.

We use the orthogonal subspace projection approach [7] to rephrase (4.4) as a function of only the denominator coefficients $\boldsymbol{a}$. Then, with the obtained solution for $\boldsymbol{a}$, the original error (4.1) can be minimized to find the numerator coefficients $\boldsymbol{b}$. This approach can be interpreted as Shanks' method similar to that used in [3].

Let us start by considering the orthogonal projection matrix onto the orthogonal complement of the range of $\boldsymbol{\Psi}_{Q+1}$

$$\boldsymbol{P}^{\perp}_{\boldsymbol{\Psi}_{Q+1}} = \boldsymbol{I}_N - \boldsymbol{\Psi}_{Q+1}\boldsymbol{\Psi}^{\dagger}_{Q+1}, \tag{4.7}$$

where $\boldsymbol{\Psi}_{Q+1}$ is better conditioned than $\boldsymbol{\Psi}_{K+1}$ used to design an FIR graph filter, because $Q < K$ and removing columns from a tall matrix improves its condition number.

Then, the modified error (4.4) can be reshaped as

$$\boldsymbol{e}'' = \boldsymbol{P}^{\perp}_{\boldsymbol{\Psi}_{Q+1}}[\boldsymbol{\Psi}_{P+1} \circ (\hat{\boldsymbol{h}}\mathbf{1}^{\mathrm{T}}_{P+1})]\boldsymbol{a} - \boldsymbol{P}^{\perp}_{\boldsymbol{\Psi}_{Q+1}}\boldsymbol{\Psi}_{Q+1}\boldsymbol{b}, \tag{4.8}$$

where the second term on the right-hand side of (4.8) is zero.

As shown in [8], [7], this projection operator preserves the solution for $\boldsymbol{a}$ when minimizing (4.8) instead of (4.4). Hence, after the projection, the LLS problem for solving $\boldsymbol{a}$ becomes

$$\min_{\boldsymbol{a}} \|\boldsymbol{P}^{\perp}_{\boldsymbol{\Psi}_{Q+1}}[\boldsymbol{\Psi}_{P+1} \circ (\hat{\boldsymbol{h}}\mathbf{1}^{\mathrm{T}}_{P+1})]\boldsymbol{a}\|^2, \text{ s.t. } a_0 = 1. \tag{4.9}$$

The reason why we prefer solving (4.9) over (4.6) for finding a solution for $\boldsymbol{a}$ is the computational complexity.

Finally, vector $\boldsymbol{b}$ can be obtained using (4.1) after plugging in the solution for $\boldsymbol{a}$ obtained from (4.9). In other words, $\boldsymbol{b}$ is found by solving

$$\min_{\boldsymbol{b}} \|\hat{\boldsymbol{h}} - \mathrm{diag}(\boldsymbol{\Psi}_{P+1}\boldsymbol{a})^{-1}\boldsymbol{\Psi}_{Q+1}\boldsymbol{b}\|^2. \tag{4.10}$$

As before, we can again show that this solution for $\boldsymbol{a}$ and $\boldsymbol{b}$ is real-valued.

Proposition 3. *Under Property 2, the ARMA filter coefficients $\boldsymbol{a}$ and $\boldsymbol{b}$ obtained by solving (4.9) and (4.10) are real-valued.*

Proof. The proof is similar to the proof of Proposition 1. □

We would like to remark that this version of Prony's projection approach has a conceptual difference from the method presented in [3]. While in [3] the desired frequency response is first fitted with an FIR filter and then the denominator coefficients are found to match that response, we here aim at approaching directly the desired response rather than its FIR approximation.

In parallel to the classical literature [5], our approach can be considered as a reshaping of the Padé approximation which first is solved for the denominator coefficients $\boldsymbol{a}$ and then for the numerator coefficients $\boldsymbol{b}$. As we show in the numerical experiments, Prony's projection approach improves, in general, the approximation accuracy of (4.6).

4.2.3. ITERATIVE APPROACH

In this section, we present the iterative approach to design the ARMA coefficients. The idea consists of updating recursively the filter coefficients while minimizing the original error (4.1). We first reformulate the problem to make it amenable to our iterative approach and then use a variant of the Steigliz-McBride method [5] to implement an iterative algorithm that can be utilized for finding the ARMA graph filter coefficients.

Problem reformulation.

The focus in the previous section was on solving (4.2). This of course comes with a lack of optimality, since we aim to solve (4.1). In the iterative approach, instead, we focus directly on minimizing (4.1).

To ease the notation, let us define

$$\beta_n = \sum_{q=0}^{Q} b_q \lambda_n^q \quad \text{and} \quad \alpha_n = \sum_{p=0}^{P} a_p \lambda_n^p,$$

and rewrite the original error (4.1) as

$$e_n = \hat{h}_n - \frac{\beta_n}{\alpha_n}. \tag{4.11}$$

Then, by defining $\gamma_n = 1/\alpha_n$, we have

$$e_n = \hat{h}_n - \beta_n \gamma_n, \tag{4.12}$$

which can be equivalently expressed as

$$e_n = (\hat{h}_n \alpha_n - \beta_n)\gamma_n. \tag{4.13}$$

Note that the expression (4.13) is linear in α_n, β_n, and γ_n, if each of them is treated as a separate variable. To avoid inversion issues when $\alpha_n = 0$, we can consider $\gamma_n = 1/(\alpha_n + \rho)$ for some $\rho \approx 0$. Note that if γ_n is fixed, e_n becomes linear in the variables α_n and β_n. This will be our starting point to minimize e_n recursively. In each iteration, having found a new set of solutions for α_n, β_n we can then find a_p and b_q as well as update γ_n.

To follow the convention of the previous sections, we write (4.13) in a more convenient vector form, by defining the vectors $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_N]^{\mathrm{T}}$, $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_N]^{\mathrm{T}}$,

and $\boldsymbol{\gamma} = [\gamma_1, \ldots, \gamma_N]^{\mathrm{T}}$. Then, the error vector $\boldsymbol{e} = [e_1, \ldots, e_N]^{\mathrm{T}}$ containing the original error for all graph frequencies can be written as

$$\boldsymbol{e} = [\hat{\boldsymbol{h}} \circ \boldsymbol{\alpha} - \boldsymbol{\beta}] \circ \boldsymbol{\gamma}. \tag{4.14}$$

Iterative algorithm.

Let $\boldsymbol{\alpha}^{(i)}$ and $\boldsymbol{\beta}^{(i)}$ respectively denote the estimates of the vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, at the i-th iteration. We can then find the value of $\boldsymbol{\gamma}$ as an element-wise inversion of $\boldsymbol{\alpha}^{(i)}$, which we label as $\boldsymbol{\gamma}^{(i)}$,

$$\boldsymbol{\gamma}^{(i)} = \begin{bmatrix} \frac{1}{\alpha_1^{(i)}+\rho} & \frac{1}{\alpha_n^{(i)}+\rho} & \cdots & \frac{1}{\alpha_N^{(i)}+\rho} \end{bmatrix}^T. \tag{4.15}$$

Using this value for $\boldsymbol{\gamma}$, we obtain the updated error

$$\boldsymbol{e}^{(i+1)} = (\hat{\boldsymbol{h}} \circ \boldsymbol{\alpha}) \circ \boldsymbol{\gamma}^{(i)} - \boldsymbol{\beta} \circ \boldsymbol{\gamma}^{(i)}, \tag{4.16}$$

which is linear in the unknown variables $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. Minimizing this error leads to the updated values $\boldsymbol{\alpha}^{(i+1)}$ and $\boldsymbol{\beta}^{(i+1)}$. This procedure is then repeated till a desirable solution is obtained.

To formalize this iteration, and express it as a direct function of the true filter coefficients $\boldsymbol{a}$ and $\boldsymbol{b}$, we can reformulate (4.16) as

$$\boldsymbol{e}^{(i+1)} = \boldsymbol{H}^{(i)} \boldsymbol{a} - \boldsymbol{B}^{(i)} \boldsymbol{b}, \tag{4.17}$$

where $\boldsymbol{H}^{(i)} = (\boldsymbol{\gamma}^{(i)} \mathbf{1}_{P+1}^{\mathrm{T}}) \circ \boldsymbol{\Psi}_{P+1} \circ (\hat{\boldsymbol{h}} \mathbf{1}_{P+1}^{\mathrm{T}})$ and $\boldsymbol{B}^{(i)} = (\boldsymbol{\gamma}^{(i)} \mathbf{1}_{Q+1}^{\mathrm{T}}) \circ \boldsymbol{\Psi}_{Q+1}$. The specific derivations that lead to (4.17) can be found in Appendix B.

With this in place, the filter coefficients at the $(i+1)$-th iteration are found by solving

$$\min_{\boldsymbol{a},\boldsymbol{b}} \left\| \left[\boldsymbol{H}^{(i)}, -\boldsymbol{B}^{(i)} \right] \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{b} \end{bmatrix} \right\|^2 \text{ s.t. } a_0 = 1. \tag{4.18}$$

The solutions $\boldsymbol{a}^{(i+1)}$ and $\boldsymbol{b}^{(i+1)}$ are again real-valued as shown in the following Proposition.

Proposition 4. *Under Property 2, the ARMA filter coefficients $\boldsymbol{a}$ and $\boldsymbol{b}$ obtained by solving (4.18) are real-valued.*

Proof. The proof is similar to the proof of Proposition 1. □

Algorithm 4.1: Iterative approach

Input: $\boldsymbol{a}^{(0)}$, $\hat{\boldsymbol{h}}$, number of iterations τ, threshold δ_c

Initialization: $\boldsymbol{\gamma}^{(0)}, \boldsymbol{H}^{(0)}, \boldsymbol{B}^{(0)}, \hat{\boldsymbol{g}}^{(0)}, \boldsymbol{e}^{(0)}$

Iteration: while $i < \tau$ and $\delta < \delta_c$

solve $\min_{\boldsymbol{a},\boldsymbol{b}} \left\| \left[\boldsymbol{H}^{(i)}, -\boldsymbol{B}^{(i)} \right] \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{b} \end{bmatrix} \right\|^2$ s.t. $a_0 = 1$.

return $\boldsymbol{a}^{(i+1)}$, $\boldsymbol{b}^{(i+1)}$

compute $\hat{\boldsymbol{g}}^{(i+1)}$, $\boldsymbol{e}^{(i+1)}$, $\delta = \|\boldsymbol{e}^{(i+1)} - \boldsymbol{e}^{(i)}\|$

update $\boldsymbol{\gamma}^{(i+1)}$

$i = i + 1$

Output: $\boldsymbol{a}^{(i+1)}$, $\boldsymbol{b}^{(i+1)}$

For the above two design methods, the design cost of Prony's method is related to the LLS solution which requires $O((P+Q+1)^2 N)$ operations, while for the iterative approach, the total design cost is τ times larger leading to a cost of $O(\tau(P+Q+1)^2 N)$. Since the number of nodes N is much smaller than the number of edges E, the design cost is smaller than the implementation cost. Algorithm 4.1 summarizes the iterative approach for ARMA graph filter design.

Remark 1. We stop the iterations when δ, representing the error difference between two successive iterations, is smaller than a given threshold δ_c. However, depending on the specific combination of P and Q, the method does not always converge fast enough or it does not converge at all. For those cases, we consider a maximum number of iterations τ and search for the minimum error over all iterations. We then assume that this iteration provides the solution to the problem. As we will see in the numerical section, for a fixed order K, the best performance for $P+Q=K$ always leads to a significant improvement in approximation accuracy over the former methods. However, for a fixed order K, some combinations of P,Q yield instabilities around the cut-off frequency. The latter is especially present in Prony's method. Therefore, a search over different combinations of P,Q is recommended.

Remark 2. For $\boldsymbol{\gamma}^{(0)} = \mathbf{1}$, the LLS procedure (4.6) can be seen as a special case of the iterative approach. With $\boldsymbol{\gamma}^{(0)} = \mathbf{1}$, the formulation of the iterative approach degenerates into the LLS solution, and the approximation error changes from the original error (4.1) to the modified error (4.2). However, since Prony's projection

approach leads to better results than Prony's LS approach, we prefer the latter to initialize the iterative approach.

4.3. NUMERICAL DATA

In this section, we present our numerical evaluation of the proposed methods and compare them with the FIR graph filters. The performance is tested with both synthetic and real data. Our tests with the Molene dataset show that ARMA filters are more suitable than FIR filters for lossy data compression, where we can save up to 50% of memory with very little error. Further, we apply ARMA filters in the context of prediction (as in [9]) and we show that ARMA graph filters outperform FIR graph filters, where with only 4 bits we achieve a reconstruction error of 10^{-3}. Throughout our simulations, we make use of the GSPBox [10].

4.3.1. SYNTHETIC SIMULATION RESULTS

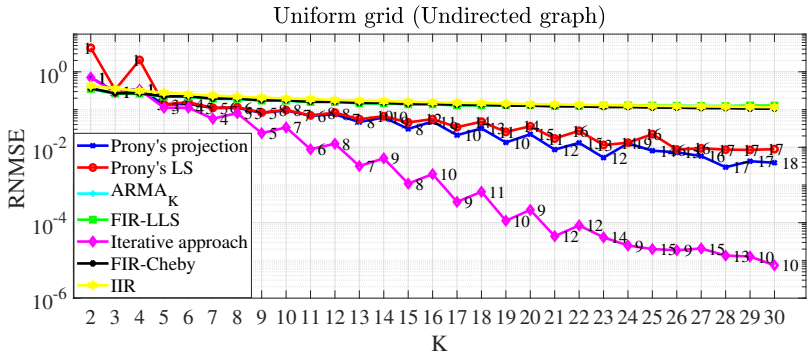
In this section, we evaluate the performance of the proposed design algorithms in approximating the desired frequency response. The performance is assessed for two different settings, namely a *universal* filter design and a filter design for an Erdős Rényi (ER) graph. For both cases we consider $N = 100$ grid points/nodes. We remark that more grid points/nodes, i.e., $N = 300, 1000$, result in similar errors and trends as for $N = 100$. In both settings, the goal is to approximate the ideal low-pass frequency response introduced in Chapter 3 and illustrated in Fig. 3.1.

Universal design: For the universal design, we follow the approach discussed in Chapters 2 and 3. For an undirected graph, we consider $\boldsymbol{S} = \boldsymbol{L}_{\text{n}}$ and sample the interval $[0, 2]$ uniformly. For a directed graph, we consider $\boldsymbol{S} = \boldsymbol{A}_{\text{n}}$ and sample the complex unit disc uniformly in amplitude and phase. We assume $N = 100$ grid points for both types of graphs.

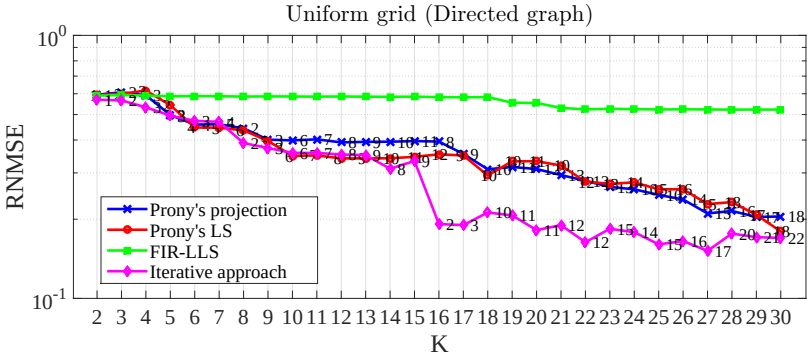
Design for Erdős Rényi graph: For the undirected ER graph [11], we assume that a pair of nodes is connected with a probability $p = 0.1$ and the shift operator is again $\boldsymbol{S} = \boldsymbol{L}_{\text{n}}$. Due to the graph randomness, we always average the results over 100 different realizations.

In the sequel, we analyze the design methods proposed in Section 4.2 and compare them to the related FIR filter design. If not mentioned otherwise, we de-

Access to the raw data is through the link: https://donneespubliques.meteofrance.fr/donnees_libres/Hackathon/RADOMEH.tar.gz



(a) Universal design by gridding the spectrum in $N = 100$ ($\boldsymbol{S} = \boldsymbol{L}_{\text{n}}$) points for an undirected graph.



(b) Universal design by gridding the spectrum in $N = 100$ ($\boldsymbol{S} = \boldsymbol{A}_{\text{n}}$) points for a directed graph.

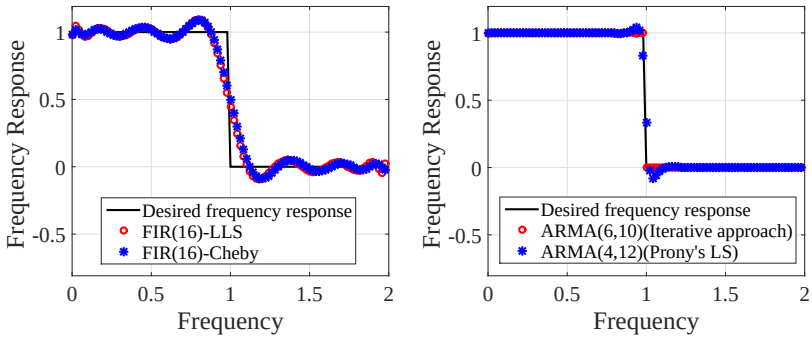
Figure 4.1: RNMSE of the proposed design methods for different orders K (such that $P + Q = K$) in approximating an ideal low-pass frequency response. For the ARMA filters, the order Q is shown in the plot.

sign the FIR filter using the LLS approach of (3.12) (FIR-LLS, or simply FIR). The universal FIR design for undirected graphs sometimes also follows the Chebyshev design of [12] (FIR-Cheby). We compare the ARMA(P,Q) filter to a FIR(K) graph filter where $P + Q = K$ is satisfied. We look for all combinations of P and Q that satisfy $P + Q = K$ and pick the combination leading to the best result. Since we want the overall order of the designed ARMA graph filter to be small, we only investigate the range $2 \leq K \leq 30$. We measure the approximation accuracy with the root normalized mean square error (RNMSE) of the frequency response of

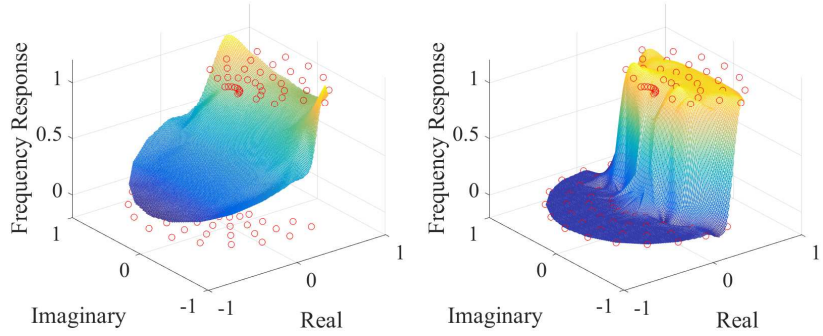
the filter:

$$\text{RNMSE} = \frac{\|\hat{\boldsymbol{h}} - \hat{\boldsymbol{g}}\|}{\|\hat{\boldsymbol{h}}\|}. \tag{4.19}$$

Note that, for a directed graph with complex frequencies, since the filter response can be complex-valued, we only compute the approximation error for the amplitude (absolute value) of the filter response under the assumption that the desired frequency response is real.



(a) Low-pass filter for undirected graph.

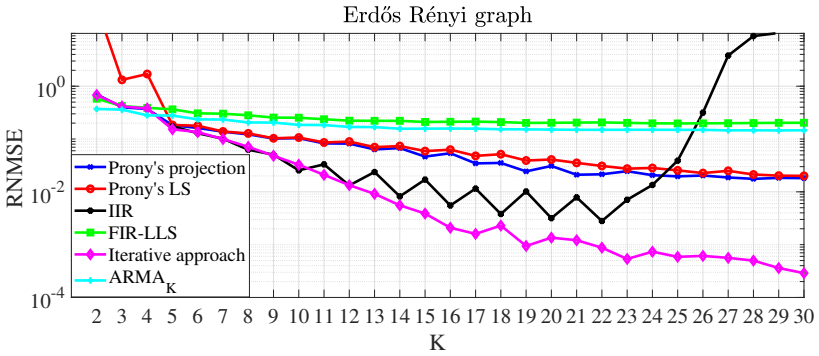


(b) Low-pass filter for directed graph

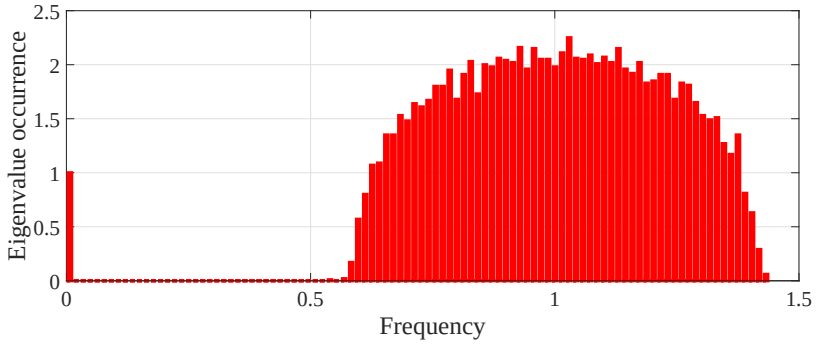
Figure 4.2: Comparison of FIR and ARMA graph filters with order $K = 16$. (a) Comparison of FIR and ARMA with same order $K = 16$ for an undirected graph. The graph filters correspond to Fig. 4.1(a). (b) Comparison of FIR and ARMA with same order $K = 16$ for a directed graph. The FIR graph filter (left) and ARMA graph filter (right) correspond to the green and pink lines in Fig. 4.1(b). The desired frequency response is shown in the plot as red points.

Performance analysis. In Fig. 4.1 we show the RNMSE for the Prony inspired methods and the iterative approach. Specifically, the depicted RNMSE in Fig. 4.1

(a) (b) are related to the best combination (P,Q) for each particular K such that $P+Q=K$. The iterative approach is initialized with the solution of Prony's projection method (4.9) and (4.10), to show its potential in improving the RNMSE. Additionally, the FIR, ARMA$_K$ [3] and IIR [4] performances are plotted as a benchmark.



(a) Results for the average of 100 Erdős Rényi graphs with $N=100$ nodes and $p=0.1$.



(b) Eigenvalue occurrence of 100 Erdős Rényi graph realizations.

Figure 4.3: Design methods of ARMA graph filters applied to the ER graph.

Based on these results we can make the following observations:

- i) We can notice that the FIR (FIR-LLS or FIR-Cheby) approximation errors for both universal designs (Fig. 4.1(a), (b)) and the design for the ER graph (Fig. 4.3(a)) is the highest, except when $K \leq 5$. Further, the FIR approximation accuracy, even when designed for the specific set of ER graph frequencies, does not improve with the order K. We believe that this effect is due

to the eigenvalue spread of the ER graph, since some of its eigenvalues are more closely spaced than in a uniform grid (see e.g., Fig. 4.3(b)).

- ii) Compared to Prony's method, the iterative approach has a larger design cost but improves the approximation for higher-order K. Prony's method gives a comparable performance to the iterative approach only up to $K = 8$. We see that Prony's LS approach is not suitable for the ER graph when $K \leq 5$, while for a universal design approach its performance is close to that of Prony's projection method. This highlights that the LS approach should be avoided in graphs that have closely spaced eigenvalues. On the other hand, this issue is overcome by Prony's projection method which gives a small RNMSE also for values $K \leq 5$.

- iii) As an example, we take the order $K = 16$ to show the difference in performance between FIR graph filters and ARMA graph filters in Fig. 4.2(a), (b). It is remarkable to highlight that the iterative approach outperforms the FIR by several orders, where the latter has a comparable performance only for $K \leq 3$. Such a finding shows that ARMA graph filters are more suitable for applications demanding higher approximation accuracies.

- iv) We observe a smaller RNMSE for undirected graphs compared to directed graphs. This is because we can do a fitting on the real line instead of in the complex plane. In contrast to undirected graphs, notice that for directed graphs, as shown in Fig. 4.1 (b), all ARMA graph filter design approaches yield similar performance.

- v) As highlighted in Fig. 4.1 (a), an important role is played by the MA order Q (which is generally larger than P). We observe that a higher Q improves the stability of the ARMA filters, specifically for Prony's projection method and the iterative approach where the numerator coefficients are found by minimizing the true error.

- vi) If the frequencies are different, the Vandermonde matrix $\boldsymbol{\Psi}$ is theoretically full rank (invertible) but generally ill-conditioned. Although this issue is encountered for both FIR and ARMA graph filters, ARMA filters improve the conditioning of the matrix because the filter orders P and Q can be selected much lower than the FIR filter order K. Hence, the solution of our design methods has uniqueness, but there might be a conditioning problem when the orders are increased.

- vii) For the universal design (Fig. 4.1(a)) and ER graph (Fig. 4.3(a)), we also compare our approach with the methods in [3, 4]. The ARMA$_K$ graph filter [3] has the same order for the nominator and denominator, therefore, we adopt the same value K as the order for both the nominator and denominator. Note that this leads to a total order that is twice the order of our ARMA(P,Q) (recall that $K = P+Q$). For the universal design, we further compare our approach with the universal Butterworth filter [4]. The IIR graph filter [4] is then tested on the ER graph. We follow the scenario of [4] and use a denominator of degree 4, leading to a nominator of degree $(K-4)$. The results show that for low orders ($K < 12$), the IIR graph filter [4] has a similar performance to our iterative approach. However, with an increasing order $K > 12$, our design method offers a better approximation accuracy.

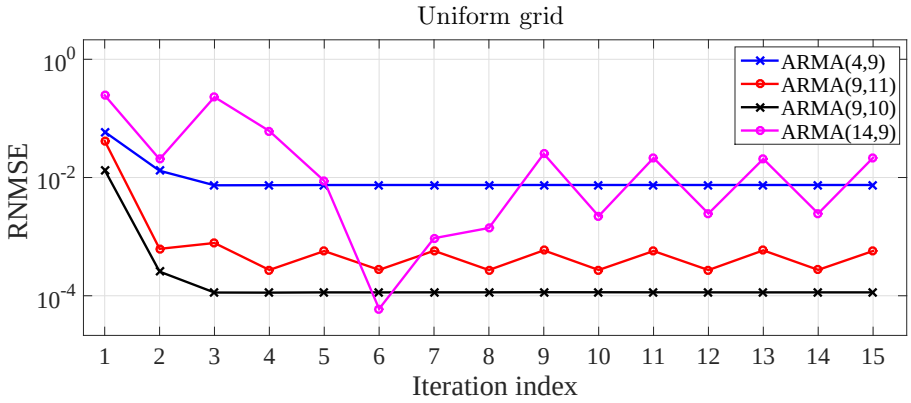


Figure 4.4: RNMSE of the iterative approach on the universal design with $N = 100$ points. Performance evaluation for different ARMA filters which are a few particular cases illustrating monotonic convergence, non-monotonic convergence, and no convergence.

Iterative approach. We now analyze in more detail the iterative approach, to highlight its benefits in improving the ARMA filter accuracy compared to Prony's projection approach.

We consider two cases with monotonic convergence, namely, an ARMA$(9,10)$ (characterized by an RNMSE of order 10^{-2} in Prony's projection method, Fig. 4.1 (a)) and an ARMA$(4,9)$ (characterized by an RNMSE of order 10^{-1} in Prony's projection method, Fig. 4.1 (a)) which are considered due to their low orders. For both cases we initialize the iterations with the solution of Prony's projection

method. Note that ARMA(9,10) is the best combination P,Q of order $K = 19$, while ARMA(4,9) is not the best combination for order $K = 13$. We also consider two filters, the ARMA(9,11) and ARMA(14,9) to illustrate that even without monotonic convergence, the approximation accuracies can be improved with our iterative approach.

In Fig. 4.4 we show the approximation error as a function of the iteration index and we can immediately notice that for those filters with monotonic convergence, the approximation errors reduce in a few iterations. More specifically, for the ARMA(9,10) the iterative approach reduces the error from 10^{-2} to 10^{-4}. It is also worth noticing that using the iterative approach, the ARMA(9,10) outperforms also the ARMA(11,17), which is the best filter that can be designed with Prony's projection method (within the considered range). Similarly, the iterative approach improves the approximation accuracy for the low order filter ARMA(4,9). Indeed, its performance is now comparable with all other ARMAs and FIRs with much greater orders. As we mentioned in the previous section, for the non-converging filters, we pick the best approximation result during the iterative procedure, e.g., the performance in the 6-th iteration of the ARMA(14,9) filter, which is better than the performance of the ARMA(9,10) filter.

4.3.2. DATA COMPRESSION WITH GRAPH FILTERS

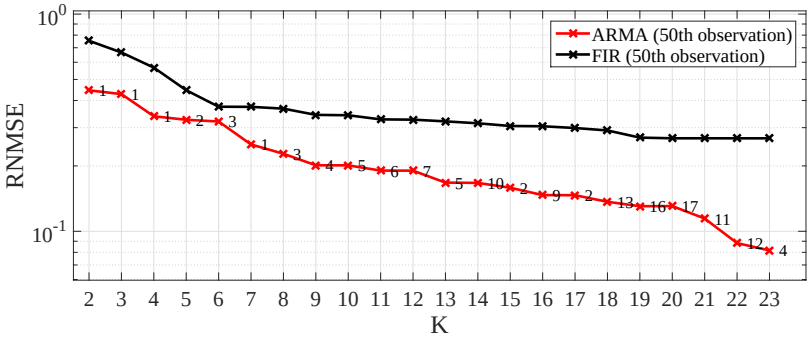
Our goal, in this subsection, is to show that ARMA filters of low orders can be used to represent the data and perform compression. We use the Molene weather data set which contains hourly observations of temperature measurements collected in January 2014 in the 32 cities (nodes) of the region Brest (France).

Experimental set up. We consider fitting a small order ARMA graph filter to each data realization and then store the filter coefficients instead of the actual data. We now create the graph as a directed 6-nearest neighbor connection. In the directed graph, each vertex is connected to its six closest nodes by means of directed edges [2]. The weight of the edge between v_m and v_n is given as

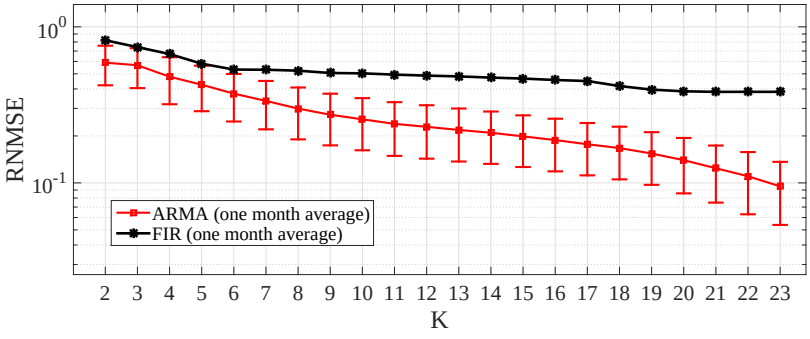
$$[\boldsymbol{A}]_{n,m} = \frac{e^{-d_{n,m}^2}}{\sqrt{\sum_{k\in\mathcal{N}_n} e^{-d_{n,k}^2} \sum_{l\in\mathcal{N}_m} e^{-d_{m,l}^2}}} \tag{4.20}$$

where $d_{n,m}$ represents the geometric distance between nodes v_n and v_m and $\mathcal{N}_n$, $\mathcal{N}_m$ represent the sets of neighbors of node v_n and v_m. Note that the resulting matrix $\boldsymbol{A}$ is normal, i.e., $\|\boldsymbol{A}\| = 1$. For every data realization $\boldsymbol{x}$, we take the GFT to compute $\hat{\boldsymbol{x}}$ and fit it to an ARMA(P,Q) graph filter. The filter coefficients are

derived using the iterative approach with the initial condition given by Prony's projection method. We measure the compression performance as the RNMSE between the compressed signal and the real one $\boldsymbol{x}$. As a benchmark, we again consider the FIR(K) with $K = P + Q$.



(a) RNMSE of graph filters for one realization.



(b) Average RNMSE over all 744 temperature realizations.

Figure 4.5: RNMSE between the data spectrum and the filter frequency response as a function of filter order K. (a) Illustration of the RNMSE of the ARMA graph filter and the same order FIR filter for the 50th observation. The order Q is shown in the plot and $P + Q = K$. (b) Average RNMSE over all 744 temperature realizations (one month) for different filter orders. For the ARMA filter, each error bar shows the standard deviation of the approximation error for order K.

Results. In Fig. 4.5(a), we show the RNMSE as a function of K for the 50-th observation. We observe that the ARMA filter achieves a smaller RNMSE than the FIR filter even for small orders K. As expected, when K approaches N, we have a smaller error but we also see that the gap in performance between the ARMA and FIR filters increases. This result goes in line with what we obtained in the

previous section for synthetic data.

To further quantify the above observations, Fig. 4.5(b) depicts the average performance over all observations. We still notice that the ARMA graph filters achieve a smaller RNMSE than FIR graph filters and that the RNMSE decreases for higher values of K. With the above approach, a compression ratio of 25% ($K = 23$) is achieved when an RNMSE of 10^{-1} can be tolerated. Note that next to signal compression, the ARMA model can also be used to reconstruct the graph power spectrum of stationary graph signals from a subset of the nodes [13].

Remark 3. To achieve further compression one can exploit also the stationarity of the signal over time. Thus, instead of fitting a graph filter to each individual observation, one approach may consider fitting a joint graph-temporal filter [14], [15] to the time-varying data.

4.3.3. LINEAR PREDICTION WITH ARMA FILTERS

Inspired by [9], we also test linear prediction (LP) on graphs using ARMA graph filters. We consider the Molene data set and again compare the ARMA graph filters with the FIR graph filters [9]. The considered problem contains two parts, namely the forward (prediction) part and the backward (synthesis) part.

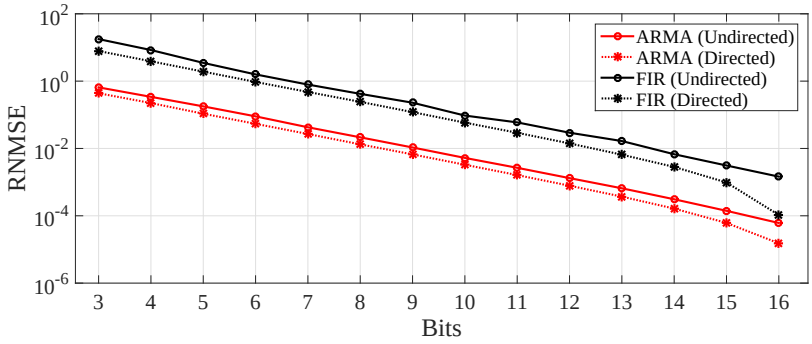
In forward filtering, the residual between the graph signal and the filter frequency response is calculated and quantized. Next, the backward filter considers building an approximation of the graph signal from the quantized residual. For the ARMA filters, we use a variant of the iterative approach to find the filter coefficients, while for the FIR filter we follow [9]. For the graph shift operator $\boldsymbol{S}$, we consider both the directed graph created by (4.20) and the undirected graph [16].

Experimental set up. For the ARMA filter, given the graph signal $\boldsymbol{x}$, the residual $\boldsymbol{r}$ related to signal prediction is given by

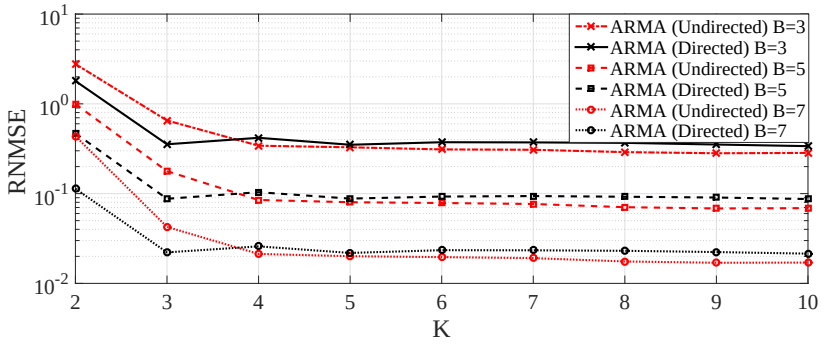
$$\boldsymbol{r} = \boldsymbol{x} - g(\boldsymbol{S})\boldsymbol{x} = \boldsymbol{x} - (\sum_{p=0}^{P} a_p \boldsymbol{S}^p)^{-1} (\sum_{q=0}^{Q} b_q \boldsymbol{S}^q)\boldsymbol{x}. \tag{4.21}$$

Notice that next to the constraint $a_0 = 1$ we had before, it is important to set $b_0 = 0$ in order to avoid the trivial solution. Similar to Prony's method, we can derive also a modified residual as

$$\boldsymbol{r}' = (\sum_{p=0}^{P} a_p \boldsymbol{S}^p)\boldsymbol{x} - (\sum_{q=0}^{Q} b_q \boldsymbol{S}^q)\boldsymbol{x}. \tag{4.22}$$



(a) Average RNMSE of the approximated signal as a function of the number of bits (B) for filter order $K = 3$.



(b) Average RNMSE of the estimated signal for different order ARMA filters evaluated for $B = 3, 5, 7$.

Figure 4.6: Average RNMSE of linear prediction on the Molene temperature data set.

To relate this prediction problem to filter design, we can look at the residual and modified residual in the frequency domain, leading to

$$\hat{\boldsymbol{r}} = \hat{\boldsymbol{x}} \circ (\mathbf{1}_N - \text{diag}(\boldsymbol{\Psi}_{P+1}\boldsymbol{a})^{-1}\boldsymbol{\Psi}_{Q+1}\boldsymbol{b}), \tag{4.23}$$

and

$$\hat{\boldsymbol{r}}' = \hat{\boldsymbol{x}} \circ [\mathbf{1}_N \circ (\boldsymbol{\Psi}_{P+1}\boldsymbol{a}) - \boldsymbol{\Psi}_{Q+1}\boldsymbol{b}]. \tag{4.24}$$

Hence, up to the element-wise multiplication with $\hat{\boldsymbol{x}}$, this residual $\hat{\boldsymbol{r}}$ and modified residual $\hat{\boldsymbol{r}}'$ look like the error $\boldsymbol{e}$ in (4.3) and modified error in $\boldsymbol{e}'$ in (4.4), respectively, with $\hat{\boldsymbol{h}}$ replaced by the all-one vector $\mathbf{1}_N$. As a result, all previous de-

sign methods can still be used. They only need to be adapted with an appropriate weighting (coming from $\hat{\boldsymbol{x}}$) and with the constraint $b_0 = 0$.

Once the filter coefficients that (approximately) minimize the residual $\boldsymbol{r}$ are found, this residual is quantized with B bits (resulting in $\boldsymbol{r}_\mathrm{q}$) and forwarded. Then, by applying the backward filter $\boldsymbol{H} = (\boldsymbol{I} - g(\boldsymbol{S}))^{-1}$ to the residual, the approximated signal $\widetilde{\boldsymbol{x}} = \boldsymbol{H}\boldsymbol{r}_\mathrm{q}$ is constructed at the receiving side.

We consider ARMA graph filters for $K \leq 10$ ($K = P + Q$) and for every order K, the residual $\boldsymbol{r}$ is quantized with different numbers of bits. From the B bits, we spend one bit on the sign, $b = \lceil \log_2(\max(|[\boldsymbol{r}]_i|)) \rceil$ bits on the integer part, and the rest of the $(B - b - 1)$ bits on the decimal fraction.

Results. We quantify the performance in terms of RNMSE between the predicted signal $\tilde{\boldsymbol{x}}$ and the original one $\boldsymbol{x}$.

The average approximation error over all 744 realizations is shown in Fig. 4.6(a) as a function of the number of bits (B) used in the quantization for $K = 3$. We can notice that in a direct comparison with the FIR filters the approximation error of the ARMA graph filters is more than one order of magnitude lower. For both filters, as expected, more quantization bits B lead to a better approximation accuracy. Such findings suggest once again that ARMA filters are more suitable than FIR filters for applications demanding higher approximation accuracies.

To better highlight the performance of the ARMA filters, in Fig.4.6(b) we show the RNMSE as a function of the filter order K for different values of B. These results show that the approximation error for $K > 4$ remains constant, similar to what was observed for FIR filters in [9]. This observation suggests that small order filters are preferred for this application. Note that the performance for directed and undirected graphs is almost the same. The directed graph gives the best performance with $K = 3$ while for $K > 3$ the undirected graph gives a lower error. To conclude, we can say that using an ARMA graph filter with $K = 4$ and $B = 7$ (instead of 16 bits) we can reconstruct the data with an error of order 10^{-2} and save 62.5% in transmission costs.

4.4. CONCLUSIONS

In this chapter, we have presented ARMA graph filters as well as different methods to perform the filter design on both directed and undirected graphs. The first two filter design approaches,which focus on minimizing the modified error, are inspired by Prony's method. The third one iteratively minimizes the original error of the design problem. The iterative approach can be initialized with

the solution from one of the previous methods, which suggests that its performance can be improved by the iterative approach. Our theoretical findings are surrogated by numerical results on both synthetic and real data. In a direct comparison with the FIR graph filters, ARMA filters have shown to be more suitable for filter approximation, data compression and linear prediction on graphs.

REFERENCES

[1] 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).

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

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

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

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

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

[7] C.-I. Chang, *Orthogonal subspace projection (osp) revisited: A comprehensive study and analysis,* IEEE transactions on geoscience and remote sensing **43**, 502 (2005).

[8] Y. Hu and G. Leus, *On a unified framework for linear nuisance parameters,* EURASIP Journal on Advances in Signal Processing **2017**, 4 (2017).

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

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

[11] P. Erdos and A. Rényi, *On the evolution of random graphs,* Publ. Math. Inst. Hung. Acad. Sci **5**, 17 (1960).

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

[13] S. P. Chepuri and G. Leus, *Graph sampling for covariance estimation,* IEEE Transactions on Signal and Information Processing over Networks **3**, 451 (2017).

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

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

[16] S. P. Chepuri, S. Liu, G. Leus, and A. O. Hero III, *Learning sparse graphs under smoothness prior,* arXiv preprint arXiv:1609.03448 (2016).

APPENDIX-B

The error of the iterative approach on $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ is given by

$$\boldsymbol{e}^{(i+1)} = \boldsymbol{\gamma}^{(i)} \circ (\hat{\boldsymbol{h}} \circ \boldsymbol{\alpha}) - \boldsymbol{\beta} \circ \boldsymbol{\gamma}^{(i)} \tag{4.25}$$

By extending $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, we can rewrite (4.25) as

$$\boldsymbol{e}^{(i+1)} = \boldsymbol{\gamma}^{(i)} \circ \hat{\boldsymbol{h}} \circ (\boldsymbol{\Psi}_{P+1}\boldsymbol{a}) - (\boldsymbol{\Psi}_{Q+1}\boldsymbol{b}) \circ \boldsymbol{\gamma}^{(i)} \tag{4.26}$$

The first term in the right hand side of (4.26) can be expressed as

$$\begin{aligned} &\boldsymbol{\gamma}^{(i)} \circ \hat{\boldsymbol{h}} \circ (\boldsymbol{\Psi}_{P+1}\boldsymbol{a}) = \boldsymbol{\gamma}^{(i)} \circ [\hat{\boldsymbol{h}} \circ (\boldsymbol{\Psi}_{P+1}\boldsymbol{a})] \\ &= \boldsymbol{\gamma}^{(i)} \circ \left\{ [\boldsymbol{\Psi}_{P+1} \circ (\hat{\boldsymbol{h}}\mathbf{1}_{P+1}^{\mathrm{T}})]\boldsymbol{a} \right\} \\ &= [(\boldsymbol{\gamma}^{(i)}\mathbf{1}_{P+1}^{\mathrm{T}}) \circ \boldsymbol{\Psi}_{P+1} \circ (\hat{\boldsymbol{h}}\mathbf{1}_{P+1}^{\mathrm{T}})]\boldsymbol{a}. \end{aligned} \tag{4.27}$$

Similarly, the second term in the right hand side of (4.26) is rewritten as

$$(\boldsymbol{\Psi}_{Q+1}\boldsymbol{b}) \circ \boldsymbol{\gamma}^{(i)} = [(\boldsymbol{\gamma}^{(i)}\mathbf{1}_{Q+1}^{\mathrm{T}}) \circ \boldsymbol{\Psi}_{Q+1}]\boldsymbol{b}. \tag{4.28}$$

Finally, we define (4.27) and (4.28) as $\boldsymbol{H}^{(i)}\boldsymbol{a}$ and $\boldsymbol{B}^{(i)}\boldsymbol{b}$, respectively. This trivially leads to (4.17).

5

ARMA-Forsythe Graph Filter Design with Orthogonal Polynomials

Graph filters play an important role in the field of signal processing on graphs, especially for processing the spectrum of graph signals. In this chapter, the focus is on designing finite impulse response (FIR) and autoregressive moving average (ARMA) graph filters using an orthogonal basis. The main result of this chapter is that by generating orthogonal polynomials, we can improve the numerical condition of FIR and ARMA graph filters on both undirected and directed graphs. To be specific, we first introduce discrete orthogonal polynomials and use them to design FIR graph filters. Then, we demonstrate an efficient method for designing ARMA graph filters with discrete orthogonal polynomials for both directed and undirected graphs. The proposed method computes the orthogonal polynomial basis using Forsythe polynomials separately on the numerator and denominator parts of the ARMA model. The performance of the proposed algorithms is evaluated and compared with well-studied FIR and ARMA graph filters.

The remainder of this chapter is organized as follows. The discrete orthogonal polynomial basis and the FIR-Forsythe graph filter are presented in Section 5.2. In Section 5.3, we first introduce the ARMA model with the proposed discrete orthogonal basis and formulate the orthogonal polynomials separately for the numerator and denominator parts. Then, we express the solution to the ARMA-Forsythe filter design problem for both directed and undirected graphs. Simulation results on a few examples illustrating the proposed framework are shown in Section 5.4. Finally, the conclusions are drawn in Section 5.5.

5.1. INTRODUCTION

Finite impulse response (FIR) graph filters [1–3], direct analogs of temporal FIR filters, are implemented as a polynomial in the graph shift operator, e.g., the graph Laplacian matrix [4], the adjacency matrix [5], or any modification of them. With more degrees of freedom for filter design, the infinite impulse response (IIR) graph filter can be seen as an alternative to the FIR filter, such as the autoregressive moving average (ARMA) graph filters [6, 7], or the gradient descent IIR graph filters [8]. These filters are characterized by a rational frequency response which improves the approximation accuracy. In a centralized fashion, as mentioned in Chapters 3 and 4, the ARMA graph filter output can be simply found by solving a linear system of equations, which can be carried out efficiently with first-order methods [9] or conjugate gradient (CG) [10]. However, as for the FIR graph filter, the aforementioned ARMA also uses monomials in the graph frequency as basis functions (monomial basis) for the filter response in both numerator and denominator parts. Thus, those ARMA filters enjoy the same numerical problems as FIR graph filters.

Since estimating the graph frequencies entails some additional complexity, graph filters are often designed without any explicit knowledge of the graph or the graph frequencies, which is called universal design [7]. Following the methods in Chapter 4, FIR and ARMA graph filters can be designed in a universal fashion [7]. The Chebyshev polynomial technique is another popular method for designing graph filters in a universal way and it adopts some orthogonal polynomial basis [1]. The orthogonal Chebyshev approximation gives a slightly lower error than the monomial approximation in some applications such as wavelets [11]. However, Chebyshev polynomials are only suitable to design FIR - Chebyshev graph filters for undirected graphs. It is not easy to extend them for directed graphs that have real and complex conjugate frequencies. Also, the Chebyshev design is used to approximate the desired frequency response over a continuous frequency range, and for a limited filter order, it minimizes some weighted fitting errors. However, when we know the graph and hence the graph frequencies, this approach might not be optimal since the fitting error might not be minimal on the particular graph frequencies.

Similar to the concept of orthogonal Chebyshev polynomials, this chapter aims to introduce a *discrete* orthogonal polynomial basis (the Forsythe polynomial basis) into the graph filter design for both directed and undirected graphs. We mainly focus on the orthogonal basis for designing finite impulse response

(FIR) and autoregressive moving average (ARMA) graph filters.

The motivation of this chapter is exploiting the advantages of an orthogonal polynomial basis to improve FIR filter design as well as the filter design methods of Chapter 4, and this for a particular set of known graph frequencies. The main advantage of using a discrete polynomial orthogonal basis is the highly improved numerical condition of the design problem in computing the filter coefficients [7]. Moreover, for the FIR graph filters, another advantage of this approach is the simplicity of computing the coefficients. The orthogonal basis makes it easy to gradually increase the FIR filter order without having to compute the full set of coefficients from scratch. Furthermore, the presented discrete orthogonal polynomial basis is not only applicable to undirected graphs but also directed graphs.

The proposed methods and the contributions of this chapter are:

- *i) We bring discrete orthogonal polynomials into the design of FIR graph filters for both undirected and directed graphs.* Since the continuous orthogonal polynomial basis (Chebyshev polynomials) for FIR graph filters on undirected graphs is well studied, we adopt the discrete orthogonal polynomial basis for FIR filter design and this for either the normalized Laplacian (undirected graph) or the normalized adjacency (directed graph) matrix. We also show the relationship between the discrete orthogonal polynomial basis and the continuous orthogonal polynomial basis, and we discuss the filter design problem based on these orthogonal polynomials.

- *ii) We introduce an efficient filter design method using the discrete orthogonal polynomial basis in the ARMA model for both directed and undirected graphs.* Inspired by Prony's method [12], a modified error between the modeled and the desired frequency response is minimized for the ARMA model. We compute the orthogonal basis separately for the numerator and denominator parts. The solutions for undirected and directed graphs are formulated.

In the next section, we will first introduce the discrete orthogonal polynomial basis. Then, we give details about the filter design using an orthogonal basis for FIR graph filters. In Section 5.3, we will formulate the orthogonal polynomial basis for ARMA graph filters. The proposed method computes the orthogonal basis using Forsythe polynomials separately on the numerator and denominator parts.

5.2. ORTHOGONAL POLYNOMIAL BASIS

In this section, we introduce the idea of an orthogonal polynomial basis into the graph filter design. We first present the discrete orthogonal polynomial basis and formulate the FIR-Forsythe graph filter. Then, we extend the discrete orthogonal polynomial basis into a general orthogonal polynomial basis and discuss the related graph filter design problem.

5.2.1. FIR-FORSYTHE GRAPH FILTER

Similar to the structure of the FIR model, suppose we have a set of functions (polynomials) $f_k(\lambda)$ to represent the frequency response of the graph filter as

$$g(\lambda_n) = \sum_{k=0}^{K} g_k f_k(\lambda_n), \tag{5.1}$$

where g_k represents the filter coefficient corresponding to the basis function $f_k(\lambda)$. Note that using this representation form, the FIR graph filter in the frequency domain $g(\lambda_n) = \sum_{k=0}^{K} g_k \lambda_n^k$ has as basis functions $f_k(\lambda) = \lambda_n^k$ which is a set of monomials.

Let us first focus on undirected graphs with real-valued graph frequencies. Following (5.1), we can then write the error of the filter design problem for all graph frequencies as

$$e = \sum_{n=1}^{N} \{h(\lambda_n) - g(\lambda_n)\}^2 = \sum_{n=1}^{N} \left\{ h(\lambda_n) - \sum_{k=0}^{K} g_k f_k(\lambda_n) \right\}^2, \tag{5.2}$$

where $h(\lambda_n)$ is the desired frequency response. The error is a quadratic function of the filter coefficients g_k. Thus, to minimize the error (5.2), we have to solve the following problem

$$\frac{\partial e}{\partial g_l} = 2 \sum_{n=1}^{N} \left\{ h(\lambda_n) - \sum_{k=0}^{K} g_k f_k(\lambda_n) \right\} \big(-f_l(\lambda_n) \big) = 0. \tag{5.3}$$

Therefore, we observe that at any minimal point, the solution can be rewritten as

$$\sum_{k=0}^{K} g_k \left\{ \sum_{n=1}^{N} f_k(\lambda_n) f_l(\lambda_n) \right\} = \sum_{n=1}^{N} f_l(\lambda_n) h(\lambda_n). \tag{5.4}$$

Let $\langle f_k(\lambda_n), f_l(\lambda_n)\rangle$ denote the discrete inner product summed over the whole frequency range, i.e.,

$$\langle f_k(\lambda_n), f_l(\lambda_n)\rangle = \sum_{n=1}^{N} f_k(\lambda_n) f_l(\lambda_n). \tag{5.5}$$

With this definition, an orthogonal polynomial basis satisfies

$$\langle f_k(\lambda_n), f_l(\lambda_n)\rangle = \sum_{n=1}^{N} f_k(\lambda_n) f_l(\lambda_n) = 0, k \neq l. \tag{5.6}$$

Now, we introduce the three-term recurrence relationship of the Forsythe polynomials [13] to generate a discrete orthogonal polynomial basis for the graph filter design problem over the frequency range $\lambda_n \in [\lambda_{\min}, \lambda_{\max}]$ as

$$f_k(\lambda_n) = \lambda_n f_{k-1}(\lambda_n) - \omega_k f_{k-1}(\lambda_n) - \varphi_k f_{k-2}(\lambda_n), \quad k > 1, \tag{5.7}$$

with initial polynomials

$$f_0(\lambda_n) = 1, \quad f_1(\lambda_n) = \lambda_n - \omega_1,$$

and where the parameters ω_k and φ_k for generating the Forsythe polynomials are expressed as

$$\omega_k = \frac{\langle \lambda_n f_{k-1}(\lambda_n), f_{k-1}(\lambda_n)\rangle}{\langle f_{k-1}(\lambda_n), f_{k-1}(\lambda_n)\rangle}, \varphi_k = \frac{\langle \lambda_n f_{k-1}(\lambda_n), f_{k-2}(\lambda_n)\rangle}{\langle f_{k-2}(\lambda_n), f_{k-2}(\lambda_n)\rangle}. \tag{5.8}$$

Note that $\omega_1 = \frac{1}{N}\sum_{n=1}^{N} \lambda_n$. In the vertex domain, with the graph shift operator $\boldsymbol{S}$, the polynomial generating procedure is formulated as

$$\begin{aligned} &f_0(\boldsymbol{S}) = \boldsymbol{I}, \quad f_1(\boldsymbol{S}) = \boldsymbol{S} - \omega_1 \boldsymbol{I}, \\ &f_k(\boldsymbol{S}) = \boldsymbol{S} f_{k-1}(\boldsymbol{S}) - \omega_k f_{k-1}(\boldsymbol{S}) - \varphi_k f_{k-2}(\boldsymbol{S}), \quad k > 1, \end{aligned} \tag{5.9}$$

where the matrix $f_k(\boldsymbol{S})$ can be computed recursively through the previous polynomials with order $k-1$ and $k-2$.

With the Forsythe orthogonal polynomials, the filter coefficients in (5.4) can be solved as

$$g_k = \frac{\langle f_k(\lambda_n), h(\lambda_n)\rangle}{\langle f_k(\lambda_n), f_k(\lambda_n)\rangle}. \tag{5.10}$$

Similar to the FIR graph filter, after generating the Forsythe polynomials for the particular set of graph frequencies and computing the filter coefficients, the filter response (5.1) can be written in a matrix-vector form as

$$\hat{\boldsymbol{g}} = \boldsymbol{\Psi}_{K+1}\boldsymbol{g}, \tag{5.11}$$

where the entries of the system matrix $\boldsymbol{\Psi}_{K+1}$ are given by the basis functions, i.e., $[\boldsymbol{\Psi}]_{n,k+1} = f_k(\lambda_n)$. Note that the filter coefficients of the FIR-Forsythe can also be computed by minimizing the error between the desired frequency response and the FIR-Forsythe filter response and solving the problem as a least-square solution.

Remark. For the well-studied FIR graph filter, the monomials have the benefit that $\int_{-a}^{a} x^k x^l dx = 0$, for $k + l = odd$. When we consider as shift operator $\boldsymbol{S}$ a modification of $\boldsymbol{L}$ (undirected graph) with eigenvalues inside the range $[-a, a]$, e.g.,$\boldsymbol{S} = \boldsymbol{L}_{\text{n}} - \boldsymbol{I}$ with frequency range $[-1, 1]$, the FIR graph filters theoretically show a better orthogonality than using other shift operators with eigenvalues inside the range $[0, a]$, e.g.,$\boldsymbol{S} = \boldsymbol{L}_{\text{n}}$ with frequency range $[0, 2]$. Meanwhile, it has been studied that Forsythe polynomials may remain of approximately uniform size by computing the orthogonal basis over the interval $[-2, 2]$ [13]. Thus, the basic frequency range $[-2, 2]$ can be utilized with the modified graph shift operator $\boldsymbol{S} = 2 \times (\boldsymbol{L}_{\text{n}} - \boldsymbol{I})$ for undirected graphs.

For directed graphs with complex-valued frequencies, discrete orthogonal polynomials satisfy

$$\left\langle f_k(\lambda_n), f_l^*(\lambda_n) \right\rangle = \sum_{n=1}^{N} f_k(\lambda_n) f_l^*(\lambda_n) = 0, k \neq l, \tag{5.12}$$

where $f_l^*(\lambda_n)$ is the conjugate value of $f_l(\lambda_n)$.

Now, the parameters ω_k, φ_k for the generating procedure with complex frequencies are expressed as

$$\omega_k = \frac{\left\langle \lambda_n f_{k-1}(\lambda_n), f_{k-1}^*(\lambda_n) \right\rangle}{\left\langle f_{k-1}(\lambda_n), f_{k-1}^*(\lambda_n) \right\rangle}, \varphi_k = \frac{\left\langle \lambda_n f_{k-1}(\lambda_n), f_{k-2}^*(\lambda_n) \right\rangle}{\left\langle f_{k-2}(\lambda_n), f_{k-2}^*(\lambda_n) \right\rangle}, \tag{5.13}$$

and thus the filter coefficients g_k can be written as

$$g_k = \frac{\left\langle h(\lambda_n), f_k^*(\lambda_n) \right\rangle}{\left\langle f_k(\lambda_n), f_k^*(\lambda_n) \right\rangle}. \tag{5.14}$$

Algorithm 5.1: FIR-Forsythe.

1	**Input**:	$\hat{\boldsymbol{h}}$, frequency range $\lambda_n \in [\lambda_{\min}, \lambda_{\max}]$,
2		filter order K, number of nodes N
3	**Initialization**:	λ_n, $f_0(\lambda_n)$
4	**Compute**:	while $k < K$
5		solve (5.8) (undirected) or (5.13) (directed)
6		return ω_k, φ_k and compute $f_k(\lambda_n)$
7		solve (5.10) (undirected) or (5.14) (directed)
8		compute and return g_k
9	**Output**:	ω_k, φ_k and g_k

The three-term generating procedure ensures that the first term of the order k orthogonal polynomial $f_k(\lambda_n)$ is λ_n^k. Algorithm 5.1 summarizes the FIR-Forsythe graph filter.

Proposition 5. *Under Property 2 [cf. Chapter 2], the parameters ω_k and φ_k for the generating procedure as well as the FIR-Forsythe filter coefficients g_k are real-valued for directed graphs with conjugate frequencies.*

Proof. For a directed graph with conjugate frequencies $\lambda_n = \lambda_{n'}^*$, the initial polynomial of the three-term Forsythe recursion is $f_0(\lambda_n) = 1$. The next polynomial has the form $f_1(\lambda_n) = \lambda_n - \omega_1$, where ω_1 is real-valued and $\omega_1 = \frac{1}{N}\sum_{n=1}^{N} \lambda_n$. In general, the form of the three-term Forsythe recursion has the property that $f_k(\lambda_n) = f_k^*(\lambda_{n'})$ for $\lambda_n = \lambda_{n'}^*$.

Since $f_k(\lambda_n) f_k^*(\lambda_n)$ is real, we have

$$
\begin{aligned}
& f_k(\lambda_n) f_k^*(\lambda_n) = f_k(\lambda_{n'}) f_k^*(\lambda_{n'}), \\
& \lambda_n f_k(\lambda_n) f_k^*(\lambda_n) + \lambda_{n'} f_k(\lambda_{n'}) f_k^*(\lambda_{n'}) \\
& = 2\mathrm{Re}(\lambda_n f_k(\lambda_n) f_k^*(\lambda_n)), \\
& \lambda_n f_k(\lambda_n) f_{k-1}^*(\lambda_n) + \lambda_{n'} f_k(\lambda_{n'}) f_{k-1}^*(\lambda_{n'}) \\
& = 2\mathrm{Re}(\lambda_n f_k(\lambda_n) f_{k-1}^*(\lambda_n)).
\end{aligned}
$$

Thus, the inner products $\langle \lambda_n f_k(\lambda_n), f_k^*(\lambda_n) \rangle$ and $\langle \lambda_n f_k(\lambda_n), f_{k-1}^*(\lambda_n) \rangle$ are real-valued. Then, the parameters ω_k and φ_k for the generating procedure are real-valued.

Also, for the conjugate frequencies $\lambda_n = \lambda_{n'}^*$, we have conjugate responses $\hat{h}_n = \hat{h}_{n'}^*$. The inner product $\langle h(\lambda_n), f_k^*(\lambda_n)\rangle$ has the property

$$\begin{aligned} &h(\lambda_n) f_k^*(\lambda_n) + h(\lambda_{n'}) f_k^*(\lambda_{n'}) \\ &= 2\mathrm{Re}(h(\lambda_n) f_k^*(\lambda_n)), \end{aligned}$$

and $\langle h(\lambda_n), f_k^*(\lambda_n)\rangle$ is real-valued. Thus, the FIR-Forsythe filter coefficients g_k are real-valued for directed graphs.

□

Remark. We can notice that the first term of the discrete orthogonal polynomial (5.7) is a monomial and the rest of the terms are depending on the generated parameters. Using some specific frequency set (or grid points for the universal design), the generating parameters ω_k and φ_k can be zero. For this instance, the orthogonal Forsythe polynomials become monomials and the FIR-Forsythe is equal to the FIR-LLS, e.g., the universal design with $N = 100$ complex conjugate pairs of points lying in the complex unit disc in Fig. 3.1 (b) [7]. The specific derivations can be found in Appendix C.

5.2.2. FIR-FORSYTHE IMPLEMENTATION.

The Forsythe polynomial generating process can be applied to the shift operator $\boldsymbol{S}$ and the output of the FIR orthogonal graph filter is

$$\boldsymbol{y} = \boldsymbol{G}\boldsymbol{x} = \sum_{k=0}^{K} g_k f_k(\boldsymbol{S})\boldsymbol{x}. \tag{5.15}$$

The term $f_k(\boldsymbol{S})\boldsymbol{x}$ inside $\boldsymbol{G}\boldsymbol{x}$ is computed recursively as

$$f_k(\boldsymbol{S})\boldsymbol{x} = \boldsymbol{S} f_{k-1}(\boldsymbol{S})\boldsymbol{x} - \omega_k f_{k-1}(\boldsymbol{S})\boldsymbol{x} - \varphi_k f_{k-2}(\boldsymbol{S})\boldsymbol{x}, \quad k > 1, \tag{5.16}$$

where the vector $f_k(\boldsymbol{S})\boldsymbol{x}$ can be computed recursively from the previous two terms $f_{k-1}(\boldsymbol{S})\boldsymbol{x}$, $f_{k-2}(\boldsymbol{S})\boldsymbol{x}$.

In conclusion, the computational cost of $\boldsymbol{G}\boldsymbol{x}$ is related to $\boldsymbol{S} f_{k-1}(\boldsymbol{S})\boldsymbol{x}$ leading to an overall complexity of $O(KE)$ [14], which scales linearly with the number of edges E. Meanwhile, those parameters ω_k and φ_k only correspond to the previous orthogonal polynomials $f_{k-1}(\lambda_n)$ and $f_{k-2}(\lambda_n)$ which will be computed in an iterative way.

5.2.3. GENERAL ORTHOGONAL POLYNOMIAL BASIS

We now extend the discrete orthogonal polynomial basis into a more general setup and discuss the related filter design problem.

Orthogonal basis. For the discrete orthogonal polynomial basis, we now add an appropriate weighting function $w(\lambda_n)$ into the orthogonal polynomial (5.5) and make it more general as

$$\left\langle f_k(\lambda_n), f_l(\lambda_n) \right\rangle = \sum_{n=1}^{N} w(\lambda_n) f_k(\lambda_n) f_l(\lambda_n) = 0, k \neq l. \tag{5.17}$$

As an alternative to the general discrete orthogonal polynomial basis, we can also extend it to the general continuous orthogonal basis. Then, the inner product becomes

$$\left\langle f_k(\lambda), f_l(\lambda) \right\rangle = \int_{\lambda_{\min}}^{\lambda_{\max}} w(\lambda) f_k(\lambda) f_l(\lambda) d\lambda = 0, k \neq l, \tag{5.18}$$

where $w(\lambda)$ is a continuous weighting function over the whole frequency range.

Related filter design. In the previous section, we have used the Forsythe polynomial, as an example of a general discrete orthogonal basis (5.17) with the weighting function $w(\lambda_n) = 1$, to design the graph filter based on an FIR filter model.

For the continuous orthogonal basis, when we only consider those undirected graphs with real-valued frequencies, e.g., $\boldsymbol{S} = \boldsymbol{L}$ (or other modifications of $\boldsymbol{L}$), after we transform the range of frequencies into $\lambda \in [-1, 1]$, a special case is the *FIR-Chebyshev graph filter*. This is well studied in [11],[1]. It considers an appropriate weighting function $w(\lambda) = (1-\lambda^2)^{-1/2}$ for the continuous orthogonal basis (5.18).

Remark. When we have infinite grid points (or graph frequencies), the discrete orthogonal case automatically becomes the continuous form. With the three-term recurrence of generating polynomials, the filter design problem is mainly related to the weighting function $w(\cdot)$ in (5.17) and (5.18). Thus, by giving an appropriate weighting function, the filter design problem has more freedom according to different requirements, i.e., the function $w(\cdot)$ can amplify the weighting of specific frequencies (or frequency range).

In the next section, we will show that with an appropriate weighting function $w(\lambda)$, the discrete orthogonal polynomials can be utilized for the ARMA model

and we can generate the orthogonal polynomial basis for the denominator and numerator parts separately. Since a directed graph has conjugate frequencies, the continuous orthogonal polynomial basis (such as the Chebyshev polynomial basis) is not easy to extend to the complex domain. Thus, we only choose the discrete orthogonal polynomial basis for the ARMA model for both directed and undirected graphs.

5.3. ARMA-FORSYTHE

To improve the approximation accuracy and reduce the number of required filter coefficients w.r.t. the FIR filter, we now consider applying an autoregressive moving average (ARMA) model to the graph filter design using orthogonal polynomials. In this section, we first introduce the ARMA model with the discrete orthogonal polynomial basis for both undirected and directed graphs. Then, we formulate the solution for the filter design problem.

5.3.1. ARMA MODEL WITH FORSYTHE POLYNOMIALS

For the rational graph filter in the frequency domain, let us indicate the filter response for every graph frequency λ_n with an ARMA model as

$$g(\lambda_n) = \frac{\beta_n}{\alpha_n} = \frac{\sum_{q=0}^{Q} b_q f_q(\lambda_n)}{\sum_{p=0}^{P} a_p \phi_p(\lambda_n)}, \tag{5.19}$$

where the filter coefficients are a_p, b_q, and $f_q(\lambda_n)$, $\phi_p(\lambda_n)$ represent the two different sets of basis polynomial functions for the numerator and denominator parts. Note that, for the ARMA graph filter of Chapter 4, the basis functions are monomials, i.e., $f_q(\lambda_n) = \lambda_n^q$ and $\phi_p(\lambda_n) = \lambda_n^p$.

The main idea for the ARMA model with orthogonal polynomials is that we compute the orthogonal polynomials for the numerator and denominator parts separately with different weighting functions.

ARMA-Forsythe. Similar to the FIR model, we first formulate the design problem as minimizing the error between the desired frequency response and the filter response for frequency λ_n as

$$e_n = h(\lambda_n) - \frac{\beta_n}{\alpha_n}. \tag{5.20}$$

Let us again start with an undirected graph characterized with real-valued frequencies. Since (5.20) is nonlinear in the filter coefficients a_p and b_q, we consider minimizing the following related (modified) error for all graph frequencies

$$e' = \sum_{n=1}^{N} \{h(\lambda_n)\alpha_n - \beta_n\}^2. \tag{5.21}$$

Similar as before, the derivatives towards the filter coefficients are given separately by

$$\begin{aligned} \frac{\partial e'}{\partial a_r} &= 2 \sum_{n=1}^{N} \{h(\lambda_n)\alpha_n - \beta_n\} \big(h(\lambda_n)\phi_r(\lambda_n)\big) \\ \frac{\partial e'}{\partial b_s} &= 2 \sum_{n=1}^{N} \{h(\lambda_n)\alpha_n - \beta_n\} \big(-f_s(\lambda_n)\big). \end{aligned} \tag{5.22}$$

5

To minimize the error, we have to solve

$$\frac{\partial e'}{\partial a_r} = 0, \frac{\partial e'}{\partial b_s} = 0$$

which leads to

$$\begin{aligned} &\sum_{p=0}^{P} a_p \left\{ \sum_{n=1}^{N} \phi_p(\lambda_n) h^2(\lambda_n) \phi_r(\lambda_n) \right\} \\ &= \sum_{q=0}^{Q} b_q \left\{ \sum_{n=1}^{N} f_q(\lambda_n) h(\lambda_n) \phi_r(\lambda_n) \right\}; \\ &\sum_{q=0}^{Q} b_q \left\{ \sum_{n=1}^{N} f_q(\lambda_n) f_s(\lambda_n) \right\} \\ &= \sum_{p=0}^{P} a_p \left\{ \sum_{n=1}^{N} \phi_p(\lambda_n) h(\lambda_n) f_s(\lambda_n) \right\}. \end{aligned} \tag{5.23}$$

Comparing (5.23) with (5.4), the two different orthogonal bases of the numerator and denominator parts should satisfy

$$\begin{aligned} &\sum_{n=1}^{N} \phi_p(\lambda_n) h^2(\lambda_n) \phi_r(\lambda_n) = 0, \quad p \neq r, \\ &\sum_{n=1}^{N} f_q(\lambda_n) f_s(\lambda_n) = 0, \quad q \neq s. \end{aligned} \tag{5.24}$$

We can notice that for the numerator-related part associated with coefficients b_q, the three-term recurrence to generate the orthogonal polynomials $f_q(\lambda)$ uses (5.7)

with coefficients given by (5.8). That is because we are using the same weighting function $w(\lambda_n) = 1$.

Meanwhile, for the denominator-related part corresponding to the coefficients a_p, we have to set the weighted inner product to zero with $w(\lambda_n) = h(\lambda_n)^2$ for all frequencies λ_n. Thus, the three-term recurrence for the denominator-related part uses the modified parameters

$$\begin{aligned} \omega_p &= \frac{\langle \lambda_n h(\lambda_n)\phi_{p-1}(\lambda_n), h(\lambda_n)\phi_{p-1}(\lambda_n)\rangle}{\langle h(\lambda_n)\phi_{p-1}(\lambda_n), h(\lambda_n)\phi_{p-1}(\lambda_n)\rangle}, \\ \varphi_p &= \frac{\langle \lambda_n h(\lambda_n)\phi_{p-1}(\lambda_n), h(\lambda_n)\phi_{p-2}(\lambda_n)\rangle}{\langle h(\lambda_n)\phi_{p-2}(\lambda_n), h(\lambda_n)\phi_{p-2}(\lambda_n)\rangle}. \end{aligned} \tag{5.25}$$

Taking (5.25) into account, the function $\phi_p(\lambda_n)$ can be calculated using the following generating procedure:

$$\phi_p(\lambda_n) = \lambda_n\phi_{p-1}(\lambda_n) - \omega_p\phi_{p-1}(\lambda_n) - \varphi_p\phi_{p-2}(\lambda_n), \tag{5.26}$$

with initial polynomials

$$\phi_0(\lambda_n) = 1, \quad \phi_1(\lambda_n) = \lambda_n - \omega_1.$$

In the vertex domain, we can use the obtained parameters to formulate the generating procedure of the orthogonal basis functions in the graph shift operator $\boldsymbol{S}$.

Similar to the previous section, for directed graphs with complex frequencies, the polynomial functions inside the orthogonal basis (5.24) should be replaced by the corresponding conjugate value, i.e.,

$$\begin{aligned} &\sum_{n=1}^{N} \phi_p(\lambda_n)h(\lambda_n)\big(h(\lambda_n)\phi_r(\lambda_n)\big)^* = 0, \quad p \neq r, \\ &\sum_{n=1}^{N} f_q(\lambda_n)f_s^*(\lambda_n) = 0, \quad q \neq s. \end{aligned} \tag{5.27}$$

Notice that, for the three-term generating processes of the denominator and numerator parts, the related parameters ω and φ also need to be changed with the corresponding conjugate values in the same way.

Thus, for the numerator-related part, the parameters have the same computational structure as (5.13) for the FIR model, while the parameters ω_p and φ_p of

the denominator-related part (5.25) become

$$\begin{aligned}\omega_p &= \frac{\left\langle \lambda_n h(\lambda_n)\phi_{p-1}(\lambda_n), \left(h(\lambda_n)\phi_{p-1}(\lambda_n)\right)^* \right\rangle}{\left\langle h(\lambda_n)\phi_{p-1}(\lambda_n), \left(h(\lambda_n)\phi_{p-1}(\lambda_n)\right)^* \right\rangle}, \\ \varphi_p &= \frac{\left\langle \lambda_n h(\lambda_n)\phi_{p-1}(\lambda_n), \left(h(\lambda_n)\phi_{p-2}(\lambda_n)\right)^* \right\rangle}{\left\langle h(\lambda_n)\phi_{p-2}(\lambda_n), \left(h(\lambda_n)\phi_{p-2}(\lambda_n)\right)^* \right\rangle}.\end{aligned} \tag{5.28}$$

With the generated parameters, the orthogonal polynomials for the filter design problem on a directed graph can also be calculated by (5.26).

Proposition 6. *Under Property 2, for the ARMA-Forsythe graph filter, the parameters ω and φ resulting from the generating procedure of the denominator and numerator parts are real-valued for a directed graph.*

Proof. Similar to the FIR-Forsythe, the conjugate frequencies $\lambda_n = \lambda_{n'}^*$ for a directed graph have the property that $\hat{h}_n = \hat{h}_{n'}^*$. For the numerator-related part of the ARMA filter, which has the same structure as the FIR model, the parameters ω_q and φ_q are real-valued. See Proposition 5.

Meanwhile, for the denominator-related part, the initial orthogonal polynomial is $\phi_0(\lambda_n) = 1$ and the next one has the form $\phi_1(\lambda_n) = \lambda_n - \omega_1$, where ω_1 is also real-valued and given by

$$\omega_1 = \frac{\sum_{n=1}^{N} \lambda_n h(\lambda_n) h^*(\lambda_n)}{\sum_{n=1}^{N} h(\lambda_n) h^*(\lambda_n)}.$$

In general, we have the property $\phi_p(\lambda_n) = \phi_p^*(\lambda_{n'})$ and

$$\begin{aligned}&\lambda_n h(\lambda_n)\phi_p(\lambda_n)(h(\lambda_n)\phi_p(\lambda_n))^* \\ &+\lambda_{n'} h(\lambda_{n'})\phi_p(\lambda_{n'})(h(\lambda_{n'})\phi_p(\lambda_{n'}))^* \\ &= 2\mathrm{Re}(\lambda_n h(\lambda_n)\phi_p(\lambda_n)(h(\lambda_n)\phi_p(\lambda_n))^*), \\ &\lambda_n h(\lambda_n)\phi_p(\lambda_n)(h(\lambda_n)\phi_{p-1}(\lambda_n))^* \\ &+\lambda_{n'} h(\lambda_{n'})\phi_p(\lambda_{n'})(h(\lambda_{n'})\phi_{p-1}(\lambda_{n'}))^* \\ &= 2\mathrm{Re}(\lambda_n h(\lambda_n)\phi_p(\lambda_n)(h(\lambda_n)\phi_{p-1}(\lambda_n))^*).\end{aligned}$$

Thus, the parameters ω_p and φ_p are real-valued for a directed graph and the three-term generating procedure of the ARMA-Forsythe is suitable for both directed and undirected graphs in the vertex domain. □

Algorithm	**5.2**: ARMA-Forsythe.	
1	**Input**:	$\hat{\boldsymbol{h}}$, frequency range $[\lambda_{\min}, \lambda_{\max}]$,
2		filter orders P, Q, number of nodes N
3	**Initialization**:	λ_n, $f_0(\lambda_n)$, $\phi_0(\lambda_n)$
4	**Compute** :	while $p < P$, $q < Q$
		Numerator-related part:
5		solve (5.8) (undirected) or (5.13) (directed)
6		return ω_q, φ_q and compute $f_q(\lambda_n)$
		Denominator-related part:
7		solve (5.25) (undirected) or (5.28) (directed)
8		return ω_p, φ_p and compute $\phi_p(\lambda_n)$
9	**Return**:	$f_q(\lambda_n)$ and $\phi_p(\lambda_n)$
10		solve (5.30)
11	**Output**:	coefficients $\boldsymbol{a}$ and $\boldsymbol{b}$

5.3.2. SOLUTION FOR THE ARMA-FORSYTHE

Since we compute the basis functions of the numerator and denominator parts separately, the design problem of (5.21) is now linear in the filter coefficients a_p and b_q with known orthogonal polynomials. We write (5.21) into a more convenient matrix-vector form over all frequencies with the vectors $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_N]^{\mathrm{T}}$, and $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_N]^{\mathrm{T}}$. The error vector $\boldsymbol{e}' = [e'_1, \ldots, e'_N]^{\mathrm{T}}$ containing the modified error for all graph frequencies can be written as

$$\boldsymbol{e} = \hat{\boldsymbol{h}} \circ \boldsymbol{\alpha} - \boldsymbol{\beta}. \tag{5.29}$$

Expressing (5.29) as a direct function of the filter coefficients $\boldsymbol{a}$ and $\boldsymbol{b}$, the coefficients can be calculated by the following LLS problem [7],

$$\min_{\boldsymbol{a},\boldsymbol{b}} \left\| [\boldsymbol{\Psi}_{P+1} \circ (\hat{\boldsymbol{h}} \mathbf{1}_{P+1}^{\mathrm{T}}), -\boldsymbol{\Psi}_{Q+1}] \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{b} \end{bmatrix} \right\|^2, \text{ s.t. } a_0 = 1, \tag{5.30}$$

where $\boldsymbol{a} = [a_0, \ldots, a_P]^T$ and $\boldsymbol{b} = [b_0, \ldots, b_Q]^T$ collect the ARMA filter coefficients. The entries inside the system matrices $\boldsymbol{\Psi}_{P+1}$ and $\boldsymbol{\Psi}_{Q+1}$ are the basis functions

(orthogonal polynomials),i.e., $[\boldsymbol{\Psi}_{P+1}]_{n,p+1} = \phi_p(\lambda_n)$, $[\boldsymbol{\Psi}_{Q+1}]_{n,q+1} = f_q(\lambda_n)$. Note that for a directed graph, the matrices $\boldsymbol{\Psi}_{P+1}$ and $\boldsymbol{\Psi}_{Q+1}$ have conjugated rows according to the corresponding conjugate frequencies. Thus, the ARMA-Forsythe filter coefficients are again real-valued as shown in the following Proposition.

Proposition 7. *Under Property 2, the ARMA-Forsythe filter coefficients* $\boldsymbol{a}$ *and* $\boldsymbol{b}$ *obtained by solving* (5.30) *are real-valued.*

Proof. The proof is similar to the proof of Proposition 1. □

For the proposed design method, the design cost mainly relates to the LLS problem (5.30) which requires $O(N(P+Q+1)^2)$ operations, where N is the number of grid points for universal design or the number of nodes. Similar to the ARMA graph filter design of [7], the specific combination of orders P and Q can offer more freedom to the design method. However, some combinations of P and Q yield instabilities and unstable performances. Thus, searching over different combinations of P and Q is recommended. Algorithm 5.2 summarizes the ARMA-Forsythe graph filter for both undirected and directed graphs.

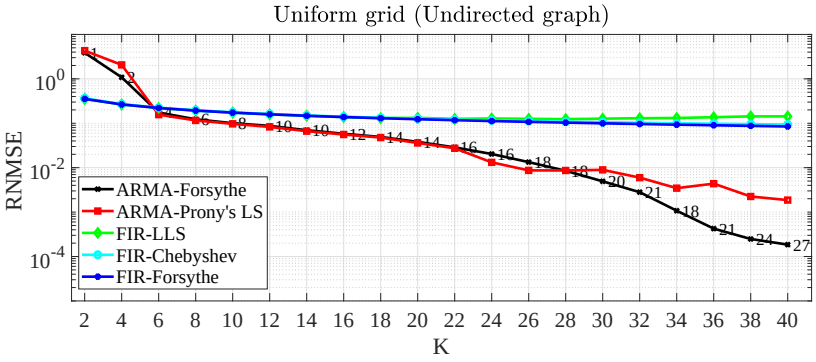
5.4. NUMERICAL DATA

In this section, we present our numerical evaluation for the proposed methods and compare them with other graph filters, i.e., FIR-LLS graph filter, FIR-Chebyshev graph filter, and the ARMA filters from Chapter 4. The performance is evaluated with design methods for both directed and undirected graphs.

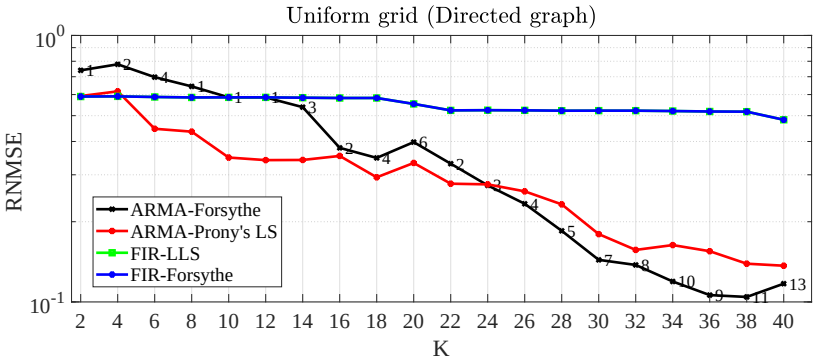
To be specific, we evaluate the proposed design methods for two scenarios: i) universal design which is the same setup as in Chapter 4, ii) design with knowledge of the graph frequencies. Our tests show that the proposed methods have a better numerical condition than the earlier considered FIR and ARMA filter designs. Meanwhile, when we design our filter for a set of known graph frequencies, our method outperforms the FIR-Chebyshev filter in terms of approximation accuracy. Throughout our simulations, we make use of the GSPBox [15].

5.4.1. UNIVERSAL DESIGN

We first evaluate the performance of the proposed design algorithms [cf. Section 5.2 and 5.3] by approximating a desired frequency response in the frequency domain.



(a) Universal design for undirected graph using $N = 100$ grid points.



(b) Universal design for directed graph using $N = 100$ grid points.

Figure 5.1: RNMSE of the proposed design methods for different orders K (such that $P + Q = K$) in approximating an ideal low-pass frequency response following the universal approach. (a) Universal design for undirected graph by gridding the spectrum with $N = 100$ points. (b) Universal design for directed graph with $N = 100$ grid points in the unit disc. For the ARMA-Forsythe graph filters, the order Q is shown in the plot. For the ARMA-Prony's LS graph filter, the design method is related to chapter 4.

Following the universal design [7], we evaluate the orthogonal design methods for both undirected and directed graphs ($N = 100$) for a low pass filter characteristic.

Experimental set up. We consider $\boldsymbol{S} = \boldsymbol{L}_{\text{n}}$ for an undirected graph and sample the interval $[0, 2]$ uniformly with $N = 100$ points. Meanwhile, for the directed case, $\boldsymbol{S} = \boldsymbol{A}_{\text{n}}$ is considered as graph shift operator and we sample the unit complex unit disc uniformly in amplitude and phase with the same number of points,

i.e., $N = 100$. Note that, for a directed graph with complex frequencies, since the filter response can be complex-valued, we only compute the approximation error for the amplitude (absolute value) of the filter response. We measure the approximation accuracy with the root normalized mean square error (RNMSE = $\|\hat{\boldsymbol{g}} - \hat{\boldsymbol{h}}\|/\|\hat{\boldsymbol{h}}\|$) of the frequency response and analyze the design methods using discrete orthogonal polynomials for undirected and directed graphs.

The discrete orthogonal basis is computed for the considered grid points and the filter coefficients are also influenced by those grid points. We compare the set of ARMA-Forsythe(P,Q) filters to the FIR(K) (FIR-Forsythe(K)) graph filters where $P+Q=K$ is satisfied. We look for all combinations of P and Q that satisfy $P+Q=K$ and pick the combination leading to the best results. Since we want the overall order of the designed orthogonal graph filter to be small, we only investigate the range of $2 \leq K \leq 40$ for our comparison.

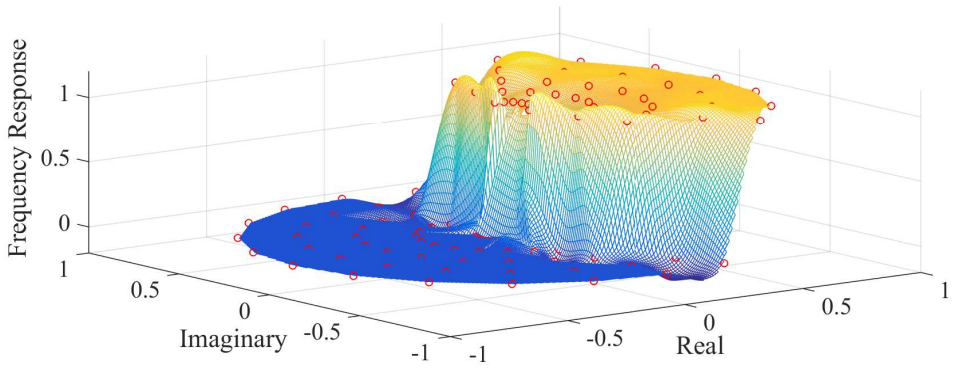


Figure 5.2: The performance of ARMA-Forsythe(23,7) for directed case with $N = 100$ grid points, which is related to the black line in Fig. 5.1(b).

Performance analysis. In Fig. 5.1 we show the RNMSE of the design methods for both the undirected and directed universal case. Specifically, the depicted RNMSE in Fig. 5.1 (a) and (b) are related to the best combination (P,Q) for the ARMA-Forsythe (ARMA-Prony's LS [7]) graph filters. Additionally, the FIR filter performances are plotted as benchmarks. From these results, we can make the following observations:

- i) For the FIR model with discrete orthogonal polynomials (FIR-Forsythe), the approximation accuracy is close to the FIR-LLS for both undirected and

directed graphs. As the filter order increases, the approximation accuracy does not really improve. With discrete orthogonal polynomials, we observe a smaller RNMSE for undirected graphs compared to directed graphs for the same number of grid points N. This is the same situation as in Chapter 4. Details can be found in Section 4.3.1.

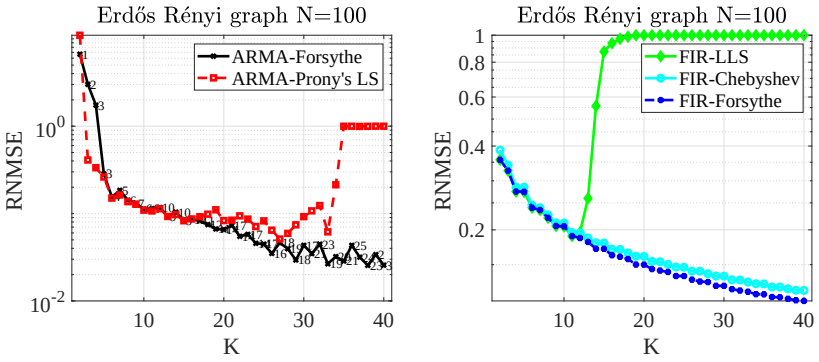
- ii) For small filter orders, i.e., $K \leq 5$, in the undirected case, the FIR model shows better performances than ARMA-Forsyth. However, for a larger filter order K, ARMA-Forsyth is more suitable for filter design and it improves the approximation accuracy for both directed and undirected graphs. As an example, Fig. 5.2 shows the performance of ARMA-Forsythe$(23,7)$ for the directed case with $N = 100$ grid points, which is the best combination of $P+Q = 30$ and related to the black line in Fig. 5.1(b). Compared to ARMA graph filter design [7], the ARMA-Forsyth generally improves the approximation accuracy for undirected and directed graphs when the filter order $K > 24$.

As we mentioned in the previous section, when we have some knowledge of graphs, such as graph frequencies, the Chebyshev polynomial may lose some freedom of design since the filter coefficients are generated only using the value of cutoff frequency. In the next section, we will evaluate the design method with a known graph (design with graph frequency) and give more comparisons with other graph filters.

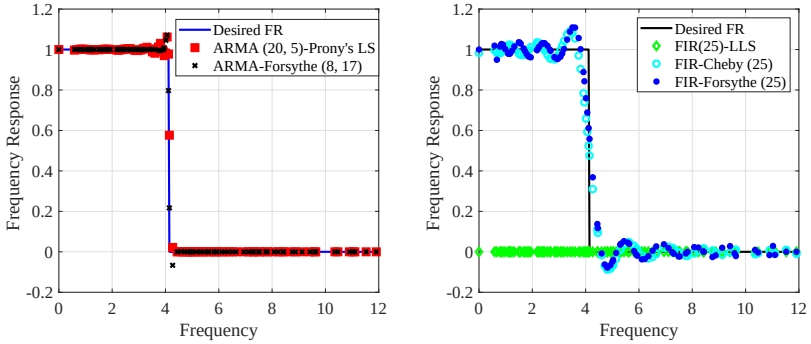
5.4.2. DESIGN WITH KNOWN GRAPH FREQUENCIES

In this section, we evaluate the performance of the proposed design algorithms in approximating a low pass filter for an Erdős Rényi (ER) graph and a directed network graph. To compare this with the earlier universal design, we consider graphs with $N = 100$ nodes.

Experimental set up. For the undirected graph, we consider $\boldsymbol{S} = \boldsymbol{L}$ for an Erdős Rényi graph ($N = 100$) with link probability $p = 0.1$ and frequencies in the interval $[0, 11.91]$. For the directed graph, we consider $\boldsymbol{S} = \boldsymbol{A}$ with $N = 100$ and maximum frequency $\lambda_{\max} = 89.03$. The other details of the set up are the same as in the previous section and we measure the approximation accuracy using the RNMSE.



(a) RNMSE of the proposed design methods for the Erdős Rényi (ER) graph, $N = 100$.

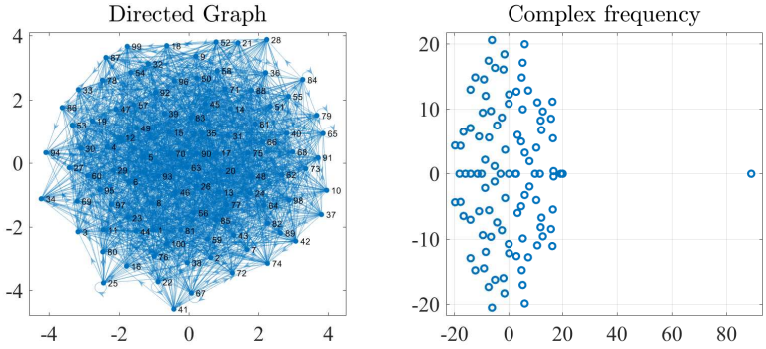


(b) The performance of the filters with order $K = 25$. For ARMA and ARMA-Forsythe graph filters, we have $P + Q = K$.

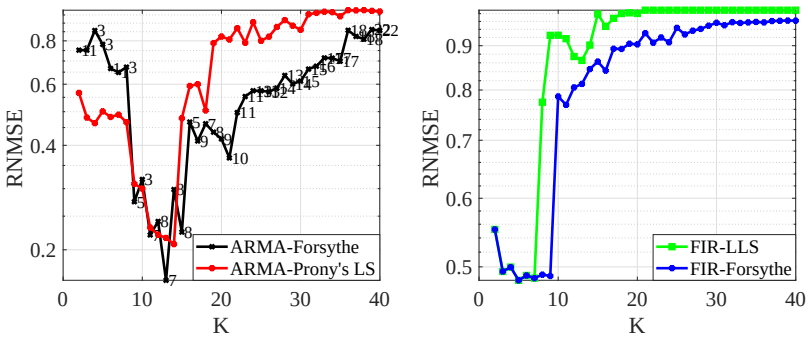
Figure 5.3: RNMSE of the proposed design methods for different orders K (such that $P + Q = K$) in approximating a low-pass frequency response with an Erdős Rényi (ER) graph of $N = 100$ nodes. (a) Results for the proposed graph filter of an Erdős Rényi (ER) graph with $\boldsymbol{S} = \boldsymbol{L}$. For the ARMA-Forsythe graph filters, the order Q is shown in the plot. (b) The performance of the filters with order $K = 25$ for the Erdős Rényi (ER) graph, which are related to Fig. 5.3(a).

Performance analysis. We apply the proposed design methods to the two scenarios and compare them to the related FIR (FIR-Chebyshev) and ARMA graph filters. Based on these results we can make the following observations:

- i) For the Erdős Rényi (ER) graph, we notice from Fig. 5.3 that the FIR-Forsythe graph filter has a better performance than the FIR-Chebyshev graph filter. Although FIR-Chebyshev minimizes some weighted approximation error over a continuous frequency range, the FIR-Forsythe mini-



(a) Directed network graph of $N = 100$ nodes.



(b) RNMSE of the proposed design methods.

Figure 5.4: RNMSE of the proposed design methods for different orders K (such that $P+Q=K$) in approximating an ideal low-pass frequency response with a directed network graph. (a) Directed network graph of $N = 100$ nodes with $E = 2018$ edges having different weights in the interval $[0,8]$. The shift operator is considered as $\boldsymbol{S} = \boldsymbol{A}$ and the maximum frequency is $\lambda_{\max} = 89.03$. (b) Results for the proposed graph filter of the directed network graph. For the ARMA-Forsythe graph filters, the order Q is shown in the plot.

mizes the true error and this over the discrete set of frequencies for that particular graph. Thus, FIR-Forsythe with discrete orthogonal polynomials is more suitable when the graph frequencies are known.

- ii) Since the FIR-Forsythe and FIR-LLS graph filters are the same for the directed universal design, we also test our proposed methods for the directed network graph (Fig. 5.4(a)) with shift operator $\boldsymbol{S} = \boldsymbol{A}$, $\lambda_{\max} = 89.03$ and $N = 100$. Fig. 5.4(b) shows that for the design with known graph fre-

quencies, the performance of FIR-Forsythe is better than FIR-LLS. However, both FIR-Forsythe and FIR-LLS are suggested with smaller filter orders ($K < 10$).

- iii) For the ARMA-Prony's LS and ARMA-Forsythe graph filters, we observe a smaller RNMSE for undirected graphs compared to directed graphs. With smaller orders ($K < 4$ for ER graph and $K < 8$ for directed network graph), the ARMA-Prony's LS graph filter performs better than the ARMA-Forsythe. Meanwhile, with higher orders ($K > 15$ for both ER graph and directed network graph), the ARMA-Forsythe gives a better approximation accuracy.

- iv) For the Erdős Rényi (ER) graph, shown in Fig. 5.3, we want to highlight that the performance of designing a graph filter with known graph frequencies is influenced by the topology of the graph. In other words, even for graphs of the same size, the approximation errors error can differ.

In the next section, we will compare the performance and computational conditions (condition number of the system matrix) of the two scenarios, including the universal design and the design for known graph frequencies.

5.4.3. COMPARISON

As mentioned in Chapters 3 and 4, the Vandermonde matrices for computing the filter coefficients are generally ill-conditioned for the FIR and ARMA graph filter designs. With orthogonal polynomials, the numerical conditions of the system matrices are improved for both the FIR and ARMA model.

Experimental set up. In Table 5.1, we show the condition numbers of the system matrices for FIR and ARMA models with different orders. For the ARMA-Prony's LS and ARMA-Forsythe, we fix the order $P + Q = K$ and average the results over all the combinations P, Q. The other details of the experiment setup for Table 5.1 are the same as in Fig. 5.1(a) and Fig. 5.3. Meanwhile, since the FIR-Forsythe and FIR-LLS graph filters are the same for the directed universal design, the two methods have the same rank and condition number for the system matrices in Table 5.2, which focuses on directed graphs and corresponds to Fig. 5.1 (b). Table 5.2 also shows the computational costs for the design using the directed network graph corresponding to Fig. 5.4.

Performance analysis. We compare the universal design with the design using known graph frequencies. The results are as follows.

- i) Comparing the filter design for a known undirected Erdős Rényi graph (in Fig. 5.3) with the universal design (in Fig. 5.1(a)), the orthogonal polynomial basis (ARMA-Forsythe, FIR-Chebyshve, and FIR-Forsythe) leads to similar trends. However, with higher graph orders, the ARMA and FIR graph filters show some limitations in approximation performance.

- ii) For the directed cases with $N = 100$, shown in Fig. 5.1(b) and Fig. 5.4, all design methods show worse approximation results with higher graph filter orders. Thus, for FIR-LLS and FIR-Forsythe, the filter order can be selected as $K < 10$, while for ARMA-Prony's LS and ARMA-Forsythe, a good choice of filter order is $8 < K < 18$.

- iii) In general, the universal way offers a better computational condition for filter design. Also, the undirected cases (Table 5.1) generally have better condition numbers than the directed cases (Table 5.2). In Table 5.1 and Table 5.2, we notice that FIR-Forsythe filters have smaller condition numbers than FIR-LLS graph filters and ARMA-Forsythe filters have smaller condition numbers than ARMA-Prony's LS graph filters. Thus, the orthogonal polynomial basis offers improvements in numerical computation effects.

- iv) Although the FIR-Chebyshve for undirected graphs has the best numerical condition in all design methods, we want to highlight that the Chebyshev polynomial always moves the frequencies (grid points) into the range $[-1,1]$. For other design methods, we compute the condition numbers and ranks in the range $[0,\lambda_{\max}]$. Thus, to have further improvement of FIR-Forsythe (and ARMA-Forsythe) for undirected graphs, we could move the frequency range to $[-2,2]$ with the shift operator $\boldsymbol{S} = 2 \times (\boldsymbol{A}_{\mathrm{n}} - \boldsymbol{I})$ which is recommended for Forsythe polynomials (mentioned in Section 5.2.1).

We remark that the above results concern the approximation accuracy of the proposed filter without considering the implementation cost and the graph topology. In the following chapters, we will propose another design method (in Chapter 6) and the implementation aspects (in Chapter 7) for these filters.

Table 5.1: Rank/Condition number for undirected case ($N = 100$).

Uniform grid $\boldsymbol{S} = \boldsymbol{L}_{\mathrm{n}}$	$K = 5$	$K = 10$	$K = 15$	$K = 20$	$K = 30$
FIR-LLS	$6/3.69 \times 10^{3}$	$11/4.90 \times 10^{7}$	$16/8.34 \times 10^{11}$	$18/1.62 \times 10^{16}$	$18/7.63 \times 10^{20}$
FIR-Chebyshev [1]	6/3.03	11/5.14	16/9.75	21/23.04	31/297.21
FIR-Forsyth	6/26.19	11/842.94	$16/2.81 \times 10^{4}$	$21/9.73 \times 10^{5}$	$31/1.40 \times 10^{9}$
ARMA-Prony's LS [7]	$7/1.22 \times 10^{3}$	$12/2.36 \times 10^{6}$	$17/1.71 \times 10^{10}$	$22/1.73 \times 10^{14}$	$26/2.73 \times 10^{19}$
ARMA-Forsyth	7/206.75	$12/5.95 \times 10^{4}$	$17/4.19 \times 10^{7}$	$22/4.15 \times 10^{10}$	$31/1.12 \times 10^{17}$
Erdős Rényi (ER) graph $\boldsymbol{S} = \boldsymbol{L}$	$K = 5$	$K = 10$	$K = 15$	$K = 20$	$K = 30$
FIR-LLS	$6/7.18 \times 10^{5}$	$11/1.53 \times 10^{12}$	$8/1.01 \times 10^{19}$	$7/8.36 \times 10^{25}$	$6/1.82 \times 10^{35}$
FIR-Chebyshev [1]	6/4.19	11/4.40	16/12.83	21/81.60	$31/8.93 \times 10^{3}$
FIR-Forsyth	6/263.73	$11/6.28 \times 10^{4}$	$16/9.96 \times 10^{6}$	$21/1.22 \times 10^{9}$	$31/1.65 \times 10^{13}$
ARMA-Prony's LS [7]	$7/6.59 \times 10^{4}$	$12/1.88 \times 10^{10}$	$16/3.50 \times 10^{16}$	$16/1.92 \times 10^{23}$	$12/4.88 \times 10^{34}$
ARMA-Forsyth	7/619.82	$12/5.04 \times 10^{5}$	$17/3.69 \times 10^{8}$	$22/2.64 \times 10^{11}$	$31/2.34 \times 10^{17}$

Table 5.2: Rank/Condition number for directed case ($N = 100$).

Uniform grid $\boldsymbol{S} = \boldsymbol{A}_{\mathrm{n}}$	$K = 5$	$K = 10$	$K = 15$	$K = 20$	$K = 30$
FIR-LLS	6/3.64	11/6.39	16/10.86	21/20.61	31/662.14
FIR-Forsyth	6/3.64	11/6.39	16/10.86	21/20.61	31/662.14
ARMA-Prony's LS [7]	7/32.14	$12/1.91 \times 10^{3}$	$17/2.54 \times 10^{5}$	$22/9.49 \times 10^{7}$	$31/1.72 \times 10^{14}$
ARMA-Forsyth	7/16.87	12/158.43	$17/2.94 \times 10^{3}$	$22/2.18 \times 10^{5}$	$32/3.64 \times 10^{10}$
Directed network graph $\boldsymbol{S} = \boldsymbol{A}$	$K = 5$	$K = 10$	$K = 15$	$K = 20$	$K = 30$
FIR-LLS	$6/5.67 \times 10^{8}$	$7/7.23 \times 10^{18}$	$4/2.12 \times 10^{28}$	$2/5.21 \times 10^{38}$	$1/2.01 \times 10^{58}$
FIR-Forsyth	$6/2.07 \times 10^{7}$	$9/2.68 \times 10^{17}$	$8/9.61 \times 10^{24}$	$9/2.57 \times 10^{35}$	$8/1.20 \times 10^{52}$
ARMA-Prony's LS [7]	$7/1.08 \times 10^{7}$	$10/6.45 \times 10^{16}$	$7/2.01 \times 10^{27}$	$5/2.39 \times 10^{37}$	$2/9.01 \times 10^{56}$
ARMA-Forsyth	$7/8.12 \times 10^{5}$	$12/9.73 \times 10^{14}$	$12/2.22 \times 10^{22}$	$11/9.67 \times 10^{31}$	$9/8.84 \times 10^{47}$

5.5. CONCLUSION

Graph signal processing extends classical digital signal processing to signals that live on the vertices of irregular graphs. Graph filters play a key tool in graph signal processing and shaping the graph signal spectrum. In this chapter, the focus is on designing graph filters using a discrete orthogonal polynomial basis for FIR and ARMA graph filters. The main result of this chapter is that by generating orthogonal polynomials, we can improve the approximation accuracies and numerical conditions of graph filters. The first method adopts orthogonal polynomials for designing FIR graph filters for both undirected and directed graphs. We further introduce an efficient filter design method using an orthogonal basis for the ARMA model for both directed and undirected graphs. Our theoretical findings are supported by numerical results.

REFERENCES

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

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

[3] S. Segarra, A. G. Marques, and A. Ribeiro, *Optimal graph-filter design and applications to distributed linear network operators,* IEEE Transactions on Signal Processing **65**, 4117 (2017).

[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 Signal Processing Magazine **30**, 83 (2013).

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

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

[7] 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).

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

[9] D. P. Bertsekas, *Convex optimization theory* (Athena Scientific Belmont, 2009).

[10] J. R. Shewchuk *et al.*, *An introduction to the conjugate gradient method without the agonizing pain,* (1994).

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

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

[13] G. E. Forsythe, *Generation and use of orthogonal polynomials for data-fitting with a digital computer,* Journal of the Society for Industrial and Applied Mathematics **5**, 74 (1957).

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

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

APPENDIX-C

In the vertex domain, with the graph shift operator $\boldsymbol{S}$, the polynomial generating procedure of Forsythe polynomials is formulated as

$$f_0(\boldsymbol{S}) = \boldsymbol{I}, \quad f_1(\boldsymbol{S}) = \boldsymbol{S} - \omega_1 \boldsymbol{I},$$
$$f_k(\boldsymbol{S}) = \boldsymbol{S} f_{k-1}(\boldsymbol{S}) - \omega_k f_{k-1}(\boldsymbol{S}) - \varphi_k f_{k-2}(\boldsymbol{S}), \quad k > 1.$$

When we consider the directed graph with the normalized adjacency matrix, we have the generating procedure in the frequency domain asc

$$f_k(\lambda_n) = \lambda_n f_{k-1}(\lambda_n) - \omega_k f_{k-1}(\lambda_n) - \varphi_k f_{k-2}(\lambda_n), \quad k > 1$$
$$f_0(\lambda_n) = 1, \quad f_1(\lambda_n) = \lambda_n - \omega_1,$$

where the parameters ω_k, φ_k with complex frequencies are expressed as

$$\omega_k = \frac{\langle \lambda_n f_{k-1}(\lambda_n), f_{k-1}^*(\lambda_n) \rangle}{\langle f_{k-1}(\lambda_n), f_{k-1}^*(\lambda_n) \rangle}, \varphi_k = \frac{\langle \lambda_n f_{k-1}(\lambda_n), f_{k-2}^*(\lambda_n) \rangle}{\langle f_{k-2}(\lambda_n), f_{k-2}^*(\lambda_n) \rangle}.$$

Note that, for complex-valued frequencies, discrete orthogonal polynomials satisfy

$$\langle f_k(\lambda_n), f_l^*(\lambda_n) \rangle = \sum_{n=1}^{N} f_k(\lambda_n) f_l^*(\lambda_n) = 0, k \neq l,$$

where $f_l^*(\lambda_n)$ is the conjugate value of $f_l(\lambda_n)$.

For the discrete points $N = 100$ in Fig. 5.5, we grid the unit disc with even phase. On another word, we have conjugate pairs as $a + bj$, $a - bj$, $-a + bj$ and $-a - bj$. Thus, we have

$$\sum_{n=1}^{N} \lambda_n = 0.$$

Note that the results of sum of grid points depends on different grid methods. For our special case in Fig.1, the $f_0(\lambda_n) = 1$ and $f_1(\lambda_n) = \lambda_n - \omega_1 = \lambda_n$, since

$$\omega_1 = \frac{\langle \lambda_n f_0(\lambda_n), f_0^*(\lambda_n) \rangle}{\langle f_0(\lambda_n), f_0^*(\lambda_n) \rangle} = \frac{1}{N} \sum_{n=1}^{N} \lambda_n = 0.$$

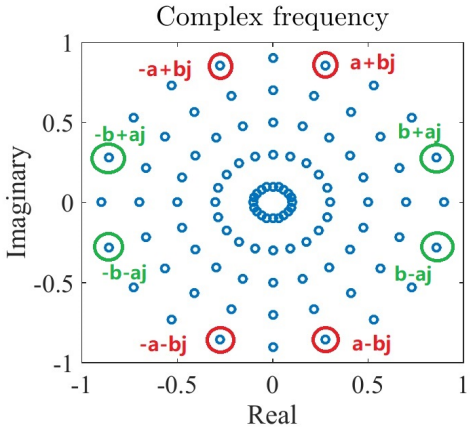


Figure 5.5: Universal grid for directed graph based on the normalized adjacency matrix.

Thus,

$$f_2(\lambda_n) = \lambda_n f_1(\lambda_n) - \omega_2 f_1(\lambda_n) - \varphi_2 f_0(\lambda_n),$$

where

$$\omega_2 = \frac{\langle \lambda_n f_1(\lambda_n), f_1^*(\lambda_n)\rangle}{\langle f_1(\lambda_n), f_1^*(\lambda_n)\rangle}, \varphi_2 = \frac{\langle \lambda_n f_1(\lambda_n), f_0^*(\lambda_n)\rangle}{\langle f_0(\lambda_n), f_0^*(\lambda_n)\rangle}.$$

For the inner product $\langle \lambda_n f_1(\lambda_n), f_1^*(\lambda_n)\rangle$, we have

$$\langle \lambda_n f_1(\lambda_n), f_1^*(\lambda_n)\rangle = \sum_{n=1}^{N} \lambda_n f_1(\lambda_n) f_1^*(\lambda_n) = \sum_{n=1}^{N} \lambda_n \lambda_n \lambda_{n'} = 0.$$

Then, for $\langle \lambda_n f_1(\lambda_n), f_0^*(\lambda_n)\rangle$, we also have

$$\langle \lambda_n f_1(\lambda_n), f_0^*(\lambda_n)\rangle = \sum_{n=1}^{N} \lambda_n f_1(\lambda_n) f_0^*(\lambda_n) = \sum_{n=1}^{N} \lambda_n \lambda_n = 0,$$

since we have $(a+bj)^2 + (-b-aj)^2 = 0$ which is related to the red and green ones inside Fig.1. Thus, the parameters ω_2 and φ_2 are zeros.

In general, as in proposition 5 (in chapter 5), we have

$$\lambda_n f_{k-1}(\lambda_n) f_{k-1}^*(\lambda_n) + \lambda_{n'} f_{k-1}(\lambda_{n'}) f_{k-1}^*(\lambda_{n'}) = 2\mathrm{Re}(\lambda_n f_{k-1}(\lambda_n) f_{k-1}^*(\lambda_n)),$$
$$\lambda_n f_{k-1}(\lambda_n) f_{k-2}^*(\lambda_n) + \lambda_{n'} f_{k-1}(\lambda_{n'}) f_{k-2}^*(\lambda_{n'}) = 2\mathrm{Re}(\lambda_n f_{k-1}(\lambda_n) f_{k-2}^*(\lambda_n)).$$

Consider $\lambda_n = a+bj$, $\lambda_m = -a-bj$, the conjugate pairs λ_n,$\lambda_{n'}$,λ_m and $\lambda_{m'}$ always lead to

$$\langle \lambda_n f_{k-1}(\lambda_n), f^*_{k-1}(\lambda_n) \rangle = 0$$
$$\langle \lambda_n f_{k-1}(\lambda_n), f^*_{k-2}(\lambda_n) \rangle = 0$$

Thus, the ω_k and φ_k are zeros.

With the graph shift operator $\boldsymbol{S}$, the generating procedure of Forsythe polynomials is rewritten as

$$f_0(\boldsymbol{S}) = \boldsymbol{I}, \quad f_1(\boldsymbol{S}) = \boldsymbol{S}, \quad f_k(\boldsymbol{S}) = \boldsymbol{S} f_{k-1}(\boldsymbol{S}), \quad k > 1,$$

which is the same as FIR graph filter.

6

RATIONAL GRAPH FILTER DESIGN USING ITERATIVE VECTOR FITTING

In this chapter a new graph filter design method is presented. Previous works (Chapters 4 and 5) on designing graph filters mainly focus on the (orthogonal) polynomial basis with FIR graph filters and ARMA graph filters. This chapter proposes an iterative framework for designing a special type of ARMA graph filter for both undirected and directed graphs. Our approach formulates the design as a least-squares problem and recursively solves the error between the desired frequency response and the filter response represented by a partial fraction belonging to a rational basis. We demonstrate our proposed method through experimental results and give a comparison with the well-known FIR graph filters. The result shows that the new design is more suitable for applications demanding higher approximation accuracies.

The remainder of this chapter is organized as follows. We first briefly give some background in Section 6.1. Then, we present the choice of the rational basis function and propose the rational graph filter (a specific formulation of the ARMA filter) in Section 6.2. We introduce our iterative approach and formulate

the solution for the rational graph filter in Section 6.3. Finally, experimental results and conclusions are shown in Section 6.4 and Section 6.5.

6.1. INTRODUCTION

FIR graph filters [1, 2], direct analogs of temporal FIR filters, are implemented as polynomials in the graph shift operator, e.g., the Laplacian [3] or adjacency matrix [4]. To accurately match some given specifications, FIR filters require a high filter order leading to a high implementation cost. An alternative to the FIR graph filters are the IIR (ARMA) graph filters [5], [6]. These filters are characterized by a rational polynomial frequency response, which brings more degrees of freedom to the design. However, as the FIR graph filter, the aforementioned ARMA graph filters mentioned in Chapters 4 and 5 still rely on polynomials as the basis functions for the filter response in both the numerator and denominator. In this chapter, we propose a new filter design framework for both undirected and directed graphs. Instead of a polynomial basis, we focus on a partial fraction belonging to a rational basis. Note that the proposed graph filter corresponds to the pole-zero form (3.38) and partial fraction form (3.40) ARMA filters discussed in Chapter 3.

Compared with polynomial basis functions, rational basis functions have a lot of numerical advantages [7–9], i.e., better interpolatory and extrapolatory performances [10]. Our approach is based on formulating the design of this specific ARMA filter formulation as a least-squares problem and solving the error between the desired frequency response and the filter response recursively with the vector fitting method [10–14].

6.2. RATIONAL GRAPH FILTER

In this section, we start with the well-studied FIR graph filter and introduce the selected rational basis. Then, we formulate the rational graph filter and briefly discuss the implementation problem.

As we discussed in previous chapters, an FIR graph filter $\boldsymbol{G}$ of order K can be expressed as a K-th order polynomial in the graph shift operator, i.e., $\boldsymbol{G} = g(\boldsymbol{S}) = g_0\boldsymbol{I} + g_1\boldsymbol{S} + \cdots + g_K\boldsymbol{S}^K$, where K is the filter order. The linear operator $\boldsymbol{G}$ is diagonalizable by $\boldsymbol{U}$ since $\boldsymbol{G} = \boldsymbol{U}g(\boldsymbol{\Lambda})\boldsymbol{U}^{-1}$ and as such it is a valid graph filter. As a result, the relation between the graph frequency response $\hat{g}_n = [g(\boldsymbol{\Lambda})]_{n,n}$ and the filter coefficients g_k is given by

$$\hat{g}_n = \sum_{k=0}^{K} g_k \phi_k(\lambda_n) = \sum_{k=0}^{K} g_k \lambda_n^k, \tag{6.1}$$

where $\phi_k(\lambda)$ is a monomial basis function in the graph frequency λ.

In this chapter, we focus on a rational graph filter, which contains another basis function $\phi_k(\lambda)$ namely a partial fraction belonging to a rational basis. We first formulate the related rational graph filter in the frequency domain and then illustrate the related parallel of the implementation procedure in the graph vertex domain.

Rational basis. Rational basis functions have a lot of numerical advantages [7], e.g., the condition number of a system matrix generated by a rational function basis is smaller than for a polynomial basis. Using a partial fraction as $\phi_k(\lambda)$, i.e.,

$$\phi_k(\lambda_n) = \frac{1}{\lambda + \varphi_k}, \tag{6.2}$$

the graph frequency response can be formulated as

$$\hat{g}_n = \sum_{k=1}^{K} \frac{\omega_k}{\lambda_n + \varphi_k}, \tag{6.3}$$

which is a special case of the partial fraction form ARMA filter introduced in (3.40). The error between the desired frequency response and the filter response becomes

$$e_n = \hat{h}_n - \hat{g}_n = \hat{h}_n - \sum_{k=1}^{K} \frac{\omega_k}{\lambda_n + \varphi_k}, \tag{6.4}$$

where ω_k are the filter coefficients (residues), and $-\varphi_k$ yields a set of prescribed poles for filter response $\hat{g}_n$.

Note that (6.4) constitutes a nonlinear problem in terms of the unknown poles $-\varphi_k$. For this nonlinear problem, we will adapt the *vector fitting* method [13] to graph filter design and solve (6.4) as a linear problem in the next section. But first, we will focus on the implementation of the proposed filter form.

Parallel filter implementation. As we mentioned in Chapter 3, the partial fraction form of an ARMA graph filter leads to a parallel implementation of subfilters. Based on the filter coefficients ω_k and poles $-\varphi_k$, we can express the rational graph filter $\boldsymbol{G}$ in the vertex domain using the graph shift operator $\boldsymbol{S}$ as

$$\boldsymbol{G} = \sum_{k=1}^{K} \omega_k (\boldsymbol{S} + \varphi_k \boldsymbol{I})^{-1}. \tag{6.5}$$

It is clear that in the vertex domain, the relation between the output $\boldsymbol{y}$ and the input $\boldsymbol{x}$ of a rational graph filter is given by

$$\boldsymbol{y} = \sum_{k=1}^{K} \boldsymbol{y}_k = \sum_{k=1}^{K} \omega_k (\boldsymbol{S} + \varphi_k \boldsymbol{I})^{-1} \boldsymbol{x}. \tag{6.6}$$

Note that the partial fraction decomposition of the filter $\boldsymbol{G}$ yields a set of parallel rational graph filters with output $\boldsymbol{y}_k$. By summing all K parallel sub-filters, the filter output can be written as

$$\boldsymbol{y} = \sum_{k=1}^{K} \boldsymbol{y}_k, \quad \text{where} \quad \boldsymbol{y}_k = \omega_k (\boldsymbol{S} + \varphi_k \boldsymbol{I})^{-1} \boldsymbol{x}. \tag{6.7}$$

The filter output $\boldsymbol{y}_k$ can be computed as a linear system and every sub-filter can be expressed in the compact form

$$(\boldsymbol{S} + \varphi_k \boldsymbol{I}) \boldsymbol{y}_k = \omega_k \boldsymbol{x}. \tag{6.8}$$

Note that there are several methods to solve the parallel linear system efficiently and reduce the computational cost of computing the matrix inverse, like the first-order methods [15], conjugate gradient (CG) [16], and some distributed iterative methods [17–23].

6.3. RATIONAL FILTER DESIGN

In this section, we focus on the design of the rational graph filter. We introduce the *vector fitting* method [13] to compute the coefficients and poles of the involved rational filters.

6.3.1. VECTOR FITTING

In order to explain the vector fitting method, we first rewrite the original error (6.4) of the design problem as

$$e_n = \hat{h}_n - \hat{g}_n = \hat{h}_n - \frac{\prod_{k=1}^{K-1} (\lambda_n + a_k)}{\prod_{k=1}^{K} (\lambda_n + \varphi_k)}, \tag{6.9}$$

where $-a_k$ are the zeros of $\hat{g}_n$. Note that the rewritten form (6.9) of $\hat{g}_n$ is also a special case of the pole-zero form (3.38) of an ARMA graph filter. In general, the

pole-zero form and partial fraction expansion of ARMA graph filters can be easily related to each other.

The main idea of vector fitting is that we start from a set of prescribed poles $-\varphi_k$, and multiply the desired frequency response $\hat{h}_n$ with a function

$$\alpha_n = 1 + \sum_{k=1}^{K} \frac{\tilde{\omega}_k}{\lambda_n + \varphi_k} = \frac{\prod_{k=1}^{K} (\lambda_n + \tilde{a}_k)}{\prod_{k=1}^{K} (\lambda_n + \varphi_k)}, \tag{6.10}$$

where $-\tilde{a}_k$ are the zeros of α_n and $\tilde{\omega}_k$ are the additional filter coefficients (residues). We can notice that the function α_n contains the same poles as $\hat{g}_n$. Then, we try to fit $\alpha_n \hat{h}_n$ to $\hat{g}_n$ with the new error

$$e'_n = \alpha_n \hat{h}_n - \hat{g}_n. \tag{6.11}$$

Taking $\hat{g}_n$ of (6.4) (or (6.9)) and α_n of (6.10) into e'_n, the new error can be expressed as

$$e'_n = \left(1 + \sum_{k=1}^{K} \frac{\tilde{\omega}_k}{\lambda_n + \varphi_k}\right) \hat{h}_n - \sum_{k=1}^{K} \frac{\omega_k}{\lambda_n + \varphi_k}. \tag{6.12}$$

Note that with the known set of poles $-\varphi_k$, (6.12) is linear in the filter coefficients ω_k and $\tilde{\omega}_k$.

Minimizing the error e'_n will bring the desired frequency response $\hat{h}_n$ close to $\hat{g}_n/\alpha_n$. Hence, coming back to the expression of the original error, we can modify e'_n as

$$e''_n = \hat{h}_n - \frac{\hat{g}_n}{\alpha_n} = \hat{h}_n - \frac{\prod_{k=1}^{K-1} (\lambda_n + a_k)}{\prod_{k=1}^{K} (\lambda_n + \tilde{a}_k)}. \tag{6.13}$$

Note from (6.13) that the poles of the desired frequency response $\hat{h}_n$ are the zeros of α_n (i.e., $-\tilde{a}_k$). Thus, using those zeros $-\tilde{a}_k$ as the new set of appropriate poles, we can solve the original problem (6.4) (or (6.9)) by solving e''_n. And with the new poles, the design problem (6.4) becomes linear in the coefficients ω_k.

We summarize the vector fitting method for the rational graph filter design problem by the following steps:

1. Given an initial set of prescribed poles, we first modify the error e_n to e'_n and e''_n.

2. Start with the modified error e''_n and solve it for the filter coefficients ω_k and $\tilde{\omega}_k$. This algorithm is explained in more detail in Section 6.3.2

3. With the filter coefficients $\tilde{\omega}_k$, the next step of the vector fitting consists of relocating the set of poles (zeros of α_n in e'_n or e''_n). The details will be demonstrated in Section 6.3.3

4. After updating the new set of poles, we solve the original problem (6.4) in Section 6.3.4 to get more appropriate coefficients for the rational graph filter.

6.3.2. ITERATIVE APPROACH

In this section, we first reformulate the design problem to make it suitable for the iterative approach and then use a variant of the Sanathanan–Koerner method [9, 24] to implement an iterative algorithm.

Problem formulation. We start with (6.13) as the error to minimize for every graph frequency λ_n, i.e.,

$$e''_n = \hat{h}_n - \frac{\hat{g}_n}{\alpha_n} = \frac{\sum_{k=1}^{K} \frac{\omega_k}{\lambda_n+\varphi_k}}{1+\sum_{k=1}^{K} \frac{\tilde{\omega}_k}{\lambda_n+\varphi_k}}. \tag{6.14}$$

Then, by defining $\gamma_n = 1/\alpha_n$, we have $e''_n = \hat{h}_n - \gamma_n \hat{g}_n$, which can be equivalently expressed as

$$e''_n = \gamma_n(\hat{h}_n\alpha_n - \hat{g}_n). \tag{6.15}$$

Iterative algorithm. Let $\alpha_n^{(i)}$ and $\hat{g}_n^{(i)}$, respectively, denote the estimates of α_n and $\hat{g}_n$ at the i-th iteration. We can then find the value of $\gamma_n^{(i)}$, as $\gamma_n^{(i)} = 1/\alpha_n^{(i)}$. Next, the updated estimates $\alpha_n^{(i+1)}$ and $\hat{g}_n^{(i+1)}$ are found by minimizing the updated error

$$e''^{(i+1)}_n = \gamma_n^{(i)}(\hat{h}_n\alpha_n - \hat{g}_n), \tag{6.16}$$

which is linear in the unknown variables α_n and $\hat{g}_n$ or equivalently linear in the filter coefficients ω_k and $\tilde{\omega}_k$. With the obtained estimates of $\alpha_n^{(i+1)}$ and $\hat{g}_n^{(i+1)}$

from (6.16), the new filter coefficients $\omega_k^{(i+1)}$ and $\tilde{\omega}_k^{(i+1)}$ are obtained as well as $\gamma_n^{(i+1)}$ and the procedure is then repeated for some iterations.

To explain this iterative algorithm in detail, we bring the rational basis formulation of α_n and $\hat{g}_n$ into (4.16), and express the error as

$$e''^{(i+1)}_n = \frac{\prod_{k=1}^{K}(\lambda_n+\varphi_k)}{\prod_{k=1}^{K}(\lambda_n+\tilde{a}_k^{(i)})}\left(\hat{h}_n\frac{\prod_{k=1}^{K}(\lambda_n+\tilde{a}_k)}{\prod_{k=1}^{K}(\lambda_n+\varphi_k)}-\frac{\prod_{k=1}^{K-1}(\lambda_n+a_k)}{\prod_{k=1}^{K}(\lambda_n+\varphi_k)}\right) \tag{6.17}$$

where $-\varphi_k$ are the set of prescribed poles and $-\tilde{a}_k^{(i)}$ are the set of zeros of $\alpha_n^{(i)}$ (or poles of $\gamma_n^{(i)}$).

Since the prescribed poles $-\varphi_k$ remain unchanged during the procedure, (6.17) can be simplified as

$$e''^{(i+1)}_n = \hat{h}_n\frac{\prod_{k=1}^{K}(\lambda_n+\tilde{a}_k)}{\prod_{k=1}^{K}(\lambda_n+\tilde{a}_k^{(i)})}-\frac{\prod_{k=1}^{K-1}(\lambda_n+a_k)}{\prod_{k=1}^{K}(\lambda_n+\tilde{a}_k^{(i)})}. \tag{6.18}$$

Again, this reduces to solving the following LS problem

$$e''^{(i+1)}_n = \left(1+\sum_{k=1}^{K}\frac{\tilde{\omega}_k}{\lambda_n+\tilde{a}_k^{(i)}}\right)\hat{h}_n-\sum_{k=1}^{K}\frac{\omega_k}{\lambda_n+\tilde{a}_k^{(i)}}. \tag{6.19}$$

Comparing (6.19) with (6.12), the main idea of the iterative approach is to replace the prescribed poles $-\varphi_k$ with the new poles $-\tilde{a}_k^{(i)}$ (zeros of $\alpha_n^{(i)}$) and to calculate the filter coefficients $\tilde{\omega}_k$ and ω_k during the iterative procedure.

Iterative solution. We express the error of the iterative approach as a direct function of $\tilde{\boldsymbol{\omega}} = [1, \tilde{\omega}_1, \cdots, \tilde{\omega}_k]^{\mathrm{T}}$ and $\boldsymbol{\omega} = [\omega_1, \cdots, \omega_k]^{\mathrm{T}}$ which are vectors stacking the filter coefficients, and write (6.19) as

$$\boldsymbol{e}''^{(i+1)} = \hat{\boldsymbol{h}} \circ (\boldsymbol{\Phi}_{K+1}^{(i)}\tilde{\boldsymbol{\omega}}) - (\boldsymbol{\Phi}_K^{(i)}\boldsymbol{\omega}) \tag{6.20}$$

where $\boldsymbol{\Phi}_{K+1}^{(i)} = [\mathbf{1}\ \boldsymbol{\Phi}_K^{(i)}]$, with $\boldsymbol{\Phi}_K^{(i)}$ the $N \times K$ Cauchy matrix with entries $[\boldsymbol{\Phi}_K^{(i)}]_{n,k} = 1/(\lambda_n + \tilde{a}_k^{(i)})$. The vector $\boldsymbol{e}''^{(i+1)} = [e''^{(i+1)}_1, \cdots, e''^{(i+1)}_N]^{\mathrm{T}}$ contains the errors for all graph frequencies and "$\circ$" represents the element-wise Hadamard product.

Minimizing $\|\boldsymbol{e}''^{(i+1)}\|^2$ over $\tilde{\boldsymbol{\omega}}$ and $\boldsymbol{\omega}$ leads to the following LLS problem

$$\min_{\tilde{\boldsymbol{\omega}},\boldsymbol{\omega}} \left\| [\boldsymbol{\Phi}_{K+1}^{(i)} \circ (\hat{\boldsymbol{h}}\mathbf{1}_{K+1}^{\mathrm{T}}), -\boldsymbol{\Phi}_{K}^{(i)}] \begin{bmatrix} \tilde{\boldsymbol{\omega}} \\ \boldsymbol{\omega} \end{bmatrix} \right\|^2, \text{s.t. } [\tilde{\boldsymbol{\omega}}]_1 = 1, \tag{6.21}$$

where $\mathbf{1}_{K+1}$ is the $(K+1)\times 1$ all-one vector.

Note that for $\boldsymbol{\Phi}_{K+1}^{(0)}$, with the set of prescribed poles $-\varphi_k$, the iterative approach leads to the LS problem based on (6.12) and can be considered as an initialization of the iterative method. The design cost of the iterative approach is related to the least-square problem (6.21) which requires $O((2K)^2 N)$ operations per iteration.

6.3.3. POLE RELOCATION

In this section, we introduce the pole relocation which is the third step of the iterative vector fitting method. We first describe the choice of prescribed poles for initializing the iterative approach and then formulate the procedure of pole relocation.

Prescribed poles. We consider conjugate pairs $-\varphi_k = -\varphi_{k'}^*$ as the prescribed poles [25] for an even filter order K, while for an odd filter order K, we consider an additional constant real pole close to zero. For $-\varphi_k = -\varphi_{k'}^*$, when the desired frequency response is real, the structure of the rational basis function $1/(\lambda_n + \varphi_k)$ automatically generates a conjugate pair of coefficients $\omega_k = \omega_{k'}^*$ or $\tilde{\omega}_k = \tilde{\omega}_{k'}^*$ for both real-valued (undirected graphs) and complex (directed graphs) frequencies λ_n. As we mentioned in Section 6.3.1, we use the zeros of the denominator part as the new set of poles. Hence, for the iterative algorithm, the new set of poles for $\hat{h}_n$ in (6.19) now are the zeros of $\gamma_n^{(i)}\alpha_n$.

Pole relocation. For calculating the poles in the $(i+1)$th iteration, we first rewrite $\gamma_n^{(i)}\alpha_n$ in (6.19) in a matrix-vector form

$$1 + \sum_{k=1}^{K} \frac{\tilde{\omega}_k}{\lambda_n + \tilde{a}_k^{(i)}} = 1 + \tilde{\boldsymbol{\omega}}^{\mathrm{T}}(\lambda_n \boldsymbol{I} - \boldsymbol{\varphi}^{(i)})^{-1}\boldsymbol{b}, \tag{6.22}$$

where $\boldsymbol{\varphi}^{(i)}$ is a diagonal matrix containing the set of poles $\{-\tilde{a}_k^{(i)}\}$ as the corresponding diagonal elements, $\boldsymbol{b}$ is a column vector with constant values 1 and $\tilde{\boldsymbol{\omega}}$ contains the filter coefficients calculated by (6.21).

With the poles $\tilde{a}_k^{(i)}$ and coefficients $\tilde{\boldsymbol{\omega}}$, the zeros of (6.22) can be solved by calculating the eigenvalues of the matrix $\boldsymbol{\varphi}^{(i)} - \boldsymbol{b}\tilde{\boldsymbol{\omega}}^{\mathrm{T}}$ [26, 27]. Thus, the set of new poles $\{-\tilde{a}_k\}$ for the $(i+1)$-th iteration are

$$\{-\tilde{a}_k^{(i+1)}\} = eig(\boldsymbol{\varphi}^{(i)} - \boldsymbol{b}\tilde{\boldsymbol{\omega}}^{\mathrm{T}}), \tag{6.23}$$

where the design cost for the eigenvalue decomposition is $O(N^3)$ operations.

For the case of conjugate poles, in order to force every element inside (6.23) to be real-valued, the corresponding diagonal entries (k and k') in the matrix $\boldsymbol{\varphi}^{(i)}$ can be expressed as $\boldsymbol{\varphi}_k^{(i)}$ which has the modified form

$$\boldsymbol{\varphi}_k^{(i)} = \begin{bmatrix} \mathrm{Re}(-\tilde{a}_k^{(i)}) & \mathrm{Im}(-\tilde{a}_k^{(i)}) \\ -\mathrm{Im}(-\tilde{a}_k^{(i)}) & \mathrm{Re}(-\tilde{a}_k^{(i)}) \end{bmatrix}.$$

Meanwhile, the corresponding elements in the vectors $\boldsymbol{b}$ and $\tilde{\boldsymbol{\omega}}^{\mathrm{T}}$ are

$$\boldsymbol{b}_k = [2,\, 0]^{\mathrm{T}},\ \tilde{\boldsymbol{\omega}}_k^{\mathrm{T}} = [\mathrm{Re}(\tilde{\omega}_k),\, \mathrm{Im}(\tilde{\omega}_k)].$$

This modification for (6.23) has the advantage that matrix $\boldsymbol{\varphi}^{(i)}$ becomes a real matrix and thus its complex eigenvalues are automatically complex conjugate pairs [7, 27] for the next iteration of the vector fitting approach. The eigenvalue decomposition offers suitable poles for completing the iterative approach.

Remark. Since every element in the matrix $\boldsymbol{\varphi}^{(i)}$ is real-valued, the eigenvalues (new poles) can appear as real and complex conjugate values $-\tilde{a}_k^{(i+1)}$. For undirected graphs with real frequencies, the corresponding filter coefficients are automatically real and complex conjugate values. Meanwhile, since some frequencies for directed graphs appear as conjugate pairs ($\lambda_n = \lambda_{n'}^*$), with the real poles, we have

$$\frac{1}{\lambda_n + \tilde{a}_k^{(i+1)}} = \left(\frac{1}{\lambda_{n'} + \tilde{a}_k^{(i+1)}}\right)^*.$$

For λ_n and $\lambda_{n'}$, the desired frequency response is the same and the corresponding coefficients ω_k and $\tilde{\omega}_k$ are thus real-valued.

6.3.4. FILTER COEFFICIENTS

After relocating the poles, calculating the residues becomes a linear problem with those identified poles [13]. For instance, the error can be simplified and reduced to the original error (6.4).

Note that, if the iterative approach converges, the filter coefficients $\boldsymbol{\omega}$ are directly solved by (6.21) during the iteration. Without convergence, the filter coefficients are computed after pole relocation as

$$e_n^{(i+1)} = \hat{h}_n - \hat{g}_n = \hat{h}_n - \sum_{k=1}^{K} \frac{\bar{\omega}_k}{\lambda_n + \tilde{a}_k^{(i+1)}}, \tag{6.24}$$

where $\bar{\omega}_k$ are the new filter coefficients corresponding to the new poles. In matrix-vector form, (6.24) becomes

$$\boldsymbol{e}^{(i+1)} = \hat{\boldsymbol{h}} - \boldsymbol{\Phi}_K^{(i+1)} \bar{\boldsymbol{\omega}}, \tag{6.25}$$

where $\bar{\boldsymbol{\omega}} = [\bar{\omega}_1, \cdots, \bar{\omega}_k]^{\mathrm{T}}$. The coefficients of the graph filter can then be found as

$$\bar{\boldsymbol{\omega}} = \boldsymbol{\Phi}_K^{(i+1)\dagger} \hat{\boldsymbol{h}}, \tag{6.26}$$

where the additional design cost is $O(K^2 N)$.

Note that the convergence of the iterative approach is sensitive to the initial poles and the specific order K [9], and the pole relocation performance is influenced by the initialization (prescribed poles). We define the breaking point of the algorithm using δ, representing the difference between the error in two successive iterations. For those filter orders without convergence, we use a maximum number of iterations and we always use the iteration leading to the minimal error as the final solution.

6.4. EXPERIMENTAL RESULTS

In this section, we will illustrate the performance of our proposed rational graph filter design method.

Universal design example.

For our simulation, we follow the *universal design* [16] for both directed and undirected graphs and the goal is to approximate an ideal low-pass frequency response. We measure the approximation accuracy with the root of the normalized mean squared error (RNMSE) as used in previous chapters. Note that for directed graphs with complex frequencies, we only compute the approximation error for the amplitude (absolute value) of the filter response. We use the FIR filter design as well as the iterative ARMA approach [16] (design method presented in Chapter 4) as benchmarks for the proposed method.

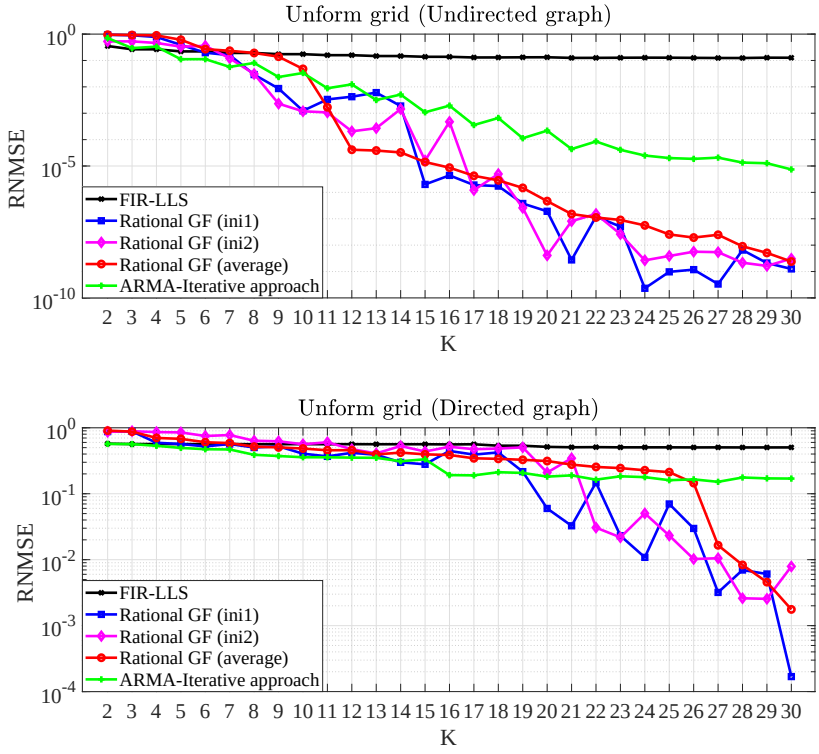


Figure 6.1: RNMSE of the proposed design method and the FIR filter design for different filter orders K in approximating an ideal low-pass frequency response. Universal design using $N = 100$ grid points for (a) an undirected graph and (b) a directed graph.

In Fig. 6.1, we show the RNMSE of the iterative design method for two different sets of initial poles, and the FIR performance is plotted as a benchmark. Specifically, for the rational basis, the depicted RNMSE is related to (6.26) for each particular filter order K. The two sets of initial poles are given as $-\varphi_k = c + di$, $-\varphi_{k'} = c - di$, where for one set we use $d = c/10$, and for the other $d = 0$, and where the parameter c is always uniformly distributed over the range $(0, 1)$. Since the best performance depends on the initial values of the poles [9], we also give the average RNMSE for 100 realizations using different initializations ($c \in [0, 1], d = c/10$).

Although the convergence and performance depend on the initial poles, our design method still shows robustness. The RNMSE of the FIR and ARMA graph

filters are higher for the undirected graph. For the directed case, the ARMA-iterative approach [16] shows a better performance for smaller orders ($K < 20$), while the vector fitting method gives lower errors for high filter order ($K > 20$). It is important to highlight that the iterative design for graph filters is more suitable for applications demanding higher approximation accuracies. Moreover, comparing the universal design for undirected graphs in Fig. 6.1 (a) with the method for directed graphs in Fig. 6.1 (b), the former shows better performance and higher approximation accuracy.

6.5. CONCLUSION

This chapter proposes an iterative framework for rational graph filter design for both undirected and directed graphs. Our iterative approach recursively solves the error between the desired frequency response and the filter response represented by a rational transfer function. Experimental results show that our algorithm can improve the performance of well-known graph filters.

REFERENCES

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

[2] A. Sandryhaila and J. M. Moura, *Discrete signal processing on graphs: Graph filters.* in *ICASSP* (2013) pp. 6163–6166.

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

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

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

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

[7] D. Deschrijver, B. Haegeman, and T. Dhaene, *Orthonormal vector fitting: A robust macromodeling tool for rational approximation of frequency domain responses,* IEEE Transactions on advanced packaging **30**, 216 (2007).

[8] D. Deschrijver, B. Gustavsen, and T. Dhaene, *Advancements in iterative methods for rational approximation in the frequency domain,* IEEE Transactions on Power Delivery **22**, 1633 (2007).

[9] S. Grivet-Talocia and B. Gustavsen, *Passive macromodeling: Theory and applications*, Vol. 239 (John Wiley & Sons, 2015).

[10] L. N. Trefethen, *Approximation theory and approximation practice* (Siam, 2013).

[11] T. Dhaene and D. Deschrijver, *Stable parametric macromodeling using a recursive implementation of the vector fitting algorithm,* IEEE Microwave and Wireless Components Letters **19**, 59 (2009).

[12] A. Chinea and S. Grivet-Talocia, *A parallel vector fitting implementation for fast macromodeling of highly complex interconnects,* in *19th Topical Meeting on Electrical Performance of Electronic Packaging and Systems* (IEEE, 2010) pp. 101–104.

[13] B. Gustavsen and A. Semlyen, *Rational approximation of frequency domain responses by vector fitting,* IEEE Transactions on power delivery **14**, 1052 (1999).

[14] B. Gustavsen, *Improving the pole relocating properties of vector fitting,* IEEE Transactions on Power Delivery **21**, 1587 (2006).

[15] D. P. Bertsekas, *Convex optimization theory* (Athena Scientific Belmont, 2009).

[16] 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).

[17] L. F. Richardson, *Ix. the approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam,* Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character **210**, 307 (1911).

[18] G. Opfer and G. Schober, *Richardson's iteration for nonsymmetric matrices,* Linear algebra and its applications **58**, 343 (1984).

[19] Y. Saad, *Iterative methods for sparse linear systems*, Vol. 82 (siam, 2003).

[20] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: numerical methods*, Vol. 23 (Prentice hall Englewood Cliffs, NJ, 1989).

[21] L. Lei, *Convergence of asynchronous iteration with arbitrary splitting form,* Linear Algebra and its Applications **113**, 119 (1989).

[22] J. M. Bull and T. Freeman, *Numerical performance of an asynchronous jacobi iteration,* in *Parallel Processing: CONPAR 92—VAPP V* (Springer, 1992) pp. 361–366.

[23] G. M. Baudet, *Asynchronous iterative methods for multiprocessors,* Journal of the ACM (JACM) **25**, 226 (1978).

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

[25] W. Hendrickx, D. Deschrijver, and T. Dhaene, *Some remarks on the vector fitting iteration,* Mathematics in Industry **8**, 134 (2006).

[26] A. J. Laub and B. Moore, *Calculation of transmission zeros using qz techniques,* Automatica **14**, 557 (1978).

[27] L. Zhang, Q. Li, W. Wang, and W. H. Siew, *A new algorithm to identify transfer functions of antennas used in emc measurement,* in *2009 Asia-Pacific Power and Energy Engineering Conference* (IEEE, 2009) pp. 1–4.

7

IMPLEMENTATION OF ARMA GRAPH FILTERS

SINCE the design methods proposed in this thesis mainly focus on the frequency domain, this chapter introduces some particular implementations for our graph filters in the vertex domain. We first formulate the filters as a linear system, and then separate the methods into two categories, e.g. centralized [1–5] and distributed [6–11] implementations. For the centralized implementation, we use conjugate gradient and biconjugate gradient methods for respectively undirected and directed graph filters. Although these centralized iterative methods are efficient, they do not allow for easy distribution. Thus, we also propose the Richardson and weighted Jacobi iterations as distributed implementations for both undirected and directed graph filters. Besides, we formulate the convergence conditions for these methods. The simulation results compare the different implementation methods and show the performance for a specific application, namely graph signal denoising.

Part of this chapter has been published in the IEEE Transactions on Signal and Information Processing over Network [1] (2019).

The remainder of this chapter is organized as follows. Section 7.1 briefly introduces the whole chapter and reviews the linear system related to ARMA graph filters. Then, Section 7.2 considers the concept of a centralized implementation for both directed and undirected graph filters. Section 7.3 proposes some distributed iterative methods. In the end, Section 7.4 concludes the chapter.

7.1. INTRODUCTION

As we mentioned in Chapter 2, the *graph filter* (GF) can be represented as the operator $\boldsymbol{G} = g(\boldsymbol{S})$ in the vertex domain. For the implementation problem, it is clear that the relation between the output $\boldsymbol{y}$ and the input $\boldsymbol{x}$ of a graph filter with an ARMA model is given by

$$\boldsymbol{y} = \boldsymbol{G}\boldsymbol{x} \tag{7.1}$$

which can be written as

$$\left(\sum_{p=0}^{P} a_p \phi_p(\boldsymbol{S})\right)\boldsymbol{y} = \left(\sum_{q=0}^{Q} b_q f_q(\boldsymbol{S})\right)\boldsymbol{x}. \tag{7.2}$$

where $\phi_p(\bullet)$ and $f_q(\bullet)$ are the basis functions.

Note that for the ARMA filter in Chapter 4, the basis functions are monomial polynomials as $\phi_p(\boldsymbol{S}) = \boldsymbol{S}^p$, $f_q(\boldsymbol{S}) = \boldsymbol{S}^q$. For the ARMA-Forsythe graph filter designed in Chapter 5, the basis functions are Forsythe polynomials. Meanwhile, with the partial fraction ARMA filter proposed in Chapter 6, (7.2) is related to the sub-filter expressed in the compact form (6.8) with $Q = 0$, $P = 1$, $\phi_p(\boldsymbol{S}) = \boldsymbol{S}^p$, $f_q(\boldsymbol{S}) = \boldsymbol{S}^q$.

Hence, by defining the matrices

$$\boldsymbol{P} = \sum_{p=0}^{P} a_p \phi_p(\mathbf{S}), \qquad \boldsymbol{Q} = \sum_{q=0}^{Q} b_q f_q(\mathbf{S}), \tag{7.3}$$

we can write (7.2) in the matrix-vector form

$$\boldsymbol{P}\boldsymbol{y} = \boldsymbol{Q}\boldsymbol{x}. \tag{7.4}$$

To compute the filter output $\boldsymbol{y}$ in (7.4), we can first calculate the right-hand side denoted as $\boldsymbol{z} = \boldsymbol{Q}\boldsymbol{x}$ (which corresponds to pre-filtering $\boldsymbol{x}$ with an FIR style filter), and then $\boldsymbol{y}$ is found by simply solving the linear system

$$\boldsymbol{P}\boldsymbol{y} = \boldsymbol{z}. \tag{7.5}$$

In this chapter, we will propose centralized and distributed iterative methods to implement the linear system (7.5). We start with the conjugate gradient method for undirected graph filters in the next section. Then, we formulate the centralized implementation for directed graph filters.

7.2. CENTRALIZED IMPLEMENTATION

In this section, we mainly focus on the centralized implementation of ARMA graph filters. For an undirected graph, we use the conjugate gradient method [3] to implement the linear system (7.5), while we also propose the biconjugate gradient [4, 5] method for a directed graph.

7.2.1. CONJUGATE GRADIENT

For undirected graphs, we take $\boldsymbol{L}$ or its modifications, e.g., $\boldsymbol{L}_n$, as shift operator $\boldsymbol{S}$, which is a symmetric and positive-definite matrix. We further consider using the conjugate gradient (CG) method [3] to implement ARMA graph filters in the vertex domain. The CG method is an algorithm to compute the numerical solution of particular systems of linear equations. Also, it has two properties: 1) orthogonality of the residuals and 2) conjugacy of the search directions. Note that there are other efficient methods to solve the linear system (7.5) for undirected graphs, like first-order methods [2], and the power method [12]. The computational cost of the implementation reduces significantly for sparse matrices $\boldsymbol{S}$, i.e., for sparse graphs [13].

We now summarize the conjugate gradient method. As shown in Algorithm 7.1, the CG approach has a computational complexity that scales linearly with the number of edges E.

- For the ARMA graph filter in Chapter 4, we first need to compute $\boldsymbol{z} = \boldsymbol{Q}\boldsymbol{x}$, which by following the efficient implementation [14] requires Q multiplications with the shift operator $\boldsymbol{S}$, since the terms can be computed as $\boldsymbol{S}^k\boldsymbol{x} = \boldsymbol{S}(\boldsymbol{S}^{k-1}\boldsymbol{x})$. This part leads to an overall complexity of $O(QE)$.

  Then, in each iteration i of the CG, it is required to compute the term $\boldsymbol{P}\boldsymbol{d}^{(i)}$, which is computed in the same way as $\boldsymbol{z}$ and requires a computational effort of order $O(PE)$.

- For the ARMA-Forsythe graph filter related to an undirected graph in Chapter 5, we can compute the term $\boldsymbol{z} = \boldsymbol{Q}\boldsymbol{x}$ efficiently as

$$\boldsymbol{Q}\boldsymbol{x} = b_0 f_0(\boldsymbol{S})\boldsymbol{x} + b_1 f_1(\boldsymbol{S})\boldsymbol{x} + \cdots + b_Q f_Q(\boldsymbol{S})\boldsymbol{x}, \tag{7.6}$$

  where $f_q(\bullet)$ is the Forsythe polynomial and the term $b_q f_q(\boldsymbol{S})\boldsymbol{x}$ inside $\boldsymbol{Q}\boldsymbol{x}$ is computed recursively as

$$b_q f_q(\boldsymbol{S})\boldsymbol{x} = b_q(\boldsymbol{S} f_{q-1}(\boldsymbol{S})\boldsymbol{x} - \omega_q f_{q-1}(\boldsymbol{S})\boldsymbol{x} - \varphi_q f_{q-2}(\boldsymbol{S})\boldsymbol{x}). \tag{7.7}$$

Algorithm 7.1 : Conjugate gradient

1	**Input**:	$\boldsymbol{y}^{(0)}$, $\boldsymbol{x}$, coefficients a_p, b_q
2		accuracy ε, number of iterations T
3	**Initialization**:	$\boldsymbol{z}$, $\boldsymbol{P}\boldsymbol{y}^{(0)}$
4		$\boldsymbol{d}^{(0)} = \boldsymbol{r}^{(0)} = \boldsymbol{z} - \boldsymbol{P}\boldsymbol{y}^{(0)}$,
5		$\delta^{(0)} = \delta^{new} = \boldsymbol{r}^{(0)T}\boldsymbol{r}^{(0)}$
6	**Iteration**:	while $i < T$ and $\delta^{new} > \varepsilon^2\delta^{(0)}$
7		$\omega^{(i)} = \frac{\delta_{new}}{\boldsymbol{d}^{(i)T}\boldsymbol{P}\boldsymbol{d}^{(i)}}$
8		$\boldsymbol{y}^{(i+1)} = \boldsymbol{y}^{(i)} + \omega^{(i)}\boldsymbol{d}^{(i)}$,
9		$\boldsymbol{r}^{(i+1)} = \boldsymbol{r}^{(i)} - \omega^{(i)}\boldsymbol{P}\boldsymbol{d}^{(i)}$
10		$\delta^{old} = \delta^{new}$, $\delta^{new} = \boldsymbol{r}^{(i+1)T}\boldsymbol{r}^{(i+1)}$
11		$\varphi^{(i+1)} = \frac{\delta^{new}}{\delta^{old}}$, $\boldsymbol{d}^{(i+1)} = \boldsymbol{r}^{(i+1)} + \varphi^{(i+1)}\boldsymbol{d}^{(i)}$
12		$i = i + 1$
13	**Output**:	$\boldsymbol{y}^{(i+1)}$

Thus, the computational cost of $\boldsymbol{Q}\boldsymbol{x}$ is related to $\boldsymbol{S}f_{q-1}(\boldsymbol{S})\boldsymbol{x}$ leading to an overall complexity of $O(QE)$ [14]. Then, for every iteration i, the term $\boldsymbol{P}\boldsymbol{d}^{(i)}$ ($\boldsymbol{P}\boldsymbol{y}^{(0)}$) can be computed using the same recursive form and it requires a computational effort of order $O(PE)$.

- If considering that the conjugate gradient is arrested after T iterations, the overall implementation costs of the ARMA and ARMA-Forsythe graph filters are of order $O((PT+Q)E)$. Since the partial fraction form (6.8) is a parallel form of an ARMA graph filter, the total cost for implementing the graph filter in Chapter 6 is $O(KTE)$, where K is the number of sub-filters.

- We would like to highlight that the ARMA filter output with CG is computed without explicitly building the matrices $\boldsymbol{P}$ and $\boldsymbol{Q}$, and only considers their operation on a specific vector.

Algorithm	**7.2** : Biconjugate gradient	
1	**Input**:	$\boldsymbol{y}^{(0)}$, $\boldsymbol{x}$, filter coefficients and parameters
2		accuracy ε, number of iterations T
3	**Initialization**:	$\boldsymbol{z}$, $\boldsymbol{P}\boldsymbol{y}^{(0)}$
4		$\boldsymbol{d}^{(0)} = \boldsymbol{r}^{(0)} = \boldsymbol{z} - \boldsymbol{P}\boldsymbol{y}^{(0)}$,
5		choose $\tilde{\boldsymbol{r}}^{(0)}$ such that $\boldsymbol{r}^{(0)\,T}\tilde{\boldsymbol{r}}^{(0)} \neq 0$,
6		$\delta^{(0)} = \delta^{new} = \boldsymbol{r}^{(0)T}\tilde{\boldsymbol{r}}^{(0)}$, $\tilde{\boldsymbol{d}}^{(0)} = \tilde{\boldsymbol{r}}^{(0)}$
7	**Iteration**:	while $i < T$ and $\delta^{new} > \varepsilon^2\delta^{(0)}$
8		$\rho^{(i)} = \frac{\delta_{new}}{\tilde{\boldsymbol{d}}^{(i)H}\boldsymbol{P}\boldsymbol{d}^{(i)}}$
9		$\boldsymbol{y}^{(i+1)} = \boldsymbol{y}^{(i)} + \rho^{(i)}\boldsymbol{d}^{(i)}$,
10		$\boldsymbol{r}^{(i+1)} = \boldsymbol{r}^{(i)} - \rho^{(i)}\boldsymbol{P}\boldsymbol{d}^{(i)}$
11		$\tilde{\boldsymbol{r}}^{(i+1)} = \tilde{\boldsymbol{r}}^{(i)} - \rho^{(i)}\boldsymbol{P}^T\tilde{\boldsymbol{d}}^{(i)}$
12		$\delta^{old} = \delta^{new}$, $\delta^{new} = \boldsymbol{r}^{(i+1)T}\tilde{\boldsymbol{r}}^{(i+1)}$
13		$\theta^{(i+1)} = \frac{\delta^{new}}{\delta^{old}}$,
14		$\boldsymbol{d}^{(i+1)} = \boldsymbol{r}^{(i+1)} + \theta^{(i+1)}\boldsymbol{d}^{(i)}$,
15		$\tilde{\boldsymbol{d}}^{(i+1)} = \tilde{\boldsymbol{r}}^{(i+1)} + \theta^{(i+1)}\tilde{\boldsymbol{d}}^{(i)}$,
16		$i = i + 1$
17	**Output**:	$\boldsymbol{y}^{(i+1)}$

7.2.2. BICONJUGATE GRADIENT

For undirected graphs, the conjugate gradient method [1] is proposed to solve the ARMA graph filter implementation. In this section, we introduce the biconjugate gradient (BiCG) method to implement the designed graph filters for a directed graph.

As we mentioned in the previous chapters, for directed graphs, we take matrix $\boldsymbol{A}$ or modifications thereof as shift operator. The conjugate gradient method is not suitable for a directed graph, because i) the adjacency matrix $\boldsymbol{A}$ is usually not symmetric and positive-definite for a directed graph, ii) the residual vectors in the method cannot be made orthogonal with short iterations. Thus, we present an alternative biconjugate gradient [4, 5] method to complete the cen-

tralized implementation. We can summarize it as follows:

- Similar to the CG approach, the BiCG [15] also has a computational complexity that scales linearly with the number of edges E. As shown in Algorithm 7.2, we also need to compute the terms $\boldsymbol{P}\boldsymbol{d}^{(i)}$ and $\boldsymbol{Q}\boldsymbol{x}$ which are the same as for the CG approach in the previous section. The computational cost of the two terms is in total $O((PT+Q)E)$, where T is the number of iterations.

- Note that for computing the term $\boldsymbol{P}^T\tilde{\boldsymbol{d}}^{(i)}$ in BiCG, we can also reduce the cost by computing it using a recurrence form. For the ARMA graph filter, we have

$$\begin{aligned}\boldsymbol{P}^T\tilde{\boldsymbol{d}}^{(i)} &= (a_0\boldsymbol{S}^0 + a_1\boldsymbol{S}^1 + \cdots + a_P\boldsymbol{S}^P)^T\tilde{\boldsymbol{d}}^{(i)} \\ &= a_0(\boldsymbol{S}^0)^T\tilde{\boldsymbol{d}}^{(i)} + a_1(\boldsymbol{S}^1)^T\tilde{\boldsymbol{d}}^{(i)} + \cdots + a_P(\boldsymbol{S}^P)^T\tilde{\boldsymbol{d}}^{(i)} \quad .\end{aligned} \tag{7.8}$$

  Every term $a_p(\boldsymbol{S}^p)^T\tilde{\boldsymbol{d}}^{(i)}$ can be computed recursively as

$$(\mathbf{S}^p)^T\tilde{\mathbf{d}}^{(i)} = (\mathbf{S}^{(p-1)}\mathbf{S})^T\tilde{\mathbf{d}}^{(i)} = \mathbf{S}^T(\mathbf{S}^{(p-1)})^T\tilde{\mathbf{d}}^{(i)}, \tag{7.9}$$

  which has a complexity $O(PE)$ for every iteration.

- For the ARMA-Forsythe graph filter in Chapter 5, we first rewrite the generating process of the orthogonal polynomial as

$$\phi_p(\boldsymbol{S}) = \phi_{p-1}(\boldsymbol{S})\boldsymbol{S} - \omega_p\phi_{p-1}(\boldsymbol{S}) - \varphi_p\phi_{p-2}(\boldsymbol{S}).$$

  Since every term of the orthogonal polynomial is a multiplication with the shift operator $\boldsymbol{S}$, rewriting the generating process will not change the polynomial structure.

  After that, $\boldsymbol{P}^T\tilde{\boldsymbol{d}}^{(i)}$ can be formulated as

$$\boldsymbol{P}^T\tilde{\boldsymbol{d}}^{(i)} = a_0\phi_0^T(\boldsymbol{S})\tilde{\boldsymbol{d}}^{(i)} + \cdots + a_P\phi_P^T(\boldsymbol{S})\tilde{\boldsymbol{d}}^{(i)}, \tag{7.10}$$

  and the term $a_p\phi_p^T(\boldsymbol{S})\tilde{\boldsymbol{d}}^{(i)}$ is computed as

$$\begin{aligned}&a_p\phi_p^T(\boldsymbol{S})\tilde{\boldsymbol{d}}^{(i)} \\ &= a_p(\phi_{p-1}(\boldsymbol{S})\boldsymbol{S} - \omega_p\phi_{p-1}(\boldsymbol{S}) - \varphi_p\phi_{p-2}(\boldsymbol{S}))^T\tilde{\boldsymbol{d}}^{(i)} \\ &= a_p(\boldsymbol{S}^T\phi_{p-1}^T(\boldsymbol{S})\tilde{\boldsymbol{d}}^{(i)} - \omega_p\phi_{p-1}^T(\boldsymbol{S})\tilde{\boldsymbol{d}}^{(i)} - \varphi_p\phi_{p-2}^T(\boldsymbol{S})\tilde{\boldsymbol{d}}^{(i)}),\end{aligned} \tag{7.11}$$

which is recursively solved by the previous two terms, such as $\phi_{p-1}^T(\boldsymbol{S})\widetilde{\boldsymbol{d}}^{(i)}$ and $\phi_{p-2}^T(\boldsymbol{S})\widetilde{\boldsymbol{d}}^{(i)}$. The computational cost of the term $a_p\phi_p^T(\boldsymbol{S})\widetilde{\boldsymbol{d}}^{(i)}$ is thus related to $\boldsymbol{S}^T\phi_{p-1}^T(\boldsymbol{S})\widetilde{\boldsymbol{d}}^{(i)}$ with a complexity $O(PE)$ for every iteration in BiCG.

- Considering the BiCG algorithm with maximum T iterations, the overall implementation cost for an ARMA graph filter of the form in Chapters 4 and 5 is of order $O((2PT+Q)E)$. With the partial fraction form (6.8) in Chapter 6, the total cost is $O(2KTE)$ corresponding to the number of sub-filters K.

7.2.3. NUMERICAL RESULTS

In this section, we evaluate the centralized implementation for both undirected and directed graph filters. To compare filters with the same implementation costs, we mainly consider the ARMA graph filters of the form in Chapters 4 and 5. In the simulation, we use the well-studied FIR and IIR [16] graph filters as comparisons.

CG implementation performance. We now aim at analyzing the ARMA implementation performance using the CG approach w.r.t. its implementation cost. We implement the universally designed ARMA and ARMA-Forsythe filters using CG on the Erdős Rényi graph [17] with a link probability of $p = 0.1$, and consider the graph size: $N = 500$. We use the universally designed FIR and IIR [16] graph filters as benchmarks.

The ARMA filter coefficients (in Chapter 4) and the parameters for generating orthogonal polynomials in the ARMA-Forsythe filter (in Chapter 5) are designed universally with 100 grid points in the range $[0,2]$. Meanwhile, the FIR filter is designed using LLS also with 100 grid points and the IIR filter following the Butterworth approach [16].

The filter is applied to a white input and the desired frequency response (low pass filter) is compared to the division of the filter output and the input in the frequency domain. We measure the approximation accuracy with the root normalized mean square error (RNMSE) of the frequency response of the filter:

$$\text{RNMSE} = \frac{\left\| \hat{\boldsymbol{y}}^{(t)} - \hat{\boldsymbol{h}} \right\|}{\left\| \hat{\boldsymbol{h}} \right\|}, \tag{7.12}$$

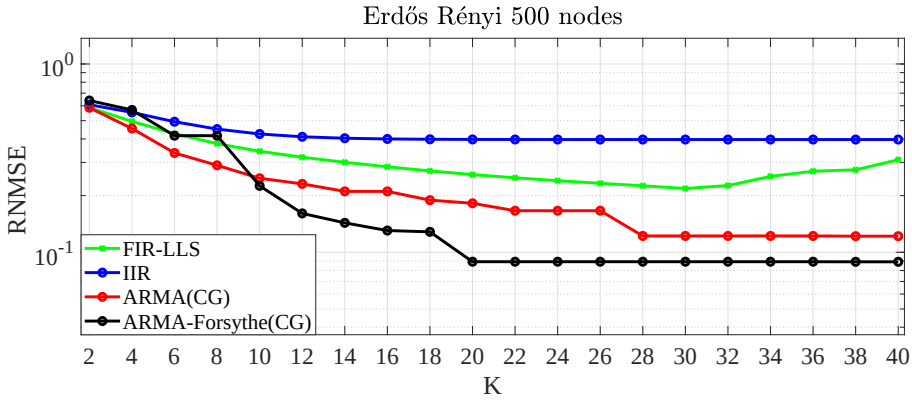


Figure 7.1: RNMSE of the ARMA and ARMA-Forsythe graph filter implementations on an Erdős Rényi graph with $N = 500$. Performance evaluation for the trade-off between computational cost and approximation accuracy. For CG, the complexity of the ARMA (ARMA-Forsythe) implementation is limited by $PT + Q \leq K$. Note that K represents the FIR and IIR graph filter order.

where $\hat{\boldsymbol{y}}^{(t)}$ is the output of every iteration and $\hat{\boldsymbol{h}}$ is the designed ideal low pass for the Erdős Rényi graph in the frequency domain.

In Fig. 7.1, we show the performance of the ARMA filter (Algorithm 7.1) when the CG is halted after T iterations such that $PT + Q \leq K$ holds, i.e., the ARMA (or ARMA-Forsythe) filter has a smaller or the same implementation cost compared to the FIR filter. For the CG, we set $\varepsilon = 10^{-3}$. The IIR filter has the same order K as the FIR filter and is given a maximum number of iterations of $T = 40$. The results show that the ARMA (or ARMA-Forsythe) filter has a lower approximation error than other alternatives with similar or smaller complexity. Note that ARMA-Forsythe has a better performance than the ARMA graph filter.

To highlight the benefits of the universal design approach, we consider an ER graph of a larger size (> 100 nodes). In Fig. 7.1, we notice that even for the case with $N = 500$, the universal design based on 100 grid points is a wise choice and yields good performance.

BiCG implementation performance. Now, we consider a directed network graph to show the trade-off between the performance and computational cost. We test the implementation for a graph with $N = 200$, shown in Fig. 7.2. Note that the frequencies in Fig. 7.2 are real and complex conjugate pairs. Similar to the CG approach, the filter performance is evaluated by (7.12). The filter coefficients and parameters for generating orthogonal polynomials are again computed by a

universal design with 100 grid points. For BiCG, the break accuracy is 10^{-3}.

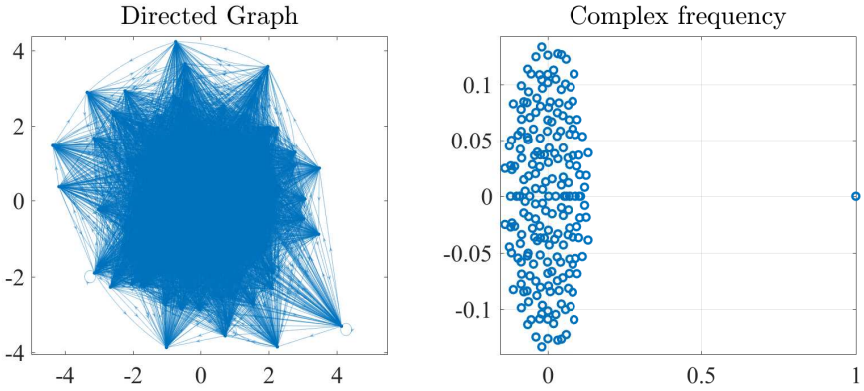


Figure 7.2: Directed network graph of $N = 200$ nodes with $E = 10597$ edges having different weights in the interval $[0,8]$.

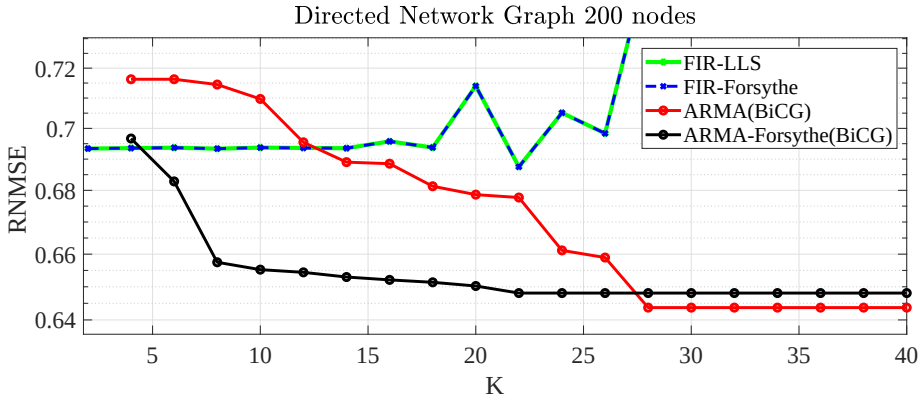


Figure 7.3: RNMSE of the ARMA and ARMA-Forsythe graph filter implementations on a directed graph with $N = 200$. Performance evaluation for the trade-off between computational cost and approximation accuracy. RNMSE of BiCG implementation on a directed network graph with filter order limited by $2PT + Q \leq K$. The minimum orders for both cases are $K = 3$ ($P = 1$, $Q = 1$, and $T = 1$).

In Fig. 7.3, we show the performance when the BiCG is halted after T iterations such that $2PT+Q \leq K$ holds for the ARMA and ARMA-Forsythe graph filters. Results show that with the same or lower implementation complexity and $K < 25$, the ARMA-Forsythe design method has a smaller approximation error than the

FIR and ARMA graph filters. With a higher-order $K > 25$, the ARMA graph filter shows a slightly better performance than the ARMA-Forsythe. We notice that the FIR and FIR-Forsythe (they overlap in the figure) show an unstable performance for high order. However, ARMA (and ARMA-Forsythe) avoid this situation due to the separation of filter order into P and Q.

Remark. The motivation of this section is to show the implementation benefits of the ARMA model and the robustness of our universal design fashion. We highlight the potential of the ARMA (and ARMA-Forsythe) graph filter regarding the performance improvement over other alternative filters such as FIR, and IIR graph filters. For some applications such as building graph filter banks, designing graph wavelets, and spectral clustering, the above results can be useful for replacing the FIR with ARMA (or ARMA-Forsythe) graph filters.

7.2.4. GRAPH SIGNAL INTERPOLATION.

In this subsection, we aim to show an application of the CG implementation. We illustrate the performance of ARMA graph filters in interpolating missing values in the Molene weather data set. The data set contains hourly observations of temperature measurements collected in January 2014 in the region of Brest (France).

The undirected graph, containing 32 cities (nodes), is built according to [18], which accounts for the smoothness of the data w.r.t. the graph structure. We consider the case that a portion of the graph signal is missing, and by exploiting the smoothness prior we aim to reconstruct the overall graph signal from noisy measurements.

Experimental setup. Given $\boldsymbol{x}'$ the observed signal and $\boldsymbol{x}$ the original graph signal, this interpolation problem is formulated as [19] [20]:

$$\min_{\boldsymbol{x}} \left\| \boldsymbol{T}(\boldsymbol{x}-\boldsymbol{x}') \right\|_2^2 + \omega \boldsymbol{x}^T \boldsymbol{L}_n \boldsymbol{x} \tag{7.13}$$

where $\boldsymbol{T}$ is a diagonal matrix with $T_{ii} = 1$ if x_i is known and $T_{ii} = 0$ otherwise; ω is the weight for the prior. The optimal solution of (7.13) is

$$\tilde{\boldsymbol{x}} = (\boldsymbol{T} + \omega \boldsymbol{L}_n)^{-1} \boldsymbol{x}', \tag{7.14}$$

Access to the raw data is through the link: https://donneespubliques.meteofrance.fr/donnees_libres/Hackathon/RADOMEH.tar.gz

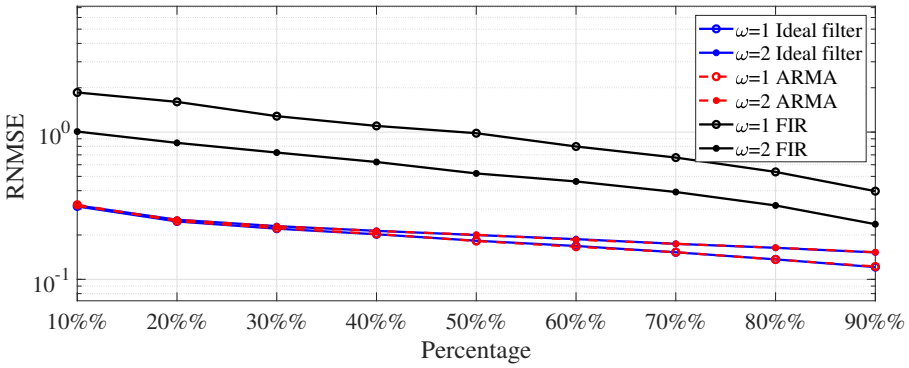


Figure 7.4: RNMSE of the ARMA graph filter for interpolation of the Molene data set, where $\omega = 1, 2$. As two comparisons, the ideal graph filter and the FIR filter with order $K = 20$ are shown with the same values of ω.

which considering $\boldsymbol{P} = \boldsymbol{T} + \omega \boldsymbol{L}_n$ is solved through an ARMA graph filter (7.5). We consider using the CG to implement (7.14) where ε is set to 10^{-2} and the maximum number of iterations T to 20. As a comparison, for the FIR graph filter, the coefficients are found as the solution of

$$\min_{g_k} \left\| (\boldsymbol{T} + \omega \boldsymbol{L}_n)^{-1} - \sum_{k=0}^{K} g_k \boldsymbol{L}_n^k \right\|_F^2 \tag{7.15}$$

where the g_k values represent the FIR coefficients.

Results. In Fig. 7.4 we show the RNMSE between the reconstructed signal $\tilde{\boldsymbol{x}}$ and the original one $\boldsymbol{x}$ as a function of the portion of missing data. Additionally, to construct the observed signal $\boldsymbol{x}'$, we add a zero-mean Gaussian noise with variance $\sigma^2 = 10^{-2}$ to the original signal $\boldsymbol{x}$ and randomly wipe off signals up to the specified percentage. The performance is averaged over all 744 observations. We plot the numerical RNMSE for different percentages and two ω values. These results show that the RNMSE reduces for the ARMA graph filter when the percentage of known values increases. As a comparison, we notice that the ARMA graph filter offers a similar performance to the ideal graph filter. The FIR graph filter ($K = 20$) yields the worse result in this case.

7.3. DISTRIBUTED IMPLEMENTATION

Although the earlier mentioned iterative methods to solve the linear system $\boldsymbol{P}\boldsymbol{y} = \boldsymbol{Q}\boldsymbol{x}$ are efficient, they do not allow for an easy distribution. In this section, we will illustrate two alternative iterative methods which are easy to distribute and implement, named Richardson iteration and weighted Jacobi iteration.

7.3.1. RICHARDSON ITERATION

For the linear system $\boldsymbol{P}\boldsymbol{y} = \boldsymbol{Q}\boldsymbol{x}$, one distributed solution is given by the Richardson method [6–8] which results in the following iteration

$$\boldsymbol{y}^{(t+1)} = \boldsymbol{y}^{(t)} - w(\boldsymbol{P}\boldsymbol{y}^{(t)} - \boldsymbol{z}) \tag{7.16}$$

with step size w, arbitrary initial point $\boldsymbol{y}^{(0)}$ and $\boldsymbol{z} = \boldsymbol{Q}\boldsymbol{x}$. Since the matrix-vector product $\boldsymbol{P}\boldsymbol{y}^{(t)}$ can be computed recursively and matrix $\boldsymbol{P}$ is a (orthogonal) polynomial in the shift operator $\boldsymbol{S}$, this method can be easily realized in a distributed manner as discussed earlier.

Convergence condition. The convergence of the Richardson method is determined by the error in every iteration step. Subtracting the exact solution $\boldsymbol{y}$, and introducing the notation for the error $\boldsymbol{e}^{(t+1)} = \boldsymbol{y}^{(t+1)} - \boldsymbol{y}$, we get the following derivation:

$$\begin{aligned}
\boldsymbol{e}^{(t+1)} &= \boldsymbol{y}^{(t)} - w(\boldsymbol{P}\boldsymbol{y}^{(t)} - \boldsymbol{z}) - \boldsymbol{y} \\
&= (\boldsymbol{I} - w\boldsymbol{P})\boldsymbol{y}^{(t)} - (\boldsymbol{I} - w\boldsymbol{P})\boldsymbol{y} \\
&= (\boldsymbol{I} - w\boldsymbol{P})(\boldsymbol{y}^{(t)} - \boldsymbol{y}) \\
&= (\boldsymbol{I} - w\boldsymbol{P})\boldsymbol{e}^{(t)}
\end{aligned}$$

Thus, we have

$$\left\| \boldsymbol{e}^{(t+1)} \right\| = \left\| (\boldsymbol{I} - w\boldsymbol{P})\boldsymbol{e}^{(t)} \right\| \leq \| \boldsymbol{I} - w\boldsymbol{P} \| \left\| \boldsymbol{e}^{(t)} \right\|,$$

for any vector norm and the corresponding induced matrix norm. Under this circumstance, the convergence condition for the Richardson method is formulated as

$$\| \boldsymbol{I} - w\boldsymbol{P} \| < 1, \tag{7.17}$$

for any matrix $\boldsymbol{P}$. Note that the standard Richardson iterative method does not require $\boldsymbol{P}$ to be symmetric. In other words, the Richardson method can handle directed graphs with shift operator $\boldsymbol{S} = \boldsymbol{A}$ (or a modification thereof) as long as the matrix $\boldsymbol{P}$ satisfies (7.17).

Suppose that $\boldsymbol{P}$ is symmetric, i.e., the corresponding graph is an undirected graph, then the convergence is determined by the spectral radius of $\boldsymbol{I} - w\boldsymbol{P}$, which can be written as

$$\rho(\boldsymbol{I} - w\boldsymbol{P}) = \max_k |1 - wP(\lambda_k)|.$$

More specifically, the iteration converges with the condition $\rho(\boldsymbol{I} - w\boldsymbol{P}) < 1$. In other words, we need $0 < w < w_{\max}$ where $w_{\max} = 2/\max_p P(\lambda_k)$. Also, the smaller $\rho(\boldsymbol{I} - w\boldsymbol{P})$ we have, the faster it converges. The fastest convergence for the specific case of an undirected graph is obtained as

$$w_{\text{opt}} = \min_w \max_k |1 - wP(\lambda_k)|, \tag{7.18}$$

which can be solved as

$$w_{\text{opt}} = \frac{2}{\min_k P(\lambda_k) + \max_k P(\lambda_k)}. \tag{7.19}$$

The Richardson iteration is an iterative method for solving a system of linear equations. The total computational cost for both undirected and directed graphs is $O((PT+Q)E)$ for the ARMA model, where T is the number of iterations and P, Q are the filter orders.

In the next section, we will provide another distributed implementation named weighted Jacobi iteration to solve the linear system which needs to decompose the matrix $\boldsymbol{P}$ into two parts.

7.3.2. WEIGHTED JACOBI ITERATION

In numerical linear algebra, the Jacobi method [8–11] is an iterative algorithm for determining the solution of a set of linear equations. The weighted Jacobi iteration uses a parameter w in the iteration as

$$\boldsymbol{y}^{(t+1)} = w\boldsymbol{D}^{-1}(\boldsymbol{z} - \boldsymbol{R}\boldsymbol{y}^{(t)}) + (1-w)\boldsymbol{y}^{(t)}. \tag{7.20}$$

where $\boldsymbol{y}^{(t)}$ is the t-th approximation or iteration of $\boldsymbol{y}$, and w is the step size. The matrix $\boldsymbol{P}$ in (7.5) can be decomposed into a diagonal component $\boldsymbol{D}$, and the remainder $\boldsymbol{R}$ as

$$\boldsymbol{P} = \boldsymbol{D} + \boldsymbol{R}, \tag{7.21}$$

where

$$\boldsymbol{D} = \begin{bmatrix} p_{11} & 0 & \cdots & 0 \\ 0 & p_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & p_{NN} \end{bmatrix}, \quad \boldsymbol{R} = \begin{bmatrix} 0 & p_{12} & \cdots & p_{1N} \\ p_{21} & 0 & \cdots & p_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ p_{N1} & p_{N2} & \cdots & p_{NN} \end{bmatrix}.$$

Note that the weighted Jacobi method is one of the splitting methods [8] where $\boldsymbol{D}$ is the diagonal component. In general, the Jacobi method is efficiently implemented in a distributed fashion, i.e., the inversion of a diagonal matrix is trivial to distribute [21]. To be specific, we could compute the matrix $\boldsymbol{R}\boldsymbol{y}^{(t)}$ by first implementing $\boldsymbol{P}\boldsymbol{x}$ in a distributed way. i.e., recursively applying signal $\boldsymbol{y}^{(t)}$ on $\boldsymbol{S}$, and then subtracting $\boldsymbol{D}\boldsymbol{x}$.

Convergence condition. Similar to the Richardson iteration, the convergence condition of the weighted Jacobi iteration is determined by the error in every iteration step using the notation $\boldsymbol{e}^{(t+1)} = \boldsymbol{y}^{(t+1)} - \boldsymbol{y}$.

Specifically, we have

$$\begin{aligned} \boldsymbol{e}^{(t+1)} &= w\boldsymbol{D}^{-1}(\boldsymbol{z} - \boldsymbol{R}\boldsymbol{y}^{(t)}) + (1-w)\boldsymbol{y}^{(t)} - \boldsymbol{y} \\ &= w\boldsymbol{D}^{-1}\boldsymbol{P}\boldsymbol{y} - w\boldsymbol{D}^{-1}\boldsymbol{R}\boldsymbol{y}^{(t)} + \boldsymbol{y}^{(t)} - w\boldsymbol{y}^{(t)} - \boldsymbol{y} \\ &= (w\boldsymbol{D}^{-1}\boldsymbol{P} - \boldsymbol{I})\boldsymbol{y} - (w\boldsymbol{D}^{-1}\boldsymbol{R} + w\boldsymbol{D}^{-1}\boldsymbol{D} - \boldsymbol{I})\boldsymbol{y}^{(t)} \\ &= (\boldsymbol{I} - w\boldsymbol{D}^{-1}\boldsymbol{P})(\boldsymbol{y}^{(t)} - \boldsymbol{y}) \\ &= (\boldsymbol{I} - w\boldsymbol{D}^{-1}\boldsymbol{P})\boldsymbol{e}^{(t)} \end{aligned}$$

And, thus we obtain

$$\left\|\boldsymbol{e}^{(t+1)}\right\| = \left\|(\boldsymbol{I} - w\boldsymbol{D}^{-1}\boldsymbol{P})\boldsymbol{e}^{(t)}\right\| \leq \left\|\boldsymbol{I} - w\boldsymbol{D}^{-1}\boldsymbol{P}\right\| \left\|\boldsymbol{e}^{(t)}\right\|.$$

As a result, the convergence condition for the weighted Jacobi iteration is

$$\left\|\boldsymbol{I} - w\boldsymbol{D}^{-1}\boldsymbol{P}\right\| < 1. \tag{7.22}$$

Note that the convergence condition (7.22) is suitable for both directed and undirected graphs.

In case the matrix $\boldsymbol{P}$ is symmetric, i.e., the graph is an undirected graph, the weighted Jacobi iteration converges if $\rho(\boldsymbol{I} - w\boldsymbol{C}) < 1$, where $\boldsymbol{C} = \boldsymbol{D}^{-1}\boldsymbol{P}$. In other words, we need $0 < w < w_{\max}$ with $w_{\max} = 2/\max_k C(\lambda_k)$. The optimal solution for the step size can be formulated as

$$w_{\text{opt}} = \frac{2}{\min_k C(\lambda_k) + \max_k C(\lambda_k)}. \tag{7.23}$$

The total computational cost of the weighted Jacobi iteration is $O((PT+Q)E+PTN)$, where T is the number of iterations, N is the number of nodes, and E is the number of edges.

For this section, we presented general guarantees for the convergence of distributed filter implementations. Differently from previous studies, we take both directed and undirected graphs into account and provide the basic convergence condition which is suitable for all situations. In the following section, we give a series of numerical experiments to demonstrate the behavior of the two mentioned distributed implementations.

7.3.3. NUMERICAL RESULTS

In this section, we illustrate the convergence results for the two distributed implementations and briefly discuss the changing of step size to provide the convergence speed. Again, we assume that the graph spectrum is unavailable to the designer and we use the coefficients from the universal design to test the implementations. Moreover, we apply a white input to the filter and consider an ideal low pass filter for all tests.

To illustrate our results, we simulate two different case studies: one with an Erdős Rényi graph (undirected case with $N = 500$ nodes) and the other one with a directed network graph with $N = 200$ nodes. For both cases, we take the filter orders as $P = 2$ and $Q = 4$. To test the different design methods, we take the ARMA graph filter (of Chapter 4) for the undirected case and use the ARMA-Forsythe graph filter (of Chapter 5) to evaluate the directed case.

Undirected case. We aim at analyzing the distributed implementation performances. For the Erdős Rényi graph, we apply the ARMA(4, 2) graph filter designed in Chapter 4.

Note that the coefficients of the ARMA filter are designed universally using the LS approach with 100 grid points corresponding to Fig. 4.1. We use the root normalized mean square error (RNMSE) (7.12) to show the performance of the proposed methods in Fig. 7.5. In the figure, the results of the two distributed implementations with the optimal step sizes overlap with each other.

We can notice that for the same step size w, the weighted Jacobi iteration and Richardson iteration give different speeds of convergence. To be specific, with a small step size, i.e., $w = 0.01$, we can see that the weighted Jacobi implementation converges faster than the Richardson implementation. In addition, we observe that the convergence speed can be increased, and the fastest speed

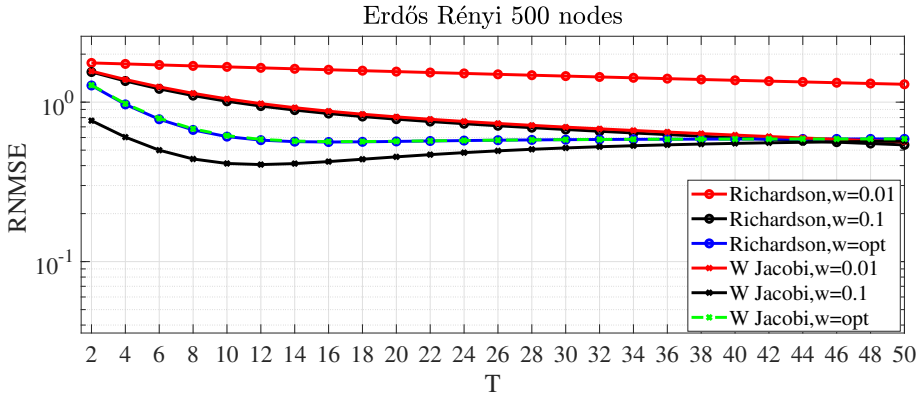


Figure 7.5: Convergence results of two distributed implementations of the ARMA(4, 2) graph filter with different step sizes. The experiment uses an Erdős Rényi graph with link probability $p = 0.1$. T is the number of iterations.

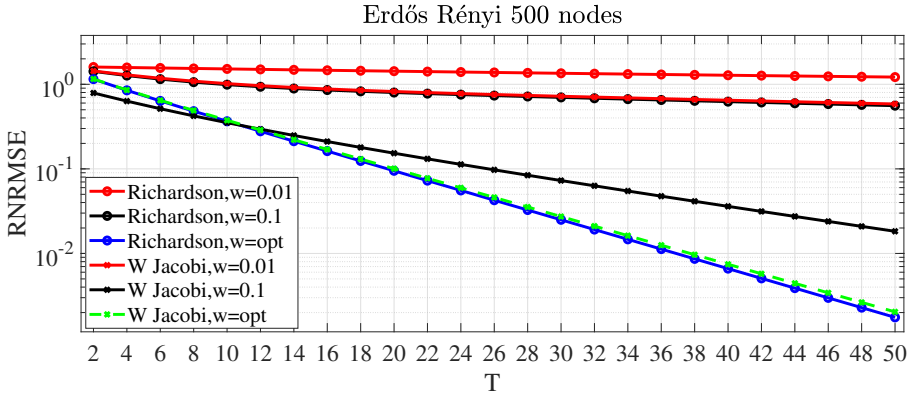


Figure 7.6: RNRMSE versus the number of iterations of two distributed implementations for ARMA(4, 2) with different step sizes. T is the number of iterations and the Erdős Rényi graph is the same as in Fig. 7.5.

is obtained by the optimal step size w_{opt}, i.e., $w_{\text{opt}} = 1.88$ for the Richardson iteration and $w_{\text{opt}} = 0.21$ for the weighted Jacobi implementation. Note that there is no guarantee of monotonic convergence to the true solution and the error at each iteration is not strictly decreasing, i.e., $w = 0.1$ using the weighted Jacobi iteration gives a lower error before the iteration reaches convergence.

To compare the results, we define the root normalized residual mean square

error (RNRMSE) as

$$\text{RNRMSE} = \frac{\|\boldsymbol{y}^{(t)} - \boldsymbol{y}\|}{\|\boldsymbol{y}\|}, \tag{7.24}$$

where $\boldsymbol{y} = \boldsymbol{P}^{-1}\boldsymbol{Q}\boldsymbol{x}$ and $\boldsymbol{y}^{(t)}$ is the filter output for every iteration in the vertex domain. Fig. 7.6 gives the RNRMSE of the two distributed implementations after different iterations. We see that the weighted Jacobi iteration and Richardson iteration converge much faster with the optimal step size w_{opt}. From Fig. 7.5 and Fig. 7.6, we notice that the optimal step sizes of the two distributed implementations give similar performances.

Directed case. In the following, we show the behavior for the directed case. First, we highlight that there is no closed-form for computing the optimal step size w in the directed case. Also, the step sizes w we selected in this section only guarantee the condition (7.17) and (7.22) for the corresponding directed graph in Fig. 7.2.

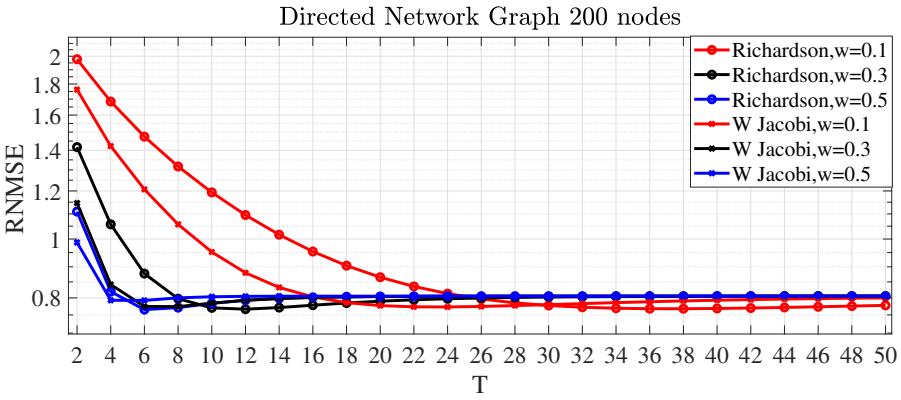


Figure 7.7: Convergence results of a distributed implementation of the ARMA-Forsythe(4, 2) graph filter with different step sizes. The experiment uses the directed network graph with $N = 200$ depicted in Fig. 7.2. T is the number of iterations.

Note that the coefficients of the ARMA-Forsythe(4, 2) graph filter are generated with 100 complex grid points in the unit disc. The filter is applied to a white input and the output for every iteration is evaluated by the error RNMSE (7.12) in the frequency domain. Fig. 7.7 gives the performances of two distributed implementations with different step sizes $w = 0.1, 0.3$ and 0.5.

We can make the following observations. After a few iterations, the errors

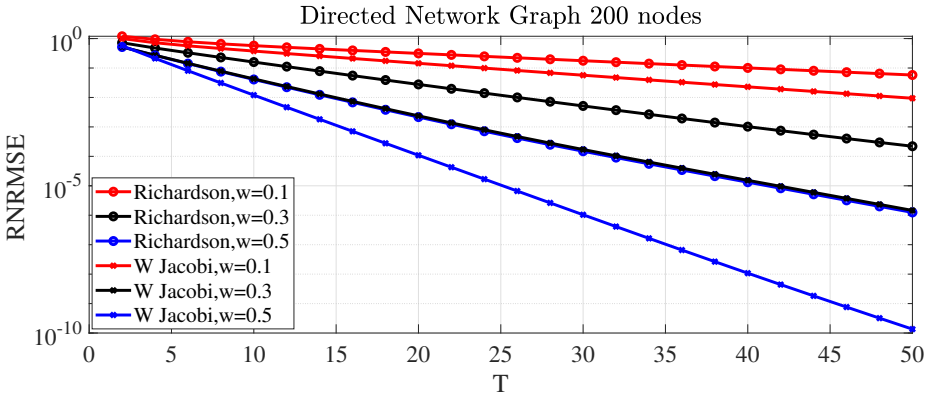


Figure 7.8: RNRMSE versus the number of iterations of two distributed implementations for ARMA-Forsythe(4, 2) with different step sizes. T is the number of iterations. The directed graph we used is the same as in Fig. 7.2.

become relatively close, meaning that the ARMA-Forsythe(4, 2) output spectra of the two distributed implementations have a different convergence speed because of the different step sizes, i.e., $w = 0.1, 0.3$ and 0.5. For both implementations, with larger step sizes, the convergence is faster.

To give further insight, Figure. 7.8 plots the RNRMSE (7.24) of the two distributed implementations for the ARMA-Forsythe(4, 2) graph filter with different step sizes. Note that, for the selected directed graph, the weighted Jacobi iteration shows better performance than the Richardson iteration for all three step sizes. In the next subsection, we will give an application of the distributed implementations and compare the results with each other.

7.3.4. GRAPH SIGNAL DENOISING.

In this subsection, the aim is to design an ARMA graph filter for disturbance suppression (denoising) [22, 23]. Consider a measurement, as in the temperature data set, which is composed of a slow-varying desired signal $\boldsymbol{x}$, and a superimposed fast changing disturbance $\boldsymbol{\varepsilon}$. The noisy data then is

$$\boldsymbol{x}' = \boldsymbol{x} + \boldsymbol{\varepsilon}. \tag{7.25}$$

For the undirected graph [18] used for this application, we take the same one (32 cities) as in Section 7.2.4.

Experimental setup. The optimal denoising system can then be formulated through a minimization of the cost function

$$\min_{\boldsymbol{x}} \left\| (\boldsymbol{x} - \boldsymbol{x}') \right\|_2^2 + \omega \boldsymbol{x}^T \boldsymbol{L}_n \boldsymbol{x}, \tag{7.26}$$

which is a special case of (7.13) with $\boldsymbol{T} = \boldsymbol{I}$.

In (7.26), the minimization of the first term forces the output signal $\boldsymbol{x}'$ to be close to the observation $\boldsymbol{x}$. The second term, $\omega \boldsymbol{x}^T \boldsymbol{L}_n \boldsymbol{x}$, formulates a measure for the signal smoothness of the graph filter output. The coefficient ω makes a compromise between the two terms.

The solution of the minimization (7.26) is

$$\tilde{\boldsymbol{x}} = (\boldsymbol{I} + 2\omega \boldsymbol{L}_n)^{-1} \boldsymbol{x}'. \tag{7.27}$$

which is an ARMA graph filter with $\boldsymbol{P} = \boldsymbol{I} + 2\omega \boldsymbol{L}_n$.

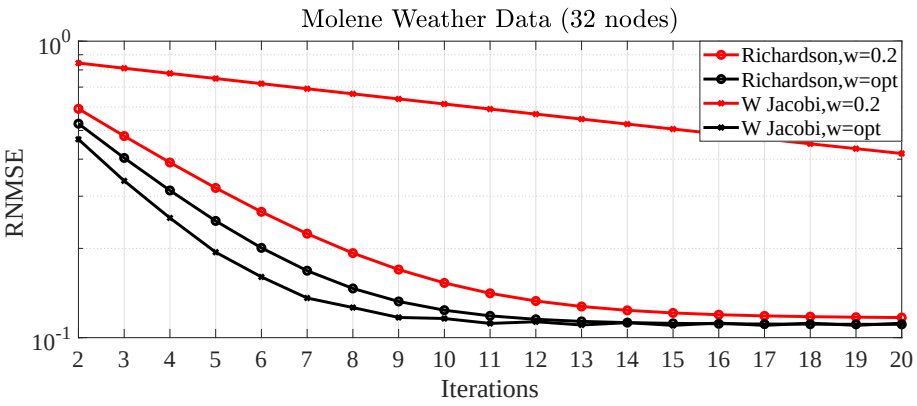


Figure 7.9: RNMSE of the ARMA graph filter for graph signal denoising of the Molene data set, where $\omega = 2$. The step size $w = 0.2$ and the optimal value w_{opt} are shown for the two distributed implementations.

Results. In Fig. 7.9, we show the RNMSE between the denoised signal and the original one $\boldsymbol{x}$. For the disturbance $\boldsymbol{\epsilon}$, we add a zero-mean Gaussian noise with variance $\sigma^2 = 10^{-2}$ to the original signal $\boldsymbol{x}$ to generate the observation $\boldsymbol{x}'$. The parameter is $\omega = 2$. Fig. 7.9 shows the performance for the 50-th observation of the temperature data. We plot the results of two distributed implementations for different step sizes w. These results show that the two methods have a similar

performance for the optimal step size, i.e., $w_{\text{opt}} = 0.25$ for the Richardson iteration and $w_{\text{opt}} = 1.5$ for the weighted Jacobi implementation. The weighted Jacobi implementation is a bit faster though.

7.4. CONCLUSION

In this chapter, we discussed some implementation aspects of graph filters. We formulated the relation between the output and the input of a graph filter for the ARMA model. For the different ARMA formulations, we first discussed an efficient implementation for both directed and undirected graphs namely the conjugate gradient and biconjugate gradient methods. Through the ARMA and ARMA-Forsythe graph filters, we illustrated the improvement of the approximation accuracy compared with the FIR graph filter for the same computational cost.

In addition, the graph filter can also be implemented in a distributed fashion leading to amenable savings in terms of distributed communication. The Richardson method and weighted Jacobi iteration are utilized to implement the graph filters for both directed and undirected cases in a distributed manner. For the undirected graph case, we can formulate the optimal step size in closed form. Alternatively, we derive a general convergence condition for the distributed implementation which is suitable for the directed graph case.

REFERENCES

[1] 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).

[2] D. P. Bertsekas, *Convex optimization theory* (Athena Scientific Belmont, 2009).

[3] J. R. Shewchuk *et al.*, *An introduction to the conjugate gradient method without the agonizing pain,* (1994).

[4] V. Faber and T. Manteuffel, *Necessary and sufficient conditions for the existence of a conjugate gradient method,* SIAM Journal on Numerical Analysis **21**, 352 (1984).

[5] J. Zhang, H. Dai, and J. Zhao, *Generalized global conjugate gradient squared algorithm,* Applied Mathematics and Computation **216**, 3694 (2010).

[6] L. F. Richardson, *Ix. the approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam,* Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character **210**, 307 (1911).

[7] G. Opfer and G. Schober, *Richardson's iteration for nonsymmetric matrices,* Linear algebra and its applications **58**, 343 (1984).

[8] Y. Saad, *Iterative methods for sparse linear systems*, Vol. 82 (siam, 2003).

[9] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: numerical methods*, Vol. 23 (Prentice hall Englewood Cliffs, NJ, 1989).

[10] L. Lei, *Convergence of asynchronous iteration with arbitrary splitting form,* Linear Algebra and its Applications **113**, 119 (1989).

[11] J. M. Bull and T. Freeman, *Numerical performance of an asynchronous jacobi iteration,* in *Parallel Processing: CONPAR 92—VAPP V* (Springer, 1992) pp. 361–366.

[12] J. H. Wilkinson and J. H. Wilkinson, *The algebraic eigenvalue problem*, Vol. 87 (Clarendon Press Oxford, 1965).

[13] M. E. Newman, *The structure and function of complex networks,* SIAM review **45**, 167 (2003).

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

[15] M. Křížek and J. Mlỳnek, *On the preconditioned biconjugate gradients for solving linear complex equations arising from finite elements,* Banach Center Publications **29**, 195 (1994).

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

[17] P. Erdos and A. Rényi, *On the evolution of random graphs,* Publ. Math. Inst. Hung. Acad. Sci **5**, 17 (1960).

[18] S. P. Chepuri, S. Liu, G. Leus, and A. O. Hero III, *Learning sparse graphs under smoothness prior,* arXiv preprint arXiv:1609.03448 (2016).

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

[20] 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.

[21] M. Coutino and G. Leus, *Asynchronous distributed edge-variant graph filters,* .

[22] S. Segarra, A. G. Marques, and A. Ribeiro, *Optimal graph-filter design and applications to distributed linear network operators,* IEEE Transactions on Signal Processing **65**, 4117 (2017).

[23] L. Stankovic, D. P. Mandic, M. Dakovic, I. Kisil, E. Sejdic, and A. G. Constantinides, *Understanding the basis of graph signal processing via an intuitive example-driven approach [lecture notes],* IEEE Signal Processing Magazine **36**, 133 (2019).

III

EPILOGUE

8

CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

In this chapter, we conclude the thesis. We also mention the related contributions and the future research directions.

8.1. SUMMARY OF RESULTS

In this thesis, we have mainly focused on graph filter design and implementations. We have presented several methods to efficiently design the ARMA graph filters in the graph frequency domain. Meanwhile, we have developed several implementations for ARMA graph filters, in the graph vertex domain, including distributed and centralized methods.

Here we briefly summarize the research results:

- **Research Question 1.** *How to efficiently design graph filters without knowing the graph topology?*

  Some filter designs are independent of the graph topology (frequencies). These designs mainly focus on undirected graphs which use the Laplacian matrix (or its modifications) as the shift operator.

In Chapter 3 of this thesis, we have first extended the concept of universal design from an undirected graph to a directed graph. Since the frequencies of a directed graph are shown as real-values and complex conjugate pairs, we grid points lying in the complex unit disc and design with the adjacency basis. The well-studied FIR graph filter is designed for a directed graph in a universal way. We have analyzed the numerical results for a universal FIR graph filter design. To improve the performance, some other graph filters have been designed in a universal way in this thesis.

- **Research Question 2.** *How to efficiently exploit the potential of different graph filter structures? Also, how to obtain the best approximation accuracy through a filter with a given order?*

  In this thesis, we have mainly focused on the ARMA graph filter designs. In Chapter 3, we have illustrated the widely used form of the ARMA graph filter and some modifications, i.e., pole-zero form and partial fraction form. We have proposed an efficient method to design the ARMA graph filter universally in Chapter 4. For either the Laplacian (undirected graph) or adjacency (directed graph) matrix, the main aim of the ARMA graph filter design is to fit the response on those grid points.

  In Chapter 5, we have introduced the orthogonal polynomial basis into the design structure of the ARMA model. We compute the orthogonal basis separately for the numerator and denominator parts. The solutions for undirected and directed graphs are formulated. It is worth mentioning that the orthogonal polynomial basis also has benefits for design problems with some graph information, e.g., the topology and the graph frequencies. Moreover, in Chapter 6, the design focuses on the pole-zero form and partial fraction form of ARMA graph filters. Since the ARMA filter splits a given filter order into two parts, more freedom is introduced to the design problem. Thus, we could search for the best combination of filter orders which can obtain the best approximation accuracy.

- **Research Question 3.** *With the designed ARMA graph filters, how to implement the filter in the vertex domain?*

  This research question was mainly answered in Chapter 7. We have first briefly discussed the implementation problem of the ARMA graph filter in Chapter 3. Then, in Chapter 7, we propose several implementations for the designs in the vertex domain, including distributed and centralized methods. For the centralized implementation, we have used the (bi)conjugate

gradient method to solve the ARMA model. Meanwhile, we have also proposed the Richardson and weighted Jacobi iterations as distributed implementations for both undirected and directed graph filters. For an undirected graph, we have formulated the convergence conditions for these distributed implementations.

8.2. FUTURE RESEARCH

In this section, we provide some research directions that can be further studied in this field.

- First of all, the design methods mainly focus on a model-driven (grid points) or graph-driven (topology and graph frequencies) technique. An extension of the proposed design methods to a data-driven approach can be a direction for the future. As we discussed in Chapters 2-3, the filter designs in this thesis are solving a fitting problem between the desired response $\hat{h}$ and $\hat{g}$. For a data-driven concept, input and output graph signals are given and we try to fit the output data to the filtered input data. The data-driven design may not be suitable to handle all cases (e.g., the universal design in this thesis), while it will bring more information of the input and output data into the design.

- Furthermore, as an application, the performance of a filter bank is often influenced by the accuracy of FIR or IIR filters. Our methods bring more degrees of freedom into FIR and ARMA graph filters which may benefit the design of a filter bank. Meanwhile, our methods do not consider graphs with changing nodes and edges. Since node-variant graph filters are more appropriate for a dynamic environment, node-variant ARMA graph filters could be considered in future works.

- Extensions of the proposed orthogonal polynomial basis could be studied in future research. For our method in chapter 5, only the Forsythe orthogonal polynomial is considered. It is obvious that other types of discrete orthogonal polynomials, even some continuous orthogonal bases, can be used for the filter design which may bring some potential benefits. Moreover, designing the weight function, discussed in Chapter 5, can also bring more freedom to some specific demands.

- The graph filter implementations carried out in this thesis are independent of the process of computing the filter coefficients. In other words, we always first compute the filter coefficients in the graph frequency domain and then implement them in the vertex domain. One potential research question would be to combine the two steps. As an example, for a directed graph, the convergence condition of the distributed implementation can be included in the design procedure which may have additional advantages for the graph filters. With this design, we can draw conclusions about the convergence without knowing the graph operator.

SUMMARY

The ability to model irregular data and the interactions between them have extended the traditional signal processing tools to the graph domain. Under these circumstances, the emergence of graph signal processing has offered a brand new framework for dealing with complex data. In particular, the graph Fourier transform (GFT) lets us analyze the spectral components of a graph signal in the graph frequency domain. Based on the GFT, graph filters provide useful tools to modify or extract spectral parts in terms of different objectives, e.g., using a low-pass graph filter to construct graph signals without noise. This thesis mainly focuses on designing and implementing graph filters. Similar to traditional signal processing, we investigate two types of graph filters: finite impulse response (FIR) and infinite impulse response (IIR) graph filters. Moreover, this thesis takes both undirected and directed graphs into account for the design methods and implementations.

We discuss the mathematical descriptions and representation of the graph filter design problem. One of the main contributions of this thesis is to extend the universal design concept from undirected to directed graphs. For either the normalized Laplacian (undirected graph) or normalized adjacency (directed graph) matrix, we sample the respective expected graph frequency area resulting in some frequency grid points. With the determined grid points, we transfer the universal linear least squares (LLS) strategy of designing FIR and ARMA graph filters from undirected to directed graphs.

We propose a centralized ARMA filter design using a monomial polynomial basis. We formulate two design methods that are inspired by Prony's method and the Steigliz-McBride iterative approach. The first method minimizes a modified error, while the second approach minimizes the true error and potentially improves the approximation accuracy of that solution. We compare the per-

formance of the developed methods with other well-studied filters, such as IIR graph filters.

We extend the monomial polynomial basis to an orthogonal polynomial basis and design the related graph filter based on FIR and ARMA models. We discuss the discrete and continuous orthogonal polynomial basis and use the discrete one to design FIR graph filters. After that, we demonstrate an efficient ARMA design method with discrete orthogonal polynomials for both directed and undirected graphs. The orthogonal polynomial basis is computed separately on the numerator and denominator parts of the ARMA model.

We also consider another iterative framework of designing a special type of ARMA graph filter corresponding to the pole-zero form for both directed and undirected graphs. The filter is represented by a partial fraction belonging to a rational basis. Our approach formulates the design as a least-squares problem and recursively solves the error between the desired frequency response and the filter response.

We present a set of practical implementations for the designed ARMA graph filters in the vertex domain. We separate the methods into two categories, e.g. centralized and distributed implementations. For the latter, we also formulate the convergence conditions. We finally provide some examples to establish the performance of the proposed implementations with the designed filter coefficients.

SAMENVATTING

De mogelijkheid om onregelmatige gegevens en de interacties daartussen te modelleren heeft de traditionele signaalverwerkingsinstrumenten uitgebreid tot het grafendomein. Onder deze omstandigheden heeft de opkomst van graph signal processing (GSP) een geheel nieuw kader geboden voor de behandeling van complexe gegevens. Met name de graph Fourier transform (GFT) maakt het mogelijk de spectrale componenten van een grafensignaal te analyseren. Gebaseerd op de GFT bieden graph filters (GFs) nuttige hulpmiddelen om spectrale delen te wijzigen of te extraheren met het oog op verschillende doelstellingen, bijv. het gebruik van een laagdoorlaat GF om grafensignalen zonder ruis te construeren. Deze dissertatie richt zich voornamelijk op het ontwerpen en implementeren van GFs. Vergelijkbaar met de traditionele signaalverwerking, onderzoeken we twee soorten GFs: finite impulse response (FIR) en infinite impulse response (IIR) GFs. Bovendien houdt deze dissertatie rekening met zowel directionele als niet-directionele grafen voor de ontwerpmethoden en implementaties.

We bespreken de wiskundige beschrijvingen en representaties van het GF ontwerpprobleem. Een van de belangrijkste bijdragen van dit proefschrift is de uitbreiding van het universele ontwerpconcept van niet-directionele naar directionele grafen. Voor de genormaliseerde Laplacian (niet-directionele graaf) of de genormaliseerde adjacency (directionele graaf), bemonsteren we het verwachte frequentiegebied van net netwerk, wat resulteert in een aantal frequentie - rasterpunten. Met de vastgestelde rasterpunten brengen we de universele linear least squares (LLS) strategie voor het ontwerpen van FIR en auto-regressive moving average (ARMA) GFs over van niet-directionele naar directionele grafen.

Wij stellen een gecentraliseerd ARMA filterontwerp voor dat gebruik maakt van een monomiale polynomiale basis. We formuleren twee ontwerpmethoden die geïnspireerd zijn op de methode van Prony en de iteratieve aanpak van Steigliz-McBride. De eerste methode minimaliseert een gewijzigde fout, terwijl de tweede benadering de werkelijke fout minimaliseert en mogelijk de nauwkeurigheid van die oplossing verbetert. Wij vergelijken de prestaties van de ontwikkelde methoden met die van andere goed bestudeerde filters, zoals andere IIR GFs.

We breiden de monomiale polynomiale basis uit tot een orthogonale polynomiale basis en ontwerpen de bijbehorende GFs op basis van FIR- en ARMA-modellen. We bespreken de discrete en continue orthogonale polynomiale basis en gebruiken de discrete basis om FIR GFs te ontwerpen. Daarna demonstreren we een efficiënte ARMA-ontwerpmethode met discrete orthogonale veeltermen voor zowel directionele als niet-directionele grafen. De orthogonale polynomiale basis wordt afzonderlijk berekend op de teller- en noemerdelen van het ARMA model.

Wij beschouwen ook een ander iteratief raamwerk voor het ontwerpen van een speciaal type ARMA GF dat overeenkomt met de pool-nul vorm voor zowel directionele als niet-directionele grafen. De filter wordt voorgesteld door een partiële breuk die behoort tot een rationale basis. Onze benadering formuleert het ontwerp als een kleinste-kwadraten probleem en lost recursief de fout op tussen de gewenste frequentierespons en de filterrespons.

Wij presenteren een reeks praktische implementaties voor de ontworpen ARMA GFs in het vertex-domein. We verdelen de methoden in twee categorieën, bijvoorbeeld gecentraliseerde en gedistribueerde implementaties. Voor de laatste formuleren we ook de convergentievoorwaarden. Tenslotte geven we enkele voorbeelden om de prestaties van de voorgestelde implementaties met de ontworpen filtercoëfficiënten vast te stellen.

ACKNOWLEDGEMENTS

My supervisor Prof. Leus mentioned to me once, that the Ph.D. stage is only a very short duration in life. Now that the journey of my Ph. D. study is about to end. It surprises me that I have changed so much during the past several years. Also, I luckily have a lot of company, and I want to express the gratitude towards those who are part of this journey.

First, I would like to give my deepest appreciation to my supervisor Prof. Geert Leus. It is my honor to be his student, and I will always be thankful. I have learned a lot of things from him, such as the high standard of research, critical and analytical thinking, the essentials of writing an article. Thanks for all the patience, wisdom, and guidance. Without the valuable feedback from him, I cannot even find the right path to finish the thesis. Beyond the research supervision, he is also an easy-going person, and he helps me to grow in my personal life as well.

I am grateful to Elvin Isufi, who has been a very helpful and knowledgeable person. At the beginning of my Ph.D. research, his advice inspired me and made my life easier. Thanks for all the discussion, and I really enjoy the collaboration with him. I would also like to thank all the committee members for their insightful comments on my thesis. Moreover, I would like to acknowledge TU Delft and CSC for funding my research.

I thank all of the colleagues in the Circuit and Systems (CAS) group in the TU Delft EWI building over the past few years. I am grateful for all the lunches, events, and group outings. We enjoyed ourselves together. Thank Jamal Amini, Andreas Koutrouvelis, Bahareh Abdi, Aydin Rajabzadeh, Thomas Sherson, Pim van der Meulen, Wangyang Yu, Jie Zhang, Mario Coutino, Tarik Kazaz, Miao Sun, Matthew Morency, Alberto Natali, Jac Romme, Yan Xie, Yongchang Hu for bringing me unforgettable memories. I thank Prof. Alle-Jan van der Veen as a great

leader of the group. Thank Minaksie Ramsoekh and Irma Zomerdijk for taking care of the paperwork.

Life beyond research in Delft is amazing. I thank Mei Liu, Juan Yan, and Yan Song for all their love and support. Especially, thanks for including me in the 'tangerine' group, and for all the moments we shared. Thank Shizhe Zhang, Qiang Liu, Jia Yan for being a great part of my life in Delft.

Last but not least, I would like to thank my family! Thank my parents for all the supports. Thank my grandparents for the unconditional love. In the end, I want to thank my husband Bo Liu who accompanied me on this journey.

Jiani Liu

Den Haag, June 2021

CURRICULUM VITAE

Jiani Liu was born in 18th January, 1991 in Shaanxi, Xi'an, China. She received the Bachelor of Science (B.Sc.) degree in information countermeasure technology, in 2013 from the Northwestern Polytechnical University (NWPU), Shaanxi, Xi'an, China. In 2016, she received the Master of Science (M.Sc.) degree in Underwater Acoustic Engineering, from NWPU, China. She received the Outstanding Graduate Award (in 2013) and National Scholarship of China (in 2012). In 2015, she started her Ph.D and joined the circuits and systems (CAS) group at the Faculty of Electrical Engineering, Mathematics and Computer Science (EEMCS) of the Delft University of Technology. Her research interests contain the areas of graph signal processing, statistical signal processing, deep learning, and mathematical modeling.