Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia

COPPE UFRJ

NTNU Innovation and Creativity

MODELING AND LEARNING STRATEGIES FOR GRAPH SIGNAL PROCESSING

Vitor Rosa Meireles Elias

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia Elétrica e como parte do acordo de duplo diploma de doutorado entre a Universidade Federal do Rio de Janeiro e a Universidade Norueguesa de Ciência e Tecnologia.

Orientadores: Wallace Alves Martins
Stefan Werner

Rio de Janeiro
Agosto de 2021

MODELING AND LEARNING STRATEGIES FOR GRAPH SIGNAL PROCESSING

Vitor Rosa Meireles Elias

RIO DE JANEIRO, RJ – BRASIL
AGOSTO DE 2021

Elias, Vitor Rosa Meireles

Modeling and Learning Strategies for Graph Signal Processing/Vitor Rosa Meireles Elias. – Rio de Janeiro: UFRJ/COPPE, 2021.

XVII, 163 p.: il.; 29, 7cm.

Orientadores: Wallace Alves Martins
Stefan Werner

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2021.

Referências Bibliográficas: p. 77 – 90.

1. Adaptive learning. 2. Graph signal processing. 3. Kernel least mean squares. 4. Kernel regression. 5. Light-field compression. I. Martins, Wallace Alves *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

To my parents.

Acknowledgments

The journey to becoming a doctor could not have been completed alone. First, I thank my family, especially my parents, Sandra and Roberto, who have supported me through all these years. I am happy to share all the good things that this path has provided, and I am immensely grateful to have them by my side through the difficult moments.

I also thank all the friends I have made along the way, who walked the path of science with me and helped me with many academic and personal matters. Starting with the great friends UFRJ has given me, I thank Lucas Arrabal, Isabela Apolinário, Maurício Costa, Luis Felipe Velloso, and all the other amazing people from the Signals, Multimedia, and Telecommunications Laboratory (SMT). Moving to Trondheim, I owe a debt of gratitude to Ashkan Moradi and Hannah Kriesell for their utmost kindness and for being part of some of the greatest moments of my life. I thank Karina Enoksen for helping me settle in a new city and presenting me with a bit of the beautiful Norwegian culture. I thank Vinay Gogineni for the fruitful collaboration. I also thank all the friendly people from the Department of Electronic Systems (IES) and the other departments at the Faculty of Information Technology and Electrical Engineering, NTNU.

I would like to express my sincere gratitude to my personal friends, who have endured my lack of social life. In particular, I would like to thank Vitor Borges, Vitor Tavares, Vitor Santos, Jefferson Oliveira, Rodrigo Salgado, and Mariana Rocha for all the encouragement during this entire journey, even though it meant we would not be able to be together as frequently.

I am grateful to all the professors with whom I have had the opportunity to work and who have taught me so much during my Ph.D. In particular, I owe a great debt of gratitude to Professors Paulo Diniz and Marcello Campos, from the SMT/UFRJ, and José Apolinário Jr., from the Military Engineering Institute (IME), Brazil, for believing in me and for giving me great opportunities.

I would also like to thank the members of the assessment committee of this thesis, Professor Geert Leus, Professor Felix Albu, Assistant Professor Roula Nassif, Professor Kimmo Kansanen, and Professor Mariane Petraglia, whose ideas, revisions, and expertise were fundamental for the construction and improvement of this work.

Finally, I thank my supervisors. I thank Professor Wallace A. Martins for being immensely supportive and for being open for conversations and for some of my strange ideas since day one. I am grateful to have worked with someone who holds on to science with such diligence and brilliance. I thank Professor Stefan Werner for all the technical expertise and sharpness put into our research. His contributions were fundamental for our publications and for this thesis.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

ESTRATÉGIAS DE MODELAGEM E APRENDIZADO PARA PROCESSAMENTO DE SINAIS EM GRAFOS

Vitor Rosa Meireles Elias

Agosto/2021

Orientadores: Wallace Alves Martins
Stefan Werner

Programa: Engenharia Elétrica

Esta tese propõe métodos para modelagem e aprendizado sobre grafos. Primeiro tratamos a construção de grafos adequados para compressão de *light fields*, explorando a concentração de energia provida pela transformada de Fourier em grafos (GFT). Nós propomos métodos para construir uma matriz de adjacência esparsa, consideravelmente reduzindo a quantidade de dados necessária para a representação do grafo. Uma segunda proposta é a matriz de adjacência estendida obtida pela introdução de novas arestas a uma matriz de adjacência esparsa inicial usando distâncias de difusão. Definimos a GFT dependente de escala, que revela informação espectral adicional sobre sinais em grafos e mostramos que ferramentas de processamento de sinais em grafos se beneficiam da informação adicional.

No escopo de aprendizado sobre grafos, esta tese introduz a filtragem não-linear sobre grafos, que consiste de uma não-linearidade aplicada a uma combinação de versões deslocadas do sinal de entrada no grafo. Para identificar os parâmetros do filtro, nós primeiro propomos o algoritmo *kernel least mean squares* sobre grafos centralizado (GKLMS). Nós usamos *coherence-check* e *random Fourier features* para reduzir o tamanho do dicionário. Usando a estrutura do grafo, nós propomos o *kernel least mean squares* de difusão no grafo (GDKLMS) completamente distribuído. Adicionalmente, propomos uma metodologia baseada em regressão *kernel* considerando o caso em que o sinal de entrada não é um sinal de grafo. Nós propomos algoritmos em lote e *online* com complexidade computacional reduzida e desempenho competitivo quando comparados a metodologias do estado-da-arte. Além disso, nós fornecemos análise teórica de convergência de primeira e segunda ordem para todos os algoritmos *online* propostos para aprendizado sobre grafos.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

MODELING AND LEARNING STRATEGIES FOR GRAPH SIGNAL PROCESSING

Vitor Rosa Meireles Elias

August/2021

Advisors: Wallace Alves Martins
Stefan Werner

Department: Electrical Engineering

This thesis proposes methods for modeling and learning over graphs. We first address the construction of graphs suitable for compressing light-field images, leveraging the energy concentration provided by the graph Fourier transform (GFT). We propose methods to construct a sparse adjacency matrix, considerably reducing the amount of data required for graph representation. A second proposal is the extended adjacency obtained by introducing new edges to an initial sparse adjacency matrix using diffusion distances. We define the scale-dependent GFT, which reveals additional spectral information of graph signals, and we show that graph signal processing tools benefit from the additional information.

In the scope of learning over graphs, this thesis introduces nonlinear graph filtering, which consists of a nonlinearity applied to a combination of graph-shifted versions of the input graph signal. To identify the parameters of filter, we first propose the centralized graph kernel least mean squares (GKLMS) algorithm. We use coherence-check and random Fourier features to reduce the dictionary size. Using the graph structure, we propose the fully distributed graph-diffusion kernel least mean squares (GDKLMS) algorithm. Additionally, we propose a methodology based on kernel regression, considering the case where the input signal is not a graph signal. We propose both batch-based and online algorithms with reduced computational-complexity and competitive performance when compared to state-of-the-art methodologies. Moreover, we provide first- and second-order theoretical convergence analysis for all the proposed online algorithms for learning over graphs.

Contents

List of Figures

List of Tables

Abbreviations and Symbols

Abbreviations

CC	Coherence check
DD	Diffusion distance
DM	Diffusion map
DSP	Digital signal processing
DSPG	Digital signal processing on graphs
GDKLMS	Graph diffusion kernel least mean squares
GFT	Graph Fourier transform
GSO	Graph-shift operator
GSP	Graph signal processing
HEVC	High efficience video coding
IDCT	Inverse discrete-cosine transform
IGFT	Inverse graph Fourier transform
KLMS	Kernel least mean squares
KRG	Kernel regression over graphs
LF	Light field
LMS	Least mean squares
MSE	Mean squared error
NMSE	Normalized mean squared error
RBF	Radial-basis function

RFF	Random Fourier features
RKHS	Reproducing kernel Hilbert spaces
RLS	Recursive least squares
SG	Stochastic gradient
sGFT	scale-dependent graph Fourier transform
SMSE	Steady-state mean squared error
WSN	Wireless sensor network

Symbols

$\bar{\mathbf{A}}(t)$	Extended-adjacency matrix
$\bar{\mathbf{L}}(t)$	Laplacian matrix at diffusion scale t
$\mathbf{0}_K$	Length-K vector with all entries equal to zero
$\cap$	Set-intersection operator
$\mathbf{C}$	Cyclic-shift matrix
Δ	Continuous Laplace operator
Δ_{G}	Graph-based Laplace operator
$\mathbf{I}_M$	$M \times M$ identity matrix
$\mathbb{C}$	Set of complex numbers
$\mathbb{R}$	Set of real numbers
$\mathbf{A}$	Adjacency matrix
$\mathbf{h}$	Function representation in the random Fourier features space
$\mathbf{L}$	Laplacian matrix
$\mathbf{r}_n$	Shifted-input vector
$\mathbf{x}_n$	Generic graph signal
$\mathbf{z}_n$	Shifted-input vector in random Fourier features space
$\mathcal{E}$	Set of edges

$\mathcal{G}$	Graph
$\mathcal{N}$	Neighborhood
$\mathcal{T}$	Unit-shift operator
$\mathcal{V}$	Set of vertices
μ	Step size of online algorithms
$\oplus$	Direct-sum operator
$\otimes$	Kronecker-product operator
σ_{RBF}	Parameter of the radial-basis function kernel
$\|\cdot\|$	Euclidean norm
$\|\cdot\|_{\mathrm{F}}$	Frobenius norm
$\lvert\mathcal{V}\rvert$	Cardinality of the set $\mathcal{V}$
$D_t^2(v_i, v_j)$	Diffusion distance between v_i and v_j
$e_{i,j}$	Graph edge from v_j to v_i
v_i	Graph node indexed by i
$\mathrm{E}[\cdot]$	Expectation operator
$\mathrm{mat}(\cdot)$	Vector-to-matrix operator
$\mathrm{TV}(\cdot)$	Total variation
$\mathrm{TV}_{\mathrm{G}}(\cdot)$	Total variation over the graph
$\mathrm{vec}(\cdot)$	Matrix-to-vector operator

Chapter 1

Introduction

Data from multidimensional variables, defined over networked structures, are continually generated, stored, and processed in systems related to several technology areas. In the current stage of the information era, the necessity of dealing with data from enormous networks, such as social networks, sensor networks, transport networks, among many others, presents a challenging task. Digital signal processing (DSP) is a broad area of engineering dedicated to processing data, usually assuming that these data are defined over well-structured and regular domains. Many of the conventional DSP techniques available today are tailored for processing and analyzing one-dimensional signals in time or frequency[1] — the extension to higher-dimensional domains deals with signals defined over regular grids, such as images. One fundamental characteristic of conventional DSP is that there is an ordering in the observed data that is inherent to the domain, such that one can infer if a sample comes before or after another. The regularity of DSP domains facilitates the development of signal processing tools. However, regular domains cannot capture more sophisticated relations between data samples, which limits the applicability of conventional DSP tools.

Consider, for example, the case of analyzing how many mail orders a subdistrict receives in a month, assuming the supply chain company wants to improve the allocation of delivery operators across the city. This problem can be regarded as a prediction problem, where the company needs to predict the number of orders in a given month to allocate the operators properly. One may define a monthly-sampled time series a_m, with $m \in \{1, 2, \ldots, 12\}$, where a_m indicates how many mail orders that subdistrict received in month m. Conventional DSP tools are perfectly fit for the manipulation of this type of data. Consider that the same data type is available for other subdistricts. One can then jointly analyze the different subdistrict time

[1]We refer to conventional DSP as the study of signals in time domain (one-dimensional signals), space domain (possibly multidimensional signals), and their corresponding frequency- and other transform-domain representations.

series with multidimensional DSP, which allows inferring characteristics of the data directly from the time series. Nonetheless, suppose the company can access other information about the subdistricts, such as the number of packages sent between two subdistricts over a time interval. In that case, this information could also improve the data analysis and the prediction process. The relations between subdistricts confer an aspect of a connected network. Conventional DSP techniques are not adequate to naturally deal with this information, and graphs emerge as an alternative domain where network-related information can be included in the data processing. The fact that conventional DSP cannot properly handle networked data composes one of the motivations for the field of research referred to as digital signal processing on graphs (DSPG) [1] or graph signal processing (GSP) [2–4].

GSP is a relatively new field, with most of its framework being developed over the past decade. Its growing importance is due to its applicability to networked data processing, as connectivity between real-world elements progressively increases with the advent of the internet-of-things, sensor networks, and better communication technologies [5–7]. GSP employs graph-structural information to model, process, and analyze signals defined over graph nodes [1–4]. By associating real-world network elements with graph nodes and encoding their interrelations through graph edges, GSP leverages the graph structure to process or analyze the network data, modeled as a graph signal. However, in contrast to conventional DSP, GSP deals with irregular and more complex domains that can vary drastically according to the application. Real networks and their corresponding data come in vastly different shapes and applications, ranging from genetic interaction networks [8] and the human brain [9] to sensor networks and smart cities [6]. The variety in shapes and applications demands additional attention when generating models and developing tools for processing networked data. Hence, a significant amount of research is still dedicated to defining the fundamentals of the emerging field of GSP [4, 7].

Similar to traditional DSP techniques, the basic building block in GSP is the graph-shift operation. This operation shifts a graph signal in the graph domain according to a graph-shift operator (GSO) matrix, which captures node interconnections [1, 4, 7]. For instance, the graph Fourier transform (GFT) is defined as the signal expansion in terms of the eigenbasis of the GSO. In contrast to conventional DSP, where the regular and ordered domain induces a straightforward definition of the shift operation, the GSP literature contains several GSO definitions that suit different applications [1, 2, 4, 6, 7, 9–23]. The two most commonly used GSOs are the adjacency matrix of the graph [1] and the graph Laplacian [2]. The diversity of applications and the fact that GSP is a field still in its early stages call for research on how the translation from real networks into graphs affects the performance of GSP tools. In this regard, part of the contribution of this thesis is the investigation of techniques to improve the modeling of real-world problems as GSP applications.

Another key research area in GSP is modeling unknown relations between a reference signal and a target signal, usually referred to as an input-output pair [24–33]. In conventional DSP, this modeling can be achieved using digital filters, which operate on shifted versions of the input signals. In the particular case of linear networks, using the aforementioned GSO definitions, the graph-shifted signal on a given node is a linear combination of adjacent node signals, where the weights relate to the edge values. This resemblance to DSP has sparked the development of a vast amount of GSP counterparts of methods related to spectral analysis [11, 14, 34–38] and traditional time-series analysis [39, 40]. Several works deal with adaptive learning of graph filters, e.g., [24–28]. These methods were later extended to multitask graphs [29, 30]. These previous works adopt the ideas of linear adaptive networks [41, 42] to estimate the graph filter through in-network processing. However, linear models cannot accurately represent many real-world systems that exhibit more sophisticated input-output relations. Prominent examples include the relations between air pressure and temperature [43], and wind speed and generated power in wind turbines [44].

Different approaches for nonlinear system modeling can be found in the literature [45–53]. In particular, approaches based on kernel methods [51, 52] have gained popularity due to their efficacy and mathematical simplicity [31–33, 53–65]. There is extensive literature on function estimation using kernel methods for both single- and multi-node networks, e.g., [31–33, 54–69]. However, most of the previous approaches inherit the well-known scaling issue of kernel methods [63, 64], since the model dimension increases with the number of training samples, which increases with the network size and with time.

This thesis contributes to the development of GSP methods by investigating approaches for learning nonlinear input-output relations over graphs that overcome the complexity issues associated with kernel methods. In particular, we introduce the concept of nonlinear graph filters that operate on graph-shifted versions of the input signal, along with adaptive methods based on reproducing kernel Hilbert spaces (RKHS) [33, 52, 70] for identifying nonlinear graph filters. These methods explore the graph structure to learn the graph filter distributively. Additionally, a methodology for kernel regression over graphs (KRG) is developed for learning input-output relations over graphs, acknowledging cases where the input signal is not a graph signal.

1.1 Objectives

The present research addresses the applications and tools for the digital signal processing of signals defined over graphs. In particular, we identify the following needs regarding the GSP framework: (i) The need to improve the translation of real-world problems into graph models suitable for GSP tools. Specifically, developing methods that can better leverage the available information on the network structure. (ii) The need to develop methods that learn sophisticated relations between signals defined over graphs, given that most available approaches are limited to linear models. (iii) The need to investigate real-world scenarios where GSP tools can be effectively applied, identifying structured data that can be defined over a suitable graph and whose relations between elements can be explored. In summary, this thesis focus on the following research objectives:

T1: Proposing an augmentation method for the GSO model to improve the performance of GSP tools;

T2: Developing efficient tools for learning nonlinear input-output relations over graphs based on graph filtering and regression methods.

T3: Investigating different applications that can benefit from the proposed GSP-based methodologies.

1.2 Methodology

This thesis investigates theoretical techniques to contribute to the GSP framework and addresses the research topics presented in Section 1.1. The methods proposed here build upon the fundamentals of GSP and improve over these with theoretical proposals for GSP modeling and for learning over graphs. We present motivations in line with the needs of the GSP literature, and the methods proposed in this thesis are compared to other state-of-the-art approaches. Theoretical analyses are provided to support the proposed concepts. The validation of the proposed methodology is conducted through numerical experiments using both synthesized and real datasets, with applications in different scenarios.

1.3 Thesis Contributions

First, we study the specific application of GSP to compress light-field images, addressing both **T1** and **T3**. The purpose is to showcase the basic concepts of GSP and motivate the need for research dedicated to modeling real-world problems into the GSP scope. Light-field imaging is a technology for capturing images, building upon conventional digital photography, that presents challenging tasks related to the amount of data generated. This work proposes methods for modeling the blocks of pixels from a light-field as graphs and uses the GFT to compress the images, leveraging the energy-concentration provided by the Fourier analysis of the corresponding graph signals. A second work that directly addresses **T1** is the proposal of an augmentation methodology for the adjacency of networks. The problem tackled in this research starts with the assumption that no information other than an initial adjacency matrix is known about the network. The proposed idea is that, by finding hidden pairwise node relations not shown in the initial adjacency matrix, the spectral analysis of the graph signal is modified, and the performance of GFT-based tools can be improved. The adjacency of graph nodes is augmented with the addition of new edges following a criterion based on a Markov relation imposed between nodes, such that it is possible to attribute relation between pairs of nodes that are not connected initially. This augmentation of the adjacency matrix directly affects the GFT, and we show that this effect can be explored to improve the performance of GSP tools that utilize graph-spectral information. The proposed methodology is tested for anomaly detection in networked data in both synthesized and real networks.

To address **T2**, two distinct research lines are pursued. In the first one, we generalize the theory of conventional nonlinear filtering and linear graph filtering to propose nonlinear graph filters. Nonlinear graph filters consist of a nonlinearity applied to a combination of graph-shifted versions of the input signal. The nonlinearity is modeled in reproducing kernel Hilbert spaces. For learning the nonlinear graph-filter parameters, we derive GSP-based kernel least mean squares algorithms. In addition to a centralized implementation for the nonlinear graph filters, we also explore diffusion over networks to derive distributed nonlinear graph filters. The proposed methods enjoy reduced computational complexity when compared to conventional algorithms based on kernel methods, and scale well for large networks and datasets. We conduct a detailed performance study of the proposed algorithms, and we derive the convergence conditions in the mean and mean-square senses.

The second research line to address **T2** is efficient batch-based and online strategies for kernel regression over graphs (KRG). In contrast to the previously proposed nonlinear graph filters, the proposed KRG algorithms do not require the input signal to be a graph signal. Similar to the previous approach, we proposed scalable algorithms with reduced computational complexity. The proposed batch-based implementation greatly reduces the complexity when compared to previous state-of-the-art KRG implementations. Additionally, we derive two online strategies: the mini-batch gradient KRG (MGKRG) and the recursive least squares KRG (RL-

SKRG). The stochastic-gradient KRG (SGKRG) is introduced as a particular case of the MGKRG. We provide a detailed stability analysis of the online algorithms and a discussion on complexity.

The research topic **T3** is addressed in all the works above. We first investigate the application of GSP methods for compressing light-field images. The proposed extended-adjacency methodology is employed for anomaly detection using graph-spectrum information. We investigate the modeling and estimation of humidity data from temperature data collected by a sensor network inside a working laboratory environment. Using the proposed KRG methodology, we tackle various applications, including predicting temperatures in a network of weather stations, estimating brain-activity intensity in different regions of the brain, and reconstructing images with corrupted pixels.

1.3.1 List of Publications

The following works were conducted by the author of the dissertation in line with the research objectives presented in Section 1.1. These works are documented in papers **P1** to **P8** and comprise the contributions listed in Section 1.3. The list is composed by eight papers, of which seven were published or accepted for publication, and one was submitted during the course of the Ph.D.

- **P1**: [71] V. R. M. Elias and W. A. Martins, "Graph Fourier transform for light field compression," in *Simpósio Brasileiro de Telecomunicações e Processamento de Sinais*, pp. 881–885, Sept. 2017.

- **P2**: [72] V. R. M. Elias and W. A. Martins, "On the use of graph Fourier transform for light-field compression," *Journal of Communication and Information Systems*, vol. 33, pp. 92–103, May 2018.

- **P3**: [73] V. R. M. Elias, W. A. Martins, and S. Werner, "Diffusion-based virtual graph adjacency for Fourier analysis of network signals," in *Simpósio Brasileiro de Telecomunicações e Processamento de Sinais*, pp. 1–5, Dec. 2020.[2]

- **P4**: [74] V. R. M. Elias, W. A. Martins, and S. Werner, "Extended adjacency and scale-dependent graph Fourier transform via diffusion distances," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 6, pp. 592–604, Aug. 2020.

- **P5**: [75] V. C. Gogineni, V. R. M. Elias, W. A. Martins, and S. Werner, "Graph diffusion kernel LMS using random Fourier features," in *Asilomar Conference on Signals, Systems, and Computers*, pp. 1–5, Nov. 2020.

[2]Paper awarded with the *Best Paper Award.*

- **P6**: [76] V. R. M. Elias, V. C. Gogineni, W. A. Martins, and S. Werner, "Adaptive graph filters in reproducing kernel Hilbert spaces: Design and performance analysis," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 7, pp. 62–74, 2021.
- **P7**: [77] V. R. M. Elias, V. C. Gogineni, W. A. Martins, and S. Werner, "Kernel regression on graphs in random Fourier features space," in *International Conference on Acoustics, Speech, and Signal Processing*, pp. 5235–5239, 2021.
- **P8**: [78] V. R. M. Elias, V. C. Gogineni, W. A. Martins, and S. Werner, "Kernel regression over graphs using random Fourier features," Submitted to *IEEE Transactions on Signal Processing*, pp. 1–12, 2021.

1.3.2 Other Contributions

In addition to the papers listed in Section 1.3.1, the following short course and book chapter were produced covering fundamentals of GSP and its applications, in line with the research objectives of this thesis:

- **Book chapter**: J. B. Lima, G. B. Ribeiro, W. A. Martins and V. R. M. Elias, "Processamento de sinais em grafos: Fundamentos e aplicações," Chapter 2 in *Livro de Minicursos SBRT 2018* (P. R. L. Júnior, E. C. Gurjão, R. D. Gomes and J. S. Rocha, eds.), Editora IFPB, 2018.
- **Short course**: J. B. Lima, G. B. Ribeiro, W. A. Martins and V. R. M. Elias, Graph signal processing: Fundamentals and applications (*original title: Processamento de sinais em grafos: Fundamentos e aplicações*). Brazilian Symposium on Telecommunications and Signal Processing, 2018. (duration: 4h)

The codes of the simulations presented in this thesis, corresponding to the numerical experiments conducted in papers **P1**-**P8**, are available at `https://github.com/vitor-elias/thesis_codes`.

1.4 Thesis Organization

This thesis begins by providing the fundamentals of GSP and the background necessary for the research developed presented here. Chapter 2 introduces the fundamentals of graphs, graph theory, and GSP. Chapter 3 applies these fundamentals of GSP, specifically the graph Fourier transform, for light-field compression. Chapter 4 proposes a methodology for augmenting the adjacency matrix and presents a scale-dependent graph-frequency analysis, which can be explored by GSP tools that use the graph spectrum. Strategies for learning over graphs are presented in Chapters 5 and 6. Chapter 5 introduces nonlinear graph filters and presents an adaptive distributed methodology for learning the filter parameters. In Chapter 6, an alternative methodology for learning over graphs is presented, based on kernel regression, suitable for scenarios where the reference signal is not defined over the graph. Finally, 7 presents the conclusions and final remarks, and future works for the topics discussed in the thesis.

Chapter 2

Graph Signal Processing and its Approaches

This chapter reviews essential concepts and definitions of graphs and graph theory used throughout the thesis. We introduce the fundamentals of GSP and discuss topics related to our research objectives. In Sections 2.1 to 2.3, we formalize the concepts of graphs, graph signals, and graph structures. In Section 2.4, the graph-shift operator is presented. Sections 2.5 and 2.6 present two of the main approaches for GSP in the literature. In Section 2.7, we discuss the frequency interpretations for the different GSP approaches. Finally, Section 2.8 presents the final remarks for this chapter.

2.1 Introduction to Graphs

A graph is a mathematical structure used to model a set of elements and their pairwise relations. Here, a graph is denoted by $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{v_1, \ldots, v_K\}$ is the set of vertices or nodes that represent the elements or objects that compose the network structure.[1] The set $\mathcal{E} = \{e_{1,1}, \ldots, e_{K,K}\}$ is the set of edges, with $|\mathcal{E}| = K^2$. Edges indicate (possibly complex) node interconnections, in the sense that an edge $e_{i,j} \neq 0$ indicates that vertex v_j influences vertex v_i. When $e_{i,j} = 0$, one can consider that there is no edge connecting nodes v_i and v_j. Usually, edges are binary, i.e., $e_{i,j} \in \{0, 1\}$. If there is a non-binary value associated with an edge by a mapping $w : \mathcal{E} \to \mathbb{C}$, the graph is said to be a weighted graph. For simplicity, we assume that edges in weighted graphs may present non-binary values, omitting the mapping w. In order to facilitate the manipulation and study of graphs, it is often convenient to represent the set of edges as a square adjacency matrix $\mathbf{A} \in \mathbb{C}^{K \times K}$, for which the value of element $A_{i,j}$ is equal to that of edge $e_{i,j}$. Hence, a graph may

[1]Throughout the text, the terms vertex (vertices) and node (nodes) will be employed interchangeably with each other

also be represented as the pair $\mathcal{G} = \{\mathcal{V}, \mathbf{A}\}$. Both notations will be used throughout the remainder of this text. In this work, unless stated otherwise, we consider only graphs for which no multiple edges connecting two vertices nor self-loops, such that $e_{i,i} \neq 0$, are allowed.

Graphs may be classified as directed or undirected graphs. If $e_{i,j} = e_{j,i}$, $\forall i, j$, i.e., if the influence of vertex v_j over v_i is equal to the influence of i over j for every pair (i, j), the graph is undirected, since the direction of the edge is not relevant. In this case, the adjacency matrix $\mathbf{A}$ is symmetric. Otherwise, if the direction of the edge matters to the connection between some (at least one) pair of nodes in the graph, such that $e_{i,j} \neq e_{j,i}$, the graph is said to be a directed graph or digraph. Directed graphs may be further classified as oriented if only one directed edge exists for any connected pair of vertices in the graph, that is, if there is only one way between any two connected vertices. An example of undirected and directed graphs is shown in Figure 2.1, with $K = 5$ vertices.

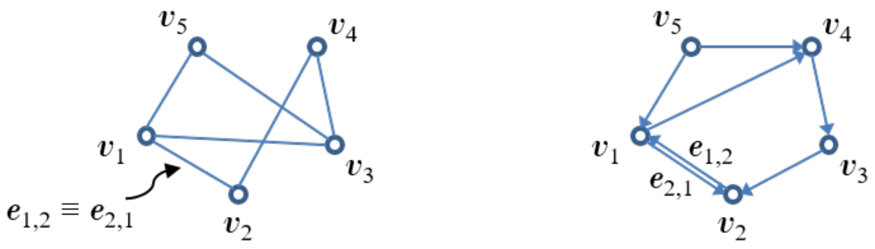


Figure 2.1: Example of undirected (left) and directed (right) graphs, for a set with $K = 5$ vertices.

Paths between any two vertices in a graph are sets of edges connecting the two vertices. The size of the shortest path between two vertices is often used to denote the distance between them. The maximum distance between two nodes in a graph is the diameter of the graph. The distance may also be weighted by the values of the edges in the case of weighted graphs. If two vertices v_i and v_j are adjacent, there is a path with a single edge between them, i.e., $e_{i,j} \neq 0$ or $e_{j,i} \neq 0$. If there exists a path between any two vertices of $\mathcal{V}$, the graph is said to be connected. For the undirected case, if every vertex is adjacent to all others, such that $e_{i,j} = e_{j,i} \neq 0$, $\forall i, j$ the graph is a complete graph. Analogously, for the directed case, if there is a pair of non-zero directed edges between every two vertices, the graph is referred to as a complete digraph. Figure 2.2 shows an example of complete and incomplete graphs; in fact, this example also illustrates a disconnected graph, since there are unreachable vertices depending on the starting vertex.

The neighborhood $\mathcal{N}_k$ of a vertex k is the set of all vertices in $\mathcal{V}$ that are adjacent to k. For an unweighted and undirected graph, the degree $\deg(k)$ of a node v_k is

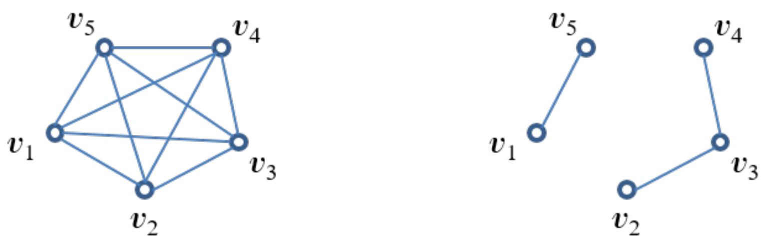


Figure 2.2: Example of complete (left) and incomplete and disconnected (right) graphs, for a set with $K = 5$ vertices.

the number of edges connected to k, which is equal to the number of vertices in $\mathcal{N}_k$. Equivalently,

$$\deg(k) = \sum_{i=1}^{K} e_{i,k} = \sum_{i=1}^{K} e_{k,i}. \tag{2.1}$$

Note that one can also consider the weighted degree resulting from the non-binary edge values.

A discrete Laplace operator, analogous to the continuous Laplace operator, is defined for graphs [79]. For a function $f(\mathbf{x})$ in an m-dimensional Euclidean space, the continuous Laplace operator, denoted by Δ, is a second-order differential operator given by

$$\Delta f(t) \triangleq \sum_{i=1}^{m} \frac{\partial^2 f}{\partial x_i^2}. \tag{2.2}$$

The discrete graph Laplacian may be defined in several forms [79]. For a function $\gamma(v_n)$ defined on the vertices of an unweighted graph, such that $\gamma : \mathcal{V} \to \mathbb{R}$, the traditional definition of the graph Laplacian Δ_{G} is given by

$$(\Delta_{\mathrm{G}}\gamma)(v_i) = \sum_{v_j \in \mathcal{N}_i} (\gamma(v_i) - \gamma(v_j)). \tag{2.3}$$

If the graph is weighted, the graph Laplacian is such that

$$(\Delta_{\mathrm{G}}\gamma)(v_i) = \sum_{v_j \in \mathcal{N}_i} A_{i,j}(\gamma(v_i) - \gamma(v_j)). \tag{2.4}$$

Then, for an undirected graph $\mathcal{G} = \{\mathcal{V}, \mathbf{A}\}$, we can define the symmetric Laplacian matrix $\mathbf{L}$ as

$$\mathbf{L} = \mathbf{D} - \mathbf{A}, \tag{2.5}$$

and the symmetric normalized Laplacian matrix, given by

$$\mathbf{L}^{\mathrm{sym}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}. \tag{2.6}$$

2.2 Signals on Graphs

For a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, a finite-duration complex-valued signal on a graph is given by the mapping $s : \mathcal{V} \to \mathbb{C}$. That is, a signal is a function whose discrete domain is given by the set of vertices $\{v_1, \ldots, v_K\}$ of the graph, equivalently to DSP cases where a signal is a function of time instants n of a discrete-time sequence $\{0, \ldots, N-1\}$. This equivalence is depicted in Figure 2.3. It is practical to represent a signal on a graph as a vector $\mathbf{s} \in \mathbb{C}^K$, such that the kth entry of the vector is given by $s(v_k)$.

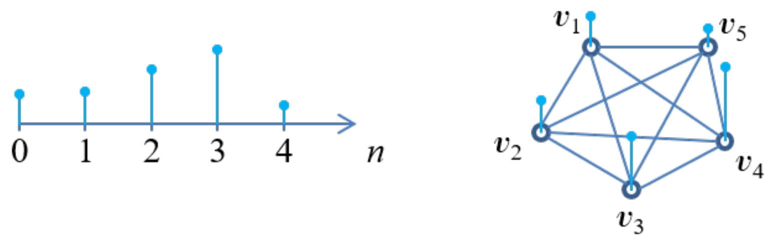


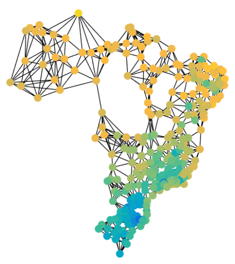
Figure 2.3: The definition of a signal on a graph is similar to that of signals in time, if the set of vertices of the graph is taken as the signal domain.

The discrete-time domain with corresponding elements $n \in \{0, \ldots, N-1\}$ is a well-structured and regular domain with straightforward relation between its elements. In fact, any two points $n_1, n_2 \in \{0, \ldots, N-1\}$ can be compared, as $n_1 > n_2$, or $n_1 < n_2$, or $n_1 = n_2$, and there is an order associated with the domain, since for any $n \in \{0, \ldots, N-1\}$, the inequalities $0 \leq n \leq N-1$ hold. In contrast, the structure of a graph depends on its edges, and, thus, the definition of a signal over a graph allows for a more generic domain. In fact, for several applications, the representation of a signal over a domain such as time is not practical, and alternatives are required.

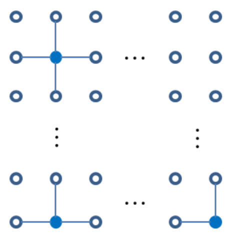
2.3 Typical Graph Structures and the Adjacency Matrix

A key feature of GSP is that graphs are suitable for modeling a range of different networks, since the graph domain can be tailored for each network. Indeed, different graph topologies are better suited for different networks and applications. Figure 2.4 presents different graph types that are commonly found in GSP applications.

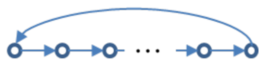
Figure 2.4a shows a nearest-neighbor (NN) graph. NN graphs are based on the distance between nodes, such that nodes closer to each other are related by edges with larger weights. In contrast, small-valued edges indicate nodes that are far



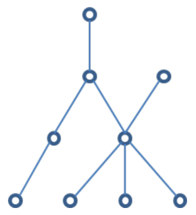
(a) Example of sensors distributed across Brazil.



(b) Grid-like nearest-neighbor graph that can be used to represent images.



(c) Directed cyclic graph that generalizes time domain.



(d) Example of tree graph.

Figure 2.4: Illustrations of some common graph types.

from one another. A classical application of NN graphs is modeling wireless sensor networks (WSNs). Sensor networks, also referred to as sensor graphs, are naturally conducive to graph representation. We associate the sensors with the graph nodes. A sensor network consists of a set of physically distributed sensors that measure local signals, as depicted in Figure 2.4a, where sensors are distributed across the territory of Brazil. Given power constraints, sensor-communication capabilities usually decay with distance, such that the NN graph is a suitable model. A common way of computing edge values in NN graphs is using a Gaussian or radial-basis function (RBF) kernel [1, 2]. Letting $\mathbf{p}_{v_i}$ denote the position of the ith node, the adjacency matrix is constructed as

$$A_{i,j} = \exp\left(\frac{-\|\mathbf{p}_{v_i} - \mathbf{p}_{v_j}\|^2}{2\sigma_{\text{RBF}}^2}\right), \tag{2.7}$$

where $\|\mathbf{p}_{v_i} - \mathbf{p}_{v_j}\|$ is the Euclidean distance between nodes v_i and v_j, and σ_{RBF} is an adjustment parameter that depends on the application. This parameter avoids that networks where all distances are relatively large make all edges close to zero.

From sensor graphs, another typical graph structure is defined to confer sparsity to the matrix $\mathbf{A}$ generated with (2.7). One can consider that edges with small values (when compared to the values of other edges in the same graph) are forced to zero, defining a threshold γ such that if $A_{i,j} < \gamma$, then $A_{i,j} := 0$. Alternatively, to construct a more regular structure, one can limit the size of the neighborhood of a sensor. The κ-nearest-neighbor (κNN) model defines a structure where each sensor is connected to the κ sensors in its neighborhood with largest edge values according to (2.7). For an undirected graph, a node can be connected to more than κ nodes after all nodes are connected to their nearest-neighbors. For example, if node v_1 is connected to its $\kappa = 3$ nearest-neighbors, v_2, v_3, v_4, and it is also a κ-nearest-neighbor of v_5, node v_1 is connected to four nodes. The graph structure on Figure 2.4a consists of a κNN graph for $\kappa = 5$.

Based on the κNN model, the nearest-neighbor (NN) image model [80] is a graph representation for images, presented in Figure 2.4b. Assuming an image as a regular grid of pixels, this model considers that each pixel in an image is associated with a vertex in a graph. Each pixel is connected to other pixels at minimal distance. A pixel in the diagonal direction, for instance, is not in the neighborhood of a given pixel.

Given the NN image model, one can observe that graphs are not only useful for modeling irregular structures. In fact, graphs can generalize traditional regular domains such as the finite-duration discrete-time domain, as shown in Figure 2.4c. This is achieved by directly associating vertices $v_k \in \{v_1, \ldots, v_K\}$ with time instants $n \in \{0, \ldots, N-1\}$, using directed edges to express the ordering of time, and connecting v_K to v_1 to express the cyclic property of the domain. The adjacency matrix associated with this graph is equal to the $K \times K$ *cyclic-shift matrix*

$$\mathbf{A} = \mathbf{C} \triangleq \begin{bmatrix} & & & 1 \\ 1 & & & \\ & \ddots & & \\ & & 1 & \end{bmatrix}. \tag{2.8}$$

Figure 2.4d shows another relevant class of graphs, called tree graphs, or simply trees. A tree is an undirected connected graph with no loops, i.e., there is only one path between two vertices.

2.4 The Shift Operator

The unit-shift operator, also known as the unit-delay operator, is fundamental in conventional DSP [81] and, thus, its translation to the GSP framework is of great relevance. As shown in the next sections, the definition of the graph-shift operator determines the essential differences among the approaches developed for GSP. For conventional DSP, consider a length-N discrete-time signal s_n. Applying the unit-shift operator, denoted here by $\mathcal{T}\{\cdot\}$, to s_n results in

$$\tilde{s}_n = \mathcal{T}\{s_n\} = s_{(n-1) \bmod N}, \tag{2.9}$$

which is the time-shifted version of s_n considering its periodic extension. Note that the operation in (2.9) is a linear transformation of s_n and, thus, can be represented by a matrix. Considering the length-N vector representation for signals s_n and $\tilde{s}_n$, it is possible to rewrite the operation from (2.9) as

$$\begin{bmatrix} \tilde{s}_0 \\ \tilde{s}_1 \\ \vdots \\ \tilde{s}_{N-1} \end{bmatrix} = \underbrace{\begin{bmatrix} & & & 1 \\ 1 & & & \\ & \ddots & & \\ & & 1 & \end{bmatrix}}_{=\mathbf{C}} \cdot \begin{bmatrix} s_0 \\ s_1 \\ \vdots \\ s_{N-1} \end{bmatrix}, \tag{2.10}$$

where the shift-operator is represented by the cyclic-shift matrix $\mathbf{C}$, which coincides with the adjacency matrix of the cyclic graph in Section 2.3.

2.5 GSP from Algebraic Signal Processing

Using concepts of algebraic signal processing [82, 83] and the connection between the representations for the shift operation (2.9) and (2.10), [1] proposes the framework of digital signal processing on graphs. This framework will be referred here as algebraic graph signal processing (AGSP). The foundation of AGSP lies in the definition of the graph shift in a way that it generalizes the shift operator presented in Section 2.4. As shown in Section 2.4, the shift operator for time domain coincides with the adjacency matrix $\mathbf{A} = \mathbf{C}$ of the cyclic graph, presented in Section 2.3. Leveraging this relation, [1] proposes the unit-delay operation in the graph domain for the corresponding graph-signal $\mathbf{s}$ given by

$$\tilde{\mathbf{s}} = \mathbf{C}\mathbf{s}. \tag{2.11}$$

The relation between the adjacency matrix $\mathbf{A}$ and the shift operation, as given in (2.11), is extended to any generic graph $\mathcal{G} = \{\mathcal{V}, \mathbf{A}\}$ as

$$\tilde{\mathbf{s}} = \mathbf{A}\mathbf{s}, \tag{2.12}$$

where each sample $\tilde{s}_i$ of the graph-shifted signal at node v_i is, thus, given by a linear combination of the original-signal samples on $\mathcal{N}_i$ — the neighborhood of v_i. In other words, in AGSP, the graph-shift operator [7, 84] for a graph $\mathcal{G} = \{\mathcal{V}, \mathbf{A}\}$ is equal to the adjacency matrix $\mathbf{A}$. Thus, any graph, directed or undirected, has a corresponding graph-shift operator, with no restrictions imposed on its adjacency matrix.

2.5.1 Graph Filtering

The natural step towards constructing a signal processing framework, given that the shift operator is formally defined, is the development of filters. From conventional DSP, the output of a finite-duration impulse response (FIR) filter with length P, at a given time instant n, is given by a linear combination of recent signal samples at the input of the filter, as follows:

$$\begin{aligned}\bar{s}_n &= h_0 s_n + h_1 s_{n-1} + \cdots + h_{P-1} s_{n-P+1},\\ &= \sum_{p=0}^{P-1} h_p \mathcal{T}^p \left\{ s_n \right\},\end{aligned} \tag{2.13}$$

where the time-invariant coefficients $h_0, \ldots, h_{P-1}$ define the FIR filter impulse response. Note that the terms associated with past-time inputs are given by powers of the unit-shift operator $\mathcal{T}$ applied to the input signal s_n. Consider the vector representation of a length-N discrete-time signal as $[s_0, s_1, \ldots, s_{N-1}]^{\mathrm{T}}$. For a causal filter ($h_p = 0$ for $p < 0$) with length $P \leq N$ ($h_p = 0$ for $p \geq P$) the operation in (2.13) can be written as a circular convolution, which, in contrast to the linear convolution, considers that the filter response is periodic, as

$$\begin{bmatrix} \bar{s}_0 \\ \bar{s}_1 \\ \vdots \\ \bar{s}_{N-1} \end{bmatrix} = \begin{bmatrix} h_0 & h_{N-1} & \cdots & h_1 \\ h_1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & h_{N-1} \\ h_{N-1} & \cdots & h_1 & h_0 \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ \vdots \\ s_{N-1} \end{bmatrix}, \tag{2.14}$$

which means that applying a causal length-P filter to a length-N discrete-time signal is equivalent to pre-multiplying this signal by a matrix $\mathbf{H}(\mathbf{C})$ constructed as a polynomial over the cyclic-shift matrix $\mathbf{C}$ as

$$\mathbf{H}(\mathbf{C}) = \begin{bmatrix} h_0 & h_{N-1} & \cdots & h_1 \\ h_1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & h_{N-1} \\ h_{N-1} & \cdots & h_1 & h_0 \end{bmatrix} = \sum_{p=0}^{P-1} h_p \mathbf{C}^p. \tag{2.15}$$

In AGSP, for a graph $\mathcal{G} = \{\mathcal{V}, \mathbf{A}\}$ with K vertices, a linear graph-filter is represented by a matrix $\mathbf{H} \in \mathbb{C}^{K \times K}$, such that the filtering operation of a graph signal $\mathbf{s}$ is given by the matrix-vector multiplication $\mathbf{Hs}$. Analogously to the conventional DSP case, a length-P linear shift-invariant (LSI) graph filter [11] in the AGSP framework is defined as a polynomial over the graph-shift operator $\mathbf{A}$ as

$$\mathbf{H}(\mathbf{A}) = \sum_{p=0}^{P-1} h_p \mathbf{A}^p. \tag{2.16}$$

The shift-invariance property for a graph filter $\mathbf{H}$ means that $\mathbf{H}(\mathbf{A}\mathbf{s}) = \mathbf{A}(\mathbf{H}\mathbf{s})$. The sufficient and necessary condition of the graph-filter being shift-invariant as a polynomial on $\mathbf{A}$ is proved in [11].

2.5.2 Graph Fourier Transform based on AGSP

Similar to graph shift and graph filtering, the Fourier transform is also translated from conventional DSP to the graph domain [10]. First, we present the definition for the spectral decomposition of a signal space $\mathcal{S}$, which consists of the identification of W filtering-invariant subspaces $\mathcal{S}_0, \ldots, \mathcal{S}_{W-1}$ of $\mathcal{S}$. The filtering-invariance property for a subspace $\mathcal{S}_w$ means that the filtering of a signal $\mathbf{s}_w \in \mathcal{S}_w$ by a filter $\mathbf{H}(\mathbf{A})$ results in a signal $\bar{\mathbf{s}}_w \in \mathcal{S}_w$. The spectral decomposition is uniquely determined for a signal space $\mathcal{S}$ if, and only if [10]:

- $\mathcal{S}_w \cap \mathcal{S}_r = \{\mathbf{0}\},\ w \neq r$;
- $\dim(\mathcal{S}_0) + \cdots + \dim(\mathcal{S}_{W-1}) = \dim(\mathcal{S}) = K$;
- Each $\mathcal{S}_w$ is irreducible to smaller subspaces,

where $\dim(\cdot)$ denotes the dimension of the subspace. If all three conditions are satisfied, one can write $\mathcal{S}$ as the direct sum of the W subspaces, i.e.,

$$\mathcal{S} = \mathcal{S}_0 \oplus \mathcal{S}_1 \oplus \cdots \oplus \mathcal{S}_{W-1}, \tag{2.17}$$

such that any signal $\mathbf{s} \in \mathcal{S}$ is univocally represented by

$$\mathbf{s} = \mathbf{s}_0 + \ldots + \mathbf{s}_{W-1}, \tag{2.18}$$

where each $\mathbf{s}_w$ is a component of the decomposition of $\mathbf{s}$ associated with subspace $\mathcal{S}_w$.

Since linear shift-invariant filters are defined as polynomials $\mathbf{H}(\mathbf{A})$ of the adjacency matrix, the diagonalization of $\mathbf{A}$ leads to a decomposition of the signal space $\mathcal{S}$ into filtering-invariant subspaces. As AGSP is defined for any graph, with no restrictions imposed on the adjacency matrix, $\mathbf{A}$ is not always diagonalizable. Hence, as stated in [11], one can consider the Jordan decomposition $\mathbf{A} = \mathbf{V}\mathbf{J}\mathbf{V}^{-1}$ to conduct the spectral decomposition, where $\mathbf{J}$ is an almost diagonal matrix representing the Jordan normal form of $\mathbf{A}$ and $\mathbf{V}$ is a matrix whose columns are the generalized eigenvectors of $\mathbf{A}$. Hence, $\mathbf{V}$ represents the basis for the spectral decomposition of the signal space $\mathcal{S}$ and (2.18) can be written as

$$\mathbf{s} = \mathbf{V}\hat{\mathbf{s}}, \tag{2.19}$$

where $\hat{\mathbf{s}}$ is the vector composed by the coefficients of the projections of signal $\mathbf{s}$ onto the subspaces of the spectral decomposition of $\mathcal{S}$. The union of the subspaces of $\mathcal{S}$ associated with the generalized eigenvectors of $\mathbf{A}$ composes the graph Fourier basis [11]. Finally, the graph Fourier transform (GFT) matrix is given by $\mathbf{F}_{\mathrm{A}} = \mathbf{V}^{-1}$ such that the coefficients of the GFT are obtained by the matrix-vector multiplication

$$\hat{\mathbf{s}} = \mathbf{F}_{\mathrm{A}}\mathbf{s} = \mathbf{V}^{-1}\mathbf{s}, \tag{2.20}$$

and the inverse graph Fourier transform (IGFT) matrix is given by $\mathbf{F}_{\mathrm{A}}^{-1} = \mathbf{V}$, such that the signal is recovered from its GFT coefficients as

$$\mathbf{s} = \mathbf{F}_{\mathrm{A}}^{-1}\hat{\mathbf{s}} = \mathbf{V}\hat{\mathbf{s}}, \tag{2.21}$$

as indicated by the spectral decomposition in (2.19). Note that, for an undirected graph, the symmetric matrix $\mathbf{A}$ is diagonalizable and, thus, the simpler decomposition into eigenvalues and eigenvectors $\mathbf{A} = \mathbf{V}\boldsymbol{\Gamma}\mathbf{V}^{-1}$ can be used instead of the Jordan decomposition. Moreover, in this case, the eigenvectors are orthonormal, i.e. $\mathbf{V}^{-1} = \mathbf{V}^{\mathrm{T}}$, which reduces the complexity of computing the GFT matrix $\mathbf{F}_{\mathrm{A}}$.

Finally, we note that the GFT, as defined for the AGSP framework, generalizes the conventional discrete-Fourier transform (DFT) when the discrete-time domain is modeled as the cyclic graph represented by the cyclic-shift matrix $\mathbf{C}$. The eigenvectors of $\mathbf{C}$ are equal to the columns of the DFT matrix $\mathbf{F}_{\mathrm{DFT}}$ given by $F_{\mathrm{DFT}\,\mathrm{n,k}} = \exp\left(-\mathrm{j}\frac{2\pi n}{N}k\right)$, where $\mathrm{j} = \sqrt{-1}$.

2.6 GSP from Spectral Graph Theory

The second GSP approach was first introduced by Shuman *et al.* [2] and is based on the graph spectral theory [85], building its framework upon the graph Laplacian (presented in Section 2.1), thus being initially restricted to undirected graphs.[2] As a reference to the graph Laplacian, this graph signal processing approach will be denoted in this text by LGSP.

[2]Recently, research that aims to extend the framework to directed graphs can be found in the literature [86, 87]

2.6.1 Graph Fourier Transform based on LGSP

In [2], the graph Fourier transform is presented as a generalization of the conventional Fourier transform, which is given by the expansion of a function $f(t)$ in terms of complex exponentials, as

$$\hat{f}(\xi) \triangleq \langle f(t), \mathrm{e}^{2\pi \mathrm{j}\xi t} \rangle = \int\limits_{\mathbb{R}} f(t)\mathrm{e}^{-2\pi \mathrm{j}\xi t}\mathrm{d}t, \tag{2.22}$$

where the complex exponentials $\mathrm{e}^{2\pi \mathrm{j}\xi t}$ are the eigenfunctions of the one-dimensional Laplace operator Δ, defined in Section (2.1) as

$$\Delta f(t) \triangleq \frac{\partial^2}{\partial t^2} f(t). \tag{2.23}$$

Indeed, one has

$$\Delta \mathrm{e}^{2\pi \mathrm{j}\xi t} = \frac{\partial^2}{\partial t^2}\mathrm{e}^{2\pi \mathrm{j}\xi t} = -(2\pi\xi)^2 \mathrm{e}^{2\pi \mathrm{j}\xi t}. \tag{2.24}$$

Consider now an undirected representation of the discrete-time domain via an infinite undirected graph, where each time instant is connected to the time instants immediately before and after it. We may represent this graph with an abstraction of the adjacency matrix given by

$$\mathbf{A} = \begin{bmatrix} \ddots & 1 & & \\ 1 & & 1 & \\ & 1 & & 1 \\ & & 1 & \ddots \end{bmatrix}. \tag{2.25}$$

The discrete Laplacian can be interpreted as a finite approximation to the continuous Laplacian operator, as a symmetric second-order difference operator given by [88]

$$\begin{aligned} \mathcal{L}\{x_i\} &= -x_{i-1} + 2x_i - x_{i+1} \\ &= \sum_{j \in \mathcal{N}_i} A_{i,j}(x_i - x_j), \end{aligned} \tag{2.26}$$

which coincides with the operation induced on graph-signals by the graph Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{A}$. Analogously to the classic Fourier transform, the GFT for the LGSP framework is, thus, defined as the expansion of a graph signal $\mathbf{s}$ in terms of the eigenvectors of the graph Laplacian $\mathbf{L} \in \mathbb{R}^{K \times K}$. Let $\mathbf{L}$ be diagonalizable as $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1}$, with eigenvectors given by $\mathbf{u}_l \in \mathbb{R}^K$ and eigenvalues $\lambda_l \in \mathbb{R}$, $l \in \{1, \ldots, K\}$. Since $\mathbf{L}$ is symmetric, the eigenvectors are orthonormal, i.e., $\mathbf{U}^{-1} = \mathbf{U}^{\mathrm{T}}$. The projection of $\mathbf{s}$

into an eigenvector $\mathbf{u}_l$ provides the GFT coefficient associated with graph frequency λ_l:

$$\hat{s}_l \triangleq \langle \mathbf{s}, \mathbf{u}_l \rangle = \sum_{k=1}^{K} s_k u_{l,k}. \tag{2.27}$$

For the entire graph signal $\mathbf{s}$, one can write the GFT for LGSP as

$$\hat{\mathbf{s}} = \mathbf{F}_{\mathrm{L}} \mathbf{s} = \mathbf{U}^{\mathrm{T}} \mathbf{s}, \tag{2.28}$$

where $\mathbf{F}_{\mathrm{L}} = \mathbf{U}^{\mathrm{T}}$ is the GFT matrix. The IGFT for LGSP is given by

$$\mathbf{s} = \mathbf{F}_{\mathrm{L}}^{-1} \hat{\mathbf{s}} = \mathbf{U} \hat{\mathbf{s}}. \tag{2.29}$$

2.6.2 Convolution and Filtering

To define the filtering operation, it is convenient to begin with the definition of convolution as given in [2]. In DSP, the convolution operation applied to two functions $f(t)$ and $g(t)$ is equivalent to the multiplication of their frequency domain representations, i.e.,

$$f * g = \mathcal{F}^{-1} \left\{ \mathcal{F}\{f\} \cdot \mathcal{F}\{g\} \right\}, \tag{2.30}$$

where $\mathcal{F}$ and $\mathcal{F}^{-1}$ denote the Fourier transform and the inverse Fourier transform, respectively. Analogously, the convolution operation for graphs is defined in terms of the GFT and IGFT, such that, for graph signals $\mathbf{s}$ and $\mathbf{h}$,

$$[\mathbf{s} * \mathbf{h}]_i \triangleq \sum_{l=1}^{K} \hat{s}_l \hat{h}_l u_{l,i}. \tag{2.31}$$

The development of the filtering operation considers that conventional filtering is given by the attenuation or amplification of the complex exponentials that represent a signal in the Fourier domain. Consider a graph signal $\mathbf{s}$ with representation on the graph Fourier domain given by $\hat{\mathbf{s}}$, and a graph filter whose frequency response is given by $\hat{h}_l = \hat{h}(\lambda_l)$. Analogous to DSP, the output of graph filtering in the frequency domain is

$$\hat{\bar{s}}_l = \hat{s}_l \hat{h}_l, \tag{2.32}$$

and, in the vertex domain,

$$\bar{s}_i = [\mathbf{s} * \mathbf{h}]_i \tag{2.33}$$

Considering the diagonalization of the graph Laplacian as $\mathbf{L} = \mathbf{U} \boldsymbol{\Lambda} \mathbf{U}^{-1}$, where $\boldsymbol{\Lambda}$ is a diagonal matrix composed by the eigenvalues $\lambda_1, \ldots, \lambda_K$ in its diagonal, we can define a graph filter $\mathbf{H}(\mathbf{L})$ for LGSP as

$$\mathbf{H}(\mathbf{L}) \triangleq \mathbf{U} \begin{bmatrix} \hat{h}(\lambda_1) & & \\ & \ddots & \\ & & \hat{h}(\lambda_K) \end{bmatrix} \mathbf{U}^{-1}. \tag{2.34}$$

The graph filtering operation is given by

$$\bar{\mathbf{s}} = \mathbf{H}(\mathbf{L})\mathbf{s}. \tag{2.35}$$

That is, the graph filtering is equivalent to taking the graph signal into the graph-frequency domain, using the GFT $\hat{\mathbf{s}} = \mathbf{U}^{-1}\mathbf{s}$, operating on the GFT coefficients, and returning the filtered signal to the vertex domain using the IGFT.

From (2.35), one can see that the graph Laplacian constitutes the basic building block of graph filters in LGSP, since the graph filter may be represented as a polynomial on matrix $\mathbf{L}$. Thus, we will refer to $\mathbf{L}$ as the GSO for the LGSP framework.

If the frequency response of the graph filter is given by a length-P polynomial on frequencies λ_l, i.e., $\hat{h}_l = \sum_{p=0}^{P-1} a_p \lambda_l^p$, for constant scalars $a_p \in \mathbb{R}$, $p \in \{0, \ldots, P-1\}$, the graph filtering operation can be writen as

$$\begin{aligned} \bar{s}_i &= \sum_{l=1}^{K} \hat{s}_l \hat{h}_l u_{l,i} \\ &= \sum_{j=1}^{K} s_j \sum_{p=0}^{P-1} a_p \sum_{l=1}^{K} \lambda_l^p u_{l,j} u_{l,i} \\ &= \sum_{j=1}^{K} s_j \sum_{p=0}^{P-1} a_p [\mathbf{L}^p]_{i,j}, \end{aligned} \tag{2.36}$$

which shows that the output of the filtering operation on a vertex v_i is given by a linear combination of signals on vertices inside a $(P-1)$-hop neighborhood of v_n, i.e., vertices that are reachable, from v_i, via paths with $(P-1)$ or fewer edges. This can be deduced by the fact that the graph Laplacian $\mathbf{L}$ is a matrix that indicates the relations between a vertex and its immediate neighborhood. The pth power of the Laplacian matrix, $\mathbf{L}^p$, indicates the relations between a vertex v_i and the set of vertices p edges away from v_i, denoted the p-hop neighborhood of v_i. Any element $[\mathbf{L}^p]_{i,j}$ not in the p-hop neighborhood of v_i is equal to zero and is not added in the summation of (2.36).

2.7 Graph-frequency Interpretation

In both AGSP and LGSP, under analogy with the conventional Fourier transform, the eigenvectors of the graph-shift operator are the frequency components of the Fourier decomposition in the graph setting. Here, we discuss how the choice of the shift operator affects the interpretation of graph frequency and spectrum, given by the set of the eigenvalues associated with the graph-frequency components.

2.7.1 Graph Frequency using the Adjacency Matrix as GSO

We begin by defining the concept of total variation (TV) for graphs, based on the TV($\mathbf{x}$) for discrete signals $\mathbf{x}$ from conventional DSP, given by [10]

$$\mathrm{TV}(\mathbf{x}) = \sum_{n} |x_n - x_{n-1}|, \tag{2.37}$$

which measures the total variation of the signal based on the differences between consecutive samples. Thus, it assumes larger values for high-frequency signals. For a finite-duration signal, the total variation is defined as

$$\mathrm{TV}(\mathbf{s}) = \sum_{n=0}^{N-1} |x_n - x_{(n-1)\bmod N}|, \tag{2.38}$$

which can be written in terms of the cyclic-shift matrix as

$$\mathrm{TV}(\mathbf{s}) = \|\mathbf{x} - \mathbf{C}\mathbf{x}\|_1, \tag{2.39}$$

where $\|\cdot\|_1$ denotes the norm-p for $p = 1$, such that $\|\mathbf{x}\|_1 = \sum_i (|x_i|^p)^{\frac{1}{p}}\Big|_{p=1}$.

From (2.39), the concept of total variation is extended to the AGSP framework by considering the similarity between a graph signal $\mathbf{s}$ and its shifted version, as follows:

$$\mathrm{TV}_{\mathrm{G}}(\mathbf{s}) = \|\mathbf{s} - \mathbf{A}^{\mathrm{norm}}\mathbf{s}\|_1, \tag{2.40}$$

where $\mathbf{A}^{\mathrm{norm}} = \mathbf{A}/|\lambda_{\max}|$ is the normalized version of the adjacency matrix used to guarantee proper comparison between original and shifted signals [10]. Consider an eigenvector $\mathbf{v}_l$ of $\mathbf{A}$, with corresponding eigenvalue given by λ_l. The total variation associated with $\mathbf{v}_l$ is given by

$$\begin{aligned}\mathrm{TV}_{\mathrm{G}}(\mathbf{v}_l) &= \left\|\mathbf{v}_l - \frac{\mathbf{A}\mathbf{v}_l}{|\lambda_{\max}|}\right\|_1 = \left\|\mathbf{v}_l - \frac{\lambda_l}{|\lambda_{\max}|}\mathbf{v}_l\right\|_1 = \left|1 - \frac{\lambda_l}{|\lambda_{\max}|}\right| \|\mathbf{v}_l\|_1 \\ &= (|\lambda_{\max}| - \lambda_l)\frac{\|\mathbf{v}_l\|_1}{|\lambda_{\max}|}.\end{aligned} \tag{2.41}$$

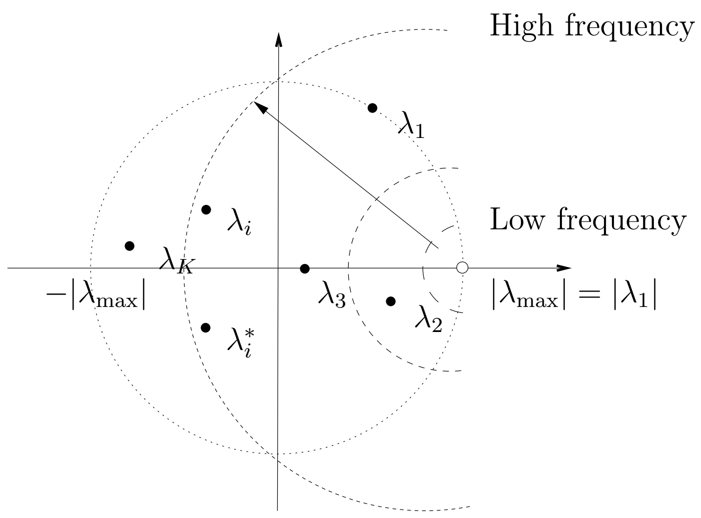


Figure 2.5: Illustration of graph-frequency ordering associated with complex eigenvalues.

Equation (2.41) shows that, since we can scale all eigenvectors to have the same norm-1, the total variation associated with frequency components given by the eigenvectors is determined by the proximity of the corresponding eigenvalue to the point $|\lambda_{\max}|$. In fact, consider an adjacency matrix $\mathbf{A}$ with real eigenvalues $\lambda_1 < \lambda_2 < \cdots < \lambda_K \triangleq \lambda_{\max}$ and corresponding eigenvectors $\mathbf{v}_l$, with $l \in \{1, \ldots, K\}$, such that $\|\mathbf{v}_l\|_1 = 1$, $\forall n$.[3] Given two eigenvectors $\mathbf{v}_i$ and $\mathbf{v}_j$, with $i, j \in \{1, \ldots, K\}$ and $i \neq j$, such that $\lambda_i < \lambda_j$, we have that

$$\mathrm{TV}_{\mathrm{G}}(\mathbf{v}_i) - \mathrm{TV}_{\mathrm{G}}(\mathbf{v}_j) = \frac{\lambda_{\max} - \lambda_i}{\lambda_{\max}} - \frac{\lambda_{\max} - \lambda_j}{\lambda_{\max}} > 0, \tag{2.42}$$

meaning that the total variation associated with smaller real eigenvalues is greater than that associated with larger eigenvalues. Thus, as total variation can be intuitively related to frequency, an ordering for graph-frequencies in the AGSP framework can be defined when eigenvalues are real, such that larger eigenvalues correspond to lower frequencies. If eigenvalues are complex, we have that

$$\Big|\lambda_i - |\lambda_{\max}|\Big| \leq \Big|\lambda_j - |\lambda_{\max}|\Big| \iff \mathrm{TV}_{\mathrm{G}}(\mathbf{v}_i) \leq \mathrm{TV}_{\mathrm{G}}(\mathbf{v}_j), \tag{2.43}$$

i.e., the total variation and, thus, the frequency decrease as the eigenvalue gets closer to the real point $|\lambda_{\max}|$. This is illustrated in Figure 2.5 for the case of $|\lambda_{\max}| = |\lambda_1|$.

[3]For instance, the case of undirected graphs with real edges.

2.7.2 Graph Frequency using the Laplacian as GSO

For a similar analysis for the LGSP framework, consider the Rayleigh quotient of the Laplacian $\mathbf{L}$, given by

$$\mathcal{R}(\mathbf{L},\mathbf{x}) = \frac{\mathbf{x}^{\mathrm{T}}\mathbf{L}\mathbf{x}}{\mathbf{x}^{\mathrm{T}}\mathbf{x}}, \tag{2.44}$$

which is bounded below and above by the extreme eigenvalues of $\mathbf{L}$. In fact, we have

$$\lambda_l = \mathbf{u}_l^{\mathrm{T}}\mathbf{L}\mathbf{u}_l. \tag{2.45}$$

The quadratic form (2.45) induces a variation metric, in the sense that

$$\begin{aligned}\mathbf{u}_l^{\mathrm{T}}\mathbf{L}\mathbf{u}_l &= \mathbf{u}_l^{\mathrm{T}}(\mathbf{D}-\mathbf{A})\mathbf{u}_l \\ &= \frac{1}{2}\sum_{i\neq j} A_{i,j}\Big(u_{l,i}-u_{l,j}\Big)^2,\end{aligned} \tag{2.46}$$

such that the Laplacian eigenvectors with more variations between its elements, weighted by the edge values, are associated with larger eigenvalues. We highlight the function of the graph structure in the graph-frequency interpretation. If two nodes v_i and v_j are not connected, then $A_{i,j} = 0$ and their contribution to the variation metric is zero. On the other hand, if two nodes are highly related, then the difference between their corresponding elements in the Laplacian eigenvectors has great contribution to the variation metric. Considering the expansion of graph signals over the basis composed by the Laplacian eigenvectors, it is intuitive to conclude that smaller eigenvalues are associated with smooth graph signals. That is, Laplacian eigenvalues are a direct representation of graph frequency.

2.8 Summary

This chapter presented the fundamental tools that comprise the GSP framework, employed to analyze and manipulate signals defined over graph nodes, as presented in Section 2.2. Two of the main approaches of GSP were described, namely the AGSP, whose tools are based on the adjacency matrix $\mathbf{A}$ as the graph-shift operator, and the LGSP, based on spectral graph theory, which uses the graph Laplacian matrix $\mathbf{L}$ as the graph-shift operator. Filtering operation and the graph-Fourier domain and transform were presented for both approaches. We highlight the following remarks to be further considered in the subsequent chapters:

- Graph signals are snapshots of the network state at a given time. This implies that the fundamental operations and tools presented in this chapter, such as shift, filtering, and Fourier analysis, operate only in the graph domain. Time dimension is not contemplated in the fundamentals of GSP.

- Although the two GSP approaches were presented separately based on their distinct motivations, most recent literature on GSP does not differentiate between these approaches. Commonly, a generic GSO $\mathbf{S}$ is adopted that satisfies the constraints of the application at hand. The dependence of the GSO on the application serves as a motivation for our research topic **T1** on the translation of real-world problems into graph models suitable for GSP.

Chapters 3 and 4 build upon the second remark and propose methods for modeling the GSO for different applications. In Chapter 3, the AGSP approach is adopted, and we investigate methods to construct adjacency matrices suitable for data compression. In Chapter 4, we propose a methodology for augmenting an initial adjacency matrix considering that no other network information is available.

Chapter 5 expands the concept of graph filters presented in Sections 2.5.1 and 2.6.2, and proposes nonlinear graph filters along with adaptive methods for learning the filter parameters. The time evolution of the graph signals is embedded in the filter model by assuming that the signal's spatio-temporal dynamics depend on the graph structure.

Chapter 6 uses the concepts of variation and smoothness discussed in Section 2.7 to propose methods for learning relations between input and output signals when the output is a graph signal, but the input is not defined over a graph.

Chapter 3

GFT for Light-Field Compression

This chapter describes an application of the GFT and graph models presented in the previous chapter in the context of light-field data compression. This work was published in **P1** [71] and in the extension paper **P2** [72]. In this work, we adopted the AGSP framework, which is based on the use of the adjacency matrix $\mathbf{A}$ as the GSO and defines the Fourier basis as the union of the subspaces spanned by the eigenvectors of $\mathbf{A}$, as described in Section 2.5.2. This research showcases basic GSP concepts and some fundamental challenges of modeling real-world problems as GSP applications.

Section 3.1 introduces light fields and presents the challenges of manipulating light-field data. A brief review of recent works using GSP for light-field applications is presented in Section 3.2. Section 3.3 presents the proposed methodology for compressing light fields using the GFT. In Section 3.4, we present simulations using the proposed compression methodology. Finally, Section 3.5 presents the conclusions and final remarks for this chapter.

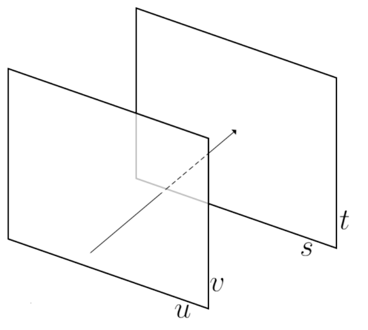
3.1 Introduction to Light Field

A light field is a scalar field, called the plenoptic function, that represents the light information that travels in all directions through all points in space for a given region (referred to as "scene," in practical applications) in space [89, 90].

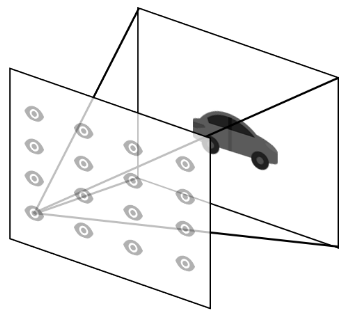
3.1.1 Light Field in Practice

Light-field imaging is a promising technology that opens new horizons for entertainment industries as cinema and photography. Applications in other areas include: microscopy [91]; medical imaging, such as monitoring of neuronal activity [92]; material recognition [69]; rendering [93]; among several others.

In practice, the plenoptic function is not available or obtainable in a feasible way and alternative representations are needed. A common way of reducing the dimensionality of the plenoptic function is via parametrization by two planes st and uv, as shown in Figure 3.1. The uv plane represents the observer viewpoint and the st plane represents the observed scene, such that the set of rays traveling from the st plane and hitting a single point (u_i, v_j) in uv consists of a view of the scene on st from the viewpoint (u_i, v_j).



(a) Planes uv (camera plane) and st (focal plane).



(b) Example of a discrete set of observation points in plane uv and a scene on plane st.

Figure 3.1: 4D parametrization of the plenoptic function. On the camera plane, each observation point is irradiated by the light rays coming from all points on the focal plane that point toward the observation point. This generates different views of the focal plane corresponding to each viewpoint.

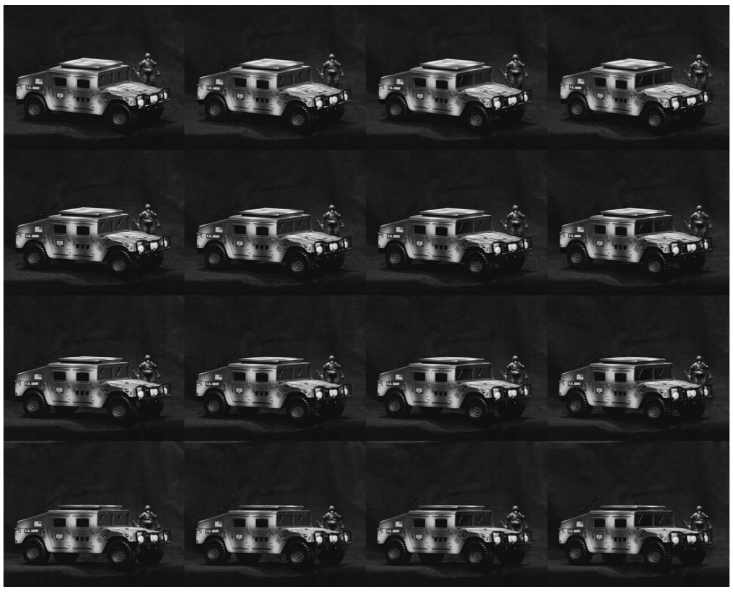
This parametrization leads to practical implementations of light field imaging setups, such as arrays of cameras, a single moving camera, or arrays of microlenses. These parametrizations represent discrete versions of the camera plane.

Synthetic light fields can also be generated by simulating any of the above implementations. In Figure 3.2, we show two examples of light-field-imaging views. Figure 3.2(a) shows 16 views of the real *Humvee* light field, captured by a moving camera, and Figure 3.2(b) shows 25 views of the synthetic *dragon* light field generated by a 3D rendering software.

Light-field applications are relatively data-intensive. For instance, while traditional photography captures a single image of the scene, light-field-based photography may capture hundreds or even thousands of views for a single scene. Hence, the manipulation of light-field data is a challenging task that has received considerable attention from research groups across the world [94–100].

3.2 GSP in Light-field Literature

More recent works available in the literature have explored GSP concepts for light-field applications. In [101, 102], graph-based regularization methods are proposed


(a) Light field *Humvee* – Real.

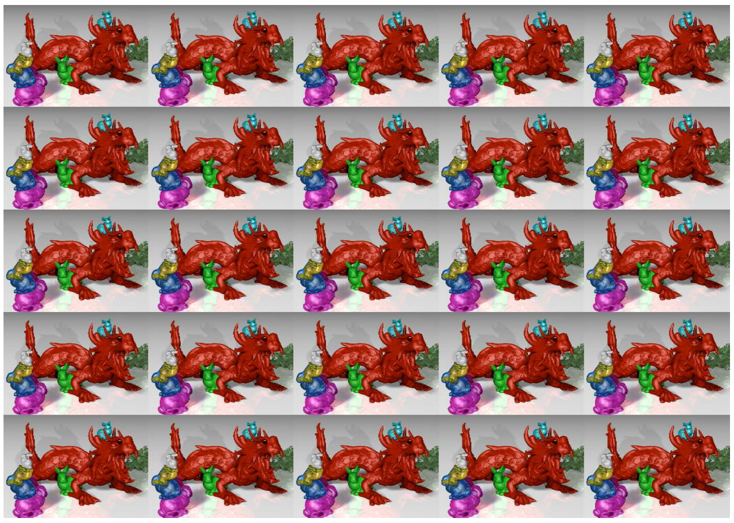

(b) Light field *Dragon* – Synthetic.

Figure 3.2: Examples of light fields.

for light-field super-resolution. Compression methods for different configurations of light-field data are also available in the literature. In [103], a graph-based lifting transform is proposed for compressing light-field data obtained by a single camera before any pre-processing is conducted and before light-field views are available. In [104, 105], graphs are constructed using super-pixels, i.e., groups of pixels with similar color values, and different Laplacian-based transforms are proposed for light-field compression. In [106], compression is achieved by using a graph learning approach to reconstruct the entire light field from an arbitrary subset of views. In [107], graph reduction and spectral clustering are used to overcome the high complexity inherent to graph-based approaches and optimize light-field compression in a rate-distortion sense.

3.3 Compression Methodology

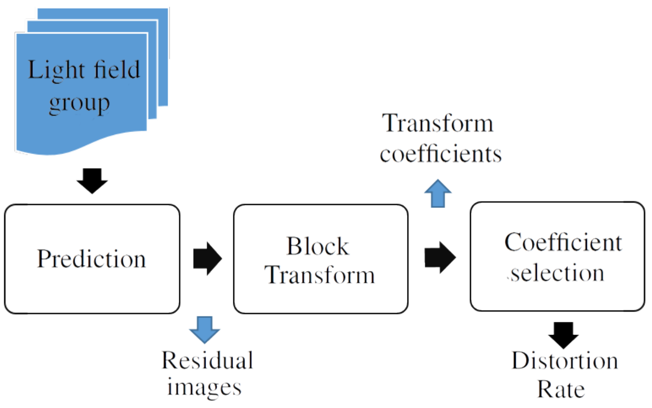


Figure 3.3: Block diagram for the simplified compression scheme adopted in **P1** and **P2**.

We propose and analyze the viability of using the GFT as an alternative to the discrete-cosine transform (DCT) employed in the transformation process of light-field encoders based on high-efficiency video coding (HEVC) while also exploiting the fact that light-field views are similar to each other. The GFT can concentrate information in few transform coefficients in a competitive manner compared to other transforms [108–110]. However, the transform and its inverse depend on the graph structure, which, in turn, depends on application and data. Thus, the impact of storing or transmitting $\mathbf{A}$ or the transform matrix $\mathbf{F}$ must be considered during compression. The method proposed in this work aims at deriving a graph model to reduce the impact of the extra data related to the graph structure by exploring the redundancy that exists among views that are situated close to each other in a set of

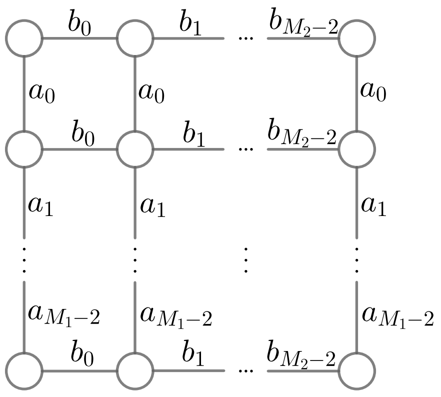


Figure 3.4: Relation edges according to the NN image model. Edges connect only pixels at minimum distance among all pixels.

light field views.

In order to assess the performance of using GFT for light-field compression, a simplified HEVC-based compression process was adopted in **P1** and **P2**, as presented in Figure 3.3. A database composed of seven light fields is used. Three of them, namely *Humvee*, *Knights*, and *Tarot*, are obtained from the *Stanford Light Field Archive* [111]. The other four light fields (namely *Boxes*, *Cotton*, *Dino*, and *Sideboard*) are generated synthetically, obtained from the *HCI 4D Light Field Dataset* [112, 113].

First, each image/view is divided into T blocks. We obtain a residual block in the prediction stage by computing the difference between blocks in a reference image and blocks in other light field views. Residual blocks should have less entropy than raw blocks, which makes compression stages more efficient. The transformation is then applied to the residual block, which is modeled as a graph following the NN image model presented in Section 2.3.

Let $\mathbf{B}_{n,t}$, $n \in \{1, \ldots, N\}$, $t \in \{1, \ldots, T\}$ denote the $M_1 \times M_2$ residual block from the nth residual image $\mathbf{R}_n$ (in a prediction-group with $N-1$ residual images). The graph signal associated with this block is $\mathbf{s}_{n,t}$. The corresponding adjacency matrix is denoted by $\mathbf{A}_{n,t}$ and the GFT matrix by $\mathbf{F}_{n,t}$. The first consideration adopted in this work to reduce the impact of transmitting the transform matrix is to build a sparse adjacency matrix and transmit $\mathbf{A}_{n,t}$ instead of $\mathbf{F}_{n,t}$. The adjacency matrix $\mathbf{A}_{n,t}$ is built according to the NN image model, presented in Section 2.3, which grants sparsity. The model also assumes that an image is a 2D NN graph constructed as a Cartesian product of two 1D NN graphs. A 1D NN graph is a possibly-directed graph where vertices are connected in a line. This model generates a structure where multiple edges assume the same value, indicated by coefficients $a_0, \ldots, a_{M_1-2}$ and $b_0, \ldots, b_{M_2-2}$ in Figure 3.4. As a result, considering an $M_1 \times M_2$

residual block $\mathbf{B}_{n,t}$, the corresponding adjacency matrix $\mathbf{A}_{n,t} \in \mathbb{R}^{K \times K}$, $K = M_1 M_2$, has at most $(M_1 - 1) + (M_2 - 1)$ unique non-zero coefficients. For blocks of size 32×32, this means 62 unique non-zero coefficients out of 1024 entries of $\mathbf{A}_{n,t}$. The coefficients $a_0, \ldots, a_{M_1-2}$ and $b_0, \ldots, b_{M_2-2}$ are defined so as to minimize the ℓ_2 distortion introduced by the shift operation, i.e., $\|\mathbf{A}_{n,t}\mathbf{s}_{n,t} - \mathbf{s}_{n,t}\|_2$. As described in [80], this minimization is solved as an overdetermined least-squares problem.

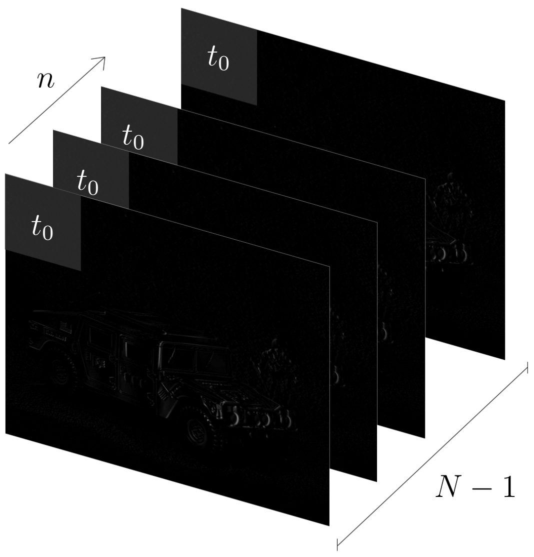


Figure 3.5: Representation of a block position t_0 for residual images from a prediction group.

The second procedure employed to reduce the impact of $\mathbf{A}_{n,t}$, besides forcing sparsity and the fixed structure via the NN image model, is to exploit the redundancy among the many views of the light field to avoid transmitting $\mathbf{A}_{n,t}$ with every single block. Considering that every view is equally divided into T blocks, only one $\mathbf{A}_{t_0}$ is transmitted for a given block position t_0 across the entire prediction group. Figure 3.5 shows an example of block position t_0 across views from a prediction group. The computation of the single adjacency matrix can be optimized using a single (central) residual image or optimized jointly over a group of images. Using a single adjacency matrix that is not explicitly computed for a given block may degrade the efficiency of the GFT. However, the impact of transmitting the matrix is significantly reduced.

Once the adjacency matrix is computed, the GFT matrix for each block position is given by $\mathbf{F}_t$, whose columns are the eigenvectors of $\mathbf{A}_t$ — the reader should keep in mind that the index n can now be dropped from $\mathbf{A}_{n,t}$ and $\mathbf{F}_{n,t}$ since it is assumed that adjacency and transform matrices do not depend on the residual image, given that only one matrix is considered for a given block position across the entire prediction group. The transform coefficients for each block from residual images in the prediction group are computed as $\hat{\mathbf{s}}_{n,t} = \mathbf{F}_t \mathbf{s}_{n,t}$, where $\mathbf{s}_{n,t}$ is the graph

Algorithm 1: LF-compression simulation

```
Input: LF images, prediction method, adjacency construction strategy,
       distortion metric, $Q^{\mathrm{DCT}}$
for each prediction group $n$ do
    Prediction
    Compute residual images according to Section IV.A of P2;
    Divide each residual image into $T$ blocks of $32 \times 32$ pixels;
    for each block in each residual image do
        Transform
        Compute the DCT;
        Construct adjacency matrix $\mathbf{A}_t$ according to Section IV.B of P2;
        Compute the GFT;
        Coefficient selection
        Set $Q^{\mathrm{DCT}}$ coefficients to zero and measure DCT distortion;
        Set $Q = Q^{\mathrm{DCT}}$ coefficients to zero and measure GFT distortion;
        if GFT distortion < DCT distortion then
            increase $Q$ while GFT distortion < DCT distortion;
        else if GFT distortion > DCT distortion then
            decrease $Q$ while GFT distortion > DCT distortion;
    end
end
```

signal corresponding to each block.

3.3.1 Coefficient Selection

A heuristic technique is adopted to assess the performance of GFT against DCT for light-field compression when employed in an HEVC-based compression system. The IGFT is given by the transpose of $\mathbf{F}_t$, since eigenvectors from $\mathbf{A}_t$ are orthonormal. If IGFT is applied to transform coefficients $\hat{\mathbf{s}}_{n,t}$, the signal $\mathbf{s}_{n,t}$ is perfectly recovered. In practical applications, compression occurs when transform coefficients are quantized, resulting also in loss of information. In this work, the Q smallest GFT coefficients are set to zero. The value of Q is adjusted so that using the GFT yields less distortion than a specific DCT compression. We consider both the mean squared error and the structural similarity index as distortion metrics. The number of compressed DCT coefficients is fixed at $Q^{\mathrm{DCT}} = 924$, i.e., only the 100 largest out of the 1024 coefficients are kept with approximately 10:1 compression ratio. The vectors with compressed GFT coefficients are denoted by $\hat{\mathbf{s}}^{Q}_{n,t}$. No coding is employed. When the IGFT is applied to these coefficients, the signal $\mathbf{s}^{Q}_{n,t}$, which is recovered by inverse transform, is an approximation of the original signal $\mathbf{s}_{n,t}$. A compressed version $\mathbf{B}^{\mathrm{GFT}}_{n,t}$ of the original block $\mathbf{B}_{n,t}$ can be constructed from the signal recovered. For the case of DCT, the 2D DCT is applied directly to block $\mathbf{B}_{n,t}$ and by setting the smallest coefficients from the transform block to zero, a compressed block $\mathbf{B}^{\mathrm{DCT}}_{n,t}$

Table 3.1: Simulation results for transform-setup analysis

	Central residual		**Part of group**		**Entire group**	
Light field	**Reduction [%]**	**Std(Q)**[†]	**Reduction [%]**	**Std(Q)**[†]	**Reduction [%]**	**Std(Q)**[†]
Humvee	8.97	6.97	9.65	4.63	8.63	1.82
Knights	13.40	11.04	16.67	8.57	17.53	1.93
Tarot	-3.91	3.50	-0.65	1.96	-0.29	0.83
Boxes	0.22	4.56	6.57	2.45	7.76	1.42
Cotton	5.90	3.05	6.28	1.94	6.07	1.00
Dino	21.22	5.14	21.92	3.61	21.18	1.92
Sideboard	-3.89	2.67	-2.29	1.23	-2.04	0.86

[†]Std(Q) denotes the standard deviation of Q

is recovered via inverse discrete-cosine transform (IDCT). The compression process adopted in the simulations is summarized in Algorithm 1.

3.4 Simulation and Results

Simulations were conducted in order to compare GFT against DCT when employed in the proposed compression system. Different simulation setups were considered, using different configurations in the prediction and transformation stages, besides different performance-assessment metrics. Some results considering different methodologies for constructing the adjacency matrix are presented in Table 3.1. Reduction values for the total number of coefficients (#) for each light field are computed as

$$\text{Reduction} = \frac{\#\ \text{DCT coefficients} - \#\ \text{GFT coefficients}}{\#\ \text{DCT coefficients}}. \tag{3.1}$$

It is worth highlighting that the number of coefficients associated with the adjacency matrices is included in # GFT coefficients, and, thus, the impact of transmitting $\mathbf{A}_t$ is considered. GFT shows improvement over DCT for most cases, yielding up to 21.92% of reduction in the number of coefficients. The analysis shows that using multiple residual images when building $\mathbf{A}_t$ improved the results for all cases when compared to results obtained using only one residual image as a reference. A relevant analysis given different transform setups is to observe the standard deviation of the number of coefficients used by the GFT across the residual images. The standard deviation is estimated for each light field, using the number of compressed GFT coefficients Q^{GFT} from each residual image as a sample for the standard deviation estimator. Table 3.1 shows that using the entire prediction group reduces the standard deviation of Q^{GFT}. This reflects that when the GFT is constructed using only the central residual image, its efficiency is high for the central residual image but decays as residual images get further apart from the central reference. This result is expected since correlation is reduced and the impact of using a single transform matrix is increased, thus requiring more coefficients.

3.5 Summary

This chapter investigated the applicability of GSP for light-field compression, highlighted the challenges related to the modeling of real-world problems into the GSP framework, and proposed ideas to overcome these challenges. The GFT is used to map blocks of pixels into the graph-frequency domain, which provides energy concentration and allows for compression. The first challenge is the complexity associated with the dimensions of the graph. The nearest-neighbor image model is used to guarantee sparsity to the adjacency matrix, and we leverage the similarities between light-field views to generate a single matrix that can be used for a group of images. Once the graph structure is defined, the second challenge is defining the edge weights suitable for the application at hand. Several methods for computing the weights were investigated and tested with numerical simulations, yielding relevant performance changes. This reflects the model dependency of the GSP framework.

Chapter 4

Extended Adjacency and Diffusion-Dependent GFT using Diffusion Distances

In Chapter 3, we observed the application dependence of the GSO when crafting an adjacency matrix suitable for LF compression. This dependency is related to the more fundamental problem of modeling the original network by a graph; different models have different properties that can be explored by GSP tools [7, 15–23]. For a given network and application, it is desirable to define a GSO that best describes node relations so that the corresponding network signals can be better analyzed/processed. A particular case that is relevant to this topic is the frequency analysis yielded by the GFT. The spectrum of a graph is directly related to the eigenvalues of the GSO. Consequently, changes in the GSO entries are reflected in the graph spectrum, possibly allowing the discrimination of different frequency contents of the same network signal.

In this chapter, we propose the augmentation of the adjacency model of networks for graph signal processing. This research is aligned with research objective **T1** and is documented in papers **P3** [73] and **P4** [74]. We assume here that adjacency matrices are initially sparse, rendering GSOs with a reduced number of edges. This is a common assumption due to sparsity constraints commonly imposed upon adjacency matrices or due to application limitations, e.g., sensors identifying their neighbors. Moreover, we consider that only the adjacency information is available and accurate network information is unknown, such that updating or deriving a new GSO becomes challenging.

Section 4.1 presents the proposed methodology. The methodology is applied to detect anomalies in numerical experiments presented in Section 4.2. Finally, a summary of the chapter is presented in Section 4.3.

4.1 Adjacency-Augmentation Methodology

In the proposed model, additional edges are created according to a virtual Markov relation imposed between nodes. The Markov property is incorporated into the GSO as a function of diffusion distances (DDs) between network elements. Markov relations occur naturally in some applications, such as in consensus [114–116] and random-walk-driven networks [117, 118]. For generic networks, we propose a derivation of the Markov matrix based on the consensus algorithm [114]. DDs are part of the diffusion-maps (DMs) framework [119–121]. DMs are applicable to datasets composed of states of high-dimensional data points, which graph nodes can represent. Note that each graph node is associated with an entire data state of the network, such that edges define transition relations between these high-dimensional data states. This view is in contrast with that of GSP, where edges associate individual network elements. Using eigenvectors of the corresponding Markov matrix, DMs uncover descriptions of the underlying geometry of the dataset [122–124]. In this framework, DDs provide a metric for relating two data states according to the random walk.

4.1.1 Background on Diffusion Maps

Let $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N] \in \mathbb{R}^{L\times N}$ be a data matrix with N data points, also called states, each of dimension L. For example, matrix $\mathbf{X}$ can describe the evolution of the state $\mathbf{x}_n$ of a network with L elements for time instants $n \in \{1, \ldots, N\}$. It is assumed that there is an underlying (hidden) process that relates the different data points, possibly driving the way data is generated. However, note that there is no underlying graph initially associated with $\mathbf{X}$. The objective of the DM framework is to make this underlying process explicit [119, 120, 124].

In DMs, a graph is generated using the data points as nodes according to a symmetric kernel κ. For instance, using the RBF kernel, we have

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma_{\text{RBF}}^2}\right), \tag{4.1}$$

where $\sigma_{\text{RBF}} > 0$ is a free parameter that controls the bandwidth of the kernel. Now, an adjacency matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$ can be defined with entries $A_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$. The corresponding degree matrix is $\mathbf{D} \in \mathbb{R}^{N\times N}$ such that $D_{ii} = \sum_{j\in\mathcal{N}_i} \kappa(\mathbf{x}_i, \mathbf{x}_j)$.

Normalizing the relation between two data points as [119]

$$p(\mathbf{x}_j|\mathbf{x}_i) = \frac{k(\mathbf{x}_i, \mathbf{x}_j)}{\sum\limits_j k(\mathbf{x}_i, \mathbf{x}_j)} = \frac{A_{ij}}{D_{ii}}, \tag{4.2}$$

$p(\mathbf{x}_j|\mathbf{x}_i)$ can be interpreted as the transition probability from $\mathbf{x}_i$ to $\mathbf{x}_j$, which establishes a Markov chain. Using matrix notation, the Markov chain can be described in terms of a right-stochastic matrix $\mathbf{M} = \mathbf{D}^{-1}\mathbf{A}$, commonly referred to as a Markov matrix, with entries $M_{ij} = p(\mathbf{x}_j|\mathbf{x}_i)$. Taking t steps of the random walk is captured by $\mathbf{M}^t$, i.e., the (i,j)th entry of $\mathbf{M}^t$ gives the transition probability, denoted by $p_t(\mathbf{x}_j|\mathbf{x}_i)$, from $\mathbf{x}_i$ to $\mathbf{x}_j$ in t steps. The probability $p_t(\mathbf{x}_j|\mathbf{x}_i)$ considers all possible paths composed of t edges that connect $\mathbf{x}_i$ to $\mathbf{x}_j$, including self-loops. The probability in (4.2) is the same as $p_t(\mathbf{x}_j|\mathbf{x}_i)$ for $t = 1$.

The diffusion distance (DD) at a certain diffusion, or time, scale t [119] is a metric for the (inverse) affinity between two data points as a function of transition probabilities, and is defined as

$$D_t^2(\mathbf{x}_i, \mathbf{x}_j) = \sum_{n=1}^{N} \frac{\left(p_t(\mathbf{x}_n|\mathbf{x}_i) - p_t(\mathbf{x}_n|\mathbf{x}_j)\right)^2}{\phi_{1,n}}, \tag{4.3}$$

where $\phi_{1,j}$ is the jth entry of the first left eigenvector of $\mathbf{M}$, $\boldsymbol{\phi}_1$, normalized as $\|\boldsymbol{\phi}_1\|_1 = 1$.

In terms of adjacency and direct similarity between nodes, the DD extends local relations into a global metric by assimilating probabilities of diffusion paths [122]. That is, if there are high-probability paths between two data points, they are considered to be close in terms of diffusion distance, even if not adjacent. Conversely, the diffusion distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ is large when the probability of reaching $\mathbf{x}_j$ from $\mathbf{x}_i$ is small even by considering non-direct paths through $\mathbf{x}_n$. A fundamental feature for the methodology proposed here is that the DD depends on t, which serves as a diffusion-scale parameter. An increased t equals more steps of the random walk, which corresponds to a larger-scale diffusion over the network.

4.1.2 GSP for Markov Networks

The networks considered here are initially constrained by adjacency rules of the network topology. For example, the connections between nodes in a wireless sensor network depend on their communication capabilities, usually dictated by physical distance. However, nodes that are not initially connected can be related to each other through collaboration. In other words, network adjacency can be associated with one step of a collaboration process, such that the network operates on the data through iterative multiplications of a stochastic matrix. In this work, similarly to the conventional adjacency, which depends on the physical distance between nodes, we develop the extended adjacency based on diffusion distances.

The DM framework in [119] treats the entire network signal as a state of a Markov chain and generates a Markov-based graph over these states. In contrast, we adapt the diffusion distances to develop a comprehensive network model by treating the original network itself as the Markov-based graph.

Let the network be represented by a connected graph $\mathcal{G} = \{\mathcal{V}, \mathbf{B}\}$ with a symmetric, irreducible, and stochastic adjacency matrix $\mathbf{B}$ with positive real edges. In analogy with DM theory, this matrix is equivalent to $\mathbf{M}$ and the states of the Markov chain are the network nodes. The DD between nodes v_i and v_j of the graph is given by

$$D_t^2(v_i, v_j) = \sum_{k=1}^{K} \frac{\left(B_{ik}^{(t)} - B_{jk}^{(t)}\right)^2}{(1/K)}, \tag{4.4}$$

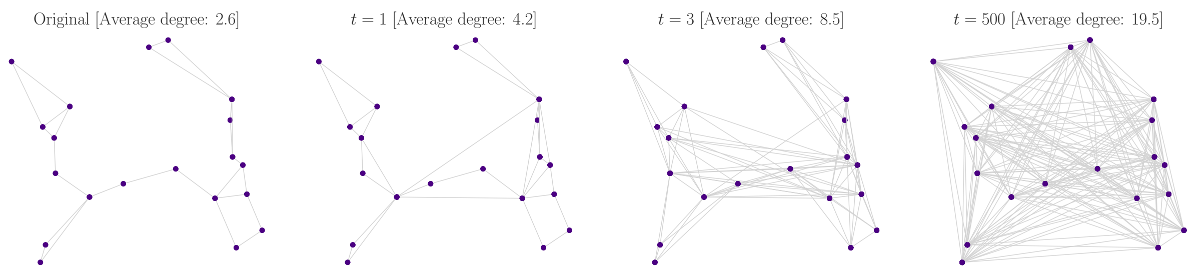
where $(1/K)$ corresponds to the elements of the first left eigenvector $\mathbf{q}_1$ of $\mathbf{B}$, as in (4.3), and $B_{ij}^{(t)}$ denotes the (i, j)th entry of $\mathbf{B}^t$. The metric in (4.4) expresses distances between nodes, including nodes that are not neighbors as defined by $\mathbf{B}$. We derive similarity from diffusion distances in a similar manner as conventional adjacency matrices are derived from geographic distances. The extended-adjacency matrix $\bar{\mathbf{A}}(t)$ is such that

$$\bar{A}_{ij}(t) = \begin{cases} B_{ij} + \exp\left(-\frac{D_t^2(v_i,v_j)}{\rho K}\right) & i \neq j \\ 0 & i = j, \end{cases} \tag{4.5}$$

where $\rho > 0$ is a free parameter and K is the size of the network. The term B_{ij} in (4.5) guarantees that original edges are maintained, whereas the RBF term is responsible for extending the adjacency. The term ρK makes the argument of the RBF kernel independent of the network size (cf. (4.4)). The range of the kernel output can be adjusted for different applications according to the free parameter ρ. Moreover, although the extended adjacency is defined for $\rho > 0$, it is possible to obtain B_{ij}, with $i \neq j$, through (4.5) by making $\rho \to 0^+$. That is, the original adjacency is a particular case of the extended adjacency.

Although traditional GSOs are local with respect to network connections, we note that this property is not present in the proposed adjacency model. A local GSO offers a straightforward visualization of the physical structure of the graph and facilitates the implementation of distributed GSP algorithms. However, in many applications, the definition of locality is unknown, or a local GSO fails to model implicit node relations. In this work, we aim to derive a model that is not restricted by locality assumptions and is useful for networks where non-adjacent nodes interact. Hence, we propose a non-local model that offers a trade-off between locality and representation of node interactions.

For the proposed extended adjacency, the following results hold, with proofs provided in **P4**:



Original [Average degree: 2.6]
$t = 1$ [Average degree: 4.2]
$t = 3$ [Average degree: 8.5]
$t = 500$ [Average degree: 19.5]

Figure 4.1: Connectivity versus diffusion scale. Only edges of $\bar{\mathbf{A}}(t)$ that exceed 30% of the highest edge value are shown.

Corollary 1. *Edge weights are non-decreasing with t according to* (4.5)*. Assuming that an edge exists only if its weight exceeds a given threshold, the number of edges is also non-decreasing with increasing t. In other words, increasing t for a fixed ρ possibly creates new edges, given the reduction in the DD.*

Lemma 1. *The asymptotic extended-adjacency matrix $\bar{\mathbf{A}} = \lim_{t\to\infty} \bar{\mathbf{A}}(t)$ is such that*

$$\bar{A}_{ij} = \begin{cases} B_{ij} + 1 & i \neq j \\ 0 & i = j, \end{cases} \tag{4.6}$$

which composes a complete graph without self-loops. That is, each node v_i is connected to every node v_j in the graph, with $i \neq j$, by an edge of value $1 + B_{ij}$.

The behavior described in Corollary 1 and Lemma 1 is illustrated in Figure 4.1 for a network with $K = 20$ nodes.

4.1.3 Scale-dependent Graph Fourier Transform

For the extended-adjacency matrix $\bar{\mathbf{A}}(t)$, the corresponding graph Fourier transform also depends on the scale t. For each diffusion scale, there is a corresponding Laplacian matrix defined as

$$\bar{\mathbf{L}}(t) = \bar{\mathbf{D}}(t) - \bar{\mathbf{A}}(t), \tag{4.7}$$

where $\bar{\mathbf{D}}(t)$ is the diagonal degree matrix associated with $\bar{\mathbf{A}}(t)$. Let $\bar{\mathbf{L}}(t) = \bar{\mathbf{U}}(t)\bar{\mathbf{\Lambda}}(t)\bar{\mathbf{U}}^{\mathrm{T}}(t)$. The scale-dependent graph Fourier analysis (sGFT) of signal $\mathbf{x}$ as

$$\hat{\mathbf{x}}(t) = \bar{\mathbf{U}}^{\mathrm{T}}(t)\mathbf{x}, \tag{4.8}$$

where, in contrast to the conventional GFT, the coefficients $\hat{\mathbf{x}}(t)$ depend on the diffusion-scale t.

Let θ_2 denote the smallest non-zero eigenvalue of $\mathbf{L}_{\mathbf{B}}$, the Laplacian of $\mathbf{B}$, and

θ_K denote the maximum eigenvalue of $\mathbf{L_B}$. The following proposition holds for the sGFT:

Proposition 1. *For a graph with K nodes, if t is increased, the range of graph-frequencies discriminated by the sGFT shifts into higher frequencies. Asymptotically, the sGFT discriminates graph-frequency ranges up to the interval $[K + \theta_2, K + \theta_K]$.*

The sGFT is a frequency-analysis tool tailored for each stage of the Markov chain. From a node-collaboration perspective, the effect of node collaboration on the graph spectrum can be interpreted intuitively: if more steps of collaboration are taken, more edges are introduced. Consequently, variations in graph signals are observed by additional node pairs and are perceived as larger frequencies. Hence, by incorporating node collaboration into the graph model, we provide a frequency-analysis tool that reveals more information on the network signal than that offered by the conventional GFT.

Numerical experiments to verify the behavior described in Proposition 1 were conducted for a network with $K = 100$ nodes. Figure 4.2 shows histograms of the eigenvalues of the graph Laplacian $\bar{\mathbf{L}}(t)$ versus diffusion scale. An increase in the diffusion scale yields an increase in the spectral gap[1], from around 14 for $t = 1$ up to around 55 for $t = 6$, and in the spectral radius[2], from around 34 up to over 74. At each scale, the sGFT yields a different spectrum according to the number of steps of node collaboration. In contrast, the conventional GFT yields a fixed spectrum. The additional information provided by the sGFT can benefit applications that make decisions based on spectrum-related features, such as classifiers and detectors.

4.2 Application for Anomaly Detection

We study how the sGFT can be used for anomaly detection in synthetic and real networks. The application is motivated by the increasing connectedness of real-world elements [125–128], which demands security and reliability in networks [129–135]. We construct classifiers based on the graph-spectral information generated by the sGFT and the conventional GFT along with other GSO-construction methods. The approach for constructing the anomaly detector is similar to those in [10, 136, 137]. More specifically, assuming that smoothness is expected in the healthy signal, we apply a high-pass filter and conduct the classification based on the filtered coefficients. If one of the coefficients exceeds a given threshold, the signal is classified as anomalous. The metric adopted to assess the performance of the classifiers is the f1 score, which measures the classifiers' accuracy considering both precision and

[1]Smallest nonzero Laplacian eigenvalue

[2]Largest Laplacian eigenvalue

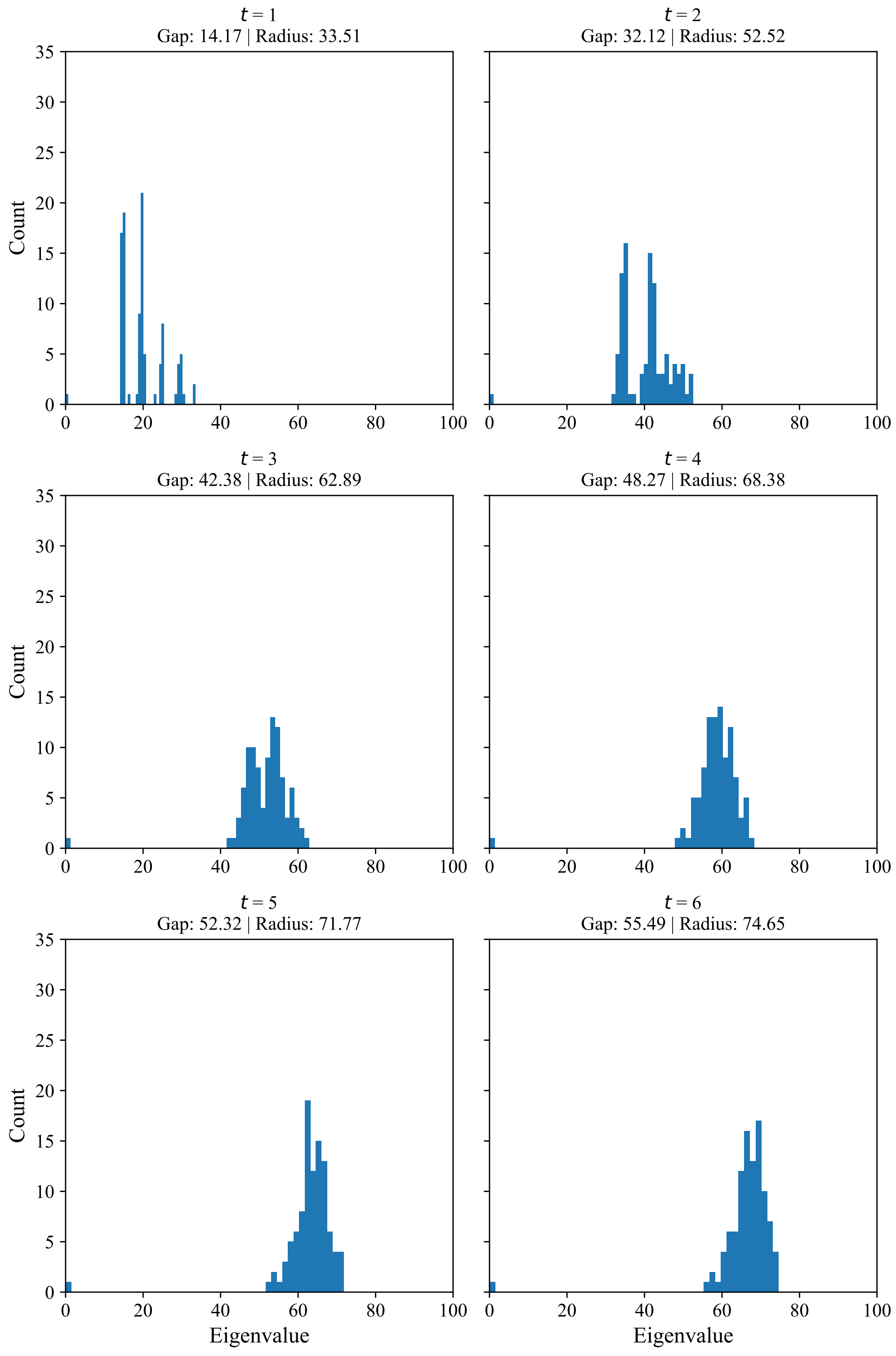
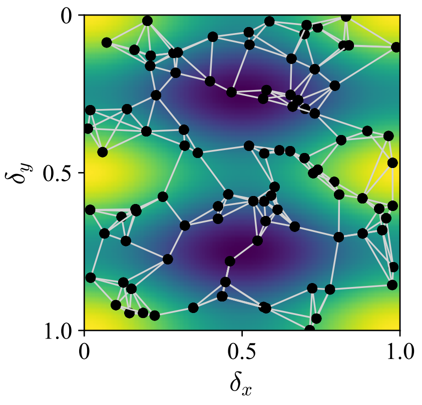
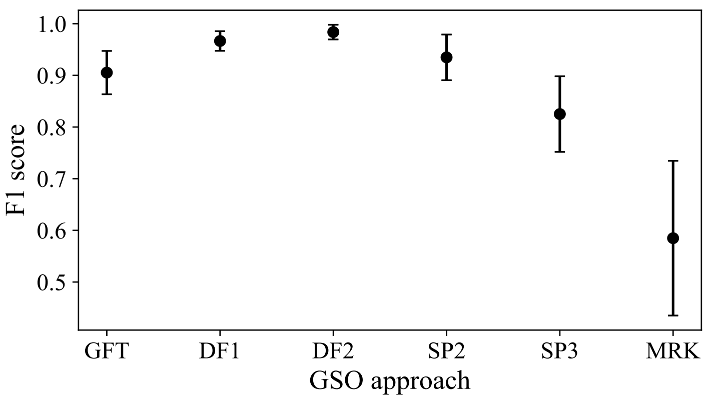


Figure 4.2: Histogram of eigenvalues of diffusion Laplacian matrices $\bar{\mathbf{L}}(t)$, for $t \in \{1, \ldots, 6\}$.





(a) (b)

Figure 4.3: Experiment 1 – setup and results: (a) spatial distribution of the sensors and their interconnections plotted over a snapshot of the observed signal; and (b) f1-scores achieved by each of the GSO-construction approaches.

recall [138]. The best possible f1 score is 1 and small increases in the score can be relevant to certain applications that require better classification accuracy.

4.2.1 Experiment 1: Spatially-spread Anomaly

We consider first $K = 100$ sensors randomly distributed in the square space $[0,1] \times [0,1]$. A network is constructed by connecting each sensor to its 4 nearest-neighbors, and the corresponding adjacency matrix $\mathbf{A}$ is known. The sensors measure a spatially-smooth wave signal given by $s(\delta_x, \delta_y) = \cos(2\pi\delta_x + \theta_x) + \cos(2\pi 2\delta_y + \theta_y)$, where $\delta_x, \delta_y \in [0,1]$ are, respectively, the horizontal and vertical spatial coordinates and θ_x and θ_y are varying phase values uniformly and independently sampled from $[0, 2\pi]$. The graph structure and a snapshot of the signal s are depicted in Figure 4.3a.

We consider the problem of detecting an additive (space-wise) high-frequency interference signal given by $s_i(\delta_x, \delta_y) = 0.1\,(\cos(2\pi 5\delta_x + \theta_x) + \cos(2\pi 6\delta_y + \theta_y))$. Training and test datasets have 150 healthy samples and 150 anomalous samples each. Results for the f1 score achieved over the 50 independent runs are presented in Figure 4.3b. The classifiers based on sGFT are denoted by "DF1" and "DF2", using diffusion scale $t = 1$ and $t = 2$, respectively. For comparison, classifiers denoted by "SP2" and "SP3" are based on shortest-paths, and "MRK" uses a Markov matrix as adjacency matrix following the method proposed in [120]. These classifiers are described in **P4**. Results show that the detector based on the spectral information provided by the extended-adjacency matrices outperform detectors based on other GSO approaches, and greatly outperforms the classifier based on the GFT using the original adjacency matrix. This showcases the benefits of considering the

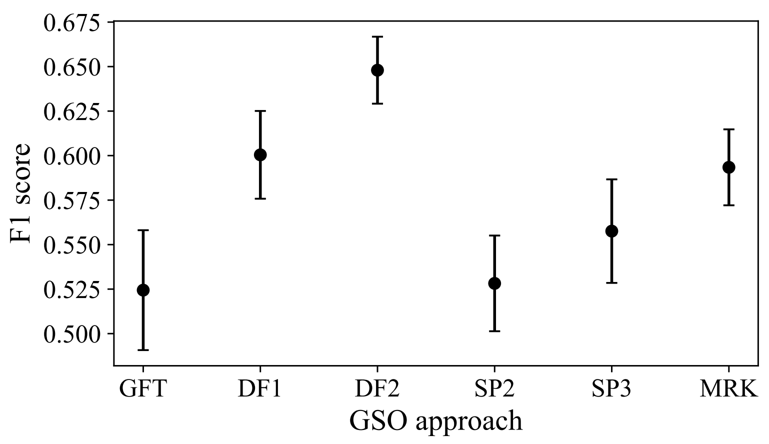


Figure 4.4: F1 scores achieved for Experiment 2.

extended-adjacency model and the additional graph-frequency information in GSP applications.

4.2.2 Experiment 2: Global Surface Summary of the Day (GSOD) - Sensor Malfunction

The database [139] consists of measurements from weather stations distributed across the United States of America. We use temperature measurements, converted from degrees Fahrenheit to degrees Celsius, obtained during 2010 by 150 randomly selected stations from the conterminous United States (excluding Alaska, Hawaii, and other off-shore insular areas) in order to keep the graph connected. The network structure is derived from available stations' latitudes and longitudes.

Figure 4.4 shows that the proposed approach outperforms the other approaches for both scales $t = 1$ and $t = 2$, while the latter offers the best result. Here, the Markov-matrix-based approach outperforms the ones using the conventional Laplacian and the shortest-path-based GSOs.

4.3 Summary

The proposed method for augmenting the adjacency matrix incorporates relations between non-adjacent nodes in a certain diffusion scale or time. Building upon the extended adjacency, we also presented the scale-dependent graph Fourier transform for data defined over these networks. The proposed transform results from the conventional GFT used together with the proposed scale-dependent model. A theoretical analysis shows that increasing the diffusion scale results in increased connectivity in the network; each scale yields a different perspective of graph frequency. Hence, tools that operate on the graph spectrum can leverage the additional information.

For instance, we employed the sGFT for anomaly detection in synthetic and real networks. We used the free parameters of the sGFT to conduct a frequency analysis tailored for the available networks, and results showed that anomaly detectors based on the sGFT outperform other approaches.

This chapter addressed modeling over graphs, in line with research topic **T1** (Section 1.1). In the following chapters, we turn to **T2** and the development of efficient tools to learn nonlinear input-output relations over graphs.

Chapter 5

LMS-based Strategies for Learning over Graphs

In Chapter 1, we presented some of the learning methods in the GSP literature. This chapter addresses **T2** (Section 1.1) by proposing a least mean squares strategy for learning the relation between reference and target graph signals. We treat the case of streaming data where both signals are defined over the graph, such that the relation between these signals can be modeled as a graph filter. In particular, we propose nonlinear graph filters that generalize the graph filters presented in Chapter 2. This research is documented in papers **P5** [75] and **P6** [76] (Section 1.3.1).

Section 5.1 reviews state-of-the-art methodologies for adaptive learning of graph filters available in the literature. Sections 5.2 and 5.3 propose nonlinear graph filters and the methodology for learning the graph-filter parameters. In Section 5.4, we present convergence conditions for the proposed learning algorithms. Numerical experiments are presented in Section 5.6 and final remarks for this chapter are presented in Section 5.7.

5.1 Graph Diffusion LMS

Consider a graph $\mathcal{G} = \{\mathcal{V}, \mathbf{A}\}$ with K nodes, and a set of streaming signals $\{\mathbf{x}_n, \mathbf{y}_n\}_{n=0}^{N-1}$, where $\mathbf{x}_n$, or $\mathbf{y}_n$, or both are graph signals. That is, at time n, a pair of signals given by $\mathbf{x}_n$ and $\mathbf{y}_n$ is available and at least one of them is defined over the graph. Under the assumption that there is an unknown relation between $\mathbf{x}_n$ and $\mathbf{y}_n$, learning over graphs addresses the identification of a set of parameters that models such relation, taking into consideration the graph structure.

In this section, we briefly present the research developed in [27, 28], which motivates the work conducted in papers **P5** and **P6**. In [28], the authors explore concepts of adaptive learning over networks together with fundamentals of GSP and propose distributed diffusion-based adaptive learning of graph filters.

From Sections 2.5.1 and 2.6.2, recall that an LSI graph filter of size $L\times 1$ combines shifted graph signals and is given by

$$\mathbf{H} = \sum_{i=0}^{L-1} h_i \mathbf{S}^i, \tag{5.1}$$

where $[h_0\, h_1\, \dots\, h_{L-1}]^{\mathrm{T}}$ is the linear graph filter coefficient vector. For a graph signal $\mathbf{x}_n$, which is a snapshot of the graph state at time n, a model for the graph-filtered vector is given by

$$\mathbf{y}_n = \sum_{i=0}^{L-1} h_i \mathbf{S}^i \mathbf{x}_n + \boldsymbol{v}_n, \tag{5.2}$$

where $\boldsymbol{v}_n = [v_{1,n}\, v_{2,n}\, \dots\, v_{K,n}]^{\mathrm{T}}$ is a zero-mean wide-sense stationary (WSS) noise with covariance matrix $\mathbf{R}_v = \mathrm{diag}\{\sigma^2_{v,1}, \sigma^2_{v,2}, \dots, \sigma^2_{v,K}\}$. When streaming data is available, a two-dimensional graph-time filter [35] that embeds time-evolution into the model can be employed, with output given by

$$\mathbf{y}_n = \sum_{i=0}^{L-1} \sum_{j=0}^{M-1} h_{i,j} \mathbf{S}^i \mathbf{x}_{n-j} + \boldsymbol{v}_n, \tag{5.3}$$

where $M-1$ is the filter memory in temporal domain. The model (5.3) uses walks of up to length $L-1$ in the graph. Thus, it requires multihop communication in distributed implementations, which limits its usage in real-time applications.

A simplified model that avoids multihop communications can be constructed by combining time and graph domains into one, as

$$\mathbf{y}_n = \sum_{i=0}^{L-1} h_i \mathbf{S}^i \mathbf{x}_{n-i} + \boldsymbol{v}_n. \tag{5.4}$$

In (5.4), samples $\{x_{k,n}, [\mathbf{S}\mathbf{x}_{n-1}]_k, \dots, [\mathbf{S}^{L-1}\mathbf{x}_{n-L+1}]_k\}$ are available locally at node k. Thus, only one graph-shift operation is needed at each time instant. An important difference between this GSP approach and conventional single- and multi-variate DSP approaches lies in the assumption that the signals' spatio-temporal dynamics depend on the graph structure.

Assume that the graph signal $\mathbf{x}_n$ is a zero-mean wide-sense stationary (WSS) process with auto-correlation $\mathbf{R}_x$. The algorithms proposed in [28] aim at learning the relation between $\mathbf{x}_n$ and $\mathbf{y}_n$ by solving the optimization problem

$$\min_{\mathbf{h}} \mathrm{E}\left[\|\mathbf{y}_n - \mathbf{X}_{S,n}\mathbf{h}\|^2\right], \tag{5.5}$$

where $\mathbf{X}_{S,n} = \begin{bmatrix}\mathbf{x}_n \; \mathbf{S}\mathbf{x}_{n-1} \; \dots \; \mathbf{S}^{L-1}\mathbf{x}_{n-L+1}\end{bmatrix} \in \mathbb{R}^{K\times L}$. That is, the learning problem is the identification of the parameters of the LSI filter that minimizes the mean square error between the filtered input in $\mathbf{X}_{S,n}\mathbf{h}$ and the target signal $\mathbf{y}_n$, according to the model (5.4).

Setting the gradient of the cost function in (5.5) to zero, the optimal parameter vector $\mathbf{h}_{\text{opt}}$ is given by the solution of

$$\mathrm{E}[\mathbf{X}_{S,n}^{\mathrm{T}}\mathbf{X}_{S,n}]\mathbf{h}_{\text{opt}} = \mathrm{E}[\mathbf{X}_{S,n}^{\mathrm{T}}\mathbf{y}_n]. \tag{5.6}$$

Since these second order moments are not commonly available in practice, the authors in [28] adopt the instantaneous approximations $\mathrm{E}[\mathbf{X}_{S,n}^{\mathrm{T}}\mathbf{X}_{S,n}] \approx \mathbf{X}_{S,n}^{\mathrm{T}}\mathbf{X}_{S,n}$ and $\mathrm{E}[\mathbf{X}_{S,n}^{\mathrm{T}}\mathbf{y}_n] \approx \mathbf{X}_{S,n}^{\mathrm{T}}\mathbf{y}_n$ and derive the stochastic-gradient (SG) algorithm

$$\mathbf{h}_{n+1}^{\text{cent}} = \mathbf{h}_i^{\text{cent}} + \mu\mathbf{X}_{S,n}^{\mathrm{T}}\left(\mathbf{y}_n - \mathbf{X}_{S,n}\mathbf{h}_n^{\text{cent}}\right), \tag{5.7}$$

where $\mu > 0$ is the step size. The algorithm (5.7) is called the centralized graph-LMS algorithm. This solution assumes that the values of the graph signals on all nodes $\{x_{k,n}, y_{k,n}\}_{k=1}^{K}$ are known by a centralized processing unit.

5.1.1 Distributed Graph-Diffusion LMS

From (5.4), the sample at node k of the target signal $\mathbf{y}_n$ can be written as

$$y_{k,n} = \sum_{i=0}^{L-1} h_i[\mathbf{S}^i\mathbf{x}_{n-i}]_k + v_{k,n}. \tag{5.8}$$

By defining $\mathbf{r}_{k,n} = \begin{bmatrix} x_{k,n} \; [\mathbf{S}\mathbf{x}_{n-1}]_k \; \dots \; [\mathbf{S}^{L-1}\mathbf{x}_{n-L+1}]_k \end{bmatrix}^{\mathrm{T}}$, i.e., the column vector with entries equal to those of the kth row of $\mathbf{X}_{S,n}$, (5.8) can be rewritten as

$$y_{k,n} = \mathbf{r}_{k,n}^{\mathrm{T}}\mathbf{h} + v_{k,n}. \tag{5.9}$$

Now, the minimization problem can be written in the equivalent form

$$\min_{\mathbf{h}} \sum_{k=1}^{K} |y_{k,n} - \mathbf{r}_{k,n}^{\mathrm{T}}\mathbf{h}|^2. \tag{5.10}$$

Problem (5.10) can be solved separately at each node k in a decentralized manner, that is, each node performs only local computations and communications with neighboring nodes, without the need of a centralized processing unit. The proposed methodology in [28] is to leverage the graph structure and use adaptive diffusion strategies [41, 42] to derive a fully distributed minimization of (5.10). The adaptive diffusion strategies used are the adapt-then-combine (ATC) and the combine-than-adapt (CTA) [42]. Let $\mathbf{h}_{k,n}$ denote the estimate of $\mathbf{h}_{\text{opt}}$ at node k at time n. The ATC strategy first updates the estimate locally at node k, generating an intermediate estimate $\boldsymbol{\psi}_{k,n+1}$, then shares its intermediate estimate with neighboring nodes. Then, each node adjusts its own estimate by combining it with the values received from its neighbors. The ATC diffusion LMS performed by the kth node is given by

$$\begin{cases} \boldsymbol{\psi}_{k,n+1} = \mathbf{h}_{k,n} + \mu_k \mathbf{r}_{k,n}(y_{k,n} - \mathbf{r}_{k,n}^{\mathrm{T}} \mathbf{h}_{k,n}), & (5.11\text{a}) \\ \mathbf{h}_{k,n+1} = \sum_{l \in \mathcal{N}_k} a_{lk}\, \boldsymbol{\psi}_{l,n+1}, & (5.11\text{b}) \end{cases}$$

where $\mu_k > 0$ is a local step size and $a_{lk} \geq 0$ are combination coefficients that satisfy $\sum_{l=1}^{K} a_{lk} = 1$ and $a_{lk} = 0$ iff $l \notin \mathcal{N}_k$.

The CTA strategy follows similar reasoning, only inverting the order of the steps. First, a node combines the estimates of its neighbors to obtain the intermediate estimate $\boldsymbol{\psi}_{k,n}$. Then, it updates the intermediate estimate locally using its own data. The CTA diffusion LMS at the kth node is given by

$$\begin{cases} \boldsymbol{\psi}_{k,n} = \sum_{l \in \mathcal{N}_k} a_{lk}\, \mathbf{h}_{l,n}, & (5.12\text{a}) \\ \mathbf{h}_{k,n+1} = \boldsymbol{\psi}_{k,n} + \mu_k \mathbf{r}_{k,n}(y_{k,n} - \mathbf{r}_{k,n}^{\mathrm{T}} \boldsymbol{\psi}_{k,n}), & (5.12\text{b}) \end{cases}$$

5.1.2 Discussion and Remarks

The methodology proposed in [28] offers diffusion strategies for adaptive learning of graph signals that can handle dynamic scenarios by blending concepts of GSP and adaptation over networks. The concepts discussed in this section are strongly related to the fundamentals of GSP discussed in Chapter 2. The graph filter is used as the model for the relation between graph signals, and it captures the temporal evolution of the signals. Additionally, the graph structure is used as a basis for the diffusion-based algorithms.

In many real-world applications, linear models such as the one presented in this section cannot fully capture more sophisticated input-output relations that induce nonlinearities in the model.

5.2 Introduction to Graph Kernel LMS

In Section 5.1, we presented the fundamentals of learning over graphs and state-of-the-art LMS-based techniques for learning linear graph filters that model relations between graph signals. As discussed, linear models cannot model more sophisticated input-output relations that often appear in real-world applications. Here, we introduce nonlinear graph filters and present two adaptive methods for function estimation over graphs, namely the centralized graph kernel least mean squares (GKLMS) and the graph diffusion kernel least mean squares (GDKLMS). The proposed nonlinear graph filters generalize conventional linear graph filters and consist of a nonlinearity applied to a combination of graph-shifted versions of the input signal.

For the estimation methods, we consider two approaches for model reduction, namely coherence check (CC) [55, 63] that sparsifies the original dictionary of the GKLMS, and random Fourier features (RFF) [140] that approximate kernel evaluations with inner products in a fixed-dimensional space. One of the main features of the CC-based implementation is the automatic tuning of the model order by selecting regressors based on a coherence measure [63]. On the other hand, RFF-based implementations use a data-independent mapping into a space where kernel evaluations can be approximated as inner-products, making them resilient to model changes. Building upon ideas of network diffusion [41, 42], the proposed RFF-based graph diffusion KLMS (GDKLMS) avoids the centralized processing and updates local estimates at each node through collaboration with neighbors. One of the main features of the RFF-based GDKLMS is its data-independent mapping that avoids using a pre-trained dictionary. This makes the GDKLMS more robust to changes in the underlying system since there is no need to retrain dictionaries associated with distributed CC-based solutions [63]. We analyze the performance of the GDKLMS and establish the convergence conditions in both mean and mean-squared senses.

5.3 Graph Kernel Adaptive Filters

As defined in Section 5.1, the shifted-input vector observed at the kth node at time n is given by

$$\mathbf{r}_{k,n} = \begin{bmatrix} x_{k,n} & [\mathbf{S}\mathbf{x}_{n-1}]_k & \dots & [\mathbf{S}^{L-1}\mathbf{x}_{n-L+1}]_k \end{bmatrix}^{\mathrm{T}}, \tag{5.13}$$

such that the linear filter operates as a linear combination of the entries of $\mathbf{r}_{k,n}$. In contrast to the linear approach, we assume a nonlinear relation between input and output, at node k, given by

$$y_{k,n} = f(\mathbf{r}_{k,n}) + v_{k,n}, \tag{5.14}$$

where $f : \mathbb{R}^L \to \mathbb{R}$ is a continuous nonlinear function on $\mathbb{R}^L$, $v_{k,n}$ is the observation noise at node k. The objective here is to identify $f(\cdot)$ at each node k given a set of data pairs $\{\mathbf{r}_{k,i}, y_{k,i}\}$, $i \in \{1, 2, \ldots, n\}$.

In order to estimate the nonlinear function $f(\cdot)$ in (5.14), kernel methods first map the input regressors $\{\mathbf{r}_{k,i}\}_{i=1,k=1}^{n,K}$ into a higher dimensional feature space where $f(\cdot)$ takes a linear form [52, 61]. This mapping is denoted by $\kappa(\cdot, \mathbf{r}_{k,i})$, in which $\kappa(\cdot,\cdot) : \mathbb{R}^L \times \mathbb{R}^L \to \mathbb{R}$ is a reproducing kernel [52, 62, 63].

5.3.1 Graph Kernel LMS

In the GSP context, K new data samples are available at each time instant. Then, given a set of regressors $\{\mathbf{r}_{k,i}\}_{i=1,k=1}^{n,K}$, the graph function $f(\cdot)$ can be expressed as a kernel expansion in terms of the mapped data as

$$f(\cdot) = \sum_{i=1}^{n} \sum_{k=1}^{K} \alpha_{ik}\, \kappa(\cdot, \mathbf{r}_{k,i}), \tag{5.15}$$

such that the corresponding estimate of $y_{l,n}$, at node l, is given by

$$\hat{y}_{l,n} = f(\mathbf{r}_{l,n}) = \sum_{i=1}^{n} \sum_{k=1}^{K} \alpha_{ik}\, \kappa(\mathbf{r}_{l,n}, \mathbf{r}_{k,i}). \tag{5.16}$$

The coefficients of the expansion in (5.16) are obtained through the following minimization problem:

$$\begin{aligned} &\min_{\alpha_{ik} \in \mathbb{R}} \sum_{l=1}^{K} \mathrm{E}\left[\left(y_{l,n} - \sum_{i=1}^{n} \sum_{k=1}^{K} \alpha_{ik}\, \kappa(\mathbf{r}_{l,n}, \mathbf{r}_{k,i})\right)^2\right] \\ &= \min_{\bar{\boldsymbol{\alpha}}_n \in \mathbb{R}^{nK}} \mathrm{E}\left[\|\mathbf{y}_n - \mathbf{K}_n\, \bar{\boldsymbol{\alpha}}_n\|_2^2\right], \end{aligned} \tag{5.17}$$

where $\mathrm{E}[\cdot]$ denotes the expected value of the argument, $\bar{\boldsymbol{\alpha}}_n^{\mathrm{T}} = [\boldsymbol{\alpha}_1^{\mathrm{T}}\, \boldsymbol{\alpha}_2^{\mathrm{T}} \, \ldots \, \boldsymbol{\alpha}_n^{\mathrm{T}}]$, with $\boldsymbol{\alpha}_i^{\mathrm{T}} = [\alpha_{i1}\, \alpha_{i2} \, \ldots \, \alpha_{iK}]$, and the matrix

$$\mathbf{K}_n = [\mathbf{K}_{1,n}\, \mathbf{K}_{2,n} \, \ldots \, \mathbf{K}_{n,n}] \in \mathbb{R}^{K \times nK} \tag{5.18}$$

is a Gram matrix with $[\mathbf{K}_{i,n}]_{l,k} = \kappa(\mathbf{r}_{l,n}, \mathbf{r}_{k,i})$ for $k, l \in \mathcal{V}$. Using a stochastic-gradient approach, the update equation for the graph KLMS (GKLMS) is

$$\bar{\boldsymbol{\alpha}}_{n+1} = \begin{bmatrix} \bar{\boldsymbol{\alpha}}_n \\ \mathbf{0}_K \end{bmatrix} + \mu\, \mathbf{K}_n^{\mathrm{T}} \left(\mathbf{y}_n - \mathbf{K}_n \begin{bmatrix} \bar{\boldsymbol{\alpha}}_n \\ \mathbf{0}_K \end{bmatrix}\right), \tag{5.19}$$

where $\mu > 0$ is the step size. Note that the number of coefficients increases by K for every time instant n, so K zeros must be appended at the end of the coefficient vector. The proposed GKLMS is summarized in Algorithm 2.

Algorithm 2: GKLMS

```
Input: step size $\mu$
Initialization: $\boldsymbol{\alpha}_0 = \textit{empty vector}$;
for each time instant $n$ do
    Input: $\mathbf{y}_n, \{\mathbf{r}_{k,n}\}_{k=1}^{K}$
    compute $\mathbf{K}_n = [\mathbf{K}_{1,n}\, \mathbf{K}_{2,n} \,\dots\, \mathbf{K}_{n,n}]$;
    update $\bar{\boldsymbol{\alpha}}_{n+1} = \begin{bmatrix} \bar{\boldsymbol{\alpha}}_n \\ \mathbf{0}_K \end{bmatrix} + \mu\, \mathbf{K}_n^{\mathrm{T}} \left( \mathbf{y}_n - \mathbf{K}_n \begin{bmatrix} \bar{\boldsymbol{\alpha}}_n \\ \mathbf{0}_K \end{bmatrix} \right)$;
    store regressors $\{\mathbf{r}_{k,n}\}_{k=1}^{K}$;
end
```

5.3.2 Graph Kernel LMS using Coherence-check

As follows from (5.16), the model order grows with both time, n, and network size, K. This increase makes this model unsuitable for real-time applications and large-scale networks. The growing dimensionality of the dictionary is a well-known issue in single-node kernel methods [55, 56, 61–65].

Several solutions have been proposed that learn a sparse, or fixed-size dictionary. Of these, the coherence-based sparsification schemes use a coherence metric [55, 63] between a candidate regressor and the current dictionary to decide whether to include the candidate in the dictionary. The coherence metric for each regressor $\mathbf{r}_{l,n}$ is given by $\delta_{l,n} = \max_{\mathbf{r}_j \in \mathcal{D}_n} |\kappa(\mathbf{r}_{l,n}, \mathbf{r}_j)|$, where $\mathcal{D}_n$ denotes the dictionary obtained before testing regressor $\mathbf{r}_{l,n}$; the dictionary starts empty before running the algorithm. Given a predefined threshold, $\delta_{\mathrm{CC}} > 0$, if $\delta_{l,n} < \delta_{\mathrm{CC}}$, the regressor is added to the dictionary. Using the coherence check criterion [55, 63], $\hat{y}_{l,n}$ in (5.16) can be rewritten as

$$\hat{y}_{l,n} = \sum_{i \in \mathcal{M}_n} \sum_{k \in \mathcal{K}_i} \widetilde{\alpha}_{ik}\, \kappa(\mathbf{r}_{l,n}, \mathbf{r}_{k,i}), \tag{5.20}$$

where $\mathcal{M}_n$ is a set of time instants (up to time instant n) in which at least one input regressor is added to the dictionary, with $|\mathcal{M}_n| \leq n$, and $\mathcal{K}_i$ is a set of node indices of the regressors that passed the coherence check at time index i, with $|\mathcal{K}_i| \leq K$. Under the CC criterion, at time index n, the dictionary $\mathcal{D}_n$ contains $|\mathcal{D}_n| = \sum_{i \in \mathcal{M}_n} |\mathcal{K}_i|$ regressors. It can be shown that the maximum number of regressors in the dictionary is finite under a set of conditions [63].

Using the stochastic-gradient approach, we obtain the following update rule of the centralized GKLMS using coherence check:

$$\bar{\widetilde{\boldsymbol{\alpha}}}_{n+1} = \begin{bmatrix} \bar{\widetilde{\boldsymbol{\alpha}}}_n \\ \mathbf{0}_{|\mathcal{K}_n|} \end{bmatrix} + \mu\, \widetilde{\mathbf{K}}_n^{\mathrm{T}} \left(\mathbf{y}_n - \widetilde{\mathbf{K}}_n \begin{bmatrix} \bar{\widetilde{\boldsymbol{\alpha}}}_n \\ \mathbf{0}_{|\mathcal{K}_n|} \end{bmatrix} \right), \tag{5.21}$$

where $\bar{\widetilde{\boldsymbol{\alpha}}}_n^{\mathrm{T}} = [\widetilde{\boldsymbol{\alpha}}_1^{\mathrm{T}}\, \widetilde{\boldsymbol{\alpha}}_2^{\mathrm{T}} \,\dots\, \widetilde{\boldsymbol{\alpha}}_{|\mathcal{M}_n|}^{\mathrm{T}}]$, with $\widetilde{\boldsymbol{\alpha}}_i^{\mathrm{T}} = [\widetilde{\alpha}_{i1}\, \widetilde{\alpha}_{i2} \,\dots\, \widetilde{\alpha}_{i|\mathcal{K}_i|}] \in \mathbb{R}^{|\mathcal{K}_i|}$, and $\widetilde{\mathbf{K}}_n =$

Algorithm 3: GKLMS using coherence check

```
Input: training data $\{\tilde{\mathbf{r}}_{l,i}, \tilde{y}_{l,i}\}_{l=1,i=1}^{K,t}$, dictionary size $|\mathcal{D}|$, CC parameters,
       and step size $\mu$
Initialization: $\mathcal{D} = \emptyset$, $\boldsymbol{\alpha}_0 =$ empty vector;
for each time instant $n$ do
    Input: $\mathbf{y}_n, \{\mathbf{r}_{k,n}\}_{k=1}^{K}$
    for $k = 1, \ldots, K$ do
        if $|\mathcal{D}| < D$ then
            compute $\delta_{k,n} = \max_{\mathbf{r}_j \in \mathcal{D}} |\kappa(\mathbf{r}_{k,n}, \mathbf{r}_j)|$;
            if $\delta_{k,n} < \delta$ then
                add $\mathbf{r}_{k,n}$ to $\mathcal{D}$;
                add $k$ to $\mathcal{K}_n$;
            end
        end
    end
    if $|\mathcal{K}_n| \neq 0$ then
        add $n$ to $\mathcal{M}_n$;
    end
    compute $\widetilde{\mathbf{K}}_n = [\widetilde{\mathbf{K}}_{1,n}\, \widetilde{\mathbf{K}}_{2,n} \ldots \widetilde{\mathbf{K}}_{|\mathcal{M}_n|,n}]$;
    update $\bar{\bar{\boldsymbol{\alpha}}}_{n+1} = \begin{bmatrix} \bar{\bar{\boldsymbol{\alpha}}}_n \\ \mathbf{0}_{|\mathcal{K}_n|} \end{bmatrix} + \mu\, \widetilde{\mathbf{K}}_n^{\mathrm{T}} \left( \mathbf{y}_n - \widetilde{\mathbf{K}}_n \begin{bmatrix} \bar{\bar{\boldsymbol{\alpha}}}_n \\ \mathbf{0}_{|\mathcal{K}_n|} \end{bmatrix} \right)$;
end
```

$[\widetilde{\mathbf{K}}_{1,n}\, \widetilde{\mathbf{K}}_{2,n} \ldots \widetilde{\mathbf{K}}_{|\mathcal{M}_n|,n}] \in \mathbb{R}^{K \times |\mathcal{D}_n|}$, with $[\widetilde{\mathbf{K}}_{i,n}]_{l,k} = \kappa(\mathbf{r}_{l,n}, \mathbf{r}_{k,i})$, for $l \in \mathcal{V}$ and $k \in \mathcal{K}_i$. Steps for the GKLMS using CC are summarized in Algorithm 3.

A CC-based distributed implementation requires a pre-trained dictionary available at each node [55]. The dictionary can be pre-trained in a centralized way and broadcasted to the entire network, or by a single node that shares its dictionary with all nodes. More importantly, the dictionary depends on available training data, and must be retrained whenever there are changes in the underlying model.

In the next section, we propose RFF-based algorithms more suitable for distributed implementations and robust to changes in model and data statistics.

5.3.3 Graph Kernel LMS using Random Fourier Features

Random Fourier features is a widely used technique to circumvent the scaling problems of kernel methods [140]. The RFF methodology presumes that the evaluation of a shift-invariant kernel, which satisfies $\kappa(\mathbf{x}_m, \mathbf{x}_n) = \kappa(\mathbf{x}_m - \mathbf{x}_n, 0)$, can be approximated as an inner product in the D-dimensional RFF space. This turns the problem into a finite-dimension linear filtering problem while avoiding the evaluation of kernel functions. The mapping of an input vector $\mathbf{x}_n$ into the RFF space $\mathbb{R}^D$

is denoted by $\mathbf{z}_n$, and is given by

$$\mathbf{z}_n = (D/2)^{-\frac{1}{2}} \left[\cos(\mathbf{v}_1^{\mathrm{T}}\mathbf{x}_n + b_1) \; \ldots \; \cos(\mathbf{v}_D^{\mathrm{T}}\mathbf{x}_n + b_D)\right]^{\mathrm{T}}, \tag{5.22}$$

where the phase terms $\{b_i\}_{i=1}^{D}$ are drawn from a uniform distribution on the interval $[0,\, 2\pi]$. Vectors $\{\mathbf{v}_i\}_{i=1}^{D}$ are realizations of a random variable with probability density function (pdf) $p(\mathbf{v})$ such that

$$\kappa(\mathbf{x}_m, \mathbf{x}_n) = \int p(\mathbf{v}) \exp\left(\mathrm{j}\mathbf{v}^{\mathrm{T}}(\mathbf{x}_m - \mathbf{x}_n)\right) \mathrm{d}\mathbf{v}. \tag{5.23}$$

In other words, the Fourier transform of $\kappa(\mathbf{x}_m, \mathbf{x}_n)$ is given by $p(\mathbf{v})$. From (5.22) and (5.23), it can be verified that $\mathrm{E}[\mathbf{z}_n^{\mathrm{T}}\mathbf{z}_m] = \kappa(\mathbf{x}_m, \mathbf{x}_n)$ [140]. Then, the kernel evaluation can be approximated as $\kappa(\mathbf{x}_m, \mathbf{x}_n) \approx \mathbf{z}_n^{\mathrm{T}}\mathbf{z}_m$.

The Gaussian kernel given by $\kappa(\mathbf{x}_m, \mathbf{x}_n) = \exp\left(-\|\mathbf{x}_m - \mathbf{x}_n\|_2^2/(2\sigma^2)\right)$ is commonly used in the literature [31, 110]. In this case, the pdf $p(\mathbf{v})$ is given in closed form as a normal distribution. Other closed-form representations of $p(\mathbf{v})$ corresponding to other kernel functions can be found in [140].

Let $\mathbf{z}_{l,n}$ be the mapping of $\mathbf{r}_{l,n}$ into the RFF space $\mathbb{R}^D$. The estimate $\hat{y}_{l,n}$ in (5.16) can be approximated by

$$\hat{y}_{l,n} \approx \Big(\sum_{i=1}^{n}\sum_{k=1}^{K} \alpha_{ik}\, \mathbf{z}_{k,i}\Big)^{\mathrm{T}} \mathbf{z}_{l,n} = \mathbf{h}^{\mathrm{T}}\mathbf{z}_{l,n}, \tag{5.24}$$

where $\mathbf{h} \in \mathbb{R}^D$ is the representation of the function $f(\cdot)$ in the RFF space.

The linear representation of $f(\cdot)$ in the RFF space, $\mathbf{h}$, can be estimated by solving the following optimization problem:

$$\min_{\mathbf{h}\in\mathbb{R}^D} \mathrm{E}\left[\|\mathbf{y}_n - \mathbf{Z}_n^{\mathrm{T}}\mathbf{h}\|_2^2\right], \tag{5.25}$$

where $\mathbf{Z}_n = [\mathbf{z}_{1,n}\, \mathbf{z}_{2,n} \, \ldots \, \mathbf{z}_{K,n}]$ represents the RFF mapping of all input vectors at time n. Approximating the solution through stochastic-gradient descent iterations yields the RFF-based centralized graph kernel LMS (GKLMS) update rule

$$\mathbf{h}_{n+1} = \mathbf{h}_n + \mu \mathbf{Z}_n \mathbf{e}_n, \tag{5.26}$$

where $\mathbf{e}_n = \mathbf{y}_n - \mathbf{Z}_n^{\mathrm{T}}\mathbf{h}_n$. The RFF-based GKLMS is summarized in Algorithm 4.

Algorithm 4: GKLMS using RFF

```
Input: RFF-space dimension $D$, pdf $p(\mathbf{v})$, step size $\mu$
Initialization:
draw vectors $\{\mathbf{v}_i\}_{i=1}^{D}$ from $p(\mathbf{v})$;
draw phase terms $\{b_i\}_{i=1}^{D}$ from $[0, 2\pi]$;
$\mathbf{h}_0 = \mathbf{0}_D$;
%Learning
for each time instant $n$ do
    Input: $\mathbf{y}_n, \{\mathbf{r}_{k,n}\}_{k=1}^{K}$
    compute $\{\mathbf{z}_{l,n}\}_{l=1}^{K}$;
    construct matrix $\mathbf{Z}_n = [\mathbf{z}_{1,n}\, \mathbf{z}_{2,n}\, \dots\, \mathbf{z}_{K,n}]$;
    update $\mathbf{h}_{n+1} = \mathbf{h}_n + \mu \mathbf{Z}_n \mathbf{e}_n$;
end
```

5.3.4 Graph Diffusion Kernel LMS using RFF

In order to derive a distributed implementation, the global optimization problem (5.25) is expressed alternatively in the following separable form:

$$\underset{\boldsymbol{\psi}_1,\dots,\boldsymbol{\psi}_K \in \mathbb{R}^D}{\arg\min} \sum_{k=1}^{K} \mathrm{E}\big[(y_{k,n} - \mathbf{z}_{k,n}^{\mathrm{T}} \boldsymbol{\psi}_k)^2\big], \tag{5.27}$$

where $\boldsymbol{\psi}_k$ is the local estimate of $\mathbf{h}$ at node k.

We use the ATC strategy, presented in Section 5.1, to solve (5.27) in a fully distributed fashion, i.e., nodes operate only with information from their local neighborhoods. The update rule for the GDKLMS using RFF is given by

$$\begin{cases} \boldsymbol{\psi}_{k,n+1} = \mathbf{h}_{k,n} + \mu\, e_{k,n} \mathbf{z}_{k,n}, & (5.28\text{a}) \\ \mathbf{h}_{k,n+1} = \sum_{l \in \mathcal{N}_k} a_{lk}\, \boldsymbol{\psi}_{l,n+1}, & (5.28\text{b}) \end{cases}$$

where $e_{k,n} = y_{k,n} - \mathbf{z}_{k,n}^{\mathrm{T}} \boldsymbol{\psi}_k$, and the combination coefficients a_{lk} are non-negative and satisfy the condition $\sum_{l \in \mathcal{N}_k} a_{lk} = 1$ [41]. The similar combine-then-adapt (CTA) strategy [141] can also be derived for the GDKLMS. Algorithm 5 summarizes the steps for the proposed GDKLMS implementation using RFF.

5.4 Convergence Analysis

Given a node $k \in \mathcal{N}$, let the RFF-mapped data signal $\mathbf{z}_{k,n}$ be drawn from a WSS multivariate random sequence with correlation matrix $\mathbf{R}_{z,k} = \mathrm{E}[\mathbf{z}_{k,n} \mathbf{z}_{k,n}^{\mathrm{T}}]$. Under a set of reasonable assumptions [76], it is possible to show that

Theorem 5.1. A sufficient condition for the proposed RFF-based GDKLMS to con-

Algorithm 5: GDKLMS using RFF

```
Input: RFF-space dimension $D$, pdf $p(\mathbf{v})$, step size $\mu$, combination
       coefficients $a_{lk}$
Initialization:
draw vectors $\{\mathbf{v}_i\}_{i=1}^{D}$ from $p(\mathbf{v})$;
draw phase terms $\{b_i\}_{i=1}^{D}$ from $[0, 2\pi]$;
$\mathbf{h}_{k,0} = \mathbf{0}_D$, $\forall k \in \{1, 2, \dots, K\}$;
$\boldsymbol{\psi}_{k,0} = \mathbf{0}_D$, $\forall k \in \{1, 2, \dots, K\}$;
%Learning
for each time instant $n$ do
    for $k = 1, \dots, K$ do
        compute $\mathbf{z}_{k,n}$;
        update $\boldsymbol{\psi}_{k,n+1} = \mathbf{h}_{k,n} + \mu\, e_{k,n}\mathbf{z}_{k,n}$;
    end
    for $k = 1, \dots, K$ do
        update $\mathbf{h}_{k,n+1} = \sum_{l \in \mathcal{N}_k} a_{lk}\, \boldsymbol{\psi}_{l,n+1}$;
    end
end
```

verge in mean is given by

$$0 < \mu < \frac{2}{\max\limits_{1 \le k \le K} \left\{ \max\limits_{1 \le i \le D} \{\lambda_i(\, \mathbf{R}_{z,k})\} \right\}}, \tag{5.29}$$

and

Theorem 5.2. Assuming that the step size μ is sufficiently small, the proposed RFF-based GDKLMS converges in mean-squared sense under

$$0 < \mu < \frac{1}{\max\limits_{1 \le k \le K} \left\{ \max\limits_{1 \le i \le D} \{\lambda_i(\mathbf{R}_{z,k})\} \right\}}. \tag{5.30}$$

Remark 5.1. The bounds established for μ are inversely proportional to the spectral radius of the covariance matrix of vectors $\mathbf{z}_k$. Hence, similar to conventional stochastic gradient algorithms, μ requires tuning according to the largest eigenmode.

5.4.1 Steady-State Mean-Squared Error

Let $\boldsymbol{\mathcal{Z}}_n = \text{blockdiag}\,\{\mathbf{z}_{1,n}, \mathbf{z}_{2,n}, \dots, \mathbf{z}_{K,n}\}$, with $\mathbf{R}_z = \mathrm{E}[\boldsymbol{\mathcal{Z}}_n \boldsymbol{\mathcal{Z}}_n^{\mathrm{T}}]$, and $\boldsymbol{\mathcal{A}} = \mathbf{A}^{\mathrm{T}} \otimes \mathbf{I}_D$, with $A_{l,k} = a_{lk}$. We assume that the observation noise is a zero-mean WSS multivariate random sequence, with diagonal correlation matrix $\mathbf{R}_v$. For μ under (5.30), it is possible to show that the network-level steady-state mean-squared error (SMSE)

Table 5.1: Computational cost of the proposed algorithms

Algorithm\Operation	**Kernel Evaluation/ RFF mapping**	**Multiplication**	**Addition**	**Training (kernel evaluation)**	**Communication**
GKLMS	nK^2	$2nK^2 + nK$	$2nK^2$	–	–
CC-based GKLMS	$K\lvert\mathcal{D}\rvert$	$\lvert\mathcal{D}\rvert(2K+1)$	$2K\lvert\mathcal{D}\rvert$	lowerbound $\lvert\mathcal{D}\rvert(\lvert\mathcal{D}\rvert-1)/2$ upperbound $t\lvert\mathcal{D}\rvert$	–
RFF-based GKLMS	KD	$D(2K+1)$	$2KD$	–	–
RFF-based GDKLMS	KD	$KD(\lvert\mathcal{N}\rvert+3)$	$KD(\lvert\mathcal{N}\rvert+1)$	–	$KD\lvert\mathcal{N}\rvert$

of the proposed RFF-based GDKLMS is given by

$$\begin{aligned} \text{SMSE} &= \frac{1}{K}\lim_{n\to\infty} \mathrm{E}[\mathbf{e}_n^{\mathrm{T}}\mathbf{e}_n] \\ &= \frac{1}{K}\left[\mu^2\boldsymbol{\gamma}^{\mathrm{T}}(\mathbf{I}_{D^2K^2} - \boldsymbol{\mathcal{F}}^{\mathrm{T}})^{-1}\mathrm{bvec}(\mathbf{R}_z) + \mathrm{tr}(\mathbf{R}_v)\right], \end{aligned} \tag{5.31}$$

where

$$\boldsymbol{\mathcal{F}} = \mathrm{E}\left[\left(\boldsymbol{\mathcal{A}}\big(\mathbf{I}_{DK} - \mu\boldsymbol{\mathcal{Z}}_n\boldsymbol{\mathcal{Z}}_n^{\mathrm{T}}\big)\right) \otimes \left(\boldsymbol{\mathcal{A}}\big(\mathbf{I}_{DK} - \mu\boldsymbol{\mathcal{Z}}_n\boldsymbol{\mathcal{Z}}_n^{\mathrm{T}}\big)\right)\right] \tag{5.32}$$

$$\gamma = (\boldsymbol{\mathcal{A}} \otimes \boldsymbol{\mathcal{A}})\mathrm{bvec}\{\mathrm{E}[\boldsymbol{\mathcal{Z}}_n\mathbf{R}_v\boldsymbol{\mathcal{Z}}_n^{\mathrm{T}}]\}. \tag{5.33}$$

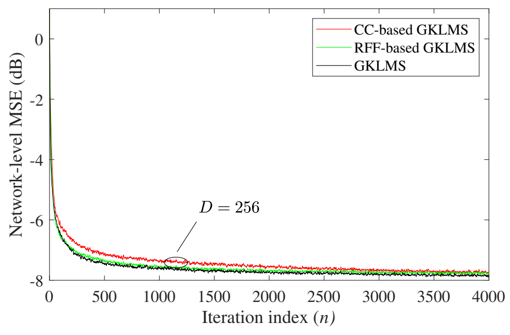
5.5 Complexity Analysis

The computational costs of the proposed algorithms are summarized in Table 5.1. The complexity of kernel evaluations is treated separately, as we do not consider a specific kernel function. Results for the GKLMS reveal that kernel methods without dimensionality control not scale well with time and network size. Considering the case where the dictionary size, $|\mathcal{D}|$, and the RFF-space dimension, D, are the same for the CC- and RFF-based implementations, their complexities per iteration are also the same.[1] The CC-based approach, however, has the added complexity of training the dictionary. For the GDKLMS using RFF, each node requires $D(|\mathcal{N}|+3)$ multiplications and $D(|\mathcal{N}|+1)$ additions, with $|\mathcal{N}|$ denoting neighborhood cardinality. Results are scaled for the entire network.

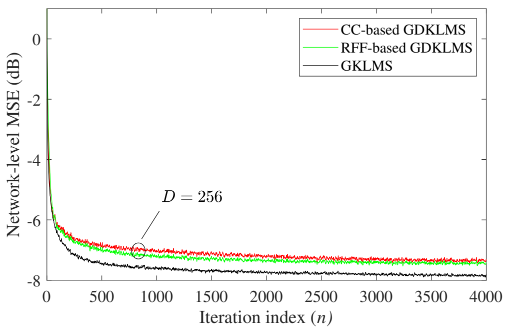
5.6 Numerical Results

This section demonstrates the performance of the proposed algorithms through extensive numerical experiments under synthetic and real network data.

[1]The assumption of $|\mathcal{D}| = D$ is reasonable as we show in the next section. The accuracies of both algorithms are comparable for similar values of $|\mathcal{D}|$ and D.

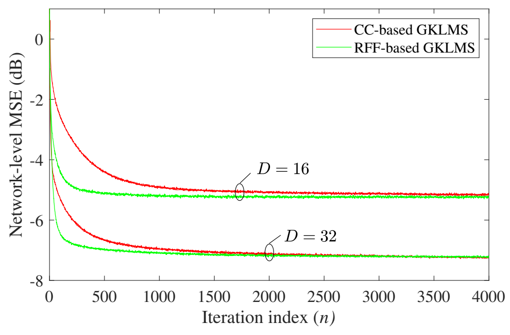


(a) Centralized solutions.

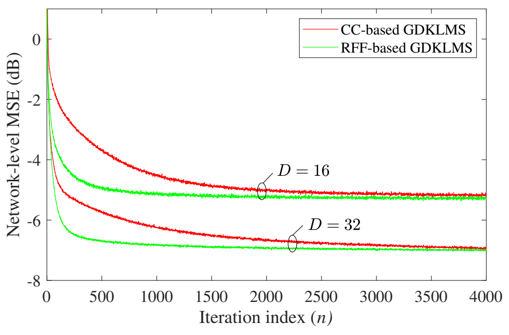


(b) Distributed solutions.

Figure 5.1: Learning curves (network-level MSE vs iteration index) for the proposed algorithms with large dictionary size and RFF-space dimension.



(a) Centralized solutions.



(b) Distributed solutions.

Figure 5.2: Learning curves (network-level MSE vs iteration index) for the proposed algorithms considering small values for D.

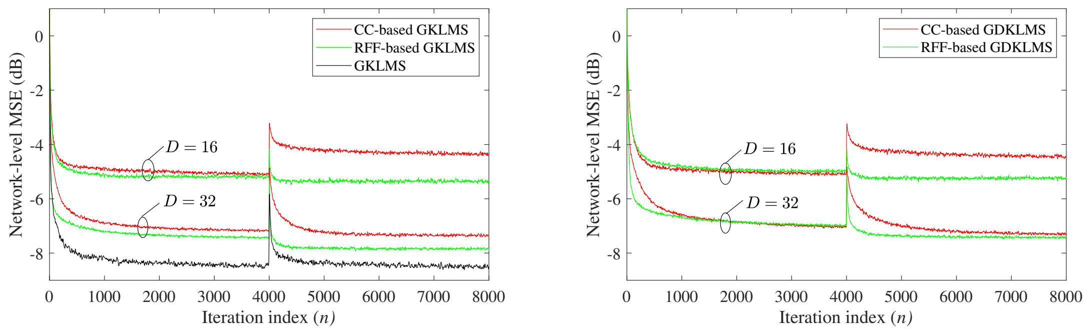
5.6.1 Nonlinear Graph Filter Identification

First, we consider a connected Erdös-Renyi graph comprising $K = 20$ nodes with edge probability equal to 0.2. For a filter of length $L = 4$, we aim at estimating the time-invariant nonlinear function given by

$$f(\mathbf{r}_{k,n}) = \sqrt{r_{k,n,1}^2 + \sin^2(r_{k,n,4}\pi)} + (0.8 - 0.5\exp(-r_{k,n,2}^2))r_{k,n,3}. \tag{5.34}$$

The network-level instantaneous MSE, given by $\text{MSE}_n = \frac{1}{K}\sum_{k=1}^{K} e_{k,n}^2$, is considered as the performance metric and results are displayed by plotting MSE_n versus the iteration index n, averaging over 1000 independent runs.

In Fig. 5.1, we present the learning curves of the approaches based on CC and RFF, with $|\mathcal{D}| = D = 256$. For large enough dictionary sizes and RFF-space dimensions, these implementations reach similar performance to that of the GKLMS implementation without sparsification. The centralized implementations can better approximate the GKLMS without sparsification when compared to the GDKLMS. This is an expected result considering that data from the entire graph is available during the learning process of the centralized approaches.



(a) Centralized solutions. (b) Distributed solutions.

Figure 5.3: Tracking performance of the proposed algorithms.

In Fig. 5.2 we compare the proposed algorithms when smaller dictionaries and RFF-space dimensions are considered. Specifically, we compare the implementations based on RFF and coherence check against each other. Results show that both CC- and RFF-based algorithms are capable of effectively representing the target function. For $|\mathcal{D}| = D$ and for similar values of network-level steady-state MSE, the RFF-based GKLMS converges faster than the CC-based one. Moreover, it can be observed that the performance of the implementations with fixed-size dictionaries greatly improves as D is increased from 16 to 32.

5.6.2 Tracking Performance of the Proposed Algorithms

We consider now the estimation of a nonlinear function given by

$$f_n(\mathbf{r}_{k,n}) = \begin{cases} \sqrt{r_{k,n,1}^2 + r_{k,n,4}^2} - r_{k,n,3}\mathrm{e}^{-r_{k,n,2}^2} & 0 < n \leq 4000 \\ \sqrt{r_{k,n,1}^2 + r_{k,n,2}^2 + r_{k,n,3}^2 + r_{k,n,4}^2} & 4000 < n. \end{cases} \tag{5.35}$$

Fig. 5.3 shows the learning curves for the centralized and distributed algorithms for two values of dictionary sizes and RFF-space dimension, namely, $D = 16$ and $D = 32$. We see that the RFF-based implementations are resilient to model changes, while the CC-based implementations suffer from noticeable performance losses, especially for small dictionaries. This is an expected behavior, since larger dictionaries can represent more functions. We also see that the GKLMS achieves the lowest MSE, however, at the cost of an unconstrained dictionary size.

5.6.3 Convergence Simulations

We verify, through numerical experiments, the theoretical bounds established for μ in Section 5.4. We compute the network-level MSE for the GDKLMS after 20000 iterations, for $\mu \in \{0.1, 0.5, 1.5, 2, 2.25, 2.5\}$, as presented in Fig. 5.4. This simu-

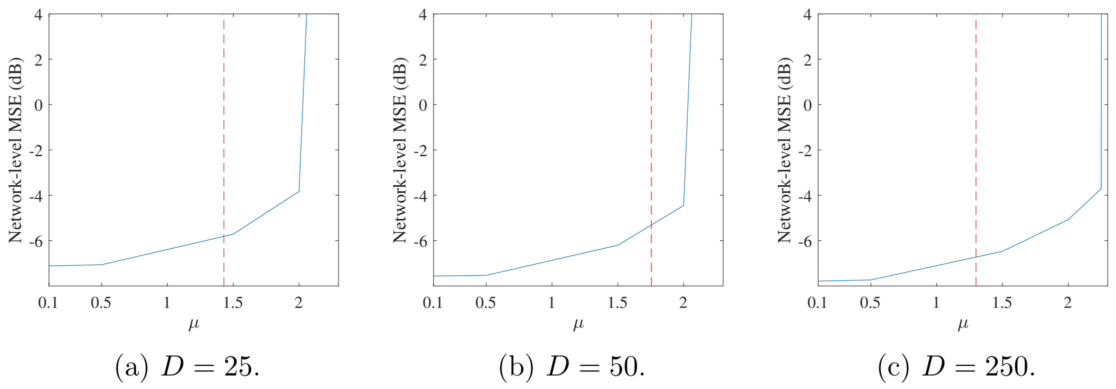


Figure 5.4: Network-level MSE after 20000 iterations vs step size for the proposed algorithms RFF-based GDKLMS. Vertical red line indicates the theoretical upper-bound according to Theorem 6.2.

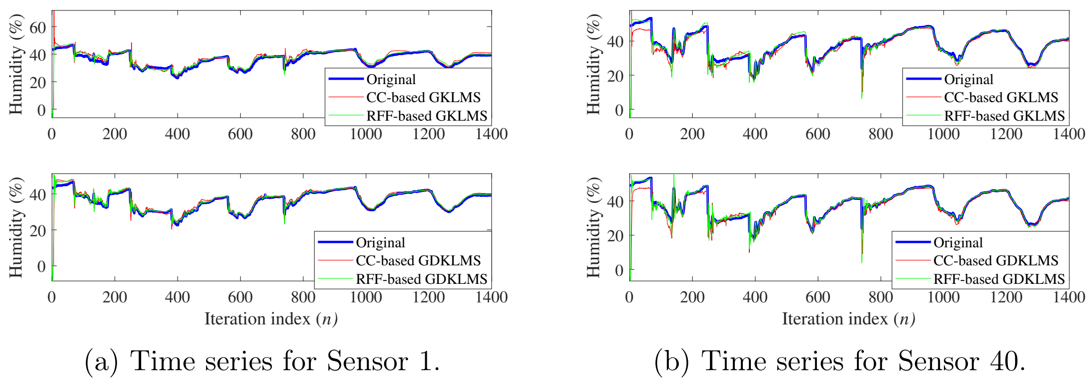


Figure 5.5: Time series of original and estimated humidity signals using the proposed algorithms for the Intel Lab dataset.

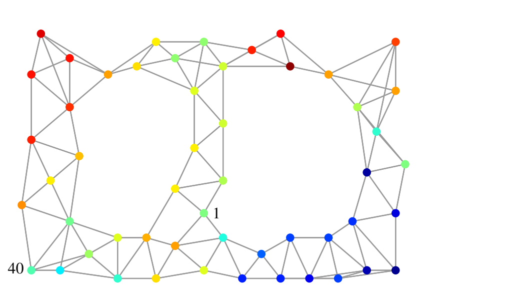
lation is repeated for different values of D and we assess if the algorithm converges in practice in accordance with the theoretical results. Results confirm that the conditions established for μ in Theorem 6.2 are sufficient for convergence.

5.6.4 Laboratory-monitoring Data

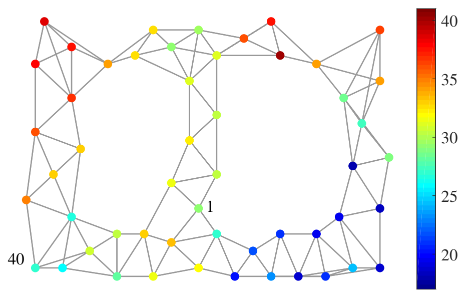
We consider the Intel Lab database [142] that contains temperature and humidity data, measured during March 2004, from 52 sensors spread across a laboratory and its common areas. The undirected graph is constructed by connecting each sensor to its four nearest neighbors and we consider the task of estimating humidity from the temperature signal.

In our simulations, we used $L = 5$ and $D = 128$, for centralized and distributed implementations. The step sizes are 0.03 for CC- and RFF-based GKLMS, and 0.5 for GDKLMS implementations.

The humidity signals from Sensors 1 and 40 are plotted in Figs. 5.5a and 5.5b, respectively, together with the estimated signals from the graph filters. The varia-



(a) Original.



(b) Estimated (RFF-based GDKLMS).

Figure 5.6: Network structure for the Intel Lab simulation and snapshots of the original and estimated humidity signals.

tions in the plots are aligned with events that induce model changes. For example, the most notable peaks are aligned with the beginning and end of work shifts. The implementations based on CC and RFF have similar performances, while the latter exhibit slightly more resilience to changes in the model. Fig. 5.6 depicts the graph representation of the Intel Lab sensor network and presents snapshots of the humidity signals, both the original and the one estimated via RFF-based GDKLMS. These results confirm that the proposed algorithms can effectively estimate the humidity level from temperature readings.

5.7 Summary

This research introduced nonlinear adaptive graph filters for model estimation in the reproducing kernel Hilbert space. A centralized graph kernel LMS algorithm was derived. Coherence-check-based dictionary-sparsification and random Fourier features were proposed to overcome complexity issues of kernel methods. Diffusion-based distributed implementations of both CC- and RFF-based GKLMS algorithms were developed using only local communications and in-network processing, and we presented the mean and mean-square-error convergence conditions for the proposed GDKLMS using RFF. Numerical simulations were conducted to demonstrate the performance of the proposed algorithms. Results confirmed that CC- and RFF-based approaches effectively estimate nonlinear graph filters, while the latter exhibits a faster convergence and is robust to model changes.

Chapter 6

Kernel Regression on Graphs using RFF

Unlike the diffusion LMS methodology presented in the previous chapter, this chapter addresses the scenario where the input signal is not necessarily a graph signal. In this sense, we propose a batch-based kernel regression method that maps a general signal, possibly agnostic to the graph, into an output signal that resides on a given graph. A penalty term, added to the loss function, achieves this mapping and enforces the graph signal at the output, whose smoothness (as discussed in Section 2.7) across the graph is defined by the graph Laplacian. This work is related to the research documented in papers **P7** [77] and **P8** [78].

Section 6.1 reviews the state-of-the art methodology for KRG. Section 6.2 proposes KRG using RFF and an efficient implementation for the batch-based algorithm. Section 6.3 presents online algorithms for KRG using RFF. Stability conditions for the online algorithms are presented in Section 6.4. In Section 6.5, a discussion on the complexity of the proposed algorithms is presented. Section 6.6 presents numerical experiments. Conclusions for this chapter are presented in Section 6.7.

6.1 Kernel Regression over Graphs

This section presents the learning methodology introduced in [31], which serves as a basis for our proposed methodology. Consider a graph $\mathcal{G} = \{\mathcal{V}, \mathbf{L}\}$ with K nodes and graph Laplacian $\mathbf{L}$. Considering a graph-based system, which takes an input vector $\mathbf{x} \in \mathbb{R}^M$ and outputs a graph signal $\mathbf{t} \in \mathbb{R}^K$, we are interested in estimating the corresponding mapping $\mathcal{M} : \mathbb{R}^M \to \mathbb{R}^K$. In [31], the model is estimated in terms of a matrix $\mathbf{W} \in \mathbb{R}^{M\times K}$ such that

$$\mathbf{y}_n = \mathbf{W}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}_n), \tag{6.1}$$

where $\mathbf{y}_n$ is an estimate of the target graph signal $\mathbf{t}_n$ and $\boldsymbol{\phi} : \mathbb{R}^M \to \mathbb{R}^M$ is an unknown function of the input signal. Recall that the quadratic form (2.46) of the graph Laplacian induces a variation metric as discussed in Section 2.7.2. By defining $\nu(\mathbf{y}) \triangleq \mathbf{y}^{\mathrm{T}}\mathbf{L}\mathbf{y}$, we have a measure of how smooth $\mathbf{y}$ is with respect to $\mathcal{G}$. Under the assumption that a graph signal is smooth according with the underlying graph, model (6.1) is expected to generate estimates $\mathbf{y}_n$ for which $\nu(\mathbf{y}_n)$ is low.

The optimal parameter matrix $\mathbf{W}$ is found by minimizing the cost function

$$C(\mathbf{W}) = \sum_{n=0}^{N-1} \|\mathbf{t}_n - \mathbf{y}_n\|_2^2 + \alpha \mathrm{tr}(\mathbf{W}^{\mathrm{T}}\mathbf{W}) + \beta \sum_{n=0}^{N-1} \nu(\mathbf{y}_n), \tag{6.2}$$

where $N \geq M$. The cost function $C(\mathbf{W})$ augments traditional regression methods by incorporating the penalty $\sum_{n=0}^{N-1} \nu(\mathbf{y}_n)$, which enforces smoothness of the output signal with respect to the graph. Defining the matrices

$$\mathbf{T} = [\mathbf{t}_0\ \mathbf{t}_1\ \ldots\ \mathbf{t}_{N-1}]^{\mathrm{T}} \in \mathbb{R}^{N\times K}, \tag{6.3}$$

$$\mathbf{Y} = [\mathbf{y}_0\ \mathbf{y}_1\ \ldots\ \mathbf{y}_{N-1}]^{\mathrm{T}} \in \mathbb{R}^{N\times K}, \tag{6.4}$$

$$\boldsymbol{\Phi} = [\boldsymbol{\phi}(\mathbf{x}_0)\ \boldsymbol{\phi}(\mathbf{x}_1)\ \ldots\ \boldsymbol{\phi}(\mathbf{x}_{N-1})]^{\mathrm{T}} \in \mathbb{R}^{N\times M}, \tag{6.5}$$

and assuming $\boldsymbol{\Phi}$ is full rank, we can make the substitution $\mathbf{W} = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Psi}$, so that the optimization is now conducted in terms of $\boldsymbol{\Psi} \in \mathbb{R}^{N\times K}$. The predicted output of the kernel regression is given by [31]

$$\begin{aligned}\mathbf{y} &= \boldsymbol{\Psi}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{\phi}(\mathbf{x}) = \boldsymbol{\Psi}^{\mathrm{T}}\mathbf{k}(\mathbf{x})\\ &= \left(\mathrm{mat}\left((\mathbf{B}+\mathbf{C})^{-1}\mathrm{vec}(\mathbf{T})\right)\right)^{\mathrm{T}} \boldsymbol{\kappa}(\mathbf{x}),\end{aligned} \tag{6.6}$$

where $\boldsymbol{\kappa}(\mathbf{x}) = [\kappa_1(\mathbf{x})\ \kappa_2(\mathbf{x})\ \ldots\ \kappa_N(\mathbf{x})]^{\mathrm{T}}$, with $\kappa_n(\mathbf{x}) = \boldsymbol{\phi}^{\mathrm{T}}(\mathbf{x}_n)\boldsymbol{\phi}(\mathbf{x})$. Also,

$$\mathbf{B} = (\mathbf{I}_K \otimes (\mathbf{K} + \alpha\mathbf{I}_N)), \tag{6.7}$$

$$\mathbf{C} = (\beta\mathbf{L} \otimes \mathbf{K}), \tag{6.8}$$

with $\mathbf{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}} \in \mathbb{R}^{N\times N}$. Here, the kernel trick is employed to avoid the explicit knowledge of $\boldsymbol{\phi}(\cdot)$, by replacing the inner product $\kappa_n(\mathbf{x}_i) = \boldsymbol{\phi}^{\mathrm{T}}(\mathbf{x}_i)\boldsymbol{\phi}(\mathbf{x}_n)$ with a kernel $\kappa(\mathbf{x}_i, \mathbf{x}_n)$ [51, 52]. The method described in (6.6), which outputs an estimate $\mathbf{y}$ for an input $\mathbf{x}$, is referred to as kernel regression on graphs.

6.1.1 Discussion and Remarks

The method proposed in [31] can handle scenarios for which previous solutions were not applicable. For example, consider the case of moving sources, whose positions correspond to the input data, and a sensor network, which corresponds to the known graph. Consider the task of estimating the signal at the sensors given the sources' positions. This case illustrates non-graph data projected onto the graph. Therefore, the input signal cannot be treated as a graph signal. Another example is the case where data on two different graphs are related to each other. In this case, a graph filter is not directly applicable to relate data from both graphs.

Remark 6.1. The regression in (6.6) is performed in a batch-based fashion, assuming that all training samples are available *a priori*. A significant drawback of this implementation is the inherent delay of batch-based implementations, as the computation of the parameter matrix $\boldsymbol{\Psi}$ must wait for all training samples $\{\mathbf{x}_n, \mathbf{t}_n\}_{n=0}^{N-1}$ to be available. Also, the increase in the computational burden of the KRG with the number of training samples is twofold. First, computing $\boldsymbol{\Psi}$ becomes more complex as the dimensions of $\mathbf{K}$ increase with N. Second, the regression dimension increases as the size of $\mathbf{k}(\mathbf{x})$ increases with N, and each additional training sample requires a kernel evaluation. The model complexity also depends on the number of training samples N, requiring N kernel evaluations for each new input signal, which is an issue if an online implementation is derived. In the following section, we address the growing complexity by proposing a batch-based approach using random Fourier features.

6.2 Batch KRG using Random Fourier Features

In Section 6.1.1, we discussed the complexity issues associated with current state-of-the-art approaches for KRG. Similar to the approach adopted in Chapter 5, we can resort to RFF to derive efficient KRG algorithms.

6.2.1 RFF-based KRG

To employ RFF in the KRG methodology, we first consider the kth entry of the estimate $\mathbf{y}$ as

$$y_k = \mathbf{w}_k^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}), \tag{6.9}$$

where $\mathbf{w}_k$ denotes the kth column of the parameter matrix $\mathbf{W}$. Using the substitution $\mathbf{W} = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Psi}$, and the kernel trick $\kappa(\mathbf{x}_m, \mathbf{x}_n) = \boldsymbol{\phi}(\mathbf{x}_m)^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)$, (6.9) can be rewritten as

$$y_k = \left(\sum_{n=1}^{N} \Psi_{n,k} \boldsymbol{\phi}(\mathbf{x}_n) \right)^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}) = \left(\sum_{n=1}^{N} \Psi_{n,k} \kappa(\mathbf{x}_n, \mathbf{x}) \right). \tag{6.10}$$

Using RFF, we can approximate (6.10) as

$$y_k \approx \sum_{n=1}^{N} \Psi_{n,k} \mathbf{z}_n^{\mathrm{T}} \mathbf{z} = \mathbf{h}_k^{\mathrm{T}} \mathbf{z}. \tag{6.11}$$

Finally, the RFF-based regression estimate for the entire graph signal is written as

$$\mathbf{y} = \mathbf{H}^{\mathrm{T}} \mathbf{z}, \tag{6.12}$$

where $\mathbf{H} = [\mathbf{h}_1\ \mathbf{h}_2\ \ldots\ \mathbf{h}_K] \in \mathbb{R}^{D\times K}$ is the representation of the regression coefficient matrix in the RFF space. Letting the matrix

$$\mathbf{Z} = [\mathbf{z}_1\ \mathbf{z}_2\ \ldots\ \mathbf{z}_N]^{\mathrm{T}} \in \mathbb{R}^{N\times D} \tag{6.13}$$

represent the RFF mapping of all training input vectors $\{\mathbf{x}_n\}_{n=1}^{N}$, and using $\mathbf{T}$ and $\mathbf{Y}$ as respectively defined in (6.3) and (6.4), the cost function (6.2) can be rewritten in terms of $\mathbf{H}$ as

$$\begin{aligned} C(\mathbf{H}) = \sum_{n=1}^{N} \|\mathbf{t}_n\|_2^2 - 2\mathrm{tr}(\mathbf{T}^{\mathrm{T}}\mathbf{Z}\mathbf{H}) + \mathrm{tr}(\mathbf{H}^{\mathrm{T}}\mathbf{Z}^{\mathrm{T}}\mathbf{Z}\mathbf{H}) \\ + \alpha(\mathbf{H}^{\mathrm{T}}\mathbf{H}) + \beta\mathrm{tr}(\mathbf{H}^{\mathrm{T}}\mathbf{Z}^{\mathrm{T}}\mathbf{Z}\mathbf{H}\mathbf{L}). \end{aligned} \tag{6.14}$$

Minimizing (6.14), the regression coefficients in the RFF space can be obtained as

$$\mathrm{vec}(\mathbf{H}_{\mathrm{opt}}) = (\mathbf{B}_{\mathrm{RFF}} + \mathbf{C}_{\mathrm{RFF}})^{-1}\mathrm{vec}(\mathbf{Z}^{\mathrm{T}}\mathbf{T}), \tag{6.15}$$

where

$$\mathbf{B}_{\mathrm{RFF}} = (\mathbf{I}_K \otimes (\mathbf{Z}^{\mathrm{T}}\mathbf{Z} + \alpha\mathbf{I}_D)), \tag{6.16}$$

$$\mathbf{C}_{\mathrm{RFF}} = (\beta\mathbf{L} \otimes \mathbf{Z}^{\mathrm{T}}\mathbf{Z}). \tag{6.17}$$

Once the regression coefficients are trained, the target estimate $\mathbf{y}$, given an input signal $\mathbf{x}$ corresponding to $\mathbf{z}$ in the RFF space, is given by

$$\mathbf{y} = \mathbf{H}_{\mathrm{opt}}^{\mathrm{T}} \mathbf{z}. \tag{6.18}$$

From (6.16) and (6.17), it can be observed that the computational burden of obtaining the regression parameters is drastically reduced when compared to the conventional KRG, as the size of the $\mathbf{B}_{\mathrm{RFF}}$ and $\mathbf{C}_{\mathrm{RFF}}$ is now $KD \times KD$, with D possibly much smaller than N. From (6.18), we see that the estimation does not depend on the number of training samples and the model has a fixed size D, requiring only the mapping of each new input sample into the RFF space.

6.2.2 Efficient Computation For Large Networks

For large networks, computing the inverses in (6.6) and (6.15) may be prohibitively complex. We propose an efficient way to compute the parameters in these cases. We adopt the notation of the conventional KRG, but the same reasoning applies directly to the RFF-based implementation. Considering the eigendecompositions $(\mathbf{I}_K + \beta\mathbf{L}) = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^{\mathrm{T}}$ and $\mathbf{K} = \mathbf{V}\mathbf{\Omega}\mathbf{V}^{\mathrm{T}}$, it is possible to show that

$$\mathbf{\Psi} = \mathbf{V}\mathbf{\Gamma}\mathbf{U}^{\mathrm{T}}, \tag{6.19}$$

where

$$\mathbf{\Gamma} = \mathrm{mat}\left((\alpha\mathbf{I}_{KN} + \mathbf{\Sigma}\otimes\mathbf{\Omega})^{-1}\mathrm{vec}(\mathbf{V}^{\mathrm{T}}\mathbf{T}\mathbf{U})\right). \tag{6.20}$$

Using this implementation, the dominating complexity is reduced from $(KN)^3$ operations due to matrix inversion to approximately K^3 and N^3 operations required for the eigendecompositions of $(\mathbf{I}_K + \beta\mathbf{L})$ and $\mathbf{K}$, respectively.

6.3 Online Kernel Regression on Graphs

In what follows, we consider online implementations of the KRG. To bypass the dimensionality problem associated with the kernel dictionary, we resort to online RFF-based KRG implementations.

6.3.1 Mini-batch Stochastic-Gradient KRG

Consider the following minimization problem:

$$\min_{\mathbf{H}} \mathrm{E}\left[\|\mathbf{t}-\mathbf{y}\|_2^2\right] + \alpha\mathrm{tr}(\mathbf{H}^{\mathrm{T}}\mathbf{H}) + \mathrm{E}[\beta\nu(\mathbf{y})]. \tag{6.21}$$

We propose the use of a mini-batch stochastic-gradient approach to solve (6.21). We define the matrices composed by the signals corresponding to each individual mini-batch, with N_{b} samples, as

$$\mathbf{Z}_n = \left[\mathbf{z}_{(n\delta-N_{\mathrm{b}}+1)} \ \dots \ \mathbf{z}_{n\delta}\right]^{\mathrm{T}} \in \mathbb{R}^{N_{\mathrm{b}}\times D}$$

and

$$\mathbf{T}_n = \left[\mathbf{t}_{(n\delta-N_{\mathrm{b}}+1)} \ \dots \ \mathbf{t}_{n\delta}\right]^{\mathrm{T}} \in \mathbb{R}^{N_{\mathrm{b}}\times K},$$

where $1 \leq \delta \leq N_{\mathrm{b}}$ is the batch displacement parameter. A particular case of the MGKRG is defined by making $N_{\mathrm{b}} = \delta = 1$. In this case, only the current sample is used to compute the approximation of the gradient. This corresponds to a stochastic-gradient approach and will be referred to as stochastic gradient KRG

Algorithm 6: MGKRG

```
Initialization:
$\mathbf{H}_1 = \mathbf{0}_{D\times K}$;
draw vectors $\{\mathbf{v}_i\}_{i=1}^D$ from $p(\mathbf{v})$;
draw phase terms $\{b_i\}_{i=1}^D$ from $[0, 2\pi]$;
%Learning
for each time instant $n$ do
    map $\mathbf{r}_n$ into $\mathbf{z}_n$;
    if $(n \bmod \delta) = 0$ then
        $\mathbf{Z}_n = [\mathbf{z}_{(n-N_\mathrm{b}+1)} \ \dots \ \mathbf{z}_n]^\mathrm{T}$;
        $\mathbf{T}_n = [\mathbf{t}_{(n-N_\mathrm{b}+1)} \ \dots \ \mathbf{t}_n]^\mathrm{T}$;
        $\mathbf{Y}_n = \mathbf{Z}_n\mathbf{H}_n$;
        $\mathbf{E}_n = \mathbf{T}_n - \mathbf{Y}_n$;
        $\mathbf{H}_{n+1} = (1-\mu\alpha)\mathbf{H}_n + \frac{\mu}{N_\mathrm{b}}\mathbf{Z}_n^\mathrm{T}(\mathbf{E}_n - \beta\mathbf{Y}_n\mathbf{L})$;
    end
    store $\mathbf{z}_n$;
    release $\mathbf{z}_{(n-N_\mathrm{b}+1)}$;
end
```

(SGKRG).

Letting $\mathbf{Y}_n = \mathbf{Z}_n\mathbf{H}_n$ be the mini-batch estimate and $\mathbf{E}_n = \mathbf{T}_n - \mathbf{Y}_n$ be the corresponding *a priori* error matrix, the update equation for the mini-batch gradient KRG is written as

$$\mathbf{H}_{n+1} = (1-\mu\alpha)\mathbf{H}_n + \frac{\mu}{N_\mathrm{b}}\mathbf{Z}_n^\mathrm{T}(\mathbf{E}_n - \beta\mathbf{Y}_n\mathbf{L}). \tag{6.22}$$

The steps for the MGKRG are summarized in Algorithm 6.

6.3.2 Recursive Least-Squares KRG

We now explore the principles of the recursive least squares algorithms [143] to solve (6.14) recursively. First, we rewrite (6.15) as

$$\begin{aligned}\mathrm{vec}(\mathbf{H}_n) &= \big((\mathbf{I}_K \otimes (\mathbf{Z}^\mathrm{T}\mathbf{Z} + \alpha\mathbf{I}_D)) + (\beta\mathbf{L}\otimes\mathbf{Z}^\mathrm{T}\mathbf{Z})\big)^{-1}\mathrm{vec}(\mathbf{Z}^\mathrm{T}\mathbf{T}) \\ &= \mathbf{R}_n^{-1}\mathbf{r}_n, \end{aligned} \tag{6.23}$$

where

$$\mathbf{R}_n = \alpha\mathbf{I}_K\otimes\mathbf{I}_D + (\mathbf{I}_K + \beta\mathbf{L})\otimes\mathbf{Z}^\mathrm{T}\mathbf{Z} \tag{6.24}$$

$$\mathbf{r}_n = \mathrm{vec}(\mathbf{Z}^\mathrm{T}\mathbf{T}). \tag{6.25}$$

Algorithm 7: RFF-based RLSKRG

```
Initialization:
$\mathbf{R}_{-1}^{-1} = \frac{1}{\alpha}\mathbf{I}_{KD}$;
$\mathbf{H}_{-1} = \mathbf{0}_{D\times K}$;
draw vectors $\{\mathbf{v}_i\}_{i=1}^{D}$ from $p(\mathbf{v})$;
draw phase terms $\{b_i\}_{i=1}^{D}$ from $[0, 2\pi]$;
%Learning
for each time instant $n$ do
    map $\mathbf{r}_n$ into $\mathbf{z}_n$;
    $\mathbf{P}_n = \mathbf{I}_K \otimes \mathbf{z}_n$;
    $\mathbf{Q}_n = (\mathbf{I}_K + \beta\mathbf{L}) \otimes \mathbf{z}_n^{\mathrm{T}}$;
    $\mathbf{G}_n = \mathbf{R}_{n-1}^{-1}\mathbf{P}_n\left(\mathbf{I}_K + \mathbf{Q}_n\mathbf{R}_{n-1}^{-1}\mathbf{P}_n\right)^{-1}$;
    $\hat{\mathbf{y}}_n = \mathbf{H}_{n-1}^{\mathrm{T}}\mathbf{z}_n$;
    $\mathbf{e}_n = \mathbf{t}_n - \hat{\mathbf{y}}_n$;
    $\mathbf{H}_n = \mathbf{H}_{n-1} + \mathrm{mat}(\mathbf{G}_n(\mathbf{e}_n - \beta\mathbf{L}\hat{\mathbf{y}}_n))$;
    $\mathbf{R}_n^{-1} = \mathbf{R}_{n-1}^{-1} - \mathbf{G}_n\mathbf{Q}_n\mathbf{R}_{n-1}^{-1}$;
end
```

Letting

$$\mathbf{G}_n = \left(\mathbf{R}_{n-1}^{-1} - \mathbf{G}_n\mathbf{Q}_n\mathbf{R}_{n-1}^{-1}\right)\mathbf{P}_n = \mathbf{R}_n^{-1}\mathbf{P}_n, \tag{6.26}$$

with

$$\begin{aligned}\mathbf{P}_n &= (\mathbf{I}_K \otimes \mathbf{z}_n)\\ \mathbf{Q}_n &= ((\mathbf{I}_K + \beta\mathbf{L}) \otimes \mathbf{z}_n^{\mathrm{T}}),\end{aligned} \tag{6.27}$$

it is possible to derive the update equation

$$\mathbf{H}_n = \mathbf{H}_{n-1} + \mathrm{mat}(\mathbf{G}_n(\mathbf{e}_n - \beta\mathbf{L}\hat{\mathbf{y}}_n)), \tag{6.28}$$

where mat$(\cdot)$ denotes the vector-to-matrix operator, $\hat{\mathbf{y}}_n = \mathbf{H}_{n-1}^{\mathrm{T}}\mathbf{z}_n$ is the *a priori* target estimate, and $\mathbf{e}_n = \mathbf{t}_n - \hat{\mathbf{y}}_n$ is the *a priori* error. Equation (6.28) is the recursive update equation for the proposed recursive least squares KRG (RLSKRG) algorithm. The steps for the implementation of the RLSKRG algorithm are summarized in Algorithm 7.

Due to its recursive nature, the RLSKRG algorithm considers past samples when computing the update matrix at each iteration. Thus, its performance is expected to match that of the batch-based approach.

Similar to the batch-based implementation, as presented in Section 6.2.2, the RLSKRG admits an efficient implementation. Using the eigendecompositions $(\mathbf{I}_K + \beta\mathbf{L}) = \mathbf{U}\boldsymbol{\Sigma}\mathbf{U}^{\mathrm{T}}$ and $\mathbf{R}_{z,n} = \mathbf{V}_n\boldsymbol{\Omega}_n\mathbf{V}_n^{\mathrm{T}}$, the update equation for the efficient RLSKRG

Algorithm 8: Efficient RLSKRG

```
Initialization:
$\mathbf{R}_{z,-1} = \mathbf{0}_{D\times D}$;
$\mathbf{H}_{-1} = \mathbf{0}_{D\times K}$;
get $\mathbf{U}$ and $\mathbf{\Sigma}$;
draw vectors $\{\mathbf{v}_i\}_{i=1}^{D}$ from $p(\mathbf{v})$;
draw phase terms $\{b_i\}_{i=1}^{D}$ from $[0, 2\pi]$;
%Learning
for each time instant $n$ do
    map $\mathbf{r}_n$ into $\mathbf{z}_n$;
    $\mathbf{R}_{z,n} = \mathbf{R}_{z,n-1} + \mathbf{z}_n\mathbf{z}_n^{\mathrm{T}}$;
    Get $\mathbf{V}_n$ and $\mathbf{\Omega}_n$;
    $\mathbf{P}_n = \mathbf{I}_K \otimes \mathbf{z}_n$;
    $\hat{\mathbf{y}}_n = \mathbf{H}_{n-1}^{\mathrm{T}}\mathbf{z}_n$;
    $\mathbf{e}_n = \mathbf{t}_n - \hat{\mathbf{y}}_n$;
    $\mathbf{\Xi} = \mathrm{mat}(\mathbf{P}_n(\mathbf{e}_n - \beta\mathbf{L}\hat{\mathbf{y}}_n))$;
    $\mathbf{\Gamma}_n = \mathrm{mat}((\alpha\mathbf{I}_{KD} + \mathbf{\Sigma}\otimes\mathbf{\Omega}_n)^{-1}\mathrm{vec}(\mathbf{V}_n^{\mathrm{T}}\mathbf{\Xi}_n\mathbf{U}))$;
    $\mathbf{H}_n = \mathbf{H}_{n-1} + \mathbf{V}_n\mathbf{\Gamma}_n\mathbf{U}^{\mathrm{T}}$;
end
```

is given by

$$\mathbf{H}_n = \mathbf{H}_{n-1} + \mathbf{V}_n\mathbf{\Gamma}_n\mathbf{U}^{\mathrm{T}}, \tag{6.29}$$

where $\mathbf{\Gamma}_n = \mathrm{mat}((\alpha\mathbf{I}_{KD}+\mathbf{\Sigma}\otimes\mathbf{\Omega}_n)^{-1}\mathrm{vec}(\mathbf{V}_n^{\mathrm{T}}\mathbf{\Xi}_n\mathbf{U}))$. The steps for the implementation of the efficient RLSKRG are presented in Algorithm 8.

6.4 Convergence Analysis

It is possible to show that the proposed online algorithms converge both in the mean and in mean squared sense. In what follows, $\mathbf{H}_{\mathrm{o}}$ denotes the optimal linear estimator in the least mean squares sense of $\mathbf{T}_n$ in the RFF domain. Consider that the RFF-mapped data signal $\mathbf{z}_n$ is drawn from a wide-sense stationary multivariate random sequence with correlation matrix $\mathbf{R}_z = \mathrm{E}[\mathbf{z}_n\mathbf{z}_n^{\mathrm{T}}]$. Let $\lambda_{\max}(\cdot)$ denote the maximum eigenvalue of the argument matrix and let $\rho(\cdot)$ denote the spectral radius of the argument matrix, i.e., the largest absolute value of its eigenvalues. For the MGKRG, the following conditions are sufficient for first- and second-order stability, respectively:

Theorem 6.1. A sufficient condition on the step size μ for the convergence of the proposed MGKRG algorithm governed by (6.22), is given by

$$0 < \mu < \frac{2}{\lambda_{\max}(\mathbf{R}_z) + \alpha + \beta\lambda_{\max}(\mathbf{L})\lambda_{\max}(\mathbf{R}_z)}, \tag{6.30}$$

and

Theorem 6.2. Then, the second-order convergence of the proposed gradient-based algorithms, namely the MGKRG and the SGKRG, is guaranteed under

$$0 < \mu < \frac{1}{\lambda_{\max}(\mathbf{R}_z) + \alpha + \beta\lambda_{\max}(\mathbf{L})\lambda_{\max}(\mathbf{R}_z)}. \tag{6.31}$$

Remark 6.2. Under the convergence condition (6.30), the MGKRG converges asymptotically to $(\alpha\mathbf{I}_{KD} + (\mathbf{I}_K + \beta\mathbf{L}) \otimes \mathbf{R}_z)^{-1}(\alpha\mathbf{I}_{KD} + \beta\mathbf{L} \otimes \mathbf{R}_z)\mathbf{h}_\text{o}$. This means that $\lim_{n\to\infty} \mathbf{H}_n$ is a biased estimate of $\mathbf{H}_\text{o}$. Also, the bias is introduced by the regularization coefficients α and β, such that a non-regularized problem leads to an unbiased solution.

Theorem 6.2 shows that the condition for second-order stability of the MGKRG is more strict than that of the first-order stability. The upper-bound imposed on the step-sizes for second-order stability is half of the upper-bound established in Theorem 6.1.

Under a reasonable set of assumptions, the following theorems hold for the RLSKRG:

Theorem 6.3. The RLSKRG described in Algorithm 7 is stable in the mean sense and converges to a steady state.

In fact, it is possible to show that $\mathbf{H}_n$ is an asymptotically biased estimate of $\mathbf{H}_\text{o}$, with $\lim_{n\to\infty} \text{E}[\widetilde{\mathbf{H}}_n] = \beta\mathbf{H}_\text{o}\mathbf{L}(\mathbf{I}_K + \beta\mathbf{L})^{-1}$.

Remark 6.3. Under the convergence condition (6.30), the bias of the MGRKG tends to the bias of the RLSKRG when $\alpha \to 0^+$. In addition, the bias in the RLSKRG is introduced solely by the regularization coefficient β, since the regularization coefficient α contributes only with an initial condition for the matrix $\mathbf{R}_n$, which plays no role in the algorithm's average behavior as n grows to infinity.

Theorem 6.4. The RLSKRG described in Algorithm 7 is stable in the mean-squared sense and converges to a steady state.

In this case, the RLSKRG converges in the mean-squared sense to

$$\lim_{n\to\infty} \text{E}[\|\tilde{\mathbf{h}}_n\|_2^2] = \|\text{vec}\left(\beta\mathbf{H}_\text{o}\mathbf{L}(\mathbf{I}_K + \beta\mathbf{L})^{-1}\right)\|_2^2.$$

6.5 Discussion on Complexity

For the MGKRG algorithm, the update (6.22) requires $DK + N_\text{b}(K^2 + 2DK + K)$ multiplication operations. That is, the complexity of the MGKRG increases linearly with N_b with a slope equal to $K^2 + 2DK + K$. Additionally, the MGKRG requires a memory to store $N_\mathrm{b} > 1$ samples. Hence, the batch-size translates into a trade-off between complexity and performance.

The proposed efficient implementation of the RLSKRG in (6.29) requires $D^3 + D^2 + 2D^2K + 5DK + 2DK^2 + K^2$ multiplication operations to update $\mathbf{H}_n$. Assuming that N_b has the same order of magnitude as D and K, the RLSKRG has a slightly heavier computational burden per iteration when compared to the MGKRG. The offline batch KRG using RFF (6.19) requires $D^3 + D^2N + 2D^2K + 3DK + 2DK^2 + KDN$ multiplications. This complexity is considerably smaller than that of the conventional implementation (6.15), which requires the inversion of a $DK \times DK$ matrix, leading to complexity equivalent to D^3K^3 multiplications for the inversion operation only.

6.6 Numerical Results

In this section, we validate the performance of the proposed algorithms with numerical experiments using both synthesized and real datasets.

6.6.1 Synthesized Data 1

Similar to the setup in [31], we consider an Erdös Rényi graph with $K = 50$ nodes and edge-probability equal to 0.1. A total of $S = 20000$ K-dimensional i.i.d. samples, $\{\mathbf{x}_n\}_{n=1}^S$, are generated, where $\mathbf{x}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{C}_S)$. The S-dimensional covariance matrix $\mathbf{C}_S \in \mathbb{R}^{S\times S}$ is drawn from the inverse Wishart distribution with an identity scale matrix. We generate the target graph signals $\{\mathbf{t}_n\}_{n=1}^S$ as in [31], i.e., by solving $\mathbf{t}_n = \arg\min_{\boldsymbol{\tau}} \left\{\|\mathbf{x}_n - \boldsymbol{\tau}\|_2^2 + \boldsymbol{\tau}^\mathrm{T}\mathbf{L}\boldsymbol{\tau}\right\}$. We assess the normalized mean squared error

$$\mathrm{NMSE} = 10\log_{10}\left(\mathrm{E}\left[\frac{\|\mathbf{Y} - \mathbf{T}_0\|_\mathrm{F}^2}{\|\mathbf{T}_0\|_\mathrm{F}^2}\right]\right), \tag{6.32}$$

where $\mathbf{T}$ denotes the true target matrix and $\mathbf{Y}$ denotes the estimated matrix. The expected value is obtained as the ensemble average over 500 independent runs.

Fig. 6.1 presents the results of the batch-based implementations and the RLSKRG. We see that the RFF implementation approximates well the conventional KRG even for relatively small $D = 32$. The performance of the RLSKRG closely matches the performance of the batch-based implementation. Results in Fig. 6.2 show that online algorithms can effectively learn the regression parameters and we analyze the performance of the MGKRG for different mini-batch sizes. Plots show an increase in convergence speed as N_b increases to 15 and then to 50 samples, demonstrating the improved accuracy when more samples are used in the stochastic-gradient approximation.

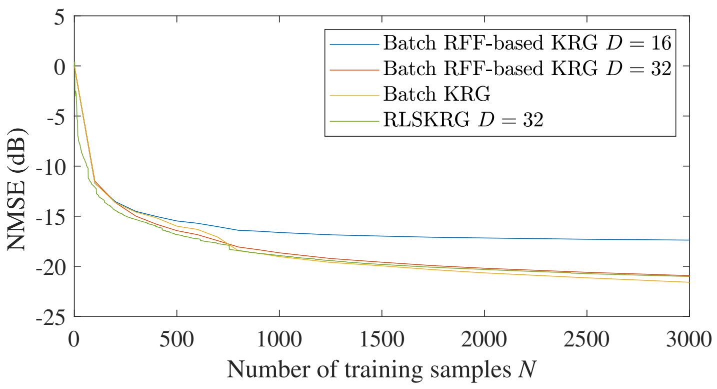


Figure 6.1: NMSE achieved by the Bacht-based and RLSKRG implementations versus number of training samples using synthesized data.

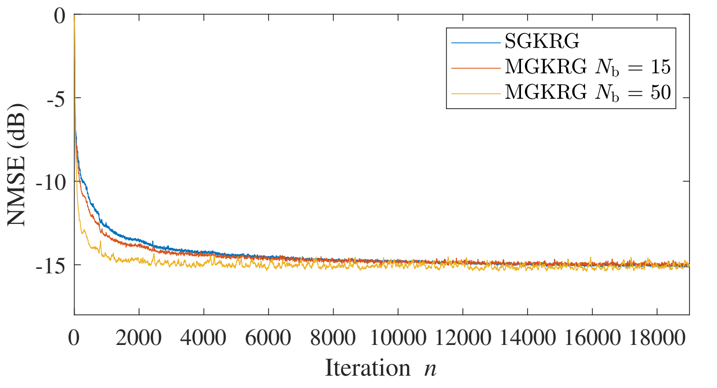


Figure 6.2: NMSE achieved by the MGKRG implementations versus number of training samples for different mini-batch sizes.

6.6.2 Real Data - fMRI Signal Extrapolation

This section reproduces the example from [31], which employs the conventional KRG to estimate the intensities of voxels in a functional magnetic resonance imaging (fMRI) dataset. The data and graph used are available in [144]. The experiment consists of estimating the fMRI signal on 90 of the voxels – volumetric units that represent regions of the brain – using the signal from ten other voxels. In other words, we consider an input signal $\mathbf{x} \in \mathbb{R}^{10}$ to estimate a graph signal $\mathbf{t} \in \mathbb{R}^{90}$. The graph corresponds to the pairwise relations of the 90 voxels.

Results in Fig. 6.3 show that the RFF-based KRG closely matches the conventional KRG, converging to approximately -23 dB, with $D = 32$. The RLSKRG matches the batch-based implementations. Results also show that the SG-based implementations can achieve low NMSE, around -20 dB. Again, increasing the number of samples when computing the stochastic approximation for the gradient increases the convergence speed while matching the same accuracy after convergence.

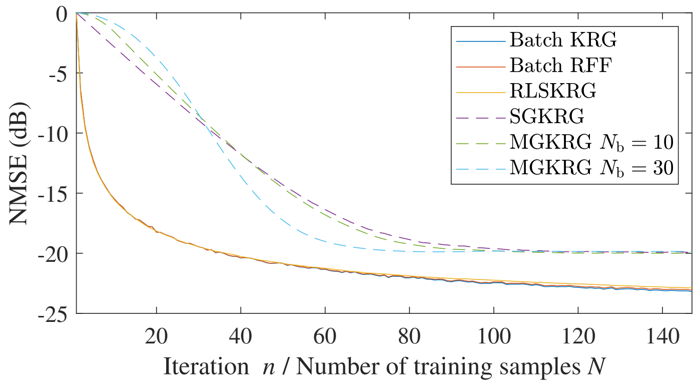


Figure 6.3: NMSE achieved by the KRG implementations versus number of training samples for the fMRI signal simulation.

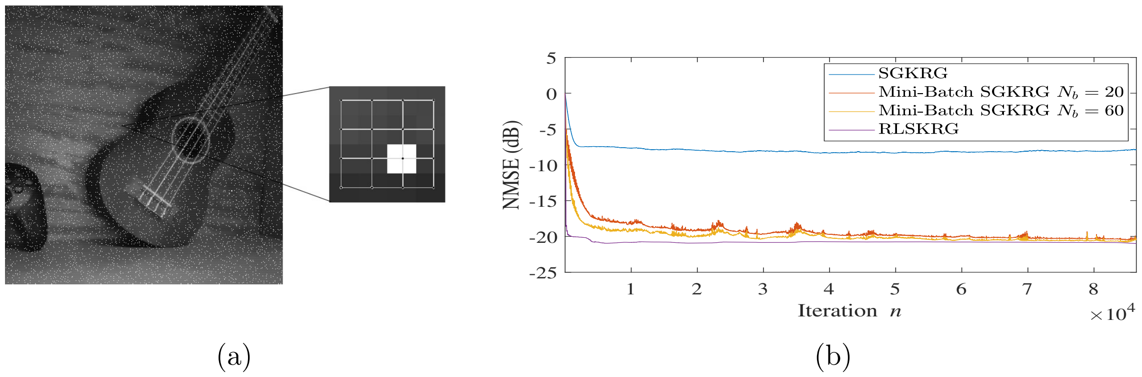


Figure 6.4: Image reconstruction process using KRG: (a) example frame and how a 4×4 block of pixels is treated as a graph; (b) NMSE achieved by the KRG implementations versus number of training samples in the image reconstruction simulation.

6.6.3 Real Data - Image Reconstruction

We now consider the application of KRG in the image and video processing scenario. This simulation showcases the performance and the capability of the online algorithms to deal with large datasets. In particular, we tackle the reconstruction of a corrupted video frame. In this setup, corrupted frames have up to one random pixel per 4×4-pixel block that is set to unity, simulating a saturated pixel. An example of a corrupted frame, along with a block of pixels with one corrupted pixel, and the corresponding graph, using the NN image model, is illustrated in Fig. 6.4a.

Fig. 6.4b shows the NMSE versus iterations for the proposed algorithms. These results are consistent with previous simulations and show that online KRG strategies can successfully learn the target model. We observe that the RLSKRG exhibits the best performance while the single-sample SGKRG exhibits the worst performance, as expected, given the complexity-performance trade-off. In this simulation, increasing the number of blocks to $N_b = 20$ and $N_b = 60$ (which corresponds to half the number

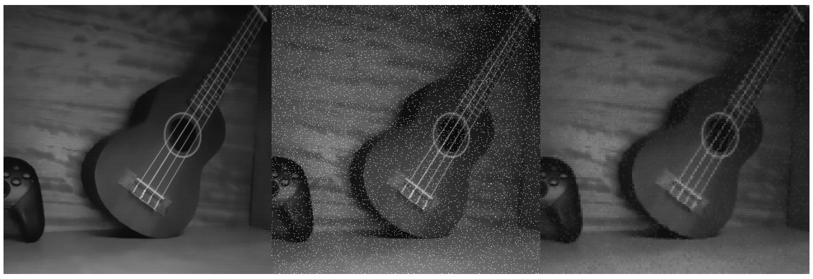


Figure 6.5: Original frame, corrupted frame, and reconstructed frame, from left to right.

of blocks on a single line in an image) considerably increases the performance of the MGKRG. Fig. 6.5 the frame reconstruction using the RLSKRG, which showcases the capabilities of the proposed algorithm.

6.7 Summary

In this chapter, we addressed learning over graphs, considering the case where the input signal is not defined over the graph. We proposed efficient batch-based KRG implementations using RFF. We also proposed online strategies based on stochastic-gradient approaches, the MGKRG and the SGKRG, and the recursion-based RLSKRG. We presented convergence conditions and a brief discussion on the trade-off between complexity and performance of the proposed algorithms. The proposed methodology was employed for a range of numerical experiments using synthesized and real data simulations, including experiments on brain-data extrapolation and image reconstruction. Results confirmed that both the proposed KRG using RFF and the RLSKRG have accuracy close to that of the conventional KRG, with a considerable reduction in complexity. Additionally, simulations showed that the MGKRG can effectively learn the regression parameters and that its performance can be improved at a small increase in computational cost by increasing the number of samples in the mini-batch.

Chapter 7

Conclusions and Future Work

This thesis provided contributions to two key research areas of GSP and investigated a range of different applications for the proposed tools. As the first research topic, we investigated the challenges of translating real-world problems into GSP models. This research topic branches into two contributions: we developed a GSP-based model for blocks of pixels that is suitable for compression of light field images, and we proposed an augmentation methodology for sparse adjacency matrices, which yields a scale-dependent GFT, called sGFT. The second research topic addresses the learning of relations between reference and target signals over graphs. We proposed two methodologies that tackle different scenarios, the first exploring concepts of graph filters and the second based on kernel regression.

When investigating the application of GSP for light-field compression, we identified the challenges of using graphs to represent multiple different networks, with each network being associated with a block of pixels. As GSP explores individual characteristics of each block, the optimal representation of all blocks in an image requires a different adjacency matrix for each block, increasing computational burden. To overcome this issue, we employed a sparse model for the adjacency matrices, reducing the complexity associated with each matrix. We leveraged the redundancy in light fields to use a single representation for multiple blocks. We developed a simplified compression scheme similar to HEVC-based methods for light-field compression, and numerical experiments showed that, using the GFT, we need a reduced number of transform coefficients to achieve the same distortion when compared to the DCT. As future directions for this research, we note the GSP-based approach for compressing light-fields needs more in-depth investigation of other methods for creating the graph model and that more complex schemes for compression can be used to develop simulations in line with practical state-of-the-art implementations. Additionally, the proposed methodology can be adapted to specific settings of light field, such as single-camera light-field imaging, to be compared against recent works that employ GSP for light-field compression.

On the scope of developing GSP-based models for real-world problems, we also proposed the extended adjacency matrix using diffusion distances and the sGFT. The additional spectral information provided by the sGFT can be explored by GSP tools based on the graph spectrum, which was demonstrated by applying the proposed methodology for anomaly detection on networked data. Results showed that the augmented model offers a significant improvement over the conventional adjacency, increasing detection scores. In this line of research, future works include the study of how other GSP tools interact with the proposed augmented adjacency matrix and the investigation of other applications that can benefit from the proposed model.

In Chapter 5, we first presented the basic concepts of learning over graphs and a state-of-the-art methodology using graph-diffusion LMS algorithms. Building upon this methodology, we first proposed the concept of nonlinear graph filters. The proposed nonlinear graph filtering consists of a nonlinear function applied to graph-shifted signals, generalizing the well-known LSI graph filters. The adaptive centralized GKLMS algorithm was derived to estimate the parameters of nonlinear graph filters, with implementations based on CC and RFF to avoid the complexity issues inherent to kernel methods. Additionally, using concepts of network diffusion, the fully decentralized GDKLMS using RFF was proposed, where each node uses only local computations and communications to estimate the filter's parameters. Mean and mean-square convergence conditions, along with complexity analysis, were presented for the proposed algorithms. Numerical simulations using both synthesized and real data showed that the proposed methods could effectively estimate nonlinear graph filters. Also, results showed that RFF-based implementations yield faster convergence than CC-based implementations, besides being robust to model changes. Future directions for this research include developing RLS-based distributed approaches for learning nonlinear graph filters and methodologies for learning the graph structure together with the parameters, i.e., including the GSO as a variable in the optimization problem.

Complementary to the methodology proposed in Chapter 5, the work presented in Chapter 6 uses kernel regression methods to learn over graphs while considering that the input signal is not necessarily a graph signal. Similar to the approach presented in Chapter 5, we adopted RFF to overcome the complexity issues of kernel methods and derived the batch-based KRG algorithm using RFF. Furthermore, we proposed distinct strategies for online KRG. First, we proposed the gradient-descent-based algorithms MGKRG and, as a particular case, the SGKRG. Using concepts of conventional RLS algorithms, we proposed the RLSKRG. We showed that the RLSKRG achieves the same accuracy as the batch-based KRG and that the SG-based algorithms can effectively estimate the regression parameters at reduced complexity. Moreover, using the MGKRG, the performance can be improved at a relatively small increase in computational cost by increasing the number of samples in the mini-batch. We also provided first- and second-order stability conditions for the proposed online algorithms. The performance of the proposed algorithms was validated over a range of numerical simulations using both synthesized and real data. Real data experiments included temperature prediction, brain-activity estimation, and image reconstruction. As future work, we consider the derivation of the RLSKRG using a forgetting factor, such that older samples have exponentially less weight, and the implementation of other complexity-reduction techniques, especially for the RLSKRG. For instance, dichotomous-coordinate descent (DCD) iterations have been used in the literature for reduced-complexity RLS algorithms and can be adapted to the KRG methodology.

Other future directions include the combination of GSP with other machine-learning approaches, such as unsupervised learning, reinforcement learning, and deep learning, in contrast with the supervised algorithms proposed in this thesis. For instance, similarly to the KRG approach, graph smoothness can be used as a regularization factor in the reward function of reinforcement learning approaches. Also, machine-learning algorithms can leverage GSP-based data attributes, such as graph-spectral coefficients, as features when knowledge regarding the relations between input elements is available. Another open direction is the construction of combinations of graph-filters, exploring concepts of combinations of filters from DSP and leveraging the degree of freedom given by the graph structure in the filter construction.

References

[1] A. Sandryhaila and J. M. Moura, “Discrete Signal Processing on Graphs,” *IEEE Transactions on Signal Processing*, vol. 61, no. 7, pp. 1644–1656, 2013.

[2] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, “The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains,” *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 83–98, 2013.

[3] G. Ribeiro and J. Lima, “Graph signal processing in a nutshell,” *Journal of Communication and Information Systems*, vol. 33, no. 1, pp. 219–233, 2018.

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

[5] 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*, vol. 31, no. 5, pp. 80–90, 2014.

[6] I. Jablonski, “Graph signal processing in applications to sensor networks, smart grids, and smart cities,” *IEEE Sensors Journal*, vol. 17, pp. 7659–7666, Dec. 2017.

[7] A. Gavili and X. Zhang, “On the shift operator, graph frequency, and optimal filtering in graph signal processing,” *IEEE Transactions on Signal Processing*, vol. 65, pp. 6303–6318, Dec. 2017.

[8] B. Boucher and S. Jenna, “Genetic interaction networks: Better understand to better predict,” *Frontiers in Genetics*, vol. 4, no. Dec., pp. 1–16, 2013.

[9] W. Huang, T. A. W. Bolton, J. D. Medaglia, D. S. Bassett, A. Ribeiro, and D. Van De Ville, “A graph signal processing perspective on functional brain imaging,” *Proceedings of the IEEE*, vol. 106, pp. 868–885, May 2018.

[10] A. Sandryhaila and J. M. Moura, "Discrete Signal Processing on Graphs: Frequency Analysis," *IEEE Transactions on Signal Processing*, vol. 62, pp. 3042–3054, June 2014.

[11] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs: Graph filters," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 6163–6166, 2013.

[12] O. Teke and P. P. Vaidyanathan, "Fundamentals of multirate graph signal processing," in *Asilomar Conference on Signals, Systems and Computers*, pp. 1791–1795, 2016.

[13] O. Teke and P. P. Vaidyanathan, "Extending Classical Multirate Signal Processing Theory to Graphs - Part I: Fundamentals," *IEEE Transactions on Signal Processing*, vol. 65, no. c, pp. 1–1, 2016.

[14] O. Teke and P. P. Vaidyanathan, "Extending Classical Multirate Signal Processing Theory to Graphs—Part II: M-Channel Filter Banks," *IEEE Transactions on Signal Processing*, vol. 65, pp. 423–437, Jan. 2017.

[15] B. Girault, "Stationary graph signals using an isometric graph translation," in *European Signal Processing Conference (EUSIPCO)*, vol. 5668, pp. 1516–1520, Aug. 2015.

[16] B. Girault, P. Gon, E. Fleury, B. Girault, and P. Gon, "Translation and Stationarity for Graph Signals," *Research Report*, 2015.

[17] T. Summers, I. Shames, J. Lygeros, and F. Dörfler, "Topology design for optimal network coherence," in *2015 European Control Conference (ECC)*, pp. 575–580, 2015.

[18] S. Shahrampour and V. M. Preciado, "Topology identification of directed dynamical networks via power spectral analysis," *IEEE Transactions on Automatic Control*, vol. 60, no. 8, pp. 2260–2265, 2015.

[19] M. Babakmehr, M. G. Simões, M. B. Wakin, and F. Harirchi, "Compressive sensing-based topology identification for smart grids," *IEEE Transactions on Industrial Informatics*, vol. 12, no. 2, pp. 532–543, 2016.

[20] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst, "Learning laplacian matrix in smooth graph signal representations," *IEEE Transactions on Signal Processing*, vol. 64, pp. 6160–6173, Dec. 2016.

[21] D. Thanou, X. Dong, D. Kressner, and P. Frossard, "Learning heat diffusion graphs," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 3, pp. 484–499, Sept. 2017.

[22] G. B. Giannakis, Y. Shen, and G. V. Karanikolas, "Topology identification and learning over graphs: Accounting for nonlinearities and dynamics," *Proceedings of the IEEE*, vol. 106, pp. 787–807, May 2018.

[23] G. Mateos, S. Segarra, A. G. Marques, and A. Ribeiro, "Connecting the dots: Identifying network structure via graph signal processing," *IEEE Signal Processing Magazine*, vol. 36, pp. 16–43, May 2019.

[24] M. J. M. Spelta and W. A. Martins, "Normalized LMS algorithm and data-selective strategies for adaptive graph signal estimation," *Signal Processing*, vol. 167, p. 107326, Feb. 2020.

[25] F. Hua, R. Nassif, C. Richard, H. Wang, and A. H. Sayed, "Diffusion LMS with communication delays: Stability and performance analysis," *IEEE Signal Processing Letters*, vol. 27, pp. 730–734, 2020.

[26] P. D. Lorenzo, P. Banelli, S. Barbarossa, and S. Sardellitti, "Distributed adaptive learning of graph signals," *IEEE Transactions on Signal Processing*, vol. 65, pp. 4193–4208, Aug. 2017.

[27] R. Nassif, C. Richard, J. Chen, and A. H. Sayed, "A graph diffusion LMS strategy for adaptive graph signal processing," in *Asilomar Conference on Signals, Systems, and Computers*, pp. 1973–1976, 2017.

[28] R. Nassif, C. Richard, J. Chen, and A. H. Sayed, "Distributed diffusion adaptation over graph signals," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4129–4133, 2018.

[29] F. Hua, R. Nassif, C. Richard, H. Wang, and A. H. Sayed, "Online distributed learning over graphs with multitask graph-filter models," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 6, pp. 63–77, Jan. 2020.

[30] R. Nassif, S. Vlaski, C. Richard, J. Chen, and A. H. Sayed, "Multitask learning over graphs: An approach for distributed, streaming machine learning," *IEEE Signal Processing Magazine*, vol. 37, pp. 14–25, May 2020.

[31] A. Venkitaraman, S. Chatterjee, and P. Händel, "Predicting graph signals using kernel regression where the input signal is agnostic to a graph," *IEEE*

Transactions on Signal and Information Processing over Networks, vol. 5, pp. 698–710, Dec. 2019.

[32] P. Bouboulis, S. Chouvardas, and S. Theodoridis, "Online distributed learning over networks in RKH spaces using random Fourier features," *IEEE Transactions on Signal Processing*, vol. 66, pp. 1920–1932, Apr. 2018.

[33] Y. Shen, G. Leus, and G. B. Giannakis, "Online graph-adaptive learning with scalability and privacy," *IEEE Transactions on Signal Processing*, vol. 67, pp. 2471–2483, May 2019.

[34] J. Mei and J. M. F. Moura, "Signal processing on graphs: Causal modeling of unstructured data," *IEEE Transactions on Signal Processing*, vol. 65, pp. 2077–2092, Apr. 2017.

[35] E. Isufi, A. Loukas, A. Simonetto, and G. Leus, "Autoregressive Moving Average Graph Filtering," *IEEE Transactions on Signal Processing*, vol. 65, pp. 274–288, Jan. 2017.

[36] F. Hua, C. Richard, J. Chen, H. Wang, P. Borgnat, and P. Gonçalves, "Learning combination of graph filters for graph signal modeling," *IEEE Signal Processing Letters*, vol. 26, pp. 1912–1916, Dec. 2019.

[37] S. K. Narang and A. Ortega, "Perfect Reconstruction Two-Channel Wavelet Filter Banks for Graph Structured Data," *IEEE Transactions on Signal Processing*, vol. 60, pp. 2786–2799, June 2012.

[38] S. Chen, R. Varma, A. Sandryhaila, and J. Kovačević, "Discrete signal processing on graphs: Sampling theory," *IEEE Transactions on Signal Processing*, vol. 63, pp. 6510–6523, Dec. 2015.

[39] A. Loukas, A. Simonetto, and G. Leus, "Distributed autoregressive moving average graph filters," *IEEE Signal Processing Letters*, vol. 22, pp. 1931–1935, Nov. 2015.

[40] E. Isufi, A. Loukas, N. Perraudin, and G. Leus, "Forecasting time series with varma recursions on graphs," *IEEE Transactions on Signal Processing*, vol. 67, pp. 4870–4885, Sept. 2019.

[41] A. H. Sayed, "Adaptation, learning, and optimization over networks," *Foundations and Trends® in Machine Learning*, vol. 7, no. 4-5, pp. 311–801, 2014.

[42] A. H. Sayed, "Diffusion adaptation over networks," in *Academic Press Library in Signal Processing*, pp. 323–453, Elsevier, 2014.

[43] M. Paluš, "Detecting nonlinearity in multivariate time series," *Physics Letters A*, vol. 213, pp. 138–147, 1996.

[44] J. Walker and N. Jenkins, *Wind Energy Technology.* Wiley, 1997.

[45] M. O. Franz and B. Schölkopf, "A unifying view of wiener and volterra theory and polynomial kernel regression," *Neural Computation*, vol. 18, pp. 3097–3118, Dec. 2006.

[46] J. P. Danilo Comminiello, *Adaptive learning methods for nonlinear system modeling.* Elsevier, 2018.

[47] V. J. Mathews, "Adaptive polynomial filters," *IEEE Signal Processing Magazine*, vol. 8, pp. 10–26, July 1991.

[48] T. Koh and E. Powers, "Second-order Volterra filtering and its application to nonlinear system identification," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 33, pp. 1445–1455, Dec. 1985.

[49] D. J. Sebald and J. A. Bucklew, "Support vector machine techniques for nonlinear equalization," *IEEE Transactions on Signal Processing*, vol. 48, pp. 3217–3226, Nov. 2000.

[50] M. Scarpiniti, D. Comminiello, G. Scarano, R. Parisi, and A. Uncini, "Steady-state performance of spline adaptive filters," *IEEE Transactions on Signal Processing*, vol. 64, pp. 816–828, Feb. 2016.

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

[52] W. Liu, J. C. Príncipe, and S. Haykin, *Kernel adaptive filtering.* Wiley, 2010.

[53] W. Gao, J. Chen, C. Richard, and J. Huang, "Online dictionary learning for kernel LMS," *IEEE Transactions on Signal Processing*, vol. 62, pp. 2765–2777, June 2014.

[54] S. Theodoridis, K. Slavakis, and I. Yamada, "Adaptive learning in a world of projections," *IEEE Signal Processing Magazine*, vol. 28, pp. 97–123, Jan. 2011.

[55] W. Gao, J. Chen, C. Richard, and J. Huang, "Diffusion adaptation over networks with kernel least-mean-square," in *IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing*, pp. 217–220, 2015.

[56] S. Chouvardas and M. Draief, “A diffusion kernel LMS algorithm for nonlinear adaptive networks,” in *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 4164–4168, 2016.

[57] D. Romero, M. Ma, and G. B. Giannakis, “Kernel-based reconstruction of graph signals,” *IEEE Transactions on Signal Processing*, vol. 65, pp. 764–778, Feb. 2017.

[58] D. Romero, V. N. Ioannidis, and G. B. Giannakis, “Kernel-based reconstruction of space-time functions on dynamic graphs,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, pp. 856–869, Sept. 2017.

[59] V. N. Ioannidis, D. Romero, and G. B. Giannakis, “Inference of spatio-temporal functions over graphs via multikernel kriged kalman filtering,” *IEEE Transactions on Signal Processing*, vol. 66, pp. 3228–3239, June 2018.

[60] B. Shin, M. Yukawa, R. L. G. Cavalcante, and A. Dekorsy, “Distributed adaptive learning with multiple kernels in diffusion networks,” *IEEE Transactions on Signal Processing*, vol. 66, pp. 5505–5519, Nov. 2018.

[61] S. Wang, L. Dang, B. Chen, S. Duan, L. Wang, and C. K. Tse, “Random Fourier filters under maximum correntropy criterion,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 10, pp. 3390–3403, 2018.

[62] K. Chen, S. Werner, A. Kuh, and Y. Huang, “Nonlinear adaptive filtering with kernel set-membership approach,” *IEEE Transactions on Signal Processing*, vol. 68, pp. 1515–1528, 2020.

[63] C. Richard, J. C. M. Bermudez, and P. Honeine, “Online prediction of time series data with kernels,” *IEEE Transactions on Signal Processing*, vol. 57, pp. 1058–1067, Mar. 2009.

[64] A. Singh, N. Ahuja, and P. Moulin, “Online learning with kernels: Overcoming the growing sum problem,” in *IEEE International Workshop on Machine Learning for Signal Processing*, pp. 1–6, 2012.

[65] P. Bouboulis, S. Pougkakiotis, and S. Theodoridis, “Efficient KLMS and KRLS algorithms: A random Fourier feature perspective,” in *2016 IEEE Statistical Signal Processing Workshop (SSP)*, pp. 1–5, 2016.

[66] N. Aronszajn, “Theory of reproducing kernels,” *Transactions of the American Mathematical Society*, vol. 68, no. 3, pp. 337–404, 1950.

[67] P. P. Pokharel, W. Liu, and J. C. Principe, “Kernel least mean square algorithm with constrained growth,” *Signal Processing*, vol. 89, pp. 257 – 265, Mar. 2009.

[68] W. D. Parreira, J. C. M. Bermudez, C. Richard, and J. Tourneret, “Stochastic behavior analysis of the Gaussian kernel least-mean-square algorithm,” *IEEE Transactions on Signal Processing*, vol. 60, pp. 2208–2222, May 2012.

[69] T.-C. Wang, J.-Y. Zhu, E. Hiroaki, M. Chandraker, A. A. Efros, and R. Ramamoorthi, “A 4d light-field dataset and CNN architectures for material recognition,” in *Computer Vision – ECCV 2016*, pp. 121–138, Springer International Publishing, 2016.

[70] D. Romero, M. Ma, and G. B. Giannakis, “Kernel-based reconstruction of graph signals,” *IEEE Transactions on Signal Processing*, vol. 65, pp. 764–778, Feb. 2017.

[71] V. R. M. Elias and W. A. Martins, “Graph Fourier transform for light field compression,” in *Simpósio Brasileiro de Telecomunicações e Processamento de Sinais*, pp. 881–885, Sept. 2017.

[72] V. R. M. Elias and W. A. Martins, “On the use of graph Fourier transform for light-field compression,” *Journal of Communication and Information Systems*, vol. 33, pp. 92–103, May 2018.

[73] V. R. M. Elias, W. A. Martins, and S. Werner, “Diffusion-based virtual graph adjacency for Fourier analysis of network signals,” in *Simpósio Brasileiro de Telecomunicações e Processamento de Sinais*, pp. 1–5, Dec. 2020.

[74] V. R. M. Elias, W. A. Martins, and S. Werner, “Extended adjacency and scale-dependent graph Fourier transform via diffusion distances,” *IEEE Transactions on Signal and Information Processing over Networks*, vol. 6, pp. 592–604, Aug. 2020.

[75] V. C. Gogineni, V. R. M. Elias, W. A. Martins, and S. Werner, “Graph diffusion kernel LMS using random Fourier features,” in *Asilomar Conference on Signals, Systems, and Computers*, pp. 1–5, Nov. 2020.

[76] V. R. M. Elias, V. C. Gogineni, W. A. Martins, and S. Werner, “Adaptive graph filters in reproducing kernel Hilbert spaces: Design and performance analysis,” *IEEE Transactions on Signal and Information Processing over Networks*, vol. 7, pp. 62–74, 2021.

[77] V. R. M. Elias, V. C. Gogineni, W. A. Martins, and S. Werner, “Kernel regression on graphs in random Fourier features space,” in *International Conference on Acoustics, Speech, & Signal Processing*, pp. 1–5, 2021.

[78] V. R. M. Elias, V. C. Gogineni, W. A. Martins, and S. Werner, “Kernel regression over graphs using random Fourier features,” *IEEE Transactions on Signal Processing*, pp. 1–12, 2021.

[79] R. Merris, “Laplacian matrices of graphs: a survey,” *Linear Algebra and its Applications*, vol. 197-198, pp. 143–176, Jan. 1994.

[80] A. Sandryhaila and J. M. Moura, “Nearest-neighbor image model,” in *IEEE International Conference on Image Processing*, pp. 2521–2524, Sept. 2012.

[81] P. S. R. Diniz, E. A. B. da Silva, and S. L. Netto, *Digital Signal Processing System Analysis and Design.* Cambridge University Press, 2009.

[82] M. Püschel and J. M. F. Moura, “Algebraic Signal Processing Theory,” *arXiv e-prints*, vol. 60, pp. 1–67, Dec. 2006.

[83] M. Püschel, “Algebraic Signal Processing Theory: An Overview,” in *IEEE Digital Signal Processing Workshop & IEEE Signal Processing Education Workshop*, pp. 386–391, IEEE, Sept. 2006.

[84] G. B. Ribeiro, “Um estudo sobre operadores para deslocamento de sinais sobre grafos,” Master’s thesis, Federal University of Pernambuco, Recife, Pernambuco, Brazil, 2018.

[85] F. R. K. Chung, *Spectral Graph Theory.* American Mathematical Society, 1997.

[86] A. G. Marques, S. Segarra, and G. Mateos, “Signal processing on directed graphs: The role of edge directionality when processing and learning from network data,” *IEEE Signal Processing Magazine*, vol. 37, pp. 99–116, Nov. 2020.

[87] S. Furutani, T. Shibahara, M. Akiyama, K. Hato, and M. Aida, “Graph signal processing for directed graphs based on the hermitian laplacian,” in *Machine Learning and Knowledge Discovery in Databases*, pp. 447–463, Springer International Publishing, 2020.

[88] L. Stankovic, M. Dakovic, and E. Sejdic, “Vertex-frequency analysis: A way to localize graph spectral components [lecture notes],” *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 176–182, 2017.

[89] A. Gershun, “The light field,” *Journal of Mathematics and Physics*, vol. 18, no. 1-4, pp. 51–151, 1939.

[90] E. H. Adelson and J. R. Bergen, “The plenoptic function and the elements of early vision,” in *Computational Models of Visual Processing*, pp. 3–20, MIT Press, 1991.

[91] M. Levoy, R. Ng, A. Adams, M. Footer, and M. Horowitz, “Light field microscopy,” *ACM Transactions on Graphics*, vol. 25, p. 924, July 2006.

[92] N. C. Pégard, H.-Y. Liu, N. Antipa, M. Gerlock, H. Adesnik, and L. Waller, “Compressive light-field microscopy for 3d neural activity recording,” *Optica*, vol. 3, p. 517, May 2016.

[93] M. Levoy and P. Hanrahan, “Light field rendering,” in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques - SIGGRAPH '96*, ACM Press, 1996.

[94] M. Magnor and B. Girod, “Data compression for light-field rendering,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, pp. 338–343, Apr. 2000.

[95] T. Sakamoto, K. Kodama, and T. Hamamoto, “A study on efficient compression of multi-focus images for dense light-field reconstruction,” in *2012 Visual Communications and Image Processing*, IEEE, Nov. 2012.

[96] C. Perra, “On the coding of plenoptic raw images,” in *22nd Telecommunications Forum Telfor (TELFOR)*, IEEE, Nov. 2014.

[97] A. Vieira, H. Duarte, C. Perra, L. Tavora, and P. Assuncao, “Data formats for high efficiency coding of lytro-illum light fields,” in *International Conference on Image Processing Theory, Tools and Applications (IPTA)*, IEEE, Nov. 2015.

[98] C. Perra and P. Assuncao, “High efficiency coding of light field images based on tiling and pseudo-temporal data arrangement,” in *IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, IEEE, July 2016.

[99] E. Cornwell, L. Li, Z. Li, and Y. Sun, “An efficient compression scheme for the multi-camera light field image,” in *IEEE 19th International Workshop on Multimedia Signal Processing (MMSP)*, IEEE, Oct. 2017.

[100] T. Ebrahimi, S. Foessel, F. Pereira, and P. Schelkens, "JPEG pleno: Toward an efficient representation of visual reality," *IEEE MultiMedia*, vol. 23, pp. 14–20, Oct. 2016.

[101] M. Rossi and P. Frossard, "Geometry-consistent light field super-resolution via graph-based regularization," *IEEE Transactions on Image Processing*, vol. 27, pp. 4207–4218, Sept. 2018.

[102] V. K. Ghassab and N. Bouguila, "Light field super-resolution using edge-preserved graph-based regularization," *IEEE Transactions on Multimedia*, vol. 22, pp. 1447–1457, June 2020.

[103] Y.-H. Chao, G. Cheung, and A. Ortega, "Pre-demosaic light field image compression using graph lifting transform," in *IEEE International Conference on Image Processing*, IEEE, Sept. 2017.

[104] M. Rizkallah, X. Su, T. Maugey, and C. Guillemot, "Graph-based transforms for predictive light field compression based on super-pixels," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, IEEE, Apr. 2018.

[105] M. Rizkallah, X. Su, T. Maugey, and C. Guillemot, "Geometry-aware graph transforms for light field compact representation," *IEEE Transactions on Image Processing*, vol. 29, pp. 602–616, 2020.

[106] T. Ebrahimi, I. Viola, P. Frossard, and H. P. Maretic, "A graph learning approach for light field image compression," in *Applications of Digital Image Processing* (A. G. Tescher, ed.), SPIE, Sept. 2018.

[107] M. Rizkallah, T. Maugey, and C. Guillemot, "Rate-distortion optimized graph coarsening and partitioning for light field coding," *IEEE Transactions on Image Processing*, pp. 1–14, 2021.

[108] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Applied and Computational Harmonic Analysis*, vol. 30, pp. 129–150, Mar. 2011.

[109] W. Hu, G. Cheung, A. Ortega, and O. C. Au, "Multiresolution graph Fourier transform for compression of piecewise smooth images," *IEEE Transactions on Image Processing*, vol. 24, pp. 419–433, Jan. 2015.

[110] G. Shen, W.-S. Kim, S. K. Narang, A. Ortega, J. Lee, and H. Wey, "Edge-adaptive transforms for efficient depth map coding," in *28th Picture Coding Symposium*, IEEE, Dec. 2010.

[111] "The (new) Stanford light field archive." `http://lightfield.stanford.edu/lfs.html`. Accessed: 2018-02-20.

[112] "4D Light Field Benchmark." `http://hci-lightfield.iwr.uni-heidelberg.de`. Accessed: 2018-02-20.

[113] K. Honauer, O. Johannsen, D. Kondermann, and B. Goldluecke, "A dataset and evaluation methodology for depth estimation on 4d light fields," in *Computer Vision – ACCV 2016*, pp. 19–34, Springer International Publishing, 2017.

[114] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, pp. 215–233, Jan. 2007.

[115] J. Qin, Q. Ma, Y. Shi, and L. Wang, "Recent advances in consensus of multi-agent systems: A brief survey," *IEEE Transactions on Industrial Electronics*, vol. 64, pp. 4972–4983, June 2017.

[116] B. Kailkhura, S. Brahma, and P. K. Varshney, "Data falsification attacks on consensus-based detection systems," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 3, pp. 145–158, Mar. 2017.

[117] L. Lovász, "Random walks on graphs: A survey," *Combinatorics Paul Erdos is Eighty*, vol. 2, no. Volume 2, pp. 1–46, 1993.

[118] C. Gkantsidis, M. Mihail, and A. Saberi, "Random walks in peer-to-peer networks," in *Proceedings of IEEE Conference on Computer Communications*, vol. 1, pp. 120–130, 2004.

[119] R. R. Coifman and S. Lafon, "Diffusion maps," *Applied and Computational Harmonic Analysis*, vol. 21, pp. 5–30, July 2006.

[120] A. Heimowitz and Y. C. Eldar, "A unified view of diffusion maps and signal processing on graphs," in *Proceedings of the International Conference on Sampling Theory and Applications*, pp. 308–312, July 2017.

[121] A. Heimowitz and Y. C. Eldar, "The Nystrom extension for signals defined on a graph," in *Proceedings IEEE International Conference on Acoustics, Speech, Signal Processing*, pp. 4199–4203, Apr. 2018.

[122] R. R. Coifman, S. Lafon, A. B. Lee, M. Maggioni, B. Nadler, F. Warner, and S. W. Zucker, "Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps," *Proceedings of the National Academy of Sciences*, vol. 102, pp. 7426–7431, May 2005.

[123] R. R. Coifman, S. Lafon, A. B. Lee, M. Maggioni, B. Nadler, F. Warner, and S. W. Zucker, "Geometric diffusions as a tool for harmonic analysis and structure definition of data: Multiscale methods," *Proceedings of the National Academy of Sciences*, vol. 102, pp. 7432–7437, May 2005.

[124] R. Talmon, I. Cohen, S. Gannot, and R. R. Coifman, "Diffusion maps for signal processing: A deeper look at manifold-learning techniques based on kernels and graphs," *IEEE Signal Processing Magazine*, vol. 30, no. 4, pp. 75–86, 2013.

[125] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer Networks*, vol. 54, pp. 2787–2805, Oct. 2010.

[126] B. Rashid and M. H. Rehmani, "Applications of wireless sensor networks for urban areas: A survey," *Journal of Network and Computer Applications*, vol. 60, pp. 192–219, Jan. 2016.

[127] E. Fadel, V. Gungor, L. Nassef, N. Akkari, M. A. Malik, S. Almasri, and I. F. Akyildiz, "A survey on wireless sensor networks for smart grid," *Computer Communications*, vol. 71, pp. 22–33, Nov. 2015.

[128] W. Ren, R. W. Beard, and E. M. Atkins, "Information consensus in multivehicle cooperative control," *IEEE Control Systems Magazine*, vol. 27, pp. 71–82, Apr. 2007.

[129] M. A. Mahmood, W. K. Seah, and I. Welch, "Reliability in wireless sensor networks: A survey and challenges ahead," *Computer Networks*, vol. 79, pp. 166–187, Mar. 2015.

[130] B. Mostefa and G. Abdelkader, "A survey of wireless sensor network security in the context of internet of things," in *International Conference on Information and Communication Technologies for Disaster Management*, pp. 1–8, 2017.

[131] S. Kar and J. M. F. Moura, "Distributed consensus algorithms in sensor networks: Quantized data and random link failures," *IEEE Transactions on Signal Processing*, vol. 58, pp. 1383–1400, Mar. 2010.

[132] I. Kayes and A. Iamnitchi, "Privacy and security in online social networks: A survey," *Online Social Networks and Media*, vol. 3-4, pp. 1–21, Oct. 2017.

[133] Y. Chen, S. Kar, and J. M. F. Moura, "The Internet of Things: Secure Distributed Inference," *IEEE Signal Processing Magazine*, vol. 35, pp. 64–75, Sept. 2018.

[134] Y. Chen, S. Kar, and J. M. F. Moura, "Resilient distributed estimation through adversary detection," *IEEE Transactions on Signal Processing*, vol. 66, pp. 2455–2469, May 2018.

[135] Y. Chen, S. Kar, and J. M. F. Moura, "Resilient distributed estimation: Sensor attacks," *IEEE Transactions on Automatic Control*, vol. 64, pp. 3772–3779, Sept. 2019.

[136] E. Drayer and T. Routtenberg, "Detection of false data injection attacks in power systems with graph Fourier transform," in *IEEE Global Conference on Signal and Information Processing*, pp. 890–894, 2018.

[137] M. Khatua, S. H. Safavi, and N. Cheung, "Sparse Laplacian component analysis for internet traffic anomalies detection," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 4, no. 4, pp. 697–711, 2018.

[138] C. Goutte and E. Gaussier, "A probabilistic interpretation of precision, recall and f-score, with implication for evaluation," in *Lecture Notes in Computer Science*, pp. 345–359, Springer Berlin Heidelberg, 2005.

[139] National Oceanic and Atmospheric Administration, Department of Commerce, "Global Surface Summary of the Day - GSOD - Data.gov." United States of America, 2021. [Online] Available: `https://catalog.data.gov/dataset/global-surface-summary-of-the-day-gsod`. Accessed: 2021-02-02.

[140] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Proceedings of the International Conference on Neural Information Processing Systems*, pp. 1177—1184, 2007.

[141] A. H. Sayed, S. Tu, J. Chen, X. Zhao, and Z. J. Towfic, "Diffusion strategies for adaptation and learning over networks: an examination of distributed strategies and network behavior," *IEEE Signal Processing Magazine*, vol. 30, pp. 155–171, May 2013.

[142] P. Bodik, W. Hong, C. Guestrin, S. Madden, M. Paskin and R. Thibaux, "Intel Lab Data." United States of America, 2004. [Online] Available: `http://db.csail.mit.edu/labdata/labdata.html`. Accessed: 2021-02-02.

[143] P. S. R. Diniz, *Adaptive Filtering*. Springer, 2013.

[144] KTH Royal Institute of Technology, Division of Information Science and Engineering, "Reproducible Research." Sweden, 2020. [Online] Available: `https://www.kth.se/ise/research/reproducibleresearch-1.433797`. Accessed: 2021-02-02.

Chapter 8

Publications on Light Field Compression using the GFT

P1 V. R. M. Elias and W. A. Martins, “Graph Fourier transform for light field compression,” in *Simpósio Brasileiro de Telecomunicações e Processamento de Sinais*, pp. 881–885, Sept. 2017.

P2 V. R. M. Elias and W. A. Martins, “On the use of graph Fourier transform for light-field compression,” *Journal of Communication and Information Systems*, vol. 33, pp. 92–103, May 2018.

Graph Fourier Transform for Light Field Compression

Vitor Rosa Meireles Elias and Wallace Alves Martins

Abstract— **This work proposes the use of graph Fourier transform (GFT) for light field data compression. GFT is a tool developed for the emerging field of digital signal processing on graphs, which combines graph theory and classical DSP in order to exploit signal-related information present on graph structures. The proposed method explores the high correlation of residual images from light field. Simulations with real light field data indicate significant reduction in the number of coefficients for data representation; for instance, an 8.97% reduction was achieved while keeping smaller mean squared error when compared to discrete cosine transform-based compression.**

Keywords— **Signal Processing on Graphs, Graph Fourier Transform, Compression, Light Field, Discrete Cosine Transform.**

I. Introduction

Light field imaging is a promising technology that opens a variety of new possibilities to entertainment industries, such as photography and cinema, by capturing 4D data from a scene. Practical light field capturing techniques usually consist of a microlens array placed between the sensor plane and the main lens of a digital camera so that each microlens generates a micro-image associated with a different perspective of the scene. An alternative way of capturing a light field is through an array of cameras, or by moving a single camera and capturing the scene on a grid of determined positions. All these methods generate a large amount of data when compared to traditional imaging, since many images are used to compose a single scene. Dealing with the resulting huge amount of data is a challenging task [1], [2].

In order to provide solutions and a standard framework to deal with data generated by light field imaging and other techniques, the JPEG standardization committee created, in 2015, the "JPEG Pleno" initiative and the process is due to continue through the next years with a first international standard in 2018 [3].

Given the advances in video coding technology and the advent of high-efficiency encoders that are suitable to various types of contents, recent researches focus on improving the compression of light field data by using HEVC-based solutions [4]–[6]. Since HEVC encompasses a range of complex signal processing steps, researchers usually focus on individual components of the encoding scheme, such as intra or inter prediction, data-arrangement, or other key processing blocks.

This work focuses on the transform block of a simplified encoding scheme. Instead of employing *discrete-cosine transform* (DCT) along with *discrete-sine transform* (DST) as in the HEVC encoder, this paper proposes the use of *graph Fourier transform* (GFT) [7] in order to reduce the number of coefficients required to represent a block of prediction residual, which is the difference between an image and its prediction. GFT has recently arisen as one of the main signal processing tools within the emerging field of *Digital Signal Processing on Graphs* (DSP_G) [8]. In DSP_G, signals are represented as graphs, which may contain more information than usual signal representations, and the traditional DSP tools are translated and adapted to process data in the graph structure.

GFT applied to a graph signal is able to provide better compression quality in terms of mean squared error (MSE), while keeping the same number of transform coefficients as the two-dimensional DCT applied to a traditional image signal. However, unlike DCT, GFT and its companion *inverse graph Fourier transform* (IGFT) depend on the signal-related adjacency matrix and the graph structure requires extra data for signal representation [7], [9]. This work proposes a method that explores the redundancy among light field images in order to reduce the impact of the extra data on the present graph structure and compares the efficiency of this method to that of the DCT.

It is worth pointing out that the use of GFT within the image processing context is not new. Indeed, some previous works use GFT for image compression. The seminal paper [7] that introduces GFT as a tool for DSP_G presents image compression as an example of GFT application. In addition, the authors in [9] propose a hybrid image transform for color and depth images based on DCT and GFT.

This paper is organized as follows. Section II provides the background knowledge on light field and signal processing on graphs required for this work. Section III introduces the proposed compression methodology. Section IV describes the simulations and discusses the results obtained. Section V compiles some of the next steps for future works, and Section VI presents some conclusions of this paper.

II. Background on Light Field and DSP_G

Although both light field and graph theories are known for a long time, only recently technology allowed the construction of real and practical light field capturing devices and, even more recently, the fundamentals of digital signal processing on graphs were proposed. This section provides the basic knowledge about light field data and the DSP_G concepts and tools used in this work.

A. Light Field

Light field data usually consist of a set of multiple images of different perspectives from a scene that are captured either

Mr. Vitor R. M. Elias and Prof. Wallace A. Martins are with the Federal University of Rio de Janeiro, Rio de Janeiro, RJ, Brazil (emails: vitor.elias@smt.ufrj.br, wallace.martins@smt.ufrj.br).

The authors are grateful for the financial support provided by CNPq, CAPES, and FAPERJ, Brazilian research councils.

Fig. 1. Example of images that compose light field data.

by an array of microlenses, an array of cameras, or by moving and capturing with a single camera. Fig. 1 shows an example of 16 images that compose a light field obtained from the *Stanford light field archive* [10]. The entire light field data for this case was captured by a moving camera on a rectangular grid with 16×16 positions, yielding a total of 256 images; this is the data set used throughout the paper.

Some important applications that justify why light field imaging is a promising technology include the *light field rendering*, which allows the creation of novel views by manipulating multiple previously captured views, and the *synthetic aperture photography*, which allows photographs to be refocused after they are taken [11].

B. Digital Signal Processing on Graphs

A graph is the pair $G = (\mathcal{V}, \mathcal{A})$, where $\mathcal{V} = \{v_0, \ldots, v_{N-1}\}$ is the set of N *vertices*, and $\mathcal{A} = \{a_{00} \ldots a_{(N-1)(N-1)}\}$ is the set of N^2 *edges*. The relation between $\mathcal{V}$ and $\mathcal{A}$ is as follows: each element a_{ij} represents the edge connecting vertex v_j to vertex v_i.[1] In other words, an edge represents a relation between two vertices, and this relation depends on the underlying application of the graph. In a directed graph, edges have orientations and a_{ij} may differ from a_{ji}. If a graph is undirected, edges have no orientations. The set $\mathcal{A}$ of all edges can also be represented by an $N \times N$ *adjacency matrix* $\mathbf{A}$, which is symmetric if the graph is undirected.

A finite-duration complex-valued discrete-time signal $s[n]$ can be regarded as a function $s : \{0, 1, \ldots, N-1\} \to \mathbb{C}$ that maps points within a well-structured domain into the complex plane. Indeed, any two points n_1, n_2 within the domain $\{0, 1, \ldots, N-1\}$ can be compared, i.e. $n_1 > n_2$ or $n_1 = n_2$ or $n_1 < n_2$; for any $n \in \{0, 1, \ldots, N-1\}$, the inequalities $N - 1 \geq n \geq 0$ always hold. These domain properties induce several useful properties in the analysis of discrete-time signals $s[n]$. Nonetheless, there are many applications that call for the use of a more general domain and, in these cases, graphs may be the appropriate structure.

[1]In fact, if $a_{ij} = 0$ one can consider there is no edge connecting v_j to v_i.

The concept of signals on graphs uses the set of vertices $\mathcal{V}$ of a graph G as the domain of a dataset of N elements, and the set of edges $\mathcal{A}$ of the graph G to encode an underlying relationship between the elements of this dataset. For example, for a dataset of temperature measurements of N sensors, vertices can represent the sensors' spatial positions, whereas edges may represent the distances between pairs of sensors on an undirected graph. Note that, in this case, the domain is not ordered — one cannot state that a sensor 3D position is larger/smaller than the other, in principle.

In this paper, an entire signal $s : \mathcal{V} \to \mathbb{C}$ on the vertices of a graph is referred to as $\mathbf{s}$, in which the n-th entry of $\mathbf{s}$ is $s_n = s[v_n]$, with $v_n \in \mathcal{V}$. Tools in $\mathrm{DSP_G}$ are usually developed as equivalent forms of existing tools in classical DSP [8]. In $\mathrm{DSP_G}$, given a graph G with adjacency matrix $\mathbf{A}$, a *graph shift* is defined as

$$\tilde{\mathbf{s}} = \mathbf{A}\mathbf{s}. \tag{1}$$

The operation defined in (1) is the graph equivalent of the *delay* or *shift*, which is the basic building block of filters in classical DSP. That is, in graph domain, shifting a signal $\mathbf{s}$ is equivalent to replacing each signal sample s_n with a linear combination of its neighborhood according to weights given by the adjacency matrix $\mathbf{A}$. Classical DSP shift can be viewed as a special case of graph shift when the adjacency matrix is the *cyclic shift matrix*

$$\mathbf{C} = \begin{bmatrix} & & & 1 \\ 1 & & & \\ & \ddots & & \\ & & 1 & \end{bmatrix}. \tag{2}$$

The concept of filters is also extended to the graph domain. Indeed, in classical DSP, a finite-duration impulse response (FIR) filter of length $L \leq N$ with coefficients h_l induces the following circular convolution:

$$\begin{bmatrix} \tilde{s}[0] \\ \tilde{s}[1] \\ \vdots \\ \tilde{s}[N-1] \end{bmatrix} = \underbrace{\begin{bmatrix} h_0 & h_{N-1} & \cdots & h_1 \\ h_1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & h_{N-1} \\ h_{N-1} & \cdots & h_1 & h_0 \end{bmatrix}}_{=\mathbf{H}(\mathbf{C}) = \sum_{l=0}^{L-1} h_l \mathbf{C}^l} \begin{bmatrix} s[0] \\ s[1] \\ \vdots \\ s[N-1] \end{bmatrix}, \tag{3}$$

where $h_l = 0$ for $l \geq L$. Similarly, a linear, shift-invariant *graph filter* is defined as a polynomial over a general adjacency matrix $\mathbf{A}$, i.e.,

$$\mathbf{H}(\mathbf{A}) = \sum_{l=0}^{L-1} h_l \mathbf{A}^l. \tag{4}$$

C. Graph Fourier Transform

In classical DSP, the basis vectors that comprise a discrete Fourier transform are the orthonormal vectors that diagonalize the cyclic shift matrix $\mathbf{C}$ in (2), and the frequencies are their corresponding eigenvalues. Similarly, a graph Fourier transform can be defined by using the vectors that diagonalize the adjacency matrix $\mathbf{A}$. In other words, the *graph Fourier transform* is defined as

$$\hat{\mathbf{s}} = \mathbf{F}\mathbf{s} = \mathbf{V}^{-1}\mathbf{s}, \tag{5}$$

where $\mathbf{V}$ is a matrix whose columns are the eigenvectors of the adjacency matrix $\mathbf{A}$ if it is diagonalizable. If $\mathbf{A}$ is not diagonalizable, $\mathbf{V}$ is the set of generalized eigenvectors from the Jordan decomposition of $\mathbf{A}$ [7], [12]. The *inverse graph Fourier transform* is defined as

$$\mathbf{s} = \mathbf{F}^{-1}\hat{\mathbf{s}} = \mathbf{V}\hat{\mathbf{s}}. \tag{6}$$

If $\mathbf{V}$ is an orthogonal matrix, which is the case when $\mathbf{V}$ is the matrix whose columns are normalized eigenvectors of a symmetric adjacency matrix $\mathbf{A}$, then $\mathbf{V}^{-1} = \mathbf{V}^{\mathrm{T}}$.

III. Light Field Compression Methodology

This section describes the compression scheme, signal models, and figures of merit adopted throughout the paper.

A. Compression process

The main interest of this work is to evaluate how efficient the proposed GFT-based compression method can be in comparison to the traditional DCT when applied to blocks of prediction residual. In order to define how the residual is computed, the prediction needs to be defined for the light field case, since images are not associated with a time sequence as usual on a video coding process.

For the light field presented in Section II-A, considering the 16×16 grid over which the multiple images are taken, this work assumes that the first image from each of the 16 grid lines is an intra image, i.e., no prediction is assumed when encoding these images. The following 15 images from each line are inter images for which GFT and DCT are used to perform a lossy encoding of their corresponding residual blocks. Given the capturing process and the format of light field data, it is clear that images next to each other tend to be more similar than images far from each other. Hence, the prediction should be restricted to the spatial neighborhood of a given image in order to reduce the prediction residual. A simple prediction scheme adopted in this work is $\mathbf{I}_k^{\mathrm{p}} = \mathbf{I}_{k-1}$, where $k \in \{2, 3, \ldots, K\}$ is the index of the k-th image at each line of the light field array and $\mathbf{I}^{\mathrm{p}}$ is the prediction for an image. For the light field data treated in this paper, $K = 16$ images per light field line. Finally, the residual is given by $\mathbf{R}_k = \mathbf{I}_k - \mathbf{I}_k^{\mathrm{p}}$.

Fig. 2 illustrates two examples of other prediction schemes that may be utilized: zig-zag sequencing starting from the upper-left corner of the light field array, or blocks of images where all residuals are relative to a central image, for example.

B. Signal model

Once the residual image $\mathbf{R}_k$ is obtained, it is divided into T blocks of size 32×32 pixels. If much smaller blocks are used, such as 4×4 or 8×8 pixels, blocks at the same position for different residual images may have low correlation with each other. The proposed compression method requires that blocks at the same position for various residual images are similar.

In order for the GFT to be applied, the residual block needs to be represented as a graph. Consider the t-th block of $M_1 \times M_2$ pixels from residual image $\mathbf{R}_k$, denoted as $\mathbf{B}_{k,t}$, with $t \in \{1, 2, \ldots, T\}$. The signal $\mathbf{s}_{k,t}$ associated with the

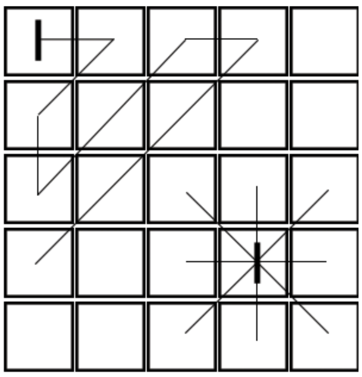


Fig. 2. Examples of alternative prediction schemes on the light field array. The bold mark in the center of a box indicates an intra image.

block is defined as the column vector formed by stacking the columns of $\mathbf{B}_{k,t}$. The corresponding adjacency matrix $\mathbf{A}_{k,t}$ is defined by the nearest-neighbor (NN) image model proposed in [13]. This model states that a pixel is only related to the nearest pixels as shown in Fig. 3, i.e., every edge from the graph representation that lies outside the neighborhood of a pixel is equal to zero. Considering a pixel from the center of the block, its neighborhood is composed by the pixels directly under, above, to the left, and to the right from the reference pixel. If a pixel is located at the upper-left corner of the block, only the pixel to the right and the pixel under it are considered as neighbors. This model leads to a sparse $\mathbf{A}_{k,t}$ matrix with a fixed structure. Sparsity comes from the fact that each vertex has at most four nonzero edges associated with its neighborhood. The structure is fixed because these edges are represented by the same entry positions in $\mathbf{A}_{k,t}$ for different blocks of the same size, although entry values (i.e. edge weights) may differ. Another key feature of the NN image model is that, for a given pair of columns of pixels in a block, all horizontal edges connecting the two columns have the same values. Likewise, for a pair of lines, all vertical edges have the same values. Considering this property, the $N \times N$ matrix $\mathbf{A}_{k,t}$ associated with an $M_1 \times M_2$ block, where $N = M_1 M_2$, has at most $(M_1 - 1) + (M_2 - 1)$ different nonzero values. The edge values are computed by minimizing $\|\mathbf{A}\mathbf{s} - \mathbf{s}\|_2$ subject to the matrix structure imposed by the previously described model properties. This is an overdetermined least-squares minimization problem.

Note that $\mathbf{A}_{k,t}$ is symmetric as pixel relations are assumed to be undirected and, thus, is diagonalizable.

C. Proposed method

Once the adjacency matrix is computed for a block, graph Fourier transform matrix $\mathbf{F}_{k,t}$ can be computed as the transpose of the matrix composed by normalized eigenvectors of $\mathbf{A}_{k,t}$, as it is diagonalizable and symmetric. Transform coefficients $\hat{\mathbf{s}}_{k,t}$ are obtained by applying $\mathbf{F}_{k,t}$ to $\mathbf{s}_{k,t}$. The original block information can be recovered from transform coefficients using $\mathbf{F}_{k,t}^{\mathrm{T}}$ to perform the IGFT.

By setting the Q smallest transform coefficients to zero, the signal $\mathbf{s}_{k,t}^{Q}$ recovered by the IGFT is a compressed version of $\mathbf{s}_{k,t}$, and the compressed block $\mathbf{B}_{k,t}^{Q}$ is obtained from the compressed graph signal. In this paper, the MSE between

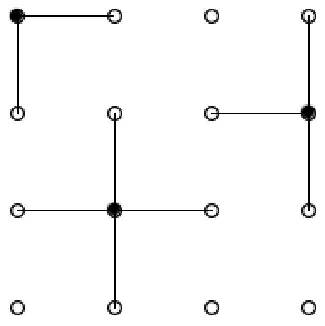


Fig. 3. Relation edges proposed in the nearest-neighbor image model for corner, edges, and central pixels.

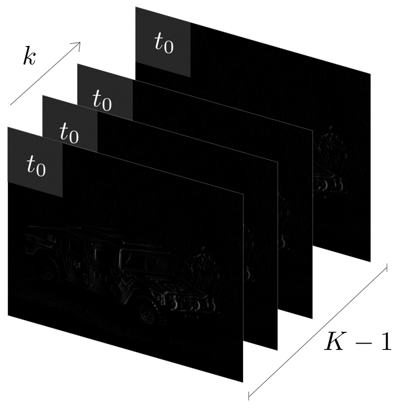


Fig. 4. Representation of a block position t_0 for $K-1$ residual images.

compressed and original residual blocks is used as figure of merit to determine how compression affects the images. It is expected that the MSE will behave as a non-decreasing function of Q, as there is a trade-off between compression rate and image quality. As both GFT and DCT concentrate most of signal energy on few transform coefficients, only a small part of the block information is lost during compression. When compared to DCT, GFT provides better MSE for the same Q, but the downside of GFT is that $\mathbf{F}_{k,t}$ depends on $\mathbf{A}_{k,t}$ associated with the block being compressed, unlike DCT, which is a fixed transform. That is, when performing inverse transform, transform coefficients and block-dependent $\mathbf{F}^{\mathrm{T}}_{k,t}$ are required in the IGFT case, and IDCT requires only the transform coefficients.

Considering light field data, images next to each other are similar and, consequently, the associated residual images are highly redundant. By exploring this redundancy, it is possible to avoid transmitting a different $\mathbf{F}_{k,t}$ or $\mathbf{A}_{k,t}$ with every single block. In the proposed scheme, for a given block position t_0, only one $\mathbf{A}_{k,t_0}$ is considered for the entire light field line, i.e., for $K-1$ residual images, as depicted in Fig. 4. In the proposed compression method, only the adjacency matrices associated with the central residual image $\mathbf{R}_8$, with blocks $\mathbf{B}_{8,t}$, are computed, for the case of a light field with 16 total images per line and 15 residual images. Note that the first image from each line is an intra image and no prediction nor residual image is associated with it.

By utilizing a single adjacency matrix for $K-1$ similar blocks, the impact of transmitting $\mathbf{A}_{k,t}$ along with the transform coefficients is reduced, but GFT efficiency may be degraded for the blocks that are not the reference blocks when the adjacency matrix is computed. Central residual image $\mathbf{R}_8$ is defined heuristically as reference, in order to minimize degradation of GFT. This definition assumes that blocks' similarity is reduced the further a image is from the reference image, and the central image is, on average, the best approximation for the entire line.

It is important to note that, by transmitting $\mathbf{A}$ instead of $\mathbf{F}$, the transmitted data is critically reduced, given the inherent sparsity and structure of the adjacency matrix discussed in Section III-B, but complexity on receiver side is increased since $\mathbf{F}$ needs to be computed from the adjacency matrix.

IV. Simulation and Results

During the simulations, only luminance component from the light field images is considered. Images are 512×640 pixels. Utilizing blocks with 32×32 pixels, DCT yields 1024 transform coefficients, from which the 100 largest coefficients are kept, i.e., $Q = 924$. MSE for reconstructed residual images from DCT compressed coefficients is computed. The algorithm searches for the largest Q for which GFT results on better MSE for a residual image when compared to DCT. That is, how many more GFT coefficents can be set to zero and still yield smaller mean squared error. In this simulation, Q is set for a whole image, thus, every block $\mathbf{B}_{k,t}$ in the k-th residual image is represented by the same number of transform coefficients.

The result for this simulation, considering only the first light field line, is shown in Fig. 5 as the difference between Q used for GFT and DCT. The resulting MSE for this case is shown in Fig. 6. As expected, the best case occurs for the reference residual image $\mathbf{R}_8$ from which $\mathbf{A}$ matrix is computed, where GFT is capable of providing better MSE while setting 28 more coefficients to zero. Results also show that, as images get further apart from the reference image, GFT efficiency decays when compared to DCT. For the last residual image $\mathbf{R}_{15}$, GFT needs two more coefficients than DCT in order to provide smaller MSE. Previous simulations are extended to the whole light field data and the result is presented in Fig. 7. This shows consistency of the method for all light field lines.

Considering the total number of coefficients, in the present simulation, DCT requires 100 transform coefficients per block. Each image is divided into $T = 320$ blocks, and there are 240 residual images. This yields 7,680,000 transform coefficients for the whole light field data in the DCT case. From the previous results, GFT requires 6,673,600 transform coefficients and 320 $\mathbf{A}$ matrices per line. As only 62 coefficients are needed to construct each $\mathbf{A}$, as shown in Section III-B, GFT requires a total of 6,991,040 coefficients. This means 8.97% reduction in number of coefficients while keeping smaller MSE than DCT.

When applied to two other light field sets from [10], namely *Lego Knights* and *Tarot cards and crystal ball*, the proposed compression methodology achieves a combined reduction in number of coefficients of 4.6% as compared to DCT.

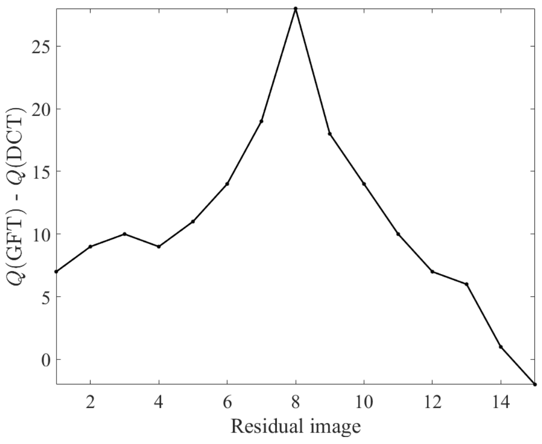


Fig. 5. Difference between Q values used for GFT and DCT on the first light field line.

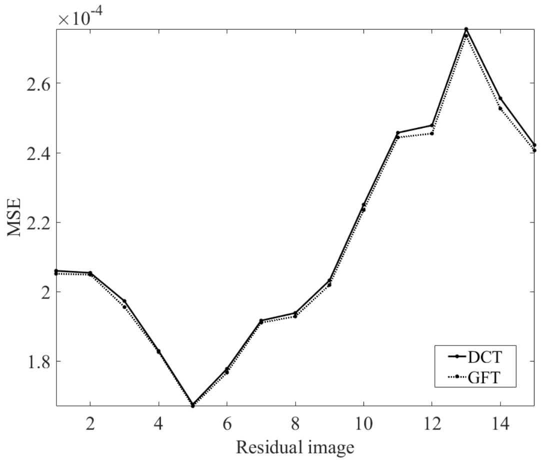


Fig. 6. MSE for each residual image for GFT and DCT.

V. Future Works

This paper contains some preliminary results of an ongoing research. Thus, some of the assumptions and heuristics adopted here admit deeper investigations, as both light field and DSP_G are developing fields, and putting them together is new to the best of our knowledge. Hence, the proposed method should be tested on a larger data base; the NN image model could be used considering data from all residual blocks instead of a single reference block for matrix $\mathbf{A}$ computation, which may improve GFT efficiency for various blocks; other image models and prediction schemes should be tested; new figures of merit for compression efficiency evaluation should be used.

VI. Conclusions

This work provided an efficient method of using GFT as an alternative to DCT when compressing residual image blocks from light field data. The proposed method allows significant reduction in the number of transmitted coefficients while providing smaller MSE when compared to DCT. The method was applied to real light field data and results indicate that GFT is a viable transform for compression purposes when there is high correlation between various images.

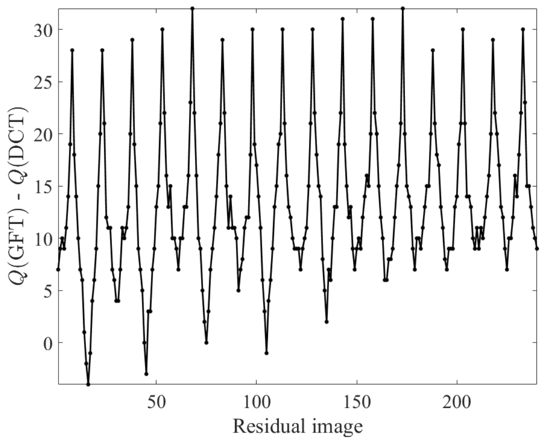


Fig. 7. Difference between Q values used for GFT and DCT on the whole light field.

References

[1] M. Magnor and B. Girod, "Data compression for light-field rendering," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, pp. 338–343, Apr 2000.

[2] T. Sakamoto, K. Kodama, and T. Hamamoto, "A study on efficient compression of multi-focus images for dense Light-Field reconstruction," *2012 IEEE Visual Communications and Image Processing, VCIP 2012*, Nov 2012.

[3] T. Ebrahimi, S. Foessel, F. Pereira, and P. Schelkens, "JPEG Pleno: toward an efficient representation of visual reality," *IEEE Multimedia*, vol. 23, pp. 14–20, Oct.-Dec 2016.

[4] C. Perra and P. Assuncao, "High efficiency coding of light field images based on tiling and pseudo-temporal data arrangement," in *2016 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, pp. 1–4, IEEE, Jul 2016.

[5] R. Monteiro, L. Lucas, C. Conti, P. Nunes, N. Rodrigues, S. Faria, C. Pagliari, E. da Silva, and L. Soares, "Light field HEVC-based image coding using locally linear embedding and self-similarity compensated prediction," in *2016 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, pp. 1–4, IEEE, Jul 2016.

[6] Y. Li, R. Olsson, and M. Sjostrom, "Compression of unfocused plenoptic images using a displacement intra prediction," in *2016 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, pp. 1–4, IEEE, Jul 2016.

[7] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs: graph Fourier transform," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 62, pp. 6167–6170, IEEE, May 2013.

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

[9] T. Kanamori, K. Mishiba, Y. Oyamada, and K. Kondo, "Sparse representation using weighted graph fourier transform based on color-depth correlation for depth map compression," *2016 IEEE 5th Global Conference on Consumer Electronics*, pp. 1–2, Oct 2016.

[10] "The (new) Stanford light field archive." `http://lightfield.stanford.edu/lfs.html`. Accessed: 2017-04-18.

[11] M. Levoy, "Light fields and computational imaging," *Computer*, vol. 39, pp. 46–55, Aug 2006.

[12] 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*, vol. 31, pp. 80–90, Sep 2014.

[13] A. Sandryhaila and J. M. F. Moura, "Nearest-neighbor image model," in *2012 19th IEEE International Conference on Image Processing*, no. 3, pp. 2521–2524, IEEE, Sep 2012.

On the Use of Graph Fourier Transform for Light-Field Compression

Vitor Rosa Meireles Elias and Wallace Alves Martins

Abstract—**This work proposes the use and analyzes the viability of graph Fourier transform (GFT) for light-field compression. GFT is employed in place of discrete-cosine transform (DCT) in a simplified compression system based on high-efficiency video coding (HEVC). The effect on GFT efficiency of different implementations for prediction procedure is analyzed, as well as different methods for computing GFT given residual images. Results indicate that the prediction scheme is sensitive to the type of light field being compressed, and a preliminary method for selecting the best prediction scheme is explored. Moreover, considering multiple residual images when computing GFT, instead of only one central image, improves compression rate and makes compression more uniform across multiple views. GFT achieves reduction of up to 21.92% in number of transform coefficients when compared to DCT-based compression, while providing better or equal mean squared reconstruction error.**

Index Terms—**Signal Processing on Graphs, Graph Fourier Transform, Light Field, Compression, High Efficiency Video Coding, Discrete-Cosine Transform, Prediction.**

I. Introduction

Light field imaging is a promising technology that opens a variety of new possibilities to entertainment industries, such as photography and cinema, by capturing 4D data from a scene [1]–[7]. Light field technology is based on the 5D plenoptic function $L(x, y, z, \theta, \phi)$, which describes the amount of light L, denominated radiance, along every position (x, y, z) in space and in any direction (θ, ϕ). Theoretically, if the plenoptic function for a region of interest is known, any image associated with that region can be recreated, from every perspective. This motivates the use of light field in entertainment industries, mainly photography and cinema [1]. Other application for light fields reside in medical imaging, such as microscopy [8] and brain imaging [9]. In practice, determining the plenoptic function is unfeasible, so light field cameras capture a 4D parametrization of the plenoptic function that consists of multiple photographs of a scene. This can be done moving a digital camera in a grid of various positions and taking photographs at each position, by using an array with multiple cameras, or by adding a microlens array in front of the camera sensor [3].

As light field data consists of multiple photographs, data size may increase drastically depending on the configuration of the light-field recording setup, making the manipulation of the resulting data a challenging task [10]–[15]. The "JPEG Pleno" initiative, conducted by the JPEG standardization committee, aims at providing solutions for framework and data manipulation considering several multiview image techniques, such as light field [6]. The delivery of a complete set of tools, including framework, coding, tests, and software, is set to 2018 [6], [16]. This requires in-depth research in order to develop and improve the various tools.

Mr. Vitor R. M. Elias and Prof. Wallace A. Martins are with the Federal University of Rio de Janeiro, Rio de Janeiro, RJ, Brazil (emails: vitor.elias@smt.ufrj.br, wallace.martins@smt.ufrj.br).

The authors are grateful for the financial support provided by CNPq, CAPES, and FAPERJ, Brazilian research councils.

Digital Object Identifier: 10.14209/jcis.2018.10

The use of graphs is specially relevant when dealing with an irregular domain or any domain that is not well represented by traditional time series [17]. In the current stage of the information era, the necessity of dealing with data from enormous networks, such as social networks, sensor networks, transport networks, among many others, increases daily. Given the non-ordered nature of these networks, using graphs as an underlying domain for the associated data becomes an interesting alternative to standard analyses [18]. Data from these networks become signals on graphs and, in order to manipulate these data, tools from classic digital signal processing (DSP) are adapted to signals on graphs, yielding the emerging field of digital signal processing on graphs (DSP$_\text{G}$) [17], [19]–[23].

Two important concepts that serve as basis for a signal processing framework for signals on graphs are the definitions of shift operator and frequency domain. As an emerging field, there are no consensus regarding the proper definitions of these concepts, giving rise to many researches addressing the approach that best fits each particular application [24]. One approach is based on the spectral graph theory [25], which uses the graph Laplacian $\mathcal{L}$ as shift operator and its eigenvectors as spectrum of the graph. This approach is usually restricted to undirected graphs, for which relations between two different elements are symmetrical, i.e., an edge from element i to element j has the same value as an edge from j to i. A second approach, valid for both directed and undirected graphs, uses the adjacency matrix of the graph $\mathbf{A}$ as shift operator [19], [26], [27]. In this case, the spectrum of the graph is defined as the eigenvalues of $\mathbf{A}$. This approach is the one adopted throughout this work, as it allows the use of more general classes of graphs.

This work is an extended version of the work presented in [28], where the application of *graph Fourier transform* (GFT) was proposed and studied as an alternative to the discrete-cosine transform (DCT) in the compression of light-field data. The objective of this work is to provide an improvement for light-field compression systems based on high-efficiency video coding (HEVC) [14], [29]. In HEVC, DCT and discrete-sine transform (DST) are used as block transforms, with the objective of mapping data into a frequency-related domain where quantization (and thus compression) is more efficient. This increase in efficiency is due to the energy compaction property related to these trigonometric transforms when applied to images. It has been shown in [30], [31] that GFT is able to concentrate energy in fewer coefficients when compared to DCT, decreasing compression *distortion* when using the same number of coefficients. GFT usually depends on the original data and, thus, is not a fixed transform. Transmitting the transform basis from encoder to decoder is required, increasing transmission *rate*, and the impact of this task must be dealt with in order for GFT to be more efficient than DCT in the *rate-distortion* sense.

A. Scope and Contributions

This work begins by providing a review on both light field and DSP_G theories and an overview on how both these concepts are employed in this work. This includes: presentation of introductory concepts on both topics, motivation of the proposed approaches, notation, and database adopted throughout this work. The remaining part of this work focuses on analyzing the viability of using GFT in place of DCT under different analysis methods. We investigate forms of improving the performance of GFT by studying some of its parameters for which no consensus has been reached. The main contributions of this work are:

- Proposal and investigation of real applications for the developing field of DSP_G, given real and practical light field data.
- Performance comparison between GFT and traditional and broadly used DCT, analyzing viability of using GFT in the proposed application.
- Study of the effects of different settings for graph representation on GFT.

B. Outline

In Section II, background review on both light field and DSP_G is provided, including theory, applications, and motivation. Section III presents the proposed approach for using GFT light-field compression in an HEVC-based system. Section IV describes the entire methodology regarding database, definitions, and other concepts adopted throughout this work. Simulations and results are presented in Section V. Section VI presents a brief discussion of the results and future works. Section VII presents a conclusion for this work.

II. Light Field and DSP_G : A Review

This section reviews the main concepts related to both light fields and DSP_G. It begins by presenting light-field theory, focusing on recent implementations and how light-field data is generated. Then basic graph concepts and notations adopted in this work are presented, along with recent advances in the area.

A. Light field

Early notions of interpreting light as a field and conceiving a vector function to represent the amount of light present at (and passing through) points in space date back to the beginning of the 20th century. In 1936, Andrey Gershun introduced the

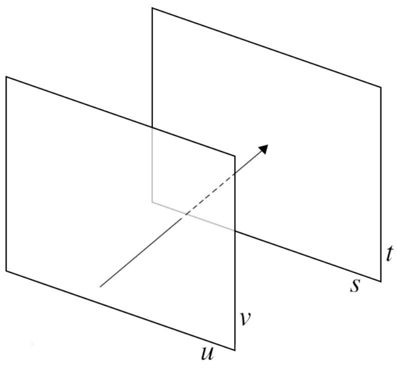


Figure 1: Planes *st* and *uv*, which serve as 4D parametrization for plenoptic function.

term *light field* [32] and an early version of the function that would later be called the *plenoptic function*. In its standard interpretation, the 5D plenoptic function $L(x, y, z, \theta, \phi)$, which is a scalar field, describes light intensity that goes through a given point in space as a function of its position and the direction toward which the light ray is headed. Light intensity is denominated radiance and is given in $^{\text{W}}/_{\text{sr}\cdot\text{m}^2}$ (watts per steradian per meter squared, i.e., power per solid angle per area). The function $L(\cdot)$ may be extended to higher dimensionality, for instance, by also considering time or wavelength. The idea of this function is to convey the complete information about a *scene*[1] associated with electromagnetic radiation. If $L(\cdot)$ is known, then every possible *view*[2] associated with a scene can be reconstructed by correctly arranging evaluations of the function for different points and directions in space, having several applications in imaging, photography, rendering, and other areas.

In practice, the plenoptic function is not available or obtainable in a feasible way. If free space is assumed, that is, the space associated with the region of interest is free of obstacles, the plenoptic function may be represented in lower dimensionality, considering a light ray sustains its radiance for different points along a given direction. The assumption of free space may be generalized to keeping the region of interest limited to the convex hull of any object. A straightforward parametrization of the plenoptic function in four dimensions is composed by two planes as shown in Figure 1. This representation of plenoptic function in four dimensions leads to current implementations of light-field-capturing devices. In devices used for capturing scenes and creating a light-field composition, the *uv* plane is taken as the *camera plane* and the *st* plane as the *focal plane*. That is, multiple light rays from the scene located at plane *st* travel along the space and hit a sensor region in plane *uv*, creating a view of the scene [1]. Common implementations are:

- array of cameras, with all cameras focused on the scene, creating a discrete version of plane *uv*;
- moving camera over a grid, capturing the scene at each point of the grid. It is actually similar to using an array of

[1]In this context, *scene* is a region of interest in space, usually containing an observable object.

[2]In the sense of a graphical projection of the scene onto a planar surface.

Figure 2: Example of light-field data, consisting of multiple views of a scene, captured by a moving camera.

cameras, but requiring a static scene. An example of light field captured by a moving camera is shown in Figure 2;
- microlens array inside a conventional digital single-lens reflex (DSLR) camera, where each microlens captures light from a different direction rendering different perspectives of the scene.

Light-field technology comes with several applications, most of them in the entertainment field. With light-field data captured by systems such as the aforementioned ones, features otherwise unfeasible become direct applications. For instance, synthetic aperture photography allows changing the focal point of a picture after it was taken. Light-field rendering allows the creation of novel views not previously captured. Light field displays may improve virtual-reality displays by using full light-field data rather than simple stereoscopic views. Light-field applications, however, are very data-intensive, since a single traditional image is now represented by a set of multiple images. Recent researches are dedicated to dealing with the high amount of data from light field [10], [11], [14].

B. *Digital signal processing on graphs*

Graphs are commonly defined as mathematical structures composed by two different sets: set $\mathcal{V} = \{v_0, v_1, \ldots, v_{N-1}\}$ composed of N vertices (also known as nodes) and set $\mathcal{E} = \{e_{00}, e_{01}, \ldots, e_{(N-1)(N-1)}\}$ of N^2 edges. Vertices are basic units and are interpreted as objects of a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, which can be used to model objects in diverse systems, e.g., points in $\mathbb{R}^2$, sensor locations in a network, social-network users, or chemical elements on a molecule, among many other applications. Edges e_{ij}, whose meaning and (possibly complex) value rely on the application of the graph, represent pairwise relations between vertices v_i and v_j, being equal to zero if there is no relation. The *neighborhood* of a vertex v_i is defined as the set of all vertices directly connected to v_i

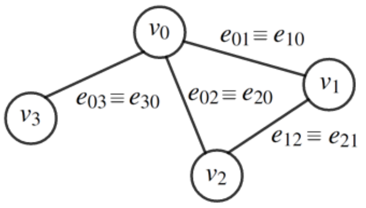


Figure 3: Example of undirected graph with $N = 4$.

by a non-zero edge. These assumptions consider that there are no multiple edges between two vertices, but there are no restrictions to self-loops, which means a vertex can be directly related to itself. In this context, relation between elements does not have a fixed definition and depends on the application. If the relation between vertices v_i and v_j is the same as the relation between vertices v_j and v_i for every pair of vertices, i.e., $e_{ij} = e_{ji}, \ \forall i, j$, the graph is denominated *undirected graph*. Otherwise, if the direction of the edge is relevant and $e_{ij} \neq e_{ji}$ for some pair of vertices, the graph is denominated *directed graph*. An example for an undirected graph is shown in Figure 3, with $N = 4$ vertices. This graph is not fully connected, since many edges are equal to zero. Another form of representing the relations between vertices is the *adjacency matrix* $\mathbf{A} \in \mathbb{C}^{N \times N}$, whose element $[\mathbf{A}]_{ij} = e_{ij}$. The graph is undirected if, and only if, $\mathbf{A}$ is symmetric. Throughout the rest of this paper, graphs will be represented by pairs $\mathcal{G} = \{\mathcal{V}, \mathbf{A}\}$.

Graphs are traditionally used as tools for data visualization and system modeling, whereas classical digital signal processing (DSP) is traditionally constructed around well-structured domains, such as time or space. Time domain is interesting for DSP as it holds properties that are particularly useful in the analysis of discrete-time signals. Consider a discrete-time finite-duration signal $s[n]$ as a function $s : \{0, 1, \ldots, N-1\} \to \mathbb{C}$ that maps instants $n \in \{0, 1, \ldots, N-1\}$ in time domain into the complex plane. Time domain is well-structured, as comparisons such as $n_1 < n_2$ and $n_1 = n_2$ are feasible for any two points n_1, n_2 within $\{0, 1, \ldots, N-1\}$, and it is a totally ordered domain. For many applications that emerge with recent advances and necessities in technology, treating signals associated with unstructured and more general domains is required. These applications are usually associated with networks, such as social, transport, sensor, and biological networks, for which representing the underlying domain with time or space would waste part of the information regarding connections among elements in the network. Graphs provide the suitable discrete domain for signals extracted from these types of network. Moreover, these applications are usually data-intensive, and graphs are a natural tool for representation of Big Data [20].

The concept of signals on graphs uses the set of N vertices $\mathcal{V}$ of a graph $\mathcal{G}$ as the domain of a dataset of N elements, equivalently to the use of N time instants $n \in \{n_0, n_1, \ldots, n_{N-1}\}$, as shown in Figure 4. The set of edges $\mathcal{E}$ of the graph is used to encode any relevant relationship between elements of the signal that could not be represented

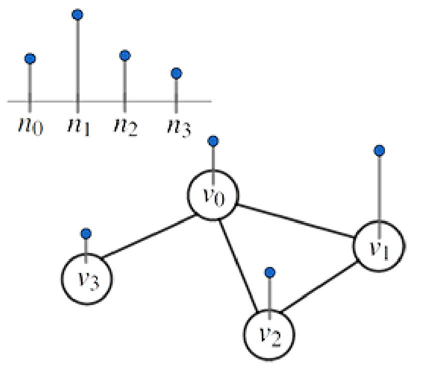


Figure 4: Relation between a signal represented in time domain and in graph domain.

in the time domain. A classic example is a sensor network that measures local temperature for N sensors distributed across several points of a country. Each location is represented by a vertex of the graph and the locally measured temperature is the signal on the vertex. Edges may be used to indicate distance between sensors, rendering an undirected graph. Another example is the measurement of user activity on a social network. Vertices would indicate each user account, for which an online-time is measured, and users are connected to each other via “following” tags, rendering a directed graph. For both cases, representing signals in time domain discards pieces of information that could be of paramount importance when processing these data.

The notation for signals on graph adopted throughout this paper is as follows: a graph signal given by $s : \mathcal{V} \rightarrow \mathbb{C}$ is referred to as a vector $\mathbf{s}$. The n-th entry of vector $\mathbf{s}$ is $s_n = s[v_n]$, with $v_n \in \mathcal{V}$.

Once graph domain and the definition of a signal over this domain are formally stated, one can build tools to process signals on graphs, which lead to two major approaches developed in the last years. The first approach is based on graph spectral theory [25] and on the graph Laplacian, being restricted to undirected graphs with non-negative edge values. This approach has received great attention and much effort was put into developing tools with these concepts [17]. Tools for DSP_G are mostly translations from already-consolidated classical DSP tools, which was mostly exploited by the second approach proposed by Sandryhaila and Moura [19], [20], [26], whose concepts are adopted and reviewed in the following definitions.

The first and most fundamental tool translated from classical DSP is the *unit-delay* or *unit-shift* operator, denoted as $\mathcal{T}^{-1}$, which consists of an essential block in filter design. In DSP, when a unit shift $\mathcal{T}^{-1}$ is applied to a length-N discrete-time signal $s[n]$, the signal is shifted in time resulting in a signal

$$\tilde{s}[n] = \mathcal{T}^{-1}\{s[n]\} = s[(n-1) \bmod N]. \tag{1}$$

The unit-shift operator $\mathcal{T}^{-1}$ is a linear transformation, implying that it can be associated with a matrix. Indeed, when

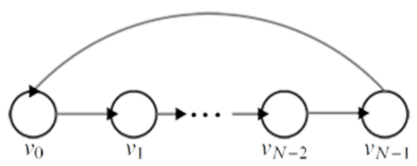


Figure 5: Cyclic graph: generalization of discrete-time domain.

using vector notation, one can rewrite Equation (1) as

$$\begin{bmatrix} \tilde{s}[0] \\ \tilde{s}[1] \\ \vdots \\ \tilde{s}[N-1] \end{bmatrix} = \underbrace{\begin{bmatrix} & & & 1 \\ 1 & & & \\ & \ddots & & \\ & & 1 & \end{bmatrix}}_{=\mathbf{C}} \cdot \begin{bmatrix} s[0] \\ s[1] \\ \vdots \\ s[N-1] \end{bmatrix}. \tag{2}$$

One can interpret the relation in Equation (2) within a graph framework. Indeed, consider the directed cyclic graph in Figure 5. Given all edges equal to 1, this graph can be interpreted as a graph generalization of the discrete-time domain, where each vertex v_n represents a time instant $n \in \{0, 1, \ldots, N-1\}$. The adjacency matrix of this graph is the *cyclic-shift matrix* $\mathbf{C}$ appearing in Equation (2).

One can bring these ideias to the graph domain by considering a graph $\mathcal{G} = \{\mathcal{V}, \mathbf{A}\}$ as the underlying structure for a signal $\mathbf{s}$, and by identifying the *graph-shift* operator with the graph adjacency matrix $\mathbf{A}$. That is, a shifted signal $\tilde{\mathbf{s}}$ on a graph is given by

$$\tilde{\mathbf{s}} = \mathbf{A}\mathbf{s}. \tag{3}$$

This definition for graph shift means that shifting a signal on graph domain is equivalent to replacing each signal sample s_n by a linear combination, given by the n-th row of $\mathbf{A}$, of its neighborhood. This approach is not restricted to undirected graphs, allowing the use of directed graphs with complex-valued edges. A straightforward property of this definition is that it generalizes the unit-shift operator from classical DSP.

Given a formal definition for unit-shift in the graph domain, defining filters is the next natural step and it is performed by translating filtering concepts from classical DSP. In discrete-time domain, the output from a finite-duration impulse response (FIR) filter with length P is defined by the linear combination of its P most recent inputs, i.e.,

$$\begin{aligned} \tilde{s}[n] &= h_0 s[n] + h_1 s[n-1] + \cdots + h_{P-1} s[n-P+1], \\ &= \sum_{p=0}^{P-1} h_p \mathcal{T}^{-p}\{s[n]\}, \end{aligned} \tag{4}$$

where the time-invariant coefficients $h_0, h_1, \ldots, h_{P-1}$ define the impulse response of the filter and each term $s[n-p]$ results from shifting $s[n]$ with a shift operator $\mathcal{T}^{-p}$. For a signal with finite duration N, applying an FIR causal filter of length $P \leq N$, that is, $h_p = 0$ for $p < 0$ and $p \geq P$, induces the following circular convolution

$$\begin{bmatrix} \bar{s}[0] \\ \bar{s}[1] \\ \vdots \\ \bar{s}[N-1] \end{bmatrix} = \underbrace{\begin{bmatrix} h_0 & h_{N-1} & \cdots & h_1 \\ h_1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & h_{N-1} \\ h_{N-1} & \cdots & h_1 & h_0 \end{bmatrix}}_{=\mathbf{H}(\mathbf{C})=\sum_{p=0}^{P-1} h_p \mathbf{C}^p} \begin{bmatrix} s[0] \\ s[1] \\ \vdots \\ s[N-1] \end{bmatrix}, \quad (5)$$

which shows that the filter is equivalent to a length-P polynomial over the cyclic-shift matrix $\mathbf{C}$. Analogously, the linear, *shift*-invariant *graph filter* is defined as a polynomial over the adjacency matrix $\mathbf{A}$, i.e.,

$$\mathbf{H}(\mathbf{A}) = \sum_{p=0}^{P-1} h_p \mathbf{A}^p. \quad (6)$$

Once signals, shift, and filters on graphs are defined, concepts of *spectral decomposition* and *Fourier transform* can be extended to graph domain. For a signal space $\mathcal{S}$, spectral decomposition of $\mathcal{S}$ is the identification of W filtering-invariant subspaces $\mathcal{S}_0, \ldots, \mathcal{S}_{W-1}$ of $\mathcal{S}$. Being invariant to filtering means that, for a signal $\mathbf{s}_w \in \mathcal{S}_w$, the output of filtering this signal is $\bar{\mathbf{s}}_w = \mathbf{H}(\mathbf{A})\mathbf{s}_w \in \mathcal{S}_w$. The spectral decomposition is uniquely determined for every signal $\mathbf{s} \in \mathcal{S}$ if, and only if:

- $\mathcal{S}_w \cap \mathcal{S}_r = \{0\},\ w \neq r$;
- $\dim(\mathcal{S}_0) + \cdots + \dim(\mathcal{S}_{W-1}) = \dim(\mathcal{S}) = N$;
- Each $\mathcal{S}_w$ is irreducible to smaller subspaces,

and, in this case,

$$\mathcal{S} = \mathcal{S}_0 \oplus \mathcal{S}_1 \oplus \cdots \oplus \mathcal{S}_{W-1}. \quad (7)$$

Given $\mathcal{S}$ as defined in Equation (7), satisfying the above conditions, any signal $\mathbf{s} \in \mathcal{S}$ is univocally represented as

$$\mathbf{s} = \mathbf{s}_0 + \ldots + \mathbf{s}_{W-1}. \quad (8)$$

The diagonalization of the adjacency matrix $\mathbf{A}$ leads to a spectral decomposition of the signal space $\mathcal{S}$ on the graph domain. Nonetheless, given the arbitrary nature of $\mathbf{A}$, as allowed in this DSP$_\mathrm{G}$ approach, it is not always diagonalizable. It is shown in [19] that the Jordan decomposition $\mathbf{A} = \mathbf{V}\mathbf{J}\mathbf{V}^{-1}$ is used to conduct spectral decomposition of $\mathcal{S}$ on graphs. $\mathbf{J}$ is the Jordan normal form and $\mathbf{V}$ is the matrix whose columns are the generalized eigenvectors of $\mathbf{A}$, which are the bases of the subspaces of $\mathcal{S}$. Hence, Equation (8) can be written as

$$\mathbf{s} = \mathbf{V}\hat{\mathbf{s}}, \quad (9)$$

where $\hat{\mathbf{s}}$ is the vector of coefficients that expand $\mathbf{s}$ into the subspaces of $\mathcal{S}$. The union of these subspaces is the *graph Fourier basis*. The *graph Fourier transform* (GFT), which provides the coefficients of the expansion of a signal over the *graph Fourier basis*, is defined as

$$\mathbf{F} = \mathbf{V}^{-1}, \quad (10)$$

such that $\hat{\mathbf{s}} = \mathbf{F}\mathbf{s}$. The *inverse graph Fourier transform* (IGFT) is given by

$$\mathbf{F}^{-1} = \mathbf{V}. \quad (11)$$

If the graph is undirected, $\mathbf{A}$ is a symmetric matrix and it is diagonalizable. The graph Fourier transform is then obtainable from the eigenvectors of $\mathbf{A}$. In this case, the eigenvectors are orthogonal and $\mathbf{V}^{-1} = \mathbf{V}^\mathrm{T}$, which makes computation of the transform matrix $\mathbf{F}$ less intensive.

III. Proposed approach to light field compression

The application of HEVC-based methods for compression of light-field data has been intensively researched over the past years [14], [29], [33], [34]. HEVC presents a complex scheme composed by intra-frame and inter-frame prediction, motion estimation and compensation, transformation, quantization, coding, and other procedures, for which several configurations are available. These procedures are applied to *coding tree units*, which are blocks of up to 64×64 pixels into which video frames are divided. Notable procedures considered in this work are inter-frame prediction, transformation, and quantization, whose general concepts are explained below.

- **Inter-frame prediction**: When encoding a block of pixels of the current frame, the algorithm searches for a similar block, denominated *reference block*, from the previously encoded frame. Instead of encoding the raw values of pixels of the current block, the algorithm encodes only the difference between current and reference blocks. This difference is denominated *prediction residual*. The prediction procedure may be a complex process, using, for example, algorithms to estimate and compensate movement of blocks between different frames. Residual blocks should have less entropy than raw blocks, which makes compression in transformation, quantization, and coding stages more efficient. It must be noted that, in order to make inter-frame prediction possible, at least one frame that was previously encoded must have been encoded without inter-frame prediction. This frame is referred to as *intra frame*.
- **Transformation**: HEVC applies two-dimensional discrete-sine transform (DST) and, mostly, discrete-cosine transform (DCT) to residual blocks. Transformation is used to map data from residual blocks into a frequency-related domain, where energy concentration in lower frequencies can be exploited during compression. The output of transformation stage is a transform-coefficient block. Transform coefficients are real values that indicate how much each frequency component contributes to build the image in the original domain, in this case, the residual block.
- **Quantization**: Quantization maps coefficient values that may assume any value from a large, possibly continuous, set into a smaller set, allowing application of coding procedures otherwise unfeasible. The stronger the quantization, the fewer bits will be necessary to encode transform coefficients, thus reducing the associated *rate*.

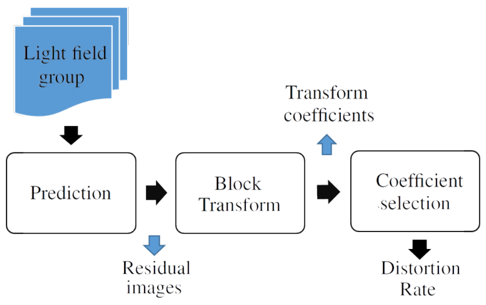


Figure 6: Block diagram describing the simplified compression process adopted throughout this work.

> Quantization is a lossy process, i.e., information is permanently lost once coefficients are quantized. The loss of information is called *distortion*, for which several metrics are available. Compression processes must consider the trade-off between rate and distortion.

This work proposes and analyzes the viability of using GFT in place of DCT in HEVC-based light-field encoders, while exploiting the similarity among light-field images. The use of GFT within data compression context, and specifically image compression, is not new. The competence of GFT for concentrating information in few transform coefficients in a competitive manner when compared to other transforms is known and has been approached in other works [30], [31], [35]. Notwithstanding GFT inducing relatively high energy concentration, the transform and its inverse IGFT depend on the adjacency matrix **A**, which has no fixed structure and depends on the application and on the data. The impact of storing or transmitting **A** or the transform matrix **F** must be considered during compression. The method proposed in this work aims at reducing the impact of the extra data related to graph structure by exploring the redundancy that exists among images near to each other in light fields.

IV. Methodology

In order to assess the performance of using GFT for light-field compression, a simplified compression process is defined, as presented in Figure 6, which is detailed in the next subsections. A database composed by 7 light fields is used. Three of them, namely *Humvee*, *Knights*, and *Tarot*, are obtained from the *Stanford Light Field Archive* [36] and some sample views are shown in Figure 7. These light fields are captured from real scenes using a moving camera on a rectangular grid with 16×16 positions, yielding 256 total images for each light field. The other four light fields are generated synthetically, obtained from the *HCI 4D Light Field Dataset* [37], [38]; sample views for *boxes*, *cotton*, *dino*, and *sideboard* are presented in Figure 8. For these light fields, views are captured over a grid of 9×9 positions, for a total of 81 images for each light field. Database information is summarized in Table I. Only the luminance component from

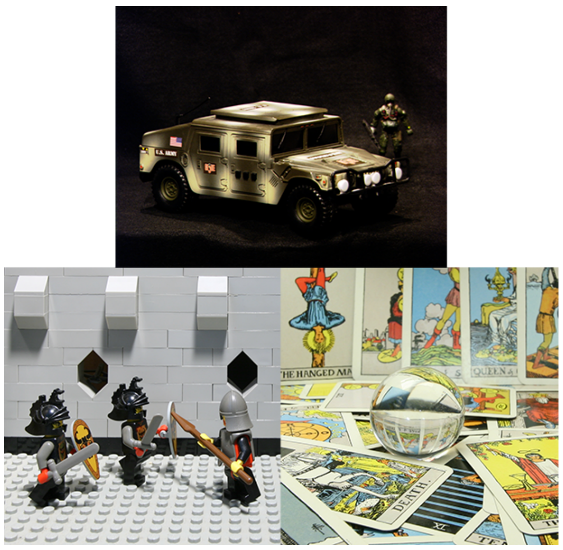


Figure 7: Sample views from light fields captured from real scenes. *Humvee* (top), *Knights* (bottom left), and *Tarot* (bottom right).

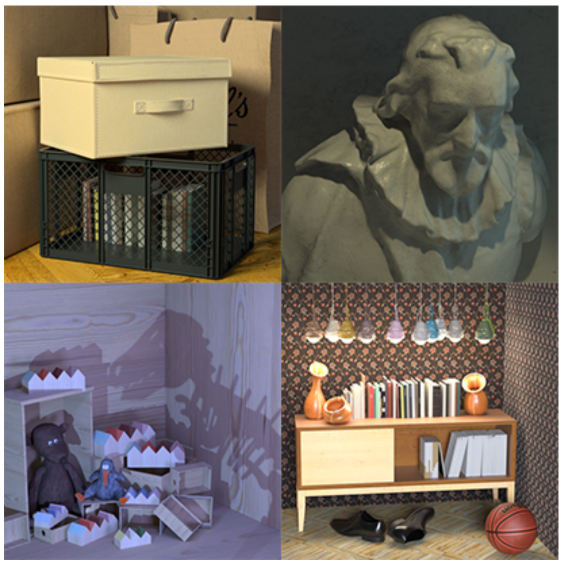


Figure 8: Sample views from light fields captured from synthetic scenes. *Boxes* (top left), *Cotton* (top right), *Dino* (bottom left), and *Sideboard* (bottom right).

these light fields is used throughout this work, despite the fact that RGB versions are depicted here.

A. Prediction

The input of video codecs is a stream of frames ordered according to their time stamps. It is reasonable to assume that similarity between frames decays when two frames are selected further apart in time if compared to similarity between two consecutive frames. Thus, prediction for video streams can be implemented by selecting the frame that comes right before the current frame. It is worth noting that complex prediction schemes are not usually limited to only one frame.

Table I: Database information

Light field	Scene	View resolution [pixels]	Grid size
Humvee	Real	640×512	16×16
Knights	Real	1024×1024	16×16
Tarot	Real	1024×1024	16×16
Boxes	Synthetic	512×512	9×9
Cotton	Synthetic	512×512	9×9
Dino	Synthetic	512×512	9×9
Sideboard	Synthetic	512×512	9×9

For light fields, a prediction order is not straightforward. It is expected that views close to each other should be more similar. However, there is no consensus on how to determine the optimal selection of views or the boundaries for spatial neighborhood used for prediction in light fields. Considering the light field *humvee* as example, with a grid of 16×16 positions, three prediction schemes are considered in this work:

- **Rows**: Prediction is performed over each row with 1×16 images, independently from other rows. The first image from each line is assumed to be an *intra image*, i.e., no prediction is used when coding this image. For the remaining 15 images from each line, prediction residuals are calculated. A simple prediction scheme is adopted. The prediction image $\mathbf{I}_k^{\mathrm{p}}$ for the k-th image $\mathbf{I}_k$ in a light field row, where $k \in \{2, 3, \ldots, K\}$ and $K = 16$ in this example, is given by $\mathbf{I}_k^{\mathrm{p}} = \mathbf{I}_{k-1}$. That is, each image is assumed to be equal to the previous image in the line, given the high similarity among adjacent views in light fields. Finally, the residual image $\mathbf{R}_k$ is computed as the difference between current image and its prediction, i.e., $\mathbf{R}_k = \mathbf{I}_k - \mathbf{I}_k^{\mathrm{p}} = \mathbf{I}_k - \mathbf{I}_{k-1}$. A total of $K-1$ residual images are computed for each row.
- **Columns**: Prediction using columns is similar to prediction using rows. Columns with 16×1 images are treated independently, and the first image from each column is an *intra image*, whereas the remaining are *inter images*. Computation of residual images $\mathbf{R}_k$ is analogous to the one described for **rows**.
- **Blocks**: When using a block scheme to perform prediction, a 3×3 block of views is selected. The central view of the block is the *intra image* and the prediction image for every *inter image* is the central view. In other words, a block is composed by $K = 9$ views on a 3×3 grid. The central image $\mathbf{I}_c$ is *intra*-encoded, for some $c \in \{1, 2, \ldots, K\}$. Per group, $K - 1$ residual images are computed as $\mathbf{R}_k = \mathbf{I}_k - \mathbf{I}_c$, for $k \in \{1, 2, \ldots, K\}$ and $k \neq c$.

Given one of the prediction schemes described, the set of views selected for prediction procedure, i.e, views from a row, column, or block, will be referred to as *prediction group*.

B. Transformation

As stated, block transform is used to map data from residual image blocks into a frequency-related domain. This allows better compression of the data. HEVC uses DCT for residual blocks from size 4×4 up to 32×32, and DST for some cases of 4×4 blocks. In this work, GFT is used to transform blocks of size 32×32 and results are compared to those of DCT. If smaller blocks, such as 4×4 or 8×8, are used, it is expected that blocks at the same position for different residual images should have low correlation with each other, given the parallax between adjacent views. For large 32×32 blocks, the impact of parallax is reduced. High correlation among blocks in the same position from several views in a prediction group is beneficial for the proposed compression scheme, as will be further explained in this section.

Up to this point, images are treated as sets of pixels in 2D space. In order to make the use of GFT possible, the signal associated with a residual block must be represented as a signal on a graph, previously defined as a vector $\mathbf{s}$, such that the n-th entry s_n is a function of the vertex $v_n \in \mathcal{V}$. Let the signal associated with a pixel from an $M_1 \times M_2$ residual block be $r : \mathbb{I}^{M1 \times M2} \to \mathbb{R}$, where $\mathbb{I}^{M1 \times M2}$ represents the set of integer indexes for the positions of pixels on the $M_1 \times M_2$ block. That is, for each position on the $M_1 \times M_2$ block, a residual-related real value is assigned. The signal on graph is defined such that $s[M_1(m_2 - 1) + m_1] = r[(m_1, m2)]$, for $(m_1, m_2) \in \mathbb{I}^{M_1 \times M_2}$. That is, the graph signal $\mathbf{s}$ is defined as a column vector formed by stacking the columns of the residual block.

Let a residual block $\mathbf{B}_{k,t}$, $k \in \{1, \ldots, K\}$, $t \in \{1, \ldots, T\}$, be the $M_1 \times M_2$ block from the k-th residual image $\mathbf{R}_k$ (in a prediction group with $K-1$ residual images) that was divided into T blocks. The graph signal associated with this block is $\mathbf{s}_{k,t}$. The corresponding adjacency matrix is denoted by $\mathbf{A}_{k,t}$ and the GFT matrix by $\mathbf{F}_{k,t}$. Note that the transform matrix, and consequently its inverse, depend on the signal, unlike the DCT, which is the same for every $M_1 \times M_2$ block. The first consideration adopted in this work in order to reduce the impact of transmitting the transform matrix is to build a sparse adjacency matrix and transmit $\mathbf{A}_{k,t}$ instead of $\mathbf{F}_{k,t}$. The adjacency matrix $\mathbf{A}_{k,t}$ is built according to the nearest-neighbor (NN) image model [39], which is shown to offer an efficient image representation whilst providing a sparse and fixed graph structure. This model defines an image as a 2D nearest-neighbor graph. An NN graph is a graph for which a vertex v_i is connected to v_j if, and only if, the distance $d(v_i, v_j)$ is minimum among the distance between v_i and all other vertices. For a regular structure like an image, the minimum distance exists for more than one pixel, as depicted in Figure 9. Using NN image model implies that each vertex of the graph will have at most four non-zero edges, and pixels at the corner, border, or interior of the block have different number of edges. The model also assumes that an image is a 2D NN graph constructed as a Cartesian product of two 1D NN graphs. A 1D NN graph is a possibly-directed line graph similar to the one presented in Figure 5, apart from the loop edge. This generates a structure where multiple edges assume the same value, indicated by coefficients $a_0, \ldots, a_{M_1-2}$ and $b_0, \ldots, b_{M_2-2}$ in Figure 9. As a result, considering an $M_1 \times M_2$ residual block $\mathbf{B}_{k,t}$, the corresponding adjacency matrix $\mathbf{A}_{k,t} \in \mathbb{R}^{N \times N}$, $N = M_1 M_2$, has at most $(M_1 - 1) + (M_2 - 1)$ unique non-zero coefficients. For blocks of size 32×32, this means 62 unique non-zero coefficients out of 1024 entires of $\mathbf{A}_{k,t}$. The coefficients $a_0, \ldots, a_{M_1-2}$ and $b_0, \ldots, b_{M_2-2}$ are defined so as

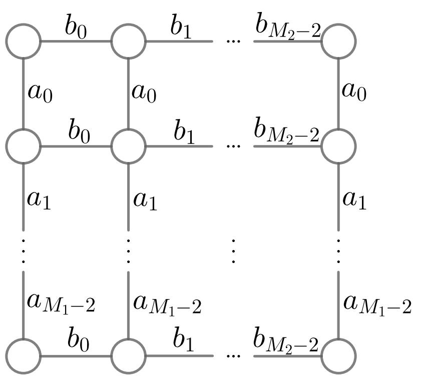


Figure 9: Relation edges according to the NN image model. Edges connect only pixels at minimum distance among all pixels.

to minimize the ℓ_2 distortion introduced by the shift operation, i.e., $||\mathbf{A}_{k,t}\mathbf{s}_{k,t} - \mathbf{s}_{k,t}||_2$. As described in [39], this minimization is solved as an overdetermined least-squares problem. This entire reasoning eventually implies that the adjacency matrix $\mathbf{A}_{k,t}$ is transmitted in place of the graph Fourier transform matrix $\mathbf{F}_{k,t}$. While this saves bandwidth, it adds complexity to the decoder, as the eigenvectors of $\mathbf{A}_{k,t}$ must be computed. Note that $\mathbf{A}_{k,t}$ is symmetric and, thus, diagonalizable. Finally, it is worth pointing out that other schemes rather than the NN image model could have been employed as well, which might induce different performances; however, the NN model proved viable, as corroborated by the results achieved in this work (see Section V).

The second consideration employed to reduce the impact of $\mathbf{A}_{k,t}$, besides forcing sparsity and fixed structure via NN image model, is to exploit the redundancy among the many views in the light field in order to avoid transmitting $\mathbf{A}_{k,t}$ with every single block. Considering that every view is equally divided into T blocks, only one $\mathbf{A}_{t_0}$ is transmitted for a given block position t_0 across the entire prediction group. Figure 10 shows an example of block position t_0 across views from a prediction group. This consideration assumes that blocks in the same position are highly correlated among several residual images. In this work, two similar methods for computing matrix $\mathbf{A}_{t_0}$ are considered. The first is using only adjacency matrices associated with one of the $K-1$ residual images $\mathbf{R}_k$. For *rows* or *columns* prediction schemes, using the central residual image (for example $\mathbf{R}_8$ when $K = 16$) is an intuitive choice, since other views are symmetrically similar to it. For *blocks* prediction scheme, there is no defined choice. The second method is to use multiple residual images $\mathbf{R}_k$ and compute the coefficients of $\mathbf{A}_{t_0}$ by minimizing $\sum_{k=k_1}^{k_2} ||\mathbf{A}_{k,t_0}\mathbf{s}_{k,t_0} - \mathbf{s}_{k,t_0}||_2$, $1 \leq k_1 < k_2 \leq K-1$. That is, the distortion introduced by the shift operator is minimized jointly for multiple, possibly all, residual images in a prediction group. For both methods, using an adjacency matrix which is not specifically computed for a given block may degrade the efficiency of the GFT, but the impact of transmitting the matrix is slightly reduced.

Once the adjacency matrix is computed, the GFT matrix for each block position is given by $\mathbf{F}_t$, whose columns are the eigenvectors of $\mathbf{A}_t$ —the reader should keep in mind that the index k can now be dropped from $\mathbf{A}_{k,t}$ and $\mathbf{F}_{k,t}$ since it is assumed that adjacency and transform matrices do not depend on the residual image, given that only one matrix is considered for a given block position across the entire prediction group. The transform coefficients for each block from residual images in the prediction group are computed as $\hat{\mathbf{s}}_{k,t} = \mathbf{F}_t\mathbf{s}_{k,t}$, where $\mathbf{s}_{k,t}$ is the graph signal corresponding to each block.

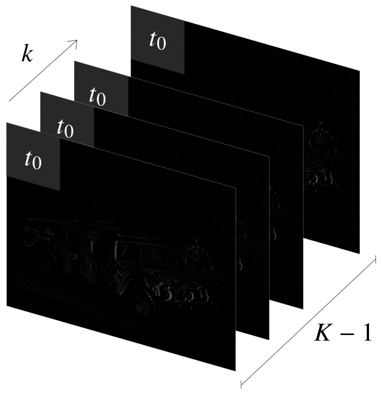


Figure 10: Representation of a block position t_0 for residual images from a prediction group.

C. Coefficient selection

A heuristic technique is adopted to assess the performance of GFT against DCT for light-field compression when employed in an HEVC-based compression system. The IGFT is given by the transpose of $\mathbf{F}_t$, since eigenvectors from $\mathbf{A}_t$ are orthogonal. If IGFT is applied to transform coefficients $\hat{\mathbf{s}}_{k,t}$, the signal $\mathbf{s}_{k,t}$ is perfectly recovered. In practical applications, compression occurs when transform coefficients are quantized, resulting also in loss of information. In this work, a simplified compression process is conducted by setting Q smallest transform coefficients to zero, resulting in compressed transform coefficients $\hat{\mathbf{s}}^Q_{k,t}$. When IGFT is applied to these coefficients, the signal $\mathbf{s}^Q_{k,t}$, which is recovered by inverse transform, is an approximation of the original signal $\mathbf{s}_{k,t}$. A compressed version $\mathbf{B}^{\mathrm{GFT}}_{k,t}$ of the original block $\mathbf{B}_{k,t}$ can be constructed from the signal recovered. For the case of DCT, the 2D DCT is applied directly to block $\mathbf{B}_{k,t}$ and by setting the smallest coefficients from the transform block to zero, a compressed block $\mathbf{B}^{\mathrm{DCT}}_{k,t}$ is recovered via inverse discrete-cosine transform (IDCT).

V. Simulations and results

Simulations were conducted in order to compare GFT against DCT when employed in the proposed compression system. The basic concept underlying all simulations presented in the next subsections is to set GFT coefficients to zero as much as possible while still recovering blocks with less distortion when compared to a specific DCT compression. The number of compressed DCT coefficients is fixed at $Q^{\mathrm{DCT}} = 924$, i.e., only the 100 largest out of 1024 coefficients are kept and DCT

Table II: Simulation results for transform-setup analysis

	Central residual		Part of group		Entire group	
Light field	**Reduction [%]**	**Standard deviation of Q**	**Reduction [%]**	**Standard deviation of Q**	**Reduction [%]**	**Standard deviation of Q**
Humvee	8.97	6.97	9.65	4.63	8.63	1.82
Knights	13.40	11.04	16.67	8.57	17.53	1.93
Tarot	-3.91	3.50	-0.65	1.96	-0.29	0.83
Boxes	0.22	4.56	6.57	2.45	7.76	1.42
Cotton	5.90	3.05	6.28	1.94	6.07	1.00
Dino	21.22	5.14	21.92	3.61	21.18	1.92
Sideboard	-3.89	2.67	-2.29	1.23	-2.04	0.86

is fixed at approximately 10:1 compression ratio. Distortion D^{DCT} is evaluated for DCT. For each residual image, the simulation searches for the largest number of compressed GFT coefficients Q^{GFT} for which the corresponding distortion D^{GFT} is still smaller or equal to D^{DCT}. It is important to note that both Q^{DCT} and Q^{GFT} are set for an entire residual image and, thus, every block in each residual image will be represented by the same number of coefficients. The figure of merit used to characterize distortion is the *mean squared error* (MSE) between compressed and original residual images. For some simulations, the *structural similarity* (SSIM) index [40] is also considered as figure of merit for distortion. While MSE represents an indication of absolute error between images, the SSIM index provides information related to changes in structural information between images.

Different simulation setups are considered given the options described in Section IV. Three prediction methods were proposed, namely: *rows*, *columns*, and *blocks*. Moreover, two methods for building the adjacency matrix are considered. The first uses only one reference residual image, whereas the second uses multiple residual images when computing the coefficients of $\mathbf{A}_t$. The effects of these different setups are analyzed in this section. The database presented in Section IV and detailed in Table I is used.

A. Transform-setup analysis

As presented in Section IV-B, the coefficients of $\mathbf{A}_t$ may be computed either for a single reference residual image or jointly for multiple residual images. For this simulation, using the *rows* prediction scheme, three setups are considered for transform computation:

- Using only one central residual image as reference. The 8-th residual image $\mathbf{R}_8$ for real light fields, where $K = 16$ images per line, and the 5-th residual image $\mathbf{R}_5$ for synthetic light fields, with $K = 9$ images per line;
- Using part of the prediction group. Residual images from $\mathbf{R}_5$ to $\mathbf{R}_{10}$ for real light fields and from $\mathbf{R}_3$ to $\mathbf{R}_6$ for synthetic light fields;
- Using all residual images from the prediction group.

Table II shows the results obtained for simulations considering these three setups. Results show the *reduction* in number of coefficients used by GFT when compared to DCT for the entire light field, so that GFT is still able to yield better or equal distortion for every residual image. Reduction values for the total number of coefficients (#) for each light field are

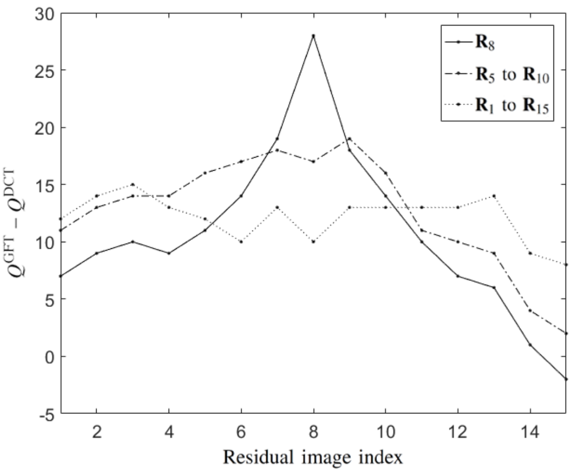


Figure 11: Number of compressed coefficients Q according to residual image position for the three proposed methods for computing $\mathbf{A}_t$.

computed as

$$\text{Reduction} = \frac{\#\text{ DCT coefficients} - \#\text{ GFT coefficients}}{\#\text{ DCT coefficients}}. \quad (12)$$

It is worth highlighting that the number of coefficients associated with the adjacency matrices is included in # GFT coefficients and, thus, the impact of transmitting $\mathbf{A}_t$ is considered. GFT shows slight improvement over DCT for most cases, yielding up to 21.92% of reduction in number of coefficients. The analysis shows that using multiple residual images when building $\mathbf{A}_t$ improved the results for all cases when compared to results obtained using only one residual image as reference. This result can be observed in Table II by considering each light field independently, which is represented by each row. For each light field, an increasing trend in the reduction value can be noted when going from "Central residual" to "Entire group" sections, with few exceptions, indicating the overall improvement when using multiple residual images.

A relevant analysis given different transform setups is to observe the standard deviation of the number of coefficients used by the GFT across the residual images. The standard deviation of Q^{GFT} is estimated for each light field, using the number of compressed GFT coefficients Q^{GFT} from each residual image as sample for the standard deviation estimator. From Table II, it is notable that using the entire prediction group reduces the standard deviation of Q^{GFT}. When GFT is

Table III: Results for simulation using SSIM index and $\mathbf{A}_t$ computed from all residual images

	Humvee	Knights	Tarot	Boxes	Cotton	Dino	Sideboard
Reduction [%]	4.22	10.56	1.15	2.38	-5.36	10.74	-1.14
Standard deviation of Q	1.50	1.83	0.99	1.06	1.15	1.73	1.33

Table IV: Simulation results for prediction-setup analysis

	Rows		Columns		Blocks	
Light field	**Reduction [%]**	**Standard deviation of Q**	**Reduction [%]**	**Standard deviation of Q**	**Reduction [%]**	**Standard deviation of Q**
Humvee	8.63	1.82	-2.80	4.06	3.15	5.52
Knights	17.53	1.93	11.50	2.24	16.09	1.86
Tarot	-0.29	0.83	-8.42	0.81	-5.55	3.07
Boxes	7.76	1.42	11.00	1.18	7.38	1.81
Cotton	6.07	1.00	6.10	0.90	6.18	3.00
Dino	21.18	1.92	15.22	1.24	15.53	5.00
Sideboard	-2.04	0.86	2.10	1.84	-0.25	2.72

built using only the central residual image, its efficiency is high for the central residual image, but decays as residual images get further apart form the central reference. This is expected, since correlation is reduced and the impact of using a single transform matrix is increased, requiring more coefficients. Constructing the transform while considering multiple images reduces the efficiency decay across the prediction group. This effect is depicted in Figure 11, where the difference $\Delta Q = Q^{\text{GFT}} - Q^{\text{DCT}}$ in number of compressed coefficients for one row of the *humvee* light field is presented. In this case, the coefficients of $\mathbf{A}_t$ are not considered. The three proposed transform setups are considered. The peak for ΔQ at $\mathbf{R}_8$ is notable when this residual image is the only one used for transform computation. When using all residual images, this effect is no longer present, allowing for a more uniform compression across all images.

This simulation was replicated using SSIM as metric when searching for Q^{GFT}. Only the transform setup based on all residual images for the construction of $\mathbf{A}_t$ was used, considering it achieved the best results in the previous simulation. Results are presented in Table III. Values for reduction in number of coefficients are lower than the ones obtained when using MSE, but GFT is still competitive when compared to DCT. Moreover, small values for standard deviation are achieved, as expected.

B. Prediction-setup analysis

In this simulation, the three proposd prediction methods, namely *rows*, *columns*, and *blocks*, are tested. The transform matrix is built using all residual images from each group when computing the matrix coefficients. Results are shown in Table IV. For real light fields, using the *rows* prediction scheme yields the best results, followed by *blocks*, which increases the standard deviation of Q^{GFT} across residual images. For synthetic light fields, the discrepancy in results among different methods is reduced and the efficiency of *columns* prediction scheme slightly increases.

These results indicate that different prediction methods may be better suitable for some specific type of light field. Video encoders usually work with several possible configurations for each processing stage. This opens the possibility of searching for the best prediction method when compressing light field images in a more complex system. An analysis of how the

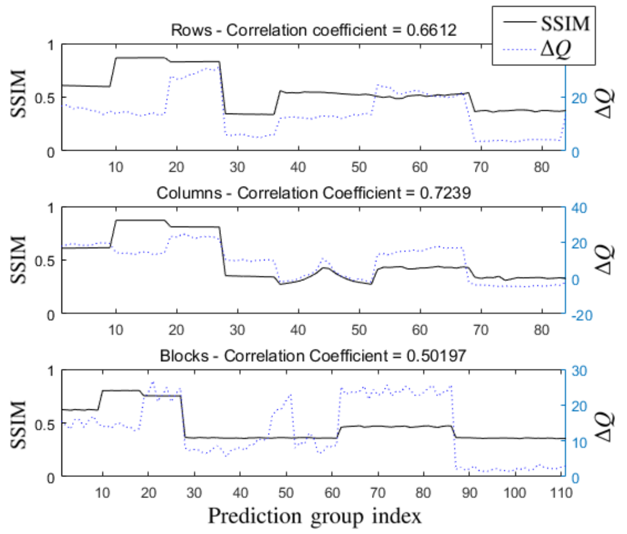


Figure 12: Analysis of the correlation between average similarity in a prediction group and the resulting efficiency of using that group for light-field compression.

similarity between images in a prediction group affects the compression efficiency in that group was conducted. For each light field, prediction groups based on the three proposed methods were constructed. For each group, the SSIM index is computed for every pairwise combination of residual images in that group and the average SSIM index value is computed. That is, for each prediction group, the corresponding average structural similarity is computed. Figure 12 shows the average SSIM results for every group for all light fields in the available database, along with the average reduction in number of coefficients used per group. This simulation is conducted for the three prediction methods. Results indicate high correlation between intra-group similarity and compression efficiency. In a more complex compression system, similarity could be used as a metric for the selection of the best prediction method.

C. Transform coding gain

The transform coding gain is a criterion commonly used in order to assess the effectiveness of a transform, by comparing the transform quantization against direct quantization in the

Table V: Transform coding gain for GFT and DCT computed for each light field, considering independent transforms $\mathbf{A}_t$

	Humvee	Knights	Tarot	Boxes	Cotton	Dino	Sideboard
G_{GFT}	9.24	28.13	19.20	3.40	2.05	5.04	5.18
G_{DCT}	5.12	27.53	21.47	2.91	1.90	4.07	4.24

Table VI: Transform coding gain for GFT and DCT computed for each light field, considering all $\mathbf{A}_t$ as a single transform

	Humvee	Knights	Tarot	Boxes	Cotton	Dino	Sideboard
G_{GFT}	5.45	8.04	7.58	1.91	1.66	2.59	2.25
G_{DCT}	5.24	8.06	9.79	1.98	1.80	2.67	2.40

original domain [41]. For orthogonal block transforms, which is the case for both GFT and DCT, and assuming high-rate, i.e., every coefficient contributes equally to distortion after optimal bit allocation, the transform coding gain is given by

$$G_{\text{T}} = \frac{\frac{1}{N}\sum_i \sigma_i^2}{\sqrt[N]{\prod_i \sigma_i^2}}, \tag{13}$$

where σ_i^2 is the variance of the i-th transform coefficient across all blocks. The transform coding gain is the ratio between arithmetic and geometric means of coefficient variances. When estimating the transform coding gain from data from a light field, it must be considered that GFT is not a single transform, since it was defined as a data-dependent transform. In order to compute the transform coding gain G_{GFT} associated with GFT, blocks are treated according to their position t, for which a single $\mathbf{A}_t$ is defined. That is, given a prediction group, an independent $G_{\text{GFT},t}$ is computed for each block position, since the transform $\mathbf{F}_t$ is restricted to that block position in that prediction group. For each light field, the final gain G_{GFT} is given by the average gain across all block positions for all prediction groups. For DCT, the transform coding gain G_{DCT} is computed in the same way as G_{GFT} to make comparison possible. Results for the estimation of transform coding gain, using *rows* prediction method and using all residual images for computing $\mathbf{A}_t$, are presented in Table V. Transform coding gain shows better efficiency for GFT when compared to DCT for all light fields but one (*Tarot*). For some block positions from *Humvee* light field, $G_{\text{GFT},t}$ could not be computed due to zero variance encountered in some coefficients, resulting in zero geometric mean. For *Humvee*, the dynamic range of $G_{\text{GFT},t}$ was set to 10 by limiting the maximum value.

Table VI shows results for transform coding gain if the entire light-field data is considered at once, i.e., assuming that GFT is a unique data-independent transform. As expected, these results are worse for GFT.

VI. Discussion and Future Work

The simulations show mixed results in the comparison between GFT and DCT for light-field compression in an HEVC-based system. When comparing the reduction in number of coefficients, GFT is rather promising, being capable of reducing the number of transform coefficients by up to 21.18% in some cases, while keeping equal or better distortion when compared to DCT. Transform coding gain was used in order to provide an insight of how well transform coefficients may be coded, but results may be biased due to GFT not being a data-independent transform. The compression system employed is a simplified model based on HEVC, therefore several possibly relevant optimization procedures were not considered. Employing GFT in a more complex system is required so as to allow practical operation analysis. Moreover, several possible methods for implementing GFT were proposed, but some analyses were restricted to a single setup using *rows* prediction scheme and computing $\mathbf{A}_t$ from all residual images. A broader analysis could offer better understanding of GFT behavior.

VII. Conclusions

This work proposed and analyzed the use of GFT for light-field compression in an HEVC-based compression system. The comparison of the proposed method against the traditionally used DCT shows that GFT greatly reduces the number of coefficients required to represent light-field residual images while providing smaller distortion when compared to DCT. Different methods for constructing and employing GFT were tested for real and synthetic light fields. Using multiple images when computing the coefficients of the adjacency matrix provides a more uniform compression across residual images within a prediction group and improves reduction when compared to computing coefficients from a single reference residual image. An analysis of different prediction methods was conducted, as well as an analysis of how similarity between images within a prediction group affects the performance. When estimated from coefficients from the entire light field, transform coding gain favors DCT. Given the fact that GFT is data-dependent and, thus, not a fixed transform, a transform coding gain analysis regarding blocks for which GFT is unique was conducted. This analysis yields better transform coding gain for GFT. The compression system adopted may be improved in order to allow the comparison with practical coding systems.

References

[1] M. Levoy and P. Hanrahan, “Light field rendering,” in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, New Orleans, LA, Aug 1996, pp. 31–42, doi: 10.1145/237170.237199.

[2] R. Ng, *Digital Light Field Photography*. PhD thesis, Stanford University, Stanford, CA, USA, 2006. AAI3219345.

[3] M. Levoy, “Light fields and computational imaging,” *Computer*, vol. 39, pp. 46–55, Aug 2006, doi: 10.1109/MC.2006.270.

[4] D. Lanman and D. Luebke, “Near-eye light field displays,” *ACM Transactions on Graphics*, vol. 32, pp. 220:1–220:10, Nov 2013, doi: 10.1145/2508363.2508366.

[5] L.-Y. Wei, C.-K. Liang, G. Myhre, C. Pitts, and K. Akeley, “Improving light field camera sample design with irregularity and aberration,” *ACM Transactions on Graphics*, vol. 34, pp. 152:1–152:11, Jul 2015, doi: 10.1145/2766885.

[6] T. Ebrahimi, S. Foessel, F. Pereira, and P. Schelkens, “JPEG Pleno: toward an efficient representation of visual reality,” *IEEE Multimedia*, vol. 23, pp. 14–20, Oct.-Dec 2016, doi: 10.1109/MMUL.2016.64.

[7] G. Wu, B. Masia, A. Jarabo, Y. Zhang, L. Wang, Q. Dai, T. Chai, and Y. Liu, “Light field image processing: An overview,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, pp. 926–954, Oct 2017, doi: 10.1109/jstsp.2017.2747126.

[8] M. Levoy, R. Ng, A. Adams, M. Footer, and M. Horowitz, “Light field microscopy,” *ACM Transactions on Graphics*, vol. 25, pp. 924–934, Jul 2006, doi: 10.1145/1141911.1141976.

[9] N. C. Pégard, H.-Y. Liu, N. Antipa, M. Gerlock, H. Adesnik, and L. Waller, “Compressive light-field microscopy for 3D neural activity recording,” *Optica*, vol. 3, pp. 517–524, May 2016, doi: 10.1364/OPTICA.3.000517.

[10] M. Magnor and B. Girod, "Data compression for light-field rendering," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, pp. 338–343, Apr 2000, doi: 10.1109/76.836278.

[11] T. Sakamoto, K. Kodama, and T. Hamamoto, "A study on efficient compression of multi-focus images for dense light-Field reconstruction," *2012 IEEE Visual Communications and Image Processing (VCIP)*, San Diego, CA, Nov 2012, doi: 10.1109/VCIP.2012.6410759.

[12] C. Perra, "On the coding of plenoptic raw images," in *22nd Telecommunications Forum (TELFOR)*, IEEE, Belgrade, Nov 2014, doi: 10.1109/telfor.2014.7034539.

[13] A. Vieira, H. Duarte, C. Perra, L. Tavora, and P. Assuncao, "Data formats for high efficiency coding of lytro-illum light fields," in *IEEE International Conference on Image Processing Theory, Tools and Applications (IPTA)*, Orleans, Nov 2015, doi: 10.1109/ipta.2015.7367195.

[14] C. Perra and P. Assuncao, "High efficiency coding of light field images based on tiling and pseudo-temporal data arrangement," in *IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, Seattle, WA, Jul 2016, pp. 1–4, doi: 10.1109/ICMEW.2016.7574671.

[15] E. Cornwell, L. Li, Z. Li, and Y. Sun, "An efficient compression scheme for the multi-camera light field image," in *19th International Workshop on Multimedia Signal Processing (MMSP)*, Luton, Oct 2017, doi: 10.1109/mmsp.2017.8122243.

[16] "ISO/IEC JTC 1/SC 29 Programme of Work." https://www.itscj.ipsj.or.jp/sc29/29w42901.htm. Accessed: 2018-02-26.

[17] 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 Process. Mag.*, vol. 30, no. 3, pp. 83–98, 2013, doi: 10.1109/MSP.2012.2235192.

[18] I. Jablonski, "Graph signal processing in applications to sensor networks, smart grids, and smart cities," *IEEE Sensors Journal*, vol. 17, pp. 7659–7666, Dec 2017, doi: 10.1109/jsen.2017.2733767.

[19] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs," *IEEE Transactions on Signal Processing*, vol. 61, pp. 1644–1656, Apr 2013, doi: 10.1109/TSP.2013.2238935.

[20] 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*, vol. 31, pp. 80–90, Sept 2014, doi: 10.1109/MSP.2014.2329213.

[21] N. Perraudin and P. Vandergheynst, "Stationary signal processing on graphs," *IEEE Transactions on Signal Processing*, vol. 65, pp. 3462–3477, Jul 2017, doi: 10.1109/tsp.2017.2690388.

[22] O. Teke and P. P. Vaidyanathan, "Extending classical multirate signal processing theory to graphs—part I: Fundamentals," *IEEE Transactions on Signal Processing*, vol. 65, pp. 409–422, Jan 2017, doi: 10.1109/tsp.2016.2617833.

[23] O. Teke and P. P. Vaidyanathan, "Extending classical multirate signal processing theory to graphs—part II: M-channel filter banks," *IEEE Transactions on Signal Processing*, vol. 65, pp. 423–437, Jan 2017, doi: 10.1109/tsp.2016.2620111.

[24] A. Gavili and X.-P. Zhang, "On the shift operator, graph frequency, and optimal filtering in graph signal processing," *IEEE Transactions on Signal Processing*, vol. 65, pp. 6303–6318, Dec 2017, doi: 10.1109/tsp.2017.2752689.

[25] F. R. K. Chung, *Spectral Graph Theory*. American Mathematical Society, 1997, doi: 10.1090/cbms/092.

[26] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs: graph Fourier transform," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, Vancoucer, BC, May 2013, vol. 62, pp. 6167–6170, doi: 10.1109/ICASSP.2013.6638850.

[27] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs: frequency analysis," *IEEE Transactions on Signal Processing*, vol. 62, pp. 3042–3054, Jun 2014, doi: 10.1109/MSP.2014.2329213.

[28] V. R. M. Elias and W. A. Martins, "Graph Fourier transform for light field compression," in *XXXV Simpósio Brasileiro de Telecomuniçes e Processamento de Sinais (SBrT)*, São Pedro, São Paulo, Sept 2017.

[29] R. Monteiro, L. Lucas, C. Conti, P. Nunes, N. Rodrigues, S. Faria, C. Pagliari, E. da Silva, and L. Soares, "Light field HEVC-based image coding using locally linear embedding and self-similarity compensated prediction," in *IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, Seatle, WA, Jul 2016, pp. 1–4, doi: 10.1109/ICMEW.2016.7574670.

[30] W. Hu, G. Cheung, A. Ortega, and O. C. Au, "Multiresolution graph Fourier transform for compression of piecewise smooth images," *IEEE Transactions on Image Processing*, vol. 24, pp. 419–433, Jan 2015, doi: 10.1109/tip.2014.2378055.

[31] G. Shen, W.-S. Kim, S. K. Narang, A. Ortega, J. Lee, and H. Wey, "Edge-adaptive transforms for efficient depth map coding," in *28th Picture Coding Symposium*, Nagoya, Dec 2010, doi: 10.1109/pcs.2010.5702565.

[32] A. Gershun, "The light field," *Journal of Mathematics and Physics*, vol. 18, no. 1-4, pp. 51–151, 1939, doi: 10.1002/sapm193918151.

[33] Y. Li, R. Olsson, and M. Sjostrom, "Compression of unfocused plenoptic images using a displacement intra prediction," in *2016 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, Seatle, WA, Jul 2016, pp. 1–4, doi: 10.1109/ICMEW.2016.7574673.

[34] A. Dricot, J. Jung, M. Cagnazzo, B. Pesquet, and F. Dufaux, "Integral images compression scheme based on view extraction," in *23rd European Signal Processing Conference (EUSIPCO)*, Nice, Aug 2015, doi: 10.1109/eusipco.2015.7362353.

[35] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Applied and Computational Harmonic Analysis*, vol. 30, pp. 129–150, Mar 2011, doi: 10.1016/j.acha.2010.04.005.

[36] "The (new) Stanford light field archive." http://lightfield.stanford.edu/lfs.html. Accessed: 2018-02-20.

[37] "4D Light Field Benchmark." http://hci-lightfield.iwr.uni-heidelberg.de. Accessed: 2018-02-20.

[38] K. Honauer, O. Johannsen, D. Kondermann, and B. Goldluecke, "A dataset and evaluation methodology for depth estimation on 4D light fields," in *Asian Conference on Computer Vision (ACCV)*, Taipei, 2016, pp. 19–34, doi: 10.1007/978-3-319-54187-7_2.

[39] A. Sandryhaila and J. M. F. Moura, "Nearest-neighbor image model," in *19th IEEE International Conference on Image Processing*, Orlando, FL, Sept 2012, no. 3, pp. 2521–2524, doi: 10.1109/ICIP.2012.6467411.

[40] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, pp. 600–612, Apr 2004, doi: 10.1109/tip.2003.819861.

[41] T. Wiegand, "Source coding: Part I of fundamentals of source and video coding," *Foundations and Trends® in Signal Processing*, vol. 4, no. 1-2, pp. 1–222, 2010, doi: 10.1561/2000000010.

Vitor R. M. Elias was born in Brazil in 1990. He received the B.Sc in Electronics and Computer Engineering (cum laude) degree from the Federal University of Rio de Janeiro (UFRJ) in 2013. In 2015, he received the M.Sc. degree in Electrical Engineering from the COPPE/UFRJ, where he is pursuing the Ph.D. degree at the Program of Electrical Engineering since 2016. He is currently enrolled at the Signals, Multimedia, and Telecommunications Lab (SMT) and his experience and research interests include the areas of image and video compression, machine learning, and digital signal processing on graphs, with applications to image, video, and biomedical signals processing.

Wallace A. Martins was born in Brazil in 1983. He received the Electronics Engineer degree from the Federal University of Rio de Janeiro (UFRJ) in 2007, the M.Sc. and D.Sc. degrees in Electrical Engineering also from UFRJ in 2009 and 2011, respectively. He was a Research Visitor at University of Notre Dame (USA, 2008), at Université Lille 1 (France, 2016), and at Universidad de Alcalá (Spain, 2018). From 2010 to 2013 he was an Associate Professor of the Federal Center for Technological Education Celso Suckow da Fonseca (CEFET/RJ). Since 2013 he has been with the Department of Electronics and Computer Engineering (DEL/Poli) and Electrical Engineering Program (PEE/COPPE) at UFRJ, where he is presently an Associate Professor. His research interests are in the fields of digital signal processing, digital communications, visible light communications, massive MIMO systems, underwater communications, microphone/sensor array processing, adaptive signal processing, and digital signal processing on graphs. Dr. Martins received the Best Student Paper Award from EURASIP at EUSIPCO-2009, Glasgow, Scotland, and the 2011 Best Brazilian D.Sc. Dissertation Award from Capes.

Chapter 9

Publications on Extended Adjacency using Diffusion Distances

P3 V. R. M. Elias, W. A. Martins, and S. Werner, “Diffusion-based virtual graph adjacency for Fourier analysis of network signals,” in *Simpósio Brasileiro de Telecomunicações e Processamento de Sinais*, pp. 1–5, Dec. 2020.

P4 © [2020] IEEE. Reprinted, with permission, from V. R. M. Elias, W. A. Martins, and S. Werner, “Extended adjacency and scale-dependent graph Fourier transform via diffusion distances,” *IEEE Transactions on Signal and Information Processing over Networks*, vol. 6, pp. 592–604, Aug. 2020.

Diffusion-based Virtual Graph Adjacency for Fourier Analysis of Network Signals

Vitor Rosa Meireles Elias, Wallace Alves Martins, and Stefan Werner

Abstract— **This work proposes a graph model for networks where node collaborations can be described by the Markov property. The proposed model augments an initial graph adjacency using diffusion distances. The resulting virtual adjacency depends on a diffusion-scale parameter, which leads to a controlled shift in the graph-Fourier-transform spectrum. This enables a frequency analysis tailored to the actual network collaboration, revealing more information on the graph signal when compared to traditional approaches. The proposed model is employed for anomaly detection in real and synthetic networks, and results confirm that using the proposed virtual adjacency yields better classification than the initial adjacency.**

Keywords— **diffusion distances, virtual adjacency matrix, graph signal processing (GSP), graph Fourier transform (GFT).**

I. Introduction

The connectivity of real-world elements and the amount of data generated in networks have been increasing consistently [1], [2]. Real networks and their corresponding data come in vastly different shapes and applications, ranging from genetic interaction networks [3] and the human brain [4] to sensor networks and smart cities [5]. Graph signal processing (GSP) explores pairwise relations between elements of a network to construct tools suitable for the processing of network data [1], [2], [6]–[12]. In GSP, networks are modeled as graphs and data defined over, or generated by, elements of these networks are modeled as a graph signal—a mapping from the set of vertices into the set of complex numbers. The relations between elements of the real network are embedded into the edges of the graph, connecting pairs of vertices.

Several GSP tools are functions of a graph-shift operator (GSO), which carries the network-structure information. Thus, the choice of the GSO and its properties impact the outcomes of many GSP tools [13]–[16]. Different approaches for GSP and the definition of the GSO have emerged over the last years [2], [6], [17] and many works are devoted to improve the framework [7], [18]. The two main approaches define the GSO either as the adjacency matrix of the graph [6] or the Laplacian matrix [2].

GSP is a highly application-dependent framework. This dependency starts in the choice of the mapping from the real network into the graph [19], [20]. Most works rely on generic approaches for modeling the network. For example, defining the structure for physical networks in terms of distance between network elements, and adopting a GSO equal to the Laplacian matrix. If the network and application are known, a GSO can be tailored to the application at hand. Here, we consider Markov networks and the frequency analysis of the associated signals. In a Markov network, nodes collaborate with each other according to a defined Markov property, which defines the initial network adjacency.

We propose the use of diffusion distances (DDs) to incorporate the Markov property into the GSO. DD is a concept within the diffusion maps (DMs) framework proposed in [21]. The DM framework is conceived as a tool for uncovering a hidden geometry of the dataset by exploring properties of the eigenfunctions of the Markov matrix associated with the network states [22], [23]. The DDs serve as a metric for the diffusion-probability-based relation between two states of data. One parameter on the computation of DDs is the number of transition steps of the Markov chain, which corresponds to the stage or level of the collaboration. Hence, the proposed model depends on this number of steps. As graph spectrum depends on graph connectivity, the corresponding Fourier analysis adapts to the collaboration embedded in the model.

The combination of DM and GSP has been considered in [24], [25]. These works proposed the use of Markov matrices as GSO. The Markov matrix has desirable properties for GSP, such as being diagonalizable, and allows the use of DM-based tools, such as dimensionality reduction and clustering [24]. In contrast, we model the relation between elements of Markov network as a function of DDs. This yields an adjacency matrix that captures virtual dependencies between non-adjacent elements of the network. This is particularly relevant when nodes collaborate with neighbors over a sequence of interactions. The virtual adjacency reveals additional spectral content, which can be exploited by applications that make decisions based on frequency features, such as classifiers and detectors.

The paper is organized as follows: Section II reviews concepts of GSP and DM. Section III presents the proposed virtual-adjacency matrix and its effects on the graph Fourier analysis. In Section IV, we analyze the proposed model using numerical experiments and use it together with spectral analysis for anomaly detection in synthetic and real networks. Section V concludes the paper.

Mr. Elias is with the Department of Electronic Systems, NTNU – Norwegian University of Science and Technology (vitor.elias@ntnu.no); Dr. Martins is with the Interdisciplinary Centre for Security, Reliability and Trust, SnT, UniLu – University of Luxembourg (wallace.alvesmartins@uni.lu); Dr. Werner is with the Department of Electronic Systems, NTNU – Norwegian University of Science and Technology (stefan.werner@ntnu.no). Mr. Elias and Dr. Martins are also with the Electrical Engineering Program, PEE/COPPE, UFRJ – Federal University of Rio de Janeiro.

The work of Mr. Elias was supported, in part, by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) -– Grant number: 88887.310189/2018-00 and Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq). The work of Dr. Martins was supported, in part, by the ERC project AGNOSTIC and, in part, by FAPERJ. The work of Dr. Werner was supported, in part, by the Research Council of Norway.

II. Background and Notation

A. Graph signal processing

The graph is denoted by $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{v_1, \ldots, v_N\}$ is the set of vertices (or nodes) and $\mathcal{E} = \{e_{11}, \ldots, e_{NN}\}$ is the set of edges. Each vertex corresponds to one element of the network being modeled. Elements e_{ij} indicate pairwise relations between nodes v_i and v_j of the graph. An edge e_{ij} exists if and only if v_i and v_j are related (adjacent). These relations may incorporate functional properties of the network, based on the network data, or structural characteristics of the network, yielding relations based on the network elements. We represent the set of, possibly weighted, edges in the adjacency matrix $\mathbf{A}$.

The degree matrix $\mathbf{D}$ is a diagonal matrix such that $D_{jj} = \deg(v_j)$, where $\deg(v_j) = \sum_{i \in \mathcal{N}_j} e_{ij}$ and $\mathcal{N}_j$ is the set of vertices that are adjacent to v_j, referred to as neighborhood of v_j. Assuming a symmetric adjacency matrix, the graph Laplacian is the positive semidefinite matrix $\mathbf{L} = \mathbf{D} - \mathbf{A}$.

In the GSP framework [2], [6], the signal on a graph is given by the mapping $s : \mathcal{V} \to \mathbb{C}$ and is usually represented by a vector $\mathbf{s}$, such that the ith entry of $\mathbf{s}$ is $s_i = s(v_i)$. The graph signal represents a snapshot of the network state at a given time instant. Let $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathrm{T}}$ be the eigendecomposition of the graph Laplacian, where $\mathbf{\Lambda}$ is a diagonal matrix whose elements are the real eigenvalues λ_i and $\mathbf{U}$ has the eigenvectors $\mathbf{u}_i$ as columns, with $i \in \{1, \ldots, N\}$. The graph Fourier transform (GFT) of a graph signal $\mathbf{s}$ is given by $\hat{\mathbf{s}} = \mathbf{U}^{\mathrm{T}}\mathbf{s}$ and the signal can be recovered from the GFT coefficients via the inverse GFT as $\mathbf{s} = \mathbf{U}\hat{\mathbf{s}}$ [2]. Here, $\mathbf{L}$ is taken as the GSO. The eigenvalues carry a notion of graph frequency that quantifies the intensity of the signal variation across the graph nodes, with larger eigenvalues indicating higher variations [2]. We note that several definitions of the GFT exist in literature. For instance, [6] defines the GFT in terms of the eigendecomposition of $\mathbf{A}$, with $\mathbf{A}$ taken as the GSO.

B. Diffusion maps and distances

In this section, we consider data-state-wise graphs, instead of network-wise graphs from GSP. Let $\mathbf{X} \in \mathbb{R}^{N \times K}$ be a data matrix that represents a set of K data points of dimension N, also called states, and assume that there is an underlying (hidden) process that relates the data points and possibly influences the data generation. The DM framework aims to make this process explicit [21]–[24]. Our focus is on one of the subproducts of the DM framework, which is the concept of *diffusion distances*. For a detailed description of DMs, see [21]–[24].

The first step of the DM framework is to create a graph in which nodes correspond to the columns of $\mathbf{X}$. The resulting graph is not intended to model a network, but to capture relations between data states. One may interpret it as a state-wise graph instead of the traditional element-wise graph described in Section II-A. The edges are computed via a symmetric non-negative kernel that maps the affinity between two states into a real value, thereby defining an adjacency matrix $\mathbf{A}$ [21]. By normalizing the adjacency matrix, a probability of transitioning from state $\mathbf{x}_i$ to $\mathbf{x}_j$ is given by $p(\mathbf{x}_j|\mathbf{x}_i) = A_{ij}/\deg(v_i)$. A right-stochastic (Markov) matrix $\mathbf{M} = \mathbf{D}^{-1}\mathbf{A}$ comprises all transition probabilities [26]. Powers $\mathbf{M}^t$ are associated with t steps of the random walk and $p_t(\mathbf{x}_j|\mathbf{x}_i) = M_{ij}^{(t)}$, the (i, j)th element of $\mathbf{M}^t$, denotes the probability of starting at $\mathbf{x}_i$ and reaching $\mathbf{x}_j$ after t steps.

The DD between two states is computed as [21], [23], [24]

$$D_t^2(\mathbf{x}_i, \mathbf{x}_j) = \sum_{z=1}^{K} \frac{(p_t(\mathbf{x}_z|\mathbf{x}_i) - p_t(\mathbf{x}_z|\mathbf{x}_j))^2}{\phi_{1,z}}, \tag{1}$$

where $\phi_{1,z}$ is the zth entry of $\boldsymbol{\phi}_1$, the top left eigenvector of $\mathbf{M}$. The DD extends relations from the local structure given by $\mathbf{A}$ into a global metric by assimilating probabilities of diffusion paths through the graph. Two points with similar posterior distributions have small DD, even if they are not initially adjacent. This can be seen in terms of paths connecting the two initial points through the end-points in the posterior distributions. The computation of the DD depends on the parameter t, which we denote as the diffusion-scale parameter. It represents the number of steps taken in the random walk. Increasing t allows an initial state to reach ending states that are further away in terms of steps of the random walk.

III. GSP for Markov Networks

Markov networks inherently allow a modeling of the relation between its elements in terms of a stochastic matrix. Some networks that admit this modeling are: consensus networks [27]–[29], which perform a possibly weighted average of neighboring nodes in order to reach consensus through the entire network; conservative diffusion networks [30]–[32]; and random-walk driven networks [26], [33]. We propose to use DDs to incorporate the Markov property into the graph model. This approach renders a GSO tailored for Markov networks and provides a graph-frequency-analysis tool that offers more information on the signal when compared to traditional approaches.

A. Virtual adjacency

Modeling of networks for GSP usually relies on some strict constraints of adjacency between nodes. For instance, a wireless sensor network (WSN) is usually modeled in terms of the direct communication capabilities between sensors, given limitations imposed by the physical distance between them. If it is a consensus network, however, it operates on the data through iterative steps according to a stochastic matrix [27]. Thus, a node of the network is related to another node that is not adjacent through collaboration. We model this relation by adapting the concept of DDs and derive the virtual-adjacency matrix. Let the graph $\mathcal{G} = \{\mathcal{V}, \mathbf{B}\}$ model the network with an initial symmetric Markov-like adjacency matrix $\mathbf{B}$. The DD between nodes v_i and v_j (elements of the network) is defined by

$$D_t^2(v_i, v_j) = \sum_{n=1}^{N} \frac{\left(B_{in}^{(t)} - B_{jn}^{(t)}\right)^2}{(1/N)}, \tag{2}$$

where $(1/N)$ corresponds to the elements of the top left eigenvector of $\mathbf{B}$ and $B_{ij}^{(t)}$ denotes the (i, j)th element of $\mathbf{B}^t$.

The DD in (2) extends the relation of nodes to other nodes that are outside their original neighborhoods and this extension is dependent on the scale parameter t. From this definition of DD between elements of the network, we propose the virtual adjacency matrix $\mathbf{A}(t)$ defined by

$$A_{ij}(t) = \begin{cases} B_{ij} + \exp\left(-\frac{D_t^2(v_i,v_j)}{\rho N}\right) & i \neq j \\ 0 & i = j. \end{cases} \tag{3}$$

where ρ is a free parameter, N is the size of the network, and ρN prevents the diffusion-related term from quickly fading to zero as network size increases. Note that N cancels out with the denominator in (2). The parameter ρ provides sufficient adjustment of the function, meeting possible requirements of different applications.

It must be noted that DDs are always real positive values, yielding $A_{ij}(t) > 0$ for every $i \neq j$. Thus, instead of using the expression in (3) we actually consider edges to exist only if $A_{ij}(t)$ is greater than a given threshold; otherwise, $A_{ij}(t)$ is set to zero. For increasing t, the DD between nodes is non-decreasing. This leads to non-decreasing edge values and new edges possibly appearing as t increases. As t tends to infinity, the graph is expected to become fully connected. GSOs associated with different scales t model different stages of node collaboration.

B. *Adaptable Fourier Analysis*

As presented in Section II-A, the GFT is defined as the expansion of a graph signal in terms of the eigenvectors of the Laplacian matrix, which in turn depends on the adjacency matrix. Hence, the dependence on the diffusion-scale parameter t in the definition of $\mathbf{A}(t)$ is carried over to the GFT, since the diffusion-scale-dependent Laplacian matrix is

$$\mathbf{L}(t) = \mathbf{D}(t) - \mathbf{A}(t), \tag{4}$$

where $\mathbf{D}(t)$ is the degree matrix associated with $\mathbf{A}(t)$. Given the eigendecomposition of the Laplacian as $\mathbf{L}(t) = \mathbf{U}(t)\mathbf{\Lambda}(t)\mathbf{U}^{\mathrm{T}}(t)$, the graph Fourier transform can be written as

$$\hat{\mathbf{x}}(t) = \mathbf{U}^{\mathrm{T}}(t)\mathbf{x}, \tag{5}$$

which, now, depends on t. Laplacian eigenvalues of connected graphs are non-decreasing with addition of edges [34]. As the number of edges in the virtual-adjacency matrix is non-decreasing with t, increasing t also increases the eigenvalues of $\mathbf{L}(t)$. This yields a frequency analysis tailored for each stage of node collaboration. Intuitively, more collaboration results in more connected nodes. Therefore, variations in graph signals are perceived by more pairs of nodes, thus corresponding to larger graph frequencies if compared to the frequency content obtained using the initial adjacency. Hence, by incorporating node collaboration into the graph model, we provide a frequency analysis that reveals more information on the network signal than that offered by the conventional GFT. Therefore, applications that use spectrum-related features, such as classifiers and detectors, can benefit from the proposed methodology.

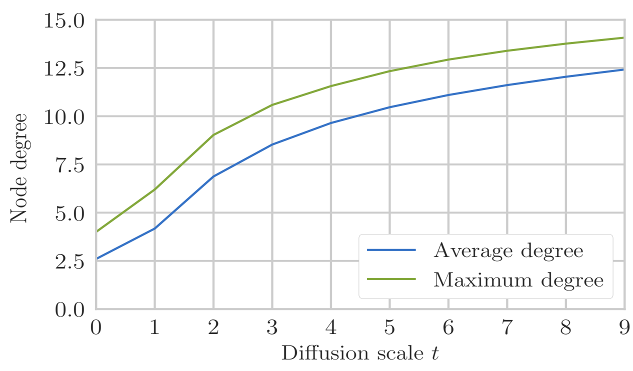


Fig. 1. Average and maximum node degree versus diffusion scale t. For $t = 0$, the values correspond to the average and maximum degrees of the original κNN network.

IV. Experiments

In this section, we present numerical experiments to showcase properties of the virtual-adjacency matrix and its effects on the Fourier analysis. Experiments are conducted over a sensor network that is modeled as a κ-nearest-neighbors (κNN) graph $\mathcal{G} = \{\mathcal{V}, \mathbf{B}\}$, such that each sensor is bidirectionally connected to the κ nearest sensors. A sensor can be a nearest neighbor of a sensor that is not one of its nearest neighbors. Note that this allows a sensor to be connected to more than κ sensors. Let the symmetric matrix $\mathbf{A}$ indicate connections between nodes, such that $A_{ij} = 1$ if and only if nodes v_i and v_j are connected. A consensus algorithm is implemented over this network, such that the updated network state at iteration $k+1$ is given by $\mathbf{x}(k+1) = \mathbf{B}\mathbf{x}(k)$, where the stochastic matrix $\mathbf{B}$ is defined as $\mathbf{B} = \mathbf{I} - \epsilon\mathbf{L}$, where $\mathbf{L}$ is the Laplacian of $\mathbf{A}$ [27]. The parameter ϵ affects the convergence of the consensus algorithm and is set to $\epsilon = 1/(1.25\Delta)$, where Δ is the maximum degree in $\mathcal{G}$. Given $\mathbf{B}$, we can associate the network with a Markov chain and, hence, the techniques in Section III can be applied.

A. *Increasing connectivity*

From the initial stochastic adjacency matrix $\mathbf{B}$, we compute the diffusion distance between sensors in the network and the virtual-adjacency matrix is constructed according to (3). We consider a κNN network with $N = 20$ sensors and $\kappa = 2$ nearest sensors. The parameter ρ in (3) is fixed at 0.35. Fig. 1 shows the graph average and maximum degrees of the virtual adjacency matrix versus diffusion scale, with scales varying from $t = 1$ to $t = 9$. As t is increased, more steps of the random walk are taken in the network and, given a reference node, the distance to nodes that are further away is expected to decrease. This results in more edges being created and, therefore, the connectivity of the graph increases. In fact, as t tends to infinity, the diffusion distance between any two nodes in a connected graph tends to zero and $\mathbf{A}(t)|_{t\to\infty}$ tends to the adjacency matrix of a fully connected graph with no self loops, i.e., each node is connected to every other node.

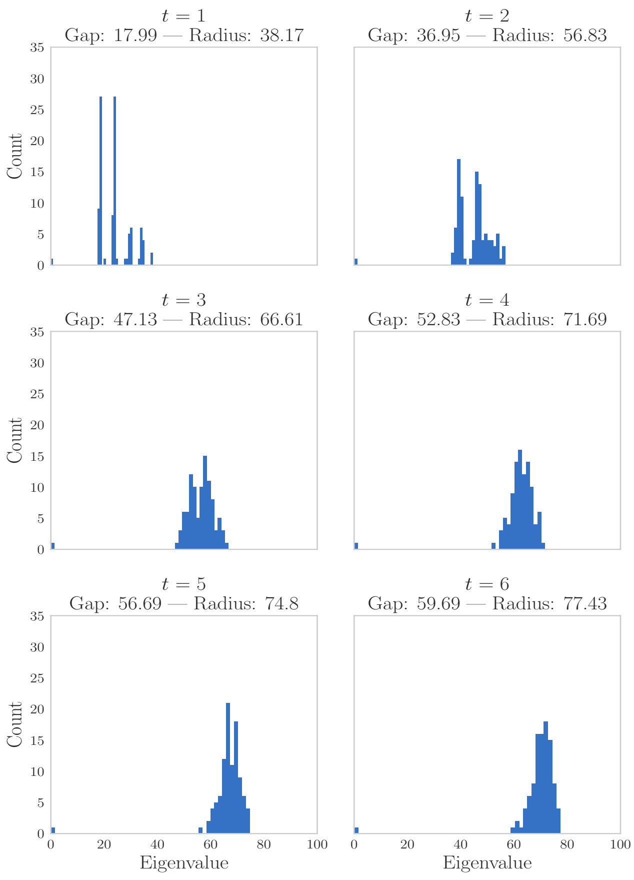


Fig. 2. Histogram of eigenvalues of diffusion Laplacian matrices $\mathbf{L}_t$, for $t \in \{1, \ldots, 6\}$. Spectral "gap" denotes the smallest non-zero eigenvalue and spectral "radius" denotes the maximum eigenvalue.

B. *Spectrum analysis*

For a network with $N = 100$ sensors and $\kappa = 4$, we analyze the behavior of the GFT spectrum for different diffusion scales, for $\mathbf{A}(t)$ constructed with $\rho = 0.4$. The histograms of the eigenvalues of $\mathbf{L}(t)$ for $t \in \{1, \ldots, 6\}$ are shown in Fig. 2. Increasing the diffusion scale and connectivity produces a shift on the graph-frequency spectral range into higher frequency ranges. In fact, as $t \to \infty$, the graph becomes fully connected and the maximum eigenvalue tends to $\{N + \theta_i\}_{i=1}^N$, where $\{\theta_i\}_{i=1}^N$, with $|\theta_i| \leq 1$, are the eigenvalues of the Laplacian of the stochastic matrix $\mathbf{B}$. Each diffusion scale t yields a frequency analysis associated with different collaboration stages. Thus, the virtual adjacency allows for additional spectrum information when compared to the initial adjacency. This additional information can be used by applications that make decisions based on the signal's frequency content, as exemplified in the next section.

C. *Application*

The virtual adjacency is implemented in a synthetic consensus network and in a real weather-station network, wherein spectral analysis is used for anomaly detection [38]. Different diffusion scales and the parameter ρ allow for a tailored frequency decomposition. We compare detectors based the spectrum yielded by virtual adjacency matrices against the detector based on that of the initial adjacency. A similar task using GFT was applied in [35]–[37]. We evaluate the f1 score, given by the harmonic mean between precision and recall of the detectors.

We use a network with $N = 150$ sensors measuring a synthetic healthy signal drawn from a normal distribution with expected value equal to 20 and variance of 0.4. The anomaly is injected in up to 2 sensors that are randomly chosen as anomalous sensors. Anomalous sensor measurements are drawn from a uniform integer distribution in the interval $[15, 25]$.

The detector is based on applying a threshold on the coefficients of the graph signal after a high-pass filter, assuming that anomalies induce high frequencies on the signal. We use grid search to optimize the filter cut-off frequency and the detection threshold for the conventional GFT. For the GFT based on virtual adjacency matrices, ρ is also optimized. We treat three scales $t \in \{1, 2, 3\}$ separately. This simulation is performed for three different stages of the consensus algorithm: raw data following previously described distributions; data after one consensus step; and data after two consensus steps. Results for the f1 score obtained by different detectors are presented in Table I, showing that the GFT using virtual adjacency achieves better results than those obtained by the conventional approach.

For anomaly detection in a real network, we use 150 randomly-selected weather stations, modeled as the graph in Fig. 3, and temperature data from the database in [38]. It must be noted that a Markovian relation is not initially defined for the nodes of this network. However, we are able to generate a Markovian relation by constructing a matrix $\mathbf{B} = \mathbf{I} - \epsilon\mathbf{L}$ based on physical adjacency. This example serves to show that the proposed methodology only requires a form of Markov model, but it is not restricted to Markov networks. Original data are converted from degrees Fahrenheit to degrees Celsius and range from -29.4°C to 38.6°C. We introduce anomaly in up to 7 stations. Anomaly is given by additive white Gaussian noise with variance equal to 1 °C^2 and mean drawn from

TABLE I
F1 SCORES — SYNTHETIC DATA

	0 step	1 step	2 steps
Virtual Adjacency $t = 1$	0.70	0.62	0.53
Virtual Adjacency $t = 2$	0.73	**0.69**	**0.62**
Virtual Adjacency $t = 3$	**0.74**	0.66	0.61
Initial Adjacency	0.66	0.60	0.57

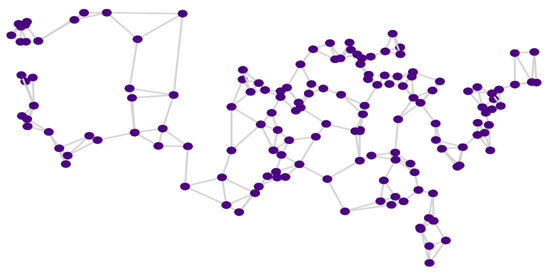


Fig. 3. Graph model for weather-stations network.

the integer uniform distribution $[-7\ ^\circ\mathrm{C}, +7\ ^\circ\mathrm{C}]$. Conventional GFT achieves f1 score of 0.50, while GFT using the proposed model achieves better results with f1 score equal to 0.55 for $t = 1$ and 0.64 for $t = 2$.

V. Conclusion

We proposed an virtual-adjacency matrix which adapts Fourier analysis to node collaboration in Markov networks. We construct the adjacency matrix as function of diffusion distances between elements of the network. The obtained virtual adjacency depends on the diffusion scale, given by the number of diffusion steps under consideration, such that increasing diffusion scales increases the connectivity of the graph. We showed that the virtual adjacency allows for an adaptable graph-frequency analysis considering different levels of collaboration between nodes. Changing the diffusion scale in the construction of the virtual-adjacency matrix shifts the range of frequencies discriminated by the GFT. Tools that operate on the graph spectrum can leverage on the additional information. For instance, we employed the proposed model for anomaly detection using spectral information. The resulting detectors leveraged on the proposed graph-frequency representation associated with different collaboration stages to obtain better results than those achieved by the GFT. As future work, we aim at exploring other applications for the proposed methodology and investigating efficient ways to determine the optimal diffusion-scale parameter.

References

[1] A. Sandryhaila and J. M. F. Moura, “Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure,” *IEEE Signal Process. Mag.*, vol. 31, pp. 80–90, Sep. 2014.

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

[3] B. Boucher and S. Jenna, “Genetic interaction networks: better understand to better predict,” *Front. in Genet.*, vol. 4, pp. 1–16, Dec. 2013.

[4] W. Huang, T. A. W. Bolton, J. D. Medaglia, D. S. Bassett, A. Ribeiro, and D. Van De Ville, “A graph signal processing perspective on functional brain imaging,” *Proc. IEEE*, vol. 106, pp. 868–885, May 2018.

[5] I. Jablonski, “Graph signal processing in applications to sensor networks, smart grids, and smart cities,” *IEEE Sensors J.*, vol. 17, pp. 7659–7666, Dec. 2017.

[6] A. Sandryhaila and J. M. F. Moura, “Discrete signal processing on graphs,” *IEEE Trans. Signal Process.*, vol. 61, pp. 1644–1656, Apr. 2013.

[7] A. Ortega, P. Frossard, J. Kovacevic, J. M. F. Moura, and P. Vandergheynst, “Graph signal processing: Overview, challenges, and applications,” *Proc. IEEE*, vol. 106, pp. 808–828, May 2018.

[8] G. Ribeiro and J. Lima, “Graph signal processing in a nutshell,” *J. of Commun. and Inf. Syst. (JCIS)*, vol. 33, no. 1, pp. 219–233, July 2018.

[9] V. R. M. Elias and W. A. Martins, “Graph Fourier transform for light field compression,” in *Proc. XXXV Simpósio Brasileiro de Telecomunicações e Processamento de Sinais (SBrT)*, Sep. 2017, pp. 881–885.

[10] V. R. M. Elias and W. A. Martins, “On the use of graph Fourier transform for light-field compression,” *J. of Commun. and Inf. Syst. (JCIS)*, vol. 33, no. 1, pp. 92–103, May 2018.

[11] G. Lewenfus, W. A. Martins, S. Chatzinotas, and B. Ottersten, “On the use of vertex-frequency analysis for anomaly detection in graph signals,” in *Proc. XXXVII Simpósio Brasileiro de Telecomunicações e Processamento de Sinais (SBrT)*, Sep. 2019, pp. 1–5.

[12] M. J. M. Spelta and W. A. Martins, “Normalized LMS algorithm and data-selective strategies for adaptive graph signal estimation,” *Signal Process.*, vol. 167, 107326, Feb. 2020.

[13] A. Sandryhaila and J. M. F. Moura, “Discrete signal processing on graphs: Graph filters,” in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, May 2013, pp. 6163–6166.

[14] O. Teke and P. P. Vaidyanathan, “Extending classical multirate signal processing theory to graphs - Part I: Fundamentals,” *IEEE Trans. Signal Process.*, vol. 65, pp. 409–422, Jan. 2017.

[15] O. Teke and P. P. Vaidyanathan, “Extending classical multirate signal processing theory to graphs - Part II: M-channel filter banks,” *IEEE Trans. Signal Process.*, vol. 65, no. 2, pp. 423–437, Jan. 2017.

[16] B. Girault, “Stationary graph signals using an isometric graph translation,” in *Proc. 23rd Eur. Signal Process. Conf.*, Aug. 2015, pp. 1516–1520.

[17] A. Gavili and X.-P. Zhang, “On the shift operator, graph frequency, and optimal filtering in graph signal processing,” *IEEE Trans. Signal Process.*, vol. 65, pp. 6303–6318, Dec. 2017.

[18] P. Djuric and C. Richard, *Cooperative and graph signal processing: principles and applications*. Orlando, FL, USA: Academic Press, Inc., 1st ed., 2018.

[19] G. B. Giannakis, Y. Shen, and G. V. Karanikolas, “Topology identification and learning over graphs: Accounting for nonlinearities and dynamics,” *Proc. IEEE*, vol. 106, pp. 787–807, May 2018.

[20] G. Mateos, S. Segarra, A. G. Marques, and A. Ribeiro, “Connecting the dots: Identifying network structure via graph signal processing,” *IEEE Signal Process. Mag.*, vol. 36, pp. 16–43, May 2019.

[21] R. R. Coifman and S. Lafon, “Diffusion maps,” *Appl. Comput. Harmon. Anal.*, vol. 21, pp. 5–30, July 2006.

[22] R. R. Coifman, S. Lafon, A. B. Lee, M. Maggioni, B. Nadler, F. Warner, and S. W. Zucker, “Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps,” *Proc. Natl. Acad. Sci.*, vol. 102, pp. 7426–7431, May 2005.

[23] R. Talmon, I. Cohen, S. Gannot, and R. R. Coifman, “Diffusion maps for signal processing: A deeper look at manifold-learning techniques based on kernels and graphs,” *IEEE Signal Process. Mag.*, vol. 30, no. 4, pp. 75–86, 2013.

[24] A. Heimowitz and Y. C. Eldar, “A unified view of diffusion maps and signal processing on graphs,” in *Proc. 12th Int. Conf. on Sampl. Theory Appl.*, July 2017, pp. 308–312.

[25] A. Heimowitz and Y. C. Eldar, “The Nystrom extension for signals defined on a graph,” in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Apr. 2018, pp. 4199–4203.

[26] L. Lovasz, “Random walks on graphs: A survey,” *Combinatorics, Paul Erdos is Eighty*, vol. 2, pp. 1–46, 1993.

[27] R. Olfati-Saber, J. A. Fax, and R. M. Murray, “Consensus and cooperation in networked multi-agent systems,” *Proc. IEEE*, vol. 95, pp. 215–233, Jan. 2007.

[28] J. Qin, Q. Ma, Y. Shi, and L. Wang, “Recent advances in consensus of multi-agent systems: A brief survey,” *IEEE Trans. Ind. Electron.*, vol. 64, pp. 4972–4983, June 2017.

[29] B. Kailkhura, S. Brahma, and P. K. Varshney, “Data falsification attacks on consensus-based detection systems,” *IEEE Trans. Signal Inf. Process. Netw.*, vol. 3, pp. 145–158, Mar. 2017.

[30] D. Thanou, X. Dong, D. Kressner, and P. Frossard, “Learning heat diffusion graphs,” *IEEE Trans. Signal Inf. Process. Netw.*, vol. 3, pp. 484–499, Sep. 2017.

[31] O. Teke and P. P. Vaidyanathan, “Time estimation for heat diffusion on graphs,” in *Proc. 51st Asilomar Conf. Signals, Syst. Comput.*, Oct. 2018, pp. 1963–1967.

[32] N. Masuda, M. A. Porter, and R. Lambiotte, “Random walks and diffusion on networks,” *Phys. Rep.*, vol. 716-717, pp. 1–58, Nov. 2017.

[33] C. Gkantsidis, M. Mihail, and A. Saberi, “Random walks in peer-to-peer networks,” in *Proc. IEEE Conf. Comput. Commun.*, vol. 1, Mar. 2004, pp. 120–130.

[34] B. Mohar, “The Laplacian spectrum of graphs”, in *Graph Theory, Combinatorics, and Applications*, vol. 2, Wiley, 1991, pp. 871–898

[35] A. Sandryhaila and J. M. F. Moura, “Discrete signal processing on graphs: Frequency analysis,” *IEEE Trans. Signal Process.*, vol. 62, pp. 3042–3054, June 2014.

[36] E. Drayer and T. Routtenberg, “Detection of false data injection attacks in power systems with graph Fourier transform,” in *Proc. IEEE Global Conf. Signal. Inf. Process.*, Nov. 2018, pp. 890–894.

[37] M. Khatua, S. H. Safavi, and N. M. Cheung, “Sparse Laplacian component analysis for internet traffic anomalies detection,” *IEEE Trans. Signal Inf. Process. Netw.*, vol. 4, no. 4, pp. 697–711, Dec. 2018.

[38] “Global Surface Summary of the Day - GSOD - Data.gov.” https://catalog.data.gov/dataset/global-surface-summary-of-the-day-gsod. Accessed: 2019-07-29.

Extended Adjacency and Scale-dependent Graph Fourier Transform via Diffusion Distances

Vitor R. M. Elias, Wallace A. Martins, *Senior Member, IEEE* and Stefan Werner, *Senior Member, IEEE*

***Abstract*—This paper proposes the augmentation of the adjacency model of networks for graph signal processing. It is assumed that no information about the network is available, apart from the initial adjacency matrix. In the proposed model, additional edges are created according to a Markov relation imposed between nodes. This information is incorporated into the extended-adjacency matrix as a function of the diffusion distance between nodes. The diffusion distance measures similarities between nodes at a certain diffusion scale or time, and is a metric adopted from diffusion maps. Similarly, the proposed extended-adjacency matrix depends on the diffusion scale, which enables the definition of a scale-dependent graph Fourier transform. We conduct theoretical analyses of both the extended adjacency and the corresponding graph Fourier transform and show that different diffusion scales lead to different graph-frequency perspectives. At different scales, the transform discriminates shifted ranges of signal variations across the graph, revealing more information on the graph signal when compared to traditional approaches. The scale-dependent graph Fourier transform is applied for anomaly detection and is shown to outperform the conventional graph Fourier transform.**

***Index Terms*—diffusion distances, diffusion maps, extended adjacency, graph signal processing, scale-dependent graph Fourier transform.**

I. Introduction

LARGE quantities of heterogeneous data are constantly collected by numerous sensors, which are often geographically dispersed. Real networks and their corresponding data come in vastly different shapes and applications, ranging from genetic interaction networks [1] and the human brain [2] to sensor networks and smart cities [3]. The increased connectivity and availability of abundant data calls for methods that can uncover hidden connections between seemingly unrelated things in complex and irregular structures.

Graph signal processing (GSP) explores pairwise relations between signals residing on nodes of a graph [4]–[6]. In GSP, elements of networks are modeled as vertices (or nodes) of a mathematical structure – the graph – and relations between two connected elements are represented by edges [1]–[4]. The dimensionality of the data matches that of the graph, such that each entry is associated with a vertex.

Most GSP tools are functions of a graph-shift operator (GSO) matrix [7]–[9] that encodes relations between the graph nodes. For instance, the graph Fourier transform (GFT) is defined as the signal expansion in terms of the eigenbasis of the GSO. The literature contains several GSO definitions that suit different applications [2]–[20]. The two most commonly used GSOs are the adjacency matrix of the graph [7], and the graph Laplacian [6].

The application dependency of the GSO is related to the more fundamental problem of modeling the original network by a graph; different models have different properties that can be explored by GSP tools [11]–[20]. For a particular network and application, it is desirable to define a GSO that best describes node relations, so that the corresponding network signals can be better analyzed/processed. Consider, for instance, the frequency analysis yielded by the GFT. The spectrum of a graph is directly related to the eigenvalues of the GSO. As a consequence, changes in the GSO entries are reflected in the graph spectrum, possibly allowing the discrimination of different frequency contents of a same network signal. We assume here that adjacency matrices are initially sparse, rendering GSOs with a reduced number of edges. This is due to sparsity constraints commonly imposed upon adjacency matrices or application limitations, e.g., sensors identifying their own neighbors. Moreover, if only the adjacency information is available but accurate network information is unknown, updating or deriving a new GSO becomes a challenging task. This work proposes a method for augmenting an initial adjacency matrix for frequency analysis of networked data.

We derive virtual Markov relations between nodes and incorporate the Markov property into the GSO as a function of diffusion distances (DDs) between network elements. Markov relations occur naturally in some applications, such as in consensus [21]–[23] and random-walk-driven networks [24], [25]. For generic networks, we propose a derivation of the Markov matrix based on the consensus algorithm [21]. DDs are part of the diffusion-maps (DMs) framework introduced in [26]. DMs are applicable to datasets composed by states of high-dimensional data points. These data points can be interpreted as data states as they vary with time. As a function of the Euclidean distance between data points, a Markov matrix can be defined by describing transition probabilities of a random walk between data states. Here, the set of data states can be represented by the nodes of a graph. Note that each node of the graph is associated with an entire data state of the network, such that edges define transition relations between

The work of V. R. M. Elias was supported, in part, by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) — Grant number: 88887.310189/2018-00. The work of Dr. Martins was supported, in part, by the ERC project AGNOSTIC and, in part, by the Brazilian funding agency Faperj. The work of Dr. Werner was supported in part by the Research Council of Norway and Academy of Finland under Grant 296849.

Vitor R. M. Elias and Stefan Werner are with the Department of Electronic Systems, NTNU-Norwegian University of Science and Technology, Trondheim 7491, Norway (e-mail: vitor.elias@ntnu.no and stefan.werner@ntnu.no).

Wallace A. Martins is with the Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg, L-1855 Luxembourg (e-mail: wallace.alvesmartins@uni.lu).

Vitor R. M. Elias and Wallace A. Martins are also with the UFRJ-Federal University of Rio de Janeiro, Rio de Janeiro 21941-972, Brazil.

these high-dimensional data states. This view is in contrast with that of GSP, where edges associate individual network elements. Using eigenvectors of the corresponding Markov matrix, DMs uncover descriptions of the underlying geometry of the dataset [27]–[29]. In this framework, DDs provide a metric for relating two states of data according to the random walk.

We consider elements of the network as nodes of a graph, as in the GSP framework, and the relations between these elements depend on a concept of transition-probability distance, as in the DM framework. The use of DDs yields an augmented version of the initial adjacency model. For instance, DDs relate nodes that are beyond the local reach of the physical neighborhood. This relation depends on how many transition steps of the Markov chain are considered in the computation of the DD. The resulting GSO, called extended-adjacency matrix, depends on the number of transition steps. We demonstrate the benefits of the extended adjacency by implementing a method that uses the proposed model together with the GFT to get a scale-dependent graph-frequency analysis, which we call scale-dependent graph Fourier transform (sGFT). We note that the proposed GSO model can be used with other GSP tools, including other graph-frequency representations. Moreover, the proposed mapping is not restricted to networks that inherently present the Markov property; indeed, it is possible to derive a Markov chain from a generic adjacency matrix as we show in Section V-A.

The combination of DM and GSP has been considered in [30] and [31]. The work in [30] proposes the use of Markov matrices as GSO. In the context of GSP, Markov matrices, of the form in [30], have desirable properties, e.g., they are diagonalizable and the inverse eigenvector matrix can be computed efficiently. The use of Markov matrices also allows DM-related tools, such as dimensionality reduction and clustering, to be incorporated in GSP [30]. Furthermore, the work in [30] studies the similarities between both frameworks, making explicit how some operations from GSP can be interpreted from a DM perspective. For instance, both graph-shifting and graph-filtering operations can be written in terms of embeddings from the DM framework. In [31], a method for graph-signal interpolation is derived using the Nyström extension [32] when employing a Markov matrix as GSO.

The use of DM for classical digital signal processing (DSP) tasks has been studied in different applications in [29]. The authors introduce two filtering schemes that leverage on properties of DMs: *non-local filtering* updates a state $\mathbf{x}_i$ according to the affinity between $\mathbf{x}_i$ and other states $\mathbf{x}_j$, while this affinity is computed by using the Gaussian kernel over the DD between $\mathbf{x}_i$ and $\mathbf{x}_j$; and *graph-based processing* explores subsets of eigenvectors acquired by DMs to extract the desired component of noisy data states. Moreover, the authors in [29] present applications of DMs in single-channel source localization and in the suppression of transient interference for speech enhancement.

In contrast to previous works, we incorporate DDs into the GSP framework and construct a graph model that captures the interaction between elements of the Markov network. Specifically, while the work in [30] proposes a Markov matrix as GSO, we use a Markov matrix only as the starting point of our work. Moreover, the Markov matrix proposed in [30] is not symmetric, whereas we adopt a doubly-stochastic matrix based on the discrete-time consensus algorithm [21]. The proposed model is derived by further implementing concepts of DMs given the initial Markov matrix. Namely, we use the Markov matrix to compute DDs between nodes and generate additional edges. The main contributions of this paper are as follows:

- We propose an extended-adjacency matrix that captures dependencies between non-adjacent nodes of the graph. The model augments the original adjacency using DDs between nodes. We show that the extended-adjacency matrix can be derived for a non-Markov network for which an associated Markov model can be constructed.
- We present a scale-dependent graph Fourier transform (sGFT), as a function of the extended-adjacency matrix, that describes the frequency content of the graph signal. The sGFT reveals the graph frequency versus time, or scale, of the associated Markov chain. The sGFT is applied for anomaly detection using synthetic and real data, and we show that the proposed GSO improves the GFT in the anomaly-detection task when compared to other GSO models.

The rest of the paper is organized as follows. Sections II and III present the fundamentals of graph signal processing and diffusion maps. Section IV introduces the proposed extended-adjacency matrix and scale-dependent graph Fourier transform. In Section V we present numerical experiments that validate the proposed methodology. In Section VI we consider the application of the scale-dependent graph Fourier transform to the problem of anomaly detection in sensor networks. Finally, conclusions are given in Section VII.

II. Graph Signal Processing

This section introduces the notation adopted throughout this work and some fundamental concepts of GSP.

A. Graphs and network modeling

Let a graph be represented by $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{v_1, \ldots, v_N\}$ denotes the set of vertices (or nodes) and $\mathcal{E} = \{e_{11}, \ldots, e_{NN}\}$ denotes the set of edges. Each vertex corresponds to one element of the network [3], [4], [6], [10], [19], [20], [33]. Each edge e_{ij} represents a pairwise connection between nodes v_i and v_j. Edges reflect the relation between elements of the original structure, if this relation exists. Nodes connected by an edge are adjacent nodes. A mapping $w : \mathcal{E} \to \mathbb{C}$ is used to model weighted edges, such that w_{ij} denotes the weight value for edge e_{ij}. A graph is often represented by the *adjacency matrix* $\mathbf{A} \in \mathbb{C}^{N \times N}$, whose (i, j)th element is $A_{ij} = w_{ij}$.

The set of vertices that are adjacent to a vertex v_j is referred to as the neighborhood of v_j. The set $\mathcal{N}_j$ comprises the indexes of vertices in the neighborhood of v_j. For an undirected graph, where $w_{ij} = w_{ji}$, the weighted degree of node v_j is given by $\deg(v_j) = \sum_{i \in \mathcal{N}_j} w_{ij}$ and we define the diagonal *degree matrix* $\mathbf{D}$, such that $D_{jj} = \deg(v_j)$. The graph *Laplacian matrix* of an undirected graph is defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$.

B. Signals over graphs and GSP

Data collected from or generated by elements of the networks can be viewed as a *graph signal*. A complex-valued graph signal is given by the mapping $s : \mathcal{V} \to \mathbb{C}$. We represent a graph signal as a vector $\mathbf{s} \in \mathbb{C}^N$, where the ith entry s_i is given by $s(v_i)$, i.e., the signal component at vertex v_i. The graph signal represents a snapshot of the network state.

The graph shift operator (GSO), denoted by $\mathbf{S}$, is an $N \times N$ matrix employed to generate the graph-shifted signal $\tilde{\mathbf{s}} = \mathbf{S}\mathbf{s}$. That is, the graph-shifted signal on node v_i is a combination of signals s_j, given by $\tilde{s}_i = \sum_{j=1}^{N} S_{ij} s_j$. One choice of $\mathbf{S}$, presented in [10], is the adjacency matrix $\mathbf{A}$. This choice is partly motivated by direct analogy with discrete-time processing of periodic signals. In this case, the resulting graph-shifted signal at node v_i is a local combination of the signal in its neighborhood.

Another popular choice for $\mathbf{S}$ is the graph Laplacian, $\mathbf{L}$, which is a local difference operator. The choice is motivated by graph spectral theory [34]. Motivations and applications using the graph Laplacian as GSO are thoroughly reviewed in [6]. As we show in Section II-C, the eigenvectors of the GSO compose the Fourier basis in graph domain. This construction of the Fourier basis has a particular link with conventional DSP, since the classical Fourier transform may be interpreted as the expansion of a continuous-time function in terms of complex exponentials, which are the eigenfunctions of the one-dimensional Laplace operator.

The aforementioned approaches yield local operators, i.e., $S_{ij} > 0$, for $i \neq j$, if and only if $A_{ij} \neq 0$ [6], [10]. We note that other matrices than $\mathbf{A}$ or $\mathbf{L}$ can be used as GSO, and the choice of the GSO depends on the application at hand [12], [13]. In this work, we propose non-local GSOs that extend the initial adjacency relations between nodes. Once a shift is defined, many results and techniques from conventional DSP theory can be extended to graph domain, e.g., convolution, filtering, transforms (Fourier and wavelets), and spectral analysis.

C. Graph Fourier transform

We assume a graph with real-valued weighted edges. Consider the diagonalizable GSO matrix $\mathbf{S} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1}$, where $\mathbf{\Lambda}$ is a diagonal matrix whose entries are the eigenvalues λ_i of $\mathbf{S}$, and $\mathbf{U}$ has as columns the eigenvectors $\mathbf{u}_i$ of $\mathbf{S}$. The GFT coefficients of a graph signal $\mathbf{s}$ are obtained from the analysis equation [6], [10]

$$\hat{\mathbf{s}} = \mathbf{U}^{-1}\mathbf{s}, \tag{1}$$

and the graph signal is recovered from the synthesis equation

$$\mathbf{s} = \mathbf{U}\hat{\mathbf{s}}. \tag{2}$$

The eigenvalues λ_i of $\mathbf{S}$, with $i \in \{1, \dots, N\}$, correspond to the graph-frequency spectrum. The eigenvectors $\mathbf{u}_i$, with $i \in \{1, \dots, N\}$, are the graph-frequency components [6], [10], [35]. In this work, we adopt the graph Laplacian as GSO, i.e., we set $\mathbf{S} = \mathbf{L}$. We show next that larger eigenvalues of the graph Laplacian correspond to higher graph frequencies. For this purpose, consider a variation metric for signal $\mathbf{x} \neq \mathbf{0}$ on $\mathcal{G}$ given by

$$\begin{aligned}\nu(\mathbf{x}) &= \frac{\mathbf{x}^{\mathrm{T}}\mathbf{L}\mathbf{x}}{\mathbf{x}^{\mathrm{T}}\mathbf{x}} && (3)\\ &= \frac{\mathbf{x}^{\mathrm{T}}\big(\mathbf{D}-\mathbf{A}\big)\mathbf{x}}{\mathbf{x}^{\mathrm{T}}\mathbf{x}} \\ &= \frac{\sum\limits_{i\neq j} A_{ij}\big(x_i - x_j\big)^2}{\mathbf{x}^{\mathrm{T}}\mathbf{x}}, && (4)\end{aligned}$$

which measures the total difference between the signal values on different vertices, weighted by the edge values. Equation (3) is the Rayleigh quotient of $\mathbf{L}$, which is bounded below and above by the extreme eigenvalues of $\mathbf{L}$, $\lambda_1 = 0$ and λ_N, respectively. As the GFT is defined as the expansion of a signal over the eigenvectors of $\mathbf{L}$, λ_2 and λ_N correspond, respectively, to the smallest and the largest non-zero variations, or graph frequencies. The eigenvalue $\lambda_1 = 0$ corresponds to frequency equal to zero, associated with a constant graph signal. The eigenvalue λ_2 is called *graph spectral gap*, and λ_N is called *graph spectral radius*.

As indicated by (4), the graph frequencies provide information on how fast a signal varies across the vertices. In this context, the signal is a single snapshot of the network state. High frequency means that the signal sample on a given vertex differs considerably from samples on neighboring vertices. Low frequency means that the graph signal is smooth across all nodes. Here, we highlight a fundamental motivation for our work: the adjacency model of a network directly affects its graph-frequency analysis, given the definition of the graph spectrum and its dependency on the elements of the adjacency matrix, as shown in (4). However, this implication of (4) is often neglected when adjacency models are constructed. Thus, we aim for a model that is capable of capturing node relations while taking into account its influence on GSP applications.

III. Diffusion Maps

This section introduces the basics of diffusion maps (DMs) and diffusion distances (DDs) [26], necessary for the development of the proposed extended adjacency and sGFT.

Let $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_K] \in \mathbb{R}^{L\times K}$ be a data matrix with K data points, also called states, each of dimension L. For example, matrix $\mathbf{X}$ can describe the evolution of the state $\mathbf{x}_k$ of a network with L elements for time instants $k \in \{1, \dots, K\}$. It is assumed that there is an underlying (hidden) process that relates the different data points, possibly driving the way data is generated. However, note that there is no underlying graph associated with $\mathbf{X}$. The objective of the DM framework is to make this underlying process explicit [26], [29], [30].

A. Construction of the similarity graph

The first step when constructing a DM is to create a graph that associates data points $\mathbf{x}_i$ with nodes, and quantifies their interrelationship [26]. In contrast to the GSP framework, this association is merely an alignment of the data with the structure. That is, the data point is not treated as a signal on the node, but rather as the node itself. In order to make the contrast between the GSP and DM approaches clear, we highlight that, throughout this section, the number of data states K corresponds to the number of nodes in the graph constructed for the DM framework.

Let $\mathcal{X} \subset \mathbb{R}^L$ represent the dataset that contains the columns of $\mathbf{X}$. The edges are created through a symmetric kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$, i.e., $k(\mathbf{x}, \mathbf{y}) = k(\mathbf{y}, \mathbf{x}) \geq 0$. The obtained graph is undirected and possibly weighted, and serves as a preliminary geometric description of the data based on the underlying driving process [26]. The choice of the kernel depends on the application. For instance, a common choice is the radial-basis-function (RBF) kernel

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma_{\mathrm{RBF}}^2}\right), \tag{5}$$

where $\sigma_{\mathrm{RBF}} > 0$ is a free parameter that controls the bandwidth of the kernel.

The RBF kernel expresses a relationship based on the affinity between data points $\mathbf{x}_i$ and $\mathbf{x}_j$, in terms of the Euclidean distance. Once a metric for similarity is established for the data points, an adjacency matrix $\mathbf{A} \in \mathbb{R}^{K \times K}$ can be defined with entries $A_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$. The corresponding degree matrix is $\mathbf{D} \in \mathbb{R}^{K \times K}$ such that $D_{ii} = \sum_{j \in \mathcal{N}_i} k(\mathbf{x}_i, \mathbf{x}_j)$.

B. Construction of the random walk

To capture how data is influenced by an underlying process, a random walk on the data is defined. The idea is to characterize how one state of the high-dimensional data transitions into another state [26]. For this purpose, the similarity between two data points is normalized as [26]

$$p(\mathbf{x}_j|\mathbf{x}_i) = \frac{k(\mathbf{x}_i, \mathbf{x}_j)}{\sum\limits_j k(\mathbf{x}_i, \mathbf{x}_j)} = \frac{A_{ij}}{D_{ii}}, \tag{6}$$

where $p(\mathbf{x}_j|\mathbf{x}_i)$ is interpreted as the transition probability from $\mathbf{x}_i$ to $\mathbf{x}_j$, which establishes a Markov chain. Using matrix notation, the Markov chain can be described in terms of a right-stochastic matrix $\mathbf{M} = \mathbf{D}^{-1}\mathbf{A}$, commonly referred to as a Markov matrix, with entries $M_{ij} = p(\mathbf{x}_j|\mathbf{x}_i)$. Taking t steps of the random walk is captured by $\mathbf{M}^t$, i.e., the (i, j)th entry of $\mathbf{M}^t$ gives the transition probability, denoted by $p_t(\mathbf{x}_j|\mathbf{x}_i)$, from $\mathbf{x}_i$ to $\mathbf{x}_j$ in t steps. The probability $p_t(\mathbf{x}_j|\mathbf{x}_i)$ considers all possible paths composed of t edges that connect $\mathbf{x}_i$ to $\mathbf{x}_j$, including self-loops. The probability in (6) is the same as $p_t(\mathbf{x}_j|\mathbf{x}_i)$ for $t = 1$.

Consider the decomposition of $\mathbf{M}$ in terms of its right and left eigenvectors $\boldsymbol{\psi}_k$ and $\boldsymbol{\phi}_k$ and the eigenvalues γ_k, with $k \in \{1, \ldots, K\}$.[1] The transition probabilities can be written as [26]

$$p_t(\mathbf{x}_j|\mathbf{x}_i) = \sum_{k=1}^{K} \gamma_k^t \psi_{k,i} \phi_{k,j}. \tag{7}$$

The eigenvalues of a right-stochastic matrix satisfy $|\gamma_k| \leq 1$. Assuming the graph is connected, the Markov chain is irreducible and $\gamma_1 = 1$ [26]. We adopt the ordering $\gamma_1 = 1 > |\gamma_2| \geq \cdots \geq |\gamma_K|$. Consequently, a random walk driven by these transition probabilities has an asymptotic behavior governed by γ_1 [26], i.e.,

$$\lim_{t\to\infty} p_t(\mathbf{x}_j|\mathbf{x}_i) = \phi_{1,j}, \tag{8}$$

where $\phi_{1,j}$ is the jth entry of the first left eigenvector of $\mathbf{M}$, $\boldsymbol{\phi}_1$, normalized as $\|\boldsymbol{\phi}_1\|_1 = 1$. In other words, $\phi_{1,j}$ is the asymptotic probability of reaching state $\mathbf{x}_j$ from any initial state. As the graph is connected, this quantity is non-zero for every j, as given by (7). Note that the right eigenvector $\boldsymbol{\psi}_1$, associated with $\gamma_1 = 1$, is a constant vector, as $\mathbf{M} \cdot \mathbf{1} = \mathbf{1}$.

C. Diffusion distances

The *diffusion distance* (DD) at a certain diffusion, or time, scale t [26] is a metric for the (inverse) affinity between two data points as a function of transition probabilities, and is given by

$$D_t^2(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^{K} \frac{\left(p_t(\mathbf{x}_k|\mathbf{x}_i) - p_t(\mathbf{x}_k|\mathbf{x}_j)\right)^2}{\phi_{1,k}}. \tag{9}$$

The DD extends local relations, in terms of adjacency and direct similarity between nodes, into a global metric by assimilating probabilities of diffusion paths [27]. If two points have similar posterior distributions, they are well connected through the end-points, indexed by k, of these distributions. This means that if there are high-probability paths between two data points, they are considered to be close in terms of diffusion distance, even if not adjacent. Conversely, the diffusion distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ is large when the probability of reaching $\mathbf{x}_j$ from $\mathbf{x}_i$ is small even by considering non-direct paths through $\mathbf{x}_k$. An alternative interpretation for the diffusion distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ is to consider a limited amount of energy placed on both nodes. This energy is then diffused through the network for t diffusion steps according to $\mathbf{M}^t$. Finally, the distance is computed as the difference between the contributions from these two nodes to the network, in the sense of how much energy was diffused to each node. If the two points have similar contribution to a given node, it means that they are well connected through that node. Note that the diffusion distance depends on t, which serves as a diffusion-scale parameter. An increased t equals more steps of the random walk, which corresponds to a larger-scale diffusion over the network.

IV. GSP for Markov Networks

The DM framework in [26] treats the entire network signal as a state of a Markov chain and generates a Markov-based graph over these states. In contrast, we adapt the diffusion distances to develop a comprehensive network model by treating the original network itself as the Markov-based graph. That is, the states of the Markov chain are no longer associated with the data (entire network state), but with the vertices (individual network elements).

We introduce the extended-adjacency matrix, which captures node collaboration of Markov networks. Some common cases of these networks are consensus networks [21]–[23],

[1]Not every Markov matrix is diagonalizable, but $\mathbf{M} = \mathbf{D}^{-1}\mathbf{A} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}})\mathbf{D}^{\frac{1}{2}}$ is, for it is similar to the symmetric matrix $\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$.

conservative diffusion networks [18], [36], [37], and random-walk driven networks [24], [25]. Moreover, we introduce the sGFT and analyze the spectral behavior of the graph through the perspective of the extended-adjacency model.

A. Extended adjacency

The networks considered here are initially constrained by adjacency rules of the network topology. For example, the connections between nodes in a wireless sensor network (WSN) depend on their communication capabilities, usually dictated by physical distance. Furthermore, the sum of the weights of the edges connecting a node to its neighbors is equal to unity, yielding a stochastic adjacency. Nodes communicate directly with their immediate neighbors. However, nodes that are not initially connected can be related to each other through collaboration. In other words, network adjacency is associated with one step of the collaboration process, such that the network operates on the data through iterative multiplications of the stochastic matrix. Similarly to the conventional adjacency, which depends on the physical distance between nodes, we develop the extended adjacency based on diffusion distances.

Let the network be represented by a connected graph $\mathcal{G} = \{\mathcal{V}, \mathbf{B}\}$ with a symmetric, irreducible, and stochastic adjacency matrix $\mathbf{B}$ with positive real edges. In analogy with the theory presented in Section III, this matrix is equivalent to $\mathbf{M}$ and the states of the Markov chain are the nodes of the graph. The diffusion distance between nodes v_i and v_j of the graph is given by

$$D_t^2(v_i, v_j) = \sum_{n=1}^{N} \frac{\left(B_{in}^{(t)} - B_{jn}^{(t)}\right)^2}{(1/N)}, \tag{10}$$

where $(1/N)$ corresponds to the elements of the first left eigenvector $\mathbf{q}_1$ of $\mathbf{B}$, as in (9), and $B_{ij}^{(t)}$ denotes the (i, j)th entry of $\mathbf{B}^t$. Since we assume $\mathbf{B}$ symmetric, its left and right eigenvectors are the same and $\mathbf{q}_1 = (1/N)\mathbf{1}$. Similar to the DM framework, the DDs between network elements depends on the scale t. The metric in (10) expresses distances between nodes, including nodes that are not neighbors as defined by $\mathbf{B}$. We derive similarity from diffusion distances in similar manner as conventional adjacency matrices are derived from geographic distances. The extended-adjacency matrix $\bar{\mathbf{A}}(t)$ is such that

$$\bar{A}_{ij}(t) = \begin{cases} B_{ij} + \exp\left(-\frac{D_t^2(v_i,v_j)}{\rho N}\right) & i \neq j \\ 0 & i = j, \end{cases} \tag{11}$$

where $\rho > 0$ is a free parameter and N is the size of the network. The term B_{ij} in (11) guarantees that original edges are maintained, whereas the RBF term is responsible for extending the adjacency. The term ρN makes the argument of the RBF kernel independent of the network size (cf. (10)). The range of the kernel output can be adjusted for different applications according to the free parameter ρ. Moreover, although the extended adjacency is defined for $\rho > 0$, it is possible to obtain B_{ij}, with $i \neq j$, through (11) by making $\rho \to 0^+$. That is, the original adjacency is a particular case of the extended adjacency.

Although traditional GSOs are local with respect to network connections, we note that this property is not present in the proposed adjacency model. A local GSO offers a straightforward visualization of the physical structure of the graph and facilitates the implementation of distributed GSP algorithms. However, in many applications, the definition of locality is unknown, or a local GSO fails to model implicit node relations. In this work, we aim to derive a model that is not restricted by locality assumptions and is useful for networks where non-adjacent nodes interact. Hence, we propose a non-local model that offers a trade-off between locality and representation of node interactions. As we show in the next section, this trade-off is indirectly controlled via the diffusion-scale parameter. In this same context, other desirable features inherent to specific graph structures might be lost after the implementation of the extended adjacency. For instance, the correspondence between the GFT over a ring graph and the conventional discrete Fourier transform (DFT) is lost when the extended-adjacency matrix is considered. Our proposal, however, is aimed at graphs whose topologies are not well-structured or perfectly known and we show that some GSP tools can benefit from the proposed model.

Moreover, the proposed extended-adjacency matrix models a graph with no self-loops. Although this assumption is also common in the GSP framework, we note that it is not necessary. Further modifications of $\bar{\mathbf{A}}(t)$ are possible, such as imposing an upper bound on edge values, or applying a threshold on the matrix values to enforce sparsity.

B. Analysis of the extended-adjacency matrix

The resulting extended-adjacency matrix $\bar{\mathbf{A}}(t)$ is a symmetric matrix with positive entries that depend on the diffusion-scale t and bandwidth ρ.

Proposition 1. *The DD given in* (10) *is a non-increasing function of the diffusion scale t.*

Proof. The symmetric matrix $\mathbf{B}$ can be decomposed as $\mathbf{B} = \sum_{l=1}^{N} \sigma_l \mathbf{q}_l \mathbf{q}_l^{\mathrm{T}}$, where σ_l and $\mathbf{q}_l$, with $l \in \{1, \dots, N\}$, are the eigenvalues and orthonormal eigenvectors of $\mathbf{B}$. We can write

$$B_{in}^{(t)} - B_{jn}^{(t)} = \sum_{l=1}^{N} \sigma_l^t (q_{l,i} - q_{l,j}) q_{l,n}. \tag{12}$$

Substituting (12) into (10), we have

$$D_t^2(v_i, v_j) = N \sum_{l=1}^{N} \sum_{m=1}^{N} \sigma_l^t \sigma_m^t (q_{l,i} - q_{l,j})(q_{m,i} - q_{m,j}) \zeta_{l,m}, \tag{13}$$

where $\zeta_{l,m} = \sum_{n=1}^{N} q_{l,n} q_{m,n} = \delta(l - m)$ and $\delta(\cdot)$ is the Kronecker delta function. Thus,

$$D_t^2(v_i, v_j) = N \sum_{l=1}^{N} \sigma_l^{2t} (q_{l,i} - q_{l,j})^2. \tag{14}$$

Since $(q_{l,i} - q_{l,j})^2 \geq 0$ and $\sigma_l^2 \in [0, 1]$, then $D_t^2(v_i, v_j) \geq D_{t+1}^2(v_i, v_j)$, $\forall t \geq 1$. $\square$

In accordance with Proposition 1, the following corollary and lemma can be established:

Corollary 1. *Edge weights are non-decreasing with t according to* (11)*. Assuming that an edge exists only if its weight exceeds a given threshold, the number of edges is also non-decreasing with increasing t. In other words, increasing t for a fixed ρ possibly creates new edges, given the reduction in the DD.*

Lemma 1. *The entries of the asymptotic extended-adjacency matrix* $\bar{\mathbf{A}} = \lim_{t\to\infty} \bar{\mathbf{A}}(t)$ *are given by*

$$\bar{A}_{ij} = \begin{cases} B_{ij} + 1 & i \neq j \\ 0 & i = j, \end{cases} \tag{15}$$

and correspond to those of an adjacency matrix of a complete graph without self-loops. That is, each node v_i is connected to every node v_j in the graph, with $i \neq j$, by an edge of value $1 + B_{ij}$.

Proof of Lemma 1. From (14), $t \to \infty$ implies that the diffusion distance between any two vertices tends to zero, since σ_l^{2t} tends to zero for $l \neq 1$ and the only remaining non-zero term σ_l^{2t} corresponds to $l = 1$, for which $(q_{1,i} - q_{1,j}) = 0$, since $\mathbf{q}_1 = (1/N)\mathbf{1}$. With $D_t^2(v_i, v_j) = 0$, we have $\bar{A}_{ij}(t) = B_{ij} + 1$, $\forall i \neq j$, and $\bar{A}_{ii}(t) = 0$, with $i, j \in \{1, \ldots, N\}$. □

C. Scale-dependent graph Fourier transform

Similar to the extended-adjacency matrix $\bar{\mathbf{A}}(t)$, the definition of other graph-related matrices and the graph Fourier transform also depend on the scale t. For each diffusion scale, there is a corresponding Laplacian matrix defined as

$$\bar{\mathbf{L}}(t) = \bar{\mathbf{D}}(t) - \bar{\mathbf{A}}(t), \tag{16}$$

where $\bar{\mathbf{D}}(t)$ is the diagonal degree matrix associated with $\bar{\mathbf{A}}(t)$. Given the eigendecomposition $\bar{\mathbf{L}}(t) = \bar{\mathbf{U}}(t)\bar{\mathbf{\Lambda}}(t)\bar{\mathbf{U}}^{\mathrm{T}}(t)$, we define the scale-dependent graph Fourier analysis of signal $\mathbf{x}$ as

$$\hat{\mathbf{x}}(t) = \bar{\mathbf{U}}^{\mathrm{T}}(t)\mathbf{x}, \tag{17}$$

where, in contrast to the conventional GFT, the coefficients $\hat{\mathbf{x}}(t)$ depend on the diffusion-scale t. The scale-dependent graph Fourier synsthesis equation is given by

$$\mathbf{x} = \bar{\mathbf{U}}(t)\hat{\mathbf{x}}(t). \tag{18}$$

Remark 1. *Note that the analysis and synthesis equations result from the use of the conventional GFT together with the proposed extended-adjacency model. Thus,* (17) *and* (18) *do not establish a novel transform. However, we will refer to* (17) *as the scale-dependent graph Fourier transform (sGFT) for simplicity throughout the text.*

D. Analysis of the sGFT

Now, we analyze how the proposed adjacency model affects the graph-frequency analysis. Recall that graph frequencies are directly associated with the eigenvalues of the graph Laplacian, as shown in (4). Hence, the analysis is conducted in terms of which graph frequencies are discriminated by the Laplacian eigenvalues for varying diffusion scales. For this purpose, we determine bounds for the spectral gap and spectral radius of the graph in function of t. Let the eigenvalues of the Laplacian be $\bar{\lambda}_1(t), \bar{\lambda}_2(t), \ldots, \bar{\lambda}_N(t)$, in ascending order with $\bar{\lambda}_1(t) = 0$. The spectral gap is $\bar{\lambda}_2(t)$ and the spectral radius is $\bar{\lambda}_N(t)$, and both depend on t. Let θ_2 denote the smallest non-zero eigenvalue of $\mathbf{L_B}$, the Laplacian of $\mathbf{B}$, and θ_N denote the maximum eigenvalue of $\mathbf{L_B}$. Moreover, let $\mathbf{L_C}$ denote the Laplacian of the unweighted complete graph with N nodes, whose entries are given by

$$L_{\mathbf{C}_{ij}} = \begin{cases} -1 & i \neq j \\ N - 1 & i = j. \end{cases} \tag{19}$$

Proposition 2. *For a graph with N nodes, if t is increased, the range of graph-frequencies discriminated by the sGFT shifts into higher frequencies. Asymptotically, the sGFT discriminates graph-frequency ranges up to the interval* $[N + \theta_2, N + \theta_N]$.

Proof. The eigenvalues of the Laplacian of a connected graph are non-decreasing with the addition of new edges in the graph [38]. As follows from Corollary 1, edges can only be added, and not removed, as t increases. Consequently, the spectral gap and radius are non-decreasing for increasing t.

As follows from Lemma 1, the asymptotic extended adjacency $\bar{\mathbf{A}}$ corresponds to that of a complete graph. The resulting graph Laplacian is $\mathbf{L}_{\bar{\mathbf{A}}} = \mathbf{L_B} + \mathbf{L_C}$. Given the structure of $\mathbf{L_C}$ and the fact the $\mathbf{L_B}$ is symmetric, these matrices commute and the eigenvalues of $\mathbf{L_B} + \mathbf{L_C}$ are given by the sums of eigenvalues of each matrix. Eigenvalues of $\mathbf{L_C}$ are: 0 with multiplicity 1 and the eigenvalue N with multiplicity $N - 1$. Consequently, for $\mathbf{L}_{\bar{\mathbf{A}}}$, the spectral gap achieves the value $N + \theta_2$, and the spectral radius achieves $N + \theta_N$. □

As t increases, the interval of graph frequencies discriminated by the sGFT is shifted into higher frequencies. We note that the maximum frequency discriminated by the sGFT, $N + \theta_N$, matches the largest possible variation, according to (4), of signals defined over the proposed adjacency matrix.

The sGFT is a frequency-analysis tool tailored for each stage of the Markov chain. From a node-collaboration perspective, the effect of node collaboration on the graph spectrum can be interpreted in an intuitive manner: if more steps of collaboration are taken, more edges are introduced. Consequently, variations in graph signals are observed by additional node pairs and are perceived as larger frequencies. Hence, by incorporating node collaboration into the graph model, we provide a frequency-analysis tool that reveals more information on the network signal than that offered by the conventional GFT.

Remark 2. *The proposed implementation of the Fourier analysis based on the extended adjacency adds a degree of freedom for tools that use the graph-spectrum. Therefore, some applications may require training data along with a hyperparameter-training method, such as grid-search and cross-validation, for determining the adequate diffusion scale.*

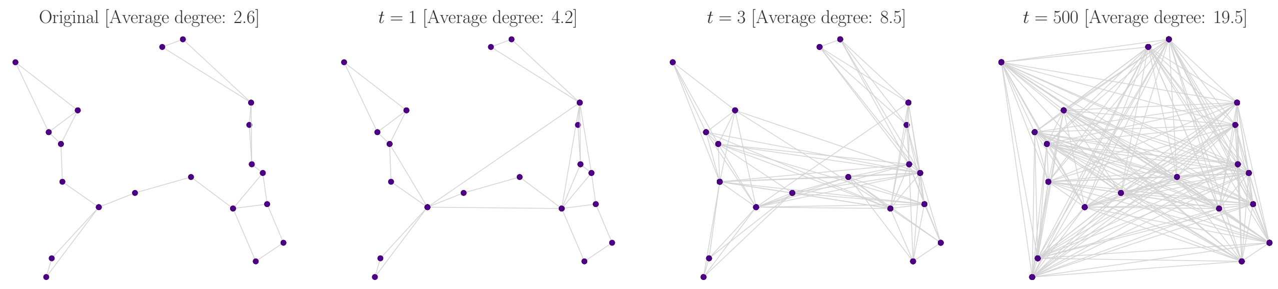

Original [Average degree: 2.6]
$t = 1$ [Average degree: 4.2]
$t = 3$ [Average degree: 8.5]
$t = 500$ [Average degree: 19.5]

Fig. 1. Connectivity versus diffusion scale. Only edges of $\bar{\mathbf{A}}(t)$ that exceed 30% of the highest edge value are shown.

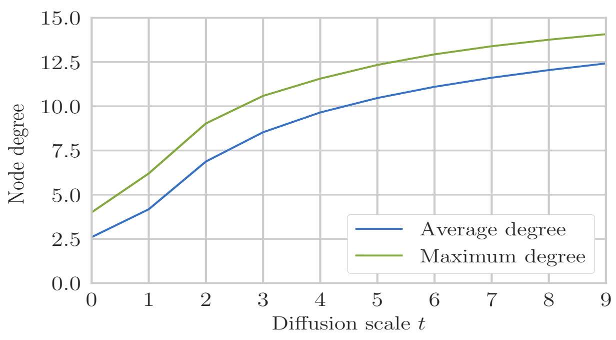


Fig. 2. Average and maximum degrees versus diffusion scale t. For $t = 0$, the values correspond to those of the original network.

V. Numerical Examples

In this section we verify through numerical examples the analyses conducted in Section IV-B and Section IV-D.

A. Nearest-neighbors and consensus network

We consider a sensor network with N sensors that collectively estimate a common parameter through collaborations, more specifically through consensus averaging [21], [22].

In an average consensus algorithm, the global average of initial sensor states, $x_i[0]$, with $i \in \{1, \ldots, N\}$, is computed in a distributed fashion through local computations and local message exchanges. More specifically, sensor v_i implements the following iterative algorithm

$$x_i[k+1] = P_{ii} x_i[k] + \sum_{j \in \mathcal{N}_i} P_{ij} x_j[k], \quad k \in \mathbb{N} \tag{20}$$

where P_{ij} are weights given to local and neighbor node values. In the case of sensor networks, the neighborhood $\mathcal{N}_i$ is usually defined by sensor nodes within the transmission radius of sensor v_i. Equation (20) can be written as

$$\mathbf{x}[k+1] = \mathbf{P}\mathbf{x}[k], \tag{21}$$

where $\mathbf{x}[k] = [x_1(k) \ldots x_N(k)]^{\mathrm{T}}$. If $\mathbf{P}$ is doubly stochastic, then the sensor states will converge to $(1/N)\mathbf{1}^{\mathrm{T}} \cdot \mathbf{x}[0]$.

Let the network be modeled by an unweighted and undirected graph $\mathcal{G} = \{\mathcal{V}, \mathbf{A}\}$ with graph Laplacian $\mathbf{L}$. One particular form of $\mathbf{P}$, which leads to Laplacian based consensus, is given by

$$\mathbf{P} = \mathbf{I} - \epsilon\mathbf{L}, \tag{22}$$

where ϵ is the consensus step size. In this work, we set $\epsilon = 1/(1.25\Delta_{\max})$, where $\Delta_{\max}$ is the maximum degree in $\mathcal{G}$ (more information on convergence of consensus algorithms and the choice of ϵ can be found in [21]). Thus, we are able to associate the network with a stochastic matrix $\mathbf{P}$, since $\mathbf{P} \cdot \mathbf{1} = \mathbf{1} - \epsilon\mathbf{L} \cdot \mathbf{1} = \mathbf{1}$.

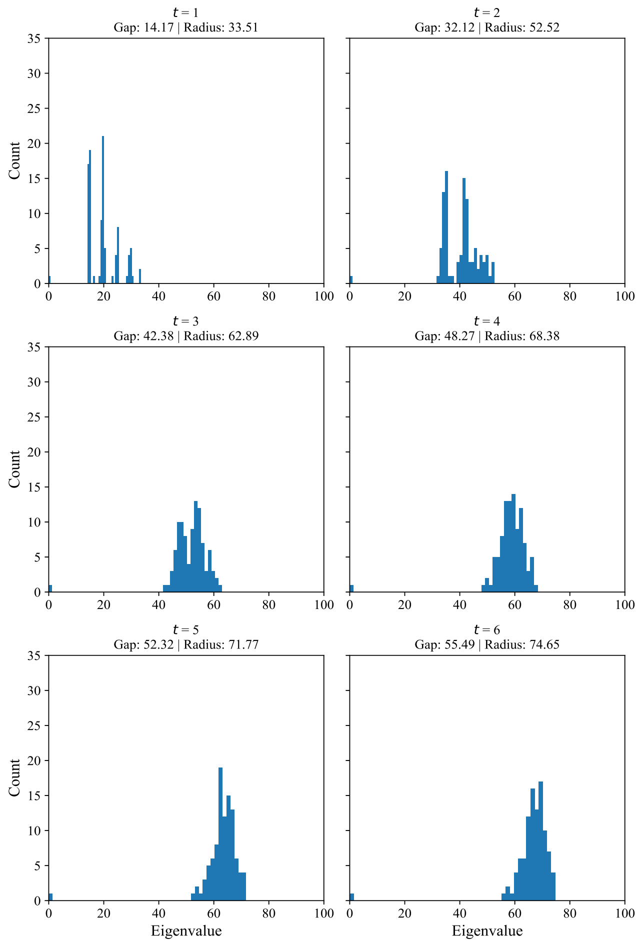


Fig. 3. Histogram of eigenvalues of diffusion Laplacian matrices $\bar{\mathbf{L}}(t)$, for $t \in \{1, \ldots, 6\}$.

B. Increasing connectivity

Given matrix $\mathbf{P}$, the diffusion distances between sensors, with $\mathbf{B} = \mathbf{P}$, and the extended-adjacency matrix $\bar{\mathbf{A}}(t)$ are given by (10) and (11), respectively. We consider a sensor network with $N = 20$ sensors, with an average node degree equal to 2.6, and $\rho = 0.35$, Fig. 1 shows the connections of the original network and the new edges introduced at larger diffusion scales. As expected, the network becomes more connected tending to a complete graph as t grows large. Fig. 2 shows the increase in average and maximum degrees versus diffusion scale. This experiment illustrates the effects of changing diffusion scales presented in Proposition 1 and Lemma 1.

C. Spectrum analysis

For a network with $N = 100$ nodes, we conduct an experimental analysis of the sGFT spectrum for different diffusion scales, while the DDs are computed for $\rho = 0.4$. Fig. 3 shows histograms of the eigenvalues of the graph Laplacian $\bar{\mathbf{L}}(t)$ versus diffusion scale. An increase in the diffusion scale yields an increase in the spectral gap (Proposition 2), from around 14 for $t = 1$ up to around 55 for $t = 6$, and in the spectral radius, from around 34 up to over 74. At each scale, the sGFT yields a different spectrum according to the number of steps of node collaboration. In contrast, the conventional GFT yields a fixed spectrum. The additional information provided by the sGFT can benefit applications that make decisions based on spectrum-related features, such as classifiers and detectors.

Numerical results indicate that the new eigenvectors $\bar{\mathbf{u}}_i(t)$, with $i \in \{1, \ldots, N\}$, virtually preserve the notion of smoothness with respect to the initial Laplacian matrix $\mathbf{L}$. For this analysis, consider $\mathbf{v} = [\mathbf{u}_1^{\mathrm{T}}\mathbf{L}\mathbf{u}_1 \ldots \mathbf{u}_N^{\mathrm{T}}\mathbf{L}\mathbf{u}_N]$, where $\mathbf{u}_i$, with $i \in \{1, \ldots, N\}$, are the eigenvectors of $\mathbf{L}$, and $\bar{\mathbf{v}}(t) = [\bar{\mathbf{u}}_1(t)^{\mathrm{T}}\mathbf{L}\bar{\mathbf{u}}_1(t) \ldots \bar{\mathbf{u}}_N(t)^{\mathrm{T}}\mathbf{L}\bar{\mathbf{u}}_N(t)]$. We assess the cosine similarity between $\bar{\mathbf{v}}(t)$ and $\mathbf{v}$ given by

$$\mathrm{c}_{\bar{\mathbf{v}}(t)\mathbf{v}} = \frac{(\bar{\mathbf{v}}(t) - \mu_{\bar{\mathbf{v}}(t)})^{\mathrm{T}}(\mathbf{v} - \mu_{\mathbf{v}})}{\|\bar{\mathbf{v}}(t) - \mu_{\bar{\mathbf{v}}(t)}\|_2 \|\mathbf{v} - \mu_{\mathbf{v}}\|_2}, \tag{23}$$

where $\mu_{\mathbf{v}}$ denotes the mean of $\mathbf{v}$. For a total of 1000 random graphs, with N in the interval $[20, 200]$ and number of neighbors in the interval $[2, 16]$, we obtain an average value for $\mathrm{c}_{\bar{\mathbf{v}}(1)\mathbf{v}}$ of 0.92 when $t = 1$. This value decreases slightly as the scale increases: $\mathrm{c}_{\bar{\mathbf{v}}(2)\mathbf{v}} = 0.90$ and $\mathrm{c}_{\bar{\mathbf{v}}(3)\mathbf{v}} = 0.86$.

VI. Application

In this section, we illustrate how the sGFT can be used for anomaly detection in synthetic and real networks. The application is motivated by increasing connectedness of real-world elements [39]–[42], which demands security and reliability in networks [43]–[49]. The free parameters of the sGFT allow for a tailored frequency decomposition when constructing an anomaly detector based on spectral information of the network state. We compare results from detectors based on the sGFT with detectors based on other different GSO approaches.

Given an initial adjacency matrix $\mathbf{A}$, the proposed method is compared to the GFT based on the eigenvectors of the corresponding Laplacian and to the GFT using the eigenvectors of the Markov GSO as proposed in [30]. Additionally, a scale-dependent GSO model based on shortest-path distances is implemented and used for comparison. This GSO depends on the length of the shortest-paths as follows: for $\mathbf{A}$, the shortest-paths and corresponding distances from each node to the rest of the graph are computed using Dijkstra's algorithm [52]. Once the paths are computed, a GSO can be constructed by connecting each node to other nodes reachable through shortest-paths no longer than a given length, with edge-weights equal to the inverse of the shortest-path distances. Note that, given an unweighted adjacency matrix, the GSO based on shortest-paths for lengths up to 1 is equal to $\mathbf{A}$.

In the following simulation results and figures, the GSO approaches are coded as follows:

- GFT: conventional GFT using the graph Laplacian;
- DF1: sGFT using scale $t = 1$;
- DF2: sGFT using scale $t = 2$;
- SP2: GFT using shortest-path-based GSO with paths up to length equal to 2 hops;
- SP3: GFT using shortest-path-based GSO with paths up to length equal to 3 hops;
- MRK: GFT using the eigenvectors of the Markov matrix from [30].

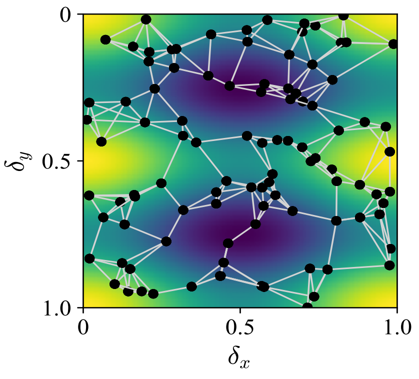
A. Anomaly detection task

We construct classifiers based on the graph-spectral information generated by the sGFT and the conventional GFT along with the aforementioned GSO-construction methods. The approach for constructing the anomaly detector is similar to those in [35], [50], [51]. In particular, [35] and [50] compute the respective high-frequency components from the adjacency matrix and the graph Laplacian. On the other hand, in [51], the Laplacian eigenvectors are obtained from a constrained optimization problem that enforces properties not considered here, such as sparsity. More specifically, assuming that smoothness is expected in the healthy signal, we apply a high-pass filter with cut-off frequency λ_{cut} and conduct the classification based on the filtered coefficients. If one of the coefficients exceeds a threshold τ, the signal is classified as anomalous. Consider a training dataset $\mathcal{X}_{\mathrm{train}} = \{\mathcal{X}_{\mathrm{H}}, \mathcal{X}_{\mathrm{A}}\}$, where $\mathcal{X}_{\mathrm{H}}$ and $\mathcal{X}_{\mathrm{A}}$ indicate the healthy and anomalous parts of the training dataset, respectively, with "healthy" indicating a signal free from anomalies. Elements of these sets are graph signals, that is, vectors with length equal to the number of nodes. The detection threshold is determined as follows:

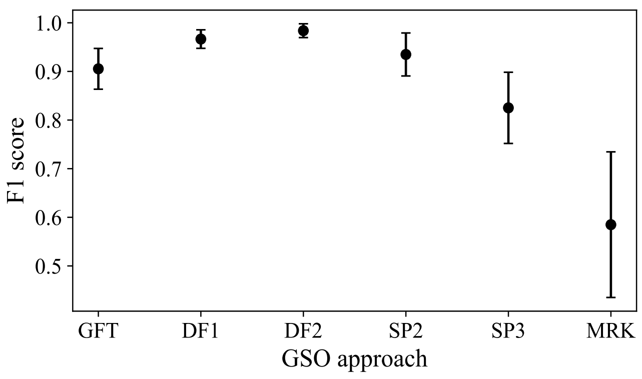
1) graph-frequency coefficients are computed for each healthy signal in $\mathcal{X}_{\mathrm{H}}$;
2) high-pass filter is applied, so that coefficients corresponding to graph-frequencies higher than λ_{cut} are kept;
3) for each signal, we select a partial τ_{p} that corresponds to the largest coefficient after filtering;
4) Once τ_{p} is computed for all signals, the detection threshold is computed as:

$$\tau = \mu_{\tau_{\mathrm{p}}} + \alpha\sigma_{\tau_{\mathrm{p}}}, \tag{24}$$

where $\mu_{\tau_{\mathrm{p}}}$ is an estimation of the average value and $\sigma_{\tau_{\mathrm{p}}}$ is an estimation of the standard deviation of all partial



(a)



(b)

Fig. 4. Experiment 1 – setup and results: (a) spatial distribution of the sensors and their interconnections plotted over a snapshot of the observed signal; and (b) f1-scores achieved by each of the GSO-construction approaches.

τ_p computed. The non-negative parameter α scales a confidence factor associated with the standard deviation of the partial thresholds.

Using the conventional GFT, with the graph Laplacian, and the GFT from the Markov matrix, λ_cut and α are the parameters that require training. The scale-dependent approaches, however, have more free parameters: the path length for the shortest-path-based approach, and both the diffusion scale t and the normalization parameter ρ for the sGFT. Two different diffusion scales $t \in \{1,2\}$ are tested separately, as well as two maximum path lengths equal to 2 and 3. Parameters λ_cut, α, and ρ are optimized via grid-search. That is, a set of pre-determined values is given for each parameter and all combinations are tested according to some metric. We choose the f1 score as metric, which is given by

$$\mathrm{f1} = \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}, \tag{25}$$

where

$$\mathrm{precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}, \tag{26}$$

and

$$\mathrm{recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}, \tag{27}$$

with TP indicating true positives (correctly classifying anomaly as anomaly), FP indicating false positives (wrongly classifying healthy signal as anomaly), and FN indicating false negatives (wrongly classifying anomaly as healthy signal). Moreover, we conduct cross-validation with 5 folds, such that the training data is split into 5 sets and, for each combination of parameters, the training (computing τ) is performed over 4 sets and the metric is evaluated over the remaining set. The combination of parameters yielding the best average result across the folds is selected. Our simulations are conducted in Python and we use the *GridSearchCV* class (which performs the grid-search with cross-validation) from the *scikit-learn* library to train the classifier.

Once the classifiers are trained, their performance is assessed by measuring the f1 score achieved over a test dataset $\mathcal{X}_\mathrm{test}$. All simulation results presented in the next subsections are averaged over 50 random training and test datasets, which are specified according to each simulation.

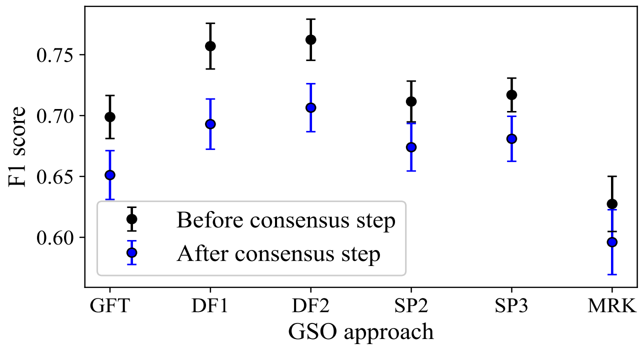
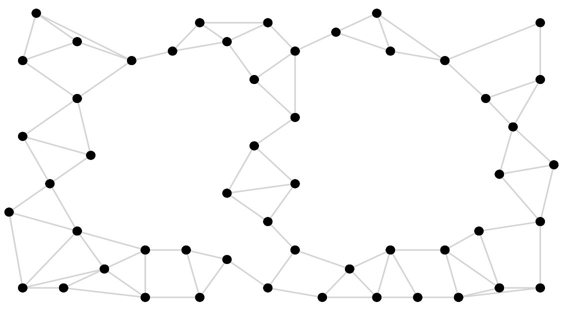


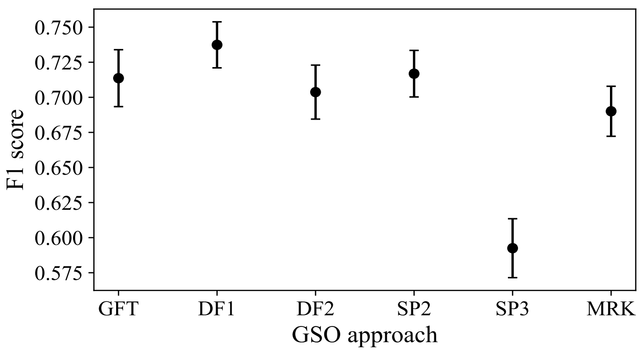
Fig. 5. F1 scores achieved by each GSO approach for the two cases of Experiment 2.

B. Simulations over synthetic networks

1) Experiment 1: spatially-spread anomaly: We consider first $N = 100$ sensors randomly distributed in the square space $[0,1] \times [0,1]$. A network is constructed by connecting each sensor to its 4 nearest-neighbors, and the corresponding adjacency matrix $\mathbf{A}$ is known. The sensors measure a spatially-smooth wave signal given by $s(\delta_x, \delta_y) = \cos(2\pi\delta_x + \theta_x) + \cos(2\pi 2\delta_y + \theta_y)$, where $\delta_x, \delta_y \in [0,1]$ are, respectively, the horizontal and vertical spatial coordinates and θ_x and θ_y are varying phase values uniformly and independently sampled from $[0, 2\pi]$. The graph structure and a snapshot of the signal s are depicted in Fig. 4a.

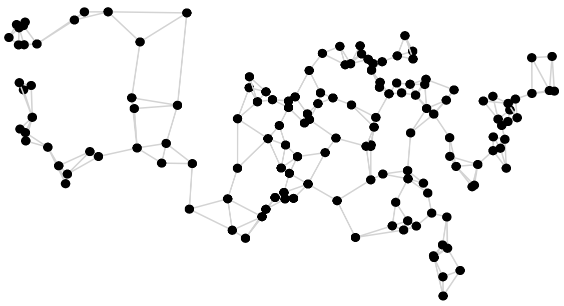
We consider the problem of detecting an additive (space-wise) high-frequency interference signal given by $s_i(\delta_x, \delta_y) = 0.1\,(\cos(2\pi 5\delta_x + \theta_x) + \cos(2\pi 6\delta_y + \theta_y))$. Training and test datasets, $\mathcal{X}_\mathrm{train}$ and $\mathcal{X}_\mathrm{test}$, have 150 healthy samples and 150 anomalous samples each. Results for the f1 score achieved over the 50 independent runs are presented in Fig. 4b. Results show that the detector based on the spectral information provided by the extended-adjacency matrices outperform detectors based on other GSO approaches. Additionally, using

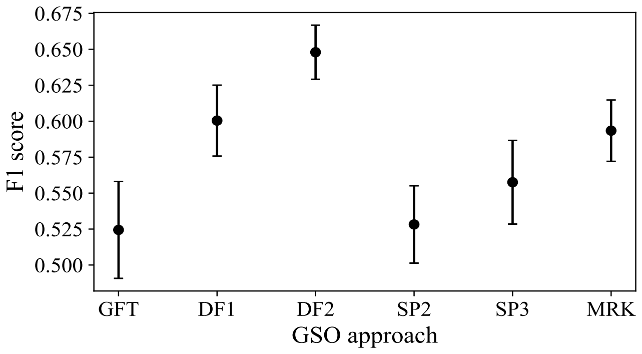

(a)



(b)

Fig. 6. Experiment 3 – setup and results: (a) graph structure for the Intel Lab dataset; and (b) f1 scores achieved by each of the GSO-construction approaches.


(a)



(b)

Fig. 7. Experiment 4 – setup and results: (a) graph structure for the GSOD dataset; and (b) f1 scores achieved by each of the GSO-construction approaches.

the Markov matrix alone offered worse results than using the Laplacian of the original adjacency matrix.

2) Experiment 2: anomaly/attack on few sensors: Consider now a network with $N = 150$ sensors in the same square space $[0,1] \times [0,1]$, each connected to its 6 nearest neighbors, that measure healthy signals $\mathbf{x} \sim \mathscr{N}(20 \cdot \mathbf{1}, 0.4 \cdot \mathbf{I})$. Up to two sensors are randomly selected as anomalous, i.e., the network will have one or two out of 150 sensors with anomalous data. This selection is conducted independently for each snapshot, such that the anomalous sensors of a given graph signal do not depend on the anomalous sensors of other signals. Each anomalous sensor measures a random value $20 \pm b$, with b drawn uniformly from $\{1,2,3,4,5\}$ and independently for each sensor and each signal. The possible values for b are heuristically chosen so that the anomaly is sufficiently strong to be detected, but still not be trivially detected, by all approaches.

We assume the anomaly is present in the initial data measured by the network and that the network performs consensus over the data according to (20). This makes the anomalies smoother as they are diffused according to the consensus algorithm. We conduct the anomaly-detection task before the consensus step is taken, and also after the consensus step is taken. These simulations are independent of each other. For each case, training and test datasets have 200 data samples each, of which 100 are anomalous samples, and experiments are run for 50 independent randomizations of these datasets.

Fig. 5 shows results for the f1 scores obtained by the best classifiers over the test datasets. Results show that the sGFT achieves better detection scores than using the other GSO approaches, before and after consensus. In both cases, the detectors based on the shortest-path approaches outperform the conventional GFT, whereas using the eigenvectors of the Markov matrix yields the worst results for both cases.

C. Simulations over real networks

Two real networks are used and their structures are presented in Fig. 6a and Fig. 7a. For both cases, sensors' positions and healthy data are extracted from the available databases [53], [54]. Networks used in this example are basic sensor networks for which a Markovian relation is not initially defined, and we construct a Markovian relation between sensors as in (22).

For these databases, we assume that all data available are healthy. Thus, anomaly must be manually introduced in the data. For Experiments 3 and 4, this is conducted with minor differences from the method described for anomaly construction in Experiment 2. The anomaly is given by an additive Gaussian noise with non-zero mean over the healthy data. The mean value of the noise is allowed to vary according to a discrete uniform distribution, assuming non-zero integer values in the interval $[-b_{\max}, +b_{\max}]$. Similar to the synthetic case, a maximum possible number of anomalous sensors is defined and specified next for each simulation. For Experiment 5, we generate an anomaly similar to the one used in Experiment 1. The anomaly is given interference signal given by $s_i(\delta_x, \delta_y) = 5\left(\cos(2\pi 0.1\delta_x + \theta_x) + \cos(2\pi 0.1\delta_y + \theta_y)\right)$. For all cases, training is conducted in similar manner as that employed for the synthetic data, with 5 folds in cross validation.

1) Experiment 3: Intel lab data [53] - sensor malfunction: The network is modeled as a κNN sensor network with $\kappa = 3$ according to the available sensor positions. We use temperature data, in degrees Celsius, from 52 of the 54 sensors measured between 00 AM and 07 AM during the week from March 01, 2003 to March 05, 2003. Sensors named 5 and 15 are not used due to unavailable measurements. Healthy data range from 13.6 °C to 21.2 °C. The anomaly is described by $b_{\max} = 3$ °C, noise variance equal to 0.4 °C^2, and up to 2 anomalous sensors

The total number of available samples from the database is 373. For each independent run, 350 samples are randomly selected, of which half receive the anomaly. The 350 samples are then equally split into training and test datasets.

Results for the Intel lab data are presented in Fig 6b. In this case, the conventional approach is competitive against the scale-dependent methods. Still, the detectors based on the extended-adjacency for scale $t = 1$ outperform all other approaches. Moreover, using the eigenvectors of the Markov matrix also yields competitive results, with average f1 score approximately 0.02 behind the conventional approach.

2) Experiment 4: Global Surface Summary of the Day (GSOD) [54] - sensor malfunction: The database provides measurements from weather stations distributed across the territory of the United States of America. We use temperature measurements, converted from degrees Fahrenheit to degrees Celsius, obtained during the year 2010 by 150 randomly-selected stations from the conterminous United States (excluding Alaska, Hawaii, and other off-shore insular areas) in order to keep the graph connected. Network structure is derived from available stations' latitudes and longitudes. Healthy data range from -29.4 °C to 38.6 °C. We use $\kappa = 3$, $b_{\max} = 7$ °C, noise variance equal to 1 °C^2, and up to 7 anomalous sensors.

Daily samples are available, for a total of 365 signals. From these, we randomly select 350 signals for each run and generate training and test datasets as in Experiment 3.

In Fig. 7b, results show that the proposed approach outperforms the oher approaches for both scales $t = 1$ and $t = 2$, while the latter offers the best result. Here, the Markov-matrix-based approach outperforms the ones using the conventional Laplacian and the shortest-path-based GSOs.

3) Experiment 5: GSOD - spatially-spread anomaly: Using the GSOD data and network, we simulate the presence of a spatially-spread additive interference signal, given by $s_i(\delta_x, \delta_y) = 5\left(\cos(2\pi 0.1\delta_x + \theta_x) + \cos(2\pi 0.1\delta_y + \theta_y)\right)$, where δ_x and δ_y correspond to weather station's longitude and latitude, respectively, and with θ_x and θ_y randomly sampled from $[0, 2\pi]$. For each independent experiment, the interference is constrained to a randomly selected interval of 5 degrees of longitude, such that only the sensors in that interval are anomalous. Training and test datasets are generated as in Experiment 4.

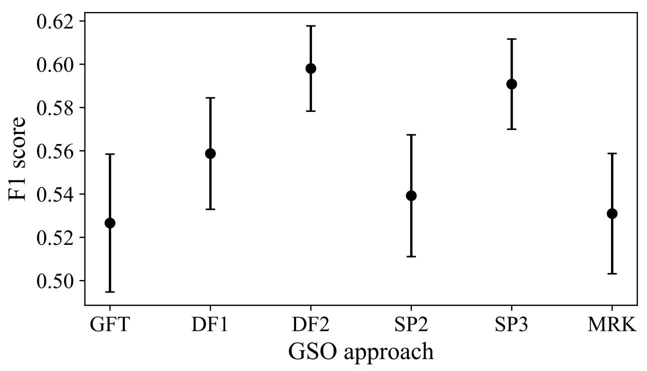


Fig. 8. F1 scores achieved by each GSO approach for Experiment 5.

Results are presented in Fig. 8 and show that the GFT considerably benefits from the proposed model when compared to the conventional approach. Moreover, the approach based on shortest-paths, for maximum length equal to 3, exhibits competitive results, while the approach based on the Markov matrix yields results similar to those of the conventional approach.

VII. Conclusion

We proposed the extended-adjacency matrix, which incorporates relations between non-adjacent nodes on a certain diffusion scale or time. We use the extended adjacency to augment the modeling of node relations in order to improve the efficiency of GSP tools. We also presented the scale-dependent graph Fourier transform for data defined over these networks, by using the conventional GFT together with the proposed scale-dependent model. We showed that increasing the diffusion scale results in an increased connectivity in the network, such that each different scale for the sGFT yields a different perspective of graph frequency, as a tailored connectivity is considered. We developed a theoretical analysis that shows that changing the diffusion scale shifts the spectral range yielded by the sGFT and corroborated the analysis with numerical experiments. Tools that operate on the graph spectrum can leverage on the additional information. For instance, we employed the sGFT for anomaly detection in synthetic and real networks. We used the free parameters of the sGFT to conduct a frequency analysis tailored for the given network. The proposed method was compared to the GFT based on the conventional graph Laplacian and to the GFT based on other augmented GSO models, and results showed that anomaly detectors based on the sGFT achieved better results than the other approaches.

References

[1] B. Boucher and S. Jenna, "Genetic interaction networks: better understand to better predict," *Front. in Genet.*, vol. 4, pp. 1–16, Dec. 2013.

[2] W. Huang, T. A. W. Bolton, J. D. Medaglia, D. S. Bassett, A. Ribeiro, and D. Van De Ville, "A graph signal processing perspective on functional brain imaging," *Proc. IEEE*, vol. 106, pp. 868–885, May 2018.

[3] I. Jablonski, "Graph signal processing in applications to sensor networks, smart grids, and smart cities," *IEEE Sensors J.*, vol. 17, pp. 7659–7666, Dec. 2017.

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

[5] 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 Process. Mag.*, vol. 31, pp. 80–90, Sep. 2014.

[6] 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 Process. Mag.*, vol. 30, pp. 83–98, May 2013.

[7] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs: Graph filters," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, May 2013, pp. 6163–6166.

[8] O. Teke and P. P. Vaidyanathan, "Extending classical multirate signal processing theory to graphs - Part I: Fundamentals," *IEEE Trans. Signal Process.*, vol. 65, pp. 409–422, Jan. 2017.

[9] O. Teke and P. P. Vaidyanathan, "Extending classical multirate signal processing theory to graphs - Part II: M-channel filter banks," *IEEE Trans. Signal Process.*, vol. 65, no. 2, pp. 423–437, Jan. 2017.

[10] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs," *IEEE Trans. Signal Process.*, vol. 61, pp. 1644–1656, Apr. 2013.

[11] B. Girault, "Stationary graph signals using an isometric graph translation," in *Proc. 23rd Eur. Signal Process. Conf.*, Aug. 2015, pp. 1516–1520.

[12] B. Girault, P. Goncalves, and E. Fleury, "Translation on graphs: An isometric shift operator," *IEEE Signal Process. Lett.*, vol. 22, pp. 2416–2420, Dec. 2015.

[13] A. Gavili and X.-P. Zhang, "On the shift operator, graph frequency, and optimal filtering in graph signal processing," *IEEE Trans. Signal Process.*, vol. 65, pp. 6303–6318, Dec. 2017.

[14] T. Summers, I. Shames, J. Lygeros, and F. Dorfler, "Topology design for optimal network coherence," in *Proc. Eur. Control Conf.*, July 2015, pp. 575–580.

[15] S. Shahrampour and V. M. Preciado, "Topology identification of directed dynamical networks via power spectral analysis," *IEEE Trans. Autom. Control*, vol. 60, pp. 2260–2265, Aug. 2015.

[16] M. Babakmehr, M. G. Simoes, M. B. Wakin, and F. Harirchi, "Compressive sensing-based topology identification for smart grids," *IEEE Trans. Ind. Informat.*, vol. 12, pp. 532–543, Apr. 2016.

[17] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst, "Learning laplacian matrix in smooth graph signal representations," *IEEE Trans. Signal Process.*, vol. 64, pp. 6160–6173, Dec. 2016.

[18] D. Thanou, X. Dong, D. Kressner, and P. Frossard, "Learning heat diffusion graphs," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 3, pp. 484–499, Sep. 2017.

[19] G. B. Giannakis, Y. Shen, and G. V. Karanikolas, "Topology identification and learning over graphs: Accounting for nonlinearities and dynamics," *Proc. IEEE*, vol. 106, pp. 787–807, May 2018.

[20] G. Mateos, S. Segarra, A. G. Marques, and A. Ribeiro, "Connecting the dots: Identifying network structure via graph signal processing," *IEEE Signal Process. Mag.*, vol. 36, pp. 16–43, May 2019.

[21] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proc. IEEE*, vol. 95, pp. 215–233, Jan. 2007.

[22] J. Qin, Q. Ma, Y. Shi, and L. Wang, "Recent advances in consensus of multi-agent systems: A brief survey," *IEEE Trans. Ind. Electron.*, vol. 64, pp. 4972–4983, June 2017.

[23] B. Kailkhura, S. Brahma, and P. K. Varshney, "Data falsification attacks on consensus-based detection systems," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 3, pp. 145–158, Mar. 2017.

[24] L. Lovasz, "Random walks on graphs: A survey," *Combinatorics, Paul Erdos is Eighty*, vol. 2, pp. 1–46, 1993.

[25] C. Gkantsidis, M. Mihail, and A. Saberi, "Random walks in peer-to-peer networks," in *Proc. IEEE Conf. Comput. Commun.*, Mar. 2004, vol. 1, pp. 120–130.

[26] R. R. Coifman and S. Lafon, "Diffusion maps," *Appl. Comput. Harmon. Anal.*, vol. 21, pp. 5–30, July 2006.

[27] R. R. Coifman, S. Lafon, A. B. Lee, M. Maggioni, B. Nadler, F. Warner, and S. W. Zucker, "Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps," *Proc. Natl. Acad. Sci.*, vol. 102, pp. 7426–7431, May 2005.

[28] R. R. Coifman, S. Lafon, A. B. Lee, M. Maggioni, B. Nadler, F. Warner, and S. W. Zucker, "Geometric diffusions as a tool for harmonic analysis and structure definition of data: Multiscale methods," *Proc. Natl. Acad. Sci.*, vol. 102, pp. 7432–7437, May 2005.

[29] R. Talmon, I. Cohen, S. Gannot, and R. R. Coifman, "Diffusion maps for signal processing: A deeper look at manifold-learning techniques based on kernels and graphs," *IEEE Signal Process. Mag.*, vol. 30, no. 4, pp. 75–86, June 2013.

[30] A. Heimowitz and Y. C. Eldar, "A unified view of diffusion maps and signal processing on graphs," in *Proc. 12th Int. Conf. on Sampl. Theory Appl.*, July 2017, pp. 308–312.

[31] A. Heimowitz and Y. C. Eldar, "The Nystrom extension for signals defined on a graph," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Apr. 2018, pp. 4199–4203.

[32] C. Fowlkes, S. Belongie, F. Chung, and J. Malik, "Spectral grouping using the Nyström method," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, pp. 214–225, Feb. 2004.

[33] J. Mei and J. M. F. Moura, "Signal processing on graphs: causal modeling of *un*structured data," *IEEE Trans. Signal Process.*, vol. 65, pp. 2077–2092, Apr. 2017.

[34] F. R. K. Chung, *Spectral Graph Theory*. American Mathematical Society, 1997.

[35] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs: Frequency analysis," *IEEE Trans. Signal Process.*, vol. 62, pp. 3042–3054, June 2014.

[36] O. Teke and P. P. Vaidyanathan, "Time estimation for heat diffusion on graphs," in *Proc. 51st Asilomar Conf. Signals, Syst. Comput.*, Oct. 2018, pp. 1963–1967.

[37] N. Masuda, M. A. Porter, and R. Lambiotte, "Random walks and diffusion on networks," *Phys. Rep.*, vol. 716-717, pp. 1–58, Nov. 2017.

[38] B. Mohar, "The Laplacian spectrum of graphs", in *Graph Theory, Combinatorics, and Applications*, vol. 2, Wiley, 1991, pp. 871–898

[39] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Comput. Netw.*, vol. 54, pp. 2787–2805, Oct. 2010.

[40] B. Rashid and M. H. Rehmani, "Applications of wireless sensor networks for urban areas: A survey," *J. Net. Comput. Appl.*, vol. 60, pp. 192–219, Jan. 2016.

[41] E. Fadel, V. Gungor, L. Nassef, N. Akkari, M. A. Malik, S. Almasri, and I. F. Akyildiz, "A survey on wireless sensor networks for smart grid," *Comput. Commun.*, vol. 71, pp. 22–33, Nov. 2015.

[42] W. Ren, R. W. Beard, and E. M. Atkins, "Information consensus in multivehicle cooperative control," *IEEE Control Syst. Mag.*, vol. 27, pp. 71–82, Apr. 2007.

[43] M. A. Mahmood, W. K. Seah, and I. Welch, "Reliability in wireless sensor networks: A survey and challenges ahead," *Comput. Netw.*, vol. 79, pp. 166–187, 2015.

[44] B. Mostefa and G. Abdelkader, "A survey of wireless sensor network security in the context of internet of things," in *4th Int. Conf. Inf. Commun. Techol. Disaster Manag.*, Dec 2017, pp. 1–8.

[45] S. Kar and J. Moura, "Distributed consensus algorithms in sensor networks: Quantized data and random link failures," *IEEE Trans. Signal Process.*, vol. 58, pp. 1383–1400, Mar. 2010.

[46] I. Kayes and A. Iamnitchi, "Privacy and security in online social networks: A survey," *Online Soc. Netw. Media*, vol. 3-4, pp. 1–21, Oct. 2017.

[47] Y. Chen, S. Kar, and J. M. F. Moura, "Resilient distributed estimation through adversary detection," *IEEE Trans. Signal Process.*, vol. 66, pp. 2455–2469, May 2018.

[48] Y. Chen, S. Kar, and J. M. Moura, "The internet of things: Secure distributed inference," *IEEE Signal Process. Mag.*, vol. 35, pp. 64–75, Sep. 2018.

[49] Y. Chen, S. Kar, and J. M. F. Moura, "Resilient distributed estimation: Sensor attacks," *IEEE Trans. Autom. Control*, pp. 1–1, 2018.

[50] E. Drayer and T. Routtenberg, "Detection of false data injection attacks in power systems with graph Fourier transform," in *Proc. IEEE Global Conf. Signal. Inf. Process.*, Nov. 2018, pp. 890–894.

[51] M. Khatua, S. H. Safavi, and N. M. Cheung, "Sparse Laplacian component analysis for internet traffic anomalies detection," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 4, no. 4, pp. 697–711, Dec. 2018.

[52] D.E. Knuth, "A generalization of Dijkstra's algorithm", *Inf. Signal Lett.*, vol 6, pp 1–5, Feb. 1997.

[53] "Intel Lab Data." http://db.csail.mit.edu/labdata/labdata.html. Accessed: 2019-11-28.

[54] "Global Surface Summary of the Day - GSOD - Data.gov." https://catalog.data.gov/dataset/global-surface-summary-of-the-day-gsod. Accessed: 2019-11-28.

Vitor R. M. Elias received the B.Sc. degree in Electronics and Computer Engineering from the Federal University of Rio de Janeiro (UFRJ) in 2013. In 2015, he received the M.Sc. degree in Electrical Engineering from the COPPE/UFRJ. He is currently pursuing a joint Ph.D. degree at the UFRJ and the Norwegian University of Science and Technology (NTNU), where he was a visiting researcher from 2019 to 2020. He is currently with the Signals, Multimedia, and Telecommunications Lab (SMT/UFRJ) and the Signal Processing Group in the Department of Electronic Systems at NTNU. His experience and research interests include image and video compression, machine learning, adaptive learning, and digital signal processing on graphs.

Wallace A. Martins (SM'20) received the Electronics Engineer degree from the Federal University of Rio de Janeiro (UFRJ, Brazil) in 2007, and both M.Sc. and D.Sc. degrees in Electrical Engineering also from UFRJ in 2009 and 2011, respectively. He was a Research Visitor at University of Notre Dame (USA, 2008), at Université de Lille 1 (France, 2016), and at Universidad de Alcalá (Spain, 2018). From 2010 to 2013 he was an Associate Professor of the Federal Center for Technological Education (CEFET/RJ, Brazil). Since 2013 he has been with the Department of Electronics and Computer Engineering (DEL/Poli) and Electrical Engineering Program (PEE/COPPE) at UFRJ, where he is presently a tenured Associate Professor. He is currently a researcher working with the Interdisciplinary Centre for Security, Reliability and Trust (SnT) at Université du Luxembourg. His research interests are in the fields of digital signal processing, especially adaptive signal processing and graph signal processing, as well as digital communications, with focus on equalization and precoding for wireless communications. Dr. Martins has authored more than 60 technical papers in refereed journals and conferences, and 1 book. Also, he received the Best Student Paper Award from EURASIP at EUSIPCO-2009, Glasgow, Scotland, and the 2011 Best Brazilian D.Sc. Dissertation Award from Capes.

Stefan Werner (SM'07) received the M.Sc. degree in electrical engineering from the Royal Institute of Technology, Stockholm, Sweden, in 1998 and the D.Sc. degree (Hons.) in electrical engineering from the Signal Processing Laboratory, Helsinki University of Technology, Espoo, Finland, in 2002. He is currently a Professor in the Department of Electronic Systems, Norwegian University of Science and Technology, Trondheim, Norway. He is also an Adjunct Professor with Aalto University in Finland, and an Adjunct Senior Research Fellow with the Institute for Telecommunications Research, University of South Australia. He was a visiting Melchor Professor with University of Notre Dame during summer 2019, and was holding an Academy Research Fellowship, funded by the Academy of Finland, from 2009 to 2014. His research interests include adaptive and statistical signal processing, signal processing for communications, and security and privacy in cyberphysical systems. He is a member of the editorial boards for the EURASIP Journal of Signal Processing and the IEEE Transactions on Signal and Information Processing over Networks.

Chapter 10

Publications on Learning over Graphs using Graph Kernel LMS

P5 [2020] IEEE. Reprinted, with permission, from V. C. Gogineni, V. R. M. Elias, W. A. Martins, and S. Werner, "Graph diffusion kernel LMS using random Fourier features," in *Asilomar Conference on Signals, Systems, and Computers*, pp. 1–5, Nov. 2020.

P6 V. R. M. Elias, V. C. Gogineni, W. A. Martins, and S. Werner, "Adaptive graph filters in reproducing kernel Hilbert spaces: Design and performance analysis," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 7, pp. 62–74, 2021.

Graph Diffusion Kernel LMS using Random Fourier Features

Vinay Chakravarthi Gogineni [†], Vitor R. M. Elias[*‡], Wallace A. Martins[‡§], Stefan Werner[*]
[†] Department of Machine Intelligence, SimulaMet, Simula Research Laboratory-Oslo, Norway
[*] Department of Electronic Systems, NTNU – Norwegian University of Science and Technology, Trondheim
[‡] Electrical Engineering Program, UFRJ – Federal University of Rio de Janeiro
[§] Interdisciplinary Centre for Security, Reliability and Trust, UniLu – University of Luxembourg

Abstract—**This work introduces kernel adaptive graph filters that operate in the reproducing kernel Hilbert space. We propose a centralized graph kernel least mean squares (GKLMS) approach for identifying the nonlinear graph filters. The principles of coherence-check and random Fourier features (RFF) are used to reduce the dictionary size. Additionally, we leverage the graph structure to derive the graph diffusion KLMS (GDKLMS). The proposed GDKLMS requires only single-hop communication during successive time instants, making it viable for real-time network-based applications. In the distributed implementation, usage of RFF avoids the requirement of a centralized pre-trained dictionary in the case of coherence-check. Finally, the performance of the proposed algorithms is demonstrated in modeling a nonlinear graph filter via numerical examples. The results show that centralized and distributed implementations effectively model the nonlinear graph filters, whereas the random-feature-based solutions are shown to outperform coherence-check based solutions.**

I. Introduction

Recently, graph signal processing (GSP) has received increased attention due to its wide applicability to model, process, and analyze network signals and large data sets [1]–[4]. For instance, in the context of a wireless sensor network, graph nodes and edges represent sensors and communication links, respectively. Similar to conventional digital signal processing (DSP) techniques, the basic building block in GSP is the graph-shift operation, which captures node interconnections. In the particular case of linear networks, the graph-shifted signal on a given node is a linear combination of adjacent node signals, where the weights relate to the edge values. The development of tools for GSP has been extensively studied over the last few years [1]–[9].

A key area of GSP research is to model the unknown relations between input and output graph signals through a filter [1], [3], [7], [8], [10]. The application of linear shift-invariant filter models is widely employed in the literature, e.g., to design graph spectral filters [7], [10] and model dynamic graph signals [8], [9]. More recently, several works deal with adaptive learning of graph filters, see, e.g., [11]–[15]. However, linear models cannot accurately model many real-world systems that exhibit more sophisticated input-output relations. Prominent examples include the relations between air pressure and temperature, and wind speed and generated power in wind turbines [16], [17].

This work was partly supported by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) — Grant number: 88887.310189/2018-00, CNPq, ERCIM Alain Bensoussan Fellowship Programme, ERC project AGNOSTIC, FAPERJ, and the Research Council of Norway.

In this work, we introduce the concept of nonlinear adaptive filtering of graph signals. Adaptive filtering in reproducing kernel Hilbert spaces (RKHS) has proven to be an effective method for modeling nonlinear relations [18]–[28]. Therefore, drawing upon the ideas of kernel methods, we propose graph kernel adaptive filters that effectively capture the nonlinear input-output relations of graph signals. We first derive the centralized graph kernel least mean square (GKLMS) for nonlinear graph system identification. To tackle the growing dimension problem in GKLMS, we first consider a coherence-check approach to construct a fixed-size dictionary. However, this approach requires a centralized pre-trained dictionary and, therefore, does not render an efficient distributed implementation. To overcome this issue, using random Fourier features [29], we propose centralized GKLMS in RFF space. By extending the concepts of distributed learning over networks [12], [25], [30]–[32], we also propose a graph diffusion KLMS (GDKLMS) using RFF that solely depends on local information exchange. Furthermore, we establish the conditions for the mean convergence of the proposed RFF based GDKLMS. Finally, numerical experiments are conducted to demonstrate the performance of the proposed algorithms.

II. Problem Formulation

Consider an undirected graph $\mathcal{G} = \{\mathcal{N}, \mathcal{E}\}$, where $\mathcal{N} = \{1, 2, \ldots, K\}$ is the set of nodes and $\mathcal{E}$ is the set of edges such that $(k, l) \in \mathcal{E}$ if and only if nodes k and l are connected. The graph is equipped with the graph shift operator, defined by a symmetric matrix $\mathbf{S} \in \mathbb{R}^{K\times K}$ whose entries $[\mathbf{S}]_{k,l} = s_{kl}$ take non-zero values only if $(k, l) \in \mathcal{E}$ [1], [2]. The graph Laplacian matrix [1] and the graph adjacency matrix [2] are the common choices for $\mathbf{S}$. At time-index n, a graph signal is defined by the mapping $x(n) : \mathcal{N} \to \mathbb{R}$ and represented by a vector $\mathbf{x}(n) = [x_1(n)\, x_2(n) \ldots x_K(n)]^{\mathrm{T}}$, where $x_k(n)$ represents the signal value at the kth node. The graph shift operation $\mathbf{S}\mathbf{x}(n)$ is performed locally at each node k by linearly combining the samples from neighboring nodes, namely, $\sum_{l\in\mathcal{N}_k} s_{kl} x_l(n)$, where $\mathcal{N}_k$ denotes the neighborhood of node k including k itself.

A length-L linear shift-invariant (LSI) graph filter linearly combines these shifted versions of a graph signal and yields an output $\mathbf{y}(n) = \sum_{i=0}^{L-1} h_i \mathbf{S}^i \mathbf{x}(n-i)$, for $n \geq L-1$, where $h_0, h_1, \ldots, h_{L-1}$ are the coefficients of the graph filter [12]. This model embeds time-evolution and is an alternative to the initial linear graph filter designs [10]–[12]. By retaining the samples $\{x_k(n), [\mathbf{S}\mathbf{x}(n-1)]_k, \ldots, [\mathbf{S}^{L-1}\mathbf{x}(n-L+1)]_k\}$, only one graph shift operation needs to be performed at each time-instant n, which makes this model suitable for real-time applications. For this model, linear graph diffusion LMS strategies have been proposed in [12] for adaptive graph signal processing.

However, in many real-world scenarios, the limited capabilities of linear models fail to represent systems with more sophisticated input-output relations reasonably [18]. This limitation on linear models is discussed in many problems such as channel regression and time-series prediction [18], [26], [28]. Adopting a nonlinear model proved to be effective when tackling this class of problems. In this context, at every node k, we assume a nonlinear relation between the input and output as given below:

$$y_k(n) = f(\mathbf{r}_k(n)) + \upsilon_k(n), \tag{1}$$

where $f : \mathbb{R}^L \to \mathbb{R}$ is a nonlinear function on $\mathbb{R}^L$, $\upsilon_k(n)$ is the observation-noise at node k, and

$$\mathbf{r}_k(n) = [x_k(n)\ [\mathbf{S}\mathbf{x}(n-1)]_k\ \ldots\ [\mathbf{S}^{L-1}\mathbf{x}(n-L+1)]_k]^{\mathrm{T}}. \tag{2}$$

Here, the goal is to estimate the function $f(\cdot)$ at each node k given a set of data pairs $\{\mathbf{r}_k(i), y_k(i)\}$ for $i \in \{1, 2, \ldots, n\}$; this refers to a nonlinear system identification task. While a linear filter can be uniquely defined by its coefficients $h_0, h_1, \ldots, h_{L-1}$, the characterization of a nonlinear filter admits a range of approaches. Several methods exist in literature to estimate the non-linear functions in an adaptive fashion [18], [28], [33]. Of these, kernel methods take a linear form in high-dimensional feature space and, thus, gained much popularity in modeling the nonlinear input-output relations [18]–[25], [27]. Thus, we characterize the nonlinear relations on graphs as graph kernel adaptive filters.

III. Graph Kernel Filters

In order to estimate the nonlinear function $f(\cdot)$ in (1), kernel methods first map the input regressor $\mathbf{r}_k(i) \in \mathbb{R}^L$ into a high-dimensional feature space where $f(\cdot)$ takes a linear form [18], [26]. This mapping is denoted by $\kappa(\cdot, \mathbf{r}_k(i))$, where $\kappa(\cdot,\cdot)$ is a reproducing kernel. The reproducing kernel $\kappa(\cdot,\cdot) : \mathbb{R}^L \times \mathbb{R}^L \to \mathbb{R}$ satisfies [18]

$$\kappa(\mathbf{r}_k(n), \mathbf{r}_k(i)) = \langle \kappa(\cdot, \mathbf{r}_k(n)), \kappa(\cdot, \mathbf{r}_k(i)) \rangle_{\mathcal{H}}, \tag{3}$$

where $\mathcal{H}$ is the induced RKHS and $\langle \cdot,\cdot \rangle_{\mathcal{H}}$ denotes the corresponding inner product. In (3), $\kappa(\cdot, \mathbf{r}_k(i))$ is a representer evaluation at $\mathbf{r}_k(i)$ [27], [28].

A. Centralized Graph Kernel LMS

In the graph setting, at every time-instant n, K new data samples are available. Then, at time-instant n, using the representer theorem [20], the estimate of $y_l(n)$ (i.e., $\hat{y}_l(n)$), given data pairs $\{\mathbf{r}_k(i), y_k(i)\}_{i=1,k=1}^{n-1,K} \bigcup \{\mathbf{r}_k(n)\}_{k=1}^{K}$, can be expressed as

$$\hat{y}_l(n) = f(\mathbf{r}_l(n)) = \sum_{i=1}^{n} \sum_{k=1}^{K} \alpha_{ik}\ \kappa(\mathbf{r}_l(n), \mathbf{r}_k(i)). \tag{4}$$

The model in (4) grows with both time, n, and network size, K. This is a well-known issue with single-node kernel methods [19], [26]–[28], [34]–[36], where several solutions have been proposed that learn a sparse, or fixed-size dictionary. Of these, the coherence-check (CC) methods use a coherence metric [19], [28] between a candidate regressor and the current dictionary to decide whether to include the candidate in the dictionary. Using coherence-check criterion, $\hat{y}_l(n)$ can be written as

$$\hat{y}_l(n) = f(\mathbf{r}_l(n)) = \sum_{i \in \mathcal{M}(n)} \sum_{k \in \mathcal{K}(i)} \alpha_{ik}\ \kappa\big(\mathbf{r}_l(n), \mathbf{r}_k(i)\big), \tag{5}$$

where $\mathcal{M}(n)$ is a set of time instants in which at least one input regressor is added to the dictionary until time-index n, with $|\mathcal{M}(n)| \leq n$, and $\mathcal{K}(i)$ is a set of node indices of regressors that passed the coherence-check at time-index i, with $|\mathcal{K}(i)| \leq K$. Under the coherence-check criterion, at time-index n, the dictionary $\mathcal{D}(n)$ contains $|\mathcal{D}(n)| = \sum_{i \in \mathcal{M}(n)} |\mathcal{K}(i)|$ regressors.

Remark 1. Given reasonable conditions on the coherence-metric threshold, the maximum number of regressors in the dictionary is finite, i.e., $|\mathcal{D}|$ stops increasing after a certain time [28].

The coefficients of the function expansion in (5) are obtained through the following minimization problem

$$\begin{aligned} \min_{\alpha_{ik}} \sum_{l=1}^{K} \mathbb{E}\Big[\Big(y_l(n) - \sum_{i \in \mathcal{M}(n)} \sum_{k \in \mathcal{K}(i)} \alpha_{ik}\ \kappa\big(\mathbf{r}_l(n), \mathbf{r}_k(i)\big)\Big)^2\Big] \\ = \min_{\boldsymbol{\alpha} \in \mathbb{R}^{|\mathcal{D}(n)|}} \mathbb{E}\left[\|\mathbf{y}(n) - \mathbf{K}(n)\boldsymbol{\alpha}\|_2^2\right], \end{aligned} \tag{6}$$

where $\boldsymbol{\alpha} \triangleq [\boldsymbol{\alpha}_1^{\mathrm{T}}\, \boldsymbol{\alpha}_2^{\mathrm{T}}\, \ldots\, \boldsymbol{\alpha}_{|\mathcal{M}(n)|}^{\mathrm{T}}]^{\mathrm{T}}$, with $\boldsymbol{\alpha}_i^{\mathrm{T}} = [\alpha_{i1}\, \alpha_{i2}\, \ldots\, \alpha_{i|\mathcal{K}(i)|}] \in \mathbb{R}^{|\mathcal{K}(i)|}$ and $\mathbf{K}(n) = [\mathbf{K}_1(n)\, \mathbf{K}_2(n)\, \ldots\, \mathbf{K}_{|\mathcal{M}(n)|}] \in \mathbb{R}^{K \times |\mathcal{D}(n)|}$, with $[\mathbf{K}_i(n)]_{lk} = \kappa(\mathbf{r}_l(n), \mathbf{r}_k(i))$, for $l \in \mathcal{N}$ and $k \in \mathcal{K}(i)$.

Considering the growing nature of the dictionary, access to second-order statistics is impractical. Therefore, we use a stochastic-gradient approach and minimize the instantaneous value of (6) recursively. The update equation for the graph KLMS (GKLMS) is given by

$$\boldsymbol{\alpha}(n+1) = \boldsymbol{\alpha}(n) + \mu\, \mathbf{K}^{\mathrm{T}}(n)\, \big(\mathbf{y}(n) - \mathbf{K}(n)\boldsymbol{\alpha}(n)\big), \tag{7}$$

where μ is a positive adaptation step size.

Remark 2. If coherence-check is employed in an online fashion, two events must be considered for each candidate regressor: if the regressor does not satisfy the coherence-check criteria, the dictionary remains the same. Otherwise, $\mathbf{K}(n)$ gets one new column and a zero-valued entry must be appended to $\boldsymbol{\alpha}(n)$ [28]. At every time instant i, $|\mathcal{K}(i)|$ regressors are added to $\mathcal{D}$. Hence, $|\mathcal{K}(i)|$ zeros must be appended to $\boldsymbol{\alpha}(n)$.

B. Centralized GKLMS using RFF

An alternative to sparsification methods is provided by random Fourier features (RFF) [29]. RFF are used to approximate the evaluation of a shift-invariant kernel $\kappa(\mathbf{r}(n), \mathbf{r}(i)) = \kappa(\mathbf{r}(n) - \mathbf{r}(i))$ as an inner-product in the D-dimensional RFF space. This turns the problem into a finite-dimension linear problem while removing the need to evaluate kernel functions [29]. Let $\mathbf{z}_k(n)$ be the mapping of $\mathbf{r}_k(n)$ into the RFF space $\mathbb{R}^D$. Then, the kernel evaluation can be approximated as $\kappa(\mathbf{r}_l(n), \mathbf{r}_k(i)) \approx \mathbf{z}_k^{\mathrm{T}}(i)\mathbf{z}_l(n)$, and the estimate $\hat{y}_l(n)$ in (4) can be approximated by

$$\hat{y}_l(n) \approx \Big(\sum_{i=1}^{n}\sum_{k=1}^{K} \alpha_{ik}\ \mathbf{z}_k(i)\Big)^{\mathrm{T}} \mathbf{z}_l(n) = \mathbf{h}^{\mathrm{T}}\mathbf{z}_l(n), \qquad (8)$$

where $\mathbf{h} \in \mathbb{R}^D$ is the representation of the function $f(\cdot)$ in the RFF space. Let the matrix $\mathbf{Z}(n) = [\mathbf{z}_1(n)\, \mathbf{z}_2(n) \ldots \mathbf{z}_K(n)]$ describe the RFF mapping of all input vectors at time n. Now, the optimization problem becomes

$$\mathbf{h}^*(n) = \arg\min_{\mathbf{h}\in\mathbb{R}^D} \mathbb{E}\left[\|\mathbf{y}(n) - \mathbf{Z}^{\mathrm{T}}(n)\mathbf{h}\|_2^2\right]. \qquad (9)$$

Similar to (7), approximating the solution through stochastic gradient descent iterations yields the update rule

$$\mathbf{h}(n+1) = \mathbf{h}(n) + \mu\mathbf{Z}(n)\mathbf{e}(n), \qquad (10)$$

where $\mathbf{e}(n) \triangleq \mathbf{y}(n) - \mathbf{Z}^{\mathrm{T}}(n)\mathbf{h}(n)$.

The estimates $\boldsymbol{\alpha}$ in (7) and $\mathbf{h}$ in (10) require knowledge of the input of the entire graph, which can be impractical in applications without a centralized processing unit. Therefore, we consider now a distributed implementation of the GKLMS, named graph diffusion KLMS (GDKLMS).

C. GDKLMS using RFF

The global optimization problem (9) can be rewritten as the following separable problem

$$(\boldsymbol{\psi}_1^*(n), \ldots, \boldsymbol{\psi}_K^*(n)) = \arg\min_{\boldsymbol{\psi}_1,\ldots,\boldsymbol{\psi}_K \in\mathbb{R}^D} \sum_{k=1}^{K} \mathbb{E}[(y_k(n) - \mathbf{z}_k^{\mathrm{T}}(n)\boldsymbol{\psi}_k)^2], \qquad (11)$$

where $\boldsymbol{\psi}_k$ is the local estimate of $\mathbf{h}$ at node k. Problem (11) is solved in a distributed fashion by minimizing $\mathbb{E}\left[(y_k(n) - \mathbf{z}_k^{\mathrm{T}}(n)\boldsymbol{\psi}_k)^2\right]$ at each node. Similar to the centralized case, denoting $e_k(n) = y_k(n) - \mathbf{z}_k^{\mathrm{T}}(n)\boldsymbol{\psi}_k(n)$, the update rule for $\boldsymbol{\psi}_k$ is given by

$$\boldsymbol{\psi}_k(n+1) = \boldsymbol{\psi}_k(n) + \mu\ e_k(n)\mathbf{z}_k(n). \qquad (12)$$

We now adopt the adapt-then-combine (ATC) strategy to improve individual estimates via graph diffusion [11], [12], [19], [30], [32]. The parameter update of $\mathbf{h}_k(n)$ at node k is obtained by combining the local estimates from its neighborhood. The ATC update rule for the GDKLMS using RFF is given by

$$\boldsymbol{\psi}_k(n+1) = \mathbf{h}_k(n) + \mu\ e_k(n)\mathbf{z}_k(n), \qquad (13a)$$

$$\mathbf{h}_k(n+1) = \sum_{l\in\mathcal{N}_k} a_{lk}\ \boldsymbol{\psi}_l(n+1), \qquad (13b)$$

where combination coefficients a_{lk} are non-negative and satisfy the condition $\sum_{l\in\mathcal{N}_k} a_{lk} = 1$ [32].

We note that if coherence-check is implemented for individual nodes, it can lead to unequal dictionaries across the graph, making it challenging to implement the algorithm in a distributed fashion [28]. As an alternative, we consider the construction of a pre-trained centralized dictionary [19]. The ATC approach using coherence-check proposed in [19] can be generalized for graph kernel filters, such that each node k updates its coefficient vector $\boldsymbol{\alpha}_k$ by combining the local estimates from its neighborhood. The dictionary can be obtained in a centralized way and broadcasted to the entire network or by a single dedicated node that shares its dictionary with all nodes. Moreover, the pre-training of the dictionary depends on available training data. Therefore, we note that RFF offer more flexibility for distributed implementations than the coherence-check approach, as the dimension of the RFF space can be set equally for all nodes.

IV. Mean Convergence Analysis

In this section, we study the mean convergence of the GDKLMS using RFF. For this, at network-level, we define the filter coefficient vector in RFF space $\mathbf{h}_g = \mathbf{1}_K \otimes \mathbf{h}$, the estimated filter coefficient vector in RFF space $\mathbf{h}_g(n) = [\mathbf{h}_1^{\mathrm{T}}(n)\, \mathbf{h}_2^{\mathrm{T}}(n) \ldots \mathbf{h}_K^{\mathrm{T}}(n)]^{\mathrm{T}}$, and the input data matrix $\boldsymbol{\mathcal{Z}}(n) = \text{blockdiag}\{\mathbf{z}_1(n), \mathbf{z}_2(n), \ldots, \mathbf{z}_K(n)\}$, where blockdiag$\{\cdot\}$ denotes the block-diagonal-stacking operator. The symbol $\mathbf{1}_K$ is a column vector of size $K \times 1$ with every element taking the value one and $\otimes$ denotes the right Kronecker product operator. Combination coefficients are gathered into a stochastic matrix $\mathbf{A}$ such that $[\mathbf{A}]_{k,l} = a_{kl}$. From these definitions, the network-level data model is given by $\mathbf{y}(n) = \boldsymbol{\mathcal{Z}}^{\mathrm{T}}(n)\mathbf{h}_g + \boldsymbol{v}(n)$, where $\boldsymbol{v}(n) = [v_1(n)\, v_2(n) \ldots v_K(n)]^{\mathrm{T}}$. Using these definitions, the network-level update recursion of the GD-KLMS using RFF can be stated as

$$\mathbf{h}_g(n+1) = \boldsymbol{\mathcal{A}}\left(\mathbf{h}_g(n) + \mu\boldsymbol{\mathcal{Z}}(n)\mathbf{e}(n)\right), \qquad (14)$$

where $\boldsymbol{\mathcal{A}} = \mathbf{A}^{\mathrm{T}} \otimes \mathbf{I}_D$. Denoting the global weight deviation vector of the proposed GDKLMS using RFF at time index n as $\tilde{\mathbf{h}}_g(n) = \mathbf{h}_g - \mathbf{h}_g(n)$, and considering that $\boldsymbol{\mathcal{A}}\mathbf{h}_g = \mathbf{h}_g$ (since the matrix $\mathbf{A}$ is left stochastic), from (14), the recursion for $\tilde{\mathbf{h}}_g(n)$ can then be obtained as

$$\tilde{\mathbf{h}}_g(n+1) = \boldsymbol{\mathcal{B}}(n)\tilde{\mathbf{h}}_g(n) - \mu\ \boldsymbol{\mathcal{A}}\boldsymbol{\mathcal{Z}}(n)\boldsymbol{v}(n), \qquad (15)$$

where $\boldsymbol{\mathcal{B}}(n) = \boldsymbol{\mathcal{A}}\big(\mathbf{I}_{DK} - \mu\boldsymbol{\mathcal{Z}}(n)\boldsymbol{\mathcal{Z}}^{\mathrm{T}}(n)\big)$.

Taking the statistical expectation on both sides of (15), assuming statistical independence between $\mathbf{h}_k(n)$ and $\mathbf{z}_k(n)$, $\forall k \in \mathcal{N}$ [31], and considering that the observation noise $v_k(n)$

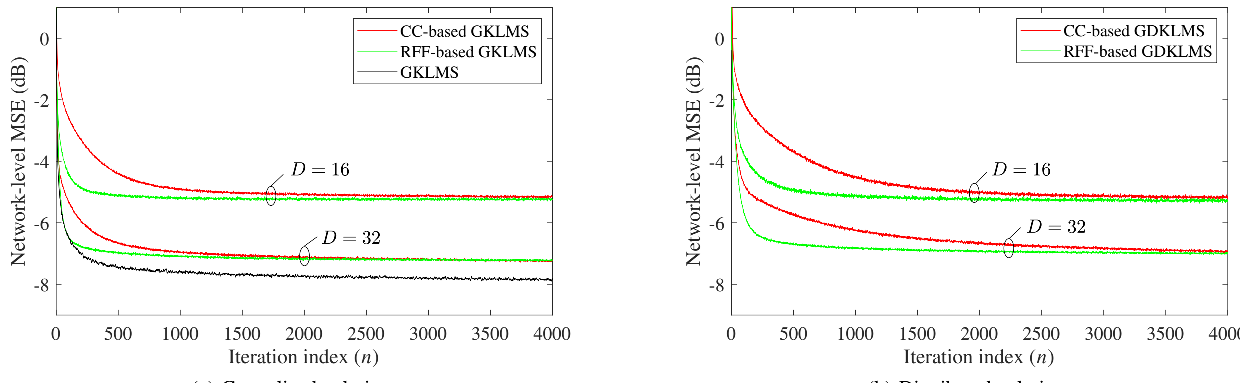


Fig. 1. Learning curves (network-level MSE vs iteration index) for the proposed algorithms.

is a zero mean i.i.d. random sequence, which is taken to be independent of any other signal, we obtain

$$\mathbb{E}[\tilde{\mathbf{h}}_g(n+1)] = \overline{\boldsymbol{\mathcal{B}}}\ \mathbb{E}[\tilde{\mathbf{h}}_g(n)], \tag{16}$$

where $\overline{\boldsymbol{\mathcal{B}}} = \mathbb{E}[\boldsymbol{\mathcal{B}}(n)] = \boldsymbol{\mathcal{A}}(\mathbf{I}_{DK} - \mu\mathbf{R}_z)$ with $\mathbf{R}_z = \mathbb{E}[\boldsymbol{\mathcal{Z}}(n)\boldsymbol{\mathcal{Z}}^{\mathrm{T}}(n)] = \text{blockdiag}(\mathbf{R}_{z,1}, \mathbf{R}_{z,2}, \ldots, \mathbf{R}_{z,K})$, with $\mathbf{R}_{z,k} = \mathbb{E}[\mathbf{z}_k(n)\mathbf{z}_k^{\mathrm{T}}(n)]$. Note that the vector $\mathbf{z}_k(n)$ is the representation of $\mathbf{r}_k(n)$ in the RFF space. So the input vectors $\mathbf{z}_k(n)$, for $k \in \mathcal{N}$, may not satisfy both zero-mean and Gaussian distribution conditions [24]. However, if the basis of the RFF space is generated in a way such that the basis vectors $\mathbf{v}_i \neq \mathbf{v}_j$ for any $i \neq j$, the autocorrelation matrix $\mathbf{R}_{z,k}$, for $k \in \mathcal{N}$ will be strictly positive definite [24]. Therefore, from (16), it is easily seen that $\lim_{n\to\infty} \mathbb{E}\big[\tilde{\mathbf{h}}_g(n)\big]$ attains a finite value if and only if $\|\overline{\boldsymbol{\mathcal{B}}}\| < 1$, where $\|\cdot\|$ denotes any matrix norm. We derive a convergence condition in terms of μ, by constraining the block maximum norm of the matrix $\overline{\boldsymbol{\mathcal{B}}}$ (i.e., $\|\overline{\boldsymbol{\mathcal{B}}}\|_{b,\infty}$). Using the properties of block maximum norm [32], we can write

$$\|\overline{\boldsymbol{\mathcal{B}}}\|_{b,\infty} \leq \|\boldsymbol{\mathcal{A}}\|_{b,\infty}\|\mathbf{I}_{DK} - \mu\mathbf{R}_z\|_{b,\infty}. \tag{17}$$

Using [32, Lemma D. 3(a), D. 5], a sufficient condition for $\mathbb{E}[\tilde{\mathbf{h}}_g(n)]$ to converge asymptotically in mean is given by $\rho(\mathbf{I}_{DK} - \mu\mathbf{R}_z) < 1$, or, equivalently, $|1 - \mu\lambda_j(\mathbf{R}_z))| < 1$ for $j \in \{1, 2, \ldots, DK\}$, where $\rho(\cdot)$ denotes the spectral radius of the argument matrix and $\lambda_j(\mathbf{R}_z)$ denotes the jth eigenvalue of $\mathbf{R}_z$. After solving this, we obtain the following condition on μ:

$$0 < \mu < \frac{2}{\max\limits_{\forall k \in \mathcal{N}}\big\{\max\limits_{\forall i}\{\lambda_i(\ \mathbf{R}_{z,k})\}\big\}}. \tag{18}$$

V. Numerical Results

We validate the performance of the proposed algorithms on a connected Erdös-Renyi graph consisting of 20 nodes with edge probability equal to 0.2. The shift matrix $\mathbf{S}$ is constructed as follows: first, the existing edges, according to the previously constructed graph, receive a weight value drawn from the uniform distribution in the interval $(0, 1]$; each entry s_{kl} receives the value of the corresponding edge weight or zero if the edge does not exist; the eigenvalues $\{\lambda_k\}_{k=1}^K$ of $\mathbf{S}$ are normalized by the largest eigenvalue such that $|\lambda_k| \leq 1$. Input signal $\mathbf{x}(n)$ and observation noise $\boldsymbol{v}(n)$ are drawn from zero-mean normal distributions with covariance matrices $\mathbf{R}_{\mathbf{x}} = \text{diag}\{\sigma^2_{x,k}\}$ and $\mathbf{R}_{\boldsymbol{v}} = \text{diag}\{\sigma^2_{v,k}\}$, respectively, where $\sigma^2_{x,k}$ are drawn from the uniform distribution in $[1, 1.5]$ and $\sigma^2_{v,k}$ from $[0.1, 0.15]$. For distributed implementations, the combination coefficients a_{kl} are computed according to the Metropolis rule [32]. We used a Gaussian kernel with $\sigma^2 = 1$. For a filter of length $L = 4$, we aim at estimating the nonlinear function given by

$$\begin{aligned} f(\mathbf{r}_k(n)) = \sqrt{r_{k,1}(n)^2 + \sin^2(r_{k,4}(n)\pi)} \\ + (0.8 - 0.5\exp(-r_{k,2}(n)^2)r_{k,3}(n) \end{aligned} \tag{19}$$

The network-level MSE given by $\text{MSE}(n) = \frac{1}{K}\sum_{k=1}^N e_k^2(n)$ was considered as the performance metric, and results are displayed by plotting the MSE versus iteration index n, averaging over 1000 independent experiments. In order to compare coherence-check-based approaches with RFF-based approaches, the adaptation step size μ was adjusted so that the learning curves achieve similar steady network-level MSE. A centralized training dataset was used for the coherence-check simulations to pre-train the dictionary and broadcast it to all nodes before the learning iterations. The number of training samples used in the pre-training is not considered in the results. Moreover, we note that the linear approaches, namely the graph LMS and the graph diffusion LMS [12], could not model the target function reasonably.

Fig. 1a shows the learning curves for centralized solutions. Specifically, we compare the GKLMS without dictionary sparsification with $\mu = 0.1$ to the solutions using RFF and coherence-check. The value $D \in \{16, 32\}$ represents both the dimension of the RFF space and the size of the pre-trained dictionary for the coherence-check approach. Results show that the GKLMS without dictionary sparsification, coherence-check, and RFF based algorithms can effectively represent the target function. Fig. 1a also shows that, for the same D and similar values of steady-state network-level MSE, the RFF based algorithm converges faster than the coherence-check-based one. Moreover, comparing the plots for $D = 32$ to the GKLMS plot shows that both the coherence-check and RFF based algorithms can approximate the graph KLMS without sparsification as D increases.

Fig. 1b shows the results for the distributed GDKLMS using coherence-check and RFF. Similar to the centralized case, the plots show that the coherence-check and RFF-based approaches can effectively represent the target function, achieving network-level MSE of approximately -10 dB for $D = 16$ and -14 dB for $D = 32$. Again, the RFF-based solution exhibits faster convergence for both values of D when the steady-state network-level MSE is matched.

VI. Conclusions

This paper introduced nonlinear graph filters that operate in the reproducing kernel Hilbert space. To this end, a centralized graph kernel LMS (GKLMS) algorithm was derived. To overcome the growing dimension problem encountered in the centralized GKLMS algorithm, coherence-check based dictionary-sparsification and random Fourier feature (RFF) based approaches were proposed. Furthermore, diffusion-based distributed implementations of coherence-check and RFF-based graph KLMS algorithms were developed to update filter parameters through local communications and in-network processing. Mean convergence conditions on the adaptation step size were established for the proposed GDKLMS using RFF. Numerical simulations were conducted to demonstrate the performance of the proposed algorithms. Although the coherence-check and RFF-based approaches effectively estimate the nonlinear graph filter, the RFF-based approach exhibits a faster convergence rate than the coherence-check based approach.

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 Process. Mag.*, vol. 30, pp. 83–98, May 2013.

[2] A. Sandryhaila and J. M. F. Moura, “Discrete signal processing on graphs,” *IEEE Trans. Signal Process.*, vol. 61, pp. 1644–1656, Apr. 2013.

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

[4] E. Ceci and S. Barbarossa, “Graph signal processing in the presence of topology uncertainties,” in *IEEE Trans. Signal Process.*, 2020.

[5] I. Jablonski, “Graph signal processing in applications to sensor networks, smart grids, and smart cities,” *IEEE Sensors J.*, vol. 17, pp. 7659–7666, Dec. 2017.

[6] A. Gavili and X. Zhang, “On the shift operator, graph frequency, and optimal filtering in graph signal processing,” *IEEE Trans. Signal Process.*, vol. 65, pp. 6303–6318, Dec. 2017.

[7] J. Mei and J. M. F. Moura, “Signal processing on graphs: Causal modeling of unstructured data,” *IEEE Trans. Signal Process.*, vol. 65, pp. 2077–2092, Apr. 2017.

[8] A. Loukas, A. Simonetto, and G. Leus, “Distributed autoregressive moving average graph filters,” *IEEE Signal Process. Lett.*, vol. 22, pp. 1931–1935, Nov. 2015.

[9] E. Isufi, A. Loukas, N. Perraudin and G. Leus, “Forecasting time series with VARMA recursions on graphs,” in *IEEE Trans. Signal Process.*, vol. 67, no. 18, pp. 4870-4885, Sep. 2019.

[10] A. Sandryhaila and J. M. F. Moura, “Discrete signal processing on graphs: Graph filters,” in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2013, pp. 6163–6166.

[11] R. Nassif, C. Richard, J. Chen, and A. H. Sayed, “A graph diffusion LMS strategy for adaptive graph signal processing,” in *Conf. Rec. Asilomar Conf. Signals Syst. Comput.*, pp. 1973–1976, Oct. 2017.

[12] R. Nassif, C. Richard, J. Chen, and A. H. Sayed, “Distributed diffusion adaptation over graph signals,” in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2018, pp. 4129–4133.

[13] P. Di Lorenzo, P. Banelli, S. Barbarossa, and S. Sardellitti, “Distributed adaptive learning of graph signals,” *IEEE Trans. Signal Process.*, vol. 65, pp. 4193–4208, Aug. 2017.

[14] F. Hua, R. Nassif, C. Richard, H. Wang and A. H. Sayed, “Online distributed learning over graphs with multitask graph-filter models," in *IEEE Trans. Signal Inf. Process. Netw.*, vol. 6, pp. 63–77, Jan. 2020.

[15] M. J. M. Spelta and W. A. Martins, “Normalized LMS algorithm and data-selective strategies for adaptive graph signal estimation," in *Signal Process.*, vol. 167, 107326, Feb. 2020.

[16] A. Venkitaraman, S. Chatterjee and P. Händel, "Predicting graph signals using kernel regression where the input signal is agnostic to a graph," in *IEEE Trans. Signal Inf. Process. Netw.*, vol. 5, no. 4, pp. 698–710, Dec. 2019.

[17] J. F. Walker, N. Jenkins and N. Jenkins *Wind Energy Technology*. Wiley, June 1997.

[18] W. Liu, J. C. Príncipe, and S. Haykin, *Kernel Adaptive Filtering*. Wiley, Feb. 2010.

[19] W. Gao, J. Chen, C. Richard, and J. Huang, “Diffusion adaptation over networks with kernel least-mean-square,” in *Proc. IEEE Int. Workshop Comput. Adv. Multisens. Adapt. Process.*, 2015, pp. 217–220.

[20] A. J. Smola and R. Kondor, “Kernels and regularization on graphs,” in *Learning Theory and Kernel Machines. Lecture Notes in Computer Science*, vol. 2777, pp. 144–158, Springer, 2003.

[21] S. Theodoridis, K. Slavakis, and I. Yamada, “Adaptive learning in a world of projections,” *IEEE Signal Process. Mag.*, vol. 28, pp. 97–123, Jan. 2011.

[22] D. Romero, M. Ma, and G. B. Giannakis, “Kernel-based reconstruction of graph signals,” *IEEE Trans. Signal Process.*, vol. 65, pp. 764–778, Feb. 2017.

[23] Y. Shen, G. Leus, and G. B. Giannakis, “Online graph-adaptive learning with scalability and privacy,” *IEEE Trans. Signal Process.*, vol. 67, pp. 2471–2483, May 2019.

[24] P. Bouboulis, S. Chouvardas, and S. Theodoridis, “Online distributed learning over networks in RKH spaces using random Fourier features,” *IEEE Trans. Signal Process.*, vol. 66, no. 7, pp. 1920–1932, 2018.

[25] B.-S. Shin, M. Yukawa, R. L. G. Cavalcante, and A. Dekorsy, “Distributed adaptive learning with multiple kernels in diffusion networks,” *IEEE Trans. Signal Process.*, vol. 66, pp. 5505–5519, Nov. 2018.

[26] S. Wang, L. Dang, B. Chen, S. Duan, L. Wang, and C. K. Tse, “Random Fourier filters under maximum correntropy criterion,” *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, pp. 3390–3403, Oct. 2018.

[27] K. Chen, S. Werner, A. Kuh, and Y.-F. Huang, “Nonlinear adaptive filtering with kernel set-membership approach,” *IEEE Trans. Signal Process.*, pp. 1–1, 2020.

[28] C. Richard, J. Bermudez, and P. Honeine, “Online prediction of time series data with kernels,” *IEEE Trans. Signal Process.*, vol. 57, pp. 1058–1067, Mar. 2009.

[29] A. Rahimi and B. Recht, “Random features for large-scale kernel machines,” in *Proc. Adv. Neural Inf. Process. Syst.*, pp. 1177–1184, 2007.

[30] A. H. Sayed, S. Tu, J. Chen, X. Zhao, and Z. J. Towfic, “Diffusion strategies for adaptation and learning over networks: An examination of distributed strategies and network behavior,” *IEEE Signal Process. Mag.*, vol. 30, pp. 155–171, May 2013.

[31] A. H. Sayed, “Adaptive networks,” *Proc. IEEE*, vol. 102, pp. 460–497, Apr. 2014.

[32] A. H. Sayed, “Adaptation, learning, and optimization over networks,” in *Foundations and Trends® in Mach. Learn.*, vol. 7, pp. 311–801, 2014.

[33] M. O. Franz and B. Schölkopf, “A unifying view of Wiener and Volterra theory and polynomial kernel regression,” *Neural Comput.*, vol. 18, pp. 3097–3118, Dec. 2006.

[34] A. Singh, N. Ahuja, and P. Moulin, “Online learning with kernels: Overcoming the growing sum problem,” in *Proc. IEEE Int. Workshop Mach. Learn. Signal Process.*, 2012, pp. 1–6.

[35] P. Bouboulis, S. Pougkakiotis, and S. Theodoridis, “Efficient KLMS and KRLS algorithms: A random Fourier feature perspective,” in *Proc. IEEE Stat. Signal Process. Workshop*, 2016, pp. 1–5.

[36] S. Chouvardas and M. Draief, “A diffusion kernel LMS algorithm for nonlinear adaptive networks,” in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2016, pp. 4164–4168.

Adaptive Graph Filters in Reproducing Kernel Hilbert Spaces: Design and Performance Analysis

Vitor R. M. Elias, Vinay Chakravarthi Gogineni, *Member, IEEE,* Wallace A. Martins, *Senior Member, IEEE* and Stefan Werner, *Senior Member, IEEE*

***Abstract*—This paper develops adaptive graph filters that operate in reproducing kernel Hilbert spaces. We consider both centralized and fully distributed implementations. We first define nonlinear graph filters that operate on graph-shifted versions of the input signal. We then propose a centralized graph kernel least mean squares (GKLMS) algorithm to identify nonlinear graph filters' model parameters. To reduce the dictionary size of the centralized GKLMS, we apply the principles of coherence check and random Fourier features (RFF). The resulting algorithms have performance close to that of the GKLMS algorithm. Additionally, we leverage the graph structure to derive the distributed graph diffusion KLMS (GDKLMS) algorithms. We show that, unlike the coherence check-based approach, the GDKLMS based on RFF avoids the use of a pre-trained dictionary through its data-independent fixed structure. We conduct a detailed performance study of the proposed RFF-based GDKLMS, and the conditions for its convergence both in mean and mean-squared senses are derived. Extensive numerical simulations show that GKLMS and GDKLMS can successfully identify nonlinear graph filters and adapt to model changes. Furthermore, RFF-based strategies show faster convergence for model identification and exhibit better tracking performance in model-changing scenarios.**

***Index Terms*—Adaptive signal processing, distributed learning, kernel graph filters, kernel LMS, random Fourier features.**

I. Introduction

Graph signal processing (GSP) has recently received increased attention due to its wide applicability to model, process, and analyze signals and large data sets, ranging from daily-life social networks to sensor networks for industrial and military applications [1]–[5]. For instance, in the context of a wireless sensor network, graph nodes and edges represent sensors and communication links, respectively, while the so-called graph signal is the measurement snapshot across sensors [6]. Similar to traditional digital signal processing (DSP) techniques, the basic building block in GSP is the graph-shift operation, which captures node interconnections [7]. In the particular case of linear networks, the graph-shifted signal on a given node is a linear combination of adjacent node signals, where the weights relate to the edge values. This resemblance to DSP has sparked the development of a vast amount of GSP counterparts of methods related to spectral analysis [8]–[14] and traditional time-series analysis [15], [16].

One of the key research areas in GSP is modeling unknown relations between input and output graph signals through a filter [11]–[18]. The application of linear shift-invariant filter models is widely employed in the literature, e.g., to design graph spectral filters [11], [12] and model dynamic graph signals [15], [16]. Several works deal with adaptive learning of graph filters, see, e.g., [19]–[23]. These methods were later extended to multitask graphs [24], [25]. The previous works adopt the ideas of linear adaptive networks [26], [27] to estimate the graph filter through in-network processing. However, linear models cannot accurately represent many real-world systems that exhibit more sophisticated input-output relations. Prominent examples include the relations between air pressure and temperature [28], and wind speed and generated power in wind turbines [29].

In conventional DSP, several approaches to nonlinear system modeling exist in the literature [30]–[38]. In particular, methods based on reproducing kernel Hilbert spaces (RKHS) have gained popularity due to their efficacy and mathematical simplicity [36]–[53]. There is extensive literature on function estimation in RKHS for both single- and multi-node networks, see, e.g., [39]–[57]. Most works on adaptive networks treat each nodal signal as time series to estimate a common filter vector. In contrast, in GSP, the graph filter operates on an instantaneous topology-dependent snapshot of the network state by exploiting graph shifts. Although some of the prior works account for the input signals' network-related characteristics, such as smoothness across the graph, existing RKHS-based approaches do not consider graph-shifted signals. The shift operator and delayed versions of graph signals have been explored for linear adaptive graph filters [22], [23].

This paper introduces nonlinear graph filters and presents two adaptive methods for function estimation over graphs, namely the centralized graph kernel least mean squares (GKLMS) and the graph diffusion kernel least mean squares (GDKLMS). Preliminary results on this topic have been presented in [58]. The proposed nonlinear graph filters generalize conventional linear graph filters and consist of a nonlinearity applied to a combination of graph-shifted versions of the input signal. For the estimation methods, we consider two approaches for model reduction, namely coherence check (CC) [40], [52] that sparsifies the original dictionary of the

This work was partly supported by: Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) –- Grant number: 88887.310189/2018-00, CNPq, ERC project AGNOSTIC (Actively Enhanced Cognition based Framework for Design of Complex Systems – Grant number: 742648), FAPERJ, and the Research Council of Norway.

Vitor R. M. Elias and Stefan Werner are with the Norwegian University of Science and Technology, Norway (e-mail: vitor.elias@ntnu.no and stefan.werner@ntnu.no).

Vinay Chakravarthi Gogineni is with the SimulaMet, Simula Research Laboratory, Norway (e-mail: vcgogineni@simula.no).

Wallace A. Martins is with the University of Luxembourg, Luxembourg (e-mail: wallace.alvesmartins@uni.lu).

Vitor R. M. Elias and Wallace A. Martins are also with the Federal University of Rio de Janeiro, Brazil.

GKLMS, and random Fourier features (RFF) [59] that approximate kernel evaluations with inner products in a fixed-dimensional space. One of the main features of the CC-based implementation is the automatic tuning of the model order by selecting regressors based on a coherence measure [52]. On the other hand, RFF-based implementations use a data-independent mapping into a space where kernel evaluations can be approximated as inner-products, making them resilient to model changes. Building upon ideas of network diffusion [26], [27], the proposed RFF-based graph diffusion KLMS (GDKLMS) avoids the centralized processing and updates local estimates at each node through collaboration with neighbors. One of the main features of the RFF-based GDKLMS is its data-independent mapping that avoids using a pre-trained dictionary. This makes the GDKLMS more robust to changes in the underlying system since there is no need to retrain dictionaries associated with distributed CC-based solutions [52]. We analyze the performance of the GDKLMS and establish the convergence conditions in both mean and mean-squared senses.

This paper is organized as follows. Section II presents the necessary concepts and notations of GSP, including the conventional models of linear graph filters, and formulates the problem of modeling nonlinear graph filters. The proposed GKLMS and GDKLMS algorithms are presented in Section III. We first derive the GKLMS as a centralized solution for the modeling problem and present the implementations based on CC and RFF. Thereafter, the RFF-based GDKLMS is derived. In Section IV, we present the convergence analysis of the RFF-based GDKLMS, along with the conditions for convergence in the mean and mean-squared senses. In this section, we also study the steady-state mean-squared error. In Section VI, numerical experiments are conducted to demonstrate the performance of the proposed solutions for identifying and tracking the nonlinear graph filters. For this, we use both synthetic and real-life networks. Synthetic examples employ generic nonlinear functions, whereas real-life examples treat the modeling of relations between temperature and humidity data from sensor networks. Finally, in Section VII, we present the concluding remarks of this work.

II. Problem Formulation

Consider an undirected graph $\mathcal{G} = \{\mathcal{N}, \mathcal{E}\}$, where $\mathcal{N} = \{1, 2, \ldots, K\}$ is the set of nodes and $\mathcal{E}$ is the set of edges such that $(k,l) \in \mathcal{E}$ if nodes k and l are connected. The graph is associated with a graph-shift operator, $\mathbf{S} \in \mathbb{R}^{K\times K}$, whose entries $[\mathbf{S}]_{k,l} = s_{kl}$ take non-zero values only if $(k,l) \in \mathcal{E}$ [1], [2]. The graph adjacency matrix [2] and the graph Laplacian matrix [1] are the most common choices for $\mathbf{S}$. At time instant n, the graph signal is defined by a vector $\mathbf{x}_n = [x_{1,n}\, x_{2,n}\, \ldots\, x_{K,n}]^{\mathrm{T}}$, with $x_{k,n}$ being the signal value at node k. The graph-shift operation $\mathbf{S}\mathbf{x}_n$ is performed locally at each node k by linearly combining the samples from neighboring nodes, namely $\sum_{l\in\mathcal{N}_k} s_{kl}x_{l,n}$, where $\mathcal{N}_k$ denotes the neighborhood of node k including k itself. In this work, we assume the graph topology and the shift matrix are known. For cases where $\mathbf{S}$ is not known, one can employ different techniques for learning the graph structure available in the GSP literature [11], [60]–[64].

A linear shift-invariant (LSI) graph filter of size $L \times 1$ combines shifted graph signals and is defined by

$$\mathbf{H} = \sum_{i=0}^{L-1} h_i \mathbf{S}^i, \tag{1}$$

where $[h_0\, h_1\, \ldots\, h_{L-1}]^{\mathrm{T}}$ is the linear graph filter coefficient vector [12], [22]. When streaming data is available, a two-dimensional graph-time filter [13] can be employed. The filter processes the signal $\mathbf{x}_n$ and yields the graph filtered vector $\mathbf{y}_n = [y_{1,n}\, y_{2,n}\, \ldots\, y_{K,n}]^{\mathrm{T}}$ as

$$\mathbf{y}_n = \sum_{i=0}^{L-1}\sum_{j=0}^{M-1} h_{i,j}\,\mathbf{S}^i\mathbf{x}_{n-j} + \boldsymbol{\upsilon}_n, \tag{2}$$

where $M-1$ is the filter memory in temporal domain, and $\boldsymbol{\upsilon}_n = [\upsilon_{1,n}\, \upsilon_{2,n}\, \ldots\, \upsilon_{K,n}]^{\mathrm{T}}$ is a zero-mean wide-sense stationary (WSS) noise with covariance matrix $\mathbf{R}_\upsilon = \mathrm{diag}\{\sigma^2_{\upsilon,1}, \sigma^2_{\upsilon,2}, \ldots, \sigma^2_{\upsilon,K}\}$. Also, $\boldsymbol{\upsilon}_n$ and $\boldsymbol{\upsilon}_m$ are i.i.d. for any $n \neq m$. The model (2) uses walks of up to length $L-1$ in the graph. Thus, it requires multihop communication in distributed implementations, which limits its usage in real-time applications.

A simplified model that avoids multihop communication can be constructed by combining time and graph domains into one, as

$$\mathbf{y}_n = \sum_{i=0}^{L-1} h_i \mathbf{S}^i \mathbf{x}_{n-i} + \boldsymbol{\upsilon}_n. \tag{3}$$

A graph diffusion LMS strategy using model (3) is proposed in [23]. In (3), samples $\{x_{k,n}, [\mathbf{S}\mathbf{x}_{n-1}]_k, \ldots, [\mathbf{S}^{L-1}\mathbf{x}_{n-L+1}]_k\}$ are available locally at node k. Thus, only one graph-shift operation is needed at each time instant. A crucial difference between our GSP approach and conventional single- and multi-variate DSP approaches lies in our assumption that the signals' spatio-temporal dynamics depend on the graph structure.

In many real-world applications, these linear models cannot fully capture the input-output relations [37]. For this purpose, we assume a nonlinear relation between input and output, at node k, given by

$$y_{k,n} = f(\mathbf{r}_{k,n}) + \upsilon_{k,n}, \tag{4}$$

where $f : \mathbb{R}^L \to \mathbb{R}$ is a continuous nonlinear function on $\mathbb{R}^L$, $\upsilon_{k,n}$ is the observation noise at node k, and

$$\mathbf{r}_{k,n} = \left[\, x_{k,n}\; [\mathbf{S}\mathbf{x}_{n-1}]_k\; \ldots\; [\mathbf{S}^{L-1}\mathbf{x}_{n-L+1}]_k \,\right]^{\mathrm{T}}. \tag{5}$$

The objective here is to identify $f(\cdot)$ at each node k given a set of data pairs $\{\mathbf{r}_{k,i}, y_{k,i}\}$, $i \in \{1, 2, \ldots, n\}$. In this paper, we characterize nonlinear graph filters using the principles of kernel adaptive filters.

III. Graph Kernel Adaptive Filters

In order to estimate the nonlinear function $f(\cdot)$ in (4), kernel methods first map the input regressors $\{\mathbf{r}_{k,i}\}_{i=1,k=1}^{n,K}$ into a higher dimensional feature space where $f(\cdot)$ takes a linear form [37], [49]. This mapping is denoted by $\kappa(\cdot, \mathbf{r}_{k,i})$, in which $\kappa(\cdot,\cdot) : \mathbb{R}^L \times \mathbb{R}^L \to \mathbb{R}$ is a reproducing kernel, which satisfies [37]

$$\kappa(\mathbf{r}_{k,n}, \mathbf{r}_{k,i}) = \langle \kappa(\cdot, \mathbf{r}_{k,n}), \kappa(\cdot, \mathbf{r}_{k,i}) \rangle_{\mathcal{H}}\,, \tag{6}$$

where $\mathcal{H}$ is the induced RKHS and $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ denotes the corresponding inner product. In (6), $\kappa(\cdot, \mathbf{r}_{k,i})$ is a representer evaluation at $\mathbf{r}_{k,i}$ [51], [52]. The definition of the kernel function is sufficient to evaluate the inner product in (6) without explicitly mapping the data into RKHS.

A. Graph Kernel LMS

In the GSP context, K new data samples are available at each time instant. Then, given a set of regressors $\{\mathbf{r}_{k,i}\}_{i=1,k=1}^{n,K}$, the graph function $f(\cdot)$ can be expressed as a kernel expansion in terms of the mapped data as

$$f(\cdot) = \sum_{i=1}^{n}\sum_{k=1}^{K} \alpha_{ik}\; \kappa(\cdot, \mathbf{r}_{k,i}). \tag{7}$$

The model (7) can approximate any continuous function $f(\cdot)$ [37]. Hence, the corresponding estimate of $y_{l,n}$, at node l, is given by

$$\hat{y}_{l,n} = f(\mathbf{r}_{l,n}) = \sum_{i=1}^{n}\sum_{k=1}^{K} \alpha_{ik}\; \kappa(\mathbf{r}_{l,n}, \mathbf{r}_{k,i}). \tag{8}$$

The coefficients of the expansion in (8) are obtained through the following minimization problem:

$$\min_{\alpha_{ik}\in\mathbb{R}} \sum_{l=1}^{K} \mathrm{E}\left[\Big(y_{l,n} - \sum_{i=1}^{n}\sum_{k=1}^{K} \alpha_{ik}\; \kappa(\mathbf{r}_{l,n}, \mathbf{r}_{k,i})\Big)^2\right]$$
$$= \min_{\boldsymbol{\alpha}\in\mathbb{R}^{nK}} \mathrm{E}\left[\|\mathbf{y}_n - \mathbf{K}_n\,\boldsymbol{\alpha}\|_2^2\right], \tag{9}$$

where $\mathrm{E}[\cdot]$ denotes the expected value of the argument, $\boldsymbol{\alpha}^{\mathrm{T}} = [\boldsymbol{\alpha}_1^{\mathrm{T}}\,\boldsymbol{\alpha}_2^{\mathrm{T}}\,\ldots\,\boldsymbol{\alpha}_n^{\mathrm{T}}]$, with $\boldsymbol{\alpha}_i^{\mathrm{T}} = [\alpha_{i1}\,\alpha_{i2}\,\ldots\,\alpha_{iK}]$, and the matrix

$$\mathbf{K}_n = [\mathbf{K}_{1,n}\,\mathbf{K}_{2,n}\,\ldots\,\mathbf{K}_{n,n}] \in \mathbb{R}^{K\times nK} \tag{10}$$

is a Gram matrix with $[\mathbf{K}_{i,n}]_{l,k} = \kappa(\mathbf{r}_{l,n}, \mathbf{r}_{k,i})$ for $k,l \in \mathcal{N}$.

Considering the growing nature of the dictionary, access to the second-order statistics is impractical. Therefore, we use a stochastic-gradient approach and minimize the instantaneous value of (9) recursively. The update equation for the graph KLMS (GKLMS) is given by

$$\boldsymbol{\alpha}_{n+1} = \boldsymbol{\alpha}_n + \mu\, \mathbf{K}_n^{\mathrm{T}}\,(\mathbf{y}_n - \mathbf{K}_n\boldsymbol{\alpha}_n), \tag{11}$$

where $\mu > 0$ is the step size.

The proposed GKLMS algorithm is summarized in Algorithm 1.

Algorithm 1: GKLMS

```
Input: step size $\mu$
Initialization: $\boldsymbol{\alpha}_0 =$ empty vector;
%Learning
for each time instant $n$ do
    Input: $\mathbf{y}_n, \{\mathbf{r}_{k,n}\}_{k=1}^{K}$
    append $K$ zeros to $\boldsymbol{\alpha}_n$;
    compute $\mathbf{K}_n = [\mathbf{K}_{1,n}\,\mathbf{K}_{2,n}\,\ldots\,\mathbf{K}_{n,n}]$;
    update $\boldsymbol{\alpha}_{n+1} = \boldsymbol{\alpha}_n + \mu\,\mathbf{K}_n^{\mathrm{T}}\,(\mathbf{y}_n - \mathbf{K}_n\boldsymbol{\alpha}_n)$;
    store regressors $\{\mathbf{r}_{k,n}\}_{k=1}^{K}$;
end
```

B. Graph Kernel LMS using Coherence-check

As follows from (8), the model order grows with both time, n, and network size, K, when new data samples arrive. This increase makes this model unsuitable for real-time applications and large-scale networks. The growing dimensionality of the dictionary is a well-known issue in single-node kernel methods [40], [41], [49]–[53], where several solutions have been proposed that learn a sparse, or fixed-size dictionary. Of these, the coherence-based sparsification schemes use a coherence metric [40], [52] between a candidate regressor and the current dictionary to decide whether to include the candidate in the dictionary. Given a set of data samples $\{\mathbf{r}_{k,i}\}_{k=1,i=1}^{K,n-1}$, various approaches can be employed to construct a CC-based sparse dictionary adaptively. In a centralized manner, one can consider regressors from all nodes at each time instant and test the coherence metric for each regressor $\mathbf{r}_{l,n}$, given by

$$\delta_{l,n} = \max_{\mathbf{r}_j\in\mathcal{D}_n} |\kappa(\mathbf{r}_{l,n}, \mathbf{r}_j)|, \tag{12}$$

where $\mathcal{D}_n$ denotes the dictionary obtained before testing regressor $\mathbf{r}_{l,n}$; the dictionary starts empty before running the algorithm. Given a predefined threshold, $\delta > 0$, if $\delta_{l,n} < \delta$, the regressor is added to the dictionary. The process continues over the remaining regressors, accounting for previous regressors added to the dictionary, until a predefined dictionary size, D, is achieved, or all the data samples are used.

Therefore, using the coherence check criterion, $\hat{y}_{l,n}$ in (8) can be rewritten as

$$\hat{y}_{l,n} = \sum_{i\in\mathcal{M}_n}\sum_{k\in\mathcal{K}_i} \alpha_{ik}\; \kappa(\mathbf{r}_{l,n}, \mathbf{r}_{k,i}), \tag{13}$$

where $\mathcal{M}_n$ is a set of time instants (up to time instant n) in which at least one input regressor is added to the dictionary, with $|\mathcal{M}_n| \le n$, and $\mathcal{K}_i$ is a set of node indices of the regressors that passed the coherence check at time index i, with $|\mathcal{K}_i| \le K$. Under the CC criterion, at time index n, the dictionary $\mathcal{D}_n$ contains $|\mathcal{D}_n| = \sum_{i\in\mathcal{M}_n} |\mathcal{K}_i|$ regressors.

Remark 1. Given a set of reasonable conditions on the threshold, δ, the maximum number of regressors in the dictionary is finite, i.e., $|\mathcal{D}_n|$ stops increasing after a certain time [52].

The coefficients of the expansion in (13) are obtained through the following minimization problem:

$$\min_{\widetilde{\alpha}_{ik}\in\mathbb{R}} \sum_{l=1}^{K} \mathrm{E}\left[\Big(y_{l,n} - \sum_{i\in\mathcal{M}_n}\sum_{k\in\mathcal{K}_i} \widetilde{\alpha}_{ik}\; \kappa(\mathbf{r}_{l,n}, \mathbf{r}_{k,i})\Big)^2\right]$$
$$= \min_{\widetilde{\boldsymbol{\alpha}}\in\mathbb{R}^{|\mathcal{D}_n|}} \mathrm{E}\left[\|\mathbf{y}_n - \widetilde{\mathbf{K}}_n\widetilde{\boldsymbol{\alpha}}\|_2^2\right], \tag{14}$$

where $\widetilde{\boldsymbol{\alpha}}^{\mathrm{T}} = [\widetilde{\boldsymbol{\alpha}}_1^{\mathrm{T}}\,\widetilde{\boldsymbol{\alpha}}_2^{\mathrm{T}}\,\ldots\,\widetilde{\boldsymbol{\alpha}}_{|\mathcal{M}_n|}^{\mathrm{T}}]$, with $\widetilde{\boldsymbol{\alpha}}_i^{\mathrm{T}} = [\widetilde{\alpha}_{i1}\,\widetilde{\alpha}_{i2}\,\ldots\,\widetilde{\alpha}_{i|\mathcal{K}_i|}] \in \mathbb{R}^{|\mathcal{K}_i|}$. The matrix $\widetilde{\mathbf{K}}_n$ is a Gram matrix given by

$$\widetilde{\mathbf{K}}_n = [\widetilde{\mathbf{K}}_{1,n}\,\widetilde{\mathbf{K}}_{2,n}\,\ldots\,\widetilde{\mathbf{K}}_{|\mathcal{M}_n|,n}] \in \mathbb{R}^{K\times|\mathcal{D}_n|}, \tag{15}$$

Algorithm 2: GKLMS using coherence check

```
Input: training data $\{\tilde{\mathbf{r}}_{l,i}, \tilde{y}_{l,i}\}_{l=1,i=1}^{K,t}$, dictionary size
       $D$, threshold $\delta$, and step size $\mu$
Initialization: $\mathcal{D} = \emptyset$, $\boldsymbol{\alpha}_0 =$ empty vector;
%Learning
for each time instant $n$ do
    Input: $\mathbf{y}_n, \{\mathbf{r}_{k,n}\}_{k=1}^{K}$
    for $k = 1, \ldots, K$ do
        if $|\mathcal{D}| < D$ then
            compute $\delta_{l,n} = \max_{\mathbf{r}_j \in \mathcal{D}} |\kappa(\mathbf{r}_{l,n}, \mathbf{r}_j)|$;
            if $\delta_{l,n} < \delta$ then
                add $\mathbf{r}_{l,n}$ to $\mathcal{D}$;
                add $l$ to $\mathcal{K}_n$;
            end
        end
    end
    if $|\mathcal{K}_n| \neq 0$ then
        append $|\mathcal{K}_n|$ zeros to $\widetilde{\alpha}_n$;
        add $n$ to $\mathcal{M}_n$;
    end
    compute $\widetilde{\mathbf{K}}_n = [\widetilde{\mathbf{K}}_{1,n}\, \widetilde{\mathbf{K}}_{2,n} \ldots \widetilde{\mathbf{K}}_{|\mathcal{M}_n|,n}]$;
    update $\widetilde{\boldsymbol{\alpha}}_{n+1} = \widetilde{\boldsymbol{\alpha}}_n + \mu\, \widetilde{\mathbf{K}}_n^{\mathrm{T}} (\mathbf{y}_n - \widetilde{\mathbf{K}}_n \widetilde{\boldsymbol{\alpha}}_n)$;
end
```

with $[\widetilde{\mathbf{K}}_{i,n}]_{l,k} = \kappa(\mathbf{r}_{l,n}, \mathbf{r}_{k,i})$, for $l \in \mathcal{N}$ and $k \in \mathcal{K}_i$.

Using the stochastic-gradient approach and minimizing the instantaneous value of (14), we obtain the following update rule of the centralized GKLMS using coherence check:

$$\widetilde{\boldsymbol{\alpha}}_{n+1} = \widetilde{\boldsymbol{\alpha}}_n + \mu\, \widetilde{\mathbf{K}}_n^{\mathrm{T}} (\mathbf{y}_n - \widetilde{\mathbf{K}}_n \widetilde{\boldsymbol{\alpha}}_n). \tag{16}$$

Algorithm 2 summarizes the steps for pre-training the dictionary according to the CC criterion and the learning stage of the CC-based GKLMS algorithm.

Remark 2. If coherence check is employed in an online fashion, two events must be considered for each candidate regressor. If the regressor does not satisfy the CC criterion, the dictionary remains the same. Otherwise, $\widetilde{\mathbf{K}}_n$ gets one new column and a zero-valued entry must be appended to $\widetilde{\boldsymbol{\alpha}}_n$ [52]. At every time instant i, for $i \in \mathcal{M}_n$, $|\mathcal{K}_i|$ regressors are added to the dictionary. Hence, $|\mathcal{K}_i|$ zeros must be appended to $\widetilde{\boldsymbol{\alpha}}_n$.

C. Graph Kernel LMS using Random Fourier Features

An alternative to sparsification methods, like CC, is provided by RFF [59]. The shift-invariant kernel evaluation $\kappa(\mathbf{r}_{l,n}, \mathbf{r}_{k,i}) = \kappa(\mathbf{r}_{l,n} - \mathbf{r}_{k,i})$ can be approximated as an inner product in the D-dimensional RFF space. This turns the problem into a finite-dimension linear filtering problem, while avoiding the evaluation of kernel functions [59]. Let $\mathbf{z}_{l,n}$ be the mapping of $\mathbf{r}_{l,n}$ into the RFF space $\mathbb{R}^D$, given by

$$\mathbf{z}_{l,n} = (D/2)^{-\frac{1}{2}} \left[\cos(\mathbf{v}_1^{\mathrm{T}}\mathbf{r}_{l,n} + b_1) \ldots \cos(\mathbf{v}_D^{\mathrm{T}}\mathbf{r}_{l,n} + b_D)\right]^{\mathrm{T}}, \tag{17}$$

Algorithm 3: GKLMS using RFF

```
Input: RFF-space dimension $D$, pdf $p(\mathbf{v})$, step size $\mu$
Initialization:
draw vectors $\{\mathbf{v}_i\}_{i=1}^{D}$ from $p(\mathbf{v})$;
draw phase terms $\{b_i\}_{i=1}^{D}$ from $[0, 2\pi]$;
$\mathbf{h}_0 = \mathbf{0}_D$;
%Learning
for each time instant $n$ do
    Input: $\mathbf{y}_n, \{\mathbf{r}_{k,n}\}_{k=1}^{K}$
    compute $\{\mathbf{z}_{l,n}\}_{l=1}^{K}$ using (17);
    construct matrix $\mathbf{Z}_n$ using (21);
    update $\mathbf{h}_{n+1} = \mathbf{h}_n + \mu \mathbf{Z}_n \mathbf{e}_n$;
end
```

where the phase terms $\{b_i\}_{i=1}^{D}$ are drawn from a uniform distribution on the interval $[0,\, 2\pi]$. Vectors $\{\mathbf{v}_i\}_{i=1}^{D}$ are drawn from the probability density function (pdf) $p(\mathbf{v})$ such that

$$k(\mathbf{r}_{l,n} - \mathbf{r}_{k,i}) = \int p(\mathbf{v}) \exp\left(\mathrm{j}\mathbf{v}^{\mathrm{T}}(\mathbf{r}_{l,n} - \mathbf{r}_{k,i})\right) \mathrm{d}\mathbf{v}, \tag{18}$$

where $\mathrm{j}^2 = -1$. In other words, the Fourier transform of $k(\mathbf{r}_{l,n} - \mathbf{r}_{k,i})$ is given by $p(\mathbf{v})$. From (17) and (18), it can be verified that $\mathrm{E}[\mathbf{z}_{k,i}^{\mathrm{T}}\mathbf{z}_{l,n}] = k(\mathbf{r}_{l,n}, \mathbf{r}_{k,i})$. Then, the kernel evaluation can be approximated as $\kappa(\mathbf{r}_{l,n}, \mathbf{r}_{k,i}) \approx \mathbf{z}_{k,i}^{\mathrm{T}}\mathbf{z}_{l,n}$ and the estimate $\hat{y}_{l,n}$ in (8) can be approximated by

$$\hat{y}_{l,n} \approx \Big(\sum_{i=1}^{n}\sum_{k=1}^{K} \alpha_{ik}\, \mathbf{z}_{k,i}\Big)^{\mathrm{T}} \mathbf{z}_{l,n} = \mathbf{h}^{\mathrm{T}}\mathbf{z}_{l,n}, \tag{19}$$

where $\mathbf{h} \in \mathbb{R}^D$ is the representation of the function $f(\cdot)$ in the RFF space. A higher value of D improves the approximation of the kernel function. Therefore, the choice of D depends mostly on the application, as it represents a trade-off between performance and complexity.

We note that, if a Gaussian kernel given by $\kappa(\mathbf{r}_{l,n}, \mathbf{r}_{k,i}) = \exp\left(-\|\mathbf{r}_{l,n} - \mathbf{r}_{k,i}\|_2^2/(2\sigma^2)\right)$ is used, the pdf $p(\mathbf{v})$ is given in closed form as a normal distribution. See [59] for closed-form representations of $p(\mathbf{v})$ when other kernel functions are used.

The linear representation of $f(\cdot)$ in the RFF space, $\mathbf{h}$, can be estimated by solving the following optimization problem:

$$\min_{\mathbf{h} \in \mathbb{R}^D} \mathrm{E}\left[\|\mathbf{y}_n - \mathbf{Z}_n^{\mathrm{T}}\mathbf{h}\|_2^2\right], \tag{20}$$

where the matrix

$$\mathbf{Z}_n = [\mathbf{z}_{1,n}\, \mathbf{z}_{2,n} \ldots \mathbf{z}_{K,n}] \tag{21}$$

represents the RFF mapping of all input vectors at time n. Similar to (16), approximating the solution through stochastic-gradient descent iterations yields the RFF-based centralized graph kernel LMS (GKLMS) update rule

$$\mathbf{h}_{n+1} = \mathbf{h}_n + \mu \mathbf{Z}_n \mathbf{e}_n, \tag{22}$$

where $\mathbf{e}_n = \mathbf{y}_n - \mathbf{Z}_n^{\mathrm{T}}\mathbf{h}_n$. The proposed GKLMS using RFF is summarized in Algorithm 3.

Notice that the estimates $\widetilde{\boldsymbol{\alpha}}$ in (16) and $\mathbf{h}$ in (22) require knowledge of the input of the entire graph, which can be impractical in applications without a centralized processing unit. Therefore, we propose a distributed implementation of the GKLMS, named graph diffusion KLMS (GDKLMS).

Remark 3. A CC-based distributed implementation requires a pre-trained dictionary available at each node [40]. The dictionary can be pre-trained in a centralized way and broadcasted to the entire network, or by a single node that shares its dictionary with all nodes. More importantly, the dictionary depends on available training data, and may be retrained whenever there are changes in the underlying model. Therefore, RFF-based algorithms seem more suitable for distributed implementations and robust to changes in model and data statistics.

D. Graph Diffusion Kernel LMS using RFF

In order to derive a distributed implementation, the global optimization problem (20) is expressed alternatively in the following separable form:

$$\underset{\boldsymbol{\psi}_1,\ldots,\boldsymbol{\psi}_K\in\mathbb{R}^D}{\arg\min} \sum_{k=1}^{K} \mathrm{E}\big[(y_{k,n}-\mathbf{z}_{k,n}^{\mathrm{T}}\boldsymbol{\psi}_k)^2\big], \tag{23}$$

where $\boldsymbol{\psi}_k$ is the local estimate of $\mathbf{h}$ at node k. The optimization problem in (23) can be solved in a distributed fashion by minimizing $\mathrm{E}\left[(y_{k,n}-\mathbf{z}_{k,n}^{\mathrm{T}}\boldsymbol{\psi}_k)^2\right]$ at each node. Let $e_{k,n}=y_{k,n}-\mathbf{z}_{k,n}^{\mathrm{T}}\boldsymbol{\psi}_k$. Following the similar lines of centralized GKLMS, the update rule for $\boldsymbol{\psi}_k$ is given by

$$\boldsymbol{\psi}_{k,n+1}=\boldsymbol{\psi}_{k,n}+\mu\, e_{k,n}\mathbf{z}_{k,n}. \tag{24}$$

We now leverage the graph structure and adopt the adapt-then-combine (ATC) strategy to improve individual estimates via graph diffusion [22], [23], [26], [40], [65]. The ATC strategy is one common diffusion strategy composed by two steps. At iteration n, the first step updates the local estimate, at a given node k, using the new input $\{\mathbf{r}_{k,n}, y_{k,n}\}$, generating an intermediate estimate. In the second step, nodes share and combine their intermediate estimates to generate the final estimate for that iteration. That is, the parameter update of $\mathbf{h}_{k,n}$ at node k is obtained by combining the estimates from its neighborhood. Note that the graph structure defines a node's neighborhood, and adjacent nodes relate to each other. The ATC update rule for the GDKLMS using RFF is given by

$$\begin{cases} \boldsymbol{\psi}_{k,n+1}=\mathbf{h}_{k,n}+\mu\, e_{k,n}\mathbf{z}_{k,n}, & \text{(25a)}\\ \mathbf{h}_{k,n+1}=\displaystyle\sum_{l\in\mathcal{N}_k} a_{lk}\,\boldsymbol{\psi}_{l,n+1}, & \text{(25b)}\end{cases}$$

where the combination coefficients a_{lk} are non-negative and satisfy the condition $\sum_{l\in\mathcal{N}_k} a_{lk}=1$ [26]. We could use a similar combine-then-adapt (CTA) strategy [65]. Both ATC and CTA strategies share fundamentally the same structure and yield similar results [27]. Algorithm 4 summarizes the steps of the GDKLMS implementation using RFF.

Algorithm 4: GDKLMS using RFF

```
Input: RFF-space dimension $D$, pdf $p(\mathbf{v})$, step size $\mu$,
       combination coefficients $a_{lk}$
Initialization:
draw vectors $\{\mathbf{v}_i\}_{i=1}^D$ from $p(\mathbf{v})$;
draw phase terms $\{b_i\}_{i=1}^D$ from $[0,2\pi]$;
$\mathbf{h}_{k,0}=\mathbf{0}_D$, $\forall k\in\{1,2,\ldots,K\}$;
$\boldsymbol{\psi}_{k,0}=\mathbf{0}_D$, $\forall k\in\{1,2,\ldots,K\}$;
%Learning
for each time instant $n$ do
    for $k=1,\ldots,K$ do
        compute $\mathbf{z}_{k,n}$ using (17);
        update $\boldsymbol{\psi}_{k,n+1}=\mathbf{h}_{k,n}+\mu\, e_{k,n}\mathbf{z}_{k,n}$;
    end
    for $k=1,\ldots,K$ do
        update $\mathbf{h}_{k,n+1}=\sum_{l\in\mathcal{N}_k} a_{lk}\,\boldsymbol{\psi}_{l,n+1}$;
    end
end
```

For this, at network level, we define the filter coefficient vector in the RFF space $\mathbf{h}_g=\mathbf{1}_K\otimes\mathbf{h}$, the estimated filter coefficient vector in RFF space $\mathbf{h}_{g,n}=[\mathbf{h}_{1,n}^{\mathrm{T}}\,\mathbf{h}_{2,n}^{\mathrm{T}}\,\ldots\,\mathbf{h}_{K,n}^{\mathrm{T}}]^{\mathrm{T}}$, and the (RFF-mapped) input data matrix $\boldsymbol{\mathcal{Z}}_n=\mathrm{blockdiag}\{\mathbf{z}_{1,n},\mathbf{z}_{2,n},\ldots,\mathbf{z}_{K,n}\}$. In these definitions, $\mathbf{1}_K$ is a vector of size $K\times 1$ with every entry taking the value one, $\otimes$ denotes the right Kronecker product operator, and $\mathrm{blockdiag}\{\cdot\}$ denotes the block-diagonal-stacking operator. Using these definitions, the network-level data model is given by

$$\mathbf{y}_n=\boldsymbol{\mathcal{Z}}_n^{\mathrm{T}}\mathbf{h}_g+\boldsymbol{\upsilon}_n. \tag{26}$$

From these definitions, the network-level recursion of the RFF-based GDKLMS can then be expressed as follows:

$$\mathbf{h}_{g,n+1}=\boldsymbol{\mathcal{A}}\big(\mathbf{h}_{g,n}+\mu\boldsymbol{\mathcal{Z}}_n\mathbf{e}_n\big), \tag{27}$$

where $\boldsymbol{\mathcal{A}}=\mathbf{A}^{\mathrm{T}}\otimes\mathbf{I}_D$. The matrix $\mathbf{A}$, with $[\mathbf{A}]_{l,k}=a_{lk}$, is a $K\times K$ left stochastic matrix (i.e., each column consists of non-negative real numbers whose sum is unity). In the following, we study the convergence behavior of the proposed RFF-based GDKLMS governed by the form (27). For this, we assume the following:

- **A1**: Given a node $k\in\mathcal{N}$, the RFF-mapped data signal $\mathbf{z}_{k,n}$ is drawn from a WSS multivariate random sequence with correlation matrix $\mathbf{R}_{z,k}=\mathrm{E}[\mathbf{z}_{k,n}\mathbf{z}_{k,n}^{\mathrm{T}}]$; in addition, the data vectors $\mathbf{z}_{k,n}$ and $\mathbf{z}_{l,m}$ are independent, for all $k\neq l\in\mathcal{N}$.
- **A2**: The observation noise $\boldsymbol{\upsilon}_n$ is a zero-mean WSS multivariate random sequence, with diagonal correlation matrix $\mathbf{R}_\upsilon=\mathrm{E}[\boldsymbol{\upsilon}_n\boldsymbol{\upsilon}_n^{\mathrm{T}}]=\mathrm{diag}\{\sigma_{\upsilon,1}^2,\sigma_{\upsilon,2}^2,\ldots,\sigma_{\upsilon,K}^2\}$, being independent of any other random signal in the model.
- **A3**: The weight vector $\mathbf{h}_{k,n}$ is taken to be independent of $\mathbf{z}_{k,n}$, for $k\in\mathcal{N}$.
- **A4**: The graph topology is assumed to be static, meaning the shift matrix $\mathbf{S}$ and the combiner coefficients a_{lk} are constant throughout the process.

IV. Convergence Analysis

In this section, we study the performance of the proposed RFF-based GDKLMS and establish the conditions for its convergence both in mean and mean-squared senses.

A5: The step size μ is sufficiently small so that the terms involving higher order powers of μ can be ignored.

The above assumptions are commonly used in the analysis of distributed adaptive schemes over networks.

Remark 4. Note that the vector $\mathbf{z}_{k,n}$ is the representation of $\mathbf{r}_{k,n}$ in the RFF space. Hence, $\mathbf{z}_{k,n}$ may not be normally distributed. If the basis of the RFF space is generated in a way such that the basis vectors $\mathbf{v}_i \neq \mathbf{v}_j$ for any $i \neq j$, the autocorrelation matrix $\mathbf{R}_{z,k}$, for $k \in \mathcal{N}$ will be strictly positive definite [47].

Apart from these assumptions, the analysis also requires properties of the block maximum norm of a matrix (i.e., $\|\cdot\|_{\mathrm{b},\infty}$), the block vectorization operator (i.e., bvec$\{\cdot\}$) [27], and the block Kronecker product of two matrices (i.e., $\otimes_{\mathrm{b}}$) [66]. Details of these operators can be found in [27], [66], [67].

A. First-order Convergence Analysis

Denoting the global weight deviation vector of the proposed GDKLMS using RFF, at time instant n, as $\tilde{\mathbf{h}}_{g,n} = \mathbf{h}_g - \mathbf{h}_{g,n}$, recalling the fact that $\boldsymbol{\mathcal{A}}\mathbf{h}_g = \mathbf{h}_g$ (since the matrix $\mathbf{A}$ is left stochastic), from (27), the recursion for $\tilde{\mathbf{h}}_{g,n}$ can then be obtained as

$$\tilde{\mathbf{h}}_{g,n+1} = \boldsymbol{\mathcal{B}}_n \tilde{\mathbf{h}}_{g,n} - \mu\, \boldsymbol{\mathcal{A}} \boldsymbol{\mathcal{Z}}_n \boldsymbol{\upsilon}_n, \tag{28}$$

where $\boldsymbol{\mathcal{B}}_n = \boldsymbol{\mathcal{A}}\big(\mathbf{I}_{DK} - \mu \boldsymbol{\mathcal{Z}}_n \boldsymbol{\mathcal{Z}}_n^{\mathrm{T}}\big)$. In the following, we establish the condition for the convergence in mean.

Theorem 1. Assume the data model (26) and the assumptions **A1**-**A4** hold (assumption **A5** is not required here). Then a sufficient condition for the proposed RFF-based GDKLMS to converge in mean is given by

$$0 < \mu < \frac{2}{\max\limits_{1\leq k\leq K} \big\{ \max\limits_{1\leq i\leq D} \{\lambda_i(\, \mathbf{R}_{z,k})\}\big\}}. \tag{29}$$

Proof. Taking the statistical expectation E$[\cdot]$ on both sides of (28), and using the assumptions **A1**-**A4**, we obtain

$$\mathrm{E}[\tilde{\mathbf{h}}_{g,n+1}] = \overline{\boldsymbol{\mathcal{B}}}\ \mathrm{E}[\tilde{\mathbf{h}}_{g,n}], \tag{30}$$

with $\overline{\boldsymbol{\mathcal{B}}} = \mathrm{E}[\boldsymbol{\mathcal{B}}_n] = \boldsymbol{\mathcal{A}}\big(\mathbf{I}_{DK} - \mu \mathbf{R}_z\big)$, where $\mathbf{R}_z = \mathrm{E}[\boldsymbol{\mathcal{Z}}_n \boldsymbol{\mathcal{Z}}_n^{\mathrm{T}}]$ $= \mathrm{blockdiag}(\mathbf{R}_{z,1}, \mathbf{R}_{z,2}, \ldots, \mathbf{R}_{z,K})$.

From (30), it is easily seen that $\lim_{n\to\infty} \mathrm{E}\big[\tilde{\mathbf{h}}_{g,n}\big]$ attains a finite value if and only if $\|\overline{\boldsymbol{\mathcal{B}}}\| < 1$, where $\|\cdot\|$ denotes any matrix norm. We derive a convergence condition in terms of μ, by constraining the block maximum norm of the matrix $\overline{\boldsymbol{\mathcal{B}}}$ (i.e., $\|\overline{\boldsymbol{\mathcal{B}}}\|_{\mathrm{b},\infty}$). Using the properties of block maximum norm [26], we can write

$$\|\overline{\boldsymbol{\mathcal{B}}}\|_{\mathrm{b},\infty} \leq \|\boldsymbol{\mathcal{A}}\|_{\mathrm{b},\infty} \|\mathbf{I}_{DK} - \mu \mathbf{R}_z\|_{\mathrm{b},\infty}. \tag{31}$$

Since the matrix $\mathbf{A}$ is left stochastic, we have $\|\boldsymbol{\mathcal{A}}\|_{\mathrm{b},\infty} = \|\mathbf{A}^{\mathrm{T}} \otimes \mathbf{I}_D\|_{\mathrm{b},\infty} = 1$. Furthermore, as the matrix $(\mathbf{I}_{DK} - \mu \mathbf{R}_z)$ is block diagonal symmetric, using [26, Lemma D. 3(a), D. 5], a sufficient condition for $\mathrm{E}[\tilde{\mathbf{h}}_{g,n}]$ to converge in mean is given by $\rho(\mathbf{I}_{DK} - \mu \mathbf{R}_z) < 1$, or, equivalently, $|1 - \mu\lambda_j(\mathbf{R}_z))| < 1$ for $j \in \{1, 2, \ldots, DK\}$, where $\rho(\cdot)$ denotes the spectral radius of the argument matrix and $\lambda_j(\mathbf{R}_z)$ denotes the jth eigenvalue of $\mathbf{R}_z$. After solving this, we arrive at (29). □

B. Second-order Convergence Analysis

Next, we focus on the second-order convergence analysis of the proposed GDKLMS using RFF. Using the energy conservation approach, we investigate the steady-state MSE performance of the proposed scheme.

Defining the $\boldsymbol{\Sigma}$-weighted norm-square of $\tilde{\mathbf{h}}_{g,n}$ as $\|\tilde{\mathbf{h}}_{g,n}\|^2_{\boldsymbol{\Sigma}} = \tilde{\mathbf{h}}_{g,n}^{\mathrm{T}} \boldsymbol{\Sigma} \tilde{\mathbf{h}}_{g,n}$, where $\boldsymbol{\Sigma}$ is a positive semi-definite matrix that can be chosen arbitrarily, and using the assumptions **A1**-**A4**, one can write

$$\begin{aligned}\mathrm{E}\left[\|\tilde{\mathbf{h}}_{g,n+1}\|^2_{\boldsymbol{\Sigma}}\right] &= \mathrm{E}\left[\|\tilde{\mathbf{h}}_{g,n}\|^2_{\boldsymbol{\Sigma}'}\right] \\ &\quad + \mu^2 \mathrm{E}[\boldsymbol{\upsilon}_n^{\mathrm{T}} \boldsymbol{\mathcal{Z}}_n^{\mathrm{T}} \boldsymbol{\mathcal{A}}^{\mathrm{T}} \boldsymbol{\Sigma} \boldsymbol{\mathcal{A}} \boldsymbol{\mathcal{Z}} \boldsymbol{\upsilon}_n],\end{aligned} \tag{32}$$

where the cross terms are zero since $\boldsymbol{\upsilon}_n$ is taken to be zero-mean and statistically independent of $\mathbf{z}_{k,n}$. The matrix $\boldsymbol{\Sigma}'$ is given by

$$\boldsymbol{\Sigma}' = \mathrm{E}[\boldsymbol{\mathcal{B}}_n^{\mathrm{T}} \boldsymbol{\Sigma} \boldsymbol{\mathcal{B}}_n]. \tag{33}$$

Now, using the block Kronecker product denoted by $\otimes_{\mathrm{b}}$ [66] and the bvec$\{\cdot\}$ operator [66], we can relate the vectors $\boldsymbol{\sigma} = \mathrm{bvec}\{\boldsymbol{\Sigma}\}$ and $\boldsymbol{\sigma}' = \mathrm{bvec}\{\boldsymbol{\Sigma}'\}$ as

$$\boldsymbol{\sigma}' = \boldsymbol{\mathcal{F}}^{\mathrm{T}} \boldsymbol{\sigma}, \tag{34}$$

with

$$\boldsymbol{\mathcal{F}} = \mathrm{E}[\boldsymbol{\mathcal{B}}_n \otimes_{\mathrm{b}} \boldsymbol{\mathcal{B}}_n] = (\boldsymbol{\mathcal{A}} \otimes \boldsymbol{\mathcal{A}}) \boldsymbol{\mathcal{H}}, \tag{35}$$

where

$$\boldsymbol{\mathcal{H}} \approx \mathbf{I}_{D^2K^2} - \mu(\mathbf{R}_z \otimes_{\mathrm{b}} \mathbf{I}_{DK}) - \mu(\mathbf{I}_{DK} \otimes_{\mathrm{b}} \mathbf{R}_z). \tag{36}$$

In the above expression, using the assumption **A5**, the terms involving high-order powers of μ are ignored, and we continue our analysis with this approximation. Note that this approximation is standard in the analysis of many distributed schemes over networks [26], [27].

Now, consider the second term on the right-hand side of (32). We can write it as

$$\begin{aligned}&\mathrm{E}[\boldsymbol{\upsilon}_n^{\mathrm{T}} \boldsymbol{\mathcal{Z}}_n^{\mathrm{T}} \boldsymbol{\mathcal{A}}^{\mathrm{T}} \boldsymbol{\Sigma} \boldsymbol{\mathcal{A}} \boldsymbol{\mathcal{Z}} \boldsymbol{\upsilon}_n] \\ &\quad = \mathrm{Tr}\left(\mathrm{E}[\boldsymbol{\upsilon}_n^{\mathrm{T}} \boldsymbol{\mathcal{Z}}_n^{\mathrm{T}} \boldsymbol{\mathcal{A}}^{\mathrm{T}} \boldsymbol{\Sigma} \boldsymbol{\mathcal{A}} \boldsymbol{\mathcal{Z}} \boldsymbol{\upsilon}_n]\right) \\ &\quad = \mathrm{Tr}\left(\boldsymbol{\mathcal{A}} \mathrm{E}[\boldsymbol{\mathcal{Z}}_n \boldsymbol{\upsilon}_n \boldsymbol{\upsilon}_n^{\mathrm{T}} \boldsymbol{\mathcal{Z}}_n^{\mathrm{T}}] \boldsymbol{\mathcal{A}}^{\mathrm{T}} \boldsymbol{\Sigma}\right) \\ &\quad = \mathrm{Tr}\left(\boldsymbol{\mathcal{A}} \mathrm{E}[\boldsymbol{\Phi}_n] \boldsymbol{\mathcal{A}}^{\mathrm{T}} \boldsymbol{\Sigma}\right),\end{aligned} \tag{37}$$

where $\boldsymbol{\Phi}_n = \boldsymbol{\mathcal{Z}}_n \mathbf{R}_\upsilon \boldsymbol{\mathcal{Z}}_n^{\mathrm{T}}$.

Using the properties of block Kronecker product [66], we finally have

$$\mathrm{Tr}\left(\boldsymbol{\mathcal{A}} \mathrm{E}[\boldsymbol{\Phi}_n] \boldsymbol{\mathcal{A}}^{\mathrm{T}} \boldsymbol{\Sigma}\right) = \boldsymbol{\gamma}^{\mathrm{T}} \boldsymbol{\sigma}, \tag{38}$$

where $\boldsymbol{\gamma} = \mathrm{bvec}\{\boldsymbol{\mathcal{A}} \mathrm{E}[\boldsymbol{\Phi}_n] \boldsymbol{\mathcal{A}}^{\mathrm{T}}\} = (\boldsymbol{\mathcal{A}} \otimes \boldsymbol{\mathcal{A}}) \boldsymbol{\gamma}_{\boldsymbol{\upsilon}}$, with

$$\begin{aligned}\boldsymbol{\gamma}_{\boldsymbol{\upsilon}} &= \mathrm{bvec}\{\mathrm{E}[\boldsymbol{\Phi}_n]\} \\ &= \mathrm{bvec}\{\mathrm{E}[\boldsymbol{\mathcal{Z}}_n \mathbf{R}_\upsilon \boldsymbol{\mathcal{Z}}_n^{\mathrm{T}}]\} \\ &= \mathrm{E}[\boldsymbol{\mathcal{Z}}_n \otimes_{\mathrm{b}} \boldsymbol{\mathcal{Z}}_n] \cdot \mathrm{bvec}\{\mathbf{R}_\upsilon\}.\end{aligned} \tag{39}$$

Combining these results, (32) can be expressed as

$$\begin{aligned}\mathrm{E}\left[\|\tilde{\mathbf{h}}_{g,n+1}\|^2_{\mathrm{bvec}^{-1}\{\boldsymbol{\sigma}\}}\right] &= \mathrm{E}\left[\|\tilde{\mathbf{h}}_{g,n}\|^2_{\mathrm{bvec}^{-1}\{\boldsymbol{\mathcal{F}}^{\mathrm{T}}\boldsymbol{\sigma}\}}\right] \\ &\quad + \mu^2 \boldsymbol{\gamma}^{\mathrm{T}} \boldsymbol{\sigma},\end{aligned} \tag{40}$$

where $\text{bvec}^{-1}\{\cdot\}$ rearranges the argument vector of size $D^2K^2 \times 1$ into a $DK \times DK$ matrix, i.e., $\boldsymbol{\Sigma} = \text{bvec}^{-1}\{\boldsymbol{\sigma}\}$.

Theorem 2. Assume the data model (26) and that assumptions **A1-A5** hold. Furthermore, assume that the step size μ is sufficiently small such that the approximation (36) is justified by ignoring the higher-order powers of μ, so that (40) can be used as a reasonable representation for studying the dynamics of the weighted mean-squared deviation (MSD). Then, the proposed RFF-based GDKLMS converges in mean-squared sense under

$$0 < \mu < \frac{1}{\max\limits_{1\le k\le K}\left\{\max\limits_{1\le i\le D}\{\lambda_i(\mathbf{R}_{z,k})\}\right\}}. \tag{41}$$

Proof. Iterating the recursion (40) backwards down to $n = 0$, we obtain

$$\begin{aligned}\mathrm{E}\left[\|\tilde{\mathbf{h}}_{g,n+1}\|^2_{\text{bvec}^{-1}\{\boldsymbol{\sigma}\}}\right] =& \mathrm{E}\left[\|\tilde{\mathbf{h}}_{g,0}\|^2_{\text{bvec}^{-1}\{(\boldsymbol{\mathcal{F}}^{\mathrm{T}})^{n+1}\boldsymbol{\sigma}\}}\right] \\ &+ \mu^2\boldsymbol{\gamma}^{\mathrm{T}}\left(\mathbf{I}_{D^2K^2} + \sum_{i=1}^{n}\left(\boldsymbol{\mathcal{F}}^{\mathrm{T}}\right)^i\right)\boldsymbol{\sigma},\end{aligned} \tag{42}$$

where $\tilde{\mathbf{h}}_{g,0} = \mathbf{h}_g - \mathbf{h}_{g,0}$. Note that under $\|\boldsymbol{\mathcal{F}}^{\mathrm{T}}\| = \|\boldsymbol{\mathcal{F}}\| < 1$, we will have $(\boldsymbol{\mathcal{F}}^{\mathrm{T}})^{n+1} \to \mathbf{0}_{D^2K^2}$ as $n \to \infty$. Hence, $\mathrm{E}[\|\tilde{\mathbf{h}}_{g,n}\|^2_{\text{bvec}^{-1}\{\boldsymbol{\sigma}\}}]$ attains a finite value. Therefore, a sufficient condition for convergence of $\mathrm{E}[\|\tilde{\mathbf{h}}_{g,n+1}\|^2_{\boldsymbol{\sigma}}]$ is then given by $\|\boldsymbol{\mathcal{F}}\| < 1$. To derive a convergence condition in terms of μ, we use the block maximum norm of the matrix $\boldsymbol{\mathcal{F}}$, i.e., $\|\boldsymbol{\mathcal{F}}\|_{\mathrm{b},\infty}$. From the properties of the block maximum norm [26], we can write

$$\|\boldsymbol{\mathcal{F}}\|_{\mathrm{b},\infty} = \|(\boldsymbol{\mathcal{A}} \otimes_{\mathrm{b}} \boldsymbol{\mathcal{A}})\boldsymbol{\mathcal{H}}\|_{\mathrm{b},\infty} \le \|(\boldsymbol{\mathcal{A}} \otimes_{\mathrm{b}} \boldsymbol{\mathcal{A}})\|_{\mathrm{b},\infty}\|\boldsymbol{\mathcal{H}}\|_{\mathrm{b},\infty}. \tag{43}$$

The term $(\boldsymbol{\mathcal{A}}\otimes_{\mathrm{b}}\boldsymbol{\mathcal{A}})$ can be written as $(\mathbf{A}\otimes\mathbf{A})^{\mathrm{T}}\otimes(\mathbf{I}_D\otimes\mathbf{I}_D)$. Again, from the properties of the block maximum norm, we have $\|\boldsymbol{\mathcal{A}} \otimes_{\mathrm{b}} \boldsymbol{\mathcal{A}}\|_{\mathrm{b},\infty} = \|(\mathbf{A}\otimes\mathbf{A})^{\mathrm{T}} \otimes \mathbf{I}_{D^2}\|_{\mathrm{b},\infty} = 1$. Now, substituting the definition of $\boldsymbol{\mathcal{H}}$ as given by (36), we have

$$\|\boldsymbol{\mathcal{F}}\|_{\mathrm{b},\infty} \le \|\mathbf{I}_{D^2K^2} - \mu(\mathbf{R}_z \otimes_{\mathrm{b}} \mathbf{I}_{DK}) - \mu(\mathbf{I}_{DK} \otimes_{\mathrm{b}} \mathbf{R}_z)\|_{\mathrm{b},\infty}. \tag{44}$$

Since the argument of the norm on the right-hand side of (44) is a block diagonal symmetric matrix, from the properties of block maximum norm, it is seen that $\mathrm{E}[\|\tilde{\mathbf{h}}_{g,n}\|^2_{\text{bvec}^{-1}\{\boldsymbol{\sigma}\}}]$ converges under

$$\rho(\|\mathbf{I}_{D^2K^2} - \mu(\mathbf{R}_z\otimes_{\mathrm{b}}\mathbf{I}_{DK}) - \mu(\mathbf{I}_{DK}\otimes_{\mathrm{b}}\mathbf{R}_z)\|_{\mathrm{b},\infty}) < 1, \tag{45}$$

or, equivalently,

$$|1 - \mu\lambda_p(\mathbf{R}_z) - \mu\lambda_q(\mathbf{R}_z)| < 1, \quad p, q \in \{1, 2, \ldots, DK\}. \tag{46}$$

Note that $(\mathbf{R}_z \otimes_{\mathrm{b}} \mathbf{I}_{DK})$ and $(\mathbf{I}_{DK} \otimes_{\mathrm{b}} \mathbf{R}_z)$ have the same set of eigenvectors and eigenvalues. Also, $\lambda_l(\mathbf{R}_z)$ has multiplicity of DK, for $l \in \mathcal{N}$. After solving the above condition, we obtain the mean-squared convergence condition on μ given in (41). □

Remark 5. We observe that the bounds established for μ are inversely proportional to the spectral radius of the covariance matrix of vectors $\mathbf{z}_k$. Hence, similar to conventional stochastic gradient algorithms, μ requires tuning according to the largest eigenmode.

C. Steady-State Mean-Squared Error

For μ under (41), letting $n \to \infty$ on both sides of (40), we have

$$\lim_{n\to\infty} \mathrm{E}\left[\|\tilde{\mathbf{h}}_{g,n}\|^2_{\text{bvec}^{-1}\{(\mathbf{I}_{D^2K^2}-\boldsymbol{\mathcal{F}}^{\mathrm{T}})\boldsymbol{\sigma}\}}\right] = \mu^2\boldsymbol{\gamma}^{\mathrm{T}}\boldsymbol{\sigma}. \tag{47}$$

By selecting $\boldsymbol{\sigma} = (\mathbf{I}_{D^2K^2} - \boldsymbol{\mathcal{F}}^{\mathrm{T}})^{-1}\text{bvec}(\mathbf{R}_z)$, (47) becomes

$$\lim_{n\to\infty} \mathrm{E}\left[\|\tilde{\mathbf{h}}_{g,n}\|^2_{\mathbf{R}_z}\right] = \mu^2\boldsymbol{\gamma}^{\mathrm{T}}(\mathbf{I}_{D^2K^2} - \boldsymbol{\mathcal{F}}^{\mathrm{T}})^{-1}\text{bvec}(\mathbf{R}_z). \tag{48}$$

Using (48), the network-level steady-state mean-squared error (SMSE) of the proposed RFF-based GDKLMS is given by

$$\begin{aligned}\text{SMSE} &= \frac{1}{K}\lim_{n\to\infty}\mathrm{E}[\mathbf{e}_n^{\mathrm{T}}\mathbf{e}_n] \\ &= \frac{1}{K}\lim_{n\to\infty}\left[\mathrm{E}[\tilde{\mathbf{h}}_{g,n}^{\mathrm{T}}\boldsymbol{\mathcal{Z}}_n\boldsymbol{\mathcal{Z}}_n^{\mathrm{T}}\tilde{\mathbf{h}}_{g,n}] + \mathrm{E}[\boldsymbol{\upsilon}_n^{\mathrm{T}}\boldsymbol{\upsilon}_n]\right] \\ &= \frac{1}{K}\left[\lim_{n\to\infty}\mathrm{E}[\|\tilde{\mathbf{h}}_{g,n}\|^2_{\mathbf{R}_z}] + \lim_{n\to\infty}\mathrm{E}[\boldsymbol{\upsilon}_n^{\mathrm{T}}\boldsymbol{\upsilon}_n]\right] \\ &= \frac{1}{K}\left[\mu^2\boldsymbol{\gamma}^{\mathrm{T}}(\mathbf{I}_{D^2K^2} - \boldsymbol{\mathcal{F}}^{\mathrm{T}})^{-1}\text{bvec}(\mathbf{R}_z) + \text{tr}(\mathbf{R}_\upsilon)\right].\end{aligned} \tag{49}$$

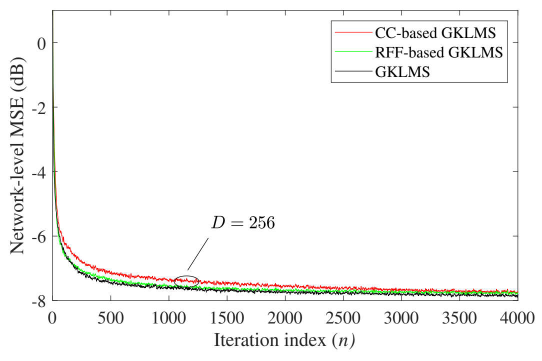
V. Complexity Analysis

This section details the computational complexity of the proposed algorithms. For the GKLMS algorithm, the Gram matrix computation (10) requires a total of nK^2 kernel evaluations. The complexity of kernel evaluations is treated separately, as we do not consider a specific kernel function. The computational cost of (11) is $2nK^2 + nk$ multiplications and $2nK^2$ additions. These values reveal that kernel methods' complexity does not scale well with time and network size without using techniques to deal with the growing dictionary.

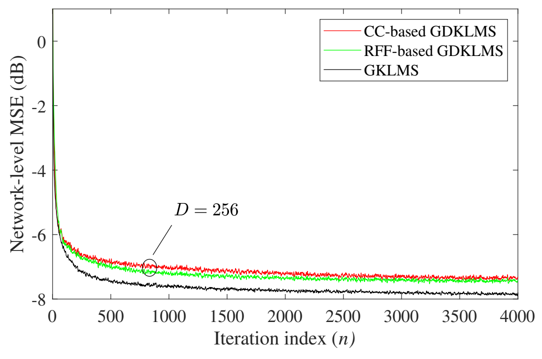
CC-based sparsification requires $K|\mathcal{D}|$ kernel evaluations per iterations for computing the Gram matrix, where $\mathcal{D}$ denotes the dictionary size, and $|\mathcal{D}|(2K+1)$ multiplications and $2K|\mathcal{D}|$ additions for the parameter update. The CC-based approach also requires dictionary training, and the minimum number of kernel evaluations for training is $|\mathcal{D}|(|\mathcal{D}|-1)/2$, assuming the first $|\mathcal{D}|$ regressors are added to the dictionary. An upper bound for the training process is $t|\mathcal{D}|$ kernel evaluations, where t is the number of training data samples.

For the RFF-based computation, the mapping's complexity into RFF space is assumed similar to that of the kernel evaluation. In this case, the RFF-GKLMS requires KD kernel evaluations for the mapping, and $D(2K + 1)$ multiplications and $2KD$ additions for the update, where D denotes the dimension of the RFF space. Considering the case where $|\mathcal{D}|$ and D are the same for the CC- and RFF-based implementations, their complexities per iteration are also the same. The CC-based approach, however, has the added complexity of training the dictionary.

Finally, The GDKLMS using RFF requires, at each node, $D(|\mathcal{N}|+3)$ multiplications and $D(|\mathcal{N}|+1)$ additions, with $|\mathcal{N}|$

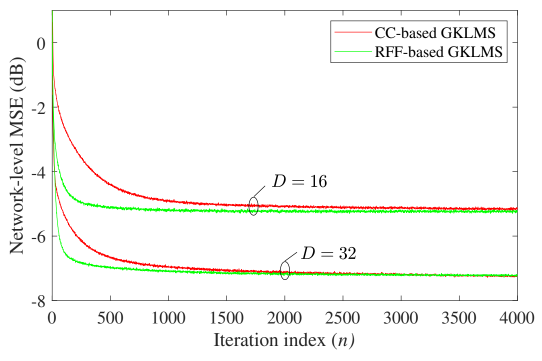


(a) Centralized solutions.

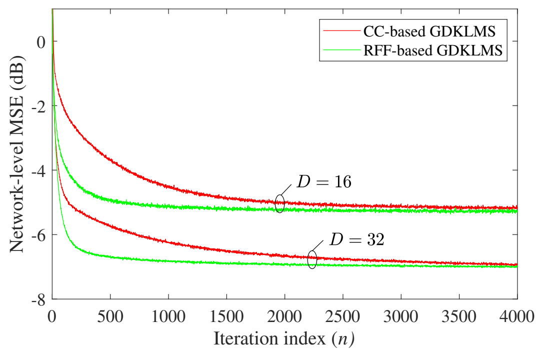


(b) Distributed solutions.

Fig. 1. Learning curves (network-level MSE vs iteration index) for the proposed algorithms with large dictionary size and RFF-space dimension.



(a) Centralized solutions.



(b) Distributed solutions.

Fig. 2. Learning curves (network-level MSE vs iteration index) for the proposed algorithms considering small values for D.

denoting the node's neighborhood cardinality. The mapping into the RFF space needs D kernel evaluations.

VI. Numerical Results

This section demonstrates the performance of the proposed algorithms through extensive numerical experiments under synthetic and real network data. We exclude comparisons with state-of-the-art methods based on the linear model (3) because their performance in the considered setting will be poor. In all simulations, the value of δ is adjusted as a function of the target dictionary size, such that we can reach the target size while still having a representative dictionary.

A. Nonlinear Graph Filter Identification

First, we consider a connected Erdös-Renyi graph comprising $K = 20$ nodes with edge probability equal to 0.2. The shift matrix $\mathbf{S}$ is constructed as follows: first, the existing edges, according to the previously constructed graph, receive a weight value drawn from the uniform distribution in the interval $(0, 1]$; each entry s_{kl} receives the value of the corresponding edge weight or zero if the edge does not exist; the eigenvalues $\{\lambda_k\}_{k=1}^{K}$ of $\mathbf{S}$ are normalized by the largest eigenvalue such that $|\lambda_k| \leq 1$. Input signal $\mathbf{x}_n$ and observation noise $\boldsymbol{\upsilon}_n$ are drawn from zero-mean normal distributions with covariance matrices $\mathbf{R}_{\mathbf{x}} = \operatorname{diag}\{\sigma^2_{x,k}\}$ and $\mathbf{R}_{\boldsymbol{\upsilon}} = \operatorname{diag}\{\sigma^2_{\upsilon,k}\}$, respectively, where $\sigma^2_{x,k}$ are drawn from the uniform distribution in $[1, 1.5]$ and $\sigma^2_{\upsilon,k}$ from $[0.1, 0.15]$. For distributed implementations, the combination coefficients a_{kl} are computed according to the Metropolis rule [26]. We used a Gaussian kernel with $\sigma^2 = 1$. For a filter of length $L = 4$, we aim at estimating the time-invariant nonlinear function given by

$$f(\mathbf{r}_{k,n}) = \sqrt{r^2_{k,n,1} + \sin^2(r_{k,n,4}\pi)} + (0.8 - 0.5\exp(-r^2_{k,n,2})r_{k,n,3}. \tag{50}$$

The network-level instantaneous MSE, given by $\text{MSE}_n = \frac{1}{K}\sum_{k=1}^{K} e^2_{k,n}$, is considered as the performance metric and results are displayed by plotting MSE_n versus the iteration index n, averaging over 1000 independent runs.

In Fig. 1a, we present the learning curves of the centralized approaches based on CC and RFF. We limit the size of the dictionary and set the dimension of the RFF space to $D = 256$. Results show that, for large enough dictionary sizes and RFF-space dimensions, these implementations are able to reach similar performance to that of the GKLMS implementation without sparsification methods. In Fig 1b, we show similar results comparing the CC- and RFF-based GDKLMS against the GKLMS without sparsification. For the CC-based GDKLMS, we pre-train the dictionary before the learning process. The centralized implementations can better approximate the GKLMS without sparsification when compared to the GDKLMS. This is an expected result considering that data from the entire graph is available during the learning process of the centralized approaches.

In Fig. 2a we compare the proposed algorithms when smaller dictionaries and RFF-space dimensions are considered. Specifically, we compare the implementations based on RFF

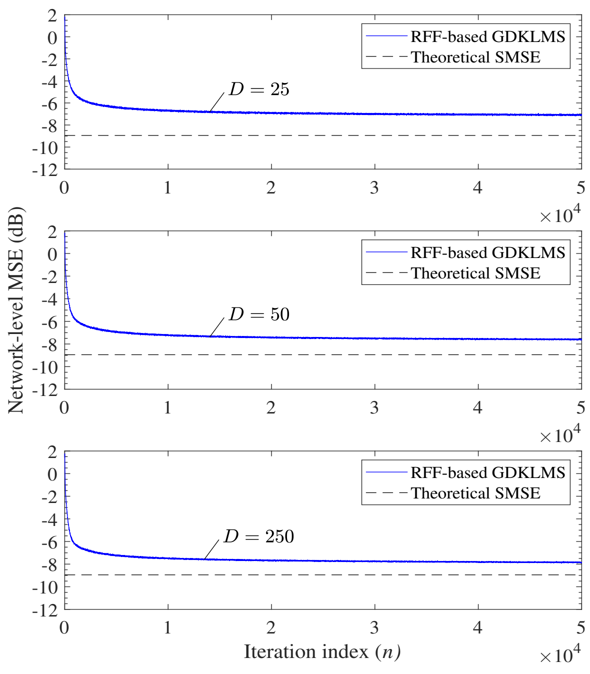
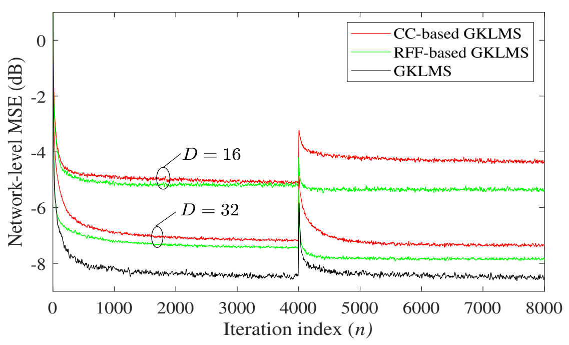
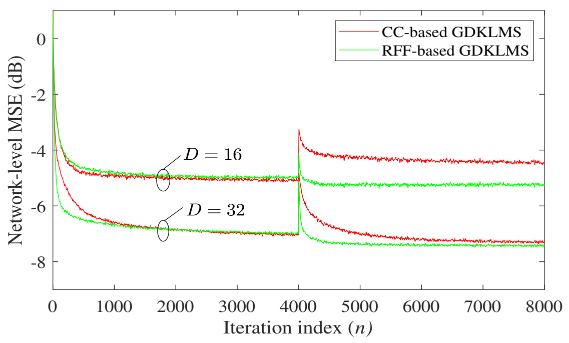


Fig. 3. Learning curves for the RFF-based GDKLMS with different values of D and the theoretical steady-state MSE values.



(a) Centralized solutions.



(b) Distributed solutions.

Fig. 4. Tracking performance of the proposed algorithms.

and coherence check against each other. For this purpose, we adjust the step-size μ and assess the convergence speed as the learning curves for both implementations achieve similar values of network-level steady-state MSE. Again, the value $D \in \{16, 32\}$ represents both the dimension of the RFF space and the size of the pre-trained dictionary for the coherence check approach. Results show that both CC- and RFF-based algorithms are capable of effectively representing the target function. Fig. 2a also shows that, for the same value of D and for similar values of network-level steady-state MSE, the RFF-based GKLMS converges faster than the CC-based one. Moreover, it can be observed that the performance of the implementations with fixed-size dictionaries greatly improves as D is increased from 16 to 32.

Fig. 2b shows the results for the distributed GDKLMS using CC and RFF. Similar to the centralized case, the plots show that both approaches can effectively represent the target function, achieving network-level MSE of approximately -5 dB for $D = 16$ and -7 dB for $D = 32$ for the noise scenario simulated. Again, the RFF-based solution exhibits faster convergence for both values of D when the network-level steady-state MSE is matched.

B. SMSE of the RFF-based GDKLMS

In this experiment, we observe the steady-state behavior of the proposed RFF-based GDKLMS. The network and data parameters employed in this simulation are the same used in Section VI-A. We run the RFF-based GDKLMS for a total of $T = 50000$ iterations, for different dimensions of the RFF space. In Fig. 3, we show the learning curves for $D \in \{25, 50, 250\}$ and the value of the SMSE computed using (49). The step-size is $\mu = 0.05$. Results show that increasing D reduces the gap between the numerical and theoretical results for the steady-state behavior of the algorithm. This observation is in line with the result presented in [47].

C. Tracking Performance of the Proposed Algorithms

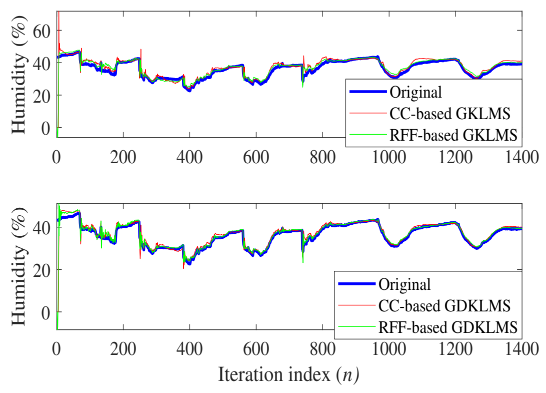
In this section we study the performance of the algorithms subjected to an abrupt change in the underlying model. The simulation setup is the same as in Section VI-A. The nonlinear function to be estimated is given by

$$f_n(\mathbf{r}_{k,n}) = \begin{cases} \sqrt{r_{k,n,1}^2 + r_{k,n,4}^2} - r_{k,n,3}e^{-r_{k,n,2}^2} & 0 < n \leq 4000 \\ \sqrt{r_{k,n,1}^2 + r_{k,n,2}^2 + r_{k,n,3}^2 + r_{k,n,4}^2} & 4000 < n. \end{cases} \quad (51)$$

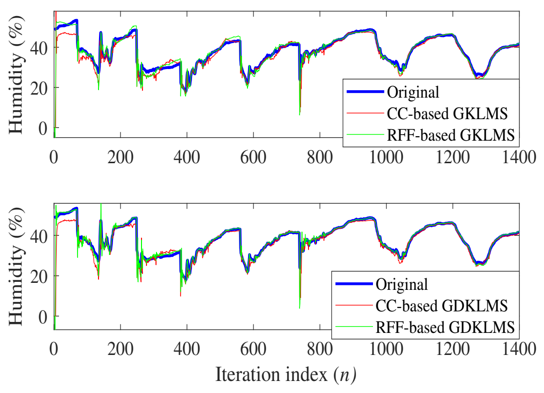
Fig. 4 shows the learning curves for the centralized and distributed algorithms for two values of dictionary sizes and RFF-space dimension, namely, $D = 16$ and $D = 32$. We see that the RFF-based implementations are resilient to model changes, while the CC-based implementations suffer from noticeable performance losses, especially for small dictionaries. This is an expected behavior, since larger dictionaries can represent more functions. We also see that the GKLMS achieves the lowest MSE, however, at the cost of an unconstrained dictionary size.

D. Laboratory-monitoring Data

We consider the Intel Lab database [68] that contains temperature and humidity data, measured during March 2004, from 52 sensors spread across a laboratory and its common areas. The undirected graph is constructed by connecting

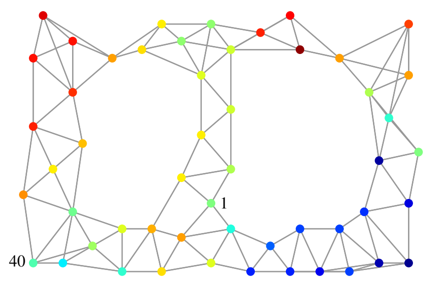


(a) Time series for Sensor 1.

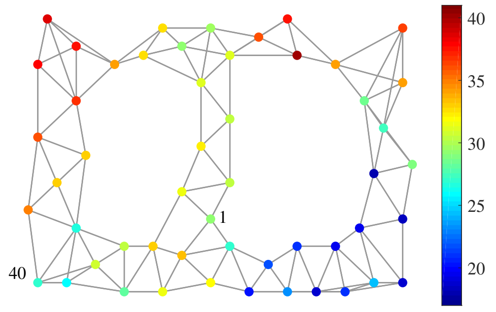


(b) Time series for Sensor 40.

Fig. 5. Time series of original and estimated humidity signals using the proposed algorithms for the Intel Lab dataset.



(a) Original.



(b) Estimated (RFF-based GDKLMS).

Fig. 6. Network structure for the Intel Lab simulation and snapshots of the original and estimated humidity signals.

each sensor to its four nearest neighbors. We consider the task of estimating humidity from the temperature signal. The data set comprises asynchronous sensor measurements, and we construct a snapshot of the graph signal by considering windows of 5 minutes from which we collect the first value available for each sensor. The model from temperature to humidity is expected to change with time. For instance, as workers arrive in the lab, the temperature and humidity are expected to change.

In our simulations, we used $L = 5$ and $D = 128$, for centralized and distributed implementations. The step sizes are 0.03 for CC- and RFF-based GKLMS, and 0.5 for GDKLMS implementations.

The humidity signals from Sensors 1 and 40 are plotted in Figs. 5a and 5b, respectively, together with the estimated signals from the graph filters. The variations in the plots are aligned with events that induce model changes. For example, the most notable peaks are aligned with the beginning and end of work shifts. The implementations based on CC and RFF have similar performances, while the latter exhibit slightly more resilience to changes in the model. Fig. 6 depicts the graph representation of the Intel Lab sensor network and presents snapshots of the humidity signals, both the original and the one estimated via RFF-based GDKLMS. These results confirm that the proposed algorithms can effectively estimate the humidity level from temperature readings.

VII. Conclusion

This paper introduced nonlinear adaptive graph filters for model estimation in the reproducing kernel Hilbert space. To this end, a centralized graph kernel LMS (GKLMS) algorithm was derived. To overcome the growing dimension problem encountered in the centralized GKLMS algorithm, coherence check based dictionary-sparsification and random Fourier features (RFF) were proposed. Furthermore, diffusion-based distributed implementations of both coherence check and RFF-based graph KLMS algorithms were developed that update filter parameters through local communications and in-network processing. Mean and mean-square-error convergence conditions were established for the proposed GDKLMS using RFF. Numerical simulations were conducted to demonstrate the performance of the proposed algorithms. Simulations confirmed that coherence check and RFF-based approaches effectively estimate nonlinear graph filters, while the latter exhibits a faster convergence and is robust to model changes.

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 Process. Mag.*, vol. 30, no. 3, pp. 83–98, May 2013.

[2] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs," *IEEE Trans. Signal Process.*, vol. 61, no. 7, pp. 1644–1656, Apr. 2013.

[3] A. Ortega, P. Frossard, J. Kovačević, J. M. F. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," *Proc. IEEE*, vol. 106, no. 5, pp. 808–828, May 2018.

[4] E. Ceci and S. Barbarossa, "Graph signal processing in the presence of topology uncertainties," *IEEE Trans. Signal Process.*, vol. 68, pp. 1558-1573, 2020.

[5] A. Sandryhaila and J. M. F. Moura, "Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure," *IEEE Signal Process. Mag.*, vol. 31, no. 5, pp. 80–90, Sep. 2014.

[6] I. Jablonski, "Graph signal processing in applications to sensor networks, smart grids, and smart cities," *IEEE Sensors J.*, vol. 17, no. 23, pp. 7659–7666, Dec. 2017.

[7] A. Gavili and X. Zhang, "On the shift operator, graph frequency, and optimal filtering in graph signal processing," *IEEE Trans. Signal Process.*, vol. 65, no. 23, pp. 6303–6318, Dec. 2017

[8] O. Teke, and P. Vaidyanathan, "Extending Classical Multirate Signal Processing Theory to Graphs - Part I: Fundamentals," *IEEE Trans. Signal Process.*, vol. 65, no. 2, pp. 409-422, Jan. 2017.

[9] S. K. Narang and A. Ortega, "Perfect Reconstruction Two-Channel Wavelet Filter Banks for Graph Structured Data," *IEEE Trans. Signal Process.*, vol. 60, no. 6, pp. 2786-2799, June 2012.

[10] S. Chen, R. Varma, A. Sandryhaila and J. Kovačević, "Discrete Signal Processing on Graphs: Sampling Theory," *IEEE Trans. Signal Process.*, vol. 63, no. 24, pp. 6510-6523, Dec. 2015.

[11] J. Mei and J. M. F. Moura, "Signal processing on graphs: Causal modeling of unstructured data," *IEEE Trans. Signal Process.*, vol. 65, no. 8, pp. 2077–2092, Apr. 2017.

[12] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs: Graph filters," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2013, pp. 6163–6166.

[13] E. Isufi, G. Leus and P. Banelli, "2-Dimensional finite impulse response graph-temporal filters," in *Proc. IEEE Global Conf. Signal Inf. Process.*, 2016, pp. 405–409.

[14] F. Hua, C. Richard, J. Chen, H. Wang, P. Borgnat and P. Gonçalves, "Learning Combination of Graph Filters for Graph Signal Modeling," *IEEE Signal Process. Lett.*, vol. 26, no. 12, pp. 1912-1916, Dec. 2019.

[15] A. Loukas, A. Simonetto, and G. Leus, "Distributed autoregressive moving average graph filters," *IEEE Signal Process. Lett.*, vol. 22, no. 11, pp. 1931–1935, Nov. 2015.

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

[17] X. Mao, K. Qiu, T. Li and Y. Gu, "Spatio-Temporal Signal Recovery Based on Low Rank and Differential Smoothness," *IEEE Trans. Signal Process.*, vol. 66, no. 23, pp. 6281-6296, Dec, 2018.

[18] K. Qiu, X. Mao, X. Shen, X. Wang, T. Li and Y. Gu, "Time-Varying Graph Signal Reconstruction," *IEEE J. Sel. Top. Signal Process.*, vol. 11, no. 6, pp. 870-883, Sep. 2017.

[19] M. J. M. Spelta and W. A. Martins, "Normalized LMS algorithm and data-selective strategies for adaptive graph signal estimation," *Signal Process.*, vol. 167, 107326, Feb. 2020.

[20] F. Hua, R. Nassif, C. Richard, H. Wang and A. H. Sayed, "Diffusion LMS With Communication Delays: Stability and Performance Analysis," *IEEE Signal Process. Lett.*, vol. 27, pp. 730-734, 2020.

[21] P. Di Lorenzo, P. Banelli, S. Barbarossa, and S. Sardellitti, "Distributed adaptive learning of graph signals," *IEEE Trans. Signal Process.*, vol. 65, no. 16, pp. 4193–4208, Aug. 2017.

[22] R. Nassif, C. Richard, J. Chen, and A. H. Sayed, "A graph diffusion LMS strategy for adaptive graph signal processing," in *Conf. Rec. Asilomar Conf. Signals Syst. Comput.*, pp. 1973–1976, Oct. 2017.

[23] R. Nassif, C. Richard, J. Chen, and A. H. Sayed, "Distributed diffusion adaptation over graph signals," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2018, pp. 4129–4133.

[24] F. Hua, R. Nassif, C. Richard, H. Wang and A. H. Sayed, "Online distributed learning over graphs with multitask graph-filter models," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 6, pp. 63–77, Jan. 2020.

[25] R. Nassif, S. Vlaski, C. Richard, J. Chen and A. H. Sayed, "Multitask Learning Over Graphs: An Approach for Distributed, Streaming Machine Learning," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 14-25, May 2020.

[26] A. H. Sayed, "Adaptation, learning, and optimization over networks," in *Foundations and Trends® in Mach. Learn.*, vol. 7, pp. 311–801, 2014.

[27] A. H. Sayed, "Diffusion Adaptation Over Networks," in *Acad. Press Libr. Signal Process.*, vol. 3, pp. 323–531, 2014.

[28] M. Paluš, "Detecting nonlinearity in multivariate time series," *Phys. Lett. A*, vol. 213, no. 3–4, pp. 138-147, Apr. 1996.

[29] J. F. Walker, N. Jenkins and N. Jenkins *Wind Energy Technology*. Wiley, June 1997.

[30] M. O. Franz and B. Schölkopf, "A unifying view of Wiener and Volterra theory and polynomial kernel regression," *Neural Comput.*, vol. 18, pp. 3097–3118, Dec. 2006.

[31] D. Comminiello and J. C. Príncipe. *Adaptive Learning Methods for Nonlinear System Modeling*. Elsevier, 2018.

[32] V. J. Mathews, "Adaptive polynomial filters," *IEEE Signal Process. Mag.*, vol. 8, no. 3, pp. 10–26, July 1991.

[33] Taiho Koh and E. Powers, "Second-order Volterra filtering and its application to nonlinear system identification," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 33, no. 6, pp. 1445–1455, Dec. 1985.

[34] D. J. Sebald and J. A. Bucklew, "Support vector machine techniques for nonlinear equalization," *IEEE Trans. Signal Process.*, vol. 48, no. 11, pp. 3217–3226, Nov. 2000.

[35] M. Scarpiniti, D. Comminiello, G. Scarano, R. Parisi and A. Uncini, "Steady-state performance of spline adaptive filters," *IEEE Trans. Signal Process.*, vol. 64, no. 4, pp. 816–828, Feb. 2016.

[36] A. J. Smola and R. Kondor, "Kernels and regularization on graphs," in *Learning Theory and Kernel Machines. Lecture Notes in Computer Science*, vol. 2777, pp. 144–158, Springer, 2003.

[37] W. Liu, J. C. Príncipe, and S. Haykin, *Kernel Adaptive Filtering*. Wiley, Feb. 2010.

[38] W. Gao, J. Chen, C. Richard and J. Huang, "Online Dictionary Learning for Kernel LMS," *IEEE Trans. Signal Process.*, vol. 62, no. 11, pp. 2765-2777, June, 2014.

[39] S. Theodoridis, K. Slavakis, and I. Yamada, "Adaptive learning in a world of projections," *IEEE Signal Process. Mag.*, vol. 28, no. 1, pp. 97–123, Jan. 2011.

[40] W. Gao, J. Chen, C. Richard, and J. Huang, "Diffusion adaptation over networks with kernel least-mean-square," in *Proc. IEEE Int. Workshop Comput. Adv. Multisens. Adapt. Process.*, 2015, pp. 217–220.

[41] S. Chouvardas and M. Draief, "A diffusion kernel LMS algorithm for nonlinear adaptive networks," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2016, pp. 4164–4168.

[42] D. Romero, M. Ma, and G. B. Giannakis, "Kernel-based reconstruction of graph signals," *IEEE Trans. Signal Process.*, vol. 65, no. 3, pp. 764–778, Feb. 2017.

[43] D. Romero, V. N. Ioannidis and G. B. Giannakis, "Kernel-based reconstruction of space-time functions on dynamic graphs," *IEEE J. Sel. Top. Signal Process.*, vol. 11, no. 6, pp. 856–869, Sep. 2017.

[44] V. N. Ioannidis, D. Romero and G. B. Giannakis, "Inference of Spatio-Temporal Functions Over Graphs via Multikernel Kriged Kalman Filtering," *IEEE Trans. Signal Process.*, vol. 66, no. 12, pp. 3228-3239, June 2018.

[45] B.-S. Shin, M. Yukawa, R. L. G. Cavalcante, and A. Dekorsy, "Distributed adaptive learning with multiple kernels in diffusion networks," *IEEE Trans. Signal Process.*, vol. 66, no. 21, pp. 5505–5519, Nov. 2018.

[46] A. Venkitaraman, S. Chatterjee and P. Händel, "Predicting graph signals using kernel regression where the input signal is agnostic to a graph," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 5, no. 4, pp. 698–710, Dec. 2019.

[47] P. Bouboulis, S. Chouvardas, and S. Theodoridis, "Online distributed learning over networks in RKH spaces using random Fourier features," *IEEE Trans. Signal Process.*, vol. 66, no. 7, pp. 1920–1932, 2018.

[48] Y. Shen, G. Leus, and G. B. Giannakis, "Online graph-adaptive learning with scalability and privacy," *IEEE Trans. Signal Process.*, vol. 67, no. 9, pp. 2471–2483, May 2019.

[49] S. Wang, L. Dang, B. Chen, S. Duan, L. Wang, and C. K. Tse, "Random Fourier filters under maximum correntropy criterion," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 10, pp. 3390–3403, Oct. 2018.

[50] A. Singh, N. Ahuja, and P. Moulin, "Online learning with kernels: Overcoming the growing sum problem," in *Proc. IEEE Int. Workshop Mach. Learn. Signal Process.*, 2012, pp. 1–6.

[51] K. Chen, S. Werner, A. Kuh, and Y.-F. Huang, "Nonlinear adaptive filtering with kernel set-membership approach," *IEEE Trans. Signal Process.*, vol. 68, pp. 1515-1528, 2020.

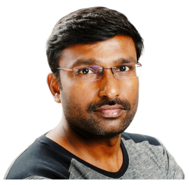
[52] C. Richard, J. Bermudez, and P. Honeine, "Online prediction of time series data with kernels," *IEEE Trans. Signal Process.*, vol. 57, no. 3, pp. 1058–1067, Mar. 2009.

[53] P. Bouboulis, S. Pougkakiotis, and S. Theodoridis, "Efficient KLMS and KRLS algorithms: A random Fourier feature perspective," in *Proc. IEEE Stat. Signal Process. Workshop*, 2016, pp. 1–5.

[54] N. Aronszajn, "Theory of reproducing kernels," *Trans. Amer. Math. Soc.*, vol. 68, May 1950.

[55] P. P. Pokharel, W. Liu and J. C. Príncipe, "Kernel least mean square algorithm with constrained growth", *Signal Process.*, vol. 89, no. 3, pp. 257–265 Mar. 2009.
[56] W. D. Parreira, J. C. M. Bermudez, C. Richard and J. Tourneret, "Stochastic behavior analysis of the Gaussian kernel least-mean-square algorithm," *IEEE Trans. Signal Process.*, vol. 60, no. 5, pp. 2208-2222, May 2012.
[57] S. Wang, Y. Zheng and C. Ling, "Regularized kernel least mean square algorithm with multiple-delay feedback," *IEEE Signal Process. Lett.*, vol. 23, no. 1, pp. 98-101, Jan. 2016.
[58] V. G. Gogineni, V. R. M. Elias, W. A. Martins and S. Werner, "Graph diffusion kernel LMS using random Fourier Features," in *Conf. Rec. Asilomar Conf. Signals Syst. Comput.*, pp. 1–5, Nov. 2020.
[59] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Proc. Adv. Neural Inf. Process. Syst.*, pp. 1177–1184, 2007.
[60] X. Dong, D. Thanou, P. Frossard and P. Vandergheynst, "Learning Laplacian matrix in smooth graph signal representations," *IEEE Trans. Signal Process.*, vol. 64, no. 23, pp. 6160-6173, Dec.1, 2016
[61] S. Segarra, A. G. Marques, G. Mateos and A. Ribeiro, "Network topology inference from spectral templates," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 3, no. 3, pp. 467-483, Sep. 2017.
[62] G. B. Giannakis, Y. Shen and G. V. Karanikolas, "Topology identification and learning over graphs: accounting for nonlinearities and dynamics," *Proc. IEEE*, vol. 106, no. 5, pp. 787-807, May 2018.
[63] G. Mateos, S. Segarra, A. G. Marques and A. Ribeiro, "Connecting the dots: identifying network structure via graph signal processing," *IEEE Signal Process. Mag.*, vol. 36, no. 3, pp. 16-43, May 2019.
[64] M. Moscu, R. Nassif, F. Hua and C. Richard, "Learning Causal Networks Topology From Streaming Graph Signals," *Proc. Eur. Signal Process. Conf.*, 2019, pp. 1-5.
[65] A. H. Sayed, S. Tu, J. Chen, X. Zhao, and Z. J. Towfic, "Diffusion strategies for adaptation and learning over networks: An examination of distributed strategies and network behavior," *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 155–171, May 2013.
[66] R. H. Koning, H. Neudecker, and T. Wansbeek, "Block Kronecker products and the vecb operator," *Linear Algebra and Its Applications*, vol. 149(C), 165–184, 1991.
[67] A. H. Sayed, "Adaptive networks," *Proc. IEEE*, vol. 102, pp. 460–497, Apr. 2014.
[68] "Intel Lab Data." http://db.csail.mit.edu/labdata/labdata.html. Accessed: 2019-11-28.

Vitor R. M. Elias received the B.Sc. degree in Electronics and Computer Engineering from the Federal University of Rio de Janeiro (UFRJ) in 2013. In 2015, he received the M.Sc. degree in Electrical Engineering from the COPPE/UFRJ. He is currently pursuing a joint Ph.D. degree at the UFRJ and the Norwegian University of Science and Technology (NTNU), where he was a visiting researcher from 2019 to 2020. He is currently with the Signals, Multimedia, and Telecommunications Lab (SMT/UFRJ) and the Signal Processing Group in the Department of Electronic Systems at NTNU. His experience and research interests include image and video compression, machine learning, adaptive learning, and digital signal processing on graphs.



Vinay Chakravarthi Gogineni received the Bachelor's degree in electronics and communication engineering from Jawaharlal Nehru Technological University, Andhra Pradesh, India, in 2005, the Master's degree in communication engineering from Vellore Institute of Technology, Vellore, India, in 2008, and the Ph.D. degree in electronics and electrical communication engineering from Indian Institute of Technology Kharagpur, India in 2019. From 2008 to 2011, he was with Cosyres Technologies, Bangalore, India, where he was involved in algorithmic-level optimization of audio codecs. From Aug. 2019 to Jul. 2020, he was an ERCIM postdoctoral research fellow at the department of electronic systems, NTNU, Norway. He is currently a postdoctoral research fellow at the machine intelligence department, SimulaMet, Simula research laboratory, Norway. His research interests include adaptive filtering and machine learning, graph signal processing, and geometric deep learning.

Wallace A. Martins (Senior Member, IEEE) received the electronics engineer degree and the M.Sc. and D.Sc. degrees in electrical engineering from the Federal University of Rio de Janeiro (UFRJ), Rio de Janeiro, Brazil, in 2007, 2009, and 2011, respectively. He was a Research Visitor with the University of Notre Dame, USA, in 2008, Université de Lille 1, France, in 2016, and Universidad de Alcalá, Spain, in 2018. Since 2013 he has been with the Department of Electronics and Computer Engineering (DEL/Poli) and Electrical Engineering Program (PEE/COPPE), UFRJ, where he is currently a tenured Associate Professor (on leave). He was Academic Coordinator and Deputy Department Chairman (DEL/Poli) in the period 2016-2017 with UFRJ. Before joining UFRJ, he worked as Associate Professor with CEFET/RJ, Brazil, from 2010 to 2013. He is currently a Research Associate working with the Interdisciplinary Centre for Security, Reliability and Trust (SnT), Université du Luxembourg. His research interests include the fields of digital signal processing, especially adaptive signal processing and graph signal processing, as well as telecommunications, with focus on equalization and precoding for wireless communications. He is a member (Associate Editor) of the Editorial Board for the IEEE Signal Processing Letters. He was the recipient of the Best Student Paper Award from EURASIP at EUSIPCO-2009, Glasgow, Scotland, and the 2011 Best Brazilian D.Sc. Dissertation Award from Capes.

Stefan Werner (SM'07) received the M.Sc. degree in electrical engineering from the Royal Institute of Technology, Stockholm, Sweden, in 1998 and the D.Sc. degree (Hons.) in electrical engineering from the Signal Processing Laboratory, Helsinki University of Technology, Espoo, Finland, in 2002. He is currently a Professor in the Department of Electronic Systems, Norwegian University of Science and Technology, Trondheim, Norway. He is also an Adjunct Professor with Aalto University in Finland, and an Adjunct Senior Research Fellow with the Institute for Telecommunications Research, University of South Australia. He was a visiting Melchor Professor with University of Notre Dame during summer 2019, and was holding an Academy Research Fellowship, funded by the Academy of Finland, from 2009 to 2014. His research interests include adaptive and statistical signal processing, signal processing for communications, and security and privacy in cyberphysical systems. He is a member of the editorial boards for the EURASIP Journal of Signal Processing and the IEEE Transactions on Signal and Information Processing over Networks.

Chapter 11

Publications on Kernel Regression over Graphs using RFF

P7 [2021] IEEE. Reprinted, with permission, from V. R. M. Elias, V. C. Gogineni, W. A. Martins, and S. Werner, “Kernel regression on graphs in random Fourier features space,” Accepted for publication in *International Conference on Acoustics, Speech, & Signal Processing*, pp. 1–5, 2021.

P8 [2021] IEEE. Reprinted, with permission, from V. R. M. Elias, V. C. Gogineni, W. A. Martins, and S. Werner, “Kernel regression over graphs using random Fourier features,” Submitted to *IEEE Transactions on Signal Processing*, pp. 1–12, 2021.

KERNEL REGRESSION ON GRAPHS IN RANDOM FOURIER FEATURES SPACE

Vitor R. M. Elias[⋆‡] *Vinay C. Gogineni*[§] *Wallace A. Martins*[†‡] *Stefan Werner*[⋆]

[⋆] Department of Electronic Systems, NTNU – Norwegian University of Science and Technology
[§]Department of Machine Intelligence, SimulaMet, Simula Research Laboratory
[†]Interdisciplinary Centre for Security, Reliability and Trust, UniLu – University of Luxembourg
[‡]Electrical Engineering Program, UFRJ – Federal University of Rio de Janeiro
Email: vitor.elias@ntnu.no, vcgogineni@simula.no, wallace.alvesmartins@uni.lu, stefan.werner@ntnu.no

ABSTRACT

This work proposes an efficient batch-based implementation for kernel regression on graphs (KRG) using random Fourier features (RFF) and a low-complexity online implementation. Kernel regression has proven to be an efficient learning tool in the graph signal processing framework. However, it suffers from poor scalability inherent to kernel methods. We employ RFF to overcome this issue and derive a batch-based KRG whose model size is independent of the training sample size. We then combine it with a stochastic gradient-descent approach to propose an online algorithm for KRG, namely the stochastic-gradient KRG (SGKRG). We also derive sufficient conditions for convergence in the mean sense of the online algorithms. We validate the performance of the proposed algorithms through numerical experiments using both synthesized and real data. Results show that the proposed batch-based implementation can match the performance of conventional KRG while having reduced complexity. Moreover, the online implementations effectively learn the target model and achieve competitive performance compared to the batch implementations.

Index Terms— kernel regression on graphs, online learning on graphs, random Fourier features, stochastic gradient.

1. INTRODUCTION

The connectivity of real-world elements and the amount of data generated in networks have increased over the last decades [1, 2]. Real networks and their corresponding data come in vastly different shapes and applications, ranging from genetic interaction networks [3] and the human brain [4] to sensor networks and smart cities [5]. Although an extensive range of classical digital signal processing (DSP) tools are available, they are not directly applicable to information processing of signals from networked structures. The graph signal processing (GSP) emerged in the last decade as a suitable framework for signal processing over networks, leveraging the network structure to process the networked data [4–7].

A major area of research in GSP is learning over graphs, which aims at discovering patterns in the data and graph structure to allow, e.g., prediction and reconstruction of graph signals [10–18]. Among the several approaches for learning over graphs, kernel regression has proved to be an innovative technique in several estimation and reconstruction applications for networked data [14–17]. In this work, we build upon the methodology of kernel regression on graphs (KRG) proposed in [16], which embeds a metric of smoothness over the graph in the optimization problem to improve the learning of regression parameters. However, the methodology in [16] suffers from scalability issues from kernel methods and is restricted to a batch-based offline approach. In this work, we first derive an efficient batch-based KRG using random Fourier features [19]. Then, we propose an online strategy for KRG using the stochastic gradient descent approach.

This paper is organized as follows. Section 2 presents the basic concepts of graph signal processing, and formulates the problem of kernel regression over graphs. The proposed batch-based KRG using RFF is presented in Section 3, and in Section 4, we present the online strategy for KRG, namely, the stochastic-gradient KRG (SGKRG). In Section 5, we derive sufficient conditions for the update step size to guarantee the proposed online algorithm's convergence. In Section 7, we present the final remarks of this work.

2. LEARNING OVER GRAPHS

2.1. Graph Signal Processing

A graph is denoted by $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{1, \ldots, K\}$ is the set of vertices, or nodes, and $\mathcal{E} = \{e_{11}, \ldots, e_{KK}\}$ is the set of edges. Elements $e_{ij} > 0$ indicate pairwise relations between nodes i and j according to a chosen metric, and an edge e_{ij} exists if and only if i and j are related [1,7]. In GSP, edges are typically represented in the adjacency matrix $\mathbf{A}$, such that the entry $A_{ij} = e_{ij}$ if e_{ij} exists and $A_{ij} = 0$ otherwise. In this work, we consider undirected graphs, such that $e_{ij} = e_{ji}$. The degree matrix $\mathbf{D}$ is a diagonal matrix such that $D_{jj} = \sum_{i\in\mathcal{N}_j} e_{ij}$ and $\mathcal{N}_j$ is the set of vertices that are adjacent to node j, referred to as neighborhood of j. The graph Laplacian is the positive semidefinite matrix $\mathbf{L} = \mathbf{D} - \mathbf{A}$ [2].

In GSP [2, 6], the signal on a graph is given by the mapping $s : \mathcal{V} \to \mathbb{R}$ and is represented by a vector $\mathbf{s}_n \in \mathbb{R}^K$. The graph signal represents a snapshot of the network state at time n. The graph Laplacian induces a variation metric for a graph signal $\mathbf{s}$ that depends on the graph structure [7]. The variation metric is given by

$$\nu(\mathbf{s}) = \mathbf{s}^{\mathrm{T}}\mathbf{L}\mathbf{s} = \sum_{i\neq j} A_{i,j}\big(s_i - s_j\big)^2, \tag{1}$$

where we can observe that the difference between two entries of the signal vector are penalized by the weight of the edge connecting the

The work of V. R. M. Elias was supported, in part, by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) — Grant number: 88887.310189/2018-00 and, in part, by CNPq. The work of Dr. Martins was supported, in part, by the ERC project AGNOSTIC and, in part, by FAPERJ. The work of Dr. Werner was supported, in part, by the Research Council of Norway.

two nodes. That is, a graph signal where values in adjacent nodes are different is associated with a large variation metric according to (1).

2.2. Learning Task

Consider a set of data pairs $\{\mathbf{x}_n, \mathbf{t}_n\}$, $n \in \{1, 2, \ldots, N\}$, such that vectors $\mathbf{x}_n$, called reference signals, are related to target signals $\mathbf{t}_n$ through an unknown function $f(\cdot)$ such that $\mathbf{t}_n = f(\mathbf{x}_n)$. The objective of typical learning strategies over graphs is to model $f(\cdot) : \mathbb{R}^K \to \mathbb{R}^K$ in the case where both $\mathbf{x}_n$ and $\mathbf{t}_n$ belong to the same graph $\mathcal{G}$. Previous research on learning over networks include, e.g., dictionary learning [11], linear [12, 13] and nonlinear graph filtering [18], kriged Kalman filtering [10], and kernel regression strategies [14–17].

In particular, the work in [16] proposes a kernel regression methodology that allows the reference signal to be agnostic to the graph. That is, the regression signal does not need to be a graph signal, whereas the target signal lies over $\mathcal{G}$, which widens the range of applications for the proposed method. For this, [16] assumes that graph signals are expected to be smooth with respect to the graph, inducing a low value of the variation metric (1). The model is then estimated in terms of a matrix $\mathbf{W}$ such that

$$\mathbf{y}_n = \mathbf{W}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}_n), \tag{2}$$

where $\mathbf{y}_n$ is an estimate of the target graph signal $\mathbf{t}_n$ and $\boldsymbol{\phi}(\cdot)$ is an unknown function of the input signal. The optimal parameter matrix $\mathbf{W}$ is found by minimizing the cost function

$$C(\mathbf{W}) = \sum_{n=1}^{N} \|\mathbf{t}_n - \mathbf{y}_n\|_2^2 + \alpha \mathrm{tr}(\mathbf{W}^{\mathrm{T}}\mathbf{W}) + \beta \sum_{n=1}^{N} \nu(\mathbf{y}_n), \tag{3}$$

where the last term on the right-hand side enforces that the model respects the smoothness of the target signal. The solution for (3) is obtained in [16] in closed form using the kernel method. This leads to a model whose dimension increases with the number of training samples. In the next sections, we propose a reduced-complexity solution for the batch-based solution of (3) using RFF, and we propose an online implementation for KRG.

3. BATCH-BASED KRG USING RANDOM FOURIER FEATURES

A way to overcome the scaling issues of kernel methods is provided by random Fourier features [19]. Using RFF, a shift-invariant kernel evaluation $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \kappa(\mathbf{x}_i - \mathbf{x}_j)$ is approximated as an inner product in the D-dimensional RFF space, where D is much lower than the number of training samples. The mapping of $\mathbf{x}_i$ into the RFF space $\mathbb{R}^D$ is given by

$$\mathbf{z}_i = (D/2)^{-\frac{1}{2}} \left[\cos(\mathbf{v}_1^{\mathrm{T}}\mathbf{x}_i + b_1) \ldots \cos(\mathbf{v}_D^{\mathrm{T}}\mathbf{x}_i + b_D)\right]^{\mathrm{T}}, \tag{4}$$

where the phase terms $\{b_i\}_{i=1}^{D}$ are drawn from a uniform distribution on the interval $[0,\, 2\pi]$, and vectors $\{\mathbf{v}_i\}_{i=1}^{D}$ are drawn from the probability density function (pdf) $p(\mathbf{v})$, which corresponds to the Fourier transform of $k(\mathbf{x}_i - \mathbf{x}_j)$ [14, 19]. To derive the KRG model in the RFF-space, consider the kth entry of the estimate $\mathbf{y}$ as $y_k = \mathbf{w}_k^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x})$ where $\mathbf{w}_k$ denotes the kth column of the parameter matrix $\mathbf{W}$. Using the substitution $\mathbf{W} = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Psi}$, and the kernel trick $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\phi}(\mathbf{x}_i)^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_j)$, we can write

$$y_k = \left(\sum_{n=1}^{N} \Psi_{n,k} \boldsymbol{\phi}(\mathbf{x}_n)\right)^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}) = \left(\sum_{n=1}^{N} \Psi_{n,k} \kappa(\mathbf{x}_n, \mathbf{x})\right). \tag{5}$$

Using RFF, (5) can be approximated by

$$y_k \approx \sum_{n=1}^{N} \Psi_{n,k} \mathbf{z}_n^{\mathrm{T}} \mathbf{z} = \mathbf{h}_k^{\mathrm{T}} \mathbf{z}. \tag{6}$$

Finally, the RFF-based regression for the entire graph signal is written as

$$\mathbf{y} = \mathbf{H}^{\mathrm{T}} \mathbf{z}, \tag{7}$$

where $\mathbf{H} = [\mathbf{h}_1\ \mathbf{h}_2\ \ldots\ \mathbf{h}_K] \in \mathbb{R}^{D \times K}$ is the representation of the regression coefficient matrix in the RFF space. Now, the cost function (3) can be rewritten for the optimization in terms of $\mathbf{H}$ as

$$C(\mathbf{H}) = \sum_{n=1}^{N} \|\mathbf{t}_n - \mathbf{y}_n\|_2^2 + \alpha \mathrm{tr}(\mathbf{H}^{\mathrm{T}}\mathbf{H}) + \beta \sum_{n=1}^{N} \nu(\mathbf{y}_n). \tag{8}$$

Letting the matrix $\mathbf{Z} = [\mathbf{z}_1\ \mathbf{z}_2\ \ldots\ \mathbf{z}_N]^{\mathrm{T}} \in \mathbb{R}^{N \times D}$ represent the RFF mapping of all training input vectors $\{\mathbf{x}_n\}_{n=1}^{N}$, the cost function (8) can be rewritten as

$$\begin{aligned} C(\mathbf{H}) = \sum_{n=1}^{N} \|\mathbf{t}_n\|_2^2 - 2\mathrm{tr}(\mathbf{T}^{\mathrm{T}}\mathbf{Z}\mathbf{H}) + \mathrm{tr}(\mathbf{H}^{\mathrm{T}}\mathbf{Z}^{\mathrm{T}}\mathbf{Z}\mathbf{H}) \\ + \alpha(\mathbf{H}^{\mathrm{T}}\mathbf{H}) + \beta \mathrm{tr}(\mathbf{H}^{\mathrm{T}}\mathbf{Z}^{\mathrm{T}}\mathbf{Z}\mathbf{H}\mathbf{L}), \end{aligned} \tag{9}$$

where $\mathbf{T} = [\mathbf{t}_1\ \mathbf{t}_2\ \ldots\ \mathbf{t}_N]^{\mathrm{T}} \in \mathbb{R}^{N \times K}$. The gradient of $C(\mathbf{H})$ with respect to $\mathbf{H}$ is given by

$$\nabla C(\mathbf{H}) = -2\mathbf{Z}^{\mathrm{T}}\mathbf{T} + 2\mathbf{Z}^{\mathrm{T}}\mathbf{Z}\mathbf{H} + 2\alpha\mathbf{H} + 2\beta\mathbf{Z}^{\mathrm{T}}\mathbf{Z}\mathbf{H}\mathbf{L}. \tag{10}$$

By making $\nabla C(\mathbf{H}) = \mathbf{0}$, we obtain

$$(\mathbf{Z}^{\mathrm{T}}\mathbf{Z} + \alpha\mathbf{I}_D)\mathbf{H} + \beta\mathbf{Z}^{\mathrm{T}}\mathbf{Z}\mathbf{H}\mathbf{L} = \mathbf{Z}^{\mathrm{T}}\mathbf{T}. \tag{11}$$

Then, vectorizing both sides of (11) and using the relation $\mathrm{vec}(\mathbf{AXB}) = (\mathbf{B}^{\mathrm{T}} \otimes \mathbf{A})\mathrm{vec}(\mathbf{X})$, where $\mathrm{vec}(\cdot)$ denotes the column-stacking operator and $\otimes$ denotes the Kronecker-product operator, the optimum regression coefficients in the RFF space can be obtained as

$$\mathrm{vec}(\mathbf{H}_{\mathrm{o}}) = (\mathbf{B}_{\mathrm{RFF}} + \mathbf{C}_{\mathrm{RFF}})^{-1}\mathrm{vec}(\mathbf{Z}^{\mathrm{T}}\mathbf{T}), \tag{12}$$

where $\mathbf{B}_{\mathrm{RFF}} = (\mathbf{I}_K \otimes (\mathbf{Z}^{\mathrm{T}}\mathbf{Z} + \alpha\mathbf{I}_D))$ and $\mathbf{C}_{\mathrm{RFF}} = (\beta\mathbf{L} \otimes \mathbf{Z}^{\mathrm{T}}\mathbf{Z})$.

Once the regression coefficients are trained, the target estimate $\mathbf{y}$ given an input signal $\mathbf{x}$ corresponding to $\mathbf{z}$ in the RFF space is given by

$$\mathbf{y} = \mathbf{H}_{\mathrm{o}}^{\mathrm{T}} \mathbf{z}.$$

It can be seen that the regression does not depend on the number of training samples and the model has a fixed size D. Therefore, the proposed RFF-approach offers an efficient batch-based KRG for large datasets. Note that the batch-based implementation requires that all samples are available.

4. ONLINE KERNEL REGRESSION ON GRAPHS

We now derive online implementations for KRG using stochastic-gradient descent approaches. These implementations avoid the delay inherent to the batch-based implementation by updating the regression parameters for each new sample. Additionally, each update has a small computational cost when compared to the batch computation.

4.1. Stochastic Gradient Descent KRG

We propose to update the parameters of the KRG in an iterative manner using a stochastic approximation of the gradient $\nabla C(\mathbf{H})$, and derive the stochastic-gradient KRG (SGKRG) algorithm. Consider the instantaneous version, at time n, of the RFF-based KRG model (7), given by

$$\mathbf{y}_n = \mathbf{H}_n^{\mathrm{T}} \mathbf{z}_n, \tag{13}$$

Then, we can write the corresponding instantaneous cost function

$$C(\mathbf{H}_n) = \|\mathbf{t}_n - \mathbf{y}_n\|_2^2 + \alpha \mathrm{tr}(\mathbf{H}_n^{\mathrm{T}}\mathbf{H}_n) + \beta\nu(\mathbf{y}_n). \tag{14}$$

The gradient of $C(\mathbf{H}_n)$ with respect to $\mathbf{H}_n$, which corresponds to a stochastic approximation of $\nabla C(\mathbf{H})$ in (10), is given by

$$\begin{aligned}\nabla C(\mathbf{H}_n) &= -2\mathbf{z}_n\mathbf{t}_n^{\mathrm{T}} + 2\mathbf{z}_n\mathbf{z}_n^{\mathrm{T}}\mathbf{H}_n + 2\alpha\mathbf{H}_n + 2\beta\mathbf{z}_n\mathbf{z}_n^{\mathrm{T}}\mathbf{H}_n\mathbf{L} \\ &= -2\left(\mathbf{z}_n(\mathbf{e}_n^{\mathrm{T}} - \beta\mathbf{y}_n^{\mathrm{T}}\mathbf{L}) - \alpha\mathbf{H}_n\right),\end{aligned} \tag{15}$$

where $\mathbf{e}_n = \mathbf{t}_n - \mathbf{y}_n$ is the network-level error at iteration n. Given (15), in order to recursively minimize $C(\mathbf{H}_n)$, the parameter matrix $\mathbf{H}_n$ is updated by taking a step in the negative-gradient direction as

$$\mathbf{H}_{n+1} = \mathbf{H}_n + \mu\left(\mathbf{z}_n(\mathbf{e}_n^{\mathrm{T}} - \beta\mathbf{y}_n^{\mathrm{T}}\mathbf{L}) - \alpha\mathbf{H}_n\right), \tag{16}$$

where $\mu > 0$ is the step size. Equation (16) represents the update equation for the proposed online SGKRG algorithm using RFF. We note the gradient obtained using all samples is expected to offer a better optimization direction than the one obtained using a single sample. Thus, the SGKRG and the batch-based KRG have opposite characteristics in the trade-off between complexity and convergence speed.

5. CONVERGENCE ANALYSIS

In this section, we analyze the convergence of the proposed online algorithm in the mean sense [20–22]. We assume the observation model $\mathbf{t}_n = \mathbf{H}_{\mathrm{o}}^{\mathrm{T}}\mathbf{z}_n + \boldsymbol{\upsilon}_n$, where $\boldsymbol{\upsilon}_n$ is observation noise vector. The noise is assumed to be zero-mean and independent of $\mathbf{z}_n$. Let $\boldsymbol{\Xi}_n = \mathbf{H}_n - \mathbf{H}_{\mathrm{o}}$ denote the deviation between the regression parameters $\mathbf{H}_n$ and the optimal parameters $\mathbf{H}_{\mathrm{o}}$ at iteration n. We have

$$\begin{aligned}\boldsymbol{\Xi}_{n+1} &= \mathbf{H}_n + \mu\left(\mathbf{z}_n(\mathbf{t}_n^{\mathrm{T}} - \mathbf{z}_n^{\mathrm{T}}\mathbf{H}_n - \beta\mathbf{z}_n^{\mathrm{T}}\mathbf{H}_n\mathbf{L}) - \alpha\mathbf{H}_n\right) - \mathbf{H}_{\mathrm{o}} \\ &= \boldsymbol{\Xi}_n + \mu\left(\mathbf{z}_n(\mathbf{t}_n^{\mathrm{T}} - \mathbf{z}_n^{\mathrm{T}}\mathbf{H}_n) - \beta\mathbf{z}_n\mathbf{z}_n^{\mathrm{T}}\mathbf{H}_n\mathbf{L} - \alpha\mathbf{H}_n\right),\end{aligned} \tag{17}$$

which can be rewritten as

$$\begin{aligned}&\boldsymbol{\Xi}_{n+1} \\ &\quad = \boldsymbol{\Xi}_n + \mu(\mathbf{z}_n\boldsymbol{\upsilon}_n^{\mathrm{T}} + \mathbf{z}_n\mathbf{z}_n^{\mathrm{T}}(\mathbf{H}_{\mathrm{o}} - \mathbf{H}_n) - \beta\mathbf{z}_n\mathbf{z}_n^{\mathrm{T}}\mathbf{H}_n\mathbf{L} - \alpha\mathbf{H}_n) \\ &\quad = (\mathbf{I}_D - \mu\mathbf{z}_n\mathbf{z}_n^{\mathrm{T}})\boldsymbol{\Xi}_n + \mu\mathbf{z}_n\boldsymbol{\upsilon}_n^{\mathrm{T}} - \mu(\beta\mathbf{z}_n\mathbf{z}_n^{\mathrm{T}}\mathbf{H}_n\mathbf{L} + \alpha\mathbf{H}_n).\end{aligned} \tag{18}$$

Vectorizing both sides of (18), we obtain

$$\begin{aligned}\boldsymbol{\xi}_{n+1} = &\left(\mathbf{I}_K \otimes (\mathbf{I}_D - \mu\mathbf{z}_n\mathbf{z}_n^{\mathrm{T}})\right)\boldsymbol{\xi}_n + \mu(\mathbf{I}_K \otimes \mathbf{z}_n)\boldsymbol{\upsilon}_n \\ &- \mu\left(\alpha\mathbf{I}_{KD} + (\beta\mathbf{L} \otimes \mathbf{z}_n\mathbf{z}_n^{\mathrm{T}})\right)\mathrm{vec}(\mathbf{H}_n),\end{aligned} \tag{19}$$

where $\boldsymbol{\xi}_n = \mathrm{vec}(\boldsymbol{\Xi}_n)$. Substituting $\mathrm{vec}(\mathbf{H}_n) = \boldsymbol{\xi}_n + \mathrm{vec}(\mathbf{H}_{\mathrm{o}})$ into (19), it can be rewritten as

$$\begin{aligned}\boldsymbol{\xi}_{n+1} = &\left[\mathbf{I}_{KD} - \mu\left(\alpha\mathbf{I}_{KD} + (\mathbf{I}_K + \beta\mathbf{L}) \otimes (\mathbf{z}_n\mathbf{z}_n^{\mathrm{T}})\right)\right]\boldsymbol{\xi}_n \\ &- \mu\left(\alpha\mathbf{I}_{KD} + \beta\mathbf{L} \otimes (\mathbf{z}_n\mathbf{z}_n^{\mathrm{T}})\right)\mathrm{vec}(\mathbf{H}_{\mathrm{o}}) \\ &+ \mu(\mathbf{I}_K \otimes \mathbf{z}_n)\boldsymbol{\upsilon}_n.\end{aligned} \tag{20}$$

We now take the expected value on both sides of (20). Given the zero-mean and independence assumptions on the observation noise, the last term's expected value on the right-hand side of (20) is zero. We obtain the following recursion on $\mathrm{E}[\boldsymbol{\xi}_n]$

$$\mathrm{E}[\boldsymbol{\xi}_{n+1}] = \boldsymbol{\mathcal{A}}\mathrm{E}[\boldsymbol{\xi}_n] - \boldsymbol{\mathcal{B}}\mathrm{vec}(\mathbf{H}_{\mathrm{o}}), \tag{21}$$

where

$$\begin{aligned}\boldsymbol{\mathcal{A}} &= \mathbf{I}_{KD} - \mu\left(\alpha\mathbf{I}_{KD} + (\mathbf{I}_K + \beta\mathbf{L}) \otimes \mathbf{R}_{\mathbf{z}}\right) \\ \boldsymbol{\mathcal{B}} &= \mu\left(\alpha\mathbf{I}_{KD} + \beta\mathbf{L} \otimes \mathbf{R}_{\mathbf{z}}\right),\end{aligned} \tag{22}$$

with $\mathbf{R}_{\mathbf{z}} = \mathrm{E}[\mathbf{z}_n\mathbf{z}_n^{\mathrm{T}}]$. Taking the recursion (21) down to zero, we obtain

$$\mathrm{E}[\boldsymbol{\xi}_n] = \boldsymbol{\mathcal{A}}^n\mathrm{E}[\boldsymbol{\xi}_0] - \mu\sum_{i=0}^{n-1}\boldsymbol{\mathcal{A}}^{n-1-i}\boldsymbol{\mathcal{B}}\mathrm{vec}(\mathbf{H}_{\mathrm{o}}). \tag{23}$$

From (23), we see that convergence is guaranteed if $\lim_{n\to\infty}\boldsymbol{\mathcal{A}}^n = 0$, which is achieved when $\rho(\boldsymbol{\mathcal{A}}) < 1$, where $\rho(\cdot)$ denotes the spectral radius of the argument, i.e., its largest absolute eigenvalue. We have that $\rho(\boldsymbol{\mathcal{A}}) < 1$ if $\rho\left(\mu\left(\alpha\mathbf{I}_{KD} + (\mathbf{I}_K + \beta\mathbf{L}) \otimes \mathbf{R}_{\mathbf{z}}\right)\right) < 2$. Therefore, a sufficient condition for the convergence of the proposed SGKRG algorithms is given by

$$0 < \mu < \frac{2}{\rho(\mathbf{R}_{\mathbf{z}}) + \alpha + \beta\rho(\mathbf{L})\rho(\mathbf{R}_{\mathbf{z}})}. \tag{24}$$

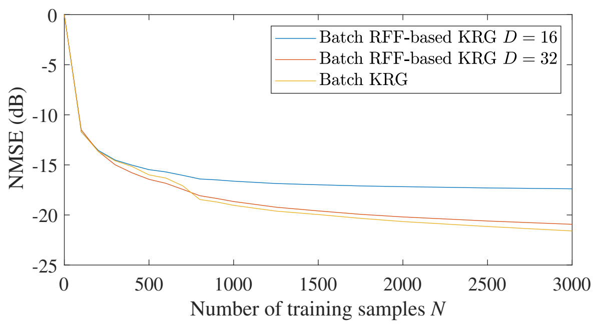
Under the convergence condition (24), (23) converges asymptotically to $(\mathbf{I}_{KD} - \boldsymbol{\mathcal{A}})^{-1}\boldsymbol{\mathcal{B}}\mathrm{vec}(\mathbf{H}_{\mathrm{o}})$, which reduces to $(\alpha\mathbf{I}_{KD} + (\mathbf{I}_K + \beta\mathbf{L}) \otimes \mathbf{R}_z)^{-1}(\alpha\mathbf{I}_{KD} + \beta\mathbf{L} \otimes \mathbf{R}_z)\mathrm{vec}(\mathbf{H}_{\mathrm{o}})$. This means that the solution of the SGKRG is asymptotically biased in the mean sense. The bias is introduced by the regularization coefficients α and β.

6. NUMERICAL EXPERIMENTS

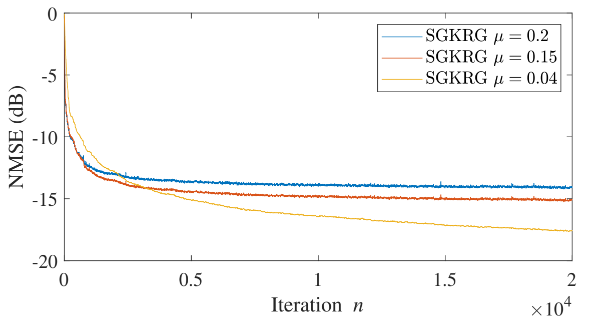
In this section we validate the performance of the proposed algorithms. We reproduce two regression problems adopted in [16] and compare the proposed algorithms against the conventional KRG. In both experiments, we use the Gaussian kernel $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2/(2\sigma^2)\right)$ when employing the kernel methods and the RFF implementations.

6.1. Synthesized Data

The first experiment uses an Erdös Rényi graph with artificially generated data. The graph has $K = 50$ nodes with edge-probability equal to 0.1. A total of $S = 20000$ K-dimensional data samples are generated as follows. First, a covariance matrix $\mathbf{C}_S \in \mathbb{R}^{S\times S}$ is drawn from the inverse Wishart distribution with an identity hyperparameter matrix and S degrees of freedom. Then, K independent vector realizations are drawn from an S-dimensional Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{C}_S)$. Letting $\{\mathbf{x}_n\}_{n=1}^S$ denote the obtained signals, the corresponding target vectors $\{\mathbf{t}_n\}_{n=1}^S$ are generated by projecting each signal onto the graph by solving $\mathbf{t}_n = \arg\min_{\boldsymbol{\tau}}\left\{\|\mathbf{x}_n - \boldsymbol{\tau}\|_2^2 + \boldsymbol{\tau}^{\mathrm{T}}\mathbf{L}\boldsymbol{\tau}\right\}$.



(a) Batch-based solutions.



(b) Online solution.

Fig. 1. NMSE achieved by the KRG implementations versus number of training samples.

From the total $S = 20000$ samples, we use 1000 samples as test dataset and up to 19000 samples as training dataset. Target data in the training dataset are corrupted with additive white Gaussian noise (AWGN) and the signal-to-noise ratio is 5 dB. The parameters α and β are estimated via 5-fold cross-validation with grid-search using a separate dataset, and minimizing the normalized squared estimation error

$$\text{NMSE} = 10 \log_{10} \left(\text{E} \left[\frac{\|\mathbf{Y} - \mathbf{T}_0\|_\text{F}^2}{\|\mathbf{T}_0\|_\text{F}^2} \right] \right), \tag{25}$$

where $\mathbf{T}$ is the true target matrix, $\mathbf{Y}$ is the estimated matrix. We select the parameters that result in the best NMSE at the end of the learning process.

The proposed batch-based algorithm is compared against the conventional KRG from [16] and results, averaged over 500 independent runs, are presented in Fig. 1a. The batch-based algorithms are trained with up to 3000 samples, which meets the limit of our computational power when running the conventional KRG. Plots show that the RFF-based approach approximates well the conventional KRG. In Fig. 1b, results show that the online algorithm is capable of learning the regression parameters. We run the SGKRG up to 20000 samples, such that the test samples are included at the end of the initial training samples. Note that the value of μ affects the convergence of the proposed stochastic-gradient-based implementation. We demonstrate the performance for different step sizes and we show that the NMSE level achieved by the SGKRG approximates that of the batch RFF-based KRG as μ decreases.

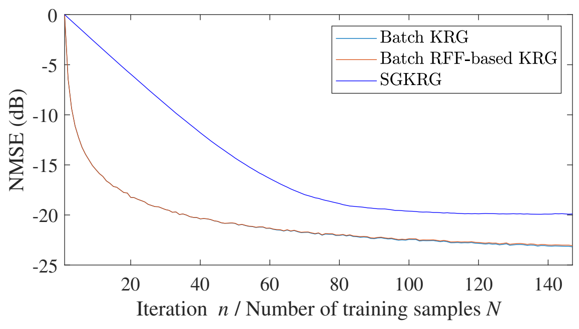


Fig. 2. NMSE achieved by the KRG implementations versus number of training samples for the fMRI signal simulation.

6.2. Real Data

The second experiment uses real data and addresses the task of estimating brain activity of voxels in a functional magnetic resonance imaging (fMRI) dataset. The data and graph used in [16] are made available in [23] and the same are used in this experiment.

A voxel is a volumetric unit that constitutes a 3-dimensional image of the brain, and each voxel is associated with a small cubic portion of the brain. Regions of the brain relate to each other anatomically and, by considering these relations, a graph can be constructed where voxels are the nodes, and edges represent relations between them. More details on this dataset and graph construction are provided in [16]. The regression experiment consists of estimating the signal on 90 of the voxels using the signal from 10 other voxels, such that the graph structure corresponds to the set of pairwise relations of the 90 voxels. Training and test datasets have the same size equal to 146 input-target pairs. The training signal is corrupted by an AWGN with covariance matrix $0.1 \cdot \mathbf{I}_K$.

We conduct 100 independent runs with different permutations of signals between training and test datasets. The RFF-space dimension is $D = 32$. Results are shown in Fig. 2 for both batch and online algorithms. In this experiment, both batch-based implementations converge together to approximately -23 dB and it can be observed that the RFF-based implementation matches the conventional KRG. Results show that the SG-based implementation is capable of successfully learning the target model and achieving low NMSE, around -20 dB, using $\mu = 0.04$.

7. CONCLUSION

This paper proposed batch-based and online implementations for kernel regression on graphs. The performance of the proposed algorithms was validated with numerical experiments using synthesized and real data. The proposed batch-based implementation uses RFF to approximate kernel evaluations as inner-products in a fixed-dimension space, reducing the complexity of the regression model compared to the conventional KRG. Results of numerical experiments showed no significant performance loss in approximating the kernel evaluations using RFF, such that the RFF-based implementation can achieve the same performance as the conventional KRG. Additionally, sufficient conditions for convergence of this algorithm in the mean sense were derived.

8. REFERENCES

[1] A. Sandryhaila and J. M. F. Moura, “Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure,” *IEEE Signal Process. Mag.*, vol. 31, pp. 80–90, Sep. 2014.

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

[3] B. Boucher and S. Jenna, “Genetic interaction networks: better understand to better predict,” *Front. in Genet.*, vol. 4, pp. 1–16, Dec. 2013.

[4] W. Huang, T. A. W. Bolton, J. D. Medaglia, D. S. Bassett, A. Ribeiro, and D. Van De Ville, “A graph signal processing perspective on functional brain imaging,” *Proc. IEEE*, vol. 106, pp. 868–885, May 2018.

[5] I. Jablonski, “Graph signal processing in applications to sensor networks, smart grids, and smart cities,” *IEEE Sensors J.*, vol. 17, pp. 7659–7666, Dec. 2017.

[6] A. Sandryhaila and J. M. F. Moura, “Discrete signal processing on graphs,” *IEEE Trans. Signal Process.*, vol. 61, pp. 1644–1656, Apr. 2013.

[7] A. Ortega, P. Frossard, J. Kovacevic, J. M. F. Moura, and P. Vandergheynst, “Graph signal processing: Overview, challenges, and applications,” *Proc. IEEE*, vol. 106, pp. 808–828, May 2018.

[8] A. Sandryhaila and J. M. F. Moura, “Discrete signal processing on graphs: Graph filters,” in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, May 2013, pp. 6163–6166.

[9] A. Gavili and X.-P. Zhang, “On the shift operator, graph frequency, and optimal filtering in graph signal processing,” *IEEE Trans. Signal Process.*, vol. 65, pp. 6303–6318, Dec. 2017.

[10] V. N. Ioannidis, D. Romero and G. B. Giannakis, ”Inference of Spatio-Temporal Functions Over Graphs via Multikernel Kriged Kalman Filtering,” *IEEE Trans. Signal Process.*, vol. 66, no. 12, pp. 3228-3239, June 2018.

[11] D. Thanou, D. I. Shuman and P. Frossard, “Learning Parametric Dictionaries for Signals on Graphs,” *IEEE Trans. Signal Process.*, vol. 62, no. 15, pp. 3849-3862, Aug 2014.

[12] R. Nassif, C. Richard, J. Chen, and A. H. Sayed, “Distributed diffusion adaptation over graph signals,” in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2018, pp. 4129–4133.

[13] F. Hua, R. Nassif, C. Richard, H. Wang and A. H. Sayed, “Online distributed learning over graphs with multitask graph-filter models,” *IEEE Trans. Signal Inf. Process. Netw.*, vol. 6, pp. 63–77, Jan. 2020.

[14] D. Romero, M. Ma, and G. B. Giannakis, “Kernel-based reconstruction of graph signals,” *IEEE Trans. Signal Process.*, vol. 65, no. 3, pp. 764–778, Feb. 2017.

[15] D. Romero, V. N. Ioannidis and G. B. Giannakis, ”Kernel-based reconstruction of space-time functions on dynamic graphs,” *IEEE J. Sel. Top. Signal Process.*, vol. 11, no. 6, pp. 856–869, Sep. 2017.

[16] A. Venkitaraman, S. Chatterjee and P. Händel, “Predicting graph signals using kernel regression where the input signal is agnostic to a graph,” *IEEE Trans. Signal Inf. Process. Netw.*, vol. 5, no. 4, pp. 698–710, Dec. 2019.

[17] Y. Shen, G. Leus, and G. B. Giannakis, “Online graph-adaptive learning with scalability and privacy,” *IEEE Trans. Signal Process.*, vol. 67, pp. 2471–2483, May 2019.

[18] V. G. Gogineni, V. R. M. Elias, W. A. Martins and S. Werner, “Graph diffusion kernel LMS using random Fourier Features,” in *Conf. Rec. Asilomar Conf. Signals Syst. Comput.*, pp. 1–5, Nov. 2020.

[19] A. Rahimi and B. Recht, “Random features for large-scale kernel machines,” in *Proc. Adv. Neural Inf. Process. Syst.*, pp. 1177–1184, 2007.

[20] W. A. Gardner, “Learning characteristics of stochastic-gradient-descent algorithms: A general study, analysis, and critique,” *Signal Process.*, vol 6, no. 2, pp. 113–133, 1984.

[21] A. H. Sayed, *Adaptive Filters*. Wiley, Jan. 2008.

[22] P. Di Lorenzo, S. Barbarossa, P. Banelli and S. Sardellitti, “Adaptive Least Mean Squares Estimation of Graph Signals,” *IEEE Trans. Signal Inf. Process. Netw.*, vol. 2, no. 4, pp. 555–568, Dec. 2016.

[23] “Reproducible Research | KTH”, `https://www.kth.se/ise/research/reproducibleresearch-1.433797`, 2020. Accessed: 2020-08-18.

Kernel Regression over Graphs using Random Fourier Features

Vitor R. M. Elias, Vinay Chakravarthi Gogineni, *Member, IEEE,* Wallace A. Martins, *Senior Member, IEEE* and Stefan Werner, *Senior Member, IEEE*

***Abstract*—This paper proposes efficient batch-based and online strategies for kernel regression over graphs (KRG). The proposed algorithms do not require the input signal to be a graph signal, whereas the target signal is defined over the graph. We first use random Fourier features (RFF) to tackle the complexity issues associated with kernel methods employed in the conventional KRG. For batch-based approaches, we also propose an implementation that reduces complexity by avoiding the inversion of large matrices. Then, we derive two distinct online strategies using RFF, namely, the mini-batch gradient KRG (MGKRG) and the recursive least squares KRG (RLSKRG). The stochastic-gradient KRG (SGKRG) is introduced as a particular case of the MGKRG. The MGKRG and the SGKRG are low-complexity algorithms that employ stochastic gradient approximations in the regression-parameter update. The RLSKRG is a recursive implementation of the RFF-based batch KRG. A detailed stability analysis is provided for the proposed online algorithms, including convergence conditions in both mean and mean-squared senses. A discussion on complexity is also provided. Numerical simulations include a synthesized-data experiment and real-data experiments on temperature prediction, brain activity estimation, and image reconstruction. Results show that the RFF-based batch implementation offers competitive performance with a reduced computational burden when compared to the conventional KRG. The MGKRG offers a convenient trade-off between performance and complexity by varying the number of mini-batch samples. The RLSKRG has a faster convergence than the MGKRG and matches the performance of the batch implementation.**

***Index Terms*—kernel regression on graphs, random Fourier features, stochastic gradient, recursive least squares**

I. Introduction

Graph signal processing (GSP) employs graph-structural information to model, process, and analyze signals defined over graph nodes [1]–[4]. The growing importance of GSP is due to its applicability to networked data processing, as connectivity between real-world elements progressively increases with the advent of the internet-of-things, sensor networks, and better communication technologies [5]–[7]. By associating real-world network elements with graph nodes and encoding their interrelations through graph edges, GSP leverages the graph structure to process or analyze the network data, modeled as a graph signal. Like conventional signal processing, the literature consists of various GSP techniques and approaches that address different needs associated with real-world networks.

Here, we are particularly interested in GSP approaches for modeling relations between a reference signal and a target signal, usually referred to as an input-output pair [8]–[18]. Typically, state-of-the-art techniques address the case where both reference and target signals share the same graph. In the context of linear system modeling, different learning problems have been studied within the GSP framework, e.g., classification on graphs [19], autoregressive models for graph signal prediction [15], [16], [20], dictionary learning [21], and distributed adaptive filtering [8]–[14]. Several learning strategies have been proposed for the nonlinear setting as well. In particular, kernel regression has been extensively employed for a range of nonlinear learning tasks, such as reconstruction [22]–[24] and prediction of graph signals [15].

In contrast to previous works, [15] proposes a batch-based kernel regression method that maps a general signal, not necessarily a graph signal, to an output signal that resides on a given graph. A penalty term, added to the loss function, achieves this mapping and enforces the graph signal at the output, whose smoothness across the graph is defined by the graph Laplacian. The batch implementation in [15] requires that all samples are available before computing the solution, which induces a delay when dealing with streaming data. Moreover, obtaining the regression parameters through the batch-based KRG for large amounts of data may be computationally prohibitive. Finally, the approach in [15] inherits the well-known scaling issue of kernel methods [25], [26] since the model dimension increases with the number of training samples, which increases with the network size and with time.

This work proposes an approach for kernel regression on graphs using random Fourier features (RFF) [27], [28], which enjoys a reduced model complexity compared to the batch-based KRG. Also, we derive and analyze two online strategies, namely, the mini-batch gradient KRG (MGKRG), with the particular case of the stochastic-gradient descent KRG (SGKRG), and the recursive least squares KRG (RLSKRG). The proposed RFF-based algorithms approximate the kernel evaluations by inner-products in a fixed-dimensional space, avoiding the model dimension dependency on the number of training samples encountered in the conventional KRG. Addi-

This work was partly supported by: Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) –- Grant number: 88887.310189/2018-00, CNPq, ERC project AGNOSTIC (Actively Enhanced Cognition based Framework for Design of Complex Systems – Grant number: 742648), FAPERJ, and the Research Council of Norway.

Vitor R. M. Elias and Stefan Werner are with the Norwegian University of Science and Technology, Norway (e-mail: vitor.elias@ntnu.no and stefan.werner@ntnu.no).

Vinay C. Gogineni is with the SimulaMet, Simula Research Laboratory, Norway (e-mail: vcgogineni@simula.no).

Wallace A. Martins is with the University of Luxembourg, Luxembourg (e-mail: wallace.alvesmartins@uni.lu).

Vitor R. M. Elias and Wallace A. Martins are also with the Federal University of Rio de Janeiro, Brazil.

tionally, we propose an efficient implementation applicable to the conventional and RFF-based KRG that avoids large-scale matrix inversions. Similar to the approach in KRG [15], the proposed algorithms produce signals that vary smoothly over the graph, while input signals need not reside on a graph.

Among the proposed online algorithms, the stochastic gradient implementations, SGKRG and MGKRG, offer low-complexity alternatives. While the SGKRG requires the least computational effort, the MGKRG can improve the performance at a small additional cost by incorporating more samples in the stochastic gradient approximation. The RLSKRG, being the most complex, has faster convergence and higher accuracy than the other online implementations.

This paper is organized as follows. In Section II, we present some basic GSP concepts, formulate the problem of learning over graphs, and briefly describe the KRG methodology proposed in [15]. Section III presents the proposed methodology for batch-based KRG using RFF, along with an efficient implementation for large networks. The proposed online algorithms, namely the MGKRG, the RLSKRG, and its efficient implementation, are presented in Section IV. Section V provides a convergence analysis of the proposed online algorithms, and Section VI provides a brief discussion on the complexity of the algorithms. Numerical experiments to validate the performance of the proposed algorithms on both synthesized and real data are presented in Section VII. In the real-data experiments, we tackle the problems of predicting temperature on a weather-station network, estimating brain activity, and reconstructing corrupted video frames. Finally, concluding remarks for this work are presented in Section VIII.

Mathematical notation: scalars are denoted by lowercase letters, column vectors by bold lowercase, and matrices by bold uppercase. Superscripts $(\cdot)^{\mathrm{T}}$ and $(\cdot)^{-1}$ denote the transpose and inverse operators, respectively. Given a matrix $\mathbf{A} = [\mathbf{a}_1\, \mathbf{a}_2\, \ldots\, \mathbf{a}_N]$, the column-stacking operation is denoted by $\mathrm{vec}(\mathbf{A}) = [\mathbf{a}_1^{\mathrm{T}}\, \mathbf{a}_2^{\mathrm{T}}\, \ldots\, \mathbf{a}_N^{\mathrm{T}}]^{\mathrm{T}}$ and the reverse operation that reshapes a column vector back to its appropriate matrix form is $\mathbf{A} = \mathrm{mat}(\mathrm{vec}(\mathbf{A}))$. The (i,j)th element of matrix $\mathbf{A}$ is denoted by $A_{i,j}$. Symbol $\otimes$ denotes the Kronecker product. $\mathbf{1}_N$ denotes the $N \times 1$ vector with all entries equal to unity and $\mathbf{I}_N$ denotes the $N \times N$ identity matrix. The $M \times N$ matrix with all entries equal to zero is denoted by $\mathbf{0}_{M\times N}$. $\|\cdot\|_2$ denotes the 2-norm of the argument vector or the spectral norm of the argument matrix. The Frobenius norm of the argument matrix is denoted by$\|\cdot\|_{\mathrm{F}}$. $\mathrm{E}[\cdot]$ denotes the expected value of the argument.

II. Background and Problem Formulation

Consider an undirected graph $\mathcal{G} = \{\mathcal{N}, \mathcal{E}\}$, where $\mathcal{N} = \{1, 2, \ldots, K\}$ is the set of nodes and $\mathcal{E}$ is the set of edges such that $(k,l) \in \mathcal{E}$ if nodes k and l are connected. To each edge $(k,l) \in \mathcal{E}$, a weight $w_{k,l} \in \mathbb{R}_+$ can be assigned, which represents the strength of the relation between nodes k and l [1]–[3]. The set of edges is usually represented by the adjacency matrix $\mathbf{A} \in \mathbb{R}_+^{K\times K}$, such that the entry $A_{k,l} = A_{l,k} = w_{k,l}$ if $(k,l) \in \mathcal{E}$ and $A_{k,l} = 0$ otherwise. At time instant n, the graph signal is defined by a vector $\mathbf{x}_n = [x_{1,n}\, x_{2,n}\, \ldots\, x_{K,n}]^{\mathrm{T}}$, with $x_{k,n} \in \mathbb{R}$ being the signal value at node k.

Let $\mathcal{N}_k$ denote the neighborhood of node k, which is the set of nodes connected to k including itself. The graph Laplacian matrix is defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D}$ is the degree matrix of $\mathcal{G}$, with $D_{k,k} = \sum_{l=1}^{K} w_{k,l}$. The graph Laplacian is associated with the total-variation metric $\nu(\mathbf{x})$ of a graph signal $\mathbf{x}$ as follows:

$$\begin{aligned}\nu(\mathbf{x}) &= \mathbf{x}^{\mathrm{T}}\mathbf{L}\mathbf{x} \\ &= \sum_{k<l} A_{k,l}\big(x_k - x_l\big)^2.\end{aligned} \tag{1}$$

The metric (1) represents how much a signal varies across the graph, taking into account the edge weights [2], [15].

Considering a graph-based system, which takes an input vector $\mathbf{x} \in \mathbb{R}^M$ and outputs a graph signal $\mathbf{t} \in \mathbb{R}^K$, we are interested in estimating the corresponding mapping $\mathcal{M}: \mathbb{R}^M \to \mathbb{R}^K$. Given a set of training (available) data pairs $\{\mathbf{x}_n, \mathbf{t}_n\}_{n=1}^N$, regression methods can estimate $\mathcal{M}$. Regression methods that leverage the graph structure to improve the estimation are proposed in [15]. These methods were shown to outperform other approaches that do not use graph information.

A. Kernel Regression on Graphs

In [15], the model is estimated in terms of a matrix $\mathbf{W} \in \mathbb{R}^{M\times K}$ such that

$$\mathbf{y}_n = \mathbf{W}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n), \tag{2}$$

where $\mathbf{y}_n$ is an estimate of the target graph signal $\mathbf{t}_n$ and $\boldsymbol{\phi}: \mathbb{R}^M \to \mathbb{R}^M$ is an unknown function of the input signal. The optimal parameter matrix $\mathbf{W}$ is found by minimizing the cost function

$$C(\mathbf{W}) = \sum_{n=1}^{N} \|\mathbf{t}_n - \mathbf{y}_n\|_2^2 + \alpha \mathrm{tr}(\mathbf{W}^{\mathrm{T}}\mathbf{W}) + \beta \sum_{n=1}^{N} \nu(\mathbf{y}_n), \tag{3}$$

where $N \geq M$. The cost function $C(\mathbf{W})$ augments traditional regression methods by incorporating the penalty $\sum_{n=1}^{N} \nu(\mathbf{y}_n)$, which enforces smoothness of the output signal with respect to the graph. Defining the matrices

$$\mathbf{T} = [\mathbf{t}_1\, \mathbf{t}_2\, \ldots\, \mathbf{t}_N]^{\mathrm{T}} \in \mathbb{R}^{N\times K}, \tag{4}$$

$$\mathbf{Y} = [\mathbf{y}_1\, \mathbf{y}_2\, \ldots\, \mathbf{y}_N]^{\mathrm{T}} \in \mathbb{R}^{N\times K}, \tag{5}$$

$$\boldsymbol{\Phi} = [\boldsymbol{\phi}(\mathbf{x}_1)\; \boldsymbol{\phi}(\mathbf{x}_2)\; \ldots\; \boldsymbol{\phi}(\mathbf{x}_N)]^{\mathrm{T}} \in \mathbb{R}^{N\times M}, \tag{6}$$

and assuming $\boldsymbol{\Phi}$ is full rank, we can make the substitution $\mathbf{W} = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Psi}$, so that the optimization is now conducted in terms of $\boldsymbol{\Psi} \in \mathbb{R}^{N\times K}$. The predicted output of the kernel regression is given by [15]

$$\begin{aligned}\mathbf{y} &= \boldsymbol{\Psi}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{\phi}(\mathbf{x}) = \boldsymbol{\Psi}^{\mathrm{T}}\mathbf{k}(\mathbf{x}) \\ &= \big(\mathrm{mat}\left((\mathbf{B}+\mathbf{C})^{-1}\mathrm{vec}(\mathbf{T})\right)\big)^{\mathrm{T}}\boldsymbol{\kappa}(\mathbf{x}),\end{aligned} \tag{7}$$

where $\boldsymbol{\kappa}(\mathbf{x}) = [\kappa_1(\mathbf{x})\;\; \kappa_2(\mathbf{x})\;\; \ldots\;\; \kappa_N(\mathbf{x})]^{\mathrm{T}}$, with $\kappa_n(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x})$. Also,

$$\mathbf{B} = (\mathbf{I}_K \otimes (\mathbf{K} + \alpha\mathbf{I}_N)), \tag{8}$$

$$\mathbf{C} = (\beta\mathbf{L} \otimes \mathbf{K}), \tag{9}$$

with $\mathbf{K} = \mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}} \in \mathbb{R}^{N\times N}$. Here, the kernel trick is employed to avoid the explicit knowledge of $\boldsymbol{\phi}(\cdot)$, by replacing the inner product $\kappa_n(\mathbf{x}_i) = \boldsymbol{\phi}(\mathbf{x}_i)^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)$ with a kernel $\kappa(\mathbf{x}_i, \mathbf{x}_n)$ [30], [31]. The method described in (7), which outputs an estimate $\mathbf{y}$ for an input $\mathbf{x}$, is referred to as kernel regression on graphs.

The regression in (7) is performed in a batch-based fashion, assuming that all training samples are available *a priori*. A significant drawback of this implementation is the inherent delay of batch-based implementations, as the computation of the parameter matrix $\mathbf{\Psi}$ must wait for all training samples $\{\mathbf{x}_n, \mathbf{t}_n\}_{n=1}^{N}$ to be available. The increase in the computational burden of the KRG with the number of training samples is twofold. First, computing $\mathbf{\Psi}$ becomes more complex as the dimensions of $\mathbf{K}$ increase with N. Second, the regression dimension increases as the size of $\mathbf{k}(\mathbf{x})$ increases with N, and each additional training sample requires a kernel evaluation. The model complexity also depends on the number of training samples N, requiring N kernel evaluations for each new input signal, which is an issue if an online implementation is derived. In the following section, we treat the growing complexity by proposing a batch-based approach using random Fourier features.

III. Batch KRG Using Random Fourier Features

Random Fourier features is a widely used technique to circumvent the scaling problems of kernel methods [27]. This methodology presumes that the evaluation of a shift-invariant kernel, which satisfies $\kappa(\mathbf{x}_m, \mathbf{x}_n) = \kappa(\mathbf{x}_m - \mathbf{x}_n, 0)$, can be approximated as an inner product in the D-dimensional RFF space. This turns the problem into a finite-dimension linear filtering problem while avoiding the evaluation of kernel functions. Let $\mathbf{z}_n$ be the mapping of $\mathbf{x}_n$ into the RFF space $\mathbb{R}^D$, given by

$$\mathbf{z}_n = (D/2)^{-\frac{1}{2}}\left[\cos(\mathbf{v}_1^{\mathrm{T}}\mathbf{x}_n + b_1)\ \ldots\ \cos(\mathbf{v}_D^{\mathrm{T}}\mathbf{x}_n + b_D)\right]^{\mathrm{T}}, \tag{10}$$

where the phase terms $\{b_i\}_{i=1}^{D}$ are drawn from a uniform distribution on the interval $[0,\,2\pi]$. Vectors $\{\mathbf{v}_i\}_{i=1}^{D}$ are realizations of a random variable with probability density function (pdf) $p(\mathbf{v})$ such that

$$\kappa(\mathbf{x}_m, \mathbf{x}_n) = \int p(\mathbf{v}) \exp\left(\mathrm{j}\mathbf{v}^{\mathrm{T}}(\mathbf{x}_m - \mathbf{x}_n)\right) \mathrm{d}\mathbf{v}, \tag{11}$$

where $\mathrm{j}^2 = -1$. In other words, the Fourier transform of $\kappa(\mathbf{x}_m, \mathbf{x}_n)$ is given by $p(\mathbf{v})$. From (10) and (11), it can be verified that $\mathrm{E}[\mathbf{z}_n^{\mathrm{T}}\mathbf{z}_m] = \kappa(\mathbf{x}_m, \mathbf{x}_n)$ [27].

A. RFF-based KRG

To employ RFF in the KRG methodology, we first consider the kth entry of the estimate $\mathbf{y}$ as

$$y_k = \mathbf{w}_k^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}), \tag{12}$$

where $\mathbf{w}_k$ denotes the kth column of the parameter matrix $\mathbf{W}$. Using the substitution $\mathbf{W} = \mathbf{\Phi}^{\mathrm{T}}\mathbf{\Psi}$, and the kernel trick $\kappa(\mathbf{x}_m, \mathbf{x}_n) = \boldsymbol{\phi}(\mathbf{x}_m)^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)$, (12) can be rewritten as

$$y_k = \left(\sum_{n=1}^{N} \Psi_{n,k}\boldsymbol{\phi}(\mathbf{x}_n)\right)^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}) = \left(\sum_{n=1}^{N} \Psi_{n,k}\kappa(\mathbf{x}_n, \mathbf{x})\right). \tag{13}$$

Using RFF, we can approximate (13) as

$$y_k \approx \sum_{n=1}^{N} \Psi_{n,k}\mathbf{z}_n^{\mathrm{T}}\mathbf{z} = \mathbf{h}_k^{\mathrm{T}}\mathbf{z}. \tag{14}$$

Finally, the RFF-based regression estimate for the entire graph signal is written as

$$\mathbf{y} = \mathbf{H}^{\mathrm{T}}\mathbf{z}, \tag{15}$$

where $\mathbf{H} = [\mathbf{h}_1\ \mathbf{h}_2\ \ldots\ \mathbf{h}_K] \in \mathbb{R}^{D\times K}$ is the representation of the regression coefficient matrix in the RFF space. Letting the matrix

$$\mathbf{Z} = [\mathbf{z}_1\ \mathbf{z}_2\ \ldots\ \mathbf{z}_N]^{\mathrm{T}} \in \mathbb{R}^{N\times D} \tag{16}$$

represent the RFF mapping of all training input vectors $\{\mathbf{x}_n\}_{n=1}^{N}$, and using $\mathbf{T}$ and $\mathbf{Y}$ as respectively defined in (4) and (5), the cost function (3) can be rewritten in terms of $\mathbf{H}$ as

$$C(\mathbf{H}) = \sum_{n=1}^{N} \|\mathbf{t}_n\|_2^2 - 2\mathrm{tr}(\mathbf{T}^{\mathrm{T}}\mathbf{Z}\mathbf{H}) + \mathrm{tr}(\mathbf{H}^{\mathrm{T}}\mathbf{Z}^{\mathrm{T}}\mathbf{Z}\mathbf{H}) + \alpha(\mathbf{H}^{\mathrm{T}}\mathbf{H}) + \beta\mathrm{tr}(\mathbf{H}^{\mathrm{T}}\mathbf{Z}^{\mathrm{T}}\mathbf{Z}\mathbf{H}\mathbf{L}). \tag{17}$$

The gradient of $C(\mathbf{H})$ with respect to $\mathbf{H}$ is given by

$$\nabla C(\mathbf{H}) = -2\mathbf{Z}^{\mathrm{T}}\mathbf{T} + 2\mathbf{Z}^{\mathrm{T}}\mathbf{Z}\mathbf{H} + 2\alpha\mathbf{H} + 2\beta\mathbf{Z}^{\mathrm{T}}\mathbf{Z}\mathbf{H}\mathbf{L}. \tag{18}$$

Setting $\nabla C(\mathbf{H}) = \mathbf{0}$, we obtain

$$(\mathbf{Z}^{\mathrm{T}}\mathbf{Z} + \alpha\mathbf{I}_D)\mathbf{H}_{\mathrm{opt}} + \beta\mathbf{Z}^{\mathrm{T}}\mathbf{Z}\mathbf{H}_{\mathrm{opt}}\mathbf{L} = \mathbf{Z}^{\mathrm{T}}\mathbf{T}. \tag{19}$$

Then, vectorizing both sides of (19) and using the relation $\mathrm{vec}(\mathbf{A}\mathbf{X}\mathbf{B}) = (\mathbf{B}^{\mathrm{T}} \otimes \mathbf{A})\mathrm{vec}(\mathbf{X})$, the regression coefficients in the RFF space can be obtained as

$$\mathrm{vec}(\mathbf{H}_{\mathrm{opt}}) = (\mathbf{B}_{\mathrm{RFF}} + \mathbf{C}_{\mathrm{RFF}})^{-1}\mathrm{vec}(\mathbf{Z}^{\mathrm{T}}\mathbf{T}), \tag{20}$$

where

$$\mathbf{B}_{\mathrm{RFF}} = (\mathbf{I}_K \otimes (\mathbf{Z}^{\mathrm{T}}\mathbf{Z} + \alpha\mathbf{I}_D)), \tag{21}$$

$$\mathbf{C}_{\mathrm{RFF}} = (\beta\mathbf{L} \otimes \mathbf{Z}^{\mathrm{T}}\mathbf{Z}). \tag{22}$$

Once the regression coefficients are trained, the target estimate $\mathbf{y}$ given an input signal $\mathbf{x}$ corresponding to $\mathbf{z}$ in the RFF space is given by

$$\mathbf{y} = \mathbf{H}_{\mathrm{opt}}^{\mathrm{T}}\mathbf{z}. \tag{23}$$

From (21) and (22), it can be observed that the computational burden of obtaining the regression parameters is drastically reduced when compared to the conventional KRG, as the size of the $\mathbf{B}_{\mathrm{RFF}}$ and $\mathbf{C}_{\mathrm{RFF}}$ is now $KD \times KD$, with D possibly much smaller than N. From (23), we see that the estimation does not depend on the number of training samples and the model has a fixed size D, requiring only the mapping of each new input sample into the RFF space.

B. Efficient Computation For Large Networks

For large networks, computing the inverses in (7) and (20) may be prohibitively complex. We propose an efficient way to compute the parameters in these cases. We adopt the notation of the conventional KRG, but the same reasoning applies directly to the RFF-based implementation. We rewrite $(\mathbf{B}+\mathbf{C})^{-1}$ as

$$\begin{aligned}(\mathbf{B}+\mathbf{C})^{-1} &= (\mathbf{I}_K \otimes (\mathbf{K}+\alpha\mathbf{I}_N) + \beta\mathbf{L}\otimes\mathbf{K})^{-1}\\ &= (\mathbf{I}_K\otimes\alpha\mathbf{I}_N + (\mathbf{I}_K+\beta\mathbf{L})\otimes\mathbf{K})^{-1}\\ &= (\alpha\mathbf{I}_{KN} + (\mathbf{I}_K+\beta\mathbf{L})\otimes\mathbf{K})^{-1}. \qquad (24)\end{aligned}$$

Consider the eigendecompositions $(\mathbf{I}_K+\beta\mathbf{L}) = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^{\mathrm{T}}$ and $\mathbf{K} = \mathbf{V}\mathbf{\Omega}\mathbf{V}^{\mathrm{T}}$. We use the mixed-product property $(\mathbf{AB})\otimes(\mathbf{CD}) = (\mathbf{A}\otimes\mathbf{C})(\mathbf{B}\otimes\mathbf{D})$ to rewrite the Kronecker product. Note also that matrices $\alpha\mathbf{I}_{KN}$ and $(\mathbf{I}_K+\beta\mathbf{L})\otimes\mathbf{K}$ are simultaneously diagonalizable. Then, (24) can be written as

$$(\mathbf{B}+\mathbf{C})^{-1} = (\mathbf{U}\otimes\mathbf{V})(\alpha\mathbf{I}_{KN}+\mathbf{\Sigma}\otimes\mathbf{\Omega})^{-1}(\mathbf{U}^{\mathrm{T}}\otimes\mathbf{V}^{\mathrm{T}}), \quad (25)$$

and

$$\begin{aligned}&(\mathbf{B}+\mathbf{C})^{-1}\mathrm{vec}(\mathbf{T})\\ &\quad= (\mathbf{U}\otimes\mathbf{V})(\alpha\mathbf{I}_{KN}+\mathbf{\Sigma}\otimes\mathbf{\Omega})^{-1}(\mathbf{U}^{\mathrm{T}}\otimes\mathbf{V}^{\mathrm{T}})\mathrm{vec}(\mathbf{T})\\ &\quad= (\mathbf{U}\otimes\mathbf{V})(\alpha\mathbf{I}_{KN}+\mathbf{\Sigma}\otimes\mathbf{\Omega})^{-1}\mathrm{vec}(\mathbf{V}^{\mathrm{T}}\mathbf{T}\mathbf{U}). \qquad (26)\end{aligned}$$

Letting

$$\mathbf{\Gamma} = \mathrm{mat}\left((\alpha\mathbf{I}_{KN}+\mathbf{\Sigma}\otimes\mathbf{\Omega})^{-1}\mathrm{vec}(\mathbf{V}^{\mathrm{T}}\mathbf{T}\mathbf{U})\right) \quad (27)$$

and using the relation $(\mathbf{B}^{\mathrm{T}}\otimes\mathbf{A})\mathrm{vec}(\mathbf{X}) = \mathrm{vec}(\mathbf{AXB})$, we have

$$\mathbf{\Psi} = \mathbf{V}\mathbf{\Gamma}\mathbf{U}^{\mathrm{T}}. \quad (28)$$

Hence, the dominating complexity is reduced from $(KN)^3$ operations due to matrix inversion to approximately K^3 and N^3 operations required for the eigendecompositions of $(\mathbf{I}_K+\beta\mathbf{L})$ and $\mathbf{K}$, respectively.

IV. Online Kernel Regression on Graphs

In what follows, we consider online implementations of the KRG. To bypass the dimensionality problem associated with the kernel dictionary, we resort to online RFF-based KRG implementations.

A. Mini-batch Stochastic-Gradient KRG

Consider the following minimization problem:

$$\min_{\mathbf{H}} \mathrm{E}\left[\|\mathbf{t}-\mathbf{y}\|_2^2\right] + \alpha\mathrm{tr}(\mathbf{H}^{\mathrm{T}}\mathbf{H}) + \mathrm{E}[\beta\nu(\mathbf{y})]. \quad (29)$$

Similar to the batch-based approach, which conducts the optimization over N batch samples, problem (29) considers the expectation of the regularized regression problem. The gradient of the cost function in (29) is

$$\nabla C(\mathbf{H}) = -2\mathbf{R}_{zt} + 2\mathbf{R}_z\mathbf{H} + 2\alpha\mathbf{H} + 2\beta\mathbf{R}_z\mathbf{H}\mathbf{L}, \quad (30)$$

where $\mathbf{R}_{zt} = \mathrm{E}[\mathbf{z}\mathbf{t}^{\mathrm{T}}]$ and $\mathbf{R}_z = \mathrm{E}[\mathbf{z}\mathbf{z}^{\mathrm{T}}]$. In practice, the statistics $\mathbf{R}_{zt}$ and $\mathbf{R}_z$ can be unavailable.

Algorithm 1: MGKRG

```
Initialization:
$\mathbf{H}_1 = \mathbf{0}_{D\times K}$;
draw vectors $\{\mathbf{v}_i\}_{i=1}^{D}$ from $p(\mathbf{v})$;
draw phase terms $\{b_i\}_{i=1}^{D}$ from $[0, 2\pi]$;
Learning:
for each time instant $n$ do
    map $\mathbf{r}_n$ into $\mathbf{z}_n$;
    if $(n \bmod \delta) = 0$ then
        $\mathbf{Z}_n = [\mathbf{z}_{(n-N_\mathrm{b}+1)} \cdots \mathbf{z}_n]^{\mathrm{T}}$;
        $\mathbf{T}_n = [\mathbf{t}_{(n-N_\mathrm{b}+1)} \cdots \mathbf{t}_n]^{\mathrm{T}}$;
        $\mathbf{Y}_n = \mathbf{Z}_n\mathbf{H}_n$;
        $\mathbf{E}_n = \mathbf{T}_n - \mathbf{Y}_n$;
        $\mathbf{H}_{n+1} = (1-\mu\alpha)\mathbf{H}_n + \frac{\mu}{N_\mathrm{b}}\mathbf{Z}_n^{\mathrm{T}}(\mathbf{E}_n - \beta\mathbf{Y}_n\mathbf{L})$;
    end
    store $\mathbf{z}_n$;
    release $\mathbf{z}_{(n-N_\mathrm{b}+1)}$;
end
```

In the proposed approach, we use mini-batch averages to approximate $\mathbf{R}_{zt}$ and $\mathbf{R}_z$. We define the matrices composed by the signals corresponding to each individual mini-batch as

$$\mathbf{Z}_n = [\mathbf{z}_{(n\delta-N_\mathrm{b}+1)} \cdots \mathbf{z}_{n\delta}]^{\mathrm{T}} \in \mathbb{R}^{N_\mathrm{b}\times D}$$

and

$$\mathbf{T}_n = [\mathbf{t}_{(n\delta-N_\mathrm{b}+1)} \cdots \mathbf{t}_{n\delta}]^{\mathrm{T}} \in \mathbb{R}^{N_\mathrm{b}\times K},$$

where $1 \le \delta \le N_\mathrm{b}$ is the batch displacement parameter. For the nth batch, we can compute the approximations $\hat{\mathbf{R}}_{zt,n} = (\mathbf{Z}_n^{\mathrm{T}}\mathbf{T}_n)/N_\mathrm{b}$ and $\hat{\mathbf{R}}_{z,n} = (\mathbf{Z}_n^{\mathrm{T}}\mathbf{Z}_n)/N_\mathrm{b}$. We implement the sliding-window MGKRG, with $\delta = 1$, such that consecutive mini-batches have maximum overlap of $N_\mathrm{b}-1$ samples. A particular case of the MGKRG is defined by making $N_\mathrm{b} = \delta = 1$. In this case, only the current sample is used to compute the approximation of the gradient. This corresponds to a stochastic-gradient approach and will be referred to as stochastic gradient KRG (SGKRG).

The regression parameters are updated at the nth batch by taking a step in the negative direction of the corresponding approximate gradient, i.e.,

$$\mathbf{H}_{n+1} = (1-\mu\alpha)\mathbf{H}_n + \frac{\mu}{N_\mathrm{b}}\mathbf{Z}_n^{\mathrm{T}}(\mathbf{T}_n - \mathbf{Z}_n\mathbf{H}_n - \beta\mathbf{Z}_n\mathbf{H}_n\mathbf{L}). \quad (31)$$

Letting $\mathbf{Y}_n = \mathbf{Z}_n\mathbf{H}_n$ be the mini-batch estimate and $\mathbf{E}_n = \mathbf{T}_n - \mathbf{Y}_n$ be the corresponding *a priori* error matrix, the update equation for the mini-batch gradient KRG is written as

$$\mathbf{H}_{n+1} = (1-\mu\alpha)\mathbf{H}_n + \frac{\mu}{N_\mathrm{b}}\mathbf{Z}_n^{\mathrm{T}}(\mathbf{E}_n - \beta\mathbf{Y}_n\mathbf{L}). \quad (32)$$

B. Recursive Least-Squares KRG

We now explore the principles of the recursive least squares algorithms [32] to derive a recursive learning of the regression coefficients of the RFFKRG. That is, we part from the same optimization problem (17) but, instead of solving (20) directly, we solve it recursively. First, we rewrite (20) as

$$\begin{aligned}\text{vec}(\mathbf{H}_n) &\\ &= \left((\mathbf{I}_K \otimes (\mathbf{Z}^{\mathrm{T}}\mathbf{Z} + \alpha\mathbf{I}_D)) + (\beta\mathbf{L} \otimes \mathbf{Z}^{\mathrm{T}}\mathbf{Z})\right)^{-1} \text{vec}(\mathbf{Z}^{\mathrm{T}}\mathbf{T}) \\ &= \mathbf{R}_n^{-1}\mathbf{r}_n, \qquad (33)\end{aligned}$$

where

$$\mathbf{R}_n = \alpha\mathbf{I}_K \otimes \mathbf{I}_D + (\mathbf{I}_K + \beta\mathbf{L}) \otimes \mathbf{Z}^{\mathrm{T}}\mathbf{Z} \qquad (34)$$

$$\mathbf{r}_n = \text{vec}(\mathbf{Z}^{\mathrm{T}}\mathbf{T}). \qquad (35)$$

Note that these terms are obtained at time n, i.e., once n training samples are available. We aim to write both $\mathbf{R}_n^{-1}$ and $\mathbf{r}_n$ in terms of $\mathbf{R}_{n-1}^{-1}$ and $\mathbf{r}_{n-1}$, respectively, to derive a recursive algorithm. First, we rewrite (34) as

$$\begin{aligned}\mathbf{R}_n &= \alpha\mathbf{I}_{KD} + (\mathbf{I}_K + \beta\mathbf{L}) \otimes \sum_{i=0}^{n} \mathbf{z}_i\mathbf{z}_i^{\mathrm{T}} \qquad (36)\\ &= \alpha\mathbf{I}_{KD} + (\mathbf{I}_K + \beta\mathbf{L}) \otimes \sum_{i=0}^{n-1} \mathbf{z}_i\mathbf{z}_i^{\mathrm{T}} + (\mathbf{I}_K + \beta\mathbf{L}) \otimes \mathbf{z}_n\mathbf{z}_n^{\mathrm{T}} \\ &= \mathbf{R}_{n-1} + (\mathbf{I}_K + \beta\mathbf{L}) \otimes \mathbf{z}_n\mathbf{z}_n^{\mathrm{T}}. \qquad (37)\end{aligned}$$

We rewrite the second term on the right-hand side (RHS) of (37) using the mixed-product property and the fact that the resulting matrix is symmetric, as

$$\begin{aligned}(\mathbf{I}_K + \beta\mathbf{L}) \otimes \mathbf{z}_n\mathbf{z}_n^{\mathrm{T}} &= ((\mathbf{I}_K + \beta\mathbf{L}) \otimes \mathbf{z}_n)(\mathbf{I}_K \otimes \mathbf{z}_n^{\mathrm{T}}) \\ &= (\mathbf{I}_K \otimes \mathbf{z}_n)((\mathbf{I}_K + \beta\mathbf{L}) \otimes \mathbf{z}_n^{\mathrm{T}})\end{aligned}$$

Now, letting $\mathbf{P}_n = (\mathbf{I}_K \otimes \mathbf{z}_n)$ and $\mathbf{Q}_n = ((\mathbf{I}_K + \beta\mathbf{L}) \otimes \mathbf{z}_n^{\mathrm{T}})$, we can use the matrix inversion lemma to derive a recursive equation for $\mathbf{R}_n^{-1}$ as

$$\mathbf{R}_n^{-1} = \mathbf{R}_{n-1}^{-1} - \underbrace{\mathbf{R}_{n-1}^{-1}\mathbf{P}_n\left(\mathbf{I}_K + \mathbf{Q}_n\mathbf{R}_{n-1}^{-1}\mathbf{P}_n\right)^{-1}}_{\mathbf{G}_n \in \mathbb{R}^{KD\times K}}\mathbf{Q}_n\mathbf{R}_{n-1}^{-1}. \qquad (38)$$

where the gain matrix $\mathbf{G}_n$ may be simplified as follows:

$$\mathbf{G}_n = \left(\mathbf{R}_{n-1}^{-1} - \mathbf{G}_n\mathbf{Q}_n\mathbf{R}_{n-1}^{-1}\right)\mathbf{P}_n = \mathbf{R}_n^{-1}\mathbf{P}_n. \qquad (39)$$

We now write (35) in a recursive manner as

$$\mathbf{r}_n = \text{vec}\left(\sum_{i=0}^{n} \mathbf{z}_i\mathbf{t}_i^{\mathrm{T}}\right) = \mathbf{r}_{n-1} + \text{vec}(\mathbf{z}_n\mathbf{t}_n^{\mathrm{T}}). \qquad (40)$$

Substituting (40) into (33), we obtain

$$\text{vec}(\mathbf{H}_n) = \mathbf{R}_n^{-1}\mathbf{r}_{n-1} + \mathbf{R}_n^{-1}\text{vec}(\mathbf{z}_n\mathbf{t}_n^{\mathrm{T}}). \qquad (41)$$

Using the relation $\text{vec}(\mathbf{AXB}) = (\mathbf{B}^{\mathrm{T}} \otimes \mathbf{A})\text{vec}(\mathbf{X})$ and the mixed-product property, $\text{vec}(\mathbf{z}_n\mathbf{t}_n^{\mathrm{T}})$ can be written as

$$\text{vec}(\mathbf{z}_n\mathbf{t}_n^{\mathrm{T}}) = \mathbf{t}_n \otimes \mathbf{z}_n = (\mathbf{I}_K \otimes \mathbf{z}_n)\mathbf{t}_n.$$

and (41) becomes

$$\begin{aligned}\text{vec}(\mathbf{H}_n) &= \mathbf{R}_n^{-1}\mathbf{r}_{n-1} + \mathbf{R}_n^{-1}(\mathbf{I}_K \otimes \mathbf{z}_n)\mathbf{t}_n \\ &= \mathbf{R}_n^{-1}\mathbf{r}_{n-1} + \mathbf{G}_n\mathbf{t}_n \qquad (42)\end{aligned}$$

Algorithm 2: RFF-based RLSKRG

```
Initialization:
$\mathbf{R}_{-1}^{-1} = \frac{1}{\alpha}\mathbf{I}_{KD}$;
$\mathbf{H}_{-1} = \mathbf{0}_{D\times K}$;
draw vectors $\{\mathbf{v}_i\}_{i=1}^{D}$ from $p(\mathbf{v})$;
draw phase terms $\{b_i\}_{i=1}^{D}$ from $[0, 2\pi]$;
Learning:
for each time instant $n$ do
    map $\mathbf{r}_n$ into $\mathbf{z}_n$;
    $\mathbf{P}_n = \mathbf{I}_K \otimes \mathbf{z}_n$;
    $\mathbf{Q}_n = (\mathbf{I}_K + \beta\mathbf{L}) \otimes \mathbf{z}_n^{\mathrm{T}}$;
    $\mathbf{G}_n = \mathbf{R}_{n-1}^{-1}\mathbf{P}_n\left(\mathbf{I}_K + \mathbf{Q}_n\mathbf{R}_{n-1}^{-1}\mathbf{P}_n\right)^{-1}$;
    $\hat{\mathbf{y}}_n = \mathbf{H}_{n-1}^{\mathrm{T}}\mathbf{z}_n$;
    $\mathbf{e}_n = \mathbf{t}_n - \hat{\mathbf{y}}_n$;
    $\mathbf{H}_n = \mathbf{H}_{n-1} + \text{mat}(\mathbf{G}_n(\mathbf{e}_n - \beta\mathbf{L}\hat{\mathbf{y}}_n))$;
    $\mathbf{R}_n^{-1} = \mathbf{R}_{n-1}^{-1} - \mathbf{G}_n\mathbf{Q}_n\mathbf{R}_{n-1}^{-1}$;
end
```

Substituting (38) into (42)

$$\begin{aligned}\text{vec}(\mathbf{H}_n) &= \mathbf{R}_{n-1}^{-1}\mathbf{r}_{n-1} - \mathbf{G}_n\mathbf{Q}_n\mathbf{R}_{n-1}^{-1}\mathbf{r}_{n-1} + \mathbf{G}_n\mathbf{t}_n \\ &= \text{vec}(\mathbf{H}_{n-1}) + \mathbf{G}_n(\mathbf{t}_n - \mathbf{Q}_n\text{vec}(\mathbf{H}_{n-1})) \\ &= \text{vec}(\mathbf{H}_{n-1}) + \mathbf{G}_n\Big(\mathbf{t}_n - (\mathbf{I}_K \otimes \mathbf{z}_n^{\mathrm{T}})\text{vec}(\mathbf{H}_{n-1}) \\ &\qquad - (\beta\mathbf{L} \otimes \mathbf{z}_n^{\mathrm{T}})\text{vec}(\mathbf{H}_{n-1})\Big) \\ &= \text{vec}(\mathbf{H}_{n-1}) + \mathbf{G}_n(\mathbf{t}_n - \mathbf{H}_{n-1}^{\mathrm{T}}\mathbf{z}_n - \beta\mathbf{L}\mathbf{H}_{n-1}^{\mathrm{T}}\mathbf{z}_n) \\ &= \text{vec}(\mathbf{H}_{n-1}) + \mathbf{G}_n(\mathbf{e}_n - \beta\mathbf{L}\hat{\mathbf{y}}_n), \qquad (43)\end{aligned}$$

or, equivalently,

$$\mathbf{H}_n = \mathbf{H}_{n-1} + \text{mat}(\mathbf{G}_n(\mathbf{e}_n - \beta\mathbf{L}\hat{\mathbf{y}}_n)), \qquad (44)$$

where $\hat{\mathbf{y}}_n = \mathbf{H}_{n-1}^{\mathrm{T}}\mathbf{z}_n$ is the *a priori* target estimate and $\mathbf{e}_n = \mathbf{t}_n - \hat{\mathbf{y}}_n$ is the *a priori* error. Equation (43) is the recursive update equation for the proposed recursive least squares KRG (RLSKRG) algorithm. The steps for the implementation of the RLSKRG algorithm are summarized in Algorithm 2.

Due to its recursive nature, the RLSKRG algorithm considers past samples when computing the update matrix at each iteration. Thus, its performance is expected to match that of the batch-based approach.

C. Efficient RLSKRG Implementation

The complexity associated with large matrix multiplications or inversions can render the RLSKRG impractical for large networks. For instance, the computations of $\mathbf{G}_n$ and $\mathbf{R}_n^{-1}$ in Algorithm 2 require multiplications of matrices with dimension $KD \times KD$ and $KD \times K$. This implies at least K^3D^2 multiplication operations for each computation. We now derive an alternative implementation with reduced complexity.

Substituting (36) into (39), and substituting the result into (44), we obtain

$$\begin{aligned}\mathbf{H}_n &= \mathbf{H}_{n-1} \\ &+ \text{mat}\left((\alpha\mathbf{I}_{KD} + (\mathbf{I}_K + \beta\mathbf{L}) \otimes \mathbf{R}_{z,n})^{-1}\boldsymbol{\xi}_n)\right), \qquad (45)\end{aligned}$$

Algorithm 3: Efficient RLSKRG

```
Initialization:
$\mathbf{R}_{z,-1} = \mathbf{0}_{D\times D}$;
$\mathbf{H}_{-1} = \mathbf{0}_{D\times K}$;
get $\mathbf{U}$ and $\boldsymbol{\Sigma}$;
draw vectors $\{\mathbf{v}_i\}_{i=1}^{D}$ from $p(\mathbf{v})$;
draw phase terms $\{b_i\}_{i=1}^{D}$ from $[0, 2\pi]$;
Learning:
for each time instant $n$ do
    map $\mathbf{r}_n$ into $\mathbf{z}_n$;
    $\mathbf{R}_{z,n} = \mathbf{R}_{z,n-1} + \mathbf{z}_n\mathbf{z}_n^{\mathrm{T}}$;
    Get $\mathbf{V}_n$ and $\boldsymbol{\Omega}_n$;
    $\mathbf{P}_n = \mathbf{I}_K \otimes \mathbf{z}_n$;
    $\hat{\mathbf{y}}_n = \mathbf{H}_{n-1}^{\mathrm{T}}\mathbf{z}_n$;
    $\mathbf{e}_n = \mathbf{t}_n - \hat{\mathbf{y}}_n$;
    $\boldsymbol{\Xi} = \mathrm{mat}(\mathbf{P}_n(\mathbf{e}_n - \beta\mathbf{L}\hat{\mathbf{y}}_n))$;
    $\boldsymbol{\Gamma}_n = \mathrm{mat}((\alpha\mathbf{I}_{KD} + \boldsymbol{\Sigma}\otimes\boldsymbol{\Omega}_n)^{-1}\mathrm{vec}(\mathbf{V}_n^{\mathrm{T}}\boldsymbol{\Xi}_n\mathbf{U}))$;
    $\mathbf{H}_n = \mathbf{H}_{n-1} + \mathbf{V}_n\boldsymbol{\Gamma}_n\mathbf{U}^{\mathrm{T}}$;
end
```

where $\mathbf{R}_{z,n} = \sum_{i=0}^{n}\mathbf{z}_n\mathbf{z}_n^{\mathrm{T}}$ and $\boldsymbol{\xi}_n = \mathbf{P}_n(\mathbf{e}_n - \beta\mathbf{L}\hat{\mathbf{y}}_n)$. We now use the eigendecompositions $(\mathbf{I}_K + \beta\mathbf{L}) = \mathbf{U}\boldsymbol{\Sigma}\mathbf{U}^{\mathrm{T}}$ and $\mathbf{R}_{z,n} = \mathbf{V}_n\boldsymbol{\Omega}_n\mathbf{V}_n^{\mathrm{T}}$. Using the mixed-product property of the Kronecker product, and considering that $\alpha\mathbf{I}_{KD}$ and $(\mathbf{I}_K + \beta\mathbf{L})\otimes\mathbf{R}_{z,n}$ share the same set of eigenvectors, (45) can be rewritten as

$$\begin{aligned}\mathbf{H}_n &= \mathbf{H}_{n-1}\\ &+ \mathrm{mat}\left((\mathbf{U}\otimes\mathbf{V}_n)(\alpha\mathbf{I}_{KD} + \boldsymbol{\Sigma}\otimes\boldsymbol{\Omega}_n)^{-1}\mathrm{vec}(\mathbf{V}_n^{\mathrm{T}}\boldsymbol{\Xi}_n\mathbf{U})\right),\end{aligned} \tag{46}$$

where $\boldsymbol{\Xi}_n = \mathrm{mat}(\boldsymbol{\xi}_n)$. Letting $\boldsymbol{\Gamma}_n = \mathrm{mat}((\alpha\mathbf{I}_{KD} + \boldsymbol{\Sigma}\otimes\boldsymbol{\Omega}_n)^{-1}\mathrm{vec}(\mathbf{V}_n^{\mathrm{T}}\boldsymbol{\Xi}_n\mathbf{U}))$, and using the relation $(\mathbf{B}^{\mathrm{T}}\otimes\mathbf{A})\mathrm{vec}(\mathbf{X}) = \mathrm{vec}(\mathbf{AXB})$, the update equation for the efficient RLSKRG algorithm is given by

$$\mathbf{H}_n = \mathbf{H}_{n-1} + \mathbf{V}_n\boldsymbol{\Gamma}_n\mathbf{U}^{\mathrm{T}}. \tag{47}$$

All steps for the implementation of the efficient RLSKRG are presented in Algorithm 3.

V. Convergence Analysis

This section examines the convergence of the proposed online algorithms; in particular, we study their first- and second-order stability conditions. In the following analysis, $\mathbf{H}_{\mathrm{o}}$ denotes the optimal linear estimator in the least mean squares sense of $\mathbf{T}_n$ in the RFF domain. In this case, $\mathbf{T}_n = \mathbf{Z}_n\mathbf{H}_{\mathrm{o}} + \boldsymbol{\Upsilon}_n$, where $\boldsymbol{\Upsilon}_n = [\boldsymbol{\upsilon}_{(n\delta-N_{\mathrm{b}}+1)}\,\cdots\,\boldsymbol{\upsilon}_{n\delta}]^{\mathrm{T}} \in \mathbb{R}^{N_{\mathrm{b}}\times K}$ denotes the corresponding optimum-error matrix, which satisfies the orthogonality condition $\mathrm{E}[\mathbf{Z}_n^{\mathrm{T}}\boldsymbol{\Upsilon}_n] = \mathbf{0}_{D\times K} \Leftrightarrow \mathrm{E}[(\mathbf{I}_K\otimes\mathbf{Z}_n^{\mathrm{T}})\mathrm{vec}(\boldsymbol{\Upsilon}_n)] = \mathbf{0}_{KD\times 1}$ [32], [33].

For the derivations that follow, let $\lambda_{\max}(\cdot)$ denote the maximum eigenvalue of the argument matrix and let $\rho(\cdot)$ denote the spectral radius of the argument matrix, i.e., the largest absolute value of its eigenvalues. Additionally, we use the following property of the Kronecker product: let the eigenvalues of a matrix $\mathbf{A}$ be $\{\lambda_1, \lambda_2, \ldots, \lambda_M\}$ and of a matrix $\mathbf{B}$ be $\{\sigma_1, \sigma_2, \ldots, \sigma_N\}$. Then, the eigenvalues of $\mathbf{A}\otimes\mathbf{B}$ and $\mathbf{B}\otimes\mathbf{A}$ are given by $\{\lambda_i\sigma_j\}_{i=1,j=1}^{M,N}$ [33].

A. First-Order Analysis of the MGKRG

Making the substitution in (31) and subtracting both sides from $\mathbf{H}_{\mathrm{o}}$ yields

$$\begin{aligned}\widetilde{\mathbf{H}}_{n+1} = \widetilde{\mathbf{H}}_n - \mu\alpha\widetilde{\mathbf{H}}_n - \frac{\mu}{N_{\mathrm{b}}}\mathbf{Z}_n^{\mathrm{T}}\mathbf{Z}_n\widetilde{\mathbf{H}}_n - \frac{\mu}{N_{\mathrm{b}}}\beta\mathbf{Z}_n^{\mathrm{T}}\mathbf{Z}_n\widetilde{\mathbf{H}}_n\mathbf{L}\\ + \frac{\mu}{N_{\mathrm{b}}}\beta\mathbf{Z}_n^{\mathrm{T}}\mathbf{Z}_n\mathbf{H}_{\mathrm{o}}\mathbf{L} + \mu\alpha\mathbf{H}_{\mathrm{o}} - \frac{\mu}{N_{\mathrm{b}}}\mathbf{Z}_n^{\mathrm{T}}\boldsymbol{\Upsilon}_n,\end{aligned} \tag{48}$$

where $\widetilde{\mathbf{H}}_n = \mathbf{H}_{\mathrm{o}} - \mathbf{H}_n$ is the parameter-deviation matrix. Defining $\tilde{\mathbf{h}}_n = \mathrm{vec}(\widetilde{\mathbf{H}}_n)$, $\mathbf{h}_{\mathrm{o}} = \mathrm{vec}(\mathbf{H}_{\mathrm{o}})$, and $\boldsymbol{\gamma}_n = \mathrm{vec}(\boldsymbol{\Upsilon}_n)$, the above recursion can be alternatively expressed as

$$\begin{aligned}\tilde{\mathbf{h}}_{n+1} = \left(\mathbf{I}_{KD} - \mu\left(\alpha\mathbf{I}_{KD} + \frac{1}{N_{\mathrm{b}}}(\mathbf{I}_K + \beta\mathbf{L})\otimes(\mathbf{Z}_n^{\mathrm{T}}\mathbf{Z}_n)\right)\right)\tilde{\mathbf{h}}_n\\ + \mu\left(\alpha\mathbf{I}_{KD} + \frac{\beta}{N_{\mathrm{b}}}\mathbf{L}\otimes(\mathbf{Z}_n^{\mathrm{T}}\mathbf{Z}_n)\right)\mathbf{h}_{\mathrm{o}}\\ - \frac{\mu}{N_{\mathrm{b}}}(\mathbf{I}_K\otimes\mathbf{Z}_n^{\mathrm{T}})\boldsymbol{\gamma}_n.\end{aligned} \tag{49}$$

To study the convergence behavior of the proposed MGKRG governed by the form (49), we make the following assumptions:

A1: The RFF-mapped data signal $\mathbf{z}_n$ is drawn from a wide-sense stationary multivariate random sequence with correlation matrix $\mathbf{R}_z = \mathrm{E}[\mathbf{z}_n\mathbf{z}_n^{\mathrm{T}}]$.

A2: For n large enough, the contribution of the batch $\mathbf{Z}_n$ to $\mathbf{H}_n$ is negligible, such that $\mathbf{H}_n$ is considered to be independent of $\mathbf{Z}_n$.

A3: The graph topology is assumed to be static, meaning that the graph Laplacian $\mathbf{L}$ is fixed throughout the process.

Theorem 1. A sufficient condition on the step size μ for the convergence of the proposed MGKRG algorithm governed by (32), is given by

$$0 < \mu < \frac{2}{\lambda_{\max}(\mathbf{R}_z) + \alpha + \beta\lambda_{\max}(\mathbf{L})\lambda_{\max}(\mathbf{R}_z)}. \tag{50}$$

Proof. Taking the expectation $\mathrm{E}[\cdot]$ on both sides of (49), using **A1**-**A2**, and using the orthogonality condition such that the error-related term can be set to zero, we obtain

$$\mathrm{E}[\tilde{\mathbf{h}}_{n+1}] = \boldsymbol{\mathcal{A}}\mathrm{E}[\tilde{\mathbf{h}}_n] + \boldsymbol{\mathcal{B}}\mathbf{h}_{\mathrm{o}}, \tag{51}$$

where

$$\begin{aligned}\boldsymbol{\mathcal{A}} &= \mathbf{I}_{KD} - \mu\left(\alpha\mathbf{I}_{KD} + (\mathbf{I}_K + \beta\mathbf{L})\otimes\mathbf{R}_z\right)\\ \boldsymbol{\mathcal{B}} &= \mu\left(\alpha\mathbf{I}_{KD} + \beta\mathbf{L}\otimes\mathbf{R}_z\right).\end{aligned} \tag{52}$$

Iterating the above recursion back down to zero, we obtain

$$\mathrm{E}[\tilde{\mathbf{h}}_n] = \boldsymbol{\mathcal{A}}^n\mathrm{E}[\tilde{\mathbf{h}}_0] + \sum_{j=0}^{n-1}\boldsymbol{\mathcal{A}}^{n-1-j}\boldsymbol{\mathcal{B}}\mathbf{h}_{\mathrm{o}}. \tag{53}$$

Therefore, we see that convergence is guaranteed if $\rho(\boldsymbol{\mathcal{A}}) < 1$. We note that a scalar matrix $a\mathbf{I}$, with $a \in \mathbb{R}$, is simultaneously diagonalizable with any arbitrary matrix with adequate dimensions. Using the properties of the Kronecker product, and recalling that the eigenvalues of $\mathbf{L}$ and $\mathbf{R}_z$ are non-negative, the above condition reduces to $0 < \mu(\alpha + (1 + \beta\lambda_{\max}(\mathbf{L}))\lambda_{\max}(\mathbf{R}_z)) < 2$. The result in (50) follows from here. $\square$

Remark 1. Under the convergence condition (50), (53) converges asymptotically to $(\mathbf{I}_{KD}-\boldsymbol{\mathcal{A}})^{-1}\boldsymbol{\mathcal{B}}\mathbf{h}_{\rm o}$, which reduces to $(\alpha\mathbf{I}_{KD}+(\mathbf{I}_K+\beta\mathbf{L})\otimes\mathbf{R}_z)^{-1}(\alpha\mathbf{I}_{KD}+\beta\mathbf{L}\otimes\mathbf{R}_z)\mathbf{h}_{\rm o}$. This means that $\lim_{n\to\infty}\mathbf{H}_n$ is a biased estimate of $\mathbf{H}_{\rm o}$. Also, the bias is introduced by the regularization coefficients α and β, such that a non-regularized problem leads to an unbiased solution.

B. Second-Order Analysis of the MGKRG

For the second-order analysis of the MGKRG, we consider the following additional assumption:

A4: The step size μ is sufficiently small so that the terms involving higher order powers of μ can be ignored.

Using **A1-A4**, the covariance matrix of the parameter deviation vector $\tilde{\mathbf{h}}_{n+1}$ is given by

$$\begin{aligned}\mathrm{E}[\tilde{\mathbf{h}}_{n+1}\tilde{\mathbf{h}}_{n+1}^{\rm T}] = \mathrm{E}[\tilde{\mathbf{h}}_n\tilde{\mathbf{h}}_n^{\rm T}] - \mu\mathrm{E}[\tilde{\mathbf{h}}_n\tilde{\mathbf{h}}_n^{\rm T}]\boldsymbol{\mathcal{C}} - \mu\boldsymbol{\mathcal{C}}\mathrm{E}[\tilde{\mathbf{h}}_n\tilde{\mathbf{h}}_n^{\rm T}]\\ - \mu\mathrm{E}[\tilde{\mathbf{h}}_n\mathbf{h}_{\rm o}^{\rm T}]\boldsymbol{\mathcal{D}} - \mu\boldsymbol{\mathcal{D}}\mathrm{E}[\mathbf{h}_{\rm o}\tilde{\mathbf{h}}_n^{\rm T}],\end{aligned} \tag{54}$$

where

$$\begin{aligned}\boldsymbol{\mathcal{C}} &= \alpha\mathbf{I}_{KD} + (\mathbf{I}_K+\beta\mathbf{L})\otimes\mathbf{R}_z\\ \boldsymbol{\mathcal{D}} &= \alpha\mathbf{I}_{KD} + \beta\mathbf{L}\otimes\mathbf{R}_z.\end{aligned} \tag{55}$$

The cross terms involving $\frac{\mu}{N_{\rm b}}(\mathbf{I}_K\otimes\mathbf{Z}_n^{\rm T})\boldsymbol{\gamma}_n$ are zero due to the orthogonality condition. By vectorizing both sides of (54) and defining $\boldsymbol{\eta}_n = \mathrm{vec}(\tilde{\mathbf{h}}_n\tilde{\mathbf{h}}_n^{\rm T})$, we can now write

$$\mathrm{E}[\boldsymbol{\eta}_{n+1}] = \boldsymbol{\Delta}\mathrm{E}[\boldsymbol{\eta}_n] + \boldsymbol{\Theta}_n, \tag{56}$$

where

$$\boldsymbol{\Delta} = \mathbf{I}_{K^2D^2} - \mu(\boldsymbol{\mathcal{C}}\otimes\mathbf{I}_{KD}) - \mu(\mathbf{I}_{KD}\otimes\boldsymbol{\mathcal{C}}) \tag{57}$$

and

$$\begin{aligned}\boldsymbol{\Theta}_n &=\\ &-\mu(\boldsymbol{\mathcal{D}}\otimes\mathbf{I}_{KD})\mathrm{vec}(\mathrm{E}[\tilde{\mathbf{h}}_n]\mathbf{h}_{\rm o}^{\rm T}) - \mu(\mathbf{I}_{KD}\otimes\boldsymbol{\mathcal{D}})\mathrm{vec}(\mathbf{h}_{\rm o}\mathrm{E}[\tilde{\mathbf{h}}_n]^{\rm T}).\end{aligned} \tag{58}$$

Theorem 2. Assume **A1-A4** hold. Then, the second-order convergence of the proposed gradient-based algorithms, namely the MGKRG and the SGKRG, is guaranteed under

$$0 < \mu < \frac{1}{\lambda_{\max}(\mathbf{R}_z) + \alpha + \beta\lambda_{\max}(\mathbf{L})\lambda_{\max}(\mathbf{R}_z)}. \tag{59}$$

Proof. Iterating the recursion (56) back down to zero, we obtain

$$\mathrm{E}[\boldsymbol{\eta}_n] = \boldsymbol{\Delta}^n\mathrm{E}[\boldsymbol{\eta}_o] + \sum_{j=0}^{n-1}\boldsymbol{\Delta}^{n-1-j}\boldsymbol{\Theta}_j. \tag{60}$$

Recalling that $\mathrm{E}[\tilde{\mathbf{h}}_n]$ is finite under (50), so $\boldsymbol{\Theta}_n$ converges asymptotically with n. Therefore, equation (60) is stable iff $\rho(\boldsymbol{\Delta}) < 1$. Since matrices $\boldsymbol{\mathcal{C}}\otimes\mathbf{I}_{KD}$ and $\mathbf{I}_{KD}\otimes\boldsymbol{\mathcal{C}}$ commute and are both diagonalizable, the eigenvalues of their sum equal the sum of their eigenvalues. Moreover, these matrices share the same eigenvalues under the properties of the Kronecker product. Then, the condition for $\rho(\boldsymbol{\Delta}) < 1$ reduces to

$$\rho(\mathbf{I}_{K^2D^2} - 2\mu(\boldsymbol{\mathcal{C}}\otimes\mathbf{I}_{KD})) < 1, \tag{61}$$

which can be written as $|1-2\mu\lambda_{\max}(\boldsymbol{\mathcal{C}})| < 1$. Substituting $\boldsymbol{\mathcal{C}}$ as in (55), the second-order convergence condition reduces to

$$0 < 2\mu\big(\alpha + (1+\beta\lambda_{\max}(\mathbf{L}))\lambda_{\max}(\mathbf{R}_z)\big) < 2, \tag{62}$$

from which (59) follows. □

Theorem 2 shows that the condition for second-order stability of the MGKRG is more strict than that of the first-order stability. The upper-bound imposed on the step-sizes for second-order stability is half of the upper-bound established in Theorem 1.

C. First-Order Analysis of the RLSKRG

In the analysis of the RLSKRG, the following additional assumption is considered:

A5: The random sequence that governs signals $\mathbf{z}_n$ is ergodic. Then, for sufficiently large n, $\mathbf{R}_n$ behaves as a deterministic matrix given by $\mathbf{R}_n = \alpha\mathbf{I}_{KD} + (\mathbf{I}_K+\beta\mathbf{L})\otimes(n+1)\mathbf{R}_z$.

Assumption **A5** is commonly employed in the analysis of RLS-based algorithms [34]. It considers that, given ergodicity, the time average of rank-one covariance matrices $\mathbf{z}_n\mathbf{z}_n^{\rm T}$ can be replaced by the expected value for large enough n.

Multiplying both sides of (33) from the left by $\mathbf{R}_n$, and using (40) in conjunction with (33) we can write

$$\begin{aligned}\mathbf{R}_n\mathrm{vec}(\mathbf{H}_n) &= \mathbf{r}_n\\ \mathbf{R}_n\mathrm{vec}(\mathbf{H}_n) &= \mathbf{r}_{n-1} + \mathrm{vec}(\mathbf{z}_n\mathbf{t}_n^{\rm T})\\ \mathbf{R}_n\mathrm{vec}(\mathbf{H}_n) &= \mathbf{R}_{n-1}\mathrm{vec}(\mathbf{H}_{n-1}) + \mathrm{vec}(\mathbf{z}_n\mathbf{t}_n^{\rm T})\end{aligned} \tag{63}$$

Substituting the model $\mathbf{t}_n = \mathbf{H}_{\rm o}^{\rm T}\mathbf{z}_n + \boldsymbol{v}_n$ into (63), we have

$$\mathbf{R}_n\mathrm{vec}(\mathbf{H}_n) = \mathbf{R}_{n-1}\mathrm{vec}(\mathbf{H}_{n-1}) + \mathrm{vec}(\mathbf{z}_n\boldsymbol{v}_n^{\rm T}) + \mathrm{vec}(\mathbf{z}_n\mathbf{z}_n^{\rm T}\mathbf{H}_{\rm o}) \tag{64}$$

We now subtract both sides from $\mathbf{R}_n\mathrm{vec}(\mathbf{H}_{\rm o})$. By recalling that $\tilde{\mathbf{h}}_n = \mathrm{vec}(\mathbf{H}_{\rm o}-\mathbf{H}_n)$, we obtain

$$\begin{aligned}\mathbf{R}_n\tilde{\mathbf{h}}_n =& \mathbf{R}_n\mathrm{vec}(\mathbf{H}_{\rm o}) - \mathbf{R}_{n-1}\mathrm{vec}(\mathbf{H}_{n-1})\\ &- \mathrm{vec}(\mathbf{z}_n\boldsymbol{v}_n^{\rm T}) - \mathrm{vec}(\mathbf{z}_n\mathbf{z}_n^{\rm T}\mathbf{H}_{\rm o}).\end{aligned} \tag{65}$$

Substituting (37) into the first term on the RHS, we rewrite (65) as

$$\mathbf{R}_n\tilde{\mathbf{h}}_n = \mathbf{R}_{n-1}\tilde{\mathbf{h}}_{n-1} - \mathrm{vec}(\mathbf{z}_n\boldsymbol{v}_n^{\rm T}) + (\beta\mathbf{L}\otimes\mathbf{z}_n\mathbf{z}_n^{\rm T})\mathrm{vec}(\mathbf{H}_{\rm o}). \tag{66}$$

Taking the recursion down to $n=0$ and solving for $\tilde{\mathbf{h}}_n$, we obtain

$$\tilde{\mathbf{h}}_n = \mathbf{R}_n^{-1}\mathbf{R}_0\tilde{\mathbf{h}}_0 + \mathbf{R}_n^{-1}\boldsymbol{\pi}_n, \tag{67}$$

where

$$\boldsymbol{\pi}_n = \sum_{i=0}^{n}\mathrm{vec}(\beta\mathbf{z}_i\mathbf{z}_i^{\rm T}\mathbf{H}_{\rm o}\mathbf{L} - \mathbf{z}_i\boldsymbol{v}_i^{\rm T}). \tag{68}$$

Using $\mathrm{vec}(\mathbf{AXB}) = (\mathbf{B}^{\rm T}\otimes\mathbf{A})\mathrm{vec}(\mathbf{X})$ and **A5**, we get

$$\boldsymbol{\pi}_n = (\beta\mathbf{L}\otimes(n+1)\mathbf{R}_z)\,\mathbf{h}_{\rm o} - \sum_{i=0}^{n}\mathrm{vec}(\mathbf{z}_i\boldsymbol{v}_i^{\rm T}). \tag{69}$$

Theorem 3. The RLSKRG described in Algorithm 2 is stable in the mean sense and converges to a steady state.

Proof. The expected value of the parameter deviation in (67) is given by

$$\mathrm{E}[\tilde{\mathbf{h}}_n] = \mathrm{E}[\mathbf{R}_n^{-1}\mathbf{R}_0\tilde{\mathbf{h}}_0] + \mathrm{E}[\mathbf{R}_n^{-1}\boldsymbol{\pi}_n]. \tag{70}$$

For sufficiently large n, we can apply **A5** so that $\mathbf{R}_n^{-1}$ can be regarded as a deterministic matrix for which $\lim_{n\to\infty}\mathbf{R}_n^{-1} = \mathbf{0}_{KD\times KD}$, since $\mathbf{R}_n$ is dominated by the term $(\mathbf{I}_K+\beta\mathbf{L})\otimes(n+1)\mathbf{R}_z$. Thus, the first term on the RHS of (70) tends to zero. As for the second term, under the same conditions we have that

$$\begin{aligned}\mathrm{E}[\mathbf{R}_n^{-1}\boldsymbol{\pi}_n] =& \mathbf{R}_n^{-1}\left(\beta\mathbf{L}\otimes(n+1)\mathbf{R}_z\right)\mathbf{h}_\mathrm{o}\\ &- \mathbf{R}_n^{-1}\sum_{i=0}^{n}\underbrace{\mathrm{vec}\left(\mathrm{E}[\mathbf{z}_i\boldsymbol{\upsilon}_i^\mathrm{T}]\right)}_{=\mathbf{0}_{KD\times 1}},\end{aligned} \tag{71}$$

where the second term on the RHS is zero due to the orthogonality condition. Regarding the first term, as $\mathbf{R}_n$ is dominated by the term $(\mathbf{I}_K+\beta\mathbf{L})\otimes(n+1)\mathbf{R}_z$, then we can write $\lim_{n\to\infty}\mathbf{R}_n^{-1}\left(\beta\mathbf{L}\otimes(n+1)\mathbf{R}_z\right)\mathbf{h}_\mathrm{o} = \boldsymbol{\mathcal{E}}\mathbf{h}_\mathrm{o}$, in which

$$\begin{aligned}\boldsymbol{\mathcal{E}} &= \lim_{n\to\infty}\left[(\mathbf{I}_K+\beta\mathbf{L})\otimes(n+1)\mathbf{R}_z\right]^{-1}\left(\beta\mathbf{L}\otimes(n+1)\mathbf{R}_z\right)\\ &= \left[(\mathbf{I}_K+\beta\mathbf{L})^{-1}\beta\mathbf{L}\right]\otimes\mathbf{I}_D\,,\end{aligned} \tag{72}$$

where we used the Kronecker product property $(\mathbf{A}\otimes\mathbf{B})^{-1} = (\mathbf{A}^{-1}\otimes\mathbf{B}^{-1})$ and the mixed-product property. Hence, we have that $\mathbf{H}_n$ is an asymptotically biased estimate of $\mathbf{H}_\mathrm{o}$, and by using the relation $(\mathbf{B}^\mathrm{T}\otimes\mathbf{A})\mathrm{vec}(\mathbf{X}) = \mathrm{vec}(\mathbf{AXB})$, we can rewrite the bias term $\boldsymbol{\mathcal{E}}\mathbf{h}_\mathrm{o}$ as follows: $\lim_{n\to\infty}\mathrm{E}[\tilde{\mathbf{H}}_n] = \beta\mathbf{H}_\mathrm{o}\mathbf{L}(\mathbf{I}_K+\beta\mathbf{L})^{-1}$. □

Remark 2. Under the convergence condition (50), the bias of the MGRKG tends to the bias of the RLSKRG when $\alpha\to 0^+$. In addition, the bias in the RLSKRG is introduced solely by the regularization coefficient β, since the regularization coefficient α contributes only with an initial condition for the matrix $\mathbf{R}_n$, which plays no role in the algorithm's average behavior as n grows to infinity.

D. Second-Order Analysis of the RLSKRG

For the second order analysis, we assume further that

A6: Variables $\mathbf{z}_n$ and $\mathbf{t}_n$ are jointly ergodic, so that, for sufficiently large n, $\sum_{i=0}^{n}\mathbf{z}_i\mathbf{t}_i^\mathrm{T} \approx (n+1)\mathrm{E}[\mathbf{z}_i\mathbf{t}_i^\mathrm{T}]$.

Assumptions **A5** and **A6** imply that, for sufficiently large n, matrix $\sum_{i=0}^{n}\mathbf{z}_i(\mathbf{t}_i^\mathrm{T}-\mathbf{z}_i^\mathrm{T}\mathbf{H}_\mathrm{o})$ can be approximated as $(n+1)\mathrm{E}[\mathbf{z}_i\boldsymbol{\upsilon}_i^\mathrm{T}]$, which is equal to $\mathbf{0}_{K\times D}$ due to the orthogonality condition.

Theorem 4. The RLSKRG described in Algorithm 2 is stable in the mean-squared sense and converges to a steady state.

Proof. From (67), we have

$$\begin{aligned}\mathrm{E}[\|\tilde{\mathbf{h}}_n\|_2^2] &= \mathrm{E}[\|\mathbf{R}_n^{-1}\mathbf{R}_0\tilde{\mathbf{h}}_0\|_2^2]\\ &\quad + 2\mathrm{E}[\tilde{\mathbf{h}}_0^\mathrm{T}\mathbf{R}_0\mathbf{R}_n^{-2}\boldsymbol{\pi}_n] + \mathrm{E}[\|\mathbf{R}_n^{-1}\boldsymbol{\pi}_n\|_2^2].\end{aligned} \tag{73}$$

For sufficiently large n, we can apply **A5** so that the first non-negative term on the RHS of (73) is upper bounded by $\|\mathbf{R}^{-1}\|_2^2\cdot\mathrm{E}[\|\mathbf{R}_0\tilde{\mathbf{h}}_0\|_2^2]$, which tends to zero since $\mathrm{E}[\|\mathbf{R}_0\tilde{\mathbf{h}}_0\|_2^2]$ is bounded and $\lim_{n\to\infty}\mathbf{R}_n^{-1} = \mathbf{0}_{KD\times KD}$. Under **A5** and **A6**, we can write $\mathbf{R}_n^{-1}\boldsymbol{\pi}_n = \mathrm{vec}\left(\beta\mathbf{H}_\mathrm{o}\mathbf{L}(\mathbf{I}_K+\beta\mathbf{L})^{-1}\right)$ for sufficiently large n. This implies both that the middle term on the RHS of (73) can be written as $2\mathrm{E}[\tilde{\mathbf{h}}_0^\mathrm{T}\mathbf{R}_0]\mathbf{R}_n^{-1}\mathrm{vec}\left(\beta\mathbf{H}_\mathrm{o}\mathbf{L}(\mathbf{I}_K+\beta\mathbf{L})^{-1}\right)$, which tends to zero as n grows to infinity, and that the last term on the RHS of (73) is finite. Therefore, the RLSKRG converges in the mean-squared sense to $\lim_{n\to\infty}\mathrm{E}[\|\tilde{\mathbf{h}}_n\|_2^2] = \|\mathrm{vec}\left(\beta\mathbf{H}_\mathrm{o}\mathbf{L}(\mathbf{I}_K+\beta\mathbf{L})^{-1}\right)\|_2^2$. □

VI. Discussion on Complexity

For the MGKRG algorithm, the update (32) requires $DK + N_\mathrm{b}(K^2+2DK+K)$ multiplication operations. That is, the complexity of the MGKRG increases linearly with N_b with a slope equal to $K^2+2DK+K$. Additionally, the MGKRG requires a memory to store $N_\mathrm{b} > 1$ samples. Hence, the batch-size translates into a trade-off between complexity and performance since the gradient approximation using more samples yields a better update direction than those using a reduced number of samples. In this sense, the SGKRG yields the lowest computational burden of the proposed online KRG implementations.

The proposed efficient implementation of the RLSKRG in (47) requires $D^3+D^2+2D^2K+5DK+2DK^2+K^2$ multiplication operations to update $\mathbf{H}_n$. The terms D^2 and D^3 correspond to the complexity of updating the matrix $\mathbf{R}_{z,n}$ and computing its eigendecomposition, respectively. Since $\mathbf{R}_{z,n}$ is only updated with $\mathbf{z}_n\mathbf{z}_n^\mathrm{T}$ at time n, and we only need its eigensystem, the complexity can be reduced using efficient techniques for rank-one updates of the singular value decomposition [35]. Other techniques for reducing the complexity of the RLSKRG can be considered. For instance, dichotomous-coordinate descent (DCD) iterations, which uses only additions and bit-shifts with no multiplications, have been considered for reduced-complexity RLS implementations [36]. Under a reasonable assumption that N_b has the same order of magnitude as D and K, we observe that the RLSKRG has a slightly heavier computational burden per iteration when compared to the MGKRG.

The efficient implementation (28) of the offline batch KRG using RFF requires $D^3+D^2N+2D^2K+3DK+2DK^2+KDN$ multiplications. We highlight that this complexity is considerably smaller than that of the conventional implementation (20), which requires the inversion of a $DK\times DK$ matrix, leading to complexity equivalent to D^3K^3 multiplications for the inversion operation only. Moreover, we note that the computations of $\mathbf{Z}^\mathrm{T}\mathbf{Z}$ and $\mathbf{Z}^\mathrm{T}\mathbf{T}$ depend on N and yield the terms D^2N and KND, respectively. This implies that the complexity of the offline RFF-based KRG is not constant with time. The batch-based KRG can be considered in an online fashion, such that matrices $\mathbf{Z}^\mathrm{T}\mathbf{Z}$ and $\mathbf{Z}^\mathrm{T}\mathbf{T}$ are stored and only rank-one updates are required at each time instant, reducing the complexity of these terms to D^2 and KD multiplications per iteration, respectively.

VII. Numerical Results

In this section, we validate the performance of the proposed algorithms with numerical experiments using both synthesized and real datasets.

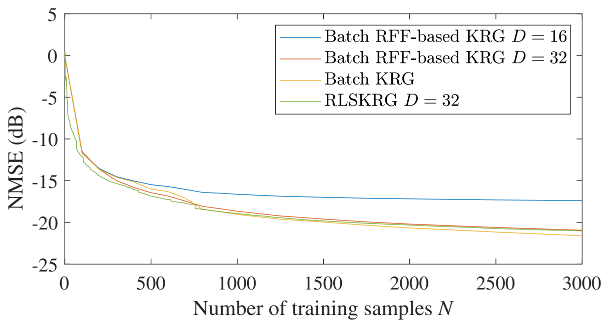


Fig. 1. NMSE achieved by the Bacht-based and RLSKRG implementations versus number of training samples using synthesized data.

A. Synthesized Data 1

Similar to the setup in [15], we consider an Erdös Rényi graph with $K = 50$ nodes and edge-probability equal to 0.1. A total of $S = 20000$ K-dimensional i.i.d. samples, $\{\mathbf{x}_n\}_{n=1}^{S}$, are generated, where $\mathbf{x}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{C}_S)$. The S-dimensional covariance matrix $\mathbf{C}_S \in \mathbb{R}^{S\times S}$ is drawn from the inverse Wishart distribution with an identity scale matrix. We generate the target graph signals $\{\mathbf{t}_n\}_{n=1}^{S}$ as in [15], i.e., by solving $\mathbf{t}_n = \arg\min_{\boldsymbol{\tau}} \left\{\|\mathbf{x}_n - \boldsymbol{\tau}\|_2^2 + \boldsymbol{\tau}^{\mathrm{T}}\mathbf{L}\boldsymbol{\tau}\right\}$. The generated signals are divided into a training set and a test set, containing N_{ts} and N samples, respectively, with $N_{\mathrm{ts}} + N \leq S$. The target signals in the training dataset are perturbed by white Gaussian noise (AWGN). The SNR is fixed across all nodes, with noise variance on the kth node $\sigma^2_{\mathrm{n},k} = \frac{\sigma^2_{\mathrm{s},k}}{\sqrt{10}}$, where $\sigma^2_{\mathrm{s},k}$ denotes the signal variance on the kth node. In our simulations, we fix $N_{\mathrm{ts}} = 1000$ and let N vary. Finally, α and β were obtained from the training set, via grid search and 5-fold cross-validation, by minimizing the normalized mean squared error

$$\mathrm{NMSE} = 10\log_{10}\left(\mathrm{E}\left[\frac{\|\mathbf{Y} - \mathbf{T}_0\|_{\mathrm{F}}^2}{\|\mathbf{T}_0\|_{\mathrm{F}}^2}\right]\right), \qquad (74)$$

where $\mathbf{T}$ denotes the true target matrix and $\mathbf{Y}$ denotes the estimated matrix. In the experiments, we use the Gaussian kernel $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2/(2\sigma^2)\right)$, with σ^2 also obtained via grid search.

We evaluate the NMSE over the entire test dataset for the proposed online algorithms at each iteration n. That is, for every n, we obtain $\mathbf{H}_n$, calculate the estimates of all N_{ts} test signals, and we compute the NMSE using (74). The expected value is obtained as the ensemble average over 500 independent runs.

Fig. 1 presents the results of the batch-based implementations and the RLSKRG. We see that the RFF implementation approximates well the conventional KRG even for relatively small $D = 32$. The performance of the RLSKRG closely matches the performance of the batch-based implementation. Results in Fig. 2a show that online algorithms can effectively learn the regression parameters. We analyze different step sizes and we show that the NMSE level achieved by the SGKRG approximates that of the batch RFF-based KRG as μ decreases. Fig. 2b shows the performance of the MGKRG for different mini-batch sizes. Plots show an increase in convergence speed as N_{b} increases to 15 and then to 50 samples.

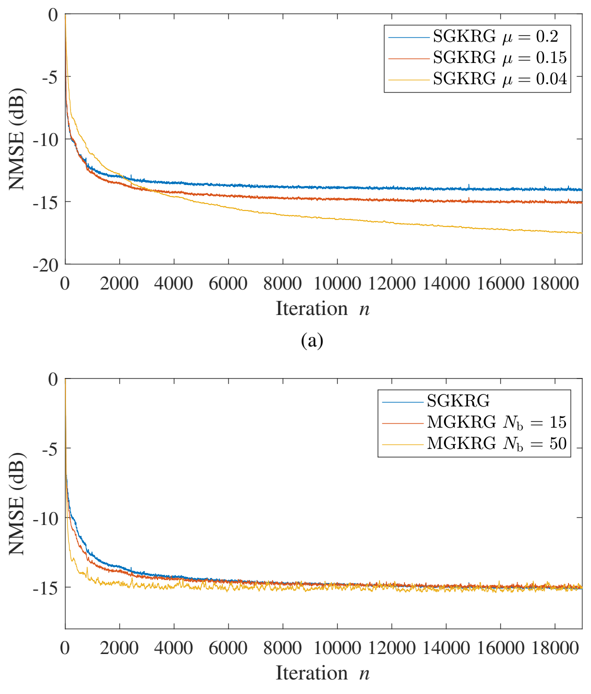


Fig. 2. NMSE achieved by the MGKRG implementations versus number of training samples for different step sizes and mini-batch sizes.

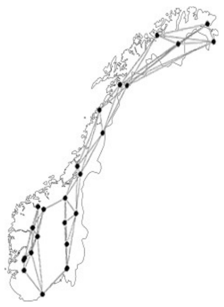
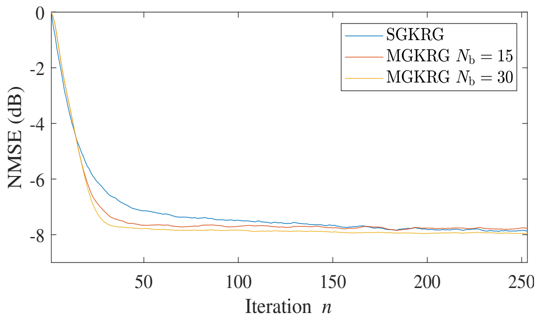
B. Real Data - Temperature Prediction

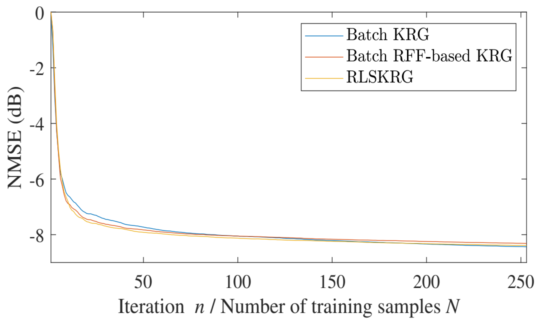
In this experiment, we use temperature data from 30 weather stations distributed across Norway's mainland, collected by the Norwegian Meteorological Institute [37]. Data from 2019 are used for the final experiment, while data from 2018 are used for training hyperparameters σ, α, and β.

We construct a nearest-neighbor graph with $K = 30$ nodes using the GSPBOX toolbox for MATLAB, such that each station is connected to its five nearest neighbors. The latitude and longitude coordinates of the stations are available in [37] and are used for computing the distance between stations. Fig. 3a shows the graph and the approximate positions of the weather stations.

We employ the KRG for a 4-days ahead temperature prediction on the 2019 data, with a 70% and 30% split between training and test data, respectively. For the RFF-based implementations, we use $D = 128$. The results are obtained as an ensemble average over 100 independent experiments, with different permutations of the data to generate the corresponding training and test datasets.

Results are presented in Fig. 3. In this example, we observe that the SG-based approaches, shown in Fig. 3b, achieve performance similar to the performance achieved by the batch-based approaches and the RLSKRG, in Fig. 3c. Comparing the SG implementations among themselves, we control the step size to achieve a similar steady-state NMSE. We observe that increasing the mini-batch size increases convergence speed. Plots in Fig. 3c show that the RLSKRG closely matches the batch-based KRG using RFF. Also, for $D = 128$ used in this example, the RFF offer an approximation that allows the RFF-



(a) (b) (c)

Fig. 3. Setup and results for temperature-prediction simulation: (a) illustration of the map of Norway and the approximate position of the stations, as represented by the graph used in the simulations; (b) results for SGKRG algorithms; and (c) results for batch-based algorithms and RLSKRG.

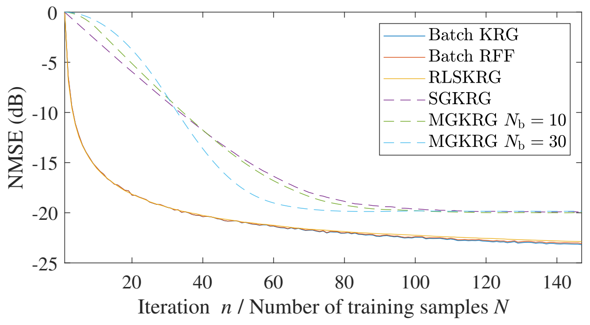


Fig. 4. NMSE achieved by the KRG implementations versus number of training samples for the fMRI signal simulation.

based KRG to match the conventional KRG in performance.

C. Real Data - fMRI Signal Extrapolation

This section reproduces the example from [15], which employs the conventional KRG to estimate the intensities of voxels in a functional magnetic resonance imaging (fMRI) dataset. The data and graph used are available in [38].

In the fMRI context, a voxel is a volumetric unit that constitutes a 3-dimensional image of the brain, analogous to pixels in 2-dimensional digital images. Each voxel is associated with a small cubic portion of the brain. The fMRI measures the changes in blood flow on each of these voxels. The blood flow is, in turn, associated with brain activity and, thus, by collecting measurements on all voxels, one can obtain a mapping of the brain activity. Regions of the brain relate to each other anatomically and, by considering these relations, a graph can be constructed where voxels are the nodes, and edges represent relations between them. For more details on this dataset and graph construction, see [15].

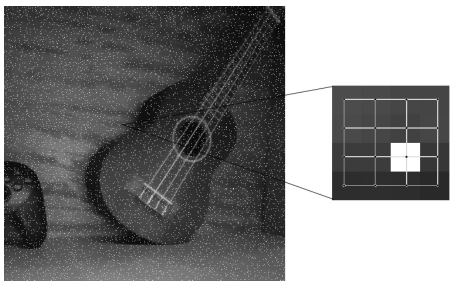
The regression experiment consists of estimating the signal on 90 of the voxels using the signal from other 10 voxels. In other words, we consider an input signal $\mathbf{x} \in \mathbb{R}^{10}$ to estimate a graph signal $\mathbf{t} \in \mathbb{R}^{90}$. The graph corresponds to the pairwise relations of the 90 voxels. A total of 292 snapshots is available. We consider training and test datasets of same size equal to 146 input-target pairs. The training signals are corrupted by an AWGN with covariance matrix $0.1 \cdot \mathbf{I}_K$. RFF-based implementations use $D = 32$.

Fig. 4 shows the results for all algorithms. We can observe that $D = 32$ is enough for the RFF-based KRG to closely match the conventional KRG, converging to approximately -23 dB. Again, the RLSKRG mostly coincides with the batch-based implementations. Results also show that the SG-based implementations can achieve low NMSE, around -20 dB, while increasing the number of samples when computing the stochastic approximation for the gradient increases the convergence speed.

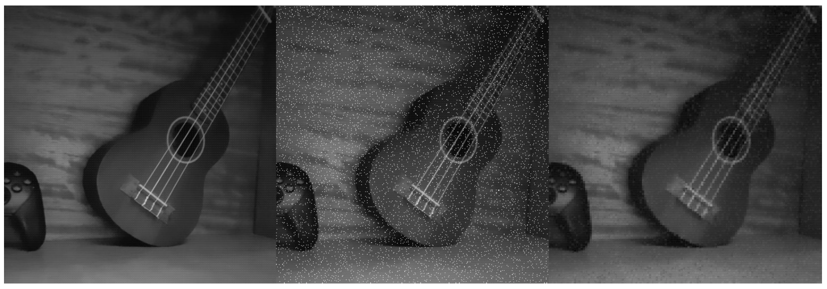
D. Real Data - Image Reconstruction

We now consider the application of KRG in the image and video processing scenario. This simulation showcases the performance and the capability of the online algorithms to deal with large datasets. In particular, we tackle the reconstruction of a corrupted video frame. Each frame is divided into blocks of 4×4 pixels. Each block of pixels is represented as a graph with $K = 16$ nodes, where each node corresponds to a pixel, and nodes are connected to their nearest neighbors inside a fixed radius equal to the minimum distance between two pixels. Frames are black and white, and pixels are treated in double format, such that a zero corresponds to black and a one corresponds to white. In this setup, corrupted frames have up to one random pixel per block that is set to unity, simulating a saturated pixel. An example of a corrupted frame, along with a block of 4×4 pixels with one corrupted pixel, and the corresponding graph are illustrated in Fig. 5a.

The video recording used in this simulation corresponds to a sequence of objects being captured against a generic wooden background as the camera pans from left to right, moving along the objects, at 30 frames per second. The video frames have a resolution of 480×480 pixels, which results in 14400 non-overlapping blocks per frame. We utilize six full frames taken with a distance of 50 frames between them to train the regression parameters, and two frames are used as the test dataset. We highlight that, in a real application, six frames correspond to approximately 0.2 seconds of video. Consistent learning during a video sequence becomes quickly impractical for the conventional KRG due to the large dataset.



(a)



(b)

Fig. 5. Example of the image reconstruction process using KRG: (a) shows an example frame and how a 4×4 block of pixels is treated as a graph; (b) shows the original frame, the corrupted frame, and the reconstructed frame, from left to right.

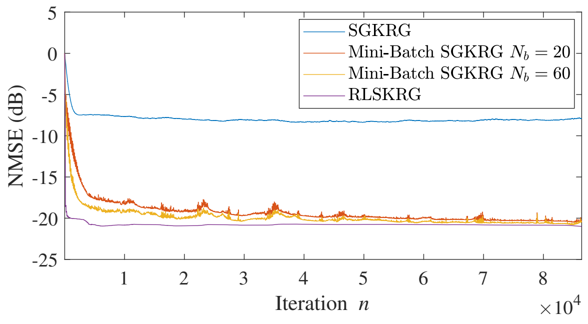


Fig. 6. NMSE achieved by the KRG implementations versus number of training samples in the image reconstruction simulation.

Fig. 6 shows the NMSE versus iterations for the proposed algorithms. These results are consistent with previous simulations and show that online KRG strategies can successfully learn the target model. We observe that the RLSKRG exhibits the best performance while the single-sample SGKRG exhibits the worst performance, as expected, given the complexity-performance trade-off. In this simulation, increasing the number of blocks to $N_\text{b} = 20$ and $N_\text{b} = 60$ (which corresponds to half the number of blocks on a single line in an image) considerably increases the performance of the MGKRG. A depiction of the frame reconstruction using the RLSKRG is presented in Fig. 5b, which showcases the capabilities of the proposed algorithm.

VIII. Conclusion

This paper proposed efficient batch-based implementations for kernel regression on graphs (KRG). The proposed implementations use random Fourier features (RFF) to overcome the growing complexity of kernel methods. Additionally, we showed that we could leverage the properties of the matrices involved in the regression process to formulate a less computationally-demanding derivation of the regression parameters. Furthermore, online strategies for RFF-based KRG were proposed, namely the mini-batch gradient KRG, the stochastic-gradient KRG, and the recursive least squares KRG. We showed that the RLSKRG also enjoys an alternative reduced-complexity implementation leveraging matrices' properties. For all online algorithms, conditions for convergence in the mean and the mean-squared sense were derived. We also presented a brief discussion on the trade-off between complexity and performance of the proposed algorithms. Finally, the performance of all algorithms was validated with numerical experiments using synthesized and real data simulations. Results confirmed that both the proposed KRG using RFF and the RLSKRG have accuracy close to that of the conventional KRG, with a considerable reduction in complexity. Additionally, simulations showed that the MGKRG can effectively learn the regression parameters and that its performance can be improved at a small increase in computational cost by increasing the number of samples in the mini-batch.

References

[1] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs," *IEEE Trans. Signal Process.*, vol. 61, pp. 1644–1656, Apr. 2013.

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

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

[4] V. R. M. Elias, W. A. Martins and S. Werner, "Extended adjacency and scale-dependent graph Fourier transform via diffusion distances," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 6, pp. 592-604, Aug. 2020.

[5] A. Sandryhaila and J. M. F. Moura, "Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure," *IEEE Signal Process. Mag.*, vol. 31, pp. 80–90, Sep. 2014.

[6] I. Jablonski, "Graph signal processing in applications to sensor networks, smart grids, and smart cities," *IEEE Sensors J.*, vol. 17, pp. 7659–7666, Dec. 2017.

[7] A. Gavili and X. Zhang, "On the shift operator, graph frequency, and optimal filtering in graph signal processing," *IEEE Trans. Signal Process.*, vol. 65, pp. 6303–6318, Dec. 2017.

[8] M. J. M. Spelta and W. A. Martins, "Normalized LMS algorithm and data-selective strategies for adaptive graph signal estimation," *Signal Process.*, vol. 167, 107326, Feb. 2020.

[9] F. Hua, R. Nassif, C. Richard, H. Wang and A. H. Sayed, "Diffusion LMS with communication delays: Stability and performance analysis," *IEEE Signal Process. Lett.*, vol. 27, pp. 730-734, 2020.

[10] P. Di Lorenzo, P. Banelli, S. Barbarossa, and S. Sardellitti, "Distributed adaptive learning of graph signals," *IEEE Trans. Signal Process.*, vol. 65, pp. 4193–4208, Aug. 2017.

[11] R. Nassif, C. Richard, J. Chen, and A. H. Sayed, "A graph diffusion LMS strategy for adaptive graph signal processing," in *Conf. Rec. Asilomar Conf. Signals Syst. Comput.*, pp. 1973–1976, Oct. 2017.

[12] R. Nassif, C. Richard, J. Chen, and A. H. Sayed, "Distributed diffusion adaptation over graph signals," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2018, pp. 4129–4133.

[13] F. Hua, R. Nassif, C. Richard, H. Wang and A. H. Sayed, "Online distributed learning over graphs with multitask graph-filter models," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 6, pp. 63–77, Jan. 2020.

[14] R. Nassif, S. Vlaski, C. Richard, J. Chen and A. H. Sayed, "Multitask learning over graphs: An approach for distributed, streaming machine learning," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 14-25, May 2020.

[15] A. Venkitaraman, S. Chatterjee and P. Händel, "Predicting graph signals using kernel regression where the input signal is agnostic to a graph," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 5, no. 4, pp. 698–710, Dec. 2019.

[16] V. R. M. Elias, V. C. Gogineni, W. A. Martins and S. Werner, "Adaptive graph filters in reproducing kernel Hilbert spaces: Design and performance analysis," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 7, pp. 62-74, Jan. 2021.

[17] P. Bouboulis, S. Chouvardas, and S. Theodoridis, "Online distributed learning over networks in RKH spaces using random Fourier features," *IEEE Trans. Signal Process.*, vol. 66, no. 7, pp. 1920–1932, 2018.

[18] Y. Shen, G. Leus, and G. B. Giannakis, "Online graph-adaptive learning with scalability and privacy," *IEEE Trans. Signal Process.*, vol. 67, pp. 2471–2483, May 2019.

[19] A. Sandryhaila and J. M. F. Moura, "Classification via regularization on graphs," in *Proc. IEEE Global Conf. Signal Inf. Process.*, 2013, pp. 495-498.

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

[21] D. Thanou, D. I. Shuman and P. Frossard, "Learning parametric dictionaries for signals on graphs," *IEEE Trans. Signal Process.*, vol. 62, no. 15, pp. 3849-3862, Aug 2014.

[22] D. Romero, M. Ma, and G. B. Giannakis, "Kernel-based reconstruction of graph signals," *IEEE Trans. Signal Process.*, vol. 65, no. 3, pp. 764–778, Feb. 2017.

[23] D. Romero, V. N. Ioannidis and G. B. Giannakis, "Kernel-based reconstruction of space-time functions on dynamic graphs," *IEEE J. Sel. Top. Signal Process.*, vol. 11, no. 6, pp. 856–869, Sep. 2017.

[24] V. N. Ioannidis, D. Romero and G. B. Giannakis, "Inference of Spatio-Temporal Functions Over Graphs via Multikernel Kriged Kalman Filtering," *IEEE Trans. Signal Process.*, vol. 66, no. 12, pp. 3228-3239, June 2018.

[25] A. Singh, N. Ahuja, and P. Moulin, "Online learning with kernels: Overcoming the growing sum problem," in *Proc. IEEE Int. Workshop Mach. Learn. Signal Process.*, 2012, pp. 1–6.

[26] C. Richard, J. Bermudez, and P. Honeine, "Online prediction of time series data with kernels," *IEEE Trans. Signal Process.*, vol. 57, no. 3, pp. 1058–1067, Mar. 2009.

[27] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Proc. Adv. Neural Inf. Process. Syst.*, pp. 1177–1184, 2007.

[28] P. Bouboulis, S. Pougkakiotis, and S. Theodoridis, "Efficient KLMS and KRLS algorithms: A random Fourier feature perspective," in *Proc. IEEE Stat. Signal Process. Workshop*, 2016, pp. 1–5.

[29] J. Kivinen, A. J. Smola and R. C. Williamson, "Online learning with kernels," *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2165-2176, Aug. 2004.

[30] A. J. Smola and R. Kondor, "Kernels and regularization on graphs," in *Learning Theory and Kernel Machines. Lecture Notes in Computer Science*, vol. 2777, pp. 144-158, Springer, 2003.

[31] W. Liu, J. C. Príncipe, and S. Haykin, *Kernel Adaptive Filtering*. Wiley, Feb. 2010.

[32] P. S. R. Diniz, *Adaptive Filtering*. Springer, 2013.

[33] A. H. Sayed, *Adaptive Filters*. Wiley, Jan. 2008.

[34] R. Arablouei, K. Doğançay, S. Werner and Y. Huang, "Adaptive distributed estimation based on recursive least-squares and partial diffusion," *IEEE Trans. Signal Process.*, vol. 62, no. 14, pp. 3510-3522, July 2014.

[35] M. Brand, "Fast low-rank modifications of the thin singular value decomposition," *Linear Algebra Appl.*, vol. 415, no 1, pp. 20–30, May 2006.

[36] R. Arablouei, K. Doğançay and S. Werner, "Recursive total least-squares algorithm based on inverse power method and dichotomous coordinate-descent iterations," *IEEE Trans. Signal Process.*, vol. 63, no. 8, pp. 1941–1949, Apr. 2015.

[37] Norwegian Meteorological Institute, *eKlima*, Norway, 2021. [Online] Available: http://sharki.oslo.dnmi.no/. [Accessed: 2021-01-08].

[38] KTH Royal Institute of Technology, Division of Information Science and Engineering, *Reproducible Research*, Sweden, 2020. [Online] Available: https://www.kth.se/ise/research/reproducibleresearch-1.433797. Accessed: 2020-08-18.