2020

Distributed Algorithms and Inverse Graph Filtering

Nazar Emirov
University of Central Florida

DISTRIBUTED ALGORITHMS AND INVERSE GRAPH FILTERING

by

NAZAR EMIROV
M.S. University of Central Florida, 2017
M.S. Western Illinois University, 2015
B.S. Fatih University, 2013

A dissertation submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy
in the Department of Mathematics
in the College of Sciences
at the University of Central Florida
Orlando, Florida

Fall Term
2020

Major Professor: Qiyu Sun

ABSTRACT

Graph signal processing provides an innovative framework to handle data residing on distributed networks, smart grids, neural networks, social networks and many other irregular domains. By leveraging applied harmonic analysis and graph spectral theory, graph signal processing has been extensively exploited, and many important concepts in classical signal processing have been extended to the graph setting such as graph Fourier transform, graph wavelets and graph filter banks. Similarly, many optimization problems in machine learning, sensor networks, power systems, control theory and signal processing can be modeled using underlying network structure. In modern applications, the size of a network is large, and amount of data needed to store and analyze is massive. Due to privacy and security concern, storage limitations and communication cost, a traditional centralized optimization methods are not suitable to solve these optimization problems, and distributed optimization methods are desirable.

Graph filters and their inverses have been widely used in denoising, smoothing, sampling, interpolating and learning. Implementation of an inverse filtering procedure on spatially distributed networks (SDNs) is a remarkable challenge, as each agent on an SDN is equipped with a data processing subsystem with limited capacity and a communication subsystem with confined range due to engineering limitations.

In this dissertation, we implement the filtering procedure associated with a polynomial graph filter of multiple shifts at the vertex level in a distributed network, where each vertex is equipped with a data processing subsystem for limited computation power and data storage, and a communication subsystem for direct data exchange to its adjacent vertices. We also consider the implementation of inverse filtering procedure associated with a polynomial graph filter of multiple shifts, and we propose two iterative approximation algorithms applicable in a distributed network and in a central

facility.

We also introduce a preconditioned gradient descent algorithm to implement the inverse filtering procedure associated with a graph filter having small geodesic-width. It is applicable for any invertible graph filters with small geodesic-width. The proposed algorithm converges exponentially, and it can be implemented at vertex level and applied to time-varying inverse filtering on SDNs.

Eigenspaces of some matrix on a network have been used for understanding the spectral clustering and influence of a vertex. Following the preconditioned gradient descent algorithm, for a matrix with small geodesic-width, we propose a distributed iterative algorithm to find eigenvectors associated with its given eigenvalue. We also consider the implementation of the proposed algorithm at the vertex/agent level in a spatially distributed network with limited data processing capability and confined communication range.

ACKNOWLEDGMENTS

First I would like to express my sincere gratitude to my advisor, Professor Qiyu Sun, for the guidance and encouragement he has given me during my entire time as a student at University of Central Florida. I have learned a lot under his supervision and my success is partially owed to him.

I am grateful to Dr. Zhisheng Shuai, Dr. Zuhair Nashed, Dr. Deguang Han and Dr. Zhihua Qu for serving in my dissertation committee.

Lastly, I would like to thank my friends and family for their support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

Graph signal processing provides an innovative framework to handle data residing on distributed networks, smart grids, neural networks, social networks and many other irregular domains [1, 2, 3]. Graphs provide a flexible tool to model the underlying topology of the networks, and the edges present the interrelationship between data elements. For instance, an edge between two vertices may indicate the availability of a direct data exchanging channel between sensors of a distributed network, or the correlation between temperature records of neighboring locations. By leveraging graph spectral theory and applied harmonic analysis, graph signal processing has been extensively exploited, and many important concepts in classical signal processing have been extended to graph setting [1, 2, 3, 4, 5, 7, 9, 60, 61].

Spatially distributed networks (SDNs) have been widely used in (wireless) sensor networks, drone fleets, smart grids and many real world applications [1, 9, 46, 60]. An SDN has a large amount of agents and each agent equipped with a data processing subsystem having limited data storage and computation power and a communication subsystem for data exchanging to its "neighboring" agents within communication range. The topology of an SDN can be described by a connected, undirected and unweighted finite graph $\mathcal{G} := (V, E)$ with a vertex in V representing an agent and an edge in E between vertices indicating that the corresponding agents are within some range in the spatial space.

In this work, we consider a graph signal processing problems such as graph filtering, inverse graph filtering and eigenvector approximation of a matrix on a spatially distributed networks.

1.1 Polynomial Graph Filters of Multiple Shifts and Distributed Implementation of Inverse Filtering

Let $\mathcal{G} := (V, E)$ be a connected, undirected and unweighted graph with vertex set $V = \{1, \ldots, N\}$ and edge set $E \subset V \times V$, and define the geodesic distance $\rho(i, j)$ between vertices $i, j \in V$ by the number of edges in a shortest path connecting $i, j \in V$. A *graph filter* on the graph $\mathcal{G}$ maps one graph signal linearly to another graph signal and it is usually represented by a matrix $\mathbf{H} = (H(i, j))_{i,j\in V}$. Graph filters and their implementations are fundamental in graph signal processing, and they have been used in denoising, smoothing, consensus of multi-agent systems, the estimation of time series and many other applications [10, 11, 12, 13]. In the classical signal processing, filters are categorized into two families, finite impulse response (FIR) filters and infinite impulse response (IIR) filters. The FIR concept has been extended to graph filters with the duration of an FIR filter being replaced by the geodesic-width of a graph filter. Here the *geodesic-width* $\omega(\mathbf{H})$ of a graph filter $\mathbf{H} = (H(i, j))_{i,j\in V}$ is the smallest nonnegative integer $\omega(\mathbf{H})$ such that $H(i, j) = 0$ hold for all $i, j \in V$ with $\rho(i, j) > \omega(\mathbf{H})$ [9, 14, 16, 17, 64].

An elementary graph filter is a *graph shift*, which has one as its geodesic-width [18, 19, 60, 64]. In this work, we introduce the concept of **multiple** commutative graph shifts $\mathbf{S}_1, \ldots, \mathbf{S}_d$, i.e.,

$$\mathbf{S}_k\mathbf{S}_{k'} = \mathbf{S}_{k'}\mathbf{S}_k,\ 1 \le k, k' \le d, \tag{1.1}$$

and we consider the implementation of filtering and inverse filtering associated with a *polynomial graph filter*

$$\mathbf{H} = h(\mathbf{S}_1, \ldots, \mathbf{S}_d) = \sum_{l_1=0}^{L_1} \cdots \sum_{l_d=0}^{L_d} h_{l_1,\ldots,l_d}\mathbf{S}_1^{l_1} \cdots \mathbf{S}_d^{l_d}, \tag{1.2}$$

where the polynomial

$$h(t_1, \ldots, t_d) = \sum_{l_1=0}^{L_1} \cdots \sum_{l_d=0}^{L_d} h_{l_1,\ldots,l_d} t_1^{l_1} \ldots t_d^{l_d}$$

in variables $t_1, \cdots, t_d$ has polynomial coefficients $h_{l_1,\ldots,l_d}$, $0 \le l_k \le L_k, 1 \le k \le d$. The com-

mutativity of graph shifts $\mathbf{S}_1, \ldots, \mathbf{S}_d$ guarantees that the polynomial graph filter $\mathbf{H}$ in (1.2) is independent on equivalent expressions of the multivariate polynomial h. The concept of commutative graph shifts $\mathbf{S}_1, \ldots, \mathbf{S}_d$ plays a similar role in graph signal processing as the one-order delay $z_1^{-1}, \ldots, z_d^{-1}$ in classical multi-dimensional signal processing, and in practice graph shifts may have specific features and physical interpretation, see Appendix and Section 2.4 for their joint spectrum and some illustrative examples. The commutative assumption on graph shifts $\mathbf{S}_1, \ldots, \mathbf{S}_d$ is trivial for $d = 1$ and polynomial graph filters of a single shift have been widely used in graph signal processing [11, 12, 18, 20, 21, 65, 66].

Polynomial graph filters $\mathbf{H}$ in (1.2) have geodesic-width $\omega(\mathbf{H})$ no more than the degree $\sum_{k=1}^{d} L_k$ of the polynomial h. Our study of polynomial graph filters of multiple shifts is motivated by signal processing on time-varying signals, such as video and data collected by a sensor network over a period of time, which carry different correlation characteristics for different dimensions/directions. In such a scenario, graph filters should be designed to reflect spectral characteristic on the vertex domain and also on the temporal domain, hence polynomial graph filters of multiple commutative shifts are **preferable**, see [2, 3, 24] and also Subsections 2.4.2 and 2.4.3. Our discussion is also motivated by directional frequency analysis in [24], feature separation in [25] and graph filtering in [26] for time-varying graph signals.

For polynomial graph filters of a single shift, algorithms have been proposed to implement their filtering procedure in finite steps, with each step including data exchanging between **adjacent** vertices only, see [10, 11, 12, 21, 27, 66] and also Algorithm 1. The first main contribution is that we provide the implementation of filtering procedure associated with polynomial graph filters of multiple shifts at vertex level, see Algorithm 2 in Section 2.1.

Inverse filtering plays an important role in graph signal processing, such as denoising, graph semi-supervised learning, non-subsampled filter banks and signal reconstruction [12, 20, 21, 27, 28,

29, 30, 64, 65, 66]. The challenge arisen in the inverse filtering is on its implementation, as the inverse filter $\mathbf{H}^{-1}$ usually has full geodesic-width even if the original filter $\mathbf{H}$ has small geodesic-width. For the case that the filter $\mathbf{H}$ is strictly positive definite, the inverse filtering procedure $\mathbf{b} \longmapsto \mathbf{H}^{-1}\mathbf{b}$ can be implemented by applying the iterative gradient descent method in a distributed network, see [27, 30, 31] and Remark 2.2.2. To consider implementation of inverse filtering of an arbitrary invertible filter $\mathbf{H}$ with small geodesic-width, in Section 2.2 we start from selecting a graph filter $\mathbf{G}$ with small geodesic-width to approximate the inverse filter $\mathbf{H}^{-1}$, and then we propose an **exponential convergent** algorithm (2.12) and (2.13) to implement the inverse filtering procedure with each iteration mainly including two filtering procedures associated with filters $\mathbf{H}$ and $\mathbf{G}$, see Theorem 2.2.1.

For an invertible polynomial graph filter of a single shift, there are several methods to implement the inverse filtering in a distributed network [12, 20, 21, 27, 66]. The second main contribution of this work is that we introduce optimal polynomial filters and Chebyshev polynomial filters to provide good approximations to the inverse of an invertible polynomial graph filter $\mathbf{H}$ of multiple shifts, see Section 2.3. Then, based on the iterative approximation algorithm in Section 2.2, we propose the iterative optimal polynomial approximation algorithm (2.31) and the iterative Chebyshev polynomial approximation algorithm (2.40) to implement the inverse filtering procedure $\mathbf{b} \longmapsto \mathbf{H}^{-1}\mathbf{b}$, see Theorems 2.3.1 and 2.3.3 for their exponential convergence. More importantly, as shown in Algorithms 3 and 4, each iteration in the proposed iterative algorithms mainly contains two filtering procedures involving data exchanging between **adjacent** vertices only and hence they can be implemented in a **distributed network** of large size, where each vertex is equipped with systems for limited data storage, computation power and data exchanging facility to its adjacent vertices. The effectiveness of these two iterative algorithms to implement the inverse filtering procedure is demonstrated in Section 2.4.

1.2 Preconditioned Gradient Descent Algorithm for Inverse Filtering on Spatially Distributed Networks

In this section, we consider SDNs equipped with a communication subsystem at each agent to directly communicate between two agents if the geodesic distance between their corresponding vertices $i, j \in V$ is at most L, i.e., $\rho(i, j) \leq L$, and we call the minimal integer $L \geq 1$ as the *communication range* of the SDN. Therefore the implementation of data processing on our SDNs is a distributed task and it should be designed at agent/vertex level with confined communication range. We also consider the implementation of graph filtering and inverse filtering on SDNs, which are required to be fulfilled at agent level with communication range no more than L.

A signal on a graph $\mathcal{G} = (V, E)$ is a vector $\mathbf{x} = (x(i))_{i\in V}$ indexed by the vertex set, and a graph filter $\mathbf{H}$ maps a graph signal $\mathbf{x}$ linearly to another graph signal $\mathbf{y} = \mathbf{H}\mathbf{x}$, which is usually represented by a matrix $\mathbf{H} = (H(i, j))_{i,j\in V}$ indexed by vertices in V. For a filter $\mathbf{H} = (H(i, j))_{i,j\in V}$ with geodesic-width $\omega(\mathbf{H})$, the corresponding filtering process

$$(x(i))_{i\in V} =: \mathbf{x} \longmapsto \mathbf{H}\mathbf{x} = \mathbf{y} := (y(i))_{i\in V} \tag{1.3}$$

can be implemented at vertex level, and the output at a vertex $i \in V$ is a "weighted" sum of the input in its $\omega(\mathbf{H})$-neighborhood,

$$y(i) = \sum_{\rho(j,i)\leq\omega(\mathbf{H})} H(i, j)x(j). \tag{1.4}$$

For SDNs with communication range $L \geq \omega(\mathbf{H})$, the above implementation at vertex level provides an essential tool for the filtering procedure (1.3), in which each agent $i \in V$ has equipped with subsystems to store $H(i, j)$ and $x(j)$ with $\rho(j, i) \leq \omega(\mathbf{H})$, to compute addition and multiplication in (1.4), and to exchange data to its neighboring agents $j \in V$ satisfying $\rho(j, i) \leq \omega(\mathbf{H})$.

For an invertible filter $\mathbf{H}$, the implementation of the inverse filtering procedure

$$\mathbf{y} \longmapsto \mathbf{H}^{-1}\mathbf{y} =: \mathbf{x} \tag{1.5}$$

cannot be directly applied for our SDNs, since the inverse filter $\mathbf{H}^{-1}$ may have geodesic-width *larger* than the communication range L. For the consideration of implementing inverse filtering on an SDN with communication range $L \geq 1$, we construct a diagonal preconditioning matrix $\mathbf{P}_{\mathbf{H}}$ in (3.1) at vertex level, and propose the preconditioned gradient descent algorithm (PGDA) (3.7) to implement inverse filtering on the SDN, see Algorithms 5 and 6.

A conventional approach to implement the inverse filtering procedure (1.5) is via the iterative quasi-Newton method

$$\mathbf{e}^{(m)} = \mathbf{H}\mathbf{x}^{(m-1)} - \mathbf{y} \text{ and } \mathbf{x}^{(m)} = \mathbf{x}^{(m-1)} - \mathbf{G}\mathbf{e}^{(m)},\ m \geq 1, \tag{1.6}$$

with arbitrary initial $\mathbf{x}^{(0)}$, where the graph filter $\mathbf{G}$ is an approximation to the inverse $\mathbf{H}^{-1}$. A challenge in the quasi-Newton method is how to select the approximation filter $\mathbf{G}$ appropriately. For the widely used polynomial graph filters $\mathbf{H} = h(\mathbf{S}) = \sum_{k=0}^{K} h_k \mathbf{S}^k$ of a graph shift $\mathbf{S}$ where $h(t) = \sum_{k=0}^{K} h_k t^k$ [11, 12, 13, 16, 21, 28, 51, 64, 66], several methods have been proposed to construct polynomial approximation filters $\mathbf{G}$ [12, 13, 21, 51, 66]. However, for the convergence of the corresponding quasi-Newton method, some prior knowledge is required for the polynomial h and the graph shift $\mathbf{S}$, such as the whole spectrum of the shift $\mathbf{S}$ in the optimal polynomial approximation method [51], the interval containing the spectrum of the shift $\mathbf{S}$ in the Chebyshev approximation method [12, 51, 66], and the spectral radius of the shift $\mathbf{S}$ and the zero set of the polynomial h in the autoregressive moving average filtering algorithm [13, 21]. For a non-polynomial graph filter $\mathbf{H}$, the approximation filter in the gradient descent method is of the form $\mathbf{G} = \beta\mathbf{H}^T$ with selection of the optimal step length β depending on maximal and minimal singular values of the

filter $\mathbf{H}$ [27, 28], and the approximation filter in the iterative matrix inverse approximation algorithm (IMIA) could be selected under a strong assumption on $\mathbf{H}$ [49, Theorem 3.2]. The proposed PGDA (3.7) is the quasi-Newton method (1.6) with $\mathbf{P}_{\mathbf{H}}^{-2}\mathbf{H}^T$ being selected as the approximation filter $\mathbf{G}$, see (3.3). Comparing with the quasi-Newton methods in [12, 13, 21, 27, 28, 49, 51, 66], one significance of the proposed PGDA is that the sequence $\mathbf{x}^{(m)}, m \geq 0$, in (3.7) converges exponentially to the output $\mathbf{x}$ of the inverse filtering procedure (1.5) whenever the filter $\mathbf{H}$ is invertible, see Theorems 3.1.3 and 3.2.1.

Data processing of time-varying signals, such as data collected by an SDN of sensors over a period of time, has been received a lot of attentions recently [2, 3, 26, 31, 41, 48, 51]. For a *time-varying* filter $\mathbf{H}_t = (H_t(i,j))_{i,j\in V}, t \geq 0$, with geodesic width $\omega(\mathbf{H}_t) \leq L$ bounded by the communication range L of the SDN, the quasi-Newton method (1.6) to implement the inverse filtering procedure $\mathbf{y}_t \longmapsto \mathbf{H}_t^{-1}\mathbf{y}_t,\ t \geq 0$, on the SDN should be designed to be *self-adaptive*, since each agent $i \in V$ of the SDN does not have the whole updated filter $\mathbf{H}_t$ and it only receives the entries $H_t(i,j)$ and $H_t(j,i), \rho(j,i) \leq L$, on the i-th row and column of $\mathbf{H}_t$ within the range L at every time instant t [9]. Clearly, the quasi-Newton method (1.6) is self-adaptive if the approximation filters $\mathbf{G}_t = (G_t(i,j))_{i,j\in V}, t \geq 0$ are locally selected without the involvement of any global information of the time-varying filter $\mathbf{H}_t$. The IMIA algorithm is self-adaptive [49, Eq. (3.4)] but the gradient descent method [27, 28] is not self-adaptive in general except that the step length β can be chosen to be *time-independent*. The second significance of the proposed PGDA is its *self-adaptivity* and *compatibility* to implement the time-varying inverse filtering procedure on our SDNs, as the preconditioner $\mathbf{P}_{\mathbf{H}}$ (and hence the approximation filter $\mathbf{P}_{\mathbf{H}}^{-2}\mathbf{H}^T$ in the PGDA) is constructed at the vertex level with confined communication range, see Algorithm 5.

1.3 Distributed Algorithms to Determine Eigenvectors of Matrices on Spatially Distributed Networks

Matrices on SDNs appear as graph filters in graph signal processing, transition matrices in Markov chains, state matrices of dynamic systems in control theory, sensing matrices in sampling theory, and in many more applications [9, 11, 12, 18, 20, 21, 51, 53, 55]. In the literature, their eigenspaces have been used to understand the communicability between vertices, spectral clustering for the network and influence of a vertex on the network [53, 54, 56, 57, 58, 59]. In this work, we consider complex-valued matrices with limited geodesic-width, where *geodesic-width* $\omega(\mathbf{A})$ of a matrix $\mathbf{A} = (A(i,j))_{i,j\in V}$ on the graph $\mathcal{G} = (V, E)$ is the smallest nonnegative integer such that $A(i,j) = 0$ for all $i, j \in V$ satisfying $\rho(i,j) > \omega(\mathbf{A})$. For a matrix $\mathbf{A}$ with small geodesic-width $\omega(\mathbf{A})$, we propose a distributed iterative algorithm in Section 4.1 to determine eigenvectors associated with its eigenvalue. The proposed algorithm is based on the preconditioned gradient descent approach in [55] for inverse filtering, and it can be implemented on SDNs with communication range $L \geq \omega(\mathbf{A})$. Moreover, the algorithm is scalable and its computational and communication expenses for subsystems equipped at every agent of the SDN is independent on the order of the graph $\mathcal{G}$. In this work, we also consider finding eigenvectors associated with the zero eigenvalue of a positive semidefinite matrix, and eigenvectors of a polynomial filter of multiple graph shifts, see Sections 4.2 and 4.3.

CHAPTER 2: POLYNOMIAL GRAPH FILTERS OF MULTIPLE SHIFTS AND DISTRIBUTED IMPLEMENTATION OF INVERSE FILTERING

In this chapter, we propose a polynomial graph filter of a multiple shifts and two distributed algorithms to implement an inverse graph filtering with multiple shifts in a distributed manner. Polynomial graph filters of multiple shifts are useful in a directional frequency analysis, feature separation, and graph filtering in time-varying graph signals.

2.1 Polynomial Filter and Distributed Implementation

Let $\mathcal{G} = (V, E)$ be a connected, undirected and unweighted graph of order N. Graph shifts $\mathbf{S}$ on $\mathcal{G}$ are building blocks of a polynomial filter. Our familiar examples of graph shifts are the adjacency matrix $\mathbf{A}_{\mathcal{G}}$, Laplacian matrix $\mathbf{L}_{\mathcal{G}} := \mathbf{D}_{\mathcal{G}} - \mathbf{A}_{\mathcal{G}}$, symmetric normalized Laplacian matrix $\mathbf{L}_{\mathcal{G}}^{\mathrm{sym}} = \mathbf{D}_{\mathcal{G}}^{-1/2}\mathbf{L}_{\mathcal{G}}\mathbf{D}_{\mathcal{G}}^{-1/2}$ and their variants, where $\mathbf{D}_{\mathcal{G}}$ is the degree matrix of the graph $\mathcal{G}$ [18, 19, 60, 64]. The filtering procedure $\mathbf{x} \longmapsto \mathbf{Sx}$ associated with a graph shift $\mathbf{S} = (S(i,j))_{i,j\in V}$ is a local operation that updates signal value at each vertex $i \in V$ by a "weighted" sum of signal values at **adjacent** vertices $j \in \mathcal{N}_i$,

$$\tilde{x}(i) = \sum_{j\in\mathcal{N}_i} S(i,j)x(j),$$

where $\mathbf{x} = (x(i))_{i\in V}$, $\mathbf{Sx} = (\tilde{x}(i))_{i\in V}$, and $\mathcal{N}_i$ is the set of adjacent vertices of $i \in V$. The above local implementation of filtering procedure has been extended to a polynomial graph filter

Algorithm 1 Realization of the filtering procedure $\mathbf{x} \longmapsto \mathbf{Hx}$ for a polynomial filter $\mathbf{H} = \sum_{l=0}^{L} h_l \mathbf{S}^l$ at a vertex $i \in V$.

Inputs: Polynomial coefficients $h_0, h_1, \ldots, h_L$, entries $S(i,j), j \in \mathcal{N}_i$ in the i-th row of the shift $\mathbf{S}$, and the value $x(i)$ of the input signal $\mathbf{x} = (x(i))_{i \in V}$ at the vertex i.
Initialization: $z^{(0)}(i) = h_L x(i)$ and $n = 0$.
1) Send $z^{(n)}(i)$ to its adjacent vertices $j \in \mathcal{N}_i$ and receive $z^{(n)}(j)$ from its adjacent vertices $j \in \mathcal{N}_i$.
2) Update $z^{(n+1)}(i) = h_{L-n-1} x(i) + \sum\limits_{j \in \mathcal{N}_i} S(i,j) z^{(n)}(j)$.
3) Set $n = n + 1$ and return to Step **1)** if $n \leq L - 1$.
Output: The value $\tilde{x}(i) = z^{(L)}(i)$ is the output signal $\mathbf{Hx} = (\tilde{x}(i))_{i \in V}$ at the vertex i.

$\mathbf{H} = \sum_{l=0}^{L} h_l \mathbf{S}^l$ of the shift $\mathbf{S}$,

$$\begin{cases} \mathbf{z}^{(0)} = h_L \mathbf{x}, \\ \mathbf{z}^{(n+1)} = h_{L-n-1} \mathbf{x} + \mathbf{S}\mathbf{z}^{(n)}, \ n = 0, \ldots, L-1, \\ \mathbf{Hx} = \mathbf{z}^{(L)}, \end{cases} \tag{2.1}$$

where the filtering procedure $\mathbf{x} \longmapsto \mathbf{Hx}$ is divided into $(L+1)$-steps with the procedure in each step being a local operation [11, 12, 21, 66]. The realization of the above implementation (2.1) at the vertex level is presented in Algorithm 1. In this section, we extend the above implementation to the filtering procedure associated with a polynomial graph filter $\mathbf{H}$ of multiple shifts, and propose a recursive algorithm containing about $\sum_{m=0}^{d-1} \prod_{k=1}^{m+1} (L_k + 1)$ steps with the output value at each vertex in each step being updated from some weighted sum of the input values at adjacent vertices of its preceding step, see Algorithm 2.

Let $\mathbf{S}_k = (S_k(i,j))_{i,j \in V}, 1 \leq k \leq d$, be commutative graph shifts, and $\mathbf{H}$ be the polynomial graph filter in (1.2) with $d \geq 2$. Define a matrix $\mathbf{U}_{d-1}$ of size $N \times \prod_{k=1}^{d-1} (L_k+1)$ with its $v_{d-1}(l_1, \ldots, l_{d-1})$-th column given by

$$\mathbf{U}_{d-1}\big(:, v_{d-1}(l_1, \ldots, l_{d-1})\big) = \sum_{l_d=0}^{L_d} h_{l_1, \ldots, l_{d-1}, l_d} \mathbf{S}_d^{l_d} \mathbf{x}, \tag{2.2}$$

where for $1 \leq m \leq d-1$,

$$v_m(l_1, \ldots, l_m) = l_m + l_{m-1}(L_m + 1) + \cdots + l_1 \prod_{k=2}^{m} (L_k + 1) \tag{2.3}$$

is the lexicographical order of $(l_1, \ldots, l_m)$ with $0 \leq l_k \leq L_k, 1 \leq k \leq m$. Follow the procedure in (2.1), we can evaluate $\mathbf{U}_{d-1}(:, v_{d-1}(l_1, \ldots, l_{d-1}))$ in $(L_d + 1)$-steps with the filtering procedure in each step being a local operation, see Step 1 in Algorithm 2 for the distributed implementation at vertex level. Moreover, one may verify that

$$\mathbf{Hx} = \sum_{l_1=0}^{L_1} \cdots \sum_{l_{d-1}=0}^{L_{d-1}} \mathbf{S}_1^{l_1} \cdots \mathbf{S}_{d-1}^{l_{d-1}} \mathbf{U}_{d-1}(:, v_{d-1}(l_1, \ldots, l_{d-1})) \tag{2.4}$$

by (1.2) and (2.2). By induction on $m = d-2, \ldots, 1$, we define matrices $\mathbf{U}_m$ of size $N \times \prod_{k'=1}^{m} (L_{k'} + 1)$ by

$$\mathbf{U}_m\big(:, v_m(l_1, \ldots, l_m)\big) = \sum_{l_{m+1}=0}^{L_{m+1}} \mathbf{S}_{m+1}^{l_{m+1}} \mathbf{U}_{m+1}\big(:, v_{m+1}(l_1, \ldots, l_m, l_{m+1})\big) \tag{2.5}$$

where $0 \leq l_k \leq L_k, 1 \leq k \leq m$. By induction on $m = d-2, \ldots, 1$ we obtain from (2.5) that every column of the matrix $\mathbf{U}_m$ can be evaluated from $\mathbf{U}_{m+1}$ in $(L_{m+1} + 1)$-steps, see Step 3 in Algorithm 2 for the distributed implementation at vertex level. By (2.4) and (2.5), we can prove

$$\mathbf{Hx} = \sum_{l_1=0}^{L_1} \cdots \sum_{l_m=0}^{L_m} \mathbf{S}_1^{l_1} \cdots \mathbf{S}_m^{l_m} \mathbf{U}_m(:, v_m(l_1, \ldots, l_m)) \tag{2.6}$$

by induction on $m = d-2, \ldots, 1$. Taking $m = 1$ in (2.6) yields

$$\mathbf{Hx} = \sum_{l_1=0}^{L_1} \mathbf{S}_1^{l_1} \mathbf{U}_1(:, l_1). \tag{2.7}$$

By (2.7), we finally evaluated the output $\mathbf{Hx}$ of the filtering procedure from the matrix $\mathbf{U}_1$ in

Algorithm 2 Realization of the filtering procedure $\mathbf{x} \longmapsto \mathbf{Hx}$ for the polynomial filter $\mathbf{H}$ of multiple graph shifts at a vertex $i \in V$.

Inputs: Polynomial coefficients $h_{l_1,\dots,l_d}, 0 \le l_1 \le L_1, \dots, 0 \le l_d \le L_d$ of the polynomial filter $\mathbf{H}$ in (1.2), entries $S_k(i,j), j \in \mathcal{N}_i$ of the i-th row of graph shifts $\mathbf{S}_k, 1 \le k \le d$, and the value $x(i)$ of the input graph signal $\mathbf{x} = (x(k))_{k\in V}$ at vertex i.

Step 1: Find the i-th row of the matrix $\mathbf{U}_{d-1}$.

for $p = 0, 1, \dots, \prod_{k=1}^{d-1}(L_k + 1) - 1$

Step 1a: write $p = v_{d-1}(l_1, \dots, l_{d-1})$ for some $0 \le l_k \le L_k, 1 \le k \le d-1$.

Step 1b: apply Algorithm 1 with polynomial coefficients and entries of the graph shift being replaced by polynomial coefficients $h_{l_1,\dots,l_{d-1},l_d}, 0 \le l_d \le L_d$, and entries $S_d(i,j), j \in N_i$ in the i-th row of the shift $\mathbf{S}_d$, and denote the corresponding output by $z^{(L_d)}(i)$.

Step 1c: set $\mathbf{U}_{d-1}(i,p) = z^{(L_d)}(i)$.

end

Step 2: **if** $d = 2$, set $\mathbf{W}(i,j) = \mathbf{U}_{d-1}(i,j), 0 \le j \le L_1$ and do **Step 4**, **otherwise** do **Step 3**.

Step 3: Find the i-th row of the matrix $\mathbf{U}_m, d-2 \ge m \ge 1$.

for $m = d-2, \dots, 2, 1$

for $p = 0, 1, \dots, \prod_{k=1}^{m}(L_k + 1) - 1$

Step 3a: apply Algorithm 1 with polynomial coefficients, entries of the graph shift and the value of input being replaced by polynomial coefficients $h_l = 1, 0 \le l \le L_{m+1}$, entries $S_{m+1}(i,j), j \in N_i$ in the i-th row of the shift $\mathbf{S}_{m+1}$, and the value $z^{(0)}(i) = \mathbf{U}_{m+1}\big(i, p(L_{m+1} + 1) + L_{m+1}\big)$ of the $(p(L_{m+1}+1)+L_{m+1})$-column of the matrix $\mathbf{U}_{m+1}$, and denote the corresponding output by $z^{(L_{m+1})}(i)$.

Step 3b: set $\mathbf{U}_m(i,p) = z^{(L_{m+1})}(i)$.

end

end

Set $\mathbf{W}(i,j) = \mathbf{U}_1(i,j), 0 \le j \le L_1$.

Step 4: Find the value of the output signal $\mathbf{Hx}$ at vertex i.

Step 4a: apply Algorithm 1 with polynomial coefficients, entries of the graph shift and the value of input being replaced by polynomial coefficients $h_l = 1, 0 \le l \le L_1$, entries $S_1(i,j), j \in N_i$ in the i-th row of the shift $\mathbf{S}_1$, and the value $u^{(0)}(i) = \mathbf{W}\big(i, L_1\big)$ of the L_1-column of the matrix $\mathbf{W}$.

Step 4b: Denote the corresponding output by $u^{(L_1)}(i)$.

Output: The value $\tilde{x}(i) = u^{(L_1)}(i)$ is the output signal $\mathbf{Hx} = (\tilde{x}(i))_{i\in V}$ at the vertex i.

$(L_1 + 1)$-steps with the filtering procedure in each step being a local operation, see Step 4 in Algorithm 2 for the implementation at vertex level.

Denote the degree of the graph $\mathcal{G}$ by $\deg \mathcal{G}$, and for two positive quantities a and b, we denote $a = O(b)$ if $a \le Cb$ for some absolute constant C. Recall that the number of nonzero entries in

every row of a graph shift on the graph $\mathcal{G}$ is no more than $\deg \mathcal{G}+1$. To implement (2.2), (2.5) and (2.7) in a central facility, the operations of addition and multiplication are about $2N(\deg \mathcal{G}+1)\prod_{k=1}^{d}(L_k+1)$, $2N(\deg \mathcal{G}+1)\sum_{m=1}^{d-2}\prod_{k=1}^{m+1}(L_k+1)$ and $2N(\deg \mathcal{G}+1)(L_1+1)$ respectively, and memory required are about $d(\deg \mathcal{G}+1)N+\prod_{k=1}^{d}(L_k+1)+2N+N\sum_{m=0}^{d-1}\prod_{k=1}^{m}(L_k+1)$ to store the graph shifts $\mathbf{S}_1,\ldots,\mathbf{S}_d$, the polynomial coefficients of the polynomial graph filter $\mathbf{H}$, the original graph signal $\mathbf{x}$, the output $\mathbf{Hx}$ of the filtering procedure and matrices $\mathbf{U}_m, 1\le m\le d-1$, in (2.2), (2.5) and (2.6). Hence for the implementation of the filter procedure $\mathbf{x}\longmapsto\mathbf{Hx}$ in a central facility via applying (2.2), (2.5) and (2.7), the total computational cost is about $O\big(N\deg \mathcal{G}+(N+L_d+1)\prod_{k=1}^{d-1}(L_k+1)\big)$ and the memory requirement is about $O\big(N(\deg \mathcal{G}+1)\prod_{k=1}^{d}(L_k+1)\big)$.

Shown in Algorithm 2 is the implementation of (2.2), (2.5) and (2.7) at the vertex level. Hence it is implementable in a distributed network where each agent is equipped with a data processing subsystem for limited data storage and computation power, and a communication subsystem for direct data exchange to its adjacent vertices. Denote the cardinality of a set E by $\#E$. To implement Algorithm 2 in a distributed network, we see that the data processing subsystem at a vertex $i\in V$ performs about $O\big((\#\mathcal{N}_i+1)\sum_{m=0}^{d-1}\prod_{k=1}^{m+1}(L_k+1)\big)=O\big((\deg \mathcal{G}+1)\prod_{k=1}^{d}(L_k+1)\big)$ operations of addition and multiplication, and it stores data of size about $O\big(\prod_{k=1}^{d}(L_k+1)+(\#\mathcal{N}_i+1)(d+2+\sum_{m=0}^{d-1}\prod_{k=1}^{m}(L_k+1))\big)=O\big((\deg \mathcal{G}+L_d+1)\prod_{k=1}^{d-1}(L_k+1)\big)$, including polynomial coefficients of the filter $\mathbf{H}$, the i-th row of graph shifts $\mathbf{S}_1,\ldots,\mathbf{S}_d$, and the i-th and its adjacent j-th components of the original graph signal $\mathbf{x}$, the output $\mathbf{Hx}$ of the filtering procedure and the matrices $\mathbf{U}_m, 1\le m\le d-1$, where $j\in\mathcal{N}_i$. Comparing the implementation of (2.2), (2.5) and (2.7) in a central facility, the total computational cost to implement Algorithm 2 in a distributed network is almost the same, while the total memory is slightly large, since the polynomial coefficients of the polynomial graph filter $\mathbf{H}$ needs to be stored at every agent in a distributed network while only one copy of the coefficients needs to be stored in a central facility. In addition to data processing in a central facility, the implementation of Algorithm 2 in a distributed network requires

that every agent $i \in V$ communicates with its adjacent agents $j \in \mathcal{N}_i$ with the j-th components of the original graph signal $\mathbf{x}$, matrices $\mathbf{U}_m, 1 \leq m \leq d-1$ and the output $\mathbf{Hx}$ of filtering procedure, which is about $O\big(\#\mathcal{N}_i \prod_{k=1}^{d}(L_k+1)\big) = O((\deg \mathcal{G}+1)\prod_{k=1}^{d}(L_k+1))$ loops. We observe that for the implementation of the proposed Algorithm 2 in a distributed network, the computational cost, memory requirement and communication expense for the data processing and communication subsystems equipped at each agent is **independent** on the size N of the network.

2.2 Inverse Filtering and Iterative Approximation Algorithm

Let $\mathbf{H}$ be an invertible graph filter on the graph $\mathcal{G}$. In some applications, such as signal denoising, inpainting, smoothing, reconstructing and semi-supervised learning [12, 20, 21, 27, 28, 30, 64, 65, 66], an inverse filtering procedure

$$\mathbf{x} = \mathbf{H}^{-1}\mathbf{b} \tag{2.8}$$

is involved. In this section, we select a graph filter $\mathbf{G}$ which provides an approximation to the inverse filter $\mathbf{H}^{-1}$, propose an iterative approximation algorithm with each iteration including filtering procedures associated with filters $\mathbf{H}$ and $\mathbf{G}$, and show that the proposed algorithm converges exponentially.

Denote the identity matrix by $\mathbf{I}$ and the spectral radius of a matrix $\mathbf{A}$ by $\rho(\mathbf{A})$. Take a graph filter $\mathbf{G}$ such that the spectral radius of $\mathbf{I}-\mathbf{HG}$ is strictly less than 1, i.e.,

$$\rho(\mathbf{I}-\mathbf{HG}) < 1. \tag{2.9}$$

By Gelfand's formula on spectral radius, the requirement (2.9) can be reformulated as

$$\rho(\mathbf{I}-\mathbf{HG}) = \lim_{n\to\infty} \|(\mathbf{I}-\mathbf{HG})^n\|_2^{1/n} < 1, \tag{2.10}$$

where $\|\mathbf{x}\|_2$ is Euclidean norm of a vector $\mathbf{x}$ and $\|\mathbf{A}\|_2 = \sup_{\|\mathbf{x}\|_2=1} \|\mathbf{A}\mathbf{x}\|_2$ is the operator norm of a matrix $\mathbf{A}$. By (2.10), we can rewrite the inverse filtering procedure (2.8) as

$$\mathbf{x} = \mathbf{G}\big(\mathbf{I} - (\mathbf{I} - \mathbf{H}\mathbf{G})\big)^{-1}\mathbf{b} = \mathbf{G}\sum_{n=0}^{\infty}(\mathbf{I} - \mathbf{H}\mathbf{G})^n\mathbf{b} \tag{2.11}$$

by applying Neumann series to $\mathbf{I} - \mathbf{H}\mathbf{G}$. Based on the above expansion, we propose the following iterative algorithm to implement the inverse filtering procedure (2.8):

$$\begin{cases} \mathbf{z}^{(m)} = \mathbf{G}\mathbf{e}^{(m-1)}, \\ \mathbf{e}^{(m)} = \mathbf{e}^{(m-1)} - \mathbf{H}\mathbf{z}^{(m)}, \\ \mathbf{x}^{(m)} = \mathbf{x}^{(m-1)} + \mathbf{z}^{(m)},\ m \geq 1, \end{cases} \tag{2.12}$$

with initials

$$\mathbf{e}^{(0)} = \mathbf{b} \ \text{ and } \ \mathbf{x}^{(0)} = \mathbf{0}. \tag{2.13}$$

Due to the approximation property (2.9) of the graph filter $\mathbf{G}$ to the inverse filter $\mathbf{H}^{-1}$, we call the above algorithm (2.12) and (2.13) as an *iterative approximation algorithm*. In the following theorem, we show that the requirement (2.9) for the approximation filter is a sufficient and necessary condition for the exponential convergence of the iterative approximation algorithm (2.12) and (2.13).

Theorem 2.2.1. Let $\mathbf{H}$ be an invertible graph filter and $\mathbf{G}$ be a graph filter. Then $\mathbf{G}$ satisfies (2.9) if and only if for any graph signal $\mathbf{b}$, the sequence $\mathbf{x}^{(m)}, m \geq 1$, in the iterative approximation algorithm (2.12) and (2.13) converges exponentially to $\mathbf{H}^{-1}\mathbf{b}$. Furthermore, for any $r \in (\rho(\mathbf{I} - \mathbf{H}\mathbf{G}), 1)$, there exists a positive constant C such that

$$\|\mathbf{x}^{(m)} - \mathbf{H}^{-1}\mathbf{b}\|_2 \leq C\|\mathbf{x}\|_2 r^m,\ m \geq 1. \tag{2.14}$$

Proof. First the sufficiency. Applying the first two equations in (2.12) gives

$$\mathbf{e}^{(m)} = (\mathbf{I} - \mathbf{HG})\mathbf{e}^{(m-1)},\ m \geq 1.$$

Applying the above expression repeatedly and using the initial in (2.13) yields

$$\mathbf{e}^{(m)} = (\mathbf{I} - \mathbf{HG})^m \mathbf{b},\ m \geq 0. \tag{2.15}$$

Combining (2.15) and the first and third equations in (2.12) gives

$$\mathbf{x}^{(m)} = \mathbf{x}^{(m-1)} + \mathbf{G}(\mathbf{I} - \mathbf{HG})^{m-1}\mathbf{b},\ m \geq 1.$$

Applying the above expression for $\mathbf{x}^{(m)}, m \geq 1$, repeatedly and using the initial in (2.13), we obtain

$$\mathbf{x}^{(m)} = \mathbf{G}\sum_{n=0}^{m-1}(\mathbf{I} - \mathbf{HG})^n \mathbf{b},\ m \geq 1. \tag{2.16}$$

By (2.10), there exists a positive constant C_0 for any $r \in (\rho(\mathbf{I} - \mathbf{HG}), 1)$ such that

$$\|(\mathbf{I} - \mathbf{HG})^n\|_2 \leq C_0 r^n,\ n \geq 1. \tag{2.17}$$

Combining (2.9), (2.11) and (2.16), we obtain

$$\|\mathbf{x}^{(m)} - \mathbf{x}\|_2 = \left\|\mathbf{G}\sum_{n=m}^{\infty}(\mathbf{I} - \mathbf{HG})^n \mathbf{b}\right\|_2. \tag{2.18}$$

From (2.17) and (2.18) it follows that

$$\|\mathbf{x}^{(m)} - \mathbf{x}\|_2 \le \|\mathbf{G}\|_2\|\mathbf{b}\|_2 \sum_{n=m}^{\infty} \|(\mathbf{I} - \mathbf{HG})^n\|_2$$
$$\le C_0\|\mathbf{G}\|_2\|\mathbf{H}\|_2\|\mathbf{x}\|_2 \sum_{n=m}^{\infty} r^n \le \frac{C_0\|\mathbf{G}\|_2\|\mathbf{H}\|_2}{1-r} r^m \|\mathbf{x}\|_2$$

for all $m \ge 1$. This proves the exponential convergence of $\mathbf{x}^{(m)}, m \ge 0$ to $\mathbf{H}^{-1}\mathbf{b}$.

Next the necessity. Suppose on the contrary that (2.9) does not hold. Then there exist an eigenvalue λ of $\mathbf{I} - \mathbf{HG}$ and an eigenvector $\mathbf{b}_0$ such that

$$|\lambda| \ge 1 \text{ and } (\mathbf{I} - \mathbf{HG})\mathbf{b}_0 = \lambda \mathbf{b}_0. \tag{2.19}$$

Then the sequence $\mathbf{x}^{(m)}, m \ge 1$, in the iterative approximation algorithm (2.12) and (2.13) with $\mathbf{b}$ replaced by $\mathbf{b}_0$ becomes

$$\mathbf{x}^{(m)} = \Big(\sum_{n=0}^{m-1} \lambda^n\Big)\mathbf{G}\mathbf{b}_0 = \begin{cases} \frac{\lambda^m - 1}{\lambda - 1}\mathbf{G}\mathbf{b}_0 & \text{if } \lambda \ne 1 \\ m\mathbf{G}\mathbf{b}_0 & \text{if } \lambda = 1 \end{cases}$$

by (2.16) and (2.19). Hence the sequence $\mathbf{x}^{(m)}, m \ge 1$, does not converge to the nonzero vector $\mathbf{H}^{-1}\mathbf{b}_0$, since it is identically zero if $\mathbf{G}\mathbf{b}_0 = \mathbf{0}$, and it diverges by the assumption that $|\lambda| \ge 1$ if $\mathbf{G}\mathbf{b}_0 \ne \mathbf{0}$. This contradicts to the exponential convergence assumption and completes the proof of the necessity.

□

By Theorem 2.2.1, the inverse filtering procedure (2.8) can be implemented by applying the iterative approximation algorithm (2.12) and (2.13) with the graph filter $\mathbf{G}$ being chosen so that (2.9) holds. The challenge is how to select the filter $\mathbf{G}$ to approximate the inverse filter $\mathbf{H}^{-1}$ appropri-

ately, which will be discussed in the next section when $\mathbf{H}$ is a polynomial filter of commutative graph shifts.

We finish this section with two remarks on the comparison among the gradient descent method [27], the autoregressive moving average (ARMA) method [21], and the proposed iterative approximation algorithm (2.12) and (2.13), cf. Remark 2.3.2.

Remark 2.2.2. For a positive definite graph filter $\mathbf{H}$, the inverse filtering procedure (2.8) can be implemented by the gradient descent method

$$\mathbf{x}^{(m)} = \mathbf{x}^{(m-1)} - \gamma(\mathbf{H}\mathbf{x}^{(m-1)} - \mathbf{b}), \;\; m \geq 1, \tag{2.20}$$

associated with the unconstrained optimization problem having the objective function $F(\mathbf{x}) = \mathbf{x}^T\mathbf{H}\mathbf{x} - \mathbf{x}^T\mathbf{b}$, where γ is an appropriate step length and $\mathbf{x}^T$ is the transpose of a vector $\mathbf{x}$. The above iterative method is shown in [27] to be convergent when $0 < \gamma < 2/\alpha_2$ and to have fastest convergence when $\gamma = 2/(\alpha_1 + \alpha_2)$, where α_1 and α_2 are the minimal and maximal eigenvalues of the matrix $\mathbf{H}$. By (2.20), we have that

$$\mathbf{x}^{(m)} = \gamma \sum_{n=0}^{m-1} (\mathbf{I} - \gamma\mathbf{H})^n \mathbf{b} + (\mathbf{I} - \gamma\mathbf{H})^m \mathbf{x}^{(0)}, \; m \geq 1. \tag{2.21}$$

By (2.21) and (2.16), the sequence $\mathbf{x}^{(m)}, m \geq 1$, in the gradient descent algorithm with zero initial **coincides** with the sequence in the iterative approximation algorithm (2.12) and (2.13) with $\mathbf{G} = \gamma\mathbf{I}$, in which the requirement (2.9) is met as the spectrum of $\mathbf{I} - \mathbf{H}\mathbf{G}$ is contained in $[1 - \gamma\alpha_2, 1 - \gamma\alpha_1] \subset (-1, 1)$ whenever $0 < \gamma < 2/\alpha_2$.

Remark 2.2.3. Let $\mathbf{S}$ be a graph shift and h be a polynomial of order L with its distinct nonzero roots $1/b_l$ satisfying

$$|b_l| \|\mathbf{S}\|_2 < 1, \; 1 \leq l \leq L. \tag{2.22}$$

Applying partial fraction decomposition to the rational function $1/h(t)$ gives $(h(t))^{-1} = \sum_{k=1}^{L} a_k(1-b_k t)^{-1}$ for some coefficients $a_k, 1 \leq k \leq L$. Then for the polynomial filter $\mathbf{H} = h(\mathbf{S})$, we can decompose the inverse filter $\mathbf{H}^{-1}$ into a family of elementary inverse filters $(\mathbf{I} - b_k\mathbf{S})^{-1}$,

$$\mathbf{H}^{-1} = \sum_{k=1}^{L} a_k(\mathbf{I} - b_k\mathbf{S})^{-1}.$$

Due to the above decomposition, the inverse filtering procedure (2.8) can be implemented as follows,

$$\mathbf{x} = \sum_{k=1}^{L} a_k(\mathbf{I} - b_k\mathbf{S})^{-1}\mathbf{b} =: \sum_{k=1}^{L} a_k\mathbf{x}_k. \tag{2.23}$$

The autoregressive moving average (ARMA) method has widely and popularly known in the time series model [21]. The ARMA can also be applied for the inverse filtering procedure (2.8), where it uses the decomposition (2.23) with the elementary inverse procedure $\mathbf{x}_k = (\mathbf{I} - b_k\mathbf{S})^{-1}\mathbf{b}$ implemented by the following iterative approach,

$$\mathbf{x}_k^{(m)} = b_k\mathbf{S}\mathbf{x}_k^{(m-1)} + \mathbf{b},\ m \geq 1$$

with initial $\mathbf{x}_k^{(0)} = \mathbf{0}$. We remark that the above approach is the same as the iterative approximation algorithm (2.12) and (2.13) with $\mathbf{H}$ and $\mathbf{G}$ replaced by $\mathbf{I} - b_k\mathbf{S}$ and $\mathbf{I}$ respectively. Moreover, in the above selection of the graph filters $\mathbf{H}$ and $\mathbf{G}$, the requirement (2.9) is met as it follows from (2.22) that

$$\rho(\mathbf{I} - \mathbf{H}\mathbf{G}) \leq \|\mathbf{I} - \mathbf{H}\mathbf{G}\|_2 \leq |b_k|\|\mathbf{S}\|_2 < 1 \tag{2.24}$$

for all $1 \leq k \leq L$. Applying (2.24), we see that the convergence rate to apply ARMA in the implementation of the inverse filtering procedure is $(\max_{1\leq k\leq L} |b_k|)\rho(\mathbf{S}) < 1$.

2.3 Iterative Polynomial Approximation Algorithms for Inverse Filtering

Let $\mathbf{S}_k = (S_k(i,j))_{i,j\in V}, 1 \le k \le d$, be commutative graph shifts on a connected, undirected and unweighted graph $\mathcal{G} = (V, E)$ of order N, Λ be the joint spectrum (A.2) of the shifts $\mathbf{S}_1, \ldots, \mathbf{S}_d$, and $\mathbf{H} = h(\mathbf{S}_1, \ldots, \mathbf{S}_d)$ be an invertible polynomial filter in (1.2). For polynomial graph filters of a single shift, there are several methods to implement the inverse filtering in a distributed network [12, 18, 20, 21, 27, 66]. In this section, we proposed two iterative algorithms to implement the inverse filtering associated with a polynomial graph filter of commutative graph shifts in a centralized facility with linear complexity and also in a distributed network with limited data processing and communication requirement for its agents. For the case that the joint spectrum Λ is fully known, we construct the polynomial interpolation approximation $\mathbf{G}_I$ and optimal polynomial approximations $\widetilde{\mathbf{G}}_L, L \ge 0$, to approximate the inverse filter $\mathbf{H}^{-1}$ in Subsection 2.3.1, and propose the iterative optimal polynomial approximation algorithm (2.31) to implement the inverse filtering procedure $\mathbf{b} \longmapsto \mathbf{H}^{-1}\mathbf{b}$, see Theorem 2.3.1. For a graph $\mathcal{G}$ of large order, it is often computationally expensive to find the joint spectrum Λ exactly. However, the graph shifts $\mathbf{S}_k, 1 \le k \le d$, in some engineering applications are symmetric and their spectrum sets are known being contained in some intervals [5, 7, 32, 33]. For instance, the normalized Laplacian matrix on a simple graph is symmetric and its spectrum is contained in $[0, 2]$. In Subsection 2.3.2, we consider the implementation of the inverse filtering procedure $\mathbf{b} \longmapsto \mathbf{H}^{-1}\mathbf{b}$ when the joint spectrum Λ of commutative shifts $\mathbf{S}_1, ..., \mathbf{S}_d$ is contained in a cubic. Based on multivariate Chebyshev polynomial approximation to the function h^{-1}, we introduce Chebyshev polynomial filters $\mathbf{G}_K, K \ge 0$, to approximate the inverse filter $\mathbf{H}^{-1}$, and propose the iterative Chebyshev polynomial approximation algorithm (2.40) to implement the inverse filtering procedure $\mathbf{b} \longmapsto \mathbf{H}^{-1}\mathbf{b}$, see Theorem 2.3.3. In addition to the exponential convergence, the proposed iterative optimal polynomial approximation algorithm and Chebyshev polynomial approximation algorithm can be implemented at vertex level in a distributed network, see Algorithms 3 and 4.

2.3.1 Polynomial Interpolation and Optimal Polynomial Approximation

Let $\mathbf{U}$ be the unitary matrix in (A.1) and denote its conjugate transpose by $\mathbf{U}^{\mathrm{H}}$. For polynomial filters $\mathbf{H} = h(\mathbf{S}_1, \ldots, \mathbf{S}_d)$ and $\mathbf{G} = g(\mathbf{S}_1, \ldots, \mathbf{S}_d)$, one may verify that $\mathbf{U}^{\mathrm{H}}(\mathbf{I} - \mathbf{HG})\mathbf{U}$ is an upper triangular matrix with diagonal entries $1 - h(\boldsymbol{\lambda}_i)g(\boldsymbol{\lambda}_i),\ \boldsymbol{\lambda}_i \in \Lambda$. Consequently, the requirement (2.9) for the polynomial graph filter $\mathbf{G}$ becomes

$$\rho(\mathbf{I} - \mathbf{GH}) = \sup_{\boldsymbol{\lambda}_i \in \Lambda} \left|1 - h(\boldsymbol{\lambda}_i)g(\boldsymbol{\lambda}_i)\right| < 1. \tag{2.25}$$

A necessary condition for the existence of a multivariate polynomial g such that (2.25) holds is that

$$h(\boldsymbol{\lambda}_i) \neq 0 \text{ for all } \boldsymbol{\lambda}_i \in \Lambda, \tag{2.26}$$

or equivalently the filter $\mathbf{H}$ is invertible. Conversely if (2.26) holds, $(\boldsymbol{\lambda}_i, 1/h(\boldsymbol{\lambda}_i)), 1 \leq i \leq N$, can be interpolated by a polynomial g_I of degree at most $N - 1$ [34], i.e.,

$$g_I(\boldsymbol{\lambda}_i) = 1/h(\boldsymbol{\lambda}_i),\ \boldsymbol{\lambda}_i \in \Lambda. \tag{2.27}$$

Take $\mathbf{G}_I = g_I(\mathbf{S}_1, \ldots, \mathbf{S}_d)$. Then all eigenvalues of $\mathbf{I} - \mathbf{G}_I\mathbf{H}$ are zero, $\rho(\mathbf{I} - \mathbf{G}_I\mathbf{H}) = 0$, and the iterative approximation algorithm (2.12) and (2.13) converges in at most N steps.

For $L \geq 0$, denote the set of all polynomials of degree at most L by $\mathcal{P}_L$. In practice, we may not use the interpolation polynomial g_I in (2.27), and hence the polynomial filter $\mathbf{G} = g_I(\mathbf{S}_1, \ldots, \mathbf{S}_d)$ in the iterative approximation algorithm (2.12) and (2.13), as it is of high degree in general. By (2.14), the convergence rate of the iterative approximation algorithm (2.12) and (2.13) depends on

the spectral radius in (2.25). Due to the above observation, we select $\tilde{g}_L \in \mathcal{P}_L$ such that

$$\tilde{g}_L = \arg\min_{g\in\mathcal{P}_L} \sup_{\boldsymbol{\lambda}_i\in\Lambda} |1 - g(\boldsymbol{\lambda}_i)h(\boldsymbol{\lambda}_i)|. \tag{2.28}$$

For a multivariate polynomial $g \in \mathcal{P}_L$, we write $g(\mathbf{t}) = \sum_{|\mathbf{k}|\le L} c_{\mathbf{k}}\mathbf{t}^{\mathbf{k}}$, where $|\mathbf{k}| = k_1 + \cdots + k_d$ and $\mathbf{t}^{\mathbf{k}} = t_1^{k_1}\cdots t_d^{k_d}$ for $\mathbf{t} = (t_1, \ldots, t_d)$ and $\mathbf{k} = (k_1, \ldots, k_d)$. Then for the case that all eigenvalues of $\mathbf{S}_k, 1 \le k \le d$, are real, i.e., $\Lambda \subset \mathbb{R}^d$, the minimization problem (2.28) can be reformulated as a linear programming,

$$\min\ s\ \text{ subject to }\ -(s-1)\mathbf{1} \le \mathbf{P}\mathbf{c} \le (s+1)\mathbf{1}, \tag{2.29}$$

where $\mathbf{P} = (h(\boldsymbol{\lambda}_i)\boldsymbol{\lambda}_i^{\mathbf{k}})_{1\le i\le N, |\mathbf{k}|\le L}$, $\mathbf{c} = (c_{\mathbf{k}})_{|\mathbf{k}|\le L}$ and $\mathbf{1}$ is the vector with all entries taking value 1.

Taking polynomial filters

$$\widetilde{\mathbf{G}}_L = \tilde{g}_L(\mathbf{S}_1, \ldots, \mathbf{S}_d),\ L \ge 0, \tag{2.30}$$

to approximate the inverse filter $\mathbf{H}^{-1}$, the iterative approximation algorithm (2.12) and (2.13) with the graph filter $\mathbf{G}$ replaced by $\widetilde{\mathbf{G}}_L$ becomes

$$\begin{cases} \mathbf{z}^{(m)} = \widetilde{\mathbf{G}}_L\mathbf{e}^{(m-1)}, \\ \mathbf{e}^{(m)} = \mathbf{e}^{(m-1)} - \mathbf{H}\mathbf{z}^{(m)}, \\ \mathbf{x}^{(m)} = \mathbf{x}^{(m-1)} + \mathbf{z}^{(m)},\ m \ge 1, \end{cases} \tag{2.31}$$

with initials $\mathbf{e}^{(0)}$ and $\mathbf{x}^{(0)}$ given in (2.13). We call the above iterative algorithm (2.31) by the *iterative optimal polynomial approximation algorithm*, or IOPA in abbreviation.

Presented in Algorithm 3 is the implementation of IOPA algorithm at the vertex level in a distributed network. In each iteration of Algorithm 3, each vertex/agent of the distributed network needs about $O((L+1)^{d-1} + \prod_{k=1}^{d-1}(L_k+1))$ steps containing data exchanging among adjacent ver-

Algorithm 3 The IOPA algorithm to implement the inverse filtering procedure $\mathbf{b} \longmapsto \mathbf{H}^{-1}\mathbf{b}$ at a vertex $i \in V$.

Inputs: Polynomial coefficients of $\mathbf{H}$ and $\widetilde{\mathbf{G}}_L$, entries $S_k(i,j), j \in \mathcal{N}_i$ in the i-th row of the shift $\mathbf{S}_k, 1 \le k \le d$, the value $b(i)$ of the input signal $\mathbf{b} = (b(i))_{i\in V}$ at the vertex i, and number M of iteration.

Initialization: Initial $e^{(0)}(i) = b(i)$, $x^{(0)}(i) = 0$ and $n = 0$.

Iteration:

For $m = 1, 2, \ldots, M$

Step 1: Use Algorithm 1 for $d = 1$ and Algorithm 2 for $d \ge 2$ to implement the filtering procedure $\mathbf{e}^{(m-1)} \longmapsto \mathbf{z}^{(m)} = \widetilde{\mathbf{G}}_L\mathbf{e}^{(m-1)}$ at the vertex i, and the output is the i-th entry $z^{(m)}(i)$ of the vector $\mathbf{z}^{(m)}$.

Step 2: Use Algorithm 1 for $d = 1$ and Algorithm 2 for $d \ge 2$ to implement the filtering procedure $\mathbf{z}^{(m)} \longmapsto \mathbf{w}^{(m)} = \mathbf{H}\mathbf{z}^{(m)}$ at the vertex i, and the output is the i-th entries $w^{(m)}(i)$ of the vector $\mathbf{w}^{(m)}$.

Step 3: Update i-th entries of $\mathbf{e}^{(m)}$ and $\mathbf{x}^{(m)}$ by $e^{(m)}(i) = e^{(m-1)}(i) - w^{(m)}(i)$ and $x^{(m)}(i) = x^{(m-1)}(i) + z^{(m)}(i)$ respectively.

end

Output: The approximated value $x(i) \approx x^{(M)}(i)$ is the output signal $\mathbf{H}^{-1}\mathbf{b} = (x(i))_{i\in V}$ at the vertex i.

tices and weighted sum of values at adjacent vertices in each iteration. The memory requirement for each vertex is about $O\big((\deg\mathcal{G} + L_k + 1)\prod_{k=1}^{d-1}(L_k+1) + (\det\mathcal{G}) + L + 1)(L+1)^{d-1})\big)$. The total operations of addition and multiplication in each iteration to implement the inverse filtering procedure $\mathbf{b} \longmapsto \mathbf{H}^{-1}\mathbf{b}$ via Algorithm 3 in a distributed network and procedure (2.31) in a central facility are almost the same, which are both about $O\big(N(\deg\mathcal{G}+1)(\prod_{k=1}^{d}(L_k+1) + (L+1)^d)\big)$.

By (2.28), we have

$$\rho(\mathbf{I} - \widetilde{\mathbf{G}}_L\mathbf{H}) = \sup_{\boldsymbol{\lambda}_i\in\Lambda} |1 - \tilde{g}_L(\boldsymbol{\lambda}_i)h(\boldsymbol{\lambda}_i)| \tag{2.32}$$

and

$$0 = \rho(\mathbf{I}-\mathbf{G}_I\mathbf{H}) = \rho(\mathbf{I}-\widetilde{\mathbf{G}}_{N-1}\mathbf{H}) \le \rho(\mathbf{I}-\widetilde{\mathbf{G}}_{L+1}\mathbf{H}) \le \rho(\mathbf{I}-\widetilde{\mathbf{G}}_L\mathbf{H}) \le \rho(\mathbf{I}-\widetilde{\mathbf{G}}_0\mathbf{H}),\ 0 \le L \le N-1. \tag{2.33}$$

In the following theorem, we show that the IOPA algorithm (2.31) converges exponentially when

L is large enough, see Section 2.4.1 for the numerical demonstration.

Theorem 2.3.1. Let $\mathbf{b}$ be a graph signal, $\mathbf{S}_1, ..., \mathbf{S}_d$ be commutative graph shifts, $\mathbf{H} = h(\mathbf{S}_1, \ldots, \mathbf{S}_d)$ be an invertible polynomial graph filter for some multivariate polynomial h, and let degree $L \geq 0$ be so chosen that

$$a_L := \sup_{\boldsymbol{\lambda}_i \in \Lambda} |1 - \tilde{g}_L(\boldsymbol{\lambda}_i) h(\boldsymbol{\lambda}_i)| < 1. \tag{2.34}$$

Then $\mathbf{x}^{(m)}, m \geq 1$, in the IOPA algorithm (2.31) converges exponentially to $\mathbf{H}^{-1}\mathbf{b}$. Moreover, for any $r \in (a_L, 1)$, there exists a positive constant C such that (2.14) holds.

Proof. The conclusion follows from (2.32), (2.34) and Theorem 2.2.1 with $\mathbf{G}$ replaced by $\widetilde{\mathbf{G}}_L$. □

Let L_0 be the minimal nonnegative integer so that $a_{L_0} < 1$. By (2.33) and Theorem 2.3.1, the inverse filtering procedure (2.8) can be implemented by applying the IOPA algorithm (2.31) with $L \geq L_0$ and the IOPA algorithm (2.31) converges faster when the higher degree L of the optimal polynomial $\tilde{g}_L$ is selected, see Section 2.4.1 for the numerical demonstration. However, the implementation of IOPA algorithm (2.31) with larger L at every agent/vertex in a distributed network has higher computational cost in each iteration and requires more memory for each agent/vertex, and also it takes higher computational cost to solve the the minimization problem (2.28) for larger L.

We finish this subsection with a remark on the IOPA algorithm (2.31) and the gradient descent method (2.20).

Remark 2.3.2. For the case that the graph filter $\mathbf{H}$ has its spectrum contained in $[\alpha_1, \alpha_2]$, the solution of the minimization problem (2.28) with $L = 0$ is given by $\tilde{g}_0 = 2/(\alpha_1 + \alpha_2)$, where $\alpha_1 = \min_{\boldsymbol{\lambda}_i \in \Lambda} h(\boldsymbol{\lambda}_i)$ and $\alpha_2 = \max_{\boldsymbol{\lambda}_i \in \Lambda} h(\boldsymbol{\lambda}_i)$ are the minimal and maximal eigenvalues of $\mathbf{H}$ respectively. Therefore, to implement the inverse filtering procedure (2.8), the gradient descent

method (2.20) with zero initial and optimal step length $\gamma = 2/(\alpha_1 + \alpha_2)$ is the **same** as the proposed IOPA algorithm (2.31) with $L = 0$, cf. Remark 2.2.2. By (2.33), we see that the IOPA algorithm with $L \geq 1$ has faster convergence than the gradient descent method does, at the cost of heavier computational cost at each iteration, see Table 2.1 and Figure 2.2 in Section 2.4.1 for numerical demonstrations.

2.3.2 Chebyshev Polynomial Approximation

In this subsection, we assume that commutative graph shifts $\mathbf{S}_1, ..., \mathbf{S}_d$ have their joint spectrum Λ contained in the cubic $[\boldsymbol{\mu}, \boldsymbol{\nu}] = [\mu_1, \nu_1] \times \cdots \times [\mu_d, \nu_d]$,

$$\boldsymbol{\lambda}_i \in [\boldsymbol{\mu}, \boldsymbol{\nu}] \text{ for all } \boldsymbol{\lambda}_i \in \Lambda, \tag{2.35}$$

and h be a multivariate polynomial satisfying

$$h(\mathbf{t}) \neq 0 \text{ for all } \mathbf{t} \in [\boldsymbol{\mu}, \boldsymbol{\nu}]. \tag{2.36}$$

Define Chebyshev polynomials $T_k, k \geq 0$, by

$$T_k(s) = \begin{cases} 1 & \text{if } k = 0, \\ s & \text{if } k = 1, \\ 2sT_{k-1}(s) - T_{k-2}(s) & \text{if } k \geq 2, \end{cases}$$

and shifted multivariate Chebyshev polynomials $\bar{T}_{\mathbf{k}}, \mathbf{k} = (k_1, \ldots, k_d) \in \mathbb{Z}_+^d$, on $[\boldsymbol{\mu}, \boldsymbol{\nu}]$ by

$$\bar{T}_{\mathbf{k}}(\mathbf{t}) = \prod_{i=1}^{d} T_{k_i}\Big(\frac{2t_i - \mu_i - \nu_i}{\nu_i - \mu_i}\Big), \ \ \mathbf{t} = (t_1, ..., t_d) \in [\boldsymbol{\mu}, \boldsymbol{\nu}].$$

By (2.36), $1/h$ is an analytic function on $[\boldsymbol{\mu}, \boldsymbol{\nu}]$, and hence it has Fourier expansion in term of shifted Chebyshev polynomials $\bar{T}_{\mathbf{k}}, \mathbf{k} \in \mathbb{Z}_+^d$,

$$\frac{1}{h(\mathbf{t})} = \sum_{\mathbf{k} \in \mathbb{Z}_+^d} c_{\mathbf{k}} \bar{T}_{\mathbf{k}}(\mathbf{t}),\ \mathbf{t} \in [\boldsymbol{\mu}, \boldsymbol{\nu}],$$

where

$$c_{\mathbf{k}} = \frac{2^{d-p(\mathbf{k})}}{\pi^d} \int_{[0,\pi]^d} \frac{\bar{T}_{\mathbf{k}}(t_1(\boldsymbol{\theta}), \ldots, t_d(\boldsymbol{\theta}))}{h(t_1(\boldsymbol{\theta}), \ldots, t_d(\boldsymbol{\theta}))} d\boldsymbol{\theta},\ \mathbf{k} \in \mathbb{Z}_+^d,$$

$p(\mathbf{k})$ is the number of zero components in $\mathbf{k} \in \mathbb{Z}_+^d$, and $t_i(\boldsymbol{\theta}) = \frac{\nu_i + \mu_i}{2} + \frac{\nu_i - \mu_i}{2} \cos(\theta_i), 1 \leq i \leq d$, for $\boldsymbol{\theta} = (\theta_1, ..., \theta_d)$. Define partial sum of the expansion (2.3.2) by

$$g_K(\mathbf{t}) = \sum_{|\mathbf{k}| \leq K} c_{\mathbf{k}} \bar{T}_{\mathbf{k}}(\mathbf{t}), \tag{2.37}$$

where $|\mathbf{k}| = \sum_{i=1}^d k_i$ for $\mathbf{k} = (k_1, ..., k_d)^T \in \mathbb{Z}_+^d$. Due to the analytic property of the polynomial h, the partial sum $g_K, K \geq 0$, converges to $1/h$ exponentially [35],

$$b_K := \sup_{\mathbf{t} \in [\boldsymbol{\mu}, \boldsymbol{\nu}]} |1 - h(\mathbf{t}) g_K(\mathbf{t})| \leq C r_0^K,\ K \geq 0, \tag{2.38}$$

for some positive constants $C \in (0, \infty)$ and $r_0 \in (0, 1)$.

Set

$$\mathbf{G}_K = g_K(\mathbf{S}_1, \ldots, \mathbf{S}_d),\ K \geq 0, \tag{2.39}$$

and call the iterative approximation algorithm (2.12) and (2.13) with the graph filter $\mathbf{G}$ replaced

Algorithm 4 The ICPA algorithm to implement the inverse filtering procedure $\mathbf{b} \longmapsto \mathbf{H}^{-1}\mathbf{b}$ at a vertex $i \in V$.

Inputs: Polynomial coefficients of polynomial filters $\mathbf{H}$ and $\mathbf{G}_K$, entries $S_k(i,j), j \in \mathcal{N}_i$ in the i-th row of the shifts $\mathbf{S}_k, 1 \le k \le d$, the value $b(i)$ of the input signal $\mathbf{b} = (b(i))_{i\in V}$ at the vertex i, and number M of iteration.

Initialization: Initial $e^{(0)}(i) = b(i)$, $x^{(0)}(i) = 0$ and $n = 0$.

Iteration: Use the iteration in Algorithm 3 except replacing $\widetilde{\mathbf{G}}_L$ by $\mathbf{G}_K$ in (2.39), and the output is $x^{(M)}(i)$.

Output: The approximated value $x(i) \approx x^{(M)}(i)$ is the output signal $\mathbf{H}^{-1}\mathbf{b} = (x(i))_{i\in V}$ at the vertex i.

by $\mathbf{G}_K$ by the *iterative Chebyshev polynomial approximation algorithm*, or ICPA in abbreviation,

$$\begin{cases} \mathbf{z}^{(m)} = \mathbf{G}_K\mathbf{e}^{(m-1)}, \\ \mathbf{e}^{(m)} = \mathbf{e}^{(m-1)} - \mathbf{H}\mathbf{z}^{(m)}, \\ \mathbf{x}^{(m)} = \mathbf{x}^{(m-1)} + \mathbf{z}^{(m)},\ m \ge 1, \end{cases} \tag{2.40}$$

with initials $\mathbf{e}^{(0)}$ and $\mathbf{x}^{(0)}$ given in (2.13), see Algorithm 4 to the distributed implementation at the vertex level.

From Algorithm 4, we can implement each iteration in the ICPA algorithm (2.40) at vertex level in about $O((K+1)^{d-1} + \prod_{k=1}^{d-1}(L_k+1))$ steps with each step containing data exchanging among adjacent vertices and weighted linear combination of values at adjacent vertices. The memory requirement for each agent in the distributed network is about $O\big((\deg\mathcal{G} + L_d + 1)\prod_{k=1}^{d-1}(L_k+1) + (\deg\mathcal{G} + K + 1)(K+1)^{d-1}\big)$. The total operations of addition and multiplication to implement each iteration of Algorithm 4 in a distributed network and to implement (2.40) in a central facility are almost the same, which are both about $O\big(N(\deg\mathcal{G}+1)(\prod_{k=1}^{d}(L_k+1) + (K+1)^d)\big)$.

In the following theorem, we show that the ICPA algorithm (2.40) converges exponentially, when the degree K is so chosen that (2.41) holds, see Section 2.4.1 for the demonstration.

Theorem 2.3.3. Let $\mathbf{S}_1, ..., \mathbf{S}_d$ be commutative graph shifts, $\mathbf{H}$ be a polynomial graph filter of the

graph shifts, $\mathbf{b}$ be a graph signal, and let degree $K \geq 0$ of Chebyshev polynomial approximation be so chosen that

$$b_K := \sup_{\mathbf{t}\in[\boldsymbol{\mu},\boldsymbol{\nu}]} |1 - h(\mathbf{t})g_K(\mathbf{t})| < 1. \tag{2.41}$$

Then $\mathbf{x}^{(m)}, m \geq 0$, in the ICPA algorithm (2.40) converges exponentially to $\mathbf{H}^{-1}\mathbf{b}$. Moreover for any $r \in (b_K, 1)$, there exists a positive constant C such that (2.14) holds.

Proof. Following the argument used in (2.32), one may verify that

$$\rho(\mathbf{I} - \mathbf{G}_K\mathbf{H}) = \sup_{\boldsymbol{\lambda}_i\in\Lambda} |1 - g_K(\boldsymbol{\lambda}_i)h(\boldsymbol{\lambda}_i)| \leq b_K, \tag{2.42}$$

where the inequality holds by (2.35) and the definition (2.38) of $b_K, K \geq 0$. Then the desired conclusion follows from (2.42) and Theorem 2.2.1 with $\mathbf{G}$ replaced by $\mathbf{G}_K$. □

By (2.38), an inverse filtering procedure (2.8) can be approximately implemented by the filter procedure $\mathbf{G}_K\mathbf{x}$ with large K, i.e., $\mathbf{H}^{-1}\mathbf{x} \approx \mathbf{G}_K\mathbf{x}$ for large K. The above implementation of the inverse filtering has been discussed in [12, 66] for the case that $\mathbf{H}$ is a polynomial graph filter of one shift, and it is known as the Chebyshev polynomial approximation algorithm (CPA). We remark that the approximation $\mathbf{G}_K\mathbf{x}$ in the CPA is the same as the first term $\mathbf{x}^{(1)}$ in the ICPA algorithm (2.40). To implement the inverse filtering with high accuracy, the CPA requires Chebyshev polynomial approximation of high degree, which means more integrals involved in coefficient calculations. On the other hand, we can select Chebyshev polynomial approximation of lower degree in the ICPA algorithm (2.40) to reach the same accuracy with few iterations. By Theorem 2.3.3, the ICPA algorithm (2.40) has exponential convergent rate b_K, which has limit zero as $K \to \infty$. This indicates that the ICPA algorithm converges faster for large K, however for each agent in a distributed network, its data processing system need more memory to store data and time to process data, and its communication system costs more for larger K too. Our simulation in the next

section confirms the above observation, see Table 2.1 and Figure 2.2 in Section 2.4.1.

2.4 Simulations

In this section, we demonstrate the performance of the proposed IOPA algorithm (2.31) and ICPA algorithm (2.40) on the implementation of the inverse filtering on circulant graphs (Section 2.4.1), on denoising time-varying graph signals on random geometric graphs (Section 2.4.2) and on denoising an hourly temperature data collected in the United States (Section 2.4.3). We also compare our algorithms with the gradient decent method (2.20) with zero initial (GD0) [27], and the autoregressive moving average (ARMA) algorithm (2.23) and (2.2.3) [21].

2.4.1 Iterative Approximation Algorithms on Circulant Graphs

Let $N \geq 1$ and $Q = \{q_1, \ldots, q_M\}$ be a set of integers ordered so that $1 \leq q_1 < \ldots < q_M < N/2$. The *circulant* graph $\mathcal{C}(N, Q)$ generated by Q has the vertex set $V_N = \{0, 1, \ldots, N-1\}$ and the edge set

$$E_N(Q) = \{(i, i \pm q \bmod N),\ i \in V_N, q \in Q\}, \tag{2.43}$$

where $a = b \bmod N$ if $(a-b)/N$ is an integer. Therefore for the circulant graph $\mathcal{C}(N, Q)$, $i \pm q_1 \bmod N, \ldots, i \pm q_M \bmod N$ are adjacent to the vertex $i \in V_N$. Circulant graphs are widely used in image processing [14, 36, 37, 38, 39]. In this section, we consider the circulant graph $\mathcal{C}(N, Q_0)$ generated by $Q_0 = \{1, 2, 5\}$, the input graph signal $\mathbf{x}$ with entries randomly selected in $[-1, 1]$, and the graph signal $\mathbf{b} = \mathbf{H}_1\mathbf{x}$ as the observation, where $h_1(t) = (9/4 - t)(3 + t)$ and $\mathbf{H}_1 = h_1(\mathbf{L}^{\rm sym}_{\mathcal{C}(N,Q_0)})$ is a polynomial graph filter of the symmetric normalized Laplacian $\mathbf{L}^{\rm sym}_{\mathcal{C}(N,Q_0)}$ on $\mathcal{C}(N, Q_0)$. We implement the inverse filtering $\mathbf{b} \longmapsto \mathbf{H}_1^{-1}\mathbf{b}$ through the IOPA algorithm (2.31) and ICPA algorithm (2.40) on the circulant graph $\mathcal{C}(N, Q_0)$. By Theorems 2.3.1 and 2.3.3, the

Table 2.1: Average relative iteration error over 1000 trials for the ARMA method, GD0 algorithm, and IOPA and ICPA algorithms with different degrees to implement the inverse filtering $\mathbf{b} \longmapsto \mathbf{H}_1^{-1}\mathbf{b}$ on the circulant graph $\mathcal{C}(1000, Q_0)$.

AE / Alg. / m	ARMA	GD0	ICPA0	ICPA1	ICPA2	IOPA1	ICPA3	IOPA2	IOPA3
1	0.3259	0.2350	0.5686	0.4494	0.1860	0.1545	0.0979	0.0365	0.0167
2	0.2583	0.0856	0.4318	0.2191	0.0412	0.0266	0.0113	0.0019	0.0003
3	0.1423	0.0349	0.3752	0.1103	0.0098	0.0047	0.0014	0.0001	0.0000
4	0.1098	0.0147	0.3521	0.0566	0.0024	0.0008	0.0002	0.0000	0.0000
5	0.0718	0.0063	0.3441	0.0295	0.0006	0.0002	0.0000	0.0000	0.0000
7	0.0381	0.0012	0.3460	0.0082	0.0000	0.0000	0.0000	0.0000	0.0000
9	0.0207	0.0002	0.3577	0.0024	0.0000	0.0000	0.0000	0.0000	0.0000
11	0.0113	0.0000	0.3743	0.0007	0.0000	0.0000	0.0000	0.0000	0.0000
14	0.0047	0.0000	0.4061	0.0001	0.0000	0.0000	0.0000	0.0000	0.0000
17	0.0019	0.0000	0.4451	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
20	0.0008	0.0000	0.4913	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

IOPA algorithm with $L \geq 0$ and the ICPA algorithm with $K \geq 1$ converge, and we denote those algorithms by IOPAL and ICPAK for abbreviation. Notice that the filter $\mathbf{H}_1$ is positive definite, and

$$\frac{1}{h_1(t)} = \frac{4/21}{9/4 - t} + \frac{4/21}{3 + t}$$

meets the requirement (2.22) for the ARMA. For the circulant graph $\mathcal{C}(N, Q_0)$ with $N = 1000$, we also implement the inverse filtering $\mathbf{b} \longmapsto \mathbf{H}_1^{-1}\mathbf{b}$ by the gradient descent method with zero initial, GD0 in abbreviation, with the optimal step length $\gamma = 2/(6.7500 + 2.5588)$, and the ARMA method, where 2.5588 and 6.7500 are the minimal and maximal eigenvalues for $\mathbf{H}_1$ respectively.

Set the relative iteration error

$$E(m, \mathbf{x}) = \|\mathbf{x}^{(m)} - \mathbf{x}\|_2 / \|\mathbf{x}\|_2, \ m \geq 1,$$

where $\mathbf{x}^{(m)}, m \geq 1$, are the output at m-th iteration. Shown in Table 2.1 are the comparisons of the ARMA algorithm, the GD0 algorithm, and IOPAL and ICPAK algorithms regard to the average of the relative iteration error for implementing the inverse filtering on the circulant graph $\mathcal{C}(1000, Q_0)$ over 1000 trials, where $0 \leq L, K \leq 3$. This confirms that exponential convergence and applicability of the inverse filtering procedure $\mathbf{b} \longmapsto \mathbf{H}_1^{-1}\mathbf{b}$ of IOPAL, $0 \leq L \leq 5$ and ICPAK, $1 \leq K \leq 5$ on the circulant graph $\mathcal{C}(1000, Q_0)$. The average exponential convergence rates of IOPAL, $0 \leq L \leq 5$, over 1000 trials are $0.4401, 0.1820, 0.0593, 0.0208, 0.0067, 0.0023$ respectively, which is close to the theoretical bound $a_L = 0.4502, 0.1852$, 0.0612, $0.0212, 0.0072, 0.0025$ for $0 \leq L \leq 5$, see (2.34) in Theorem 2.3.1. Similarly, the average exponential convergence rates of ICPAK, $1 \leq K \leq 5$, are $0.5485, 0.2804, 0.1459, 0.0685, 0.0334$ respectively, while their theoretical estimate $b_K, 1 \leq K \leq 5$, in (2.41) of Theorem 2.3.3 are $0.5837, 0.2924, 0.1467, 0.0728, 0.0367$ respectively. By the third column in Table 2.1, we see that the ICPA0 does not yield the desired inverse filtering result. The reason for the divergence is that the theoretical bound $b_0 = 1.0463$ in (2.41) is strictly larger than one. From Table 2.1, we observe that the IOPAL algorithms with higher degree L (resp. the ICPAK with higher degree K) have faster convergence, and the IOPAL algorithm outperforms the ICPAK algorithm when the same degree $L = K$ is selected. Comparing with the ARMA algorithm and the GD0 algorithm, we observe that the proposed IOPAL algorithms with $L \geq 1$ and ICPAK algorithms with $K \geq 2$ have faster convergence, while the GD0=IOPA0 algorithm outperforms the ICPAK when $K = 1$.

We also apply ARMA, GD0, and IOPAL and ICPAK with $1 \leq L, K \leq 5$ to implement inverse filtering procedure on the circulant graph $\mathcal{C}(N, Q_0)$ with $N \geq 100$. All experiments were performed on MATLAB R2017b, running on a DELL T7910 workstation with two Intel Core E5-2630 v4 CPUs (2.20 GHz) and 32GB memory. From the simulations, we observe that the exponential convergence rate r for the proposed algorithms is almost independent on $N \geq 100$, see Figure 2.1, and the number of iterations to ensure the relative iteration error $E(m, \mathbf{x}) \leq 10^{-3}$ are

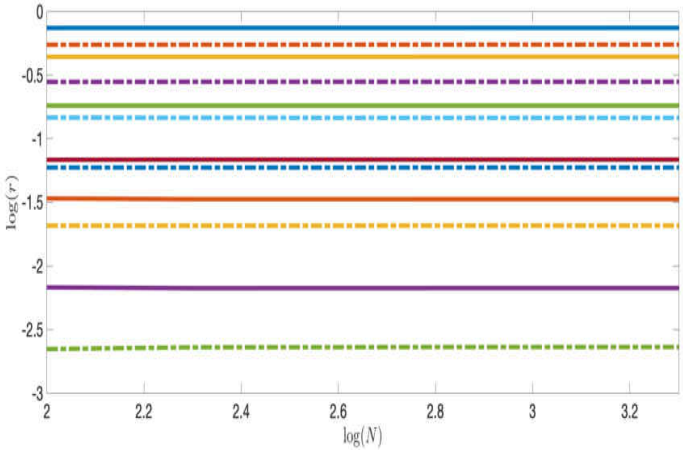

Figure 2.1: Plotted from top to bottom are the average exponential convergence rate r in the logarithmic scale over 1000 trials by ARMA, ICPA1, GD0, ICPA2, IOPA1, ICPA3, ICPA4, IOPA2, ICPA5, IOPA3, IOPA4, IOPA5 to implement the inverse filtering on circulant graphs $\mathcal{C}(N, Q_0)$ with $100 \leq N \leq 2000$, respectively.

$20, 8, 11, 5, 4, 4, 3, 3, 2, 2, 2, 2$ for ARMA, GD0, ICPA1, ICPA2, IOPA1, ICPA3, ICPA4, IOPA2, ICPA5, IOPA3, IOPA4, IOPA5 respectively. Shown in Figure 2.2 is the average running time T in the logarithmic scale over 1000 trials, where the running time T is measured in seconds to ensure the relative iteration error $E(m, \mathbf{x}) \leq 10^{-3}$. From our simulations, we see that there is a complicated trade-off between the convergence rate and the running time to apply our proposed algorithms, ARMA and GD0 for the implementation of an inverse filtering procedure.

2.4.2 Denoising Time-Varying Signals

In this section, we consider denoising noisy sampling data

$$\mathbf{b}_i = \mathbf{x}(t_i) + \boldsymbol{\eta}_i, \ 1 \leq i \leq M, \tag{2.44}$$

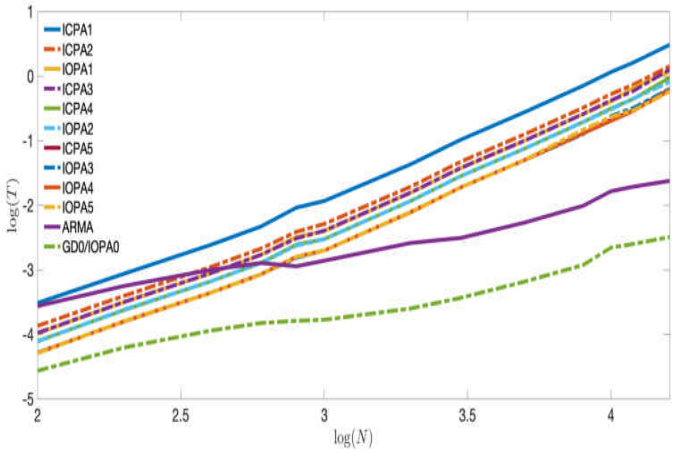

Figure 2.2: Plotted are the average of total running time T in the logarithmic scale for the GD0, ARMA and the IOPAL and ICPAK algorithms with $1 \leq L, K \leq 5$ to implement the inverse filtering on circulant graphs $\mathcal{C}(N, Q_0)$ with $100 \leq N \leq 16000$.

of some time-varying graph signal $\mathbf{x}(t)$ governed by a differential equation

$$\mathbf{x}''(t) = \mathbf{P}\mathbf{x}(t), \tag{2.45}$$

where $\boldsymbol{\eta}_i, 1 \leq i \leq M$, are noises with noise level $\eta = \max_{1 \leq i \leq M} \|\boldsymbol{\eta}_i\|_\infty$, the sampling procedure is taken uniformly at $t_i = t_1 + (i-1)\delta, 1 \leq i \leq M$, with uniform sampling gap $\delta > 0$, and $\mathbf{P}$ is a graph filter with small geodesic-width.

Discretizing the differential equation (2.45) gives

$$\delta^{-2}\big(\mathbf{x}(t_{i+1}) + \mathbf{x}(t_{i-1}) - 2\mathbf{x}(t_i)\big) \approx \mathbf{P}\mathbf{x}(t_i), \tag{2.46}$$

where $i = 1, \ldots, M$. Applying the trivial extension $\mathbf{x}(t_0) = \mathbf{x}(t_1)$ and $\mathbf{x}(t_{M+1}) = \mathbf{x}(t_M)$ around the boundary, we can reformulate (2.46) in a recurrence relation,

$$\mathbf{x}(t_i) \approx (2\mathbf{I} + \delta^2\mathbf{P})\mathbf{x}(t_{i-1}) - \mathbf{x}(t_{i-2}), 2 \leq i \leq M, \tag{2.47}$$

with $\mathbf{x}(t_0) = \mathbf{x}(t_1)$. Let $\mathcal{T} = (T, F)$ be the line graph with the vertex set $T = \{t_1, \cdots, t_M\}$ and edge set $F = \{(t_1, t_2), \ldots, (t_{M-1}, t_M)\} \cup \{(t_M, t_{M-1}), \ldots, (t_2, t_1)\}$. Denote Kronecker product of two matrices $\mathbf{A}$ and $\mathbf{B}$ by $\mathbf{A} \otimes \mathbf{B}$, and the Laplacian matrix of the line graph $\mathcal{T}$ with vertices $\{t_1, \ldots, t_M\}$ by $\mathbf{L}_{\mathcal{T}}$. Then we can reformulate the recurrence relation (2.47) in the matrix form

$$(\delta^{-2}\mathbf{L}_{\mathcal{T}} \otimes \mathbf{I} + \mathbf{I} \otimes \mathbf{P})\mathbf{X} \approx \mathbf{0}, \tag{2.48}$$

where $\mathbf{X}$ is the vectorization of $\mathbf{x}(t_1), \ldots, \mathbf{x}(t_M)$. In most of applications [11, 14, 40, 41], the time-varying signal $\mathbf{x}(t)$ at every moment t has certain smoothness in the vertex domain, which is usually described by

$$(\mathbf{x}(t_i))^T \mathbf{L}_{\mathcal{G}}^{\mathrm{sym}} \mathbf{x}(t_i) \approx 0,\ 1 \leq i \leq M, \tag{2.49}$$

where $\mathbf{L}_{\mathcal{G}}^{\mathrm{sym}}$ is the symmetric normalized Laplacian on the connected, undirected and unweighted graph $\mathcal{G} = (V, E)$. Based on the observations (2.48) and (2.49), we propose the following Tikhonov regularization approach,

$$\widehat{\mathbf{X}} := \arg\min_{\mathbf{Y}} \|\mathbf{Y} - \mathbf{B}\|_2^2 + \alpha \mathbf{Y}^T (\mathbf{I} \otimes \mathbf{L}_{\mathcal{G}}^{\mathrm{sym}})\mathbf{Y} + \beta \mathbf{Y}^T (\delta^{-2}\mathbf{L}_{\mathcal{T}} \otimes \mathbf{I} + \mathbf{I} \otimes \mathbf{P})\mathbf{Y}, \tag{2.50}$$

where $\mathbf{B}$ is the vectorization of the observed noisy data $\mathbf{b}_1, \ldots, \mathbf{b}_M$, and α, β are penalty constants in the vertex and "temporal" domains to be appropriately chosen [24].

Set

$$\mathbf{D}_{\alpha,\beta} = \mathbf{I} + \alpha \mathbf{I} \otimes \mathbf{L}_{\mathcal{G}}^{\mathrm{sym}} + \beta(\delta^{-2}\mathbf{L}_{\mathcal{T}} \otimes \mathbf{I} + \mathbf{I} \otimes \mathbf{P}),\ \alpha, \beta \geq 0.$$

The minimization problem (2.50) has an explicit solution

$$\widehat{\mathbf{X}} = (\mathbf{D}_{\alpha,\beta})^{-1}\mathbf{B}, \tag{2.51}$$

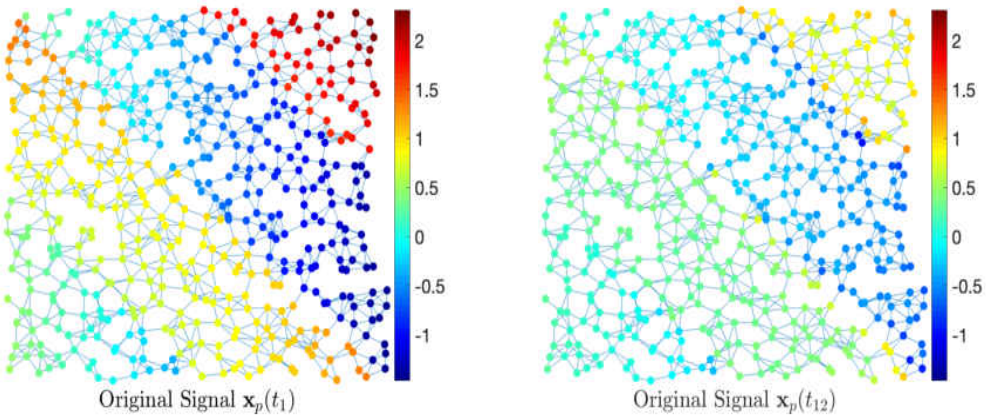

Figure 2.3: Presented on the left and right are the first snapshot $\mathbf{x}_p(t_1)$ and the middle snapshot $\mathbf{x}_p(t_{12})$ of a time-varying signal $\mathbf{x}_p(t_m), 1 \leq m \leq 24$, on the random geometric graph $\mathcal{G}_{512}$ respectively, where the qualities $(\mathbf{x}_p(t_m))^T \mathbf{L}^{\rm sym}_{\mathcal{G}_{512}} \mathbf{x}_p(t_m)$ to measure smoothness of $\mathbf{x}_p(t_m)$ in the vertex domain are 84.1992 and 42.4746 for $m = 1, 12$ respectively.

when $\mathbf{I} + \alpha \mathbf{L}^{\rm sym}_{\mathcal{G}} + \beta \mathbf{P}$ is positive definite. Set $\mathbf{S}_1 = \mathbf{I} \otimes \mathbf{L}^{\rm sym}_{\mathcal{G}}$ and $\mathbf{S}_2 = \frac{1}{2}\mathbf{L}_{\mathcal{T}} \otimes \mathbf{I}$. One may verify that $\mathbf{S}_1$ and $\mathbf{S}_2$ are commutative graph shifts on the Cartesian product graph $\mathcal{T} \times \mathcal{G}$, see Proposition A.2.2, and their joint spectrum is contained in $[0, 2]^2$. Therefore for the case that $\mathbf{P} = p(\mathbf{L}^{\rm sym}_{\mathcal{G}})$ for some polynomial p, $\mathbf{D}_{\alpha,\beta} = h_{\alpha,\beta}(\mathbf{S}_1, \mathbf{S}_2)$ is a polynomial graph filter of commutative graph filters $\mathbf{S}_1$ and $\mathbf{S}_2$, where $h_{\alpha,\beta}(t_1, t_2) = 1 + \alpha t_1 + \beta p(t_1) + 2\beta\delta^{-2} t_2$. Moreover, one may verify that $\mathbf{D}_{\alpha,\beta}$ is positive definite if

$$h_{\alpha,\beta}(t_1, t_2) > 0, \quad 0 \leq t_1, t_2 \leq 2,$$

which is satisfied if $1 + \beta p(t_1) > 0$ for all $0 \leq t_1 \leq 2$. Hence we may use the IOPA algorithm (2.31) and the ICPA algorithm (2.40) with the polynomial filter $\mathbf{H}$ being replaced by $\mathbf{D}_{\alpha,\beta}$ to implement the denoising procedure (2.51). By the exponential convergence of the proposed algorithms, we may use their outputs at m-th iteration with large m as denoised time-varying signals.

Let $\mathcal{G}_{512}$ be the random geometric graph reproduced by the GSPToolbox, which has 512 vertices randomly deployed in the region $[0, 1]^2$ and an edge existing between two vertices if their physical

distance is not larger than $\sqrt{2/512} = 1/16$ [42, 64]. Denote the symmetric normalized Laplacian matrix on $\mathcal{G}_{512}$ by $\mathbf{L}_{\mathcal{G}_{512}}^{\rm sym}$ and the coordinates of a vertex i in $\mathcal{G}_{512}$ by (i_x, i_y). For the simulations in this section, the time-varying signal $\mathbf{x}(t_m), 1 \le m \le M$, is given in (2.46), where $M = 24, \delta = 0.1$, the governing filter is given by $\mathbf{P} = -\mathbf{I} + \mathbf{L}_{\mathcal{G}_{512}}^{\rm sym}/2$, and the initial graph signal $\mathbf{x}_p(t_1)$ is a blockwise polynomial consisting of four strips and imposing $(0.5 - 2i_x)$ on the first and third diagonal strips and $(0.5 + i_x^2 + i_y^2)$ on the second and fourth strips respectively [64]. Shown in Figure 2.3 are two snapshots of the above time-varying graph signal.

Appropriate selection of the penalty constants α, β in the vertex and temporal domains are crucial to have a satisfactory denoising performance. In the simulations, we let noise entries of $\boldsymbol{\eta}_i, 1 \le i \le 24$ in (2.45), be i.i.d. variables uniformly selected in the range $[-\eta, \eta]$, and we take

$$\alpha = \frac{\mathbb{E}\|\mathbf{B} - \mathbf{X}\|_2^2}{\mathbb{E}\big(\mathbf{B}^T(\mathbf{I} \otimes \mathbf{L}_{\mathcal{G}_{512}}^{\rm sym})\mathbf{B}\big)} = \frac{MN\eta^2/3}{\mathbf{X}^T(\mathbf{I} \otimes \mathbf{L}_{\mathcal{G}_{512}}^{\rm sym})\mathbf{X} + MN\eta^2/3} \approx \frac{\eta^2}{0.2306 + \eta^2}, \tag{2.52}$$

and

$$\beta = \frac{\mathbb{E}\|\mathbf{B} - \mathbf{X}\|_2^2}{2\mathbb{E}\big(\mathbf{B}^T(\delta^{-2}\mathbf{L}_{\mathcal{T}} \otimes \mathbf{I} + \mathbf{I} \otimes \mathbf{P})\mathbf{B}\big)} \approx 0.0026 \tag{2.53}$$

to balance the fidelity term and the regularization terms on the vertex and temporal domains in (2.50).

We use the IOPA algorithm (2.31) with $L = 1$, the ICPA algorithm (2.40) with $K = 1$ and the gradient descent method (2.20) with zero initial to implement the inverse filter procedure $\mathbf{B} \longmapsto \widehat{\mathbf{X}} = \mathbf{D}_{\alpha,\beta}^{-1}\mathbf{B}$, denoted by IOPA1$(\alpha, \beta)$, ICPA1$(\alpha, \beta)$ and GD0(α, β) respectively. Let $\widehat{\mathbf{X}}^{(m)}, m \ge 1$, be the outputs of either the IOPA1(α, β) algorithm, or the ICPA1(α, β) algorithm, or the GD0(α, β) method at m-th iteration. To measure the denoising performance of our approaches, we define the input signal-to-noise ratio

$$\text{ISNR} = -20\log_{10}\|\mathbf{B} - \mathbf{X}\|_2/\|\mathbf{X}\|_2,$$

and the output signal-to-noise ratio

$$\mathrm{SNR}(m) = -20\log_{10}\|\widehat{\mathbf{X}}^{(m)} - \mathbf{X}\|_2/\|\mathbf{X}\|_2,\ m \geq 1,$$

and

$$\mathrm{SNR}(\infty) = -20\log_{10}\|\widehat{\mathbf{X}} - \mathbf{X}\|_2/\|\mathbf{X}\|_2.$$

Presented in Table 2.2 are the average over 1000 trials of ISNR and $\mathrm{SNR}(m), m = 1, 2, 4, 6, \infty$. From Table 2.2, we observe that the denoising procedure $\mathbf{B} \longmapsto \widehat{\mathbf{X}} = \mathbf{D}_{\alpha,\beta}^{-1}\mathbf{B}$ via Tikhonov regularization (2.50) on the temporal-vertex domain can improve the signal-to-noise ratio in the range from 2dBs to 5dBs, depending on the noise level η. Also we see that the denoising procedure $\mathbf{B} \longmapsto \widehat{\mathbf{X}}^{(m)}$ via the output of the m-th iteration in IOPA1(α, β) algorithm with $m \geq 2$, the GD0(α, β) method and the ICPA1(α, β) algorithm with $m \geq 4$ have similar denoising performance. Due to the correlation of time-varying signals across the joint temporal-vertex domains, it is expected that the Tikhonov regularization (2.50) on the temporal-vertex domain has better denoising performance than Tikhonov regularization either only on the vertex domain (i.e., $\beta = 0$ in (2.50)) or only on the temporal domain (i.e., $\alpha = 0$ in (2.50)) do. The above performance expectation is confirmed in Table 2.2. We remark that denoising approach via the Tikhonov regularization on the temporal-vertex domain is an inverse filtering procedure of a polynomial graph filter of **two** shifts, while the one either on the vertex domain or on the temporal domain only is an inverse filtering procedure of a polynomial graph filter of **one** shift.

2.4.3 Denoising an Hourly Temperature Dataset

In the section, we consider denoising the hourly temperature dataset collected at 218 locations in the United States on August 1st, 2010, measured in Fahrenheit [43]. The above real-world dataset is of size 218×24, and it can be modelled as a time-varying signal $\mathbf{w}(i), 1 \leq i \leq 24$, on the

product graph $\mathcal{C} \times \mathcal{W}$, where $\mathcal{C} := \mathcal{C}(24, \{1\})$ is the circulant graph with 24 vertices and generator $\{1\}$, and $\mathcal{W}$ is the undirected graph with 218 locations as vertices and edges constructed by the 5 nearest neighboring algorithm.

Given noisy temperature data

$$\widetilde{\mathbf{w}}_i = \mathbf{w}_i + \boldsymbol{\eta}_i,\ i = 1, \ldots, 24,$$

we propose the following denoising approach,

$$\widehat{\mathbf{W}} := \arg\min_{\mathbf{Z}} \|\mathbf{Z} - \widetilde{\mathbf{W}}\|_2^2 + \tilde{\alpha}\mathbf{Z}^T(\mathbf{I} \otimes \mathbf{L}_{\mathcal{W}}^{\mathrm{sym}})\mathbf{Z} + \tilde{\beta}\mathbf{Z}^T(\mathbf{L}_{\mathcal{C}}^{\mathrm{sym}} \otimes \mathbf{I})\mathbf{Z}, \tag{2.54}$$

where $\widetilde{\mathbf{W}}$ is the vectorization of the noisy temperature data $\widetilde{\mathbf{w}}_1, \ldots, \widetilde{\mathbf{w}}_{24}$ with noises $\boldsymbol{\eta}_i, 1 \leq i \leq 24$ in (2.45) having their components randomly selected in $[-\eta, \eta]$ in a uniform distribution, $\mathbf{L}_{\mathcal{W}}^{\mathrm{sym}}$ and $\mathbf{L}_{\mathcal{C}}^{\mathrm{sym}}$ are normalized Laplacian matrices on the graph $\mathcal{W}$ and $\mathcal{C}$ respectively, and $\tilde{\alpha}, \tilde{\beta} \geq 0$ are penalty constants in the vertex and temporal domains to be appropriately selected.

Set $\tilde{\mathbf{S}}_1 = \mathbf{I} \otimes \mathbf{L}_{\mathcal{W}}^{\mathrm{sym}}$, $\tilde{\mathbf{S}}_2 = \mathbf{L}_{\mathcal{C}}^{\mathrm{sym}} \otimes \mathbf{I}$ and $\mathbf{F}_{\tilde{\alpha},\tilde{\beta}} = \mathbf{I} + \tilde{\alpha}\tilde{\mathbf{S}}_1 + \tilde{\beta}\tilde{\mathbf{S}}_2,\ \tilde{\alpha}, \tilde{\beta} \geq 0$. One may verify that the explicit solution of the minimization problem (2.54) is given by $\widehat{\mathbf{W}} = (\mathbf{F}_{\tilde{\alpha},\tilde{\beta}})^{-1}\widetilde{\mathbf{W}}$, and the proposed approach to denoise the temperature dataset becomes an inverse filtering procedure (2.8) with $\mathbf{H}$ and $\mathbf{b}$ replaced by $\mathbf{F}_{\tilde{\alpha},\tilde{\beta}}$ and $\widetilde{\mathbf{W}}$ respectively. In absence of notation, we still denote the IOPA algorithm (2.31) with $L = 1$, the ICPA algorithm (2.40) with $K = 1$ and the gradient descent method (2.20) with initial zero to implement the inverse filter procedure $\widetilde{\mathbf{W}} \longmapsto \mathbf{F}_{\tilde{\alpha},\tilde{\beta}}^{-1}\widetilde{\mathbf{W}}$ by IOPA1$(\tilde{\alpha}, \tilde{\beta})$, ICPA1$(\tilde{\alpha}, \tilde{\beta})$ and GD0$(\tilde{\alpha}, \tilde{\beta})$ respectively.

In our simulations, we take

$$\tilde{\alpha} = \frac{\mathbb{E}\|\mathbf{Z} - \widetilde{\mathbf{W}}\|_2^2}{\mathbb{E}\big(\widetilde{\mathbf{W}}^T\tilde{\mathbf{S}}_1\widetilde{\mathbf{W}}\big)} = \frac{1744\eta^2}{\mathbf{W}^T\tilde{\mathbf{S}}_1\mathbf{W} + 1744\eta^2}$$

and

$$\tilde{\beta} = \frac{\mathbb{E}\|\mathbf{Z} - \widetilde{\mathbf{W}}\|_2^2}{\mathbb{E}(\widetilde{\mathbf{W}}^T \tilde{\mathbf{S}}_2 \widetilde{\mathbf{W}})} = \frac{1744\eta^2}{\mathbf{W}^T \tilde{\mathbf{S}}_2 \mathbf{W} + 1744\eta^2}$$

to balance three terms in the regularization approach (2.54). Presented in Table 2.3 are the average over 1000 trials of the input signal-to-noise ratio ISNR and the output signal-to-noise ratio

$$\mathrm{SNR}(m) = -20 \log_{10} \frac{\|\widehat{\mathbf{W}}^{(m)} - \mathbf{W}\|_2}{\|\mathbf{W}\|_2},\ m \geq 1,$$

which are used to measure the denoising performance of the IOPA1$(\tilde{\alpha}, \tilde{\beta})$, ICPA1$(\tilde{\alpha}, \tilde{\beta})$ and GD0$(\tilde{\alpha}, \tilde{\beta})$ at the mth iteration, where $\widehat{\mathbf{W}}^{(\infty)} := \widehat{\mathbf{W}}$ and $\widehat{\mathbf{W}}^{(m)}, m \geq 1$, are outputs of the IOPA1$(\tilde{\alpha}, \tilde{\beta})$ algorithm, or the ICPA1$(\tilde{\alpha}, \tilde{\beta})$, or the GD0$(\tilde{\alpha}, \tilde{\beta})$ at m-th iteration. From Table 2.3, we see that the Tikhonov regularization on the temporal-vertex domain has better performance on denoising the hourly temperature dataset than the Tikhonov regularization only either on the vertex domain (i.e. $\tilde{\beta} = 0$) or on the temporal domain (i.e. $\tilde{\alpha} = 0$) do.

2.5 Conclusions

Polynomial graph filters of multiple shifts are preferable for denoising and extracting features for multidimensional graph signals, such as video or time-varying signals. Some Tikhonov regularization approaches on the temporal-vertex domain to denoise a time-varying signal can be reformulated as an inverse filtering procedure for a polynomial graph filter of two shifts which represent the features on the temporal and vertex domain respectively. Two exponentially convergent iterative algorithms are introduced for the inverse filtering procedure of a polynomial graph filter, and each iteration of the proposed algorithms can be implemented in a distributed network, where each vertex is equipped with systems for limited data storage, computation power and data exchanging facility to its adjacent vertices. The proposed iterative algorithms are demonstrated to implement the inverse filtering procedure effectively and to have satisfactory performance on de-

noising multidimensional graph signals. Future works will concentrate on the design methodology of polynomial filters of multiple graph shifts and their inverses with certain spectral characteristic.

Table 2.2: The average of the signal-to-noise ratio $\mathrm{SNR}(m), m = 1, 2, 4, 6, \infty$ for the noise level $\eta = 3/4, 1/2, 1/4$ over 1000 trials, where penalty constants α and β are given in (2.52) and (2.53) respectively.

SNR \ m / Alg.	1	2	4	6	∞
η=3/4, ISNR= 3.3755					
IOPA1(α, 0)	6.5777	6.8047	6.7927	6.7926	6.7926
IOPA1(0, β)	6.0597	6.0907	6.0735	6.0735	6.0735
IOPA1(α, β)	7.4797	8.5330	8.4942	8.4931	8.4930
ICPA1(α, 0)	6.4581	6.8169	6.7928	6.7926	6.7926
ICPA1(0, β)	6.0433	6.0899	6.0735	6.0735	6.0735
ICPA1(α, β)	7.4036	8.4602	8.4924	8.4930	8.4930
GD0(α, 0)	4.9399	6.7283	6.8062	6.7943	6.7926
GD0(0, β)	5.0027	6.3873	6.1225	6.0787	6.0735
GD0(α, β)	4.1778	6.9998	8.3432	8.4750	8.4930
η=1/2, ISNR=6.8975					
IOPA1(α, 0)	9.2211	9.3576	9.3544	9.3544	9.3544
IOPA1(0, β)	9.4981	9.6116	9.5949	9.5949	9.5949
IOPA1(α, β)	10.0425	11.0678	11.0624	11.0620	11.0620
ICPA1(α, 0)	9.1525	9.3617	9.3544	9.3544	9.3544
ICPA1(0, β)	9.5037	9.6110	9.5949	9.5949	9.5949
ICPA1(α, β)	9.7218	11.0092	11.0613	11.0620	11.0620
GD0(α, 0)	7.1610	9.2163	9.3568	9.3546	9.3544
GD0(0, β)	6.8746	9.5953	9.6392	9.6000	9.5949
GD0(α, β)	5.3263	8.9866	10.8804	11.0423	11.0620
η=1/4, ISNR= 12.9164					
IOPA1(α, 0)	13.8837	13.9053	13.9053	13.9053	13.9053
IOPA1(0, β)	15.0923	15.6251	15.6109	15.6108	15.6108
IOPA1(α, β)	14.6334	15.9121	15.9192	15.9192	15.9192
ICPA1(α, 0)	13.8693	13.9055	13.9053	13.9053	13.9053
ICPA1(0, β)	15.2045	15.6255	15.6109	15.6108	15.6108
ICPA1(α, β)	14.1329	15.8756	15.9190	15.9192	15.9192
GD0(α, 0)	12.2195	13.8694	13.9052	13.9053	13.9053
GD0(0, β)	8.5703	14.2275	15.6302	15.6153	15.6108
GD0(α, β)	7.2800	12.7687	15.7309	15.9044	15.9192

Table 2.3: The average over 1000 trials of the signal-to-noise ratio $\mathrm{SNR}(m), m = 1, 2, 4, 6, \infty$ denoise the US hourly temperature dataset collected at 218 locations on August 1st, 2010, where $\eta = 35, 20, 10$.

SNR \ m / Alg.	1	2	4	6	∞
η=35, ISNR= 11.5496					
IOPA1($\tilde{\alpha}$, 0)	14.8906	16.2623	16.2499	16.2497	16.2497
IOPA1(0, $\tilde{\beta}$)	13.3792	15.7143	15.6925	15.6911	15.6911
IOPA1($\tilde{\alpha}$, $\tilde{\beta}$)	11.2985	18.1294	19.0536	19.0491	19.0487
ICPA1($\tilde{\alpha}$, 0)	14.2783	16.3118	16.2509	16.2498	16.2497
ICPA1(0, $\tilde{\beta}$)	14.0451	15.7475	15.6925	15.6911	15.6911
ICPA1($\tilde{\alpha}$, $\tilde{\beta}$)	9.8634	16.9294	19.0281	19.0486	19.0487
GD0($\tilde{\alpha}$, 0)	7.2407	13.2001	16.1692	16.2523	16.2497
GD0(0, $\tilde{\beta}$)	5.7453	10.8805	15.3374	15.7069	15.6911
GD0($\tilde{\alpha}$, $\tilde{\beta}$)	3.9579	7.8606	14.4865	17.9663	19.0487
η=20, ISNR= 16.4086					
IOPA1($\tilde{\alpha}$, 0)	18.3271	20.2473	20.2470	20.2470	20.2470
IOPA1(0, $\tilde{\beta}$)	15.4936	20.4129	20.5195	20.5183	20.5183
IOPA1($\tilde{\alpha}$, $\tilde{\beta}$)	12.3927	21.0773	22.8075	22.8097	22.8095
ICPA1($\tilde{\alpha}$, 0)	17.5792	20.2654	20.2474	20.2470	20.2470
ICPA1(0, $\tilde{\beta}$)	16.73029	20.5223	20.5196	20.5183	20.5183
ICPA1($\tilde{\alpha}$, $\tilde{\beta}$)	10.7460	19.4217	22.7759	22.8092	22.8095
GD0($\tilde{\alpha}$, 0)	8.4637	15.7834	20.1310	20.2470	20.2470
GD0(0, $\tilde{\beta}$)	5.9817	11.7217	19.1824	20.4607	20.5183
GD0($\tilde{\alpha}$, $\tilde{\beta}$)	4.2594	8.4753	16.1761	21.0514	22.8095
η=10, ISNR=22.4320					
IOPA1($\tilde{\alpha}$, 0)	23.3572	24.5564	24.5565	24.5565	24.5565
IOPA1(0, $\tilde{\beta}$)	16.9511	25.9123	26.4291	26.4284	26.4284
IOPA1($\tilde{\alpha}$, $\tilde{\beta}$)	14.2863	24.9125	26.9961	26.9990	26.9990
ICPA1($\tilde{\alpha}$, 0)	22.5720	24.5572	24.5565	24.5565	24.5565
ICPA1(0, $\tilde{\beta}$)	18.6319	26.2493	26.4294	26.4285	26.4284
ICPA1($\tilde{\alpha}$, $\tilde{\beta}$)	12.7428	23.3488	26.9816	26.9989	26.9990
GD0($\tilde{\alpha}$, 0)	11.7089	21.2276	24.5387	24.5566	24.5565
GD0(0, $\tilde{\beta}$)	6.2342	12.3916	22.7545	26.1414	26.4284
GD0($\tilde{\alpha}$, $\tilde{\beta}$)	4.9806	9.9239	19.2003	25.2121	26.9990

CHAPTER 3: PRECONDITIONED GRADIENT DESCENT ALGORITHM FOR INVERSE FILTERING ON SPATIALLY DISTRIBUTED NETWORKS

Polynomial graph filters are favorable for a distributed implementation where at each step every node $i \in V$ replaces its signal value with the linear combination of the signal values at the one-hope neighbors. However, not every graph filters can be represented as polynomial of some commutative graph shifts. The existing algorithms for implementation of an arbitrary inverse graph filtering in a distributed manner requires some global information, such as spectral radius and entire spectrum of the filter $\mathbf{H}$. For some applications such as graph filtering in time-varying graph signals, the spectrum of a graph filter may be sensitive to noise or graph topology may change in time, and for this case an inverse graph filtering algorithm with the selection of parameters depending on some global information are not ideal. In this chapter, inspired by the preconditioning method in numerical analysis, we propose the preconditioned gradient descent algorithm (PGDA) to implement an inverse filtering procedure associated with an arbitrary invertible filter $\mathbf{H}$ with small geodesic-width, which converges exponentially. The proposed PGDA is designed based on the local information of the graph and the filter within communication range only.

3.1 Preconditioned Gradient Descent Algorithm for Inverse Filtering

Let $\mathcal{G} := (V, E)$ be a connected, undirected and unweighted graph and $\mathbf{H} = (H(i,j))_{i,j\in V}$ be a filter on the graph $\mathcal{G}$ with geodesic-width $\omega(\mathbf{H})$. In this section, we induce a diagonal matrix $\mathbf{P}_{\mathbf{H}}$

Algorithm 5 Realization of the preconditioner $\mathbf{P_H}$ at a vertex $i \in V$.

Inputs: Geodesic width $\omega(\mathbf{H})$ of the filter $\mathbf{H}$ and nonzero entries $H(i,j)$ and $H(j,i)$ for $j \in B(i,\omega(\mathbf{H}))$ in the i-th row and column of the filter $\mathbf{H}$.

1) Calculate

$d(i) = \max\Big\{ \sum\limits_{j\in B(i,\omega(\mathbf{H}))} |H(i,j)|, \sum\limits_{j\in B(i,\omega(\mathbf{H}))} |H(j,i)| \Big\}$.

2) Send $d(i)$ to all neighbors $k \in B(i,\omega(\mathbf{H}))\backslash\{i\}$ and receive $d(k)$ from neighbors $k \in B(i,\omega(\mathbf{H}))\backslash\{i\}$.

3) Calculate $P_{\mathbf{H}}(i,i) = \max\limits_{k\in B(i,\omega(\mathbf{H}))} d(k)$.

Output: $P_{\mathbf{H}}(i,i)$.

with diagonal elements $P_{\mathbf{H}}(i,i), i \in V$, given by

$$P_{\mathbf{H}}(i,i) := \max_{k\in B(i,\omega(\mathbf{H}))} \Big\{ \max\Big(\sum_{j\in B(k,\omega(\mathbf{H}))} |H(j,k)|, \sum_{j\in B(k,\omega(\mathbf{H}))} |H(k,j)| \Big) \Big\}, \tag{3.1}$$

where we denote the set of all s-hop neighbors of a vertex $i \in V$ by $B(i,s) = \{j \in V,\ \rho(j,i) \leq s\},\ s \geq 0$. The above diagonal matrix $\mathbf{P_H}$ can be evaluated at vertex level and constructed on SDNs with communication range $L \geq \omega(\mathbf{H})$, see Algorithm 5.

For symmetric matrices $\mathbf{A}$ and $\mathbf{B}$, we use $\mathbf{B} \preceq \mathbf{A}$ and $\mathbf{B} \prec \mathbf{A}$ to denote the positive semidefiniteness and positive definiteness of their difference $\mathbf{A} - \mathbf{B}$ respectively. A crucial observation about the diagonal matrix $\mathbf{P_H}$ is as follows.

Theorem 3.1.1. *Let* $\mathbf{H}$ *be a graph filter with geodesic-width* $\omega(\mathbf{H})$ *and* $\mathbf{P_H}$ *be as in* (3.1)*. Then*

$$\mathbf{H}^T\mathbf{H} \preceq \mathbf{P}_{\mathbf{H}}^2. \tag{3.2}$$

Proof. Write $\mathbf{H} = (H(i,j))_{i,j\in V}$. For $\mathbf{x} = (x(i))_{i\in V}$, we have

$$
\begin{aligned}
0 \le \mathbf{x}^T\mathbf{H}^T\mathbf{H}\mathbf{x} &= \sum_{j\in V}\Big|\sum_{i\in V} H(j,i)x(i)\Big|^2 \\
&\le \sum_{j\in V}\Big(\sum_{i\in V}|H(j,i)||x(i)|^2\Big)\times\Big(\sum_{i'\in V}|H(j,i')|\Big) \\
&= \sum_{i\in V}|x(i)|^2\sum_{j\in B(i,\omega(\mathbf{H}))}|H(j,i)|\times\Big(\sum_{i'\in V}|H(j,i')|\Big) \\
&\le \sum_{i\in V}|x(i)|^2 P_{\mathbf{H}}(i,i)\sum_{j\in B(i,\omega(\mathbf{H}))}|H(j,i)| \\
&\le \sum_{i\in V}(P_{\mathbf{H}}(i,i))^2|x(i)|^2 = \mathbf{x}^T\mathbf{P}_{\mathbf{H}}^2\mathbf{x}.
\end{aligned}
$$

This proves (3.2) and completes the proof. □

Denote the spectral radius and operator norm of a matrix $\mathbf{A}$ by $r(\mathbf{A})$ and $\|\mathbf{A}\|_2 = \sup_{\|\mathbf{x}\|_2=1}\|\mathbf{A}\mathbf{x}\|_2$ respectively, where $\|\mathbf{x}\|_2 = (\sum_{j\in V}|x(j)|^2)^{1/2}$ for $\mathbf{x} = (x_j)_{j\in V}$. By Theorem 3.1.1, $\mathbf{P}_{\mathbf{H}}^{-2}\mathbf{H}^T$ is an approximation filter to the inverse filter $\mathbf{H}^{-1}$ in the sense that

$$
r(\mathbf{I}-\mathbf{P}_{\mathbf{H}}^{-2}\mathbf{H}^T\mathbf{H}) = r(\mathbf{I}-\mathbf{P}_{\mathbf{H}}^{-1}\mathbf{H}^T\mathbf{H}\mathbf{P}_{\mathbf{H}}^{-1}) = \|\mathbf{I}-\mathbf{P}_{\mathbf{H}}^{-1}\mathbf{H}^T\mathbf{H}\mathbf{P}_{\mathbf{H}}^{-1}\|_2 < 1. \tag{3.3}
$$

Remark 3.1.2. Define the Schur norm of a matrix $\mathbf{H} = (H(i,j))_{i,j\in V}$ by

$$
\|\mathbf{H}\|_{\mathcal{S}} = \max\Big\{\max_{i\in V}\sum_{j\in V}|H(i,j)|,\ \max_{j\in V}\sum_{i\in V}|H(i,j)|\Big\},
$$

and denote the zero and identity matrices of appropriate size by $\mathbf{O}$ and $\mathbf{I}$ respectively. One may verify that

$$
\mathbf{O} \prec \mathbf{H}^T\mathbf{H} \preceq \|\mathbf{H}\|_{\mathcal{S}}^2\mathbf{I}. \tag{3.4}
$$

By (3.1), we have $\mathbf{P}_{\mathbf{H}} \preceq \|\mathbf{H}\|_{\mathcal{S}}\mathbf{I}$. Then we may consider the conclusion (3.2) for the preconditioner $\mathbf{P}_{\mathbf{H}}$ as a *distributed* version of the well-known matrix dominance (3.4) for the graph filter $\mathbf{H}$.

Preconditioning technique has been widely used in numerical analysis to solve a linear system,

Algorithm 6 Implementation of the PGDA (3.7) at a vertex $i \in V$.

Inputs: Iteration number M, geodesic-width $\omega(\mathbf{H})$, preconditioning constant $P_{\mathbf{H}}(i,i)$, observation $y(i)$ at vertex i, and filter coefficients $H(i,j)$ and $H(j,i),\ j \in B(i,\omega(\mathbf{H}))$.

1) Calculate $\widetilde{H}(j,i) = H(j,i)/(P_{\mathbf{H}}(i,i))^2$.

Initialization: Initial $x^{(0)}(j), j \in B(i,\omega(\mathbf{H}))$, and $m = 1$.

2) Calculate $v^{(m)}(i) = y(i) - \sum_{j \in B(i,\omega(\mathbf{H}))} H(i,j)x^{(m-1)}(j)$.

3) Send $v^{(m)}(i)$ to neighbors $j \in B(i,\omega(\mathbf{H}))$ and receive $v^{(m)}(j)$ from neighbors $j \in B(i,\omega(\mathbf{H}))$.

4) Update
$x^{(m)}(i) = x^{(m-1)}(i) + \sum_{j \in B(i,\omega(\mathbf{H}))} \widetilde{H}(j,i)v^{(m)}(j)$.

5) Send $x^{(m)}(i)$ to neighbors $j \in B(i,\omega(\mathbf{H}))$ and receive $x^{(m)}(j)$ from neighbors $j \in B(i,\omega(\mathbf{H}))$.

6) Set $m = m + 1$ and return to Step **2)** if $m \leq M$.

Outputs: $x(j) := x^{(M)}(j),\ j \in B(i,\omega(\mathbf{H}))$.

where the difficulty is how to select the preconditioner appropriately. Here, we use $\mathbf{P_H}$ as a right preconditioner to the linear system

$$\mathbf{Hx} = \mathbf{y} \tag{3.5}$$

associated with the inverse filtering procedure (1.5), and we solve the following right preconditioned linear system

$$\mathbf{HP_H^{-1}z} = \mathbf{y} \ \text{ and } \ \mathbf{x} = \mathbf{P_H^{-1}z}, \tag{3.6}$$

via the gradient descent algorithm

$$\begin{cases} \mathbf{z}^{(m)} = \mathbf{z}^{(m-1)} - \mathbf{P_H^{-1}H}^T\big(\mathbf{HP_H^{-1}z}^{(m-1)} - \mathbf{y}\big) \\ \mathbf{x}^{(m)} = \mathbf{P_H^{-1}z}^{(m)}, \ \ m \geq 1, \end{cases}$$

with initial $\mathbf{z}^{(0)}$. The above iterative algorithm can be reformulated as a quasi-Newton method (1.6)

with $\mathbf{G}$ replaced by $\mathbf{P}_{\mathbf{H}}^{-2}\mathbf{H}^T$,

$$\begin{cases} \mathbf{e}^{(m)} = \mathbf{H}\mathbf{x}^{(m-1)} - \mathbf{y} \\ \mathbf{x}^{(m)} = \mathbf{x}^{(m-1)} - \mathbf{P}_{\mathbf{H}}^{-2}\mathbf{H}^T\mathbf{e}^{(m)}, \ \ m \geq 1 \end{cases} \tag{3.7}$$

with initial $\mathbf{x}^{(0)}$. We call the above approach to implement the inverse filtering procedure (1.5) by the *preconditioned gradient descent algorithm*, or PGDA for abbreviation.

Define $\mathbf{w}_m := \mathbf{P}_{\mathbf{H}}(\mathbf{x}^{(m)} - \mathbf{H}^{-1}\mathbf{y}),\ m \geq 0$. Then

$$\mathbf{w}_m = \left(\mathbf{I} - \mathbf{P}_{\mathbf{H}}^{-1}\mathbf{H}^T\mathbf{H}\mathbf{P}_{\mathbf{H}}^{-1}\right)\mathbf{w}_{m-1},\ m \geq 1 \tag{3.8}$$

by (3.7). Therefore the iterative algorithm (3.7) converges exponentially by (3.3) and (3.8).

Theorem 3.1.3. Let $\mathbf{H}$ be an invertible graph filter and $\mathbf{x}^{(m)}, m \geq 0$, be as in (3.7). Then

$$\|\mathbf{P}_{\mathbf{H}}(\mathbf{x}^{(m)} - \mathbf{H}^{-1}\mathbf{y})\|_2 \leq \left\|\mathbf{I} - \mathbf{P}_{\mathbf{H}}^{-1}\mathbf{H}^T\mathbf{H}\mathbf{P}_{\mathbf{H}}^{-1}\right\|_2^m \|\mathbf{P}_{\mathbf{H}}(\mathbf{x}^{(0)} - \mathbf{H}^{-1}\mathbf{y})\|_2,\ m \geq 0.$$

In addition to the exponential convergence in Theorem 3.1.3, the PGDA is that each iteration can be implemented at vertex level, see Algorithm 6. Therefore for an invertible filter $\mathbf{H}$ with $\omega(\mathbf{H}) \leq L$, the PGDA (3.7) can implement the inverse filtering procedure (1.5) on SDNs with each agent only storing, computing and exchanging the information in a L-hop neighborhood.

3.2 Symmetric Preconditioned Gradient Descent Algorithm for Inverse Filtering

In this section, we consider implementing the inverse filtering procedure (1.5) associated with a **positive definite** filter $\mathbf{H} = (H(i,j))_{i,j\in V}$ on a connected, undirected and unweighted graph $\mathcal{G}$.

Algorithm 7 Implementation of the SPGDA (3.11) at a vertex $i \in V$.

Inputs: Iteration number M, geodesic-width $\omega(\mathbf{H})$, observation $y(i)$ at vertex i, and filter coefficients $H(i,j)$ and $H(j,i), j \in B(i,\omega(\mathbf{H}))$.

1) Calculate $P_{\mathbf{H}}^{\mathrm{sym}}(i,i) = \sum_{j \in B(i,\omega(\mathbf{H}))} |H(i,j)|$, $\widetilde{H}(i,j) = H(i,j)/P_{\mathbf{H}}^{\mathrm{sym}}(i,i)$ and $\tilde{y}(i) = y(i)/P_{\mathbf{H}}^{\mathrm{sym}}(i,i), j \in B(i,\omega(\mathbf{H}))$.

Initialization: Initial $x^{(0)}(j), j \in B(i,\omega(\mathbf{H}))$ and $m = 1$.

2) Compute
$x^{(m)}(i) = x^{(m-1)}(i) + \tilde{y}(i) - \sum_{j \in B(i,\omega(\mathbf{H}))} \widetilde{H}(i,j) x^{(m-1)}(j)$.

3) Send $x^{(m)}(i)$ to neighbors $j \in B(i,\omega(\mathbf{H}))$ and receive $x^{(m)}(j)$ from neighbors $j \in B(i,\omega(\mathbf{H}))$.

4) Set $m = m+1$ and return to Step **2)** if $m \leq M$.

Outputs: $x(j) := x^{(M)}(j), j \in B(i,\omega(\mathbf{H}))$.

Define the diagonal matrix $\mathbf{P}_{\mathbf{H}}^{\mathrm{sym}}$ with diagonal entries

$$P_{\mathbf{H}}^{\mathrm{sym}}(i,i) = \sum_{j \in B(i,\omega(\mathbf{H}))} |H(i,j)|, \ i \in V, \tag{3.9}$$

and set

$$\widehat{\mathbf{H}} = (\mathbf{P}_{\mathbf{H}}^{\mathrm{sym}})^{-1/2} \mathbf{H} (\mathbf{P}_{\mathbf{H}}^{\mathrm{sym}})^{-1/2}. \tag{3.10}$$

We remark that the normalized matrix in (3.10) associated with a diffusion matrix has been used to understand diffusion process [52], and the one corresponding to the Laplacian $\mathbf{L}_{\mathcal{G}}$ on the graph $\mathcal{G}$ is half of its normalized Laplacian $\mathbf{L}_{\mathcal{G}}^{\mathrm{sym}} := (\mathbf{D}_{\mathcal{G}})^{-1/2} \mathbf{L}_{\mathcal{G}} (\mathbf{D}_{\mathcal{G}})^{-1/2}$, where $\mathbf{D}_{\mathcal{G}}$ is degree matrix of $\mathcal{G}$ [64]. Similar to the PGDA (3.7), we propose the following *symmetric preconditioned gradient descent algorithm*, or SPGDA for abbreviation,

$$\mathbf{x}^{(m)} = \mathbf{x}^{(m-1)} - (\mathbf{P}_{\mathbf{H}}^{\mathrm{sym}})^{-1} (\mathbf{H}\mathbf{x}^{(m-1)} - \mathbf{y}), \ m \geq 1, \tag{3.11}$$

with initial $\mathbf{x}^{(0)}$, to solve the following preconditioned linear system

$$\widehat{\mathbf{H}}\mathbf{z} = (\mathbf{P}_{\mathbf{H}}^{\mathrm{sym}})^{-1/2} \mathbf{y} \ \text{ and } \ \mathbf{x} = (\mathbf{P}_{\mathbf{H}}^{\mathrm{sym}})^{-1/2} \mathbf{z}. \tag{3.12}$$

Comparing with the PGDA (3.7), the SPGDA for a positive definite graph filter has less computation and communication cost in each iteration and it also can be implemented at vertex level, see Algorithm 7.

For $\mathbf{x} = (x(i))_{i\in V}$, we obtain from (3.9) and the symmetry of the matrix $\mathbf{H}$ that

$$\mathbf{x}^T\mathbf{H}\mathbf{x} \le \sum_{i,j\in V} |H(i,j)| \frac{(x(i))^2 + (x(j))^2}{2} = \mathbf{x}^T\mathbf{P}_{\mathbf{H}}^{\rm sym}\mathbf{x}.$$

Combining (3.1) and (3.9) proves that $\mathbf{H} \preceq \mathbf{P}_{\mathbf{H}}^{\rm sym} \preceq \mathbf{P}_{\mathbf{H}}$, cf. (3.2). This together with (3.10) implies that

$$r(\mathbf{I} - (\mathbf{P}_{\mathbf{H}}^{\rm sym})^{-1}\mathbf{H}) = r(\mathbf{I} - \widehat{\mathbf{H}}) = \|\mathbf{I} - \widehat{\mathbf{H}}\|_2 < 1. \tag{3.13}$$

Similar to the proof of Theorem 3.1.3, we have

Theorem 3.2.1. Let $\mathbf{H}$ be a positive definite graph filter. Then $\mathbf{x}^{(m)}, m \ge 0$, in (3.11) converges exponentially,

$$\|(\mathbf{P}_{\mathbf{H}}^{\rm sym})^{1/2}(\mathbf{x}^{(m)} - \mathbf{H}^{-1}\mathbf{y})\|_2 \|\mathbf{I} - \widehat{\mathbf{H}}\|_2^m \big\|(\mathbf{P}_{\mathbf{H}}^{\rm sym})^{1/2}(\mathbf{x}^{(0)} - \mathbf{H}^{-1}\mathbf{y})\big\|_2.$$

3.3 Simulations

Let $\mathcal{G}_N = (V_N, E_N), N \ge 2$, be random geometric graphs with N vertices deployed on $[0,1]^2$ and an undirected edge between two vertices if their physical distance is not larger than $\sqrt{2/N}$ [42, 64]. In the first simulation, we consider the inverse filtering procedure associated with the graph filter $\mathbf{H} = \mathbf{H}_o + (\mathbf{L}_{\mathcal{G}_N}^{\rm sym})^2$, where $K \ge 1$, $\mathbf{L}_{\mathcal{G}_N}^{\rm sym}$ is the normalized Laplacian on the graph $\mathcal{G}_N$, the filter $\mathbf{H}_o = (H_o(i,j))_{i,j\in V_N}$ is defined by $H_o(i,j) = 0$ if $\rho(i,j) \ge 3$ and

$$H_o(i,j) = \exp\Big(-2K\|(i_x,i_y) - (j_x,j_y)\|_2^2 - \frac{\|(i_x,i_y) + (j_x,j_y)\|_2^2}{2}\Big) + \frac{\gamma_{ij} + \gamma_{ji}}{2} \text{ if } \rho(i,j) \le 2,$$

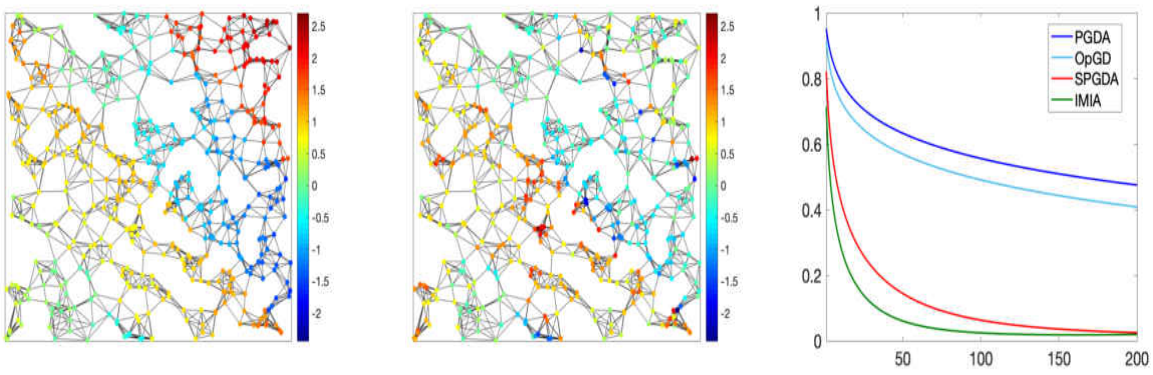

Figure 3.1: Plotted on the left is a corrupted blockwise polynomial signal $\mathbf{x}$ and in the middle is the output $\mathbf{y} = \mathbf{H}\mathbf{x}$ of the filtering procedure, where $\|\mathbf{x}\|_2 = 24.8194, \|\mathbf{y}\|_2 = 21.5317$ and the condition number of the filter $\mathbf{H}$ is 107.40. Shown on the right is average of the relative inverse filtering error $E_2(m) = \|\mathbf{x}^{(m)} - \mathbf{x}\|_2/\|\mathbf{x}\|_2, 1 \leq m \leq 200$ over 1000 trials, where $N = K = 512, \eta = 0.2, \gamma = 0.05$ and $\mathbf{x}^{(m)}$, $m \geq 1$, are the outputs of SPGDA, PGDA, OpGD and IMIA.

(i_x, i_y) is the coordinator of a vertex $i \in V_N$ and γ_{ij} are i.i.d random noises uniformly distributed on $[-\gamma, \gamma]$. Let $\mathbf{x}_o$ be the blockwise polynomial consisting of four strips and imposes $(0.5 - 2i_x)$ on the first and third diagonal strips and $(0.5 + i_x^2 + i_y^2)$ on the second and fourth strips respectively [51, 64]. In the simulation, the signals $\mathbf{x} = \mathbf{x}_o + \boldsymbol{\eta}$ are obtained by a blockwise polynomial $\mathbf{x}_o$ corrupted by noises $\boldsymbol{\eta}$ with their components being i.i.d. random variables with uniform distribution on $[-\eta, \eta]$, and the observations $\mathbf{y}$ of the filtering procedure are given by $\mathbf{y} = \mathbf{H}\mathbf{x}$, see the left and middle images of Figure 3.1. In the simulation, we use the SPGDA (3.11) and the PGDA (3.7) with zero initial to implement the inverse filtering procedure $\mathbf{y} \mapsto \mathbf{H}^{-1}\mathbf{y}$, and also we compare their performances with the gradient decent algorithm

$$\mathbf{x}^{(m)} = (\mathbf{I} - \beta_{op}\mathbf{H}^T\mathbf{H})\mathbf{x}^{(m-1)} + \beta_{op}\mathbf{H}^T\mathbf{y}, \ m \geq 1$$

with zero initial and optimal step length β_{op} selected in [28, 51, 27], OpGD in abbreviation, and the iterative matrix inverse approximation algorithm,

$$\mathbf{x}^{(m)} = (\mathbf{I} - \tilde{\mathbf{D}}\mathbf{H})\mathbf{x}^{(m-1)} + \tilde{\mathbf{D}}\mathbf{y},\ m \geq 1$$

IMIA in abbreviation, where $\mathbf{x}^{(0)} = \mathbf{0}$ and the diagonal matrix $\tilde{\mathbf{D}}$ has entries

$$H(i,i)/(\sum_{\rho(j,i)\leq 2} |H(i,j)|^2), i \in V,$$

see [49, Eq. (3.4)] with $\tilde{\sigma} = 0$. Shown in Figure 3.1 is the average of the relative inverse filtering error $E_2(m), 1 \leq m \leq 200$ over 1000 trials, and it reaches the relative error 5% at about 57th iteration for IMIA, 118th iteration for SPGDA, and more than 3000 iterations for PGDA and OpGD. This confirms that $\mathbf{x}^{(m)}, m \geq 1$, in the SPGDA, PGDA, OpGD and IMIA converge exponentially to the output $\mathbf{x}$ of the inverse filtering, and the convergence rate are spectral radii of matrices $\mathbf{I} - (\mathbf{P}_{\mathbf{H}}^{\mathrm{sym}})^{-1}\mathbf{H}$, $\mathbf{I} - \mathbf{P}_{\mathbf{H}}^{-2}\mathbf{H}^T\mathbf{H}$, $\mathbf{I} - \beta_{op}\mathbf{H}^T\mathbf{H}$ and $\mathbf{I} - \tilde{\mathbf{D}}\mathbf{H}$, see Theorems 3.1.3 and 3.2.1. Here the average of spectral radii in SPGDA, PGDA, OpGD and IMIA shown in Figure 3.1 are $0.9786, 0.9996, 0.9993, 0.9566$ respectively. We remark that the reason for PGDA and OpGD to have slow convergence in the above simulation could be that their spectral radii are too close to 1. Our simulation shows that for the graph filter on some random geometric graphs of order $N = 1024$, which has one as its diagonal entries and nondiagonal entries of $\mathbf{H}_o$ in (3.14) with $\gamma = 0$ and $K = 512$ as its nondiagonal entries, the corresponding PGDA, OpGD, SPGDA converge and the IMIA diverges.

Let $\mathcal{G}_T = (V_T, E_T)$ be the undirected graph with 218 locations in the United States as vertices and edges constructed by the 5 nearest neighboring locations, and let $\mathbf{x}_{12}$ be the recorded temperature vector of those 218 locations on August 1st, 2010 at 12:00 PM, see Figure 3.2 [43, 51]. In the second simulation, we consider to implement the inverse filtering procedure $\tilde{\mathbf{x}} = (\mathbf{I} + \alpha\mathbf{L}_{\mathcal{G}_T}^{\mathrm{sym}})^{-1}\mathbf{b}$

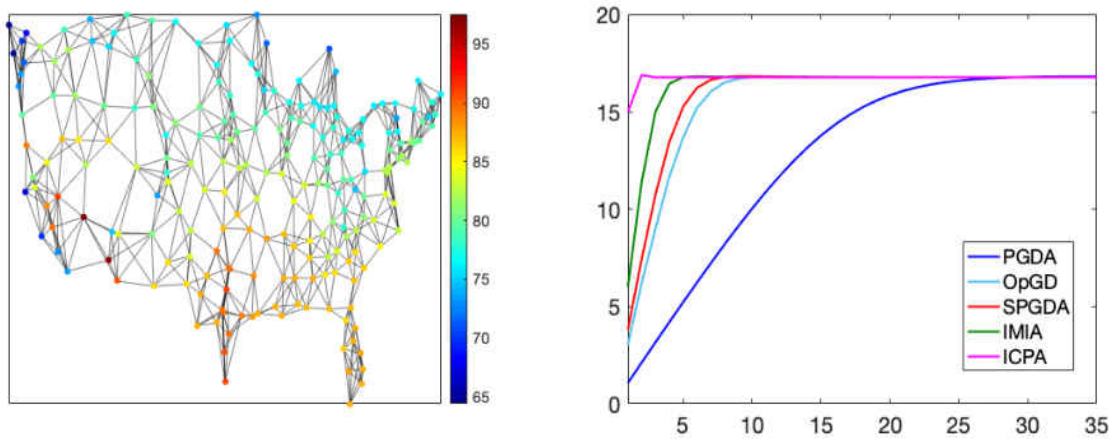

Figure 3.2: Plotted on the left is the original temperature data $\mathbf{x}_{12}$. Shown on the right is average of the signal-to-noise ratio $\text{SNR}(m) = -20\log_{10}\|\mathbf{x}^{(m)} - \mathbf{x}_{12}\|_2/\|\mathbf{x}_{12}\|_2, 1 \leq m \leq 35$, over 1000 trials, where $\mathbf{x}^{(m)}$, $m \geq 1$, are the outputs of PGDA, SPGDA, OpGD, IMIA and ICPA, and average of the limit SNR is 16.7869.

arisen from the minimization problem $\tilde{\mathbf{x}} := \arg\min_{\mathbf{z}} \|\mathbf{z}-\mathbf{b}\|_2^2 + \alpha \mathbf{z}^T \mathbf{L}_{\mathcal{G}_T}^{\text{sym}} \mathbf{z}$ in denoising the hourly temperature data $\mathbf{x}_{12}$, where $\mathbf{L}_{\mathcal{G}_T}^{\text{sym}}$ is the normalized Laplacian on $\mathcal{G}_T$, α is a penalty constraint and $\mathbf{b} = \mathbf{x}_{12} + \boldsymbol{\eta}$ is the temperature vector corrupted by i.i.d. random noise $\boldsymbol{\eta}$ with its components being randomly selected in $[-\eta, \eta]$ in a uniform distribution [43, 51]. Shown in Figure 3.2 is the performance of the SPGDA, PGDA, OpGD, IMIA and ICPA to implement the above inverse filtering procedure with noise level $\eta = 35$ and the penalty constraint $\alpha = 0.9075$ [51], where ICPA is the iterative Chebyshev polynomial approximation algorithm of order one [12, 51, 66]. This indicates that the 3rd term in ICPA, the 5th term in IMIA, the 8th term of SPGDA, the 10th term of OpGD and the 30th term of PGDA can be used as the denoised temperature vector $\tilde{\mathbf{x}}$.

To implement the inverse filter procedure (1.5) on SDNs, we observe from the above two simulations that OpGD outperforms PGDA while the selection of optimal step length in OpGD is computationally expensive. If the filter is positive definite, SPGDA, IMIA and ICPA may have better performance than OpGD and PGDA have. On the other hand, SPGDA always converges, but the

requirement in [49, Theorem 3.2] to guarantee the convergence of IMIA may not be satisfied and ICPA is applicable for polynomial filters.

CHAPTER 4: DISTRIBUTED ALGORITHM TO DETERMINE EIGENVECTORS OF MATRICES ON SPATIALLY DISTRIBUTED NETWORKS

In the literature, the eigenspaces of a matrix on the graph $\mathcal{G}$ have been used to understand the communicability between vertices, spectral clustering for the network and influence of a vertex on the network [53, 54, 56, 57, 58, 59]. In this chapter, following the preconditioned gradient descent algorithm we propose the distributed algorithm to approximate an eigenvectors of a complex-valued matrices with limited geodesic-width.

4.1 A Distributed Iterative Algorithm for Determining Eigenvectors

Let $\mathcal{G} = (V, E)$ be a connected, undirected and unweighted graph of order N. Denote the set of all s-hop neighbors of a vertex $i \in V$ by $B(i, s) = \{j \in V,\ \rho(j, i) \leq s\}, s \geq 0$. For a complex-valued matrix $\mathbf{A} = (A(i, j))_{i,j\in V}$, we denote its Hermitian transpose by $\mathbf{A}^*$ and define the diagonal preconditioning matrix $\mathbf{P_A}$ with diagonal elements

$$P_{\mathbf{A}}(i, i) := \max_{k\in B(i,\omega(\mathbf{A}))} \Big\{ \max\Big(\sum_{j\in B(k,\omega(\mathbf{A}))} |A(j, k)|, \sum_{j\in B(k,\omega(\mathbf{A}))} |A(k, j)|\Big)\Big\},\ i \in V \tag{4.1}$$

as in (3.1) [55]. In this section, we introduce a distributed iterative algorithm to find eigenvectors associated with a given eigenvalue for complex-valued matrices with small geodesic-width.

Theorem 4.1.1. Let $\mathbf{A}$ be a complex-valued matrix on the graph $\mathcal{G}$, $\mathbf{P_A}$ be the diagonal matrix given in (4.1), and $\mathbf{Q}$ be a nonsingular diagonal matrix such that

$$\mathbf{Q} - \mathbf{P_A} \text{ is positive semidefinite.} \tag{4.2}$$

Then for any initial $\mathbf{x}_0 \in \mathbb{C}^N$, the sequence $\mathbf{x}_n, n \geq 1$, defined inductively by

$$\mathbf{x}_{n+1} = (\mathbf{I} - \mathbf{Q}^{-2}\mathbf{A}^*\mathbf{A})\mathbf{x}_n,\ n \geq 0, \tag{4.3}$$

converges exponentially to either the zero vector or an eigenvector associated with the zero eigenvalue of the matrix $\mathbf{A}$.

Proof. By nonsingularity of the matrix $\mathbf{Q}$, the proof reduces to showing

$$\|\mathbf{Q}(\mathbf{x}_n - \mathbf{u})\|_2 \leq \|\mathbf{Q}\mathbf{x}_0\|_2 r^n,\ \ n \geq 0 \tag{4.4}$$

for some $\mathbf{u}$ satisfying $\mathbf{A}\mathbf{u} = \mathbf{0}$, where $r \in (0, 1)$. Set $\mathbf{B} = \mathbf{I} - \mathbf{Q}^{-1}\mathbf{A}^*\mathbf{A}\mathbf{Q}^{-1}$ and let $\mathbf{u}_i$ be orthonormal eigenvectors associated with eigenvalues γ_i of the Hermitian matrix $\mathbf{B}$,

$$\mathbf{B}\mathbf{u}_i = \gamma_i \mathbf{u}_i,\ \ 1 \leq i \leq N. \tag{4.5}$$

Following the argument in Theorem 3.1.1 [55, Theorem II.1] and applying (4.2), we obtain that $\mathbf{Q}^2 - \mathbf{A}^*\mathbf{A}$ is positive semidefinite. This together with nonsingularity of the matrix $\mathbf{Q}$ implies that

$$0 \leq \gamma_i \leq 1,\ 1 \leq i \leq N. \tag{4.6}$$

Write $\mathbf{Q}\mathbf{x}_0 = \sum_{i=1}^N \langle \mathbf{Q}\mathbf{x}_0, \mathbf{u}_i \rangle \mathbf{u}_i$, where $\langle \cdot, \cdot \rangle$ is the standard inner product on $\mathbb{C}^N$. By (4.3), we have that $\mathbf{Q}\mathbf{x}_n = \mathbf{B}\mathbf{Q}\mathbf{x}_{n-1},\ n \geq 1$. This together with (4.5) implies that

$$\mathbf{Q}\mathbf{x}_n = \mathbf{B}^n \mathbf{Q}\mathbf{x}_0 = \sum_{i=1}^N \gamma_i^n \langle \mathbf{Q}\mathbf{x}_0, \mathbf{u}_i \rangle \mathbf{u}_i,\ n \geq 0. \tag{4.7}$$

Define $\mathbf{u} = \sum_{\gamma_i = 1} \langle \mathbf{Q}\mathbf{x}_0, \mathbf{u}_i \rangle \mathbf{Q}^{-1}\mathbf{u}_i$. Then by (4.6), (4.7) and the orthonormality of $\mathbf{u}_i, 1 \leq i \leq N$,

we obtain

$$\|\mathbf{Q}(\mathbf{x}_n - \mathbf{u})\|_2 = \Big(\sum_{0 \le \gamma_i < 1} |\langle \mathbf{Q}\mathbf{x}_0, \mathbf{u}_i \rangle|^2 \gamma_i^{2n} \Big)^{1/2} \le r^n \|\mathbf{Q}\mathbf{x}_0 - \mathbf{Q}\mathbf{u}\|_2 \le r^n \|\mathbf{Q}\mathbf{x}_0\|_2, \tag{4.8}$$

where $r = \max_{0 \le \gamma_i < 1} \gamma_i$. This proves (4.4) and the desired exponential convergence of the sequence $\mathbf{x}_n, n \ge 0$.

Taking the limit in (4.3) and applying the convergence in (4.4) gives $\mathbf{Q}^{-2}\mathbf{A}^*\mathbf{A}\mathbf{u} = \mathbf{0}$. This proves that $\mathbf{u}$ is either the zero vector or an eigenvector associated with eigenvalue zero.

□

We remark that a nonsingular diagonal matrix $\mathbf{Q}_c = \mathrm{diag}(Q_c(i,i))_{i \in V}$ satisfying (4.2) can be constructed at the vertex level by setting

$$Q_c(i,i) = \max(P_{\mathbf{A}}(i,i), c), \ i \in V, \tag{4.9}$$

where c is a positive constant and the i-th diagonal entries $P_{\mathbf{A}}(i,i)$ of preconditioning matrix $\mathbf{P}_{\mathbf{A}}$ can be obtained by the distributed algorithm, see Algorithm 5 and [55, Algorithm II.1].

Let $\mathbf{H} = (H(i,j))_{i,j \in V}$ be an arbitrary matrix on the graph $\mathcal{G}$ and λ be its eigenvalue. By selecting the initial $\mathbf{x}_0$ with entries i.i.d variable randomly selected from [0,1] and applying the iterative algorithm (4.3) to the matrix $\mathbf{A} = \mathbf{H} - \lambda \mathbf{I}$, we obtain from the proof of Theorem 4.1.1 that the limit of the sequence $\mathbf{x}_n, n \ge 0$, is a nonzero vector with probability one and hence it is an eigenvector of the matrix $\mathbf{H}$ associated with eigenvalue λ. Following the terminology in [55], we call the above algorithm to find eigenvectors of a matrix as a *preconditioned gradient descent algorithm*, PGDA for abbreviation.

Algorithm 8 Realization of the PGDA to find an eigenvector at a vertex $i \in V$.

Inputs: The total iteration number M, the geodesic-width $\omega(\mathbf{H})$ of the matrix $\mathbf{H} = (H(i,j))_{i,j\in V}$, the set $B(i,\omega(\mathbf{H}))$ of $\omega(\mathbf{H})$-hop neighbors of the vertex i, the eigenvalue λ of the matrix $\mathbf{H}$, entries $H(i,j)$ and $H(j,i), j \in B(i,\omega(\mathbf{H}))$ in the i-th row and column of the matrix $\mathbf{H}$, and the i-th diagonal entry $Q(i,i)$ of the matrix $\mathbf{Q}$.

Pre-iteration: Compute $A(i,j) = H(i,j) - \lambda\delta(i,j)$ and $\tilde{A}(j,i) = (Q(i,i))^{-2}\big(\overline{H(j,i)} - \overline{\lambda}\delta(j,i)\big)$ for $j \in B(i,\omega(\mathbf{H}))$, where δ is the Kronecker delta.

Initial: Select the i-th component $x_0(i) \in [0,1]$ of the initial vector $\mathbf{x}_0$ randomly, and set $n = 0$.

Iteration:

1. Send $x_n(i)$ to all neighbors $k \in B(i,\omega(\mathbf{H}))\backslash\{i\}$ and receive $x_n(k)$ from neighbors $k \in B(i,\omega(\mathbf{H}))\backslash\{i\}$.
2. Evaluate $\tilde{x}_n(i) = \sum_{j\in B(i,\omega(\mathbf{H}))} A(i,j)x_n(j)$.
3. Send $\tilde{x}_n(i)$ to all neighbors $k \in B(i,\omega(\mathbf{H}))\backslash\{i\}$ and receive $\tilde{x}_n(k)$ from neighbors $k \in B(i,\omega(\mathbf{H}))\backslash\{i\}$.
4. Evaluate $\widehat{x}_n(i) = \sum_{j\in B(i,\omega(\mathbf{H}))} \tilde{A}(j,i)\tilde{x}_n(j)$.
5. Set $x_{n+1}(i) = x_n(i) - \widehat{x}_n(i)$ and $n = n+1$.
6. return to step 1 if $n \leq M$, go to Output otherwise.

Output: $u(i) \approx x_M(i)$, where $\mathbf{u} = (u(i))_{i\in V}$ is the eigenvector.

The significance of the proposed PGDA is the distributed implementation at the vertex level, see Algorithm 5. For the implementation of Algorithm 5, every vertex $i \in V$ is required to have the information of its $\omega(\mathbf{H})$-hop neighbors, equipped direct communication with its $\omega(\mathbf{H})$-hop neighbors, and need memory to store the eigenvalue λ, the iteration number M, the i-th diagonal entries of the matrix $\mathbf{Q}$, and entries $H(i,j)$ and $H(j,i), j \in B(i,\omega(\mathbf{H}))$ in the i-th row and column of the matrix $\mathbf{H}$. Moreover, the computational and communication expenses for each vertex is independent on the order N of the graph $\mathcal{G}$. With the selection of the nonsingular diagonal matrix $\mathbf{Q}$ as in (4.2), we conclude that the proposed PGDA can be applied for an SDN with communication range L to find eigenvectors associated with a given eigenvalue for **arbitrary** matrix $\mathbf{H}$ with geodesic width $\omega(\mathbf{H}) \leq L$.

Principal eigenvectors associated with eigenvalue one of some left stochastic matrix on a network have been used to determine the influence of a vertex, see [53, 54] and references therein. Let $\mathbf{W} = (w(i,j))_{i,j\in V}$ be the hyperlink matrix on a network described by a graph $\mathcal{G} = (V, E)$, where weights $w(i,j) = 0$ for $(i,j) \notin E$ and $w(i,j) = 1/d_j$ for $(i,j) \in E$, the reciprocal of the degree d_j of a node j. The matrix $\mathbf{W}$ is a left stochastic matrix with one as the leading eigenvalue and the principal eigenvector associated with eigenvalue one has positive entries by Perron-Frobenius theorem. Applying the proposed PGDA to the hyperlink matrix $\mathbf{W}$, we can locally evaluate principal eigenvectors of the hyperlink matrix and hence identify the local influence of a vertex on its neighborhood.

4.2 Evaluation of Eigenvectors of Positive Semidefinite Matrices

In this section, we consider finding eigenvectors associated with the zero eigenvalue of a positive semidefinite matrix on a connected, undirected and unweighted graph in a distributed manner.

Theorem 4.2.1. Let $\mathbf{A} = (A(i,j))_{i,j\in V}$ be a positive semidefinite matrix on the graph $\mathcal{G} = (V, E)$ of order N with its geodesic-width denoted by $\omega(\mathbf{A})$, and $\mathbf{Q}^{\mathrm{sym}} = \mathrm{diag}(Q^{\mathrm{sym}}(i,i))_{i\in V}$ be a non-singular diagonal matrix with diagonal entries satisfying

$$Q^{\mathrm{sym}}(i,i) \geq \sum_{j\in B(i,\omega(\mathbf{A}))} |A(i,j)|,\ i \in V. \tag{4.10}$$

Then for any $\mathbf{x}_0 \in \mathbb{C}^N$, the sequence $\mathbf{x}_n, n \geq 0$, defined by

$$\mathbf{x}_{n+1} = (\mathbf{I} - (\mathbf{Q}^{\mathrm{sym}})^{-1}\mathbf{A})\mathbf{x}_n, \tag{4.11}$$

converges exponentially to either the zero vector or an eigenvector associated with the zero eigenvalue of the matrix $\mathbf{A}$.

Proof. By the nonsingularity of the matrix $\mathbf{Q}^{\rm sym}$, the proof of the exponential convergence reduces to establishing

$$\|(\mathbf{Q}^{\rm sym})^{1/2}(\mathbf{x}_n - \mathbf{u})\|_2 \leq \|(\mathbf{Q}^{\rm sym})^{1/2}\mathbf{x}_0\|_2 r^n, \;\; n \geq 0 \tag{4.12}$$

for some $r \in (0,1)$ and a vector $\mathbf{u} \in \mathbb{C}^N$ satisfying $\mathbf{A}\mathbf{u} = \mathbf{0}$. Following the argument in Theorem 3.2.1 and applying (4.10), we obtain that $\mathbf{Q}^{\rm sym} - \mathbf{A}$ is positive semidefinite. This together with the positive semidefiniteness of the matrix $\mathbf{A}$ implies that all eigenvalues of the Hermitian matrix $\mathbf{B}^{\rm sym} := \mathbf{I} - (\mathbf{Q}^{\rm sym})^{-1/2}\mathbf{A}(\mathbf{Q}^{\rm sym})^{-1/2}$ are in the unit interval $[0,1]$, cf. (4.6). Applying similar argument used in the proof of Theorem 4.1.1 with $\mathbf{Q}$ and $\mathbf{A}^*\mathbf{A}$ replaced by $(\mathbf{Q}^{\rm sym})^{1/2}$ and $\mathbf{A}$ respectively, we can prove the exponential convergence of $\mathbf{x}_n, n \geq 0$ in (4.12) with r being the largest eigenvalue of $\mathbf{B}^{\rm sym}$ in $[0,1)$. □

For a positive semidefinite matrix $\mathbf{A} = (A(i,j))_{i,j\in V}$ with geodesic-width $\omega(\mathbf{A})$, a nonsingular diagonal matrix $\mathbf{Q}_c^{\rm sym} = {\rm diag}(Q_c^{\rm sym}(i,i))_{i\in V}$ satisfying (4.10) can be constructed at the vertex level by setting

$$Q_c^{\rm sym}(i,i) = \max\Big(\sum_{j\in B(i,\omega(\mathbf{A}))} |A(i,j)|, c\Big), \; i \in V, \tag{4.13}$$

where c is a positive constant, cf. (4.9). With the above selection of the preconditioning matrix in (4.11), we can find eigenvectors of the positive semidefinite matrix $\mathbf{A}$ associated with eigenvalue zero by the distributed iterative algorithm (4.11) implemented at the vertex level, see Algorithm 9. Following the terminology in [55], we call the above algorithm as a *symmetric preconditioned gradient descent algorithm*, SPGDA for abbreviation. Comparing with Algorithm 8 to find eigenvectors for an arbitrary matrix, the Algorithm 9 for a positive semidefinite matrix takes shorter running time and less communication expense in each iteration. Our numerical simulations in Section 4.4 also indicate that it may have faster convergence.

Algorithm 9 Realization of the SPGDA at a vertex $i \in V$.

Inputs: The total iteration number M, the geodesic-width $\omega(\mathbf{A})$ of the positive semidefinite matrix $\mathbf{A}$, the set $B(i, \omega(\mathbf{A}))$ of $\omega(\mathbf{A})$-hop neighbors of the vertex i, entries $A(i, j), j \in B(i, \omega(\mathbf{A}))$ in the i-th row of the matrix $\mathbf{A}$ and the i-th entry $Q^{\rm sym}(i, i)$ of the diagonal matrix $\mathbf{Q}^{\rm sym}$.
Pre-iteration: Evaluate $\tilde{A}(i, j) = (Q^{\rm sym}(i, i))^{-1} A(i, j)$, $j \in B(i, \omega(\mathbf{A}))$.
Initial: Select $x_0(i)$ randomly in $[0, 1]$, and set $n = 0$.
Iteration:

1. Send $x_n(i)$ to all neighbors $k \in B(i, \omega(\mathbf{A}))\backslash\{i\}$ and receive $x_n(k)$ from neighbors $k \in B(i, \omega(\mathbf{A}))\backslash\{i\}$.
2. Evaluate $x_{n+1}(i) = x_n(i) - \sum_{j \in B(i, \omega(\mathbf{A}))} \tilde{A}(i, j) x_n(j)$ and set $n = n + 1$.
3. return to step 1 if $n \leq M$, go to Output otherwise.

Output: $v(i) \approx y_M(i)$, where $\mathbf{v} = (v(i))_{i \in V}$.

4.3 Eigenvectors of Polynomial Filters

Graph filter is a fundamental concept in graph signal processing and it has been used in many applications such as denoising and consensus of multi-agent systems [1, 3, 4, 5, 11, 12, 51, 55, 60, 61, 62, 63, 64]. Graph filters in most of literature are designed to be polynomials

$$\mathbf{A} = h(\mathbf{S}_1, \ldots, \mathbf{S}_d) = \sum_{l_1=0}^{L_1} \cdots \sum_{l_d=0}^{L_d} h_{l_1, \ldots, l_d} \mathbf{S}_1^{l_1} \cdots \mathbf{S}_d^{l_d} \tag{4.14}$$

of commutative graph shifts $\mathbf{S}_1, ..., \mathbf{S}_d$, i.e., $\mathbf{S}_k \mathbf{S}_{k'} = \mathbf{S}_{k'} \mathbf{S}_k$ for all $1 \leq k, k' \leq d$, where the multivariate polynomial $h(t_1, \ldots, t_d) = \sum_{l_1=0}^{L_1} \cdots \sum_{l_d=0}^{L_d} h_{l_1, \ldots, l_d} t_1^{l_1} \ldots t_d^{l_d}$ has polynomial coefficients $h_{l_1, \ldots, l_d}$, $0 \leq l_k \leq L_k, 1 \leq k \leq d$ [11, 12, 18, 20, 21, 64, 65, 66]. On the graph $\mathcal{G} = (V, E)$, a polynomial filter $\mathbf{A}$ in (4.14) can be represented by a matrix $\mathbf{A} = (A(i, j))_{i,j \in V}$, which has geodesic-width no more than the degree of the polynomial h, i.e., $\omega(\mathbf{A}) \leq \sum_{k=1}^{d} L_k$. Then we can apply the PGDA (and the SPGDA if $\mathbf{A}$ is positive semidefinite) to find eigenvectors associated with eigenvalue zero on SDNs with communication range $L \geq \sum_{k=1}^{d} L_k$. In this section, we propose ierative algorithm to determine eigenvectors associated with a polynomial graph filter $\mathbf{A}$

Algorithm 10 Realization of each iteration in the iterative algorithms (4.3) and (4.11) at a vertex $i \in V$ for a polynomial filter $\mathbf{A}$.

Inputs: Polynomial coefficients $h_{l_1,\ldots,l_d}, 0 \le l_1 \le L_1, \ldots, 0 \le l_d \le L_d$ of the polynomial filter $\mathbf{A}$ in (4.14), the set $\mathcal{N}_i$ of all adjacent vertices j of the vertex i, entries $S_k(i,j)$ and $S_k(j,i), j \in \mathcal{N}_i$ of graph shifts $\mathbf{S}_k, 1 \le k \le d$, the i-th diagonal entry $Q(i,i)$ of the matrix $\mathbf{Q}$, and the i-th entry $x_n(i)$ of the input vector $\mathbf{x}_n = (x_n(k))_{k\in V}$ at n-th iteration,

1: Apply Algorithm 2 to implement the polynomial filter procedure $\mathbf{x} \longmapsto \mathbf{A}\mathbf{x}$ at the vertex i. The input is the i-th entry $x_n(i)$ of $\mathbf{x}_n$ and the output is the i-th entry $\hat{x}_n(i)$ of $\hat{\mathbf{x}}_n = \mathbf{A}\mathbf{x}_n =: (\hat{x}_n(k))_{k\in V}$.

2: Apply Step 1 with the matrix $\mathbf{A}$ replaced by its complex conjugate $\mathbf{A}^*$ and the input $x_n(i)$ by $\hat{x}_n(i)$. The output is the i-th entry $\check{x}_n(i)$ of the vector $\check{\mathbf{x}}_n = \mathbf{A}^*\hat{\mathbf{x}}_n =: (\check{x}_n(k))_{k\in V}$.

3: Evaluate $x_{n+1}(i) = x_n(i) - (Q(i,i))^{-2}\check{x}_n(i)$ and $\tilde{x}_{n+1}(i) = x_n(i) - (Q(i,i))^{-1}\hat{x}_n(i)$.

Outputs: The outputs $x_{n+1}(i)$ and $\tilde{x}_{n+1}(i)$ are the i-th entry of $\mathbf{x}_{n+1}$ at n-th iteration in (4.3) and (4.11) respectively.

in (4.14) which can be implemented on an SDN with 1 as its communication range, i.e., direct communication exists between all adjacent vertices.

Observe that

$$\mathbf{A}^* = \sum_{l_1=0}^{L_1} \cdots \sum_{l_d=0}^{L_d} \overline{h_{l_1,\ldots,l_d}} (\mathbf{S}_d^*)^{l_d} \cdots (\mathbf{S}_1^*)^{l_1} \tag{4.15}$$

is a polynomial graph filter of commutative shifts $\mathbf{S}_1^*, ..., \mathbf{S}_d^*$. Then applying Algorithm 2 to implement the filtering procedure associated with polynomial graph filters $\mathbf{A}$ and $\mathbf{A}^*$, we can implement each iteration in the PGDA (4.3) and the SPGDA (4.11) in finite steps with each step including data exchanging between adjacent vertices only, see Algorithm 10. This concludes that eigenvectors associated with a given eigenvalue for a polynomial graph filter on SDNs with communication range one can be obtained by applying Algorithm 10 in each iteration.

Now it remains to construct diagonal matrices satisfying (4.2) and (4.10) on SDNs with communication range one. For the polynomial grah filter $\mathbf{A}$ in (4.14), define diagonal matrices $\widehat{\mathbf{Q}}_c =$

$\mathrm{diag}(\widehat{Q}_c(i,i))_{i\in V}$ and $\widehat{\mathbf{Q}}_c^{\mathrm{sym}} = \mathrm{diag}(\widehat{Q}_c^{\mathrm{sym}}(i,i))_{i\in V}$ by

$$\widehat{Q}_c(i,i) = \max_{\rho(j,i)\le L} \max\Big\{ \sum_{k\in V} \widehat{A}(j,k), \sum_{k\in V} \widehat{A}(k,j), c\Big\}, \tag{4.16}$$

and

$$\widehat{Q}_c^{\mathrm{sym}}(i,i) = \max\Big\{ \sum_{k\in V} \widehat{A}(j,k), c\Big\},\ i\in V, \tag{4.17}$$

where c is a positive number, $|\mathbf{S}_k| = (|S_k(i,j)|)_{i,j\in V}, 1\le k\le d$, and

$$(\widehat{A}(i,j))_{i,j\in V} =: \widehat{\mathbf{A}} := \sum_{l_1=0}^{L_1} \cdots \sum_{l_d=0}^{L_d} |h_{l_1,\ldots,l_d}| |\mathbf{S}_1|^{l_1} \cdots |\mathbf{S}_d|^{l_d}.$$

One may verify that $|A(i,j)| \le \widehat{A}(i,j)$ for all $i,j\in V$. Therefore the matrices $\widehat{\mathbf{Q}}_c$ and $\widehat{\mathbf{Q}}_c^{\mathrm{sym}}$ in (4.16) and (4.17) satisfy (4.2) and (4.10) respectively. Moreover, as shown in Algorithm 11, they can be constructed at the vertex level in finite steps such that in each step, every vertex needs to exchange data with adjacent vertices only.

4.4 Simulations

Let $\mathcal{G}_N = (V_N, E_N), N\ge 2$, be random geometric graphs with N vertices deployed on $[0,1]^2$ and an undirected edge between two vertices in V_N existing if their physical distance is not larger than $\sqrt{2/N}$ [64, 42]. In this section, we consider finding eigenvectors associated with eigenvalue 1 of lowpass spline filters $\mathbf{H}_{0,m}^{\mathrm{spln}} = (\mathbf{I} - \mathbf{L}^{\mathrm{sym}}/2)^m, m\ge 1$, where $\mathbf{L}^{\mathrm{sym}}$ is the symmetric normalized Laplacian matrix on the graph $\mathcal{G}_N$ [64, 38]. In the simulations, we take $c = 0.01$ and use PGDA and PGDA1h to denote the PGDA with $\mathbf{A}$ replaced by $\mathbf{I} - \mathbf{H}_{0,m}^{\mathrm{spln}}$ and $\mathbf{Q}$ by $\mathbf{Q}_c$ in (4.9) and $\widehat{\mathbf{Q}}_c$ in (4.16) respectively, and similarly we use SPGDA and SPGDA1h to denote the SPGDA with $\mathbf{A}$ replaced by $\mathbf{I} - \mathbf{H}_{0,m}^{\mathrm{spln}}$ and $\mathbf{Q}$ by $\mathbf{Q}_c^{\mathrm{sym}}$ in (4.13) and $\widehat{\mathbf{Q}}_c^{\mathrm{sym}}$ in (4.17) respectively. Set

Algorithm 11 Construction of diagonal entries $\widehat{Q}_c(i,i)$ and $\widehat{Q}_c^{\rm sym}(i,i)$ at a vertex $i \in V$ for a polynomial filter $\mathbf{A}$.

Inputs: The positive constant c, polynomial coefficients $h_{l_1,\ldots,l_d}, 0 \le l_1 \le L_1, \ldots, 0 \le l_d \le L_d$, of the polynomial filter $\mathbf{A}$, entries $S_k(i,j)$ and $S_k(j,i)$ for all $1 \le k \le d$ and $j \in \mathcal{N}_i$.

1: Apply Algorithm 2 to implement the polynomial filter procedure $\mathbf{1} \longmapsto \widehat{\mathbf{A}}\mathbf{1}$ at the vertex i. The input is the i-th entry 1 of the all-one vector $\mathbf{1}$ and the output is the i-th entry $a_1(i)$ of the vector $\widehat{\mathbf{A}}\mathbf{1} =: (a_1(k))_{k\in V}$.

2: Apply Step 1 with the same input but the filter $\widehat{\mathbf{A}}$ replaced by $\widehat{\mathbf{A}^*}$. The output is the i-th entry $a_2(i)$ of the vector $\widehat{\mathbf{A}^*}\mathbf{1} =: (a_2(k))_{k\in V}$.

3: Evaluate $q_0(i) = \max(a_1(i), a_2(i), c)$ and set $l = 0$.

4: Finite-step Iteration:

4a) Send $q_l(i)$ to all adjacent vertices $k \in \mathcal{N}_i$ and receive $q_l(k)$ from all adjacent vertices $k \in \mathcal{N}_i$.

4b) Compare $q_l(i)$ with $q_l(k), k \in \mathcal{N}_i$ and define $q_{l+1}(i) = \max(q_l(i), \max_{k\in\mathcal{N}_i} q_l(k))$ and set $l := l + 1$.

4c) Return to step 1 if $l \le L_1 + \ldots + L_d$, go to Outputs otherwise.

Outputs: $\widehat{Q}_c(i,i) = q_L$ and $\widehat{Q}_c^{\rm sym}(i,i) = \max(a_1(i), c)$.

$\|\mathbf{x}\|_2 = \left(\sum_{j\in V} |x(j)|^2\right)^{1/2}$ for $\mathbf{x} = (x_j)_{j\in V}$. For the sequences $\mathbf{x}_n, n \ge 0$, in the PGDA, SPGDA, PGDA1h and SPGDA1h, and their limits $\mathbf{u}$, define convergence errors $\mathrm{CE}(n) = \log_{10}\|\tilde{\mathbf{x}}_n - \tilde{\mathbf{u}}\|_2$ and normalized residues $\mathrm{NR}(n) = \log_{10}\|(\mathbf{I} - \mathbf{H}_{0,m}^{\rm spln})\tilde{\mathbf{x}}_n\|_2, n \ge 0$, in the logarithmic scale, where $\tilde{\mathbf{x}}_n = \mathbf{x}_n/\|\mathbf{x}_n\|_2$ and $\tilde{\mathbf{u}} = \mathbf{u}/\|\mathbf{u}\|_2$. Shown in Figure 4.1 are the average of convergence errors $\mathrm{CE}(n)$ and normalized residues $\mathrm{RE}(n), n \ge 0$, over 500 trials. This demonstrates the exponential convergence of $\mathbf{x}_n, n \ge 0$, in the proposed distributed iterative algorithms to some eigenvector associated with eigenvalue 1 of $\mathbf{H}_{0,m}^{\rm spln}, m \ge 1$, which is proved in Theorems 4.1.1 and 4.2.1.

For a matrix $\mathbf{A}$ on a graph $\mathcal{G} = (V, E)$, define its Schur norm by $\|\mathbf{A}\|_{\mathcal{S}} = \max_{i\in V} P_{\mathbf{A}}(i,i)$, where $P_{\mathbf{A}}(i,i), i \in V$, are given by (4.1) [9, 55]. For the case that the constant c in (4.9) and (4.13) is chosen that $c \ge \|\mathbf{A}\|_{\mathcal{S}}$, the matrices $\mathbf{Q}_c$ and $\mathbf{Q}_c^{\rm sym}$ become a multiple of the identity $\mathbf{I}$ and the corresponding PGDA and SPGDA are the conventional gradient descent algorithm and the symmetric gradient descent algorithm respectively [12, 21, 28, 51, 55]. We denote the above algorithms with

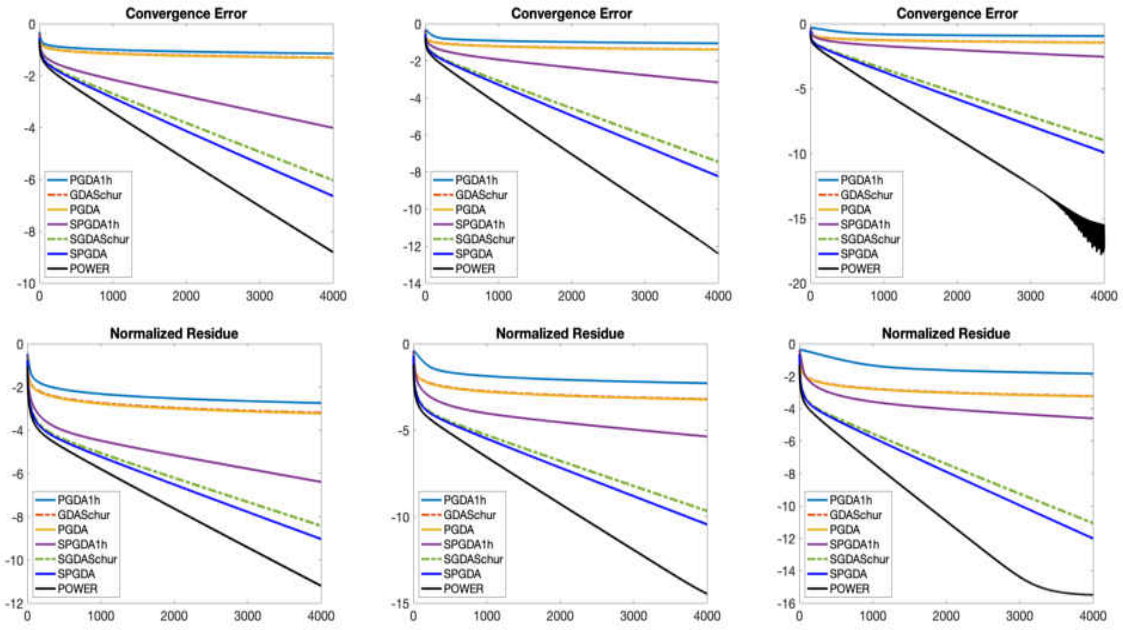

Figure 4.1: Plotted on the first and second rows are average of the convergence errors $\mathrm{CE}(n)$ and the normalized residues $\mathrm{RE}(n), 1 \leq n \leq 4000$, over 500 trials, while from left to right are lowpass spline filters $\mathbf{H}_{0,m}^{\mathrm{spln}}$ of orders $m = 2, 3, 4$ on the random geometric graph $\mathcal{G}_N$ with $N = 512$.

$c = \|\mathbf{A}\|_{\mathcal{S}}$ by GDASchur and SGDASchur respectively, see Figure 4.1 for their performances to determine eigenvectors associated with eigenvalue 1 of $\mathbf{H}_{0,m}^{\mathrm{spln}}, m \geq 1$. As matrices $\mathbf{H}_{0,m}^{\mathrm{spln}}, m \geq 1$, have 1 as their maximal eigenvalue in absolute value, we can use the conventional power iteration method with entries of the initial $\mathbf{x}_0$ randomly selected in $[0, 1]$, POWER for abbreviation, to find eigenvectors associated with eigenvalue 1 [67]. Presented in Figure 4.1 is its performance. From Figure 4.1, we observe that the centralized algorithm POWER has fastest convergence rate to find eigenvectors associated with eigenvalue 1 of matrices $\mathbf{H}_{0,m}^{\mathrm{spln}}, 2 \leq m \leq 4$, as followed are the distributed algorithm SPGDA, the centralized algorithm SPGDASchur and the distributed algorithm SPGDA1h, the next are the distributed algorithm PGDA and the centralized algorithm GDASchur, and the distributed algorithm PGDA1h has slowest convergence.

APPENDIX A: COMMUTATIVE SHIFTS AND JOINT SPECTRUM

The assumption that the graph shifts $\mathbf{S}_1, \ldots, \mathbf{S}_d$ are commutative, is indispensable for us to develop the IOPA and ICPA algorithms for inverse filtering. In the first part of the Appendix, we discuss the joint spectrum of commutative graph shifts. Graph shifts are building blocks of a polynomial filter and the concept of commutative graph shifts is similar to the one-order delay $z_1^{-1}, \ldots, z_d^{-1}$ in classical multi-dimensional signal processing. In this appendix, we introduce two illustrative families of commutative graph shifts on circulant graphs and product graphs, see also Sections 2.4.2 and 2.4.3 for commutative graph shifts with specific features.

A.1 Joint Spectrum of Commutative Shifts

Let $\mathbf{S}_1, ..., \mathbf{S}_d$ be commutative graph shifts. Then they can be upper-triangularized simultaneously by [44, Theorem 2.3.3], i.e.,

$$\widehat{\mathbf{S}}_k = \mathbf{U}^{\mathrm{H}} \mathbf{S}_k \mathbf{U},\ 1 \leq k \leq d, \tag{A.1}$$

are upper triangular matrices for some unitary matrix $\mathbf{U}$. Write $\widehat{\mathbf{S}}_k = (\widehat{S}_k(i,j))_{1 \leq i,j \leq N}, 1 \leq k \leq d$, and set

$$\Lambda = \big\{ \boldsymbol{\lambda}_i = \big(\widehat{S}_1(i,i), ..., \widehat{S}_d(i,i)\big), 1 \leq i \leq N \big\}. \tag{A.2}$$

As $\widehat{S}_k(i,i), 1 \leq i \leq N$, are eigenvalues of $\mathbf{S}_k,\ 1 \leq k \leq d$, we call Λ as the *joint spectrum* of $\mathbf{S}_1, \ldots, \mathbf{S}_d$. The joint spectrum Λ of commutative shifts $\mathbf{S}_1, \ldots, \mathbf{S}_d$ plays an essential role in Section 2.3 to construct optimal polynomial approximation filters and Chebyshev polynomial approximation filters to the inverse filter of a polynomial filter of $\mathbf{S}_1, ..., \mathbf{S}_d$.

A.2 Commutative Shifts on Circulant Graphs and Product Graphs

Let $\mathcal{C}(N, Q)$ be the circulant graph of order N generated by $Q = \{q_1, \ldots, q_M\}$, where $1 \leq q_1 < \ldots < q_M < N/2$. Observe that $E_N(Q) = \cup_{1 \leq k \leq d}\{(i, i \pm q_k \bmod N), i \in V_N\}$. Then the symmetric normalized Laplacian matrix $\mathbf{L}^{\mathrm{sym}}_{\mathcal{C}(N,Q)}$ on $\mathcal{C}(N, Q)$ is the average of symmetric normalized Laplacian matrices $\mathbf{L}^{\mathrm{sym}}_{\mathcal{C}(N,Q_k)}$ on $\mathcal{C}(N, Q_k), 1 \leq k \leq d$, i.e.,

$$\mathbf{L}^{\mathrm{sym}}_{\mathcal{C}(N,Q)} = \frac{1}{d}\sum_{k=1}^{d} \mathbf{L}^{\mathrm{sym}}_{\mathcal{C}(N,Q_k)},$$

where $Q_k = \{q_k\}, 1 \leq k \leq d$. In the following proposition, we establish the commutativity of $\mathbf{L}^{\mathrm{sym}}_{\mathcal{C}(N,Q_k)}, 1 \leq k \leq d$.

Proposition A.2.1. The symmetric normalized Laplacian matrices $\mathbf{L}^{\mathrm{sym}}_{\mathcal{C}(N,Q_k)}$ of the circulant graphs $\mathcal{C}(N, Q_k), 1 \leq k \leq d$, are commutative graph shifts on the circulant graph $\mathcal{C}(N, Q)$.

Proof. Clearly $\mathbf{L}^{\mathrm{sym}}_{\mathcal{C}(N,Q_k)}, 1 \leq k \leq d$, are graph shifts on the circulant graph $\mathcal{C}(N, Q)$. Define

$$\mathbf{B} = (b(i - j \bmod N))_{1 \leq i,j \leq N},$$

where $b(0) = \cdots = b(N - 2) = 0$ and $b(N - 1) = 1$. Then one may verify that

$$\mathbf{L}^{\mathrm{sym}}_{\mathcal{C}(N,Q_k)} = \mathbf{I} - \frac{1}{2}(\mathbf{B}^{q_k} + \mathbf{B}^{-q_k}) = -\frac{1}{2}\mathbf{B}^{-q_k}(\mathbf{B}^{q_k} - \mathbf{I})^2,$$

where $1 \leq k \leq d$. Therefore for $1 \leq k, k' \leq d$,

$$\mathbf{L}^{\mathrm{sym}}_{\mathcal{C}(N,Q_{k'})}\mathbf{L}^{\mathrm{sym}}_{\mathcal{C}(N,Q_k)} = \frac{1}{4}\mathbf{B}^{-q_k-q_{k'}}(\mathbf{B}^{q_k} - \mathbf{I})^2(\mathbf{B}^{q_{k'}} - \mathbf{I})^2 = \mathbf{L}^{\mathrm{sym}}_{\mathcal{C}(N,Q_k)}\mathbf{L}^{\mathrm{sym}}_{\mathcal{C}(N,Q_{k'})}.$$

This completes the proof. □

For undirected and unweighted finite graphs $\mathcal{G}_1$ and $\mathcal{G}_2$, let $\mathcal{G}_1 \times \mathcal{G}_2$ be the Cartesian product graph of $\mathcal{G}_1$ and $\mathcal{G}_2$ [40, 41]. Denote symmetric normalized Laplacian matrices and orders of the graph $\mathcal{G}_i, i = 1, 2$ by $\mathbf{L}_i^{\mathrm{sym}}$ and N_i respectively. One may verify that $\mathbf{L}_1^{\mathrm{sym}} \otimes \mathbf{I}_{N_2}$ and $\mathbf{I}_{N_1} \otimes \mathbf{L}_2^{\mathrm{sym}}$ are graph filters of the Cartesian product graph $\mathcal{G}_1 \times \mathcal{G}_2$. In the following proposition, we show that they are commutative.

Proposition A.2.2. Filters $\mathbf{L}_1^{\mathrm{sym}} \otimes \mathbf{I}_{N_2}$ and $\mathbf{I}_{N_1} \otimes \mathbf{L}_2^{\mathrm{sym}}$ are commutative graph shifts on $\mathcal{G}_1 \times \mathcal{G}_2$.

Proof. Set $\mathbf{C}_1 = \mathbf{L}_1^{\mathrm{sym}} \otimes \mathbf{I}_{N_2}$ and $\mathbf{C}_2 = \mathbf{I}_{N_1} \otimes \mathbf{L}_2^{\mathrm{sym}}$. Then

$$\mathbf{C_1 C}_2 = \mathbf{L}_1^{\mathrm{sym}} \otimes \mathbf{L}_2^{\mathrm{sym}} = \mathbf{C}_2 \mathbf{C}_1,$$

where the equality follows from the mixed-product property $(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = (\mathbf{AC}) \otimes (\mathbf{BD})$ for Kronecker product of matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ of appropriate sizes [45]. □

LIST OF 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, "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, Sept. 2014.

[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] D. K. Hammod, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Appl. Comput. Harmon. Anal.*, vol. 30, no. 4, pp. 129-150, Mar. 2011.

[5] 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, Jun. 2012.

[6] 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.

[7] 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.

[8] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs: Frequency analysis," *IEEE Trans. Signal Process.*, vol. 62, no. 12, pp. 3042-3054, Jun. 2014.

[9] C. Cheng, Y. Jiang, and Q. Sun, "Spatially distributed sampling and reconstruction," *Appl. Comput. Harmon. Anal.*, vol. 47, no. 1, pp. 109-148, Jul. 2019.

[10] J. Yi and L. Chai, "Graph filter design for multi-agent system consensus," in *IEEE 56th Annual Conference on Decision and Control (CDC)*, Melbourne, VIC, 2017, pp. 1082-1087.

[11] W. Waheed and D. B. H. Tay, "Graph polynomial filter for signal denoising," *IET Signal Process.*, vol. 12, no. 3, pp. 301-309, Apr. 2018.

[12] D. I. Shuman, P. Vandergheynst, D. Kressner, and P. Frossard, "Distributed signal processing via Chebyshev polynomial approximation," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 4, no. 4, pp. 736-751, Dec. 2018.

[13] 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, Sept. 2019.

[14] V. N. Ekambaram, G. C. Fanti, B. Ayazifar, and K. Ramchandran, "Circulant structures and graph signal processing," in *Proc. IEEE Int. Conf. Image Process.*, 2013, pp. 834-838.

[15] J. Jiang, C. Cheng, and Q. Sun, "Nonsubsampled graph filter banks: Theory and distributed algorithms," *IEEE Trans. Signal Process.*, vol. 67, no. 15, pp. 3938-3953, Aug. 2019.

[16] J. Jiang, D. B. Tay, Q. Sun, and S. Ouyang, "Design of nonsubsampled graph filter banks via lifting schemes," *IEEE Signal Process. Lett.*, vol. 27, pp. 441-445, Feb. 2020.

[17] C. Cheng, N. Emirov, and Q. Sun, "Preconditioned gradient descent algorithm for inverse filtering on spatially distributed networks", arXiv:2007.11491

[18] S. Segarra, A. G. Marques, and A. Ribeiro, "Optimal graph-filter design and applications to distributed linear network operators," *IEEE Trans. Signal Process.*, vol. 65, no. 15, pp. 4117-4131, Aug. 2017.

[19] 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.

[20] M. Coutino, E. Isufi, and G. Leus, "Advances in distributed graph filtering," *IEEE Trans. Signal Process.*, vol. 67, no. 9, pp. 2320-2333, May 2019.

[21] E. Isufi, A. Loukas, A. Simonetto, and G. Leus, "Autoregressive moving average graph filtering," *IEEE Trans. Signal Process.*, vol. 65, no. 2, pp. 274-288, Jan. 2017.

[22] K. Lu, A. Ortega, D. Mukherjee and Y. Chen, "Efficient rate-distortion approximation and transform type selection using Laplacian operators," in *2018 Picture Coding Symposium (PCS)*, San Francisco, CA, 2018, pp. 76-80.

[23] C. Cheng, J. Jiang, N. Emirov, and Q. Sun, "Iterative Chebyshev polynomial algorithm for signal denoising on graphs," in *Proceeding 13th Int. Conf. on SampTA*, Bordeaux, France, Jul. 2019, pp. 1-5.

[24] T. Kurokawa, T. Oki, and H. Nagao, "Multi-dimensional graph Fourier transform," *arXiv*: 1712.07811, Dec. 2017.

[25] J. Fan, C. Tepedelenlioglu, and A. Spanias, "Graph filtering with multiple shift matrices," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3557-3561, May 2019.

[26] A. W. Bohannon, B. M. Sadler, and R. V. Balan, "A filtering framework for time-varying graph signals," in *Vertex-Frequency Analysis of Graph Signals*, Springer, pp. 341-376, 2019.

[27] X. Shi, H. Feng, M. Zhai, T. Yang, and B. Hu, "Infinite impulse response graph filters in wireless sensor networks," *IEEE Signal Process. Lett.*, vol. 22, no. 8, pp. 1113-1117, Aug. 2015.

[28] S. Chen, A. Sandryhaila, J. M. F. Moura, and J. Kovačević, "Signal recovery on graphs: variation minimization," *IEEE Trans. Signal Process.*, vol. 63, no. 17, pp. 4609-4624, Sept. 2015.

[29] M. Onuki, S. Ono, M. Yamagishi, and Y. Tanaka, "Graph signal denoising via trilateral filter on graph spectral domain," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 2, no. 2, pp. 137-148, Jun. 2016.

[30] S. Chen, A. Sandryhaila, and J. Kovačević, "Distributed algorithm for graph signal inpainting," *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Brisbane, QLD, 2015, pp. 3731-3735.

[31] K. Qiu, X. Mao, X. Shen, X. Wang, T. Li, and Y. Gu, "Time-varying graph signal reconstruction," *IEEE J. Sel. Topics Signal Process.*, vol. 11, no. 6, pp. 870-883, Sept. 2017.

[32] F. Chung, *Spectral Graph Theory*, CBMS Regional Conference Series in Mathematics, No. 92. Providence, RI, Amer. Math. Soc., 1997.

[33] A. Sakiyama, K. Watanabe, Y. Tanaka, and A. Ortega, "Two-channel critically sampled graph filter banks with spectral domain sampling," *IEEE Trans. Signal Process.*, vol. 67, no. 6, pp. 1447-1460, Mar. 2019.

[34] W. Cheney and W. Light. *A Course in Approximation Theory*, Brook/Cole Publishing Company, 2000.

[35] G. M. Phillips, *Interpolation and Approximation by Polynomials*, CMS Books Math., Springer-Verlag, 2003.

[36] V. N. Ekambaram, G. C. Fanti, B. Ayazifar, and K. Ramchandran, "Multiresolution graph signal processing via circulant structures," in *Proc. IEEE Digital Signal Process. Signal Process. Educ. Meeting (DSP/SPE)*, 2013, pp. 112-117.

[37] M. S. Kotzagiannidis and P. L. Dragotti, "Splines and wavelets on circulant graphs," *Appl. Comput. Harmon. Anal.*, vol. 47, no. 2, pp. 481-515, Sept. 2019.

[38] M. S. Kotzagiannidis and P. L. Dragotti, "Sampling and reconstruction of sparse signals on circulant graphs – an introduction to graph-FRI," *Appl. Comput. Harmon. Anal.*, vol. 47, no. 3, pp. 539-565, Nov. 2019.

[39] D. Valsesia, G. Fracastoro, and E. Magli, "Deep graph-convolutional image denoising," *arXiv*:1907.08448, Jul. 2019.

[40] A. Loukas and D. Foucard, "Frequency analysis of time-varying graph signals," in *IEEE Global Conf. Signal Inf. Process. (GlobalSIP)*, 2016, pp. 346-350.

[41] F. Grassi, A. Loukas, N. Perraudin, and B. Ricaud, "A time-vertex signal processing framework: scalable processing and meaningful representations for time-series on graphs," *IEEE Trans. Signal Process.*, vol. 66, no. 3, pp. 817-829, Feb. 2018.

[42] P. Nathanael, J. Paratte, D. Shuman, L. Martin, V. Kalofolias, P. Vandergheynst, and D. K. Hammond, "GSPBOX: A toolbox for signal processing on graphs," *arXiv*:1408.5781, Aug. 2014.

[43] J. Zeng, G. Cheung, and A. Ortega, "Bipartite approximation for graph wavelet signal decomposition," *IEEE Trans. Signal Process.*, vol. 65, no. 20, pp. 5466-5480, Oct. 2017.

[44] R. A. Horn and C. R. Johnson. *Matrix Analysis*, Cambridge University Press, 2012.

[45] A. J. Laub, *Matrix Analysis for Scientists and Engineers*, PA, Philadelphia, SIAM, 2005.

[46] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Comput. Netw.*, vol. 52, no. 12, pp. 2292-2330, Aug. 2008.

[47] R. Hebner, "The power grid in 2030," *IEEE Spectrum*, vol. 54, no. 4, pp. 50-55, Apr. 2017.

[48] J. Jiang, D. B. Tay, Q. Sun and S. Ouyang, "Recovery of time-varying graph signals via distributed algorithms on regularized problems", *IEEE Trans. Signal Inf. Process. Netw.*, vol. 6, pp. 540-555, 2020.

[49] J. Jiang, and D. B. Tay, "Decentralised signal processing on graphs via matrix inverse approximation," *Signal Process.*, vol. 165, pp. 292-302, Dec. 2019.

[50] J. Jiang, C. Cheng, and Q. Sun, "Nonsubsampled graph filter banks: theory and distributed algorithms," *IEEE Trans. Signal Process.*, vol. 67, no. 15, pp. 3938-3953, Aug. 2019.

[51] N. Emirov, C. Cheng, J. Jiang, and Q. Sun, "Polynomial graph filter of multiple shifts and distributed implementation of inverse filtering," arXiv: 2003.11152, Mar. 2020.

[52] B. Nadler, S. Lafon, I. Kevrekidis, and R. Coifman, "Diffusion maps, spectral clustering and eigenfunctions of Fokker-Planck operators," In *Advances in Neural Information Processing Systems 18*, Y. Weiss, B. Schölkopf, and J. Platt eds, MIT Press, Cambridge, 2006, pp. 955-962.

[53] D. F. Gleich, "Pagerank beyond the web," *SIAM Review*, vol. 57, no. 3, pp. 321-363, 2015.

[54] A. Langville and C. Meyer, *Google's PageRank and Beyond: The Science of Search Engine Rankings*, Princeton University Press, 2006.

[55] C. Cheng, N. Emirov, and Q. Sun, "Preconditioned gradient descent algorithm for inverse filtering on spatially distributed networks," *IEEE Signal Process. Lett.*, vol. 27, pp. 1834-1838, 2020.

[56] X. Ma, L. Gao, and X. Yong, "Eigenspaces of networks reveal the over- lapping and hierarchical community structure more precisely," *J. Stat. Mech. Theory Exp.*, vol. 2010, no. 8, pp. P08012, Aug. 2010.

[57] R. J. Snchez-Garca, M. Fennelly, S. Norris, N. Wright, G. Niblo, J. Brodzki, and J. W. Bialek, "Hierarchical spectral clustering of power grids," *IEEE Trans. Power Syst.*, vol. 29, no. 5, pp. 2229-2237, Sept. 2014.

[58] D. I. Shuman, M. J. Faraji, and P Vandergheynst, "A multiscale pyramid transform for graph signals," *IEEE Trans. Signal Process.*, vol. 64, no. 8, pp. 2119-2134, Apr. 2016.

[59] A. Gusrialdi and Z. Qu, "Distributed estimation of all the eigenvalues and eigenvectors of matrices associated with strongly connected digraphs," *IEEE Control Syst. Lett.*, vol. 1, no. 2, pp. 329–333, Oct. 2017.

[60] 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.

[61] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs: Frequency analysis," *IEEE Trans. Signal Process.*, vol. 62, no. 12, pp. 3042-3054, June 2014.

[62] 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.

[63] J. Yi and L. Chai, "Graph filter design for multi-agent system consensus," in *IEEE 56th Annual Conference on Decision and Control (CDC)*, Melbourne, VIC, 2017, pp. 1082-1087.

[64] J. Jiang, C. Cheng, and Q. Sun, "Nonsubsampled graph filter banks: Theory and distributed algorithms," *IEEE Trans. Signal Process.*, vol. 67, no. 15, pp. 3938-3953, Aug. 2019.

[65] K. Lu, A. Ortega, D. Mukherjee, and Y. Chen, "Efficient rate-distortion approximation and transform type selection using Laplacian operators," in *2018 Picture Coding Symposium (PCS)*, San Francisco, CA, 2018, pp. 76-80.

[66] C. Cheng, J. Jiang, N. Emirov, and Q. Sun, “Iterative Chebyshev polynomial algorithm for signal denoising on graphs,” in *Proceeding 13th Int. Conf. on SampTA*, Bordeaux, France, July 2019, pp. 1-5.

[67] G. H. Golub and C. F. V. Loan, *Matrix Computations*, The Johns Hopkins University Press, 2013.