A Preconditioned Graph Diffusion LMS for Adaptive Graph Signal Processing

Fei Hua[*†], Roula Nassif[‡], Cédric Richard[†], Haiyan Wang[*], Ali H. Sayed[‡]
[*]School of Marine Science and Technology, Northwestern Polytechnical University, Xi'an 710072, China
[†] Laboratoire Lagrange, Université Côte d'Azur, OCA, CNRS, Nice 06108, France
[‡]Ecole Polytechnique Fédérale de Lausanne, Switzerland
Email: fei.hua@oca.eu; roula.nassif@epfl.ch; cedric.richard@unice.fr; hywang@nwpu.edu.cn; ali.sayed@epfl.ch

***Abstract*—Graph filters, defined as polynomial functions of a graph-shift operator (GSO), play a key role in signal processing over graphs. In this work, we are interested in the adaptive and distributed estimation of graph filter coefficients from streaming graph signals. To this end, diffusion LMS strategies can be employed. However, most popular GSOs such as those based on the graph Laplacian matrix or the adjacency matrix are not energy preserving. This may result in a large eigenvalue spread and a slow convergence of the graph diffusion LMS. To address this issue and improve the transient performance, we introduce a graph diffusion LMS-Newton algorithm. We also propose a computationally efficient preconditioned diffusion strategy and we study its performance.**

***Index Terms*—Graph signal processing, graph filter, diffusion LMS, LMS-Newton, preconditioned LMS.**

I. Introduction

Graphs are a powerful modeling tool for network-structured applications such as sensor networks, smart grids, communication networks, social networks, etc. Data generated by these applications often lie on complex and irregular structures, and require appropriate graph signal processing (GSP) techniques to be analyzed and manipulated. As an extension of classical signal processing techniques applied to data on graphs, GSP has attracted increasing attention in the last few years [1]–[3]. Recent results include sampling [4]–[6], spectral analysis [7]–[9], and filtering [10]. Most of this literature however focuses on static graph signals, i.e., signals that need not evolve with time. There have been extensive prior work on handling streaming signals over graphs and on designing adaptive learning strategies for such scenarios, e.g., [11], [12] and the references therein. Since many network-structured applications involve dynamic signals and need an appropriate formalism, some recent studies started applying these ideas to streaming graph signals [13]–[15]. Graph filters play a key role in GSP. The aim of this paper is to blend concepts from adaptive networks [16] and GSP to estimate graph filter coefficients from dynamic graph signals.

We consider a graph $\mathcal{G}$ consisting of a set $\mathcal{N}$ of N nodes, and a set $\mathcal{E}$ of edges such that if node k is connected to node ℓ, then $(k,\ell) \in \mathcal{E}$. Let $\mathcal{N}_k$ denote the neighborhood of node k including k, namely, $\mathcal{N}_k = \{\ell : \ell = k \vee (\ell,k) \in \mathcal{E}\}$. Graph signals are defined as $\boldsymbol{x} = [x_1,\dots,x_N]^\top \in \mathbb{R}^N$ where x_k is the signal sample residing on node k. Let $\boldsymbol{x}(i)$ denote the graph signal at time i. We assume that the graph is endowed with a GSO defined as an $N \times N$ shift matrix $\boldsymbol{S}$ whose entry $s_{k\ell}$ can be non-zero only if $k = \ell$ or $(k,\ell) \in \mathcal{E}$. The shift matrix captures the graph topology. Possible choices include the adjacency matrix, the graph Laplacian matrix, and their normalized counterparts [1]. Operation $\boldsymbol{Sx}$ is called graph shift. It can be performed locally at each node k by linearly combing the samples x_ℓ from its neighboring nodes $\ell \in \mathcal{N}_k$.

In this paper we consider linear shift-invariant FIR graph filters $\boldsymbol{H} : \mathbb{R}^{N\times N} \to \mathbb{R}^{N\times N}$ of order M, which are polynomials of the graph-shift operator [3]:

$$\boldsymbol{H} \triangleq \sum_{m=0}^{M-1} h_m^o \boldsymbol{S}^m, \tag{1}$$

where $\boldsymbol{h}^o = \{h_m^o\}_{m=0}^{M-1}$ are the filter coefficients. One common filtering model assumes that the filtered graph signal $\boldsymbol{y}(i)$ is generated from the input graph signal $\boldsymbol{x}(i)$ as follows [3], [14]:

$$\boldsymbol{y}(i) = \boldsymbol{H}\boldsymbol{x}(i) + \boldsymbol{v}(i) = \sum_{m=0}^{M-1} h_m^o \boldsymbol{S}^m \boldsymbol{x}(i) + \boldsymbol{v}(i), \tag{2}$$

where $\boldsymbol{v}(i) = [v_1(i),\dots,v_N(i)]^\top \in \mathbb{R}^N$ denotes an i.i.d. zero-mean noise independent of any other signal and with covariance matrix $\boldsymbol{R}_v = \operatorname{diag}\{\sigma_{v,k}^2\}_{k=1}^N$.

The problem is to estimate the filter coefficients $\boldsymbol{h}^o$ adaptively from streaming input graph signals $\boldsymbol{x}(i)$ and output signals $\boldsymbol{y}(i)$. To this end, graph diffusion LMS strategies [15] can be employed. Nevertheless, most popular GSO are based on the graph Laplacian matrix or the adjacency matrix, which are not energy preserving. This may result in a large eigenvalue spread in the mean-square-error criterion to minimize and, consequently, a slow convergence speed of the diffusion LMS. To address this issue and improve the transient performance, we introduce a graph diffusion LMS-Newton algorithm. Centralized and distributed solutions are studied in Sections II and III,

The work of F. Hua was supported in part by China Scholarship Council and NSFC grant 61471298. The work of C. Richard was supported in part by Idex UCA-Jedi (ACADY project) and CNRS Mastodons (AGADIR project). The work of H. Wang was partly supported by NSFC under grants 61401364, 61571365, 61671386 and National Key R&D Program of China (2016YFC1400200). The work of A. H. Sayed was supported in part by NSF grants CCF-1524250 and ECCS-1407712.

ISBN 978-90-827970-1-5 © EURASIP 2018

respectively. A performance analysis is provided in Section IV, and simulation results are reported in Section V.

Notation: We use the symbol $\otimes$ to denote Kronecker operation and the symbol $\mathrm{Tr}(\cdot)$ to denote the trace operator. The symbol $\lambda_{\max}(\cdot)$ denotes the maximum eigenvalue of its matrix argument. The m-th entry of a vector $\boldsymbol{x}$ is denoted by $[\boldsymbol{x}]_m$, the (m,n)-th entry of a matrix $\boldsymbol{X}$ is denoted by $[\boldsymbol{X}]_{mn}$, the k-th row of a matrix $\boldsymbol{X}$ is $[\boldsymbol{X}]_{k,\bullet}$.

II. Centralized Solution

As explained in [15], model (2) is static and assumes the instantaneous diffusion of information. Specifically, relation (2) assumes that the powers of the shift operator, $\boldsymbol{S}^m$, are applied to the same graph vector $\boldsymbol{x}(i)$. This assumption is a serious limitation. To enrich the model, and to introduce a temporal dimension into (2), we consider the more general model introduced in [15]:

$$\boldsymbol{y}(i) = \sum_{m=0}^{M-1} h_m^o \boldsymbol{S}^m \boldsymbol{x}(i-m) + \boldsymbol{v}(i), \qquad i \geq M-1. \tag{3}$$

In this way, the shift $\boldsymbol{S}^m$ is carried out in m time slots. By retaining the following shifted signals at each node ℓ:

$$x_\ell(i-1), [\boldsymbol{S}\boldsymbol{x}(i-2)]_\ell, \ldots, [\boldsymbol{S}^{M-2}\boldsymbol{x}(i-M+1)]_\ell,$$

only one graph shift is required at each time i to carry out the filtered signal. Let $\boldsymbol{Z}(i)$ denote an $N \times M$ matrix given by:

$$\boldsymbol{Z}(i) \triangleq \Big[\boldsymbol{x}(i),\, \boldsymbol{S}\boldsymbol{x}(i-1), \ldots, \boldsymbol{S}^{M-1}\boldsymbol{x}(i-M+1)\Big], \tag{4}$$

then model (3) can be written alternatively:

$$\boldsymbol{y}(i) = \boldsymbol{Z}(i)\boldsymbol{h}^o + \boldsymbol{v}(i), \qquad i \geq M-1. \tag{5}$$

To estimate $\boldsymbol{h}^o$, we consider the mean-square-error criterion:

$$J(\boldsymbol{h}) = \mathbb{E}\|\boldsymbol{y}(i) - \boldsymbol{Z}(i)\boldsymbol{h}\|^2 \tag{6}$$

By setting the gradient vector of $J(\boldsymbol{h})$ to zero, the optimal parameter vector $\boldsymbol{h}^o$ can be found by solving:

$$\boldsymbol{R}_Z \boldsymbol{h}^o = \boldsymbol{r}_{Zy} \tag{7}$$

where, assuming stationarity, the $M \times M$ matrix $\boldsymbol{R}_Z$ and the $M \times 1$ vector $\boldsymbol{r}_{Zy}$ are given by:

$$\boldsymbol{R}_Z \triangleq \mathbb{E}\{\boldsymbol{Z}^\top(i)\boldsymbol{Z}(i)\}, \quad \boldsymbol{r}_{Zy} \triangleq \mathbb{E}\{\boldsymbol{Z}^\top(i)\boldsymbol{y}(i)\}. \tag{8}$$

In the sequel, we assume that $\boldsymbol{x}(i)$ is a zero-mean wide-sense stationary process, that is, $\mathbb{E}\{\boldsymbol{x}(i)\} = 0$ and its autocorrelation sequence $\boldsymbol{R}_x(m) \triangleq \mathbb{E}\{\boldsymbol{x}(i)\boldsymbol{x}^\top(i-m)\}$ is a function of the time lag m only. It can be checked that the (m,n)-th entry of the matrix $\boldsymbol{R}_Z$ is given by [15]:

$$[\boldsymbol{R}_Z]_{m,n} = \mathrm{Tr}\left((\boldsymbol{S}^{m-1})^\top \boldsymbol{S}^{n-1} \boldsymbol{R}_x(m-n)\right). \tag{9}$$

The m-th entry of the vector $\boldsymbol{r}_{Zy}$ is given by [15]:

$$[\boldsymbol{r}_{Zy}]_m = \mathrm{Tr}\left((\boldsymbol{S}^{m-1})^\top \boldsymbol{R}_{xy}(m)\right), \tag{10}$$

where $\boldsymbol{R}_{xy}(m) \triangleq \mathbb{E}\{\boldsymbol{y}(i)\boldsymbol{x}^\top(i-m)\}$ is the cross correlation function, which is independent of time i.

Instead of solving (7), $\boldsymbol{h}^o$ can be sought iteratively by using the gradient-descent method:

$$\boldsymbol{h}(i+1) = \boldsymbol{h}(i) + \mu\big[\boldsymbol{r}_{Zy} - \boldsymbol{R}_Z\boldsymbol{h}(i)\big], \tag{11}$$

with $\mu > 0$ a small step-size. Since the second-order moments are usually unavailable, one possible way is to replace them by the following instantaneous approximations $\boldsymbol{R}_Z \approx \boldsymbol{Z}^\top(i)\boldsymbol{Z}(i)$ and $\boldsymbol{r}_{Zy} \approx \boldsymbol{Z}^\top(i)\boldsymbol{y}(i)$. This yields the following LMS graph filter [15]:

$$\boldsymbol{h}(i+1) = \boldsymbol{h}(i) + \mu\boldsymbol{Z}^\top(i)\big[\boldsymbol{y}(i) - \boldsymbol{Z}(i)\boldsymbol{h}(i)\big]. \tag{12}$$

This stochastic-gradient algorithm is referred to as the *centralized graph-LMS* algorithm. In this centralized setting, each node at each time instant sends its data $\{x_k(i), y_k(i)\}$ to a fusion center which will update $\boldsymbol{h}(i)$ according to (12). Note that the step-size μ in (12) must satisfy $0 < \mu < \frac{2}{\lambda_{\max}(\boldsymbol{R}_Z)}$ in order to guarantee stability in the mean under certain independence conditions on the data [17].

Observe that $\boldsymbol{R}_Z$ in (8) is the correlation function of shifted graph signals. It is shown in [18] that graph shift operators do not preserve energy in general. This is due to the fact that the modulus of the eigenvalues of the shift operator $\boldsymbol{S}$ are not uniformly equal to 1. It is known that the convergence rate of the LMS is slow when the eigenvalue spread is large [17]. Based on diffusion strategies and pre-conditioning, we address this issue in the following with two algorithms that allow to estimate $\boldsymbol{h}^o$ efficiently in a fully distributed and adaptive manner. Compared with the standard graph diffusion LMS solution presented in [15], pre-conditioning will improve the convergence rate albeit at an increased computational cost.

III. Diffusion LMS Strategies over Graph Signals

From model (5), sample $y_k(i)$ at node k can be written as:

$$y_k(i) = \boldsymbol{z}_k^\top(i)\boldsymbol{h}^o + v_k(i), \;\text{ with }\; i \geq M-1, \tag{13}$$

where $\boldsymbol{z}_k^\top(i)$ is k-th row of $\boldsymbol{Z}(i)$ given by:

$$\boldsymbol{z}_k(i) \triangleq \mathrm{col}\{[\boldsymbol{x}(i)]_k,\, [\boldsymbol{S}\boldsymbol{x}(i-1)]_k, \ldots, [\boldsymbol{S}^{M-1}\boldsymbol{x}(i-M+1)]_k\}. \tag{14}$$

As explained before, it is important to note that by retaining the shifted signals $\{\boldsymbol{S}^{m-1}\boldsymbol{x}(i-m) : m = 1, \ldots, M-1\}$ at each node ℓ in the network from previous iterations, $\boldsymbol{z}_k(i)$ can be computed locally at node k from its one-hop neighbors at each iteration i. Let $\boldsymbol{R}_{z,k} \triangleq \mathbb{E}\{\boldsymbol{z}_k(i)\boldsymbol{z}_k^\top(i)\}$ denote the $M \times M$ covariance matrix with (m,n)-th entry given by [15]:

$$[\boldsymbol{R}_{z,k}]_{m,n} = \mathrm{Tr}\left([\boldsymbol{S}^{m-1})]_{k,\bullet}^\top [\boldsymbol{S}^{n-1}]_{k,\bullet} \boldsymbol{R}_x(m-n)\right). \tag{15}$$

Considering the mean-square-error cost $J_k(\boldsymbol{h})$ at node k:

$$J_k(\boldsymbol{h}) = \mathbb{E}|y_k(i) - \boldsymbol{z}_k^\top(i)\boldsymbol{h}|^2, \tag{16}$$

the global cost (6) can be written as:

$$J(\boldsymbol{h}) = \sum_{k=1}^{N} J_k(\boldsymbol{h}). \tag{17}$$

Diffusion strategies minimize (17) in an adaptive and fully distributed manner [12]. In particular, the adapt-then-combine

ISBN 978-90-827970-1-5 © EURASIP 2018

(ATC) diffusion LMS takes the following form at node k [15]:

$$\boldsymbol{\psi}_k(i+1) = \boldsymbol{h}_k(i) + \mu_k \boldsymbol{z}_k(i)\big[y_k(i) - \boldsymbol{z}_k^\top(i)\boldsymbol{h}_k(i)\big], \quad (18a)$$

$$\boldsymbol{h}_k(i+1) = \sum_{\ell\in\mathcal{N}_k} a_{\ell k}\boldsymbol{\psi}_\ell(i+1), \quad (18b)$$

where $\mu_k > 0$ is a local step-size parameter and $\{a_{\ell k}\}$ are non-negative combination coefficients chosen to satisfy:

$$a_{\ell k} > 0, \quad \sum\nolimits_{\ell=1}^{N} a_{\ell k} = 1, \text{ and } a_{\ell k} = 0 \text{ if } \ell \notin \mathcal{N}_k. \quad (19)$$

This implies that the matrix $\boldsymbol{A}$ with (ℓ, k)-th entry $a_{\ell k}$ is a left-stochastic matrix, which means that the sum of each of its columns is equal to 1. In the first step (18a), which corresponds to the adaptation step, each node k uses the data from its one-hop neighbors to compute $\boldsymbol{z}_k(i)$, then updates its local estimate $\boldsymbol{h}_k(i)$ to an intermediate estimate $\boldsymbol{\psi}_k(i+1)$. In the second step (18b), which is the combination step, node k aggregates all the intermediate estimates $\boldsymbol{\psi}_\ell(i+1)$ from its neighbors to obtain the updated estimate $\boldsymbol{h}_k(i+1)$.

From (14), we observe that the regressors $\boldsymbol{z}_k(i)$ used in the adaptation step (18a) result from shifted graph signals where the shift matrix $\boldsymbol{S}$ is not energy preserving in general. In this case, the eigenvalue spread of $\boldsymbol{R}_{z,k}$ may be large and the LMS update may suffer from slow convergence speed. One way to address this issue, albeit at an increased computational complexity, is to employ a form of Newton's method, i.e.,

$$\boldsymbol{\psi}_k(i{+}1) = \boldsymbol{h}_k(i) {-} \mu_k[\nabla^2_{\boldsymbol{h}} J_k(\boldsymbol{h}_k(i))]^{-1}[\nabla_{\boldsymbol{h}} J_k(\boldsymbol{h}_k(i))], \quad (20)$$

where $\nabla^2_{\boldsymbol{h}} J_k(\cdot)$ is the Hessian matrix of $J_k(\cdot)$ and $\nabla_{\boldsymbol{h}} J_k(\cdot)$ is its gradient vector. For the quadratic cost function (16), the adaptation step (20) can be written as:

$$\boldsymbol{\psi}_k(i+1) = \boldsymbol{h}_k(i) + \mu_k \boldsymbol{R}_{z,k}^{-1}\big[\boldsymbol{r}_{zy,k} - \boldsymbol{R}_{z,k}\boldsymbol{h}_k(i)\big], \quad (21)$$

where $\boldsymbol{r}_{zy,k} = \mathbb{E}\{\boldsymbol{z}_k(i)y_k(i)\}$. Note that the second term on the RHS of (21) requires second-order moments that are rarely known beforehand. We replace $\boldsymbol{r}_{zy,k} - \boldsymbol{R}_{z,k}\boldsymbol{h}_k(i)$ by the instantaneous approximation:

$$\boldsymbol{r}_{zy,k} - \boldsymbol{R}_{z,k}\boldsymbol{h}_k(i) \approx \boldsymbol{z}_k(i)e_k(i) \quad (22)$$

where $e_k(i) = y_k(i) - \boldsymbol{z}_k^\top(i)\boldsymbol{h}_k(i)$. The LMS-Newton for the adaptation step (21) can be written as:

$$\boldsymbol{\psi}_k(i+1) = \boldsymbol{h}_k(i) + \mu_k \widehat{\boldsymbol{R}}_{z,k}^{-1}(i)\boldsymbol{z}_k(i)e_k(i), \quad (23)$$

where $\widehat{\boldsymbol{R}}_{z,k}^{-1}(i)$ can be obtained recursively. Since most popular GSOs are not energy preserving, $\boldsymbol{R}_{z,k}$ may be close to singular. The inverse $\boldsymbol{R}_{z,k}^{-1}$ then would be ill-conditioned and may have undesirable effects. To address this problem, it is common to employ regularization [17]. We obtain the diffusion LMS-Newton algorithm:

$$\boldsymbol{\psi}_k(i+1) = \boldsymbol{h}_k(i) + \mu_k\big[\epsilon\boldsymbol{I} + \widehat{\boldsymbol{R}}_{z,k}(i)\big]^{-1}\boldsymbol{z}_k(i)e_k(i), \quad (24a)$$

$$\boldsymbol{h}_k(i+1) = \sum_{\ell\in\mathcal{N}_k} a_{\ell k}\boldsymbol{\psi}_\ell(i+1), \quad (24b)$$

with $\epsilon \geq 0$ a small regularization parameter. Compared with the diffusion LMS algorithm (18), the LMS-Newton method (24) requires the computation of $\big[\epsilon\boldsymbol{I} + \widehat{\boldsymbol{R}}_{z,k}(i)\big]^{-1}$. This algorithm can produce better performance as shown in the sequel, but at the expense of additional computation.

In order to reduce the computational complexity of the LMS-Newton algorithm, we propose to use a preconditioning matrix $\boldsymbol{P}_k$ that is independent of the graph signal $\boldsymbol{x}(i)$ in the adaptation step, instead of the Hessian matrix $\boldsymbol{R}_{z,k}$ or its estimate $\widehat{\boldsymbol{R}}_{z,k}$. Since the large eigenvalue spread of the input covariance matrix $\boldsymbol{R}_{z,k}$ results mainly from the shift matrix $\boldsymbol{S}$ and the filter order M, we construct the $M\times M$ preconditioning matrix $\boldsymbol{P}_k$ from the local knowledge of $\boldsymbol{S}$ and M as follows:

$$\boldsymbol{P}_k \triangleq \operatorname{diag}\{\|[\boldsymbol{S}^{(m-1)}]_{k,\bullet}\|^2\}_{m=1}^{M}. \quad (25)$$

The rationale behind (25) is that, in the case where $\boldsymbol{x}(i)$ is i.i.d. with variance σ^2, it follows from (15) that $\boldsymbol{R}_{z,k} = \sigma^2\boldsymbol{P}_k$. The main advantages of $\boldsymbol{P}_k$ over $\boldsymbol{R}_{z,k}$ are that it is a diagonal matrix, independent of $\boldsymbol{x}(i)$, and it can be evaluated beforehand at each node k during an initial step by using the information from its M-hop neighbors. Following the same line of reasoning as for the LMS-Newton method, we arrive at the following preconditioned graph diffusion LMS strategy:

$$\boldsymbol{\psi}_k(i+1) = \boldsymbol{h}_k(i) + \mu_k\boldsymbol{D}_k\boldsymbol{z}_k(i)e_k(i), \quad (26a)$$

$$\boldsymbol{h}_k(i+1) = \sum_{\ell\in\mathcal{N}_k} a_{\ell k}\boldsymbol{\psi}_\ell(i+1). \quad (26b)$$

with

$$\boldsymbol{D}_k = (\epsilon\boldsymbol{I} + \boldsymbol{P}_k)^{-1}, \quad (27)$$

where the regularization term $\epsilon\boldsymbol{I}$ has also been added to $\boldsymbol{P}_k$. At each iteration i, node k uses the local information to update the intermediate estimate $\boldsymbol{\psi}_k(i{+}1)$ in the adaptation step (26a). Then, in the combination step (26b), the intermediate estimates $\boldsymbol{\psi}_\ell(i{+}1)$ from the neighborhood of node k are combined to get $\boldsymbol{h}_k(i+1)$. Although the preconditioning matrix here is not the true Hessian matrix, we prove in Section IV that the algorithm converges to the optimal solution $\boldsymbol{h}^o$ provided that it is stable.

IV. Stochastic Behavior analysis

We analyze the transient and steady-state performance of the diffusion preconditioned LMS (PLMS) algorithm (26). We denote by $\tilde{\boldsymbol{h}}(i) \triangleq \operatorname{col}\{\boldsymbol{h}^o - \boldsymbol{h}_k(i)\}_{k=1}^N$ the network error vector. For tractability, we introduce the following assumption.

Assumption 1. *The regressors $\boldsymbol{z}_k(i)$ arise from a zero-mean random process that is temporally white.*

Although the independence assumption does not hold here, it turns out that the resulting expressions match well the simulation results for sufficiently small step-sizes. This independence assumption is commonly used in adaptive filtering literature [17], [19] since it helps to simplify the derivations without constraining the conclusions.

A. Mean-error behavior analysis

Using the data model (13), the network error vector $\tilde{\boldsymbol{h}}(i)$ corresponding to algorithm (26) evolves according to [19]:

$$\tilde{\boldsymbol{h}}(i+1) = \boldsymbol{\mathcal{B}}(i)\tilde{\boldsymbol{h}}(i) - \boldsymbol{\mathcal{A}}^\top \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{D}}\boldsymbol{p}_{zv}(i) \tag{28}$$

where $\boldsymbol{\mathcal{B}}(i) \triangleq \boldsymbol{\mathcal{A}}^\top(\boldsymbol{I}_{MN} - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{R}}_z(i))$, $\boldsymbol{\mathcal{A}} \triangleq \boldsymbol{A} \otimes \boldsymbol{I}_M$, $\boldsymbol{\mathcal{M}} \triangleq \text{diag}\{\mu_k \boldsymbol{I}_M\}_{k=1}^N$, $\boldsymbol{\mathcal{D}} = \text{diag}\{\boldsymbol{D}_k\}_{k=1}^N$, $\boldsymbol{\mathcal{R}}_z(i) \triangleq \text{diag}\{\boldsymbol{z}_k(i)\boldsymbol{z}_k^\top(i)\}_{k=1}^N$, and $\boldsymbol{p}_{zv}(i) \triangleq \text{col}\{\boldsymbol{z}_k(i)v_k(i)\}_{k=1}^N$.

Taking expectations of both sides of (28), using the fact that $\mathbb{E}\{\boldsymbol{p}_{zv}(i)\} = 0$, and applying the independence Assumption 1, we obtain:

$$\mathbb{E}\tilde{\boldsymbol{h}}(i+1) = \boldsymbol{\mathcal{B}}\mathbb{E}\tilde{\boldsymbol{h}}(i), \tag{29}$$

where $\boldsymbol{\mathcal{B}} \triangleq \boldsymbol{\mathcal{A}}^\top(\boldsymbol{I}_{MN} - \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{R}}_z)$ and $\boldsymbol{\mathcal{R}}_z \triangleq \text{diag}\{\boldsymbol{R}_{z,k}\}_{k=1}^N$. From (29), we observe that algorithm (26) converges asymptotically in the mean toward the optimal vector $\boldsymbol{h}^o$ if, and only if, $\boldsymbol{\mathcal{B}}$ is stable, namely, all its eigenvalues lie strictly inside the unit disc. It turns out that, when the signal $\boldsymbol{x}(i)$ is i.i.d, the stability of $\boldsymbol{\mathcal{B}}$ is ensured by choosing μ_k according to:

$$0 < \mu_k < \frac{2}{\lambda_{\max}(\boldsymbol{D}_k\boldsymbol{R}_{z,k})}, \qquad k = 1, \ldots, N. \tag{30}$$

B. Mean-square-error behavior analysis

We now study the mean-square-error behavior of algorithm (26). From the independence Assumption 1, it can be verified that the weighted mean-square-error $\mathbb{E}\|\tilde{\boldsymbol{h}}(i)\|^2_{\boldsymbol{\Sigma}}$ satisfies the following relation:

$$\mathbb{E}\|\tilde{\boldsymbol{h}}(i+1)\|^2_{\boldsymbol{\Sigma}} = \mathbb{E}\|\tilde{\boldsymbol{h}}(i)\|^2_{\boldsymbol{\Sigma}'} + \mathbb{E}\|\boldsymbol{\mathcal{A}}^\top\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{D}}\boldsymbol{p}_{zv}(i)\|^2_{\boldsymbol{\Sigma}} \tag{31}$$

where $\boldsymbol{\Sigma}$ is a positive semi-definite matrix that we are free to choose, and $\boldsymbol{\Sigma}' \triangleq \mathbb{E}\{\boldsymbol{\mathcal{B}}^\top(i)\boldsymbol{\Sigma}\boldsymbol{\mathcal{B}}(i)\}$.

The second term on the RHS of (31) can be written as:

$$\mathbb{E}\|\boldsymbol{\mathcal{A}}^\top\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{D}}\boldsymbol{p}_{zv}(i)\|^2_{\boldsymbol{\Sigma}} = \text{Tr}(\boldsymbol{\Sigma}\boldsymbol{\mathcal{G}}) \tag{32}$$

where $\boldsymbol{\mathcal{G}} \triangleq \boldsymbol{\mathcal{A}}^\top\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{S}}\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{S}} \triangleq \mathbb{E}\{\boldsymbol{p}_{zv}(i)\boldsymbol{p}_{zv}^\top(i)\} = \text{diag}\{\sigma^2_{v,k}\boldsymbol{R}_{z,k}\}_{k=1}^N$. Let $\boldsymbol{\sigma} = \text{vec}(\boldsymbol{\Sigma})$ and $\boldsymbol{\sigma}' = \text{vec}(\boldsymbol{\Sigma}')$. The notation $\|\boldsymbol{h}\|^2_{\boldsymbol{\sigma}}$ is used to refer to the same quantity $\|\boldsymbol{h}\|^2_{\boldsymbol{\Sigma}}$. It can be verified that $\boldsymbol{\sigma}' = \boldsymbol{\mathcal{F}}\boldsymbol{\sigma}$, where $\boldsymbol{\mathcal{F}} \triangleq \mathbb{E}\{\boldsymbol{\mathcal{B}}^\top(i) \otimes \boldsymbol{\mathcal{B}}^\top(i)\}$. For sufficiently small step-sizes, by neglecting the influence of higher-order terms, $\boldsymbol{\mathcal{F}}$ can be approximated by $\boldsymbol{\mathcal{F}} \approx \boldsymbol{\mathcal{B}}^\top \otimes \boldsymbol{\mathcal{B}}^\top$. Using the property $\text{Tr}(\boldsymbol{\Sigma}\boldsymbol{W}) = [\text{vec}(\boldsymbol{W}^\top)]^\top\text{vec}(\boldsymbol{\Sigma})$, the variance relation (31) can be written as:

$$\mathbb{E}\|\tilde{\boldsymbol{h}}(i+1)\|^2_{\boldsymbol{\sigma}} = \mathbb{E}\|\tilde{\boldsymbol{h}}(i)\|^2_{\boldsymbol{\mathcal{F}}\boldsymbol{\sigma}} + [\text{vec}(\boldsymbol{\mathcal{G}}^\top)]^\top\boldsymbol{\sigma}. \tag{33}$$

Observe from the above expression that the variance $\mathbb{E}\|\tilde{\boldsymbol{h}}(i)\|^2_{\boldsymbol{\sigma}}$ converges if $\boldsymbol{\mathcal{F}}$ is stable [19]. Under the approximation $\boldsymbol{\mathcal{B}}^\top \otimes \boldsymbol{\mathcal{B}}^\top$, the stability of $\boldsymbol{\mathcal{F}}$ is ensured by choosing μ_k according to (30). Iterating (33) starting from $i = 0$, we obtain:

$$\mathbb{E}\|\tilde{\boldsymbol{h}}(i+1)\|^2_{\boldsymbol{\sigma}} = \mathbb{E}\|\tilde{\boldsymbol{h}}(0)\|^2_{\boldsymbol{\mathcal{F}}^{i+1}\boldsymbol{\sigma}} + [\text{vec}(\boldsymbol{\mathcal{G}}^\top)]^\top\sum_{j=0}^{i}\boldsymbol{\mathcal{F}}^j\boldsymbol{\sigma}. \tag{34}$$

Comparing (34) at time $i+1$ and i, we obtain the transient learning recursion:

$$\begin{aligned}\mathbb{E}\|\tilde{\boldsymbol{h}}(i+1)\|^2_{\boldsymbol{\sigma}} &= \mathbb{E}\|\tilde{\boldsymbol{h}}(i)\|^2_{\boldsymbol{\sigma}} + [\text{vec}(\boldsymbol{\mathcal{G}}^\top)]^\top\boldsymbol{\mathcal{F}}^i\boldsymbol{\sigma} \\ &+ [\text{vec}(\mathbb{E}\{\tilde{\boldsymbol{h}}(0)\tilde{\boldsymbol{h}}^\top(0)\})]^\top(\boldsymbol{\mathcal{F}} - \boldsymbol{I})\boldsymbol{\mathcal{F}}^i\boldsymbol{\sigma}.\end{aligned} \tag{35}$$

Consider the steady-state network mean-square-deviation (MSD) defined as:

$$\zeta = \lim_{i\to\infty}\frac{1}{N}\mathbb{E}\{\|\tilde{\boldsymbol{h}}(i)\|^2\}. \tag{36}$$

If $\boldsymbol{\mathcal{F}}$ is stable, we obtain from (33) as $i \to \infty$:

$$\lim_{i\to\infty}\mathbb{E}\|\tilde{\boldsymbol{h}}(i)\|^2_{(\boldsymbol{I}-\boldsymbol{\mathcal{F}})\boldsymbol{\sigma}} = [\text{vec}(\boldsymbol{\mathcal{G}}^\top)]^\top\boldsymbol{\sigma}. \tag{37}$$

Replacing $\boldsymbol{\sigma}$ in (37) by $\frac{1}{N}(\boldsymbol{I} - \boldsymbol{\mathcal{F}})^{-1}\ \text{vec}(\boldsymbol{I})$, we obtain the steady-state network MSD:

$$\zeta = \frac{1}{N}[\text{vec}(\boldsymbol{\mathcal{G}}^\top)]^\top(\boldsymbol{I} - \boldsymbol{\mathcal{F}})^{-1}\ \text{vec}(\boldsymbol{I}). \tag{38}$$

This concludes the analysis of algorithm (26). Note that we also studied algorithm (24) but, due to space limitations, we are not able to include the theoretical findings. However, for illustration purposes, the theoretical curves obtained from this analysis are reported in all the plots of Section V.

V. Simulations

We applied the diffusion LMS algorithm (18), the diffusion LMS-Newton (LMSN) algorithm (24), and the diffusion preconditioned LMS (PLMS) algorithm (26) to estimate graph filter coefficients using different types of shift operators $\boldsymbol{S}$. An undirected weighted sensor network of $N = 20$ nodes was generated using GSPBOX [20]. Each node was connected to its 5 nearest neighbors. We assumed that the graph signal process $\boldsymbol{x}(i)$ is i.i.d. zero-mean Gaussian with covariance matrix $\boldsymbol{R}_x = \text{diag}\{\sigma^2_{x,k}\}_{k=1}^N$ where the variances $\sigma^2_{x,k}$ were randomly generated from the uniform distribution $\mathcal{U}(1, 1.5)$. The noise $\boldsymbol{v}(i)$ was zero-mean Gaussian with covariance matrix $\boldsymbol{R}_v = \text{diag}\{\sigma^2_{v,k}\}_{k=1}^N$. The variances $\sigma^2_{v,k}$ were randomly generated from the uniform distribution $\mathcal{U}(0.1, 0.15)$. The graph filter coefficients h^o_m were randomly generated from the uniform distribution $\mathcal{U}(0, 1)$. We assumed the linear data model (3). All simulated results were averaged over 200 Monte-Carlo runs.

In the first experiment, we adopted the normalized adjacency matrix as shift operator [3], [21], i.e., $\boldsymbol{S} = \frac{\boldsymbol{A}_w}{1.5\lambda_{\max}(\boldsymbol{A}_w)}$, where $\boldsymbol{A}_w$ is the weighted adjacency matrix. In this case, all the eigenvalues of $\boldsymbol{S}$ are smaller than 1. Thus, the energy of the shifted signal $\boldsymbol{S}^m\boldsymbol{x}$ diminishes for large m. We set $M = 3$. In this case, the smallest eigenvalue $\lambda_{\min}(\boldsymbol{R}_{z,k})$ was very small, and, for some node, it was close to 0. For the LMSN and PLMS algorithms, a large regularization strength ϵ results in a slow convergence rate and a small MSD, while a small ϵ results in a larger MSD but a fast convergence rate. We set $\epsilon = 0.01$ in this experiment for both algorithms. We ran algorithms (18), (24), and (26) by setting $a_{\ell,k} = \frac{1}{|\mathcal{N}_k|}$ for $\ell \in \mathcal{N}_k$. We used uniform step-size for all nodes, i.e., $\mu_k = \mu$ for all k, and we set $\mu = 0.13$ for the LMS, $\mu = 0.0115$ for the LMSN, and $\mu = 0.01$ for the PLMS. The network MSD performance of each algorithm is reported in Fig. 1 (left). The theoretical transient and steady-state MSD are also reported. Observe that the diffusion LMSN and PLMS algorithms converge faster than

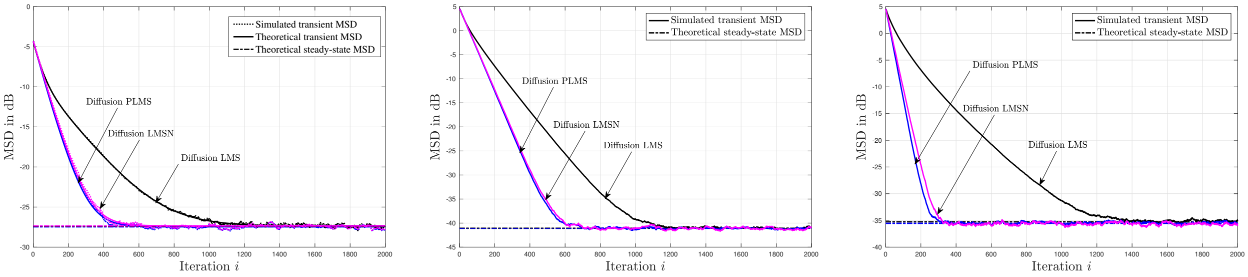

Fig. 1: Network MSD performance for different types of shift operator. *(Left)* Normalized Adjacency Matrix. *(Middle)* Normalized Laplacian Matrix. *(Right)* Adjacency Matrix.

the LMS algorithm. Also, note that the theoretical results match well the simulated curves.

In the second experiment, we used the normalized graph Laplacian matrix [2] defined as $\boldsymbol{S} = \boldsymbol{D}^{-\frac{1}{2}}\mathcal{L}\boldsymbol{D}^{-\frac{1}{2}}$ where $\boldsymbol{D}$ is the degree matrix and $\mathcal{L}$ is the graph Laplacian matrix. The filter order was set to $M = 5$. In this case, $\lambda_{\max}(\boldsymbol{R}_{z,k})$ was large for all nodes. Therefore, for the diffusion LMS algorithm, the step-size was chosen relatively small to guarantee convergence. We set $\mu = 0.004$ for the LMS, $\mu = 0.01$ for the LMSN, and $\mu = 0.008$ for the PLMS. Figure 1 (middle) shows that, for the same steady-state MSD, LMSN and PLMS perform better than LMS in terms of convergence rate.

In the last scenario, the adjacency matrix $\boldsymbol{A}_w$ was chosen as a shift operator. We set $M = 5$. For the LMS update, the step-size was chosen as $\mu_k = 0.05 \cdot \frac{2}{\lambda_{\max}(\boldsymbol{R}_{z,k})}$ for node k. For the LMSN and the PLMS, we used a uniform step-size $\mu = 0.018$ and $\mu = 0.016$, respectively. As shown in Fig. 1 (right), LMSN and PLMS algorithms converge faster than the LMS solution.

VI. Conclusion

In this paper, diffusion LMS strategies were employed to estimate graph filter coefficients in an adaptive and distributed manner. Particularly, a diffusion LMS-Newton algorithm was proposed to improve the convergence rate of existing diffusion LMS based strategies that may suffer from large eigenvalue spread of the input signals due to the use of non-energy preserving graph shift operators. Furthermore, a computationally efficient preconditioned LMS was devised and its stochastic performance analysis was provided. Simulation results validated the theoretical models and showed the efficiency of the proposed algorithms.

References

[1] A. Ortega, P. Frossard, J. Kovačević, J. M. Moura, and P. Vandergheynst, "Graph signal processing," *arXiv preprint arXiv:1712.00468*, 2017.

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

[3] A. Sandryhaila and J. M. Moura, "Discrete signal processing on graphs," *IEEE Transactions on Signal Processing*, vol. 61, no. 7, pp. 1644–1656, 2013.

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

[5] A. Anis, A. Gadde, and A. Ortega, "Efficient sampling set selection for bandlimited graph signals using graph spectral proxies," *IEEE Transactions on Signal Processing*, vol. 64, no. 14, pp. 3775–3789, 2016.

[6] P. Di Lorenzo, P. Banelli, and S. Barbarossa, "Optimal sampling strategies for adaptive learning of graph signals," in *Proc. IEEE 25th European Signal Processing Conference (EUSIPCO)*, 2017, pp. 1684–1688.

[7] B. Girault, "Stationary graph signals using an isometric graph translation," in *Proc. IEEE 23rd European Signal Processing Conference (EUSIPCO)*, 2015, pp. 1516–1520.

[8] A. G. Marques, S. Segarra, G. Leus, and A. Ribeiro, "Stationary graph processes and spectral estimation," *IEEE Transactions on Signal Processing*, vol. 65, no. 22, pp. 5911–5926, 2016.

[9] N. Perraudin and P. Vandergheynst, "Stationary signal processing on graphs," *IEEE Transactions on Signal Processing*, vol. 65, no. 13, pp. 3462–3477, 2017.

[10] E. Isufi, A. Loukas, A. Simonetto, and G. Leus, "Autoregressive moving average graph filtering," *IEEE Transactions on Signal Processing*, vol. 65, no. 2, pp. 274–288, 2017.

[11] A. H. Sayed, S.-Y. Tu, J. Chen, X. Zhao, and Z. J. Towfic, "Diffusion strategies for adaptation and learning over networks: an examination of distributed strategies and network behavior," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 155–171, 2013.

[12] A. H. Sayed, "Adaptive networks," *Proceedings of the IEEE*, vol. 102, no. 4, pp. 460–497, 2014.

[13] F. Grassi, A. Loukas, N. Perraudin, and B. Ricaud, "A time-vertex signal processing framework," *arXiv preprint arXiv:1705.02307*, 2017.

[14] R. Nassif, C. Richard, J. Chen, and A. H. Sayed, "A graph diffusion LMS strategy for adaptive graph signal processing," in *Proc. Asilomar Conference on Signals, Systems, and Computers*, 2017, pp. 1–4.

[15] ——, "Distributed diffusion adaptation over graph signals," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 1–5.

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

[17] ——, *Adaptive Filters*. John Wiley & Sons, 2008.

[18] A. Gavili and X.-P. Zhang, "On the shift operator, graph frequency, and optimal filtering in graph signal processing," *IEEE Transactions on Signal Processing*, vol. 65, no. 23, pp. 6303–6318, 2017.

[19] A. H. Sayed, "Diffusion adaptation over networks," in *Academic Press Library in Signal Processing*, S. Theodoridis and R. Chellappa, Eds. Academic Press, Elsevier, 2013, vol. 3, pp. 322–454.

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

[21] J. Mei and J. M. Moura, "Signal processing on graphs: Causal modeling of unstructured data," *IEEE Transactions on Signal Processing*, vol. 65, no. 8, pp. 2077–2092, 2017.