# Autoregressive Moving Average Graph Filters a Stable Distributed Implementation

Isufi, Elvin; Loukas, Andreas; Leus, Geert

**DOI**
10.1109/ICASSP.2017.7952931

**Publication date**
2017

**Document Version**
Final published version

**Published in**
2017 IEEE International Conference on Acoustics, Speech, and Signal Processing - Proceedings

# AUTOREGRESSIVE MOVING AVERAGE GRAPH FILTERS A STABLE DISTRIBUTED IMPLEMENTATION

*Elvin Isufi*[†]*, Andreas Loukas*[‡] *and Geert Leus*[†]

[†] Circuits and Systems Group, Delft University of Technology, The Netherlands
[‡] Swiss Federal Insitute of Technology in Lausanne (EPFL), Switzerland
e-mails: {e.isufi-1, g.j.t.leus}@tudelft.nl, andreas.loukas@epfl.ch

**ABSTRACT**

We present a novel implementation strategy for distributed autoregressive moving average (ARMA) graph filters. Differently from the state of the art implementation, the proposed approach has the following benefits: (i) the designed filter coefficients come with stability guarantees, (ii) the linear convergence time can now be controlled by the filter coefficients, and (iii) the stable filter coefficients that approximate a desired frequency response are optimal in a least squares sense. Numerical results show that the proposed implementation outperforms the state of the art distributed infinite impulse response (IIR) graph filters. Further, even at fixed distributed costs, compared with the popular finite impulse response (FIR) filters, at high orders our method achieves tighter low-pass responses, suggesting that it should be preferable in accuracy-demanding applications.

***Index Terms*—** graph signal processing, graph filters, autoregressive moving average graph filters.

## 1. INTRODUCTION

Graphs are a useful tool to mathematically represent pairwise relationships between entities of a network, such as in social, sensor or biological networks. The field of signal processing on graphs [1], focuses on processing signals (data) that live on top of these networks. Specifically, classical tools used in processing time and space signals, like the Fourier transform and filters [2], have now a well established extension to these signals that have an irregular underlying support. While the *graph Fourier transform* gives a specific definition to the *graph frequency* content of a signal [3], *graph filters* are the tools that can shape the graph signal spectrum.

Graph filters find applications in signal denoising and smoothing [4,5], interpolation [6] and solving diffusion tasks [7]. They also enjoy a distributed implementation and have been used to distributively perform linear operations like finite time consensus and network coding [8]. Generally, these tasks are solved with the so-called finite impulse response (FIR) graph filter, i.e., the analog of the homonym temporal filter characterized by a polynomial frequency response, but now applied in the graph frequency domain.

In this work, we will focus on distributed implementation of autoregressive moving average (ARMA) graph filters [9]. Due to their rational frequency response, these filters are particularly useful for diffusion filtering [7], solving Tikhonov denoising problems [1] and interpolation under smoothness prior [6, 10]. Further, they are robust to deterministic [9] and stochastic [11] time-variations in both the graph topology and graph signal. However, distributed ARMA filters achieve their designed frequency response theoretically only after infinite iterations [9], thus two main challenges need to be addressed: (i) the guarantee that the filter will converge (i.e., stability) and (ii) the convergence time.

Differently from our previous implementation of ARMA graph filter implementation [9], in this paper we propose a novel implementation strategy where (i) the filter coefficients are designed to ensure stability, (ii) it preserves the linear convergence property of [9], and (iii) it has the potential to achieve rational frequency responses of different orders in the numerator and denominator. Further, in the proposed filter we have a handle to control the convergence time and we also came up with a design method which finds the optimal stable filter coefficients in a least squares sense.

Our numerical results show that the proposed implementation outperforms the distributed IIR filters [9, 12] and also for high filter orders it performs better than distributed FIR graph filter even at fixed distributed complexity.

## 2. BACKGROUND

In this section, we first recall the basics of signal processing on graphs and then discuss the state of the art distributed FIR and ARMA graph filters.

**Basics.** Consider an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ indicates the set of $N$ vertices and $\mathcal{E}$ the set of $M$ edges. We indicate with $\boldsymbol{x}$ the $N \times 1$ vector, where the $i$-th entry $x_i \in \mathbb{C}$ lives on the $i$-th node of $\mathcal{G}$. We will commonly refer to $\boldsymbol{x}$ as the *graph signal*.

The local structure of $\mathcal{G}$ is captured by the *graph shift* operator $\boldsymbol{S}$, defined as an $N \times N$ symmetric matrix with $S_{i,j} \neq 0$ if there exists a connection between the nodes $i$ and $j$, or differently $(i, j) \in \mathcal{E}$. Common choices of $\boldsymbol{S}$ are the adjacency matrix of the graph [2], the graph discrete and normalized Laplacian [1] or their translated versions [9]. For convenience let us indicate with $\varrho$ the spectral norm of the shift operator matrix, i.e., $\|\boldsymbol{S}\| = \varrho$.

Due to its symmetric structure, $\boldsymbol{S}$ can always be decomposed as $\boldsymbol{S} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{\mathsf{H}}$, where the eigenvectors $\boldsymbol{U} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n]$ and eigenvalues $\boldsymbol{\Lambda} = \text{diag}[\lambda_1, \ldots, \lambda_N]$ carry the notion of frequency in the graph setting [1,2]. The eigenvalues $\{\lambda_n\}_{n=1}^N$ indicate the graph frequencies and the eigenvector matrix $\boldsymbol{U}$ is used as the Fourier basis for the graph spectral analysis of $\boldsymbol{x}$. Specifically, the graph Fourier transform $\hat{\boldsymbol{x}}$ of $\boldsymbol{x}$ and its inverse are respectively calculated as $\hat{\boldsymbol{x}} = \boldsymbol{U}^{\mathsf{H}}\boldsymbol{x}$ and $\boldsymbol{x} = \boldsymbol{U}\hat{\boldsymbol{x}}$.

**Distributed FIR graph filters.** An FIR filter of order $K$ ($\text{FIR}_K$) can be defined as a $K$-th order polynomial of $\boldsymbol{S}$, which can be implemented distributively due to the local structure of $\boldsymbol{S}$. The output signal and the graph frequency response of the filter are respectively given by

$$\boldsymbol{y} = \sum_{k=0}^{K} \varphi_k \boldsymbol{S}^k \boldsymbol{x} \quad \text{and} \quad h(\lambda_n) = \sum_{k=0}^{K} \varphi_k \lambda_n^k, \tag{1}$$

where $\varphi_k$ are the filter coefficients. Since $\boldsymbol{S}^K\boldsymbol{x} = \boldsymbol{S}(\boldsymbol{S}^{K-1}\boldsymbol{x})$, each

node can compute the $K$-th term from the values of the $(K-1)$-th terms in its neighborhood. In a distributed implementation, the overall communication and computational complexity is $O(MK)$. For time-varying graph signals, extensions to 2-dimensional FIR filters are proposed in [13–15].

**Distributed ARMA graph filters.** We can obtain an ARMA graph filter of order $K$ (ARMA$_K$) using a parallel bank of $K$ ARMA$_1$ graph filters [9]. Let us denote with the superscript $(k)$ the corresponding terms of the $k$-th branch for $k = 1, 2, \ldots, K$. Then, given a graph signal $\boldsymbol{x}$, the output of the $k$-th branch $\boldsymbol{y}_t^{(k)}$ and the filter output $\boldsymbol{y}_t$ at time instant $t$ are

$$\boldsymbol{y}_t^{(k)} = \psi^{(k)} \boldsymbol{S} \boldsymbol{y}_{t-1}^{(k)} + \varphi^{(k)} \boldsymbol{x} \quad \text{and} \quad \boldsymbol{y}_t = \sum_{k=1}^{K} \boldsymbol{y}_t^{(k)}, \tag{2}$$

for arbitrary $\boldsymbol{y}_0^{(k)}$, and where $\psi^{(k)}$ and $\varphi^{(k)}$ are the complex filter coefficients. From [9], the graph frequency response of the parallel ARMA$_K$ filter is

$$h(\lambda_n) = \sum_{k=1}^{K} \frac{r_k}{\lambda_n - p_k} \quad \text{subject to} \quad |p_k| > \varrho, \tag{3}$$

with residuals $r_k = -\varphi^{(k)}/\psi^{(k)}$ and poles $p_k = 1/\psi^{(k)}$. In contrast to the FIR filters, where the output is obtained after $K$ time instants[1], recursion (2) will achieve the frequency response (3) theoretically at infinity, yet it is characterized by a linear convergence in practice. In a distributed implementation, to obtain the filter output after $T$ iterations, the communication and computational complexity of the ARMA$_K$ is $O(TMK)$. Extensions of distributed ARMA filters for time-varying graph signals are proposed in [16].

The parallel design suffers from two main drawbacks: It is challenging to perform the filter design with theoretical guarantees that always ensure a stable implementation of (2). Especially, this issue is enhanced for shift operators $\boldsymbol{S}$ with a high spectral radius $\varrho$. Further, the convergence time of the ARMA$_K$ cannot be controlled and it is directly imposed by the filter coefficients. What we propose next is a novel distributed implementation of ARMA filters, which can be designed with theoretical guarantees of convergence and where the convergence time can now be controlled by the filter coefficients.

## 3. PROPOSED SOLUTION

This section contains our proposed solution to implement a distributed ARMA graph filter. We first introduce the new implementation algorithm and derive theoretical results on the convergence time. Then, we propose a filter design strategy that ensures algorithm stability while also allowing to trade off convergence time with approximation accuracy. We conclude the section by analyzing the distributed implementation cost of the algorithm,

Consider the recursion

$$\boldsymbol{y}_t = -\sum_{p=1}^{P} \psi_p \boldsymbol{S}^p \boldsymbol{y}_{t-1} + \sum_{q=0}^{Q} \varphi_q \boldsymbol{S}^q \boldsymbol{x} \tag{4}$$

where $\psi_p$ and $\varphi_q$ will again indicate the complex filter coefficients. In contrast to the implementation (2), in computing $\boldsymbol{y}_t$ now we perform $P$ graph shifts of the past filter memory $\boldsymbol{y}_{t-1}$ and $Q$ of the graph signal $\boldsymbol{x}$. We will refer to this implementation as an ARMA$_{P,Q}$ graph filter.

**Proposition 1** *The graph frequency response of ARMA$_{P,Q}$ is*

$$h(\lambda_n) = \frac{\sum_{q=0}^{Q} \varphi_q \lambda_n^q}{1 + \sum_{p=1}^{P} \psi_p \lambda_n^p} \quad \textit{subject to} \quad \sum_{p=1}^{P} |\psi_p| \varrho^p < 1. \tag{5}$$

[1]We consider that one iteration is performed in one time instant.

*Recursion* (4) *converges to it linearly independently from the initial condition $\boldsymbol{y}_0$ and the choice of the shift operator $\boldsymbol{S}$.*

**Proof** (Sketch) To ease the notation, let's set $\boldsymbol{P} = -\sum_{p=1}^{P} \psi_p \boldsymbol{S}^p$ and $\boldsymbol{Q} = \sum_{q=0}^{Q} \varphi_q \boldsymbol{S}^q$. Then, (4) can be written as

$$\boldsymbol{y}_t = \boldsymbol{P}^t \boldsymbol{y}_0 + \sum_{\tau=0}^{t-1} \boldsymbol{P}^\tau \boldsymbol{Q} \boldsymbol{x}. \tag{6}$$

When $\|\boldsymbol{P}\| < 1$ and for $t \to \infty$ we approach the steady state

$$\boldsymbol{y} = \lim_{t \to \infty} \boldsymbol{y}_t = \sum_{\tau=0}^{\infty} \boldsymbol{P}^\tau \boldsymbol{Q} \boldsymbol{x} = (\boldsymbol{I} - \boldsymbol{P})^{-1} \boldsymbol{Q} \boldsymbol{x}. \tag{7}$$

Substituting back the expressions for $\boldsymbol{P}$ and $\boldsymbol{Q}$ in (7) and applying the graph Fourier transform, we obtain the relationship between the $n$-th frequency component of the output $\hat{y}_n$ and input $\hat{x}_n$:

$$\hat{y}_n = \left(1 + \sum_{p=1}^{P} \psi_p \lambda_n^p\right)^{-1} \left(\sum_{q=0}^{Q} \varphi_q \lambda_n^q\right) \hat{x}_n. \tag{8}$$

The frequency response $\hat{y}_n$ in (5) can simply be obtained by a pointwise division between $\hat{y}_n$ and $\hat{x}_n$. The sufficient condition for convergence can be obtained by substituting the expression of $\boldsymbol{P}$ in $\|\boldsymbol{P}\| < 1$ and then applying the triangle and Cauchy-Schwarz inequality while remembering that $\|\boldsymbol{S}\| = \varrho$. □

Proposition 1 formally states what we intuitively said regarding recursion (4). Indeed, from (5) we can see that ARMA$_{P,Q}$ is characterized by a rational frequency response of order $P$ in the denominator and $Q$ in the numerator. Further, from (3) and (5) with the ARMA$_{P,Q}$ we can easily obtain frequency responses of different orders in the numerator and denominator. Even-though it is true that with recursion (2) we can also obtain different orders, it is recommended there to use the same order to obtain a stable recursion with a decent approximation accuracy. Secondly, in contrast to recursion (2), the stability condition in (5) is now convex in the denominator coefficients $\psi_p$ rather than in the poles[2].

Similar to the ARMA$_K$ filter, the largest stability region is obtained for a small $\varrho$. This can be for instance the case when we consider as a shift operator the translated normalized Laplacian ($\boldsymbol{L}_n$), i.e., $\boldsymbol{S} = \boldsymbol{L}_n - \boldsymbol{I}$. In the latter case, the *sufficient* stability condition reduces to $\sum_{p=1}^{P} |\psi_p| < 1$.

**Convergence Analysis.** As the ARMA$_K$ filter, the ARMA$_{P,Q}$ filter achieves the rational frequency response (5) theoretically for $t \to \infty$. However, in practice the output becomes close to the steady state after few iterations due to the linear convergence of the algorithm. The following proposition gives a characterizes the linear convergence time of the algorithm and shows our handle on reducing the convergence time.

**Proposition 2** *For a stable ARMA$_{P,Q}$ filter* (4)*, the necessary number of iterations $t$ to be $\epsilon$-close to the frequency response* (5) *is*

$$t \geq ln(\epsilon/\alpha) \left(ln\left(\|\boldsymbol{P}\|\right)\right)^{-1}, \tag{9}$$

*for a desired small error $\epsilon$ and $\alpha = \|\boldsymbol{y}_0\| + (1 - \|\boldsymbol{P}\|)^{-1} \|\boldsymbol{Q}\| \|\boldsymbol{x}\|$, where $\|\cdot\|$ indicates the 2-norm.*

**Proof** (Sketch) Given the ARMA$_{P,Q}$ output in (6), the error norm w.r.t. the steady state (7) at time $t$ is

$$\|\boldsymbol{y}_t - \boldsymbol{y}\| \leq \|\boldsymbol{P}\|^t \|\boldsymbol{y}_0\| + \sum_{\tau=t}^{\infty} \|\boldsymbol{P}\|^\tau \|\boldsymbol{Q}\| \|\boldsymbol{x}\|, \tag{10}$$

[2]This can be observed by expressing (3) from the partial fraction form into a ratio of two polynomials [9].

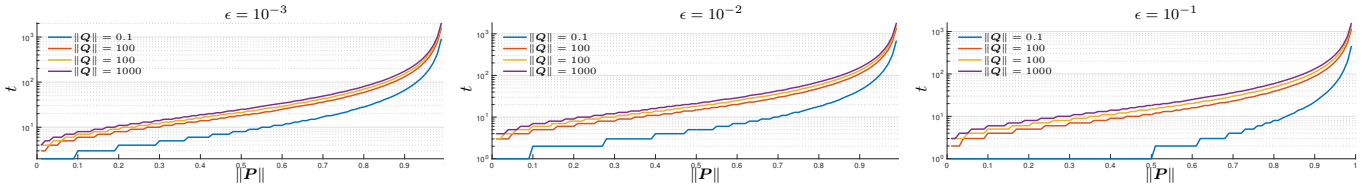


**Fig. 1**. Convergence time of the ARMA$_{P,Q}$ (4) to be $\epsilon$-close to the steady state (7) as a function of $\|\boldsymbol{P}\|$. The results are analyzed for different values of $\|\boldsymbol{Q}\|$ and for different values of the approximation error $\epsilon$.

where in (10) we have applied the triangle and Cauchy-Schwarz inequality of the norms. Then, by considering that ARMA$_{P,Q}$ is stable, i.e., $\|\boldsymbol{P}\| < 1$, we can express the geometric series in closed form

$$\begin{aligned}\|\boldsymbol{y}_t - \boldsymbol{y}\| &\leq \|\boldsymbol{P}\|^t \|\boldsymbol{y}_0\| + (1 - \|\boldsymbol{P}\|)^{-1} \|\boldsymbol{P}\|^t \|\boldsymbol{Q}\| \|\boldsymbol{x}\| \\ &\leq \|\boldsymbol{P}\|^t \left(\|\boldsymbol{y}_0\| + (1 - \|\boldsymbol{P}\|)^{-1} \|\boldsymbol{Q}\| \|\boldsymbol{x}\|\right).\end{aligned} \tag{11}$$

Then, for $\|\boldsymbol{y}_t - \boldsymbol{y}\| \leq \epsilon$ we have

$$t \left(ln \left(\|\boldsymbol{P}\|\right)\right) \leq ln \left(\frac{\epsilon}{\|\boldsymbol{y}_0\| + (1 - \|\boldsymbol{P}\|)^{-1} \|\boldsymbol{Q}\| \|\boldsymbol{x}\|}\right), \tag{12}$$

which can be reformulated into (9) by dividing both sides of (12) by $ln \left(\|\boldsymbol{P}\|\right) < 0$. □

Proposition 2 asserts that there is a minimum number of iterations $t$ that algorithm (4) must be run in order to be $\epsilon$-close to its steady state. As we can see from (9), a handle to reduce $t$ is the spectral norm of $\boldsymbol{P}$. From $\left(ln \left(\|\boldsymbol{P}\|\right)\right)^{-1}$ (where $\|\boldsymbol{P}\| \leq 1$ to ensure stability) $t$ can be reduced for a $\boldsymbol{P}$ with a smaller spectral norm. However, this is in contrast to the effect of $\|\boldsymbol{P}\|$ in $ln(\epsilon/\alpha)$, where a smaller spectral norm of $\boldsymbol{P}$ gives a larger $ln(\epsilon/\alpha)$ and thus potentially a higher $t$. The latter is shown to have a lower influence on $t$. Further, since the $ln(\|\boldsymbol{P}\|) < 0$, $t$ can also be reduced by choosing the coefficients $\varphi_q$ to reduce the spectral norm of $\boldsymbol{Q}$ (this is true when $\ln(\epsilon/\alpha) < 0$, otherwise we have directly $t = 0$).

To quantify what is said above, we consider a scenario where $\epsilon = [10^{-3}, 10^{-2}, 10^{-1}]$, $\|\boldsymbol{y}_0\| = \boldsymbol{0}$ and $\|\boldsymbol{x}\| = 1$. Fig. 1 shows the convergence time (rounded to the next integer) of the algorithm when (9) is met with equality, as a function of $\|\boldsymbol{P}\| \in [0, 1[$ and for different values of $\|\boldsymbol{Q}\|$. The first thing to notice is the impact that $\|\boldsymbol{P}\|$ has on $\left(ln \left(\|\boldsymbol{P}\|\right)\right)^{-1}$ prevails on that of $ln(\epsilon/\alpha)$. Indeed, $t$ increases monotonically with $\|\boldsymbol{P}\|$. Secondly, as previously mentioned, also a higher value $\|\boldsymbol{Q}\|$ will give a larger convergence time. From these results we suggest to avoid values of $\|\boldsymbol{P}\|$ close to 1 in order to obtain reasonable convergence times. On the other hand, this means that the stability region for convergence is reduced and thus we have less degrees of freedom to design our coefficients. As we will see next, this will potentially lead to a lower approximation accuracy. Similar considerations hold also for $\|\boldsymbol{Q}\|$, but since it has less effect on the convergence time we can allow a higher $\|\boldsymbol{Q}\|$ to improve the approximation accuracy. To summarize this part, the filter coefficients are designed as a trade-off between approximation accuracy and convergence time.

**ARMA$_{P,Q}$ design.** Given an ARMA$_{P,Q}$ graph filter and a desired frequency response $\hat{h}_n : \{\lambda_n\}_{n=1}^N \to \mathbb{R}$, we would like to find the filter coefficients $\psi_p$ and $\varphi_q$ that make the frequency response (5) as close as possible to $\hat{h}(\lambda_n)$, yet ensuring a stable implementation. Differently, we would like to minimize the error

$$e'(\lambda_n) = \hat{h}(\lambda_n) - \frac{\sum_{q=0}^{Q} \varphi_q \lambda_n^q}{1 + \sum_{p=1}^{P} \psi_p \lambda_n^p}. \tag{13}$$

Finding the filter coefficients by minimizing, in a least squares error sense, the error (13) leads to a set of nonlinear equations in the filter coefficients. Similar to the Padé approximation in the time domain [17], we can multiply both sides of (13) by the denominator expression of the frequency response to obtain the new (not equivalent) error

$$e(\lambda_n) = \hat{h}(\lambda_n) + \hat{h}(\lambda_n) \sum_{p=1}^{P} \psi_p \lambda_n^p - \sum_{q=0}^{Q} \varphi_q \lambda_n^q, \tag{14}$$

which is now linear in the filter coefficients. Then, by staking the errors for different $\lambda_n$ in $\boldsymbol{e} = [e(\lambda_1), \dots e(\lambda_N)]^\top$ and defining $\boldsymbol{\psi} = [\psi_1, \dots, \psi_P]^\top$, $\boldsymbol{\varphi} = [\varphi_1, \dots, \varphi_Q]^\top$ as the vectors containing the filter coefficients, we rewrite (14) as

$$\boldsymbol{e} = \hat{\boldsymbol{h}} + \text{diag}(\hat{\boldsymbol{h}}) \boldsymbol{\Psi}_P \boldsymbol{\psi} - \boldsymbol{\Phi}_{Q+1} \boldsymbol{\varphi}, \tag{15}$$

where $\hat{\boldsymbol{h}} = [\hat{h}(\lambda_1), \dots \hat{h}(\lambda_N)]^\top$ is the $N \times 1$ vector containing the desired frequency response, $\boldsymbol{\Psi}_P$ is a Vandermonde-like $N \times P$ matrix with $(r, p)$-th entry $[\boldsymbol{\Psi}_P]_{r,p} = \lambda_r^p$ and $\boldsymbol{\Phi}_{Q+1}$ is the $N \times (Q+1)$ matrix still with a Vandermonde-like structure, but with $(r, q)$-th entry $[\boldsymbol{\Phi}_{Q+1}]_{r,q} = \lambda_r^{q-1}$. With the new notation in place, the stability condition of the filter $\|\boldsymbol{P}\| < 1$ can be expressed as $\|\boldsymbol{\Psi}_P \boldsymbol{\psi}\|_\infty < 1$, where the latter inequality can be derived from the expression of $\boldsymbol{P}$ and by considering that $\boldsymbol{S} = \boldsymbol{U} \boldsymbol{\Lambda} \boldsymbol{U}^{\mathsf{H}}$. Then, stable filter coefficients that minimize $\boldsymbol{e}$ are obtained by solving the convex constrained least squares problem

$$\begin{aligned} \underset{\boldsymbol{\psi}, \boldsymbol{\varphi}}{\text{minimize}} \quad & \|\hat{\boldsymbol{h}} + \text{diag}(\hat{\boldsymbol{h}}) \boldsymbol{\Psi}_P \boldsymbol{\psi} - \boldsymbol{\Phi}_{Q+1} \boldsymbol{\varphi}\|_2 \\ \text{subject to} \quad & \|\boldsymbol{\Psi}_P \boldsymbol{\psi}\|_\infty \leq \delta_P, \quad \delta_P < 1, \end{aligned} \tag{16}$$

where $\delta_P$ is our handle to trade convergence speed with approximation accuracy. From our tests we suggest that a good choice for $\delta_P$ is in the range $[0.6, 0.8]$. Such consideration comes from the fact that a higher value of $\delta_P$ lead to slower convergence (see also Fig. 1) and thus higher computational costs. On the other hand, values of $\delta_P < 0.6$ are generally not recommended since the approximation accuracy is highly affected and the gain in convergence time is not worth it.

While it is true that solving (16) will produce the stable filter coefficients that minimize the error $e(\lambda_n)$, we are more interested in the minimization of $e'(\lambda_n)$. As we previously said, this minimization problem is highly non convex due to the denominator expresion in $\boldsymbol{\psi}$. However, once the optimal solution $\boldsymbol{\psi}^*$ of (16) is obtained, we can plug it back into (13) and then simply minimize it w.r.t. $\boldsymbol{\varphi}$.

**Distributed complexity.** To obtain in a distributed manner the ARMA$_{P,Q}$ output after $T$ iterations, recursion (4) considers the multiplication of the terms $\boldsymbol{S}^p \boldsymbol{y}_{t-1}$ for $p = 1 \dots P$ and $t = 0, \dots, T-1$. Further, we also need to compute $\boldsymbol{S}^q \boldsymbol{x}$ for $q = 1, \dots Q$. While for the past output $\boldsymbol{y}_{t-1}$ all terms $\boldsymbol{S}^1 \boldsymbol{y}_{t-1} \dots \boldsymbol{S}^P \boldsymbol{y}_{t-1}$ must be computed, for the input signal the terms $\boldsymbol{S}^q \boldsymbol{x}$ can be calculated only once as $\boldsymbol{S}(\boldsymbol{S}^{q-1} \boldsymbol{x})$. Thus, for obtaining $\boldsymbol{y}_T$, $Q$ multiplications are performed in the moving average and $P$ multiplications for each $t \in [0, \dots, T-1]$ in the autoregressive part. Considering that the graph shift matrix $\boldsymbol{S}$ is a local matrix, the multiplication of $\boldsymbol{S}$ with

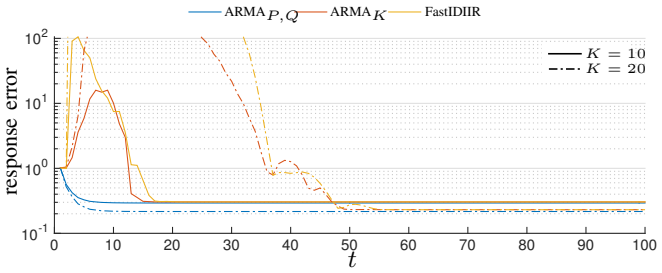


**Fig. 2**. Normalized response error as a function of time for the three distributed algorithms to implement a rational frequency response. The results are shown for orders $K = 10,\ 20$. The ARMA$_{P,Q}$ is designed such that $P + Q \leq K$ to ensure the same distributed cost.

any graph signal can be computed distributively by exchanging $2M$ values. It follows that the overall communication complexity is of order $O(M(TP + Q))$.

## 4. NUMERICAL RESULTS

In this section we evaluate the performance of the proposed method. First we compare the ARMA$_{P,Q}$ filter with the parallel ARMA$_K$ (2) [9] and with the parallel FastIDIIR algorithm of [12], which uses a gradient descent algorithm implemented distributively. Then, we compare the ARMA$_{P,Q}$ with some FIR filters.

We analyze the performance on a random geometric graph of $N = 100$ nodes randomly distributed in a squared area with two nodes being neighbors if they are closer than 15% of the maximum distance. We consider as shift operator $\boldsymbol{S} = \boldsymbol{L}_{\text{n}} - \frac{\lambda_{\max}(\boldsymbol{L}_{\text{n}})}{2}\boldsymbol{I}$, with $\lambda_{\max}(\boldsymbol{L}_{\text{n}})$ the maximum eigenvalue of $\boldsymbol{L}_{\text{n}}$ for the *particular* graph. All the filters are designed to approximate an ideal low pass filter in the frequency domain of $\boldsymbol{S}$. The cut-off frequency of the filter is $\lambda_c = \lambda_{N/2}(\boldsymbol{S})$, i.e., the half of the band. The input signal $\boldsymbol{x}$ is considered to have a white unitary spectrum w.r.t. the underlying graph and the filters are initialized as $\boldsymbol{y}_0 = \boldsymbol{x}$. Our results are averaged over 10 iterations.

**Rational distributed filters.** We first compare the convergence time of the three distributed algorithms that are characterized by a rational frequency response for $K = 10,\ 20$. Note that the *per iteration* complexity of the ARMA$_{P,Q}$ is at most equal to that of the other two approaches (equality when $P + Q = K$). For the ARMA$_K$ and for the FastIDIIR the Shank's method proposed in [9] is used to design the filter coefficients, while for the ARMA$_{P,Q}$ we follow the design method proposed in Section 3. In the latter case, for each graph, the values of $P$ and $Q$ ($P + Q \leq K$ to have at most same cost) with the smallest approximation error are selected with $\delta_P = 0.65$.

Fig. 2 depicts the normalized approximation error between the filter frequency responses and the desired one as a function of time. We can see that ARMA$_{P,Q}$ converges faster than the other algorithms yet ensuring the same approximation accuracy. The large values in the transition phase of ARMA$_K$ and FastIDIIR are due to their large coefficients, while the ARMA$_{P,Q}$ error always reduces with $t$.

**Comparison with FIR.** We compare the performance of the ARMA$_{P,Q}$ graph filter with the FIR$_K$ with the same distributed cost. Specifically, we consider the FIR output after $K$ iterations and the ARMA$_{P,Q}$ output after $T$ iterations such that $TP + Q = K$. Then, we compare both filter outputs with the desired frequency response to calculate the normalized response error. The FIR filter coefficients are calculated in a least squares sense matching the desired frequency response. These results are shown in Table 1.

**Table 1**. Normalized response error as a function of filter order $K$ and convergence parameter $\delta_P$ for the ARMA and FIR filters. To improve visibility we color in red the cell where FIR performs better, in orange where the ARMA$_{P,Q}$ is worst by at most 0.05 and in green where the ARMA$_{P,Q}$ performs better.

| $\delta$ \ $K$ | 3 | 9 | 15 | 21 | 27 | 33 | 39 |
|---|---|---|---|---|---|---|---|
| 0.1 | 0.560 | 0.338 | 0.274 | 0.234 | 0.210 | 0.203 | 0.200 |
| 0.2 | 0.570 | 0.344 | 0.279 | 0.236 | 0.210 | 0.203 | 0.198 |
| 0.3 | 0.583 | 0.354 | 0.280 | 0.237 | 0.210 | 0.201 | 0.194 |
| 0.4 | 0.601 | 0.360 | 0.280 | 0.237 | 0.210 | 0.200 | 0.190 |
| 0.5 | 0.601 | 0.360 | 0.280 | 0.236 | 0.204 | 0.191 | 0.183 |
| 0.6 | 0.596 | 0.362 | 0.281 | 0.235 | 0.194 | 0.182 | 0.177 |
| 0.7 | 0.601 | 0.366 | 0.282 | 0.235 | 0.185 | 0.173 | 0.170 |
| 0.8 | 0.598 | 0.380 | 0.287 | 0.237 | 0.183 | 0.166 | 0.162 |
| 0.9 | 0.597 | 0.412 | 0.307 | 0.249 | 0.197 | 0.171 | 0.165 |
| 0.99 | 0.596 | 0.456 | 0.368 | 0.321 | 0.291 | 0.245 | 0.223 |
| FIR | 0.450 | 0.311 | 0.255 | 0.222 | 0.197 | 0.194 | 0.194 |

Before comparing with the FIR filters, let us point out a couple observations regarding the ARMA$_{P,Q}$ behaviour *when the output is not considered at steady state but in finite time*: (i) for low filter orders $K \leq 18$, a choice of $\delta_P \leq 0.6$ is preferred to obtain a better approximation accuracy; (ii) on the other hand, when $K > 18$ the best choices for our trading handle are in the range $\delta_P \in [0.6, 0.9]$.

When analyzing the performance of the FIR filter for a specific order $K$, we have that the FIR performance degrades for high filter order ($K > 24$), especially when the graph frequencies are numerically close to each other. For this purpose, filter orders smaller than $K$ can also be used for the FIR filter if this leads to a better performance. In comparison with the ARMA$_{P,Q}$ with fixed distributed costs, the FIR filters performs much better for $K < 9$, and have still a better performance but are closer to the ARMA$_{P,Q}$ for $K \leq 21$. On the other hand, when $K > 21$ the ARMA$_{P,Q}$ filter dominates the FIR in terms of approximation accuracy. Specifically, for the considered scenario, the approximation accuracy of the FIR does not go below 0.2 while for the ARMA$_{P,Q}$ for $K = 39$ and $\delta = 0.8$ we have an approximation accuracy of 0.16.

We summarize this part with the following two observations: (i) in a fixed distributed cost comparison the choice of the FIR is recommended for small filter orders, while the ARMA$_{P,Q}$ for higher orders; (ii) in case the distributed cost is not limited, i.e., where the ARMA$_{P,Q}$ can be run for enough iterations to be close to its steady state, it has the potential to achieve much better approximation accuracies. In the latter case, even FIRs with higher orders cannot achieve the same performance.

## 5. CONCLUSIONS

In this work, we propose a novel strategy to distributively implement rational graph filters. Similar to state of the art algorithms that implement IIR graph filters the designed frequency response is achieved in linear convergence time. The proposed algorithm enjoys a stable distributed implementation and now the filter coefficients can be designed as a tradeoff between the convergence time and approximation accuracy. Simulation results show that the proposed implementation strategy outperforms other distributed IIR graph filters and for higher filter orders it also beats the distributed FIR filters.

## 6. REFERENCES


[1] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The Emerging Field of Signal Processing on Graphs: Extending High-Dimensional Data Analysis to Networks and Other Irregular Domains," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 83–98, 2013.

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

[3] ——, "Discrete Signal Processing on Graphs: Frequency Analysis," *IEEE Transactions on Signal Processing*, vol. 62, no. 12, pp. 3042–3054, 2014.

[4] F. Zhang and E. R. Hancock, "Graph Spectral Image Smoothing Using the Heat Kernel," *Pattern Recognition*, vol. 41, no. 11, pp. 3328–3342, 2008.

[5] D. I. Shuman, P. Vandergheynst, and P. Frossard, "Chebyshev Polynomial Approximation for Distributed Signal Processing," in *IEEE International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)*, 2011, pp. 1–8.

[6] S. K. Narang, A. Gadde, and A. Ortega, "Signal Processing Techniques for Interpolation in Graph Structured Data," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 5445–5449.

[7] J. Ma, W. Huang, S. Segarra, and A. Ribeiro, "Diffusion Filtering of Graph Signals and Its Use in Recommendation Systems," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 4563–4567.

[8] S. Segarra, A. G. Marques, and A. Ribeiro, "Distributed Linear Network Operators Using Graph Filters," *arXiv preprint arXiv:1510.03947*, 2015.

[9] 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.

[10] M. Belkin and P. Niyogi, "Semi-Supervised Learning on Riemannian Manifolds," *Machine learning*, vol. 56, no. 1-3, pp. 209–239, 2004.

[11] E. Isufi, A. Simonetto, A. Loukas, and G. Leus, "Stochastic Graph Filtering on Time-Varying Graphs," in *IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, 2015.

[12] X. Shi, H. Feng, M. Zhai, T. Yang, and B. Hu, "Infinite Impulse Response Graph Filters in Wireless Sensor Networks," *IEEE Signal Processing Letters*, Jan 2015.

[13] A. Sandryhaila and J. M. Moura, "Big Data Analysis with Signal Processing on Graphs: Representation and Processing of Massive Data Sets with Irregular Structure," *IEEE Signal Processing Magazine*, vol. 31, no. 5, pp. 80–90, 2014.

[14] E. Isufi, G. Leus, and P. Banelli, "2-Dimensional Finite Impulse Response Graph-Temporal Filters," in *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, Washington D.C, USA, 2016.

[15] A. Loukas and D. Foucard, "Frequency Analysis of Temporal Graph Signals," in *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. Washington D.C, USA, 2016.

[16] E. Isufi, A. Loukas, A. Simonetto, and G. Leus, "Separable Autoregressive Moving Average Graph-Temporal Filters," in *Signal Processing Conference (EUSIPCO), 2016 24th European*. IEEE, 2016, pp. 200–204.

[17] M. H. Hayes, *Statistical Digital Signal Processing and Modeling*. John Wiley & Sons, 2009.