NODE-ASYNCHRONOUS IMPLEMENTATION OF RATIONAL FILTERS ON GRAPHS

Oguzhan Teke and P. P. Vaidyanathan

Department of Electrical Engineering, MC 136-93,
California Institute of Technology, Pasadena, CA 91125, USA
E-mail: oteke@caltech.edu, ppvnath@systems.caltech.edu

ABSTRACT

This paper considers a node-asynchronous implementation of rational ("IIR") filters on graphs, in which the nodes are assumed to wake up randomly and independently from each other, and communicate only with their immediate neighbors. The underlying graph is allowed to be directed, possibly with a non-diagonalizable adjacency matrix. Since the nodes are allowed to act independently, the proposed implementation is practical for very large or autonomous networks where synchronization is difficult to achieve. Furthermore, the proposed algorithm is 1-hop localized on the graph irrespective of the order of the filter. The method is shown to converge in the mean-squared sense under a boundedness assumption on the filter as well as the graph operator. The result follows from the convergence of a more general randomized asynchronous state recursion, which is also presented in this paper. The algorithm is simulated on a random geometric graph, which numerically verifies the convergence.

Index Terms— Graph signal processing, node-asynchronous iterations, rational graph filters, distributed processing.

1. INTRODUCTION

In the present information era, observed data is often no longer represented as time series. In such cases, one useful model is to use a network structure in which the underlying graph is assumed to represent the dependency between the data points. This broad model has found applications in many different contexts such as social, economic, and biological networks, among others [1, 2].

The recent area of graph signal processing [3–6] extends classical signal processing techniques to such network modeled data, in which the analysis is based on the "graph operator," whose eigenvectors serve as the graph Fourier basis (GFB). With the use of GFB, sampling, reconstruction, multirate processing of graph signals and some uncertainty results have been extended to the case of graphs in [7–17].

One important aspect of graph signal processing is the use of polynomial graph filters, which can be considered as an extension of finite impulse response (FIR) filters to the case of graphs [5–7]. It is well-known that a polynomial graph filter of order L is localized on the graph, that is, nodes are required to communicate only with its L-hop neighbors in order to implement the filter. For this reason it is very natural to think of polynomial graph filtering as a way of distributed signal processing, in which the low-order polynomials are favored to keep the communications localized. The papers [18–23] (and references therein) made explicit connections between polynomial graph filters and distributed computation, and studied various problems including smoothing, regularization, and consensus.

This work was supported in parts by the ONR grants N00014-17-1-2732 and N00014-18-1-2390, the NSF grant CCF-1712633, and the Electrical Engineering Carver Mead Research Seed Fund of the California Institute of Technology.

Filters with infinite impulse responses (IIR) form another important class of filters in classical signal processing, and they provide better approximations for a given filter specification. When extended to the case of graphs, an IIR filter of order L can be implemented via state recursions running on a polynomial of order L that preserves the locality of the communications. The studies in [24–27] analyzed the convergence behavior of such filters and showed successful applications on graph signals with distributed processing.

Although both FIR (polynomial) and IIR (rational rather than a polynomial) filters can be implemented in a distributed fashion, aforementioned implementations implicitly assume synchronization between the nodes. That is, all the nodes should send and receive data at the same time instance, or nodes should wait until all the communications are terminated before proceeding to the next iteration. Synchronization becomes an important limitation when the size of the network, N, is large, or the network has autonomous behavior, in which case there is no centralized control over the network.

In this study, we will consider an asynchronous implementation of rational graph filtering that eliminates the need for synchronization. In the proposed algorithm, a node is assumed to wake up at a random time instance independent of the remaining nodes, perform the necessary computations, and broadcast back to its neighbors. By carefully designing the computation scheme, the proposed algorithm is shown to converge to the desired filter output in the mean-squared sense, where the filter is assumed to be a rational function with denominator having order $L < N$ (in practice $L \ll N$). In addition, the algorithm proposed here requires data exchange between only the neighboring nodes irrespective of the order of the filter L. Thus, the algorithm is truly distributed since it requires only the neighboring nodes to communicate with each other without any synchronization.

In the following, we first provide an overview of the notation. In Section 2, we define a randomized asynchronous state recursion with nonzero input and present its convergence behavior (Theorem 1). In Section 3, we consider rational filters on graphs and propose a scheme that implements the given rational filter in a node-asynchronous fashion (Algorithm 1), and present its convergence (Theorem 2). In Section 4, we experimentally verify the convergence of the proposed algorithm.

1.1. Preliminaries and Notation

We will assume that $\mathbf{A} \in \mathbb{C}^{N\times N}$ is an operator on the graph with N nodes. We consider $\mathbf{A}$ to be a *local* operator, that is, $A_{i,j} = 0$ when the nodes i and j are not neighbors. In particular, $A_{i,j}$ denotes the weight of the edge from node j to node i. The graph is allowed to be directed possibly with a non-diagonalizable adjacency matrix. We will use $\mathcal{N}_{\text{in}}(i)$ and $\mathcal{N}_{\text{out}}(i)$ to denote the incoming and outgoing neighbors of the node i. More precisely we have:

$$\mathcal{N}_{\text{in}}(i) = \{j \mid A_{i,j} \neq 0\}, \qquad \mathcal{N}_{\text{out}}(i) = \{j \mid A_{j,i} \neq 0\}. \tag{1}$$

We will use $\mathbb{P}[\cdot]$ and $\mathbb{E}[\cdot]$ to denote the probability and expectation, respectively. For a vector $\mathbf{x}$ we will use $\|\mathbf{x}\|_2$ to denote its ℓ_2-norm. For a matrix $\mathbf{X}$ we will use $\mathrm{tr}(\mathbf{X})$ to denote its trace, $\|\mathbf{X}\|_2$ to denote its spectral norm (the largest singular value), $\sigma_{\min}(\mathbf{X})$ to denote its smallest singular value, $\rho(\mathbf{X})$ to denote its spectral radius (the largest eigenvalue in absolute sense). We will use $\mathcal{T}$ to denote a subset of $\{1, \cdots, N\}$, and its size is denoted as $t = |\mathcal{T}|$.

2. ASYNCHRONOUS STATE RECURSIONS

Given a matrix of interest $\mathbf{A}$ and an input signal $\mathbf{u}$, we will consider the following type of recursion on a state vector $\mathbf{x}_k$:

$$\mathbf{x}_{k+1} = \mathbf{A}\,\mathbf{x}_k + \mathbf{u}, \tag{2}$$

where $\mathbf{x}_0$ is the initial state vector, and $\mathbf{A}$ here is assumed to be an arbitrary matrix. Later in Section 3, $\mathbf{A}$ will be considered as a graph operator, and the model in (2) will represent communications between the nodes (instead of a simple graph shift as in graph signal processing [3–6]). In this setting, $\mathbf{u}$ will be a signal defined on the graph, where the nodes will be the "domain" analogous to time. The index k will denote the round of communication, so the input signal $\mathbf{u}$ does not have any dependency on the iteration index k.

The fixed point of (2), denoted as $\mathbf{x}^\star$, can be found by solving the following equation:

$$\mathbf{x}^\star = \mathbf{A}\,\mathbf{x}^\star + \mathbf{u}, \qquad \text{so that} \qquad \mathbf{x}^\star = (\mathbf{I} - \mathbf{A})^{-1}\,\mathbf{u}, \tag{3}$$

which assumes that $\mathbf{A}$ does *not* have eigenvalue 1 so that $\mathbf{I} - \mathbf{A}$ is invertible. In this sense, results here differ from those in [28–30] that consider the case $\mathbf{u} = \mathbf{0}$ and assume that 1 is an eigenvalue of $\mathbf{A}$.

In order to analyze the convergence behavior of the state recursions in (2), we first define the residual vector as follows:

$$\mathbf{r}_k = \mathbf{x}_k - \mathbf{x}^\star. \tag{4}$$

Then, under the state recursions in (2), the residual defined in (4) can be shown to evolve as follows:

$$\mathbf{r}_{k+1} = \mathbf{A}\,\mathbf{r}_k. \tag{5}$$

Thus, one can argue that the condition $\rho(\mathbf{A}) < 1$ (where $\rho(\mathbf{A})$ is the spectral radius) is both necessary and sufficient for the residual $\mathbf{r}_k$ to convergence to zero, which also implies that the state vector $\mathbf{x}_k$ converges to $\mathbf{x}^\star$.

In this study we will consider a randomized asynchronous variation of the state recursion in (2), in which only a random subset of indices are updated simultaneously and the remaining ones stay unchanged. More precisely, we consider the following update model:

$$(\mathbf{x}_{k+1})_i = \begin{cases} (\mathbf{A}\,\mathbf{x}_k)_i + u_i + (\mathbf{w}_k)_i, & i \in \mathcal{T}_k, \\ (\mathbf{x}_k)_i, & i \notin \mathcal{T}_k, \end{cases} \tag{6}$$

where u_i denotes the i^{th} index of the input signal, and $\mathbf{w}_k$ denotes noise with the following statistics:

$$\mathbb{E}[\mathbf{w}_k] = \mathbf{0}, \qquad \mathbb{E}[\,\mathbf{w}_k\,\mathbf{w}_s^{\mathrm{H}}\,] = \delta(k-s)\;\mathbf{\Gamma}, \tag{7}$$

where $\delta(\cdot)$ denotes the discrete Dirac delta function. We note that the noise is assumed to be uncorrelated between two different iterations, but $\mathbf{\Gamma}$ is allowed to be non diagonal. The update set $\mathcal{T}_k$ is also selected randomly and independently in every iteration of the model in (6). Similar to the setting in [28–30], we consider the case where both the content and the size of $\mathcal{T}_k$ are random. We assume that the size of $\mathcal{T}_k$ has a constant mean denoted as follows:

$$\mu_T = \mathbb{E}[\,|\mathcal{T}_k|\,]. \tag{8}$$

We note that $1 \leqslant \mu_T \leqslant N$ holds true since $\mathcal{T}_k$ is always a non-empty subset of the set $\{1, \cdots, N\}$. Moreover, $\mu_T = 1$ implies that only one index is updated per iteration, and $\mu_T = N$ implies that all the indices are updated in every iteration. Thus, noise-free model with $\mu_T = N$ is equivalent to the synchronous case of (2). We also assume that the probability of $\mathcal{T}_k$ being equal to a specific set $\mathcal{T}$ is determined entirely by the size of the set $\mathcal{T}$. More precisely,

$$\mathbb{P}[\,\mathcal{T}_k = \mathcal{T}\,] = \mathbb{P}[\,|\mathcal{T}_k| = t\,]\,\binom{N}{t}^{-1}, \quad \text{where} \quad t = |\mathcal{T}|, \tag{9}$$

which ensures that there is no bias toward any index in (6).

2.1. Convergence in the Mean-Squared Sense

Asynchronous fixed point iterations are well studied problems in the literature [31, 32]. The earliest analysis can be traced back to the study in [33]. These studies provided fundamental conditions under which the asynchronous iterations are guaranteed to converge for a given index sequence. Thus, the studies in [31–33] do not consider a probabilistic framework. More recently, studies in [34, 35] (and references therein) studied the randomized variations of asynchronous iterations. However, the model considered here in (6) is different since it includes the possibility of updating different number of indices in each iteration, and the possibility of noise. Thus, the model in (6) requires a separate convergence analysis. In fact, the conditions for convergence are different from those in [28–30].

It is important to note that the state vector $\mathbf{x}_k$ in (6) is a random variable due to the random selection of the update sets and the random noise. For a given state transition matrix $\mathbf{A}$ and an input signal $\mathbf{u}$, the following theorem, whose proof is omitted due to space limitations, proves that the random vector $\mathbf{x}_k$ converges to $\mathbf{x}^\star$ in (3) in the mean squared sense as k goes to infinity:

Theorem 1. *If $\|\mathbf{A}\|_2 < 1$, the limit of the mean squared error in the asynchronous model in* (6) *is bounded as follows:*

$$\frac{\mathrm{tr}(\mathbf{\Gamma})}{1 - \sigma_{\min}^2(\mathbf{A})} \;\leqslant\; \lim_{k\to\infty} \mathbb{E}\Big[\,\big\|\mathbf{x}_k - \mathbf{x}^\star\big\|_2^2\,\Big] \;\leqslant\; \frac{\mathrm{tr}(\mathbf{\Gamma})}{1 - \|\mathbf{A}\|_2^2}. \tag{10}$$

When the updates are noise-free ($\mathbf{\Gamma} = \mathbf{0}$) it is clear from Theorem 1 that the vector $\mathbf{x}_k$ converges to $\mathbf{x}^\star$ in the mean squared sense independent of the value of μ_T. That is, the convergence is guaranteed irrespective of the number of indices that are updated concurrently. However, it should be clear that *the rate* of convergence does depend on μ_T, which will be elaborated in a later study.

If the noise is present in the system Theorem 1 reveals an error floor: no matter how many iterations are used, the expected residual error is always bounded away from zero. Thus, $\mathbf{x}_k$ does not converge to $\mathbf{x}^\star$ exactly, which is expected due to the presence of noise. It should be clear that the system in (6) amplifies the noise in general. To see this, note that the assumption $\|\mathbf{A}\|_2 < 1$ in Theorem 1 implies that $0 \leqslant \sigma_{\min}(\mathbf{A}) < 1$. Thus, the lower bound in (10) is always larger than $\mathrm{tr}(\mathbf{\Gamma}) = \mathbb{E}[\|\mathbf{w}_k\|_2^2]$, which shows that the residual error is always larger than the noise itself. As $\sigma_{\min}(\mathbf{A})$ approaches unity, the noise amplification becomes more severe, and even a very small amount of noise results in a very large error term.

It is also important to note that the condition $\|\mathbf{A}\|_2 < 1$ in Theorem 1 is only sufficient to ensure the convergence, but *not necessary* in general. In this regard, consider the following matrix:

$$\mathbf{A} = \alpha \begin{bmatrix} 0 & & \cdots & 0 \\ 1 & 0 & & \vdots \\ 0 & \ddots & \ddots & \\ & 0 & 1 & 0 \end{bmatrix}, \tag{11}$$

where $\alpha \in \mathbb{C}$ is an arbitrary complex number. One can verify that $\rho(\mathbf{A}) = 0$, $\sigma_{\min}(\mathbf{A}) = 0$, but $\|\mathbf{A}\|_2 = |\alpha|$. When $|\alpha| \geqslant 1$, Theorem 1 is inconclusive regarding the convergence of the asynchronous updates on the matrix. However, for the asynchronous model running on (11), one can argue that x_i converges to the following:

$$x_i = \sum_{j=1}^{i} \alpha^{i-j}\, u_j, \qquad \text{for} \qquad 1 \leqslant i \leqslant N, \tag{12}$$

which is, in fact, equivalent to $\mathbf{x}^\star$ in (3). In this particular example the spectral norm of $\mathbf{A}$ can be made arbitrarily large, yet the asynchronous iterations still converge to $\mathbf{x}^\star$, which provides a counter-example against the necessity of the condition $\|\mathbf{A}\|_2 < 1$.

Although Theorem 1 presents only a sufficiency condition for the convergence, its importance follows from the fact that it does not require the diagonalizability of the matrix $\mathbf{A}$. When $\mathbf{A}$ is considered as a graph operator in the next section, this property will be very important to claim the convergence on *an arbitrary directed graph* as long as $\mathbf{A}$ is scaled properly.

3. ASYNCHRONOUS RATIONAL FILTERS ON GRAPHS

In this section we will focus on the following rational graph filter:

$$\tilde{\mathbf{u}} = p(\gamma\,\mathbf{A})\,\big(q(\gamma\,\mathbf{A})\big)^{-1}\,\mathbf{u}, \tag{13}$$

where $\gamma \in \mathbb{C}$ is a parameter to scale the given graph operator $\mathbf{A}$, and the polynomials $p(\cdot)$ and $q(\cdot)$ are assumed to be given as follows:

$$p(x) = \sum_{n=0}^{L-1} p_n\, x^n, \qquad q(x) = 1 + \sum_{n=1}^{L} q_n\, x^n, \tag{14}$$

where $p_n \in \mathbb{C}$ and $q_n \in \mathbb{C}$. It should be noticed that (13) reduces to the polynomial (FIR) filtering in the case of $q_1 = \cdots = q_L = 0$. Thus, any polynomial filter can be studied in this framework as well.

On a graph of size N, a graph operator has (at most) N eigenvalues. Due to the finite size of the spectrum, in the case of a diagonalizable operator, an arbitrary IIR graph filter can be represented as an FIR filter of order at most N-1 [3]. In fact, the overall rational filtering in (13) can be equivalently written as an FIR filtering of order at most N-1. Thus, one way to implement (13) is to compute N-1 graph shifts and take an appropriate linear combination. However this approach carries two drawbacks: 1) FIR representation of (13) has N-1 taps (in general) irrespective of the order of the filters. If the underlying graph is large, the equivalent FIR implementation requires too many graph shifts compared to the order of the actual filters. 2) As discussed in [28], a graph shift forces all nodes to communicate at the same time, which requires a synchronization among the network. In a large network synchronization introduces delays, or it may not be even possible in the case of autonomous networks. In order to overcome these limitations, we present Algorithm 1 that implements the rational filter in (13) in a node-asynchronous manner. In the algorithm, the filter polynomials enter through the following matrices:

$$\mathbf{Q} = \begin{bmatrix} -q_1 & 1 & 0 & 0 \\ -q_2 & 0 & \ddots & 0 \\ \vdots & \vdots & \ddots & 1 \\ -q_L & 0 & \cdots & 0 \end{bmatrix}, \quad \mathbf{p} = \begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ p_{L-1} \end{bmatrix}, \quad \mathbf{e}_1 = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \in \mathbb{C}^L. \tag{15}$$

Algorithm 1 consists of three stages: initialization, passive and active stages. In the initialization, a node initializes its state vector of size L with zeros (not to be confused with the global state vector $\mathbf{x}_k$ in (2) of size N). Then, the node gets into the passive stage, in which the node receives and stores the state vectors sent by its neighbors.

Algorithm 1 Node-Asynchronous Rational Filtering

procedure INITIALIZATION(i)
  Initialize the state vector $\mathbf{s}_i = \begin{bmatrix} s_{i,1} & \cdots & s_{i,L} \end{bmatrix} \in \mathbb{C}^{1\times L}$ as zero.
procedure ACTIVE STAGE(i)
  $\mathbf{s}_i \leftarrow \gamma \sum_{j\in\mathcal{N}_{\text{in}}(i)} A_{i,j}\,\mathbf{s}_j\,\mathbf{Q} + u_i\,\mathbf{e}_1^{\text{H}}$.
  $y_i \leftarrow \mathbf{s}_i\,\mathbf{p}$.
  Broadcast $\mathbf{s}_i$ to all $j \in \mathcal{N}_{\text{out}}(i)$.
procedure PASSIVE STAGE(i)
  if $\mathbf{s}_j$ is received from the node $j \in \mathcal{N}_{\text{in}}(i)$ **then**
    Store the most recent value of $\mathbf{s}_j$.

The node stores only the most recent data. Thus, the node i requires a buffer of size $L \cdot |\mathcal{N}_{\text{in}}(i)|$. A node gets into the active stage at a random time instance independent of the other nodes, in which the node updates its own state vector using its input signal u_i and the state vectors of its neighbors that are already available in the buffer. Then, the node updates its output signal, broadcasts its state vector to its neighbors, and finally gets back into the passive stage.

There are four remarks regarding the algorithm: 1) The random update model in (9) implies that all nodes have the same probability of going into the active stage. Thus, all nodes behave similarly. 2) Since the graph is allowed to be directed, a node may receive a state vector from the node j in the passive stage, but may not send its vector back to the node j in the active stage. 3) Although $\mathbf{Q}$ is a square matrix of size L, a multiplication with it requires only L operations due to its structure. 4) The following theorem, whose proof is presented only as a sketch due to space limitations, proves that Algorithm 1 indeed converges to the output signal in (13).

Theorem 2. *Assume that the polynomials in* (14) *and the scale parameter γ are selected in such a way that the following is satisfied:*

$$|\gamma| < \|\mathbf{Q}\|_2^{-1}\,\|\mathbf{A}\|_2^{-1}. \tag{16}$$

Then, the signal $\mathbf{y}$ in Algorithm 1 converges to $\tilde{\mathbf{u}}$ in (13) *in the mean squared sense, i.e. $\mathbb{E}[\|\mathbf{y}_k - \tilde{\mathbf{u}}\|_2^2] \to 0$. In particular, y_i of the node i converges to $\tilde{u}_i$.*

Proof Sketch: Each node has a state vector of length L, and there are N nodes in total. We cascade the state vectors of all the nodes into a single vector of size $L\,N$, and construct an equivalent asynchronous recursion similar to the model in (6) running on the constructed vector of size $L\,N$. Assuming (16), we apply Theorem 1 and prove that the equivalent model converges, hence Algorithm 1 converges. ■

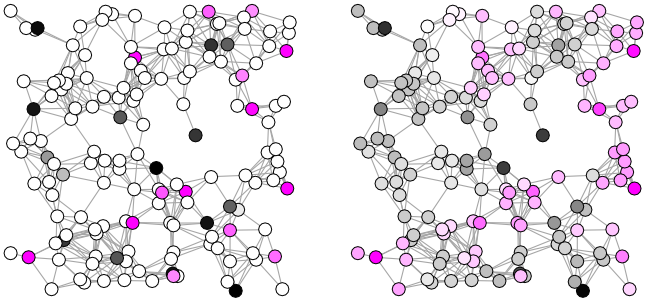
4. EXPERIMENTAL RESULTS

In this section we will simulate the asynchronous rational filtering described in Algorithm 1 on the graph visualized in Figure 1. This is a random geometric graph on $N = 150$ nodes, in which nodes are distributed over the region $[0\ 1] \times [0\ 1]$ uniformly at random. Two nodes are connected to each other if the distance between them is less than 0.15, and the graph is undirected. The graph operator, the matrix $\mathbf{A}$, is selected as the Laplacian matrix whose eigenvalues can be sorted as follows:

$$0 = \lambda_1 < \lambda_2 \leqslant \cdots \leqslant \lambda_N = \|\mathbf{A}\|_2 = 16.8891, \tag{17}$$

where the spectral norm of $\mathbf{A}$ is computed numerically.

We consider the following smoothing problem: assume that we are given the graph signal $\mathbf{u}$ that has only 30 non-zero entries, which is visualized in Figure 1(a). It is clear that the signal $\mathbf{u}$ is not smooth on the graph. In order to obtain a smoothed version of $\mathbf{u}$, which will be denoted by $\tilde{\mathbf{u}}$, we will apply a low-pass filter to the signal $\mathbf{u}$.



(a) (b)

Fig. 1. Visualization of the signals on the graph. Colors black and pink represent positive and negative values, respectively. Intensity of a color represents the magnitude. (a) The graph signal $\mathbf{u}$ that has non-zero values on 30 nodes. (b) The output signal $\tilde{\mathbf{u}}$ in (19).

In order to find the filter that approximates the desired low-pass response the best, one can pose an optimization problem over $2L+1$ parameters (p_i's, q_i's, and γ) under the constraint in (16) for a fixed order L. Due to Theorem 2, the node-asynchronous algorithm running on the designed filter would be guaranteed to converge. However, in this study we will not consider the design phase. As a proof of concept we will consider a very simple rational low-pass filter constructed with the following polynomials of order $L = 2$:

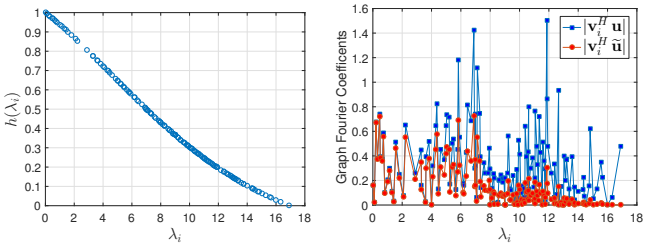
$$p(x) = 1 - x, \qquad q(x) = 1 + x^2, \qquad \gamma = 0.059. \tag{18}$$

Then, the overall rational filter in (13) can be written as follows:

$$h(\lambda) = \frac{1 - \gamma\,\lambda}{1 + \gamma^2\,\lambda^2}, \qquad \text{so that} \qquad \tilde{\mathbf{u}} = h(\mathbf{A})\,\mathbf{u}. \tag{19}$$

Graph frequency response of the filter in (19) is visualized in Figure 2(a), which shows that the filter has low-pass characteristics on the graph. Given the graph operator $\mathbf{A}$, which is the graph Laplacian in this case, one can compute the filter matrix $h(\mathbf{A})$, then obtain the output signal $\tilde{\mathbf{u}}$ via direct matrix multiplication as in (19).

When compared with $\mathbf{u}$, the signal $\tilde{\mathbf{u}}$ has a lower amount of projection on the eigenvectors with larger eigenvalues. Magnitudes of these projections are presented in Figure 2(b). Since $\tilde{\mathbf{u}}$ mainly contains low frequency components (eigenvectors with small eigenvalues [6]), $\tilde{\mathbf{u}}$ is smoother on the graph as visualized in Figure 1(b).



(a) (b)

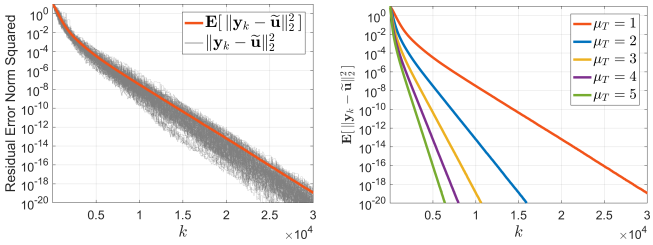
Fig. 2. (a) Response of the rational filter $h(\lambda)$ in (19). (b) Magnitude of the graph Fourier transforms of $\mathbf{u}$ and $\tilde{\mathbf{u}}$, where $(\lambda_i, \mathbf{v}_i)$ denotes an eigenpair of $\mathbf{A}$.

In the following, we will implement the same filter in (19) using the node-asynchronous scheme described in Algorithm 1. We construct the following matrix and vectors according to (15) and (18):

$$\mathbf{Q} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, \qquad \mathbf{p} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \qquad \mathbf{e}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \tag{20}$$

which define the way variables are updated in the algorithm. We note that $\|\mathbf{Q}\|_2 = 1$, and the condition in (16) is satisfied for the selected value of γ in (18), that is, $|\gamma| < \|\mathbf{Q}\|_2^{-1}\,\|\mathbf{A}\|_2^{-1}$. Thus, Theorem 2 guarantees that the signal $\mathbf{y}$ in Algorithm 1 converges to $\tilde{\mathbf{u}}$ in (19) in the mean-squared sense.

In the first implementation of Algorithm 1 we consider the case of $\mu_T = 1$, i.e., only one randomly selected node is updated per iteration. In order to verify the convergence numerically, we simulated 10^5 independent runs of Algorithm 1 and computed the mean squared error by averaging over the independent runs. Figure 3(a) presents the mean squared error together with the error in 100 different realizations. Although Algorithm 1 is initialized deterministically, the signal $\mathbf{y}_k$, hence the error norm, are random variables due to random selection of the nodes. More importantly, the error norm does not decrease monotonically with the iterations as seen in Figure 3(a). Nevertheless, the expectation of the error norm decreases monotonically, which verifies that the error itself converges to zero.



(a) (b)

Fig. 3. (a) Squared residual norm in 100 different realizations of Algorithm 1 together with the mean squared error. (b) Mean-squared error when more than one node are updated simultaneously.

In the second implementation of the algorithm we consider the case of $\mu_T \geqslant 1$, that is, randomly selected μ_T nodes are updated simultaneously where μ_T takes values from 1 to 5. Notice that the case of $\mu_T = 1$ corresponds to the node-asynchronous behavior. Figure 3(b) presents the mean squared error for each case obtained by averaging over 10^5 independent runs. As seen in the figure, convergence is achieved no matter how many nodes are updated concurrently. This is an expected consequence of the fact that Theorem 1, hence Theorem 2, asserts the convergence irrespective of the value of μ_T. Figure 3(b) shows also that lesser number of iterations are needed to reach the same error level when larger number of nodes are updated per iteration. However, the required total number of updates to reach a specific error level stays similar in each case.

5. CONCLUDING REMARKS & FUTURE DIRECTIONS

In this study we proposed a node-asynchronous algorithm that implements a rational filter on the graph of interest. The graph was allowed to be directed, possibly with a non-diagonalizable adjacency matrix. In the proposed algorithm, nodes were assumed to wake up randomly and independently from each other, then communicate with their immediate neighbors. Under a boundedness condition depending on the filter and the graph operator, the proposed algorithm was shown to converge to the output of the given filter. The result followed from the convergence of a more general model of randomized asynchronous state recursions, which was also discussed in this study. The algorithm was simulated on a random geometric graph, and its convergence was validated numerically as well.

In future we will analyze the convergence of the proposed algorithm in the case of imperfect communications. We will consider the scenario where nodes have different rate of updates. We will also consider the filter design problem, in which the filter $h(\lambda)$ is expected to approximate a desired response while satisfying the conditions for convergence of the proposed algorithm.

6. REFERENCES

[1] M. E. J. Newman, *Networks: An Introduction*. Oxford Uni. Press, 2010.

[2] M. Jackson, *Social and Economic Networks*. Princeton Uni. Press, 2008.

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

[4] ——, "Discrete signal processing on graphs: Frequency analysis," *IEEE Trans. Signal Process.*, vol. 62, no. 12, pp. 3042–3054, June 2014.

[5] ——, "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.

[6] D. Shuman, S. 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.

[7] O. Teke and P. P. Vaidyanathan, "Extending classical multirate signal processing theory to graphs – Part I: Fundamentals," *IEEE Trans. Signal Process.*, vol. 65, no. 2, pp. 409–422, Jan. 2017.

[8] ——, "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.

[9] S. Narang and A. Ortega, "Perfect reconstruction two-channel wavelet filter banks for graph structured data," *IEEE Trans. Signal Process.*, vol. 60, no. 6, pp. 2786–2799, June 2012.

[10] ——, "Compact support biorthogonal wavelet filterbanks for arbitrary undirected graphs," *IEEE Trans. Signal Process.*, vol. 61, no. 19, pp. 4673–4685, Oct. 2013.

[11] A. Anis, A. Gadde, and A. Ortega, "Towards a sampling theorem for signals on arbitrary graphs," in *Proc. Int. Conf. Acoust. Speech, Signal Process. (ICASSP)*, May 2014, pp. 3864–3868.

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

[13] A. Gadde and A. Ortega, "A probabilistic interpretation of sampling theory of graph signals," in *Proc. Int. Conf. Acoust. Speech, Signal Process. (ICASSP)*, April 2015, pp. 3257–3261.

[14] X. Wang, P. Liu, and Y. Gu, "Local-set-based graph signal reconstruction," *IEEE Trans. Signal Process.*, vol. 63, no. 9, pp. 2432–2444, May 2015.

[15] A. Agaskar and Y. M. Lu, "A spectral graph uncertainty principle," *IEEE Trans. on Inf. Theory*, vol. 59, no. 7, pp. 4338–4356, July 2013.

[16] M. Tsitsvero, S. Barbarossa, and P. D. Lorenzo, "Signals on graphs: Uncertainty principle and sampling," *IEEE Trans. Signal Process.*, vol. 64, no. 18, pp. 4845–4860, Sept. 2016.

[17] O. Teke and P. P. Vaidyanathan, "Uncertainty principles and sparse eigenvectors of graphs," *IEEE Trans. Signal Process.*, vol. 65, no. 20, pp. 5406–5420, Oct. 2017.

[18] D. I. Shuman, P. Vandergheynst, and P. Frossard, "Chebyshev polynomial approximation for distributed signal processing," in *International Conference on Distributed Computing in Sensor Systems and Workshops*, June 2011, pp. 1–8.

[19] D. I. Shuman, P. Vandergheynst, D. Kressner, and P. Frossard, "Distributed signal processing via chebyshev polynomial approximation," *IEEE Trans. on Sig. and Inf. Process. Net.*, vol. 4, no. 4, pp. 736–751, Dec. 2018.

[20] S. Safavi and U. A. Khan, "Revisiting finite-time distributed algorithms via successive nulling of eigenvalues," *IEEE Sig. Process. Letters*, vol. 22, no. 1, pp. 54–57, Jan. 2015.

[21] A. Sandryhaila, S. Kar, and J. M. F. Moura, "Finite-time distributed consensus through graph filters," in *Proc. Int. Conf. Acoust. Speech, Signal Process. (ICASSP)*, May 2014, pp. 1080–1084.

[22] S. Segarra, A. G. Marques, and A. Ribeiro, "Distributed implementation of linear network operators using graph filters," in *Allerton Conference on Communication, Control, and Computing*, Sept. 2015, pp. 1406–1413.

[23] 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.

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

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

[26] ——, "Filtering random graph processes over random time-varying graphs," *IEEE Trans. Signal Process.*, vol. 65, no. 16, pp. 4406–4421, Aug. 2017.

[27] A. Loukas, A. Simonetto, and G. Leus, "Distributed autoregressive moving average graph filters," *IEEE Sig. Process. Letters*, vol. 22, no. 11, pp. 1931–1935, Nov. 2015.

[28] O. Teke and P. P. Vaidyanathan, "Random node-asynchronous updates on graphs," *Submitted to IEEE Trans. Signal Process.*, 2018.

[29] ——, "The asynchronous power iteration: A graph signal perspective," in *Proc. Int. Conf. Acoust. Speech, Signal Process. (ICASSP)*, Apr. 2018, pp. 4059–4063.

[30] ——, "Asynchronous nonlinear updates on graphs," in *Asilomar Conf. on Signals, Systems and Computers*, Nov. 2018.

[31] G. M. Baudet, "Asynchronous iterative methods for multiprocessors," *J. ACM*, vol. 25, no. 2, pp. 226–244, Apr. 1978.

[32] D. P. Bertsekas, "Distributed asynchronous computation of fixed points," *Mathematical Programming*, vol. 27, no. 1, pp. 107–120, Sep. 1983.

[33] D. Chazan and W. Miranker, "Chaotic relaxation," *Linear Algebra and its Applications*, vol. 2, pp. 199–222, Apr. 1969.

[34] H. Avron, A. Druinsky, and A. Gupta, "Revisiting asynchronous linear solvers: Provable convergence rate through randomization," *J. ACM*, vol. 62, no. 6, pp. 51:1–51:27, Dec. 2015.

[35] Z. Peng, Y. Xu, M. Yan, and W. Yin, "Arock: An algorithmic framework for asynchronous parallel coordinate updates," *SIAM Journal on Scientific Computing*, vol. 38, no. 5, pp. A2851–A2879, 2016.