Chebyshev Polynomial Approximation for Distributed Signal Processing

David I Shuman, Pierre Vandergheynst, and Pascal Frossard
Ecole Polytechnique Fédérale de Lausanne (EPFL)
Signal Processing Laboratory
CH-1015 Lausanne, Switzerland
{david.shuman, pierre.vandergheynst, pascal.frossard}@epfl.ch

***Abstract*—Unions of graph Fourier multipliers are an important class of linear operators for processing signals defined on graphs. We present a novel method to efficiently distribute the application of these operators to the high-dimensional signals collected by sensor networks. The proposed method features approximations of the graph Fourier multipliers by shifted Chebyshev polynomials, whose recurrence relations make them readily amenable to distributed computation. We demonstrate how the proposed method can be used in a distributed denoising task, and show that the communication requirements of the method scale gracefully with the size of the network.**

***Index Terms*—Chebyshev polynomial approximation, denoising, distributed optimization, regularization, signal processing on graphs, spectral graph theory, wireless sensor networks**

I. Introduction

Wireless sensor networks are now prevalent in applications such as environmental monitoring, target tracking, surveillance, medical diagnostics, and manufacturing process flow. The sensor nodes are often deployed *en masse* to collectively achieve tasks such as estimation, detection, classification, and localization. While such networks have the ability to collect large amounts of data in a short time, they also face a number of resource constraints. First, they are energy constrained, as they are often expected to operate for long periods of time without human intervention, despite being powered by batteries or energy harvesting. Second, they may have limited communication range and capacity due to the need to save energy. Third, they may have limited on-board processing capabilities. Therefore, it is critical to develop distributed algorithms for in-network data processing that help balance the trade-offs between performance, communication bandwidth, and computational complexity.

Due to the limited communication range of wireless sensor nodes, each sensor node in a large network is likely to communicate with only a small number of other nodes in the network. To model the communication patterns, we can write down a graph with each vertex corresponding to a sensor node and each edge corresponding to a pair of nodes that communicate. Moreover, because the communication graph is a function of the distances between nodes, it often captures spatial correlations between sensors' observations as well. That is, if two sensors are close enough to communicate, their observations are more likely to be correlated. We can further specify these spatial correlations by adding weights to the edges of the graph, with higher weights associated to edges connecting sensors with closely correlated observations. For example, it is common to construct the graph with a thresholded Gaussian kernel weighting function based on the physical distance between nodes, where the weight of edge e connecting nodes i and j that are a distance $d(i,j)$ apart is

$$w(e) = \begin{cases} \exp\left(-\frac{[d(i,j)]^2}{2\sigma^2}\right) & \text{if } d(i,j) \leq \kappa \\ 0 & \text{otherwise} \end{cases}, \qquad (1)$$

for some parameters σ and κ.

In this paper, we consider signals collected by a sensor network whose nodes can only send messages to their local neighbors (i.e., they cannot communicate directly with a central entity). While much of the literature on distributed signal processing (see, e.g., [1]-[4] and references therein) focuses on coming to an agreement on simple features of the observed signal (e.g., consensus averaging, parameter estimation), we are more interested in processing the full function in a distributed manner, with each node having its own objective. Some example tasks under this umbrella include:

- *Distributed denoising* – In a sensor network of N sensors, a noisy N-dimensional signal is observed, with each component of the signal corresponding to the observation at one sensor location. Using the prior knowledge that the denoised signal should be smooth or piecewise smooth with respect to the underlying weighted graph structure, the sensors' task is to denoise each of their components of the signal by iteratively passing messages to their local neighbors and performing computations.
- *Distributed semi-supervised learning / binary classification* – A binary label (-1 or 1) is associated with each sensor node; however, only a small number of nodes in the network have knowledge of their labels. The cooperative task is for each node to learn its label by iteratively passing messages to its local neighbors and performing computations.

These and similar tasks have been considered in centralized settings in the relatively young field of signal processing on graphs. For example, [5]-[7] consider general regularization

The authors would like to thank Javier Pérez-Trufero for his help producing some of the graphics in this paper.

frameworks on weighted graphs; [8]-[10] present graph-based semi-supervised learning methods; and [11]-[14] consider regularization and filtering on weighted graphs for image and mesh processing. In distributed settings, [15] considers denoising via wavelet processing and [16] presents a denoising algorithm that projects the measured signal onto a low-dimensional subspace spanned by smooth functions. References [17]-[19] consider different distributed regression problems.

Our main contributions in this paper are i) to show that a key component of many distributed signal processing tasks is the application of linear operators that are unions of graph Fourier multipliers; and ii) to present a novel method to efficiently distribute the application of the graph Fourier multiplier operators to the high-dimensional signals collected by sensor networks.

To elaborate a bit, graph Fourier multiplier operators are the graph analog of filter banks, one of the most commonly used tools in digital signal processing. Multiplying a signal on the graph by one of these matrices is analogous to reshaping the signal's frequencies by multiplying it by a filter in the Fourier domain in classical signal processing. The crux of our novel distributed computational method is to approximate each graph Fourier multiplier by a truncated Chebyshev polynomial expansion. In a centralized setting, [20] shows that the truncated Chebyshev polynomial expansion efficiently approximates the application of a spectral graph wavelet transform, which is a specific example of a union of graph Fourier multipliers. In this paper, we extend the Chebyshev polynomial approximation method to the general class of unions of graph Fourier multiplier operators, and show how the recurrence properties of the Chebyshev polynomials also enable distributed application of these operators. The communication requirements for distributed computation using this method scale gracefully with the number of sensors in the network (and, accordingly, the size of the signals).

The remainder of the paper is as follows. In the next section, we provide some background from spectral graph theory. In Section III, we introduce graph Fourier multiplier operators and show how they can be efficiently approximated with shifted Chebyshev polynomials in a centralized setting. We then discuss the distributed computation of quantities involving these operators in Section IV, and provide some application examples in Section V. Section VI concludes the paper.

II. Spectral Graph Theory

Before proceeding, we introduce some basic notations and definitions from spectral graph theory [21]. We model the sensor network with an undirected, weighted graph $G = \{E, V, w\}$, which consists of a set of vertices V, a set of edges E, and a weight function $w : E \to \mathbb{R}^+$ that assigns a non-negative weight to each edge. We assume the number of sensors in the network, $N = |V|$, is finite, and the graph is connected. The adjacency (or weight) matrix A for a weighted graph G is the $N \times N$ matrix with entries $A_{m,n}$, where

$$A_{m,n} = \begin{cases} w(e), & \text{if } e \in E \text{ connects vertices } m \text{ and } n \\ 0, & \text{otherwise} \end{cases}.$$

The degree of each vertex is the sum of the weights of all the edges incident to it. We define the degree matrix D to have diagonal elements equal to the degrees, and zeros elsewhere. The non-normalized graph Laplacian is defined as $\mathcal{L} := D - A$. For any $f \in \mathbb{R}^N$ on the vertices of the graph, $\mathcal{L}$ satisfies

$$(\mathcal{L}f)(m) = \sum_{m \sim n} A_{m,n} \cdot [f(m) - f(n)],$$

where $m \sim n$ indicates vertices m and n are connected.

As the graph Laplacian $\mathcal{L}$ is a real symmetric matrix, it has a complete set of orthonormal eigenvectors. We denote these by χ_ℓ for $\ell = 0, \ldots, N-1$, with associated real, non-negative eigenvalues λ_ℓ satisfying $\mathcal{L}\chi_\ell = \lambda_\ell \chi_\ell$. Zero appears as an eigenvalue with multiplicity equal to the number of connected components of the graph [21]. Without loss of generality, we assume the eigenvalues of the Laplacian of the connected graph G to be ordered as

$$0 = \lambda_0 < \lambda_1 \leq \lambda_2 ... \leq \lambda_{N-1} := \lambda_{\max}.$$

Just as the classical Fourier transform is the expansion of a function f in terms of the eigenfunctions of the Laplace operator

$$\hat{f}(\omega) = \langle e^{i\omega x}, f \rangle = \int_{\mathbb{R}} f(x) e^{-i\omega x} \, dx,$$

the *graph Fourier transform* $\hat{f}$ of any function $f \in \mathbb{R}^N$ on the vertices of G is the expansion of f in terms of the eigenfunctions of the graph Laplacian. It is defined by

$$\hat{f}(\ell) := \langle \chi_\ell, f \rangle = \sum_{n=1}^{N} \chi_\ell^*(n) f(n), \tag{2}$$

where we adopt the convention that the inner product be conjugate-linear in the first argument. The *inverse graph Fourier transform* is given by

$$f(n) = \sum_{\ell=0}^{N-1} \hat{f}(\ell) \chi_\ell(n). \tag{3}$$

III. Chebyshev Polynomial Approximation of Graph Fourier Multipliers

In this section, we introduce graph Fourier multiplier operators, unions of graph Fourier multiplier operators, and a computationally efficient approximation to unions of graph Fourier multiplier operators based on shifted Chebyshev polynomials. All methods discussed here are for a centralized setting, and we extend them to a distributed setting in Section IV.

A. Graph Fourier Multiplier Operators

For a function f defined on the real line, a *Fourier multiplier operator* or *filter* Ψ reshapes the function's frequencies through multiplication in the Fourier domain:

$$\widehat{\Psi f}(\omega) = g(\omega) \hat{f}(\omega), \text{ for every frequency } \omega.$$

Equivalently, denoting the Fourier and inverse Fourier transforms by $\mathcal{F}$ and $\mathcal{F}^{-1}$, we have

$$\begin{aligned}\Psi f(x) &= \mathcal{F}^{-1}\Big(g(\omega)\mathcal{F}(f)(\omega)\Big)(x) \\ &= \frac{1}{2\pi}\int_{\mathbb{R}} g(\omega)\hat{f}(\omega)e^{i\omega x}\, d\omega.\end{aligned} \tag{4}$$

We can extend this straightforwardly to functions defined on the vertices of a graph (and in fact to any group with a Fourier transform) by replacing the Fourier transform and its inverse in (4) with the graph Fourier transform and its inverse, defined in (2) and (3). Namely, a *graph Fourier multiplier operator* is a linear operator $\Psi : \mathbb{R}^N \to \mathbb{R}^N$ that can be written as

$$\begin{aligned}\Psi f(n) &= \mathcal{F}^{-1}\Big(g(\lambda_\ell)\mathcal{F}(f)(\ell)\Big)(n) \\ &= \sum_{\ell=0}^{N-1} g(\lambda_\ell)\hat{f}(\ell)\chi_\ell(n).\end{aligned} \tag{5}$$

We refer to $g(\cdot)$ as the *multiplier*. A high-level intuition behind (5) is as follows. The eigenvectors corresponding to the lowest eigenvalues of the graph Laplacian are the "smoothest" in the sense that $|\chi_\ell(m) - \chi_\ell(n)|$ is small for neighboring vertices m and n. At the extreme is χ_0, which is a constant vector ($\chi_0(m) = \chi_0(n)$ for all m and n). The inverse graph Fourier transform (3) provides a representation of a signal f as a superposition of the orthonormal set of eigenvectors of the graph Laplacian. The effect of the graph Fourier multiplier operator Ψ is to modify the contribution of each eigenvector. For example, applying a multiplier $g(\cdot)$ that is 1 for all λ_ℓ below some threshold, and 0 for all λ_ℓ above the threshold is equivalent to projecting the signal onto the eigenvectors of the graph Laplacian associated with the lowest eigenvalues. This is analogous to low-pass filtering in the continuous domain. Section V contains further intuition about and examples of graph Fourier multiplier operators. For more properties of the graph Laplacian eigenvectors, see [22].

B. Unions of Graph Fourier Multiplier Operators

In order for our distributed computation method of the next section to be applicable to a wider range of applications, we can generalize slightly from graph Fourier multipliers to *unions of graph Fourier multiplier operators*. A union of graph Fourier multiplier operators is a linear operator $\Phi : \mathbb{R}^N \to \mathbb{R}^{\eta N}$ ($\eta \in \{1, 2, \ldots\}$) whose application to a function $f \in \mathbb{R}^N$ can be written as (see also Figure 1)

$$\begin{aligned}\Phi f &= [\Psi_1; \Psi_2; \ldots; \Psi_\eta]\, f \\ &= [(\Psi_1 f)_1; \ldots; (\Psi_1 f)_N; \ldots; (\Psi_\eta f)_1; \ldots; (\Psi_\eta f)_N] \\ &= [(\Phi f)_1; (\Phi f)_2; \ldots; (\Phi f)_{\eta N}]\,,\end{aligned}$$

where for every j, $\Psi_j : \mathbb{R}^N \to \mathbb{R}^N$ is a graph Fourier multiplier operator with multiplier $g_j(\cdot)$, and

$$\begin{aligned}(\Phi f)_{(j-1)N+n} = \sum_{\ell=0}^{N-1} g_j(\lambda_\ell)\hat{f}(\ell)\chi_\ell(n), \\ \text{for } j \in \{1, 2, \ldots, \eta\},\ n \in \{1, 2, \ldots, N\}.\end{aligned} \tag{6}$$

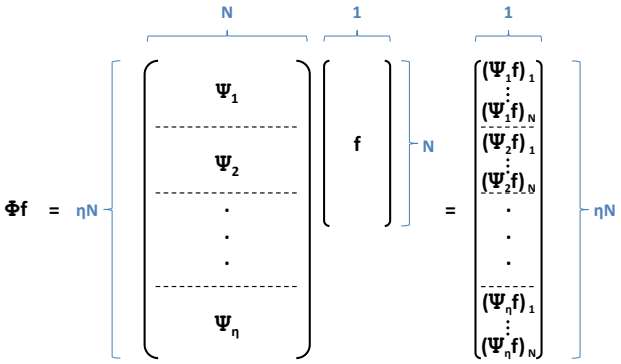

Fig. 1. Application of a union of graph Fourier multiplier operators.

C. The Chebyshev Polynomial Approximation

Exactly computing Φf requires explicit computation of the entire set of eigenvectors and eigenvalues of $\mathcal{L}$, which becomes computationally challenging as the size of the network, N, increases, even in a centralized setting. As discussed in detail in [20, Section 6], a computationally efficient approximation $\tilde{\Phi} f$ of Φf can be computed by approximating each multiplier $g_j(\cdot)$ by a truncated series of shifted Chebyshev polynomials. Doing so circumvents the need to compute the full set of eigenvectors and eigenvalues of $\mathcal{L}$. We summarize this approach below.

For $y \in [-1, 1]$, the Chebyshev polynomials $\{T_k(y)\}_{k=0,1,2,\ldots}$ are generated by

$$T_k(y) := \begin{cases} 1, & \text{if } k = 0 \\ y, & \text{if } k = 1 \\ 2yT_{k-1}(y) - T_{k-2}(y), & \text{if } k \geq 2 \end{cases}.$$

These Chebyshev polynomials form an orthogonal basis for $L^2\left([-1, 1], \frac{dy}{\sqrt{1-y^2}}\right)$. So every function h on $[-1, 1]$ that is square integrable with respect to the measure $dy/\sqrt{1-y^2}$ can be represented as $h(y) = \frac{1}{2}b_0 + \sum_{k=1}^{\infty} b_k T_k(y)$, where $\{b_k\}_{k=0,1,\ldots}$ is a sequence of Chebyshev coefficients that depends on $h(\cdot)$. For a detailed overview of Chebyshev polynomials, including the above definitions and properties, see [23]-[25].

By shifting the domain of the Chebyshev polynomials to $[0, \lambda_{\max}]$ via the transformation $x = \frac{\lambda_{\max}}{2}(y + 1)$, we can represent each multiplier as

$$g_j(x) = \frac{1}{2}c_{j,0} + \sum_{k=1}^{\infty} c_{j,k}\overline{T}_k(x), \text{ for all } x \in [0, \lambda_{\max}], \tag{7}$$

where

$$\begin{aligned}\overline{T}_k(x) &:= T_k\left(\frac{x - \alpha}{\alpha}\right), \\ \alpha &:= \frac{\lambda_{\max}}{2}, \text{ and} \\ c_{j,k} &:= \frac{2}{\pi}\int_0^{\pi} \cos(k\theta)\; g_j\Big(\alpha\big(\cos(\theta) + 1\big)\Big)\, d\theta.\end{aligned} \tag{8}$$

For $k \geq 2$, the shifted Chebyshev polynomials satisfy

$$\overline{T}_k(x) = \frac{2}{\alpha}(x-\alpha)\overline{T}_{k-1}(x) - \overline{T}_{k-2}(x).$$

Thus, for any $f \in \mathbb{R}^N$, we have

$$\overline{T}_k(\mathcal{L})f = \frac{2}{\alpha}(\mathcal{L}-\alpha I)\left(\overline{T}_{k-1}(\mathcal{L})f\right) - \overline{T}_{k-2}(\mathcal{L})f, \quad (9)$$

where $\overline{T}_k(\mathcal{L}) \in \mathbb{R}^{N\times N}$ and the n^{th} element of $\overline{T}_k(\mathcal{L})f$ is given by

$$\left(\overline{T}_k(\mathcal{L})f\right)_n := \sum_{\ell=0}^{N-1} \overline{T}_k(\lambda_\ell)\hat{f}(\ell)\chi_\ell(n). \quad (10)$$

Now, to approximate the operator Φ, we can approximate each multiplier $g_j(\cdot)$ by the first M terms in its Chebyshev polynomial expansion (7). Then, for every $j \in \{1,2,\ldots,\eta\}$ and $n \in \{1,2,\ldots,N\}$, we have

$$\begin{aligned}
&\left(\tilde{\Phi}f\right)_{(j-1)N+n} \\
&\quad := \left(\frac{1}{2}c_{j,0}f + \sum_{k=1}^{M} c_{j,k}\overline{T}_k(\mathcal{L})f\right)_n \quad (11)\\
&\quad \overset{(3),(10)}{=} \sum_{\ell=0}^{N-1}\left[\frac{1}{2}c_{j,0} + \sum_{k=1}^{M} c_{j,k}\overline{T}_k(\lambda_\ell)\right]\hat{f}(\ell)\chi_\ell(n)\\
&\quad \approx \sum_{\ell=0}^{N-1}\left[\frac{1}{2}c_{j,0} + \sum_{k=1}^{\infty} c_{j,k}\overline{T}_k(\lambda_\ell)\right]\hat{f}(\ell)\chi_\ell(n)\\
&\quad \overset{(7)}{=} \sum_{\ell=0}^{N-1} g_j(\lambda_\ell)\hat{f}(\ell)\chi_\ell(n)\\
&\quad \overset{(6)}{=} (\Phi f)_{(j-1)N+n}.
\end{aligned}$$

To recap, we propose to compute $\tilde{\Phi}f$ by first computing the Chebyshev coefficients $\{c_{j,k}\}_{j=1,2,\ldots,\eta;\ k=1,2,\ldots,M}$ according to (8), and then computing the sum in (11). The computational benefit of the Chebyshev polynomial approximation arises in (11) from the fact the vector $\overline{T}_k(\mathcal{L})f$ can be computed recursively from $\overline{T}_{k-1}(\mathcal{L})f$ and $\overline{T}_{k-2}(\mathcal{L})f$ according to (9). The computational cost of doing so is dominated by the matrix-vector multiplication of the graph Laplacian $\mathcal{L}$, which is proportional to the number of edges, $|E|$ [20]. Therefore, if the underlying communication graph is sparse (i.e., $|E|$ scales linearly with the network size N), it is far more computationally efficient to compute $\tilde{\Phi}f$ than Φf. Finally, we note that in practice, setting the Chebyshev approximation order M to around 20 results in $\tilde{\Phi}$ approximating Φ very closely in all of the applications we have examined.

IV. Distributed Computation

In the previous section, we showed that the Chebyshev polynomial approximation to a union of graph Fourier multipliers provides computational efficiency gains, even in a centralized computation setting. In this section, we discuss the second benefit of the Chebyshev polynomial approximation: it is easily distributable.

A. Distributed Computation of $\tilde{\Phi}f$

We consider the following scenario. There is a network of N nodes, and each node n begins with the following knowledge:

- $f(n)$, the n^{th} component of the signal f
- The identity of its neighbors, and the weights of the graph edges connecting itself to each of its neighbors
- The first M Chebyshev coefficients, $c_{j,k}$, for $j \in \{1,2,\ldots,\eta\}$ and $k \in \{0,1,2,\ldots,M\}$. These can either be computed centrally according to (8) and then transmitted throughout the network, or each node can begin with knowledge of the multipliers, $\{g_j(\cdot)\}_{j=1,2,\ldots,\eta}$, and precompute the Chebyshev coefficients according to (8)
- An upper bound on $\lambda_{\max}$, the largest eigenvalue of the graph Laplacian. This bound need not be tight, so we can precompute a bound such as $\lambda_{\max} \leq \max\{d(m) + d(n); m \sim n\}$, where $d(n)$ is the degree of node n [26]

The task is for each network node n to compute

$$\left\{\left(\tilde{\Phi}f\right)_{(j-1)N+n}\right\}_{j=1,2,\ldots,\eta} \quad (12)$$

by iteratively exchanging messages with its local neighbors in the network and performing some computations.

As a result of (11), for node n to compute the desired sequence in (12), it suffices to learn $\left\{\left(\overline{T}_k(\mathcal{L})f\right)_n\right\}_{k=1,2,\ldots,M}$. Note that $\left(\overline{T}_1(\mathcal{L})f\right)_n = \left(\frac{1}{\alpha}(\mathcal{L}-\alpha I)f\right)_n$ and $\mathcal{L}_{n,m} = 0$ for all nodes m that are not neighbors of node n. Thus, to compute $\left(\overline{T}_1(\mathcal{L})f\right)_n$, sensor node n just needs to receive $f(m)$ from all neighbors m. So once all nodes send their component of the signal to their neighbors, they are able to compute their respective components of $\overline{T}_1(\mathcal{L})f$. In the next step, each node n sends the newly computed quantity $\left(\overline{T}_1(\mathcal{L})f\right)_n$ to all of its neighbors, enabling the distributed computation of $\overline{T}_2(\mathcal{L})f$ according to (9). The iterative process of computation and local communication continues for M rounds until each node n has computed the required sequence $\left\{\left(\overline{T}_k(\mathcal{L})f\right)_n\right\}_{k=1,2,\ldots,M}$. In all, $2M|E|$ messages of length 1 are required for every node n to compute its sequence of coefficients in (12) in a distributed fashion. This distributed computation process is summarized in Algorithm 1.

An important point to emphasize again is that although the operator Φ and its approximation $\tilde{\Phi}$ are defined through the eigenvectors of the graph Laplacian, the Chebyshev polynomial approximation helps the sensor nodes apply the operator to the signal without explicitly computing (individually or collectively) the eigenvalues or eigenvectors of the Laplacian, other than the upper bound on its spectrum. Rather, they initially communicate their component of the signal to their neighbors, and then communicate simple weighted combinations of the messages received in the previous stage in subsequent iterations. In this way, information about each component of the signal f diffuses through the network without direct communication between non-neighboring nodes.

B. Distributed Computation of $\tilde{\Phi}^* a$

The application of the adjoint $\tilde{\Phi}^*$ of the Chebyshev polynomial approximate operator $\tilde{\Phi}$ can also be computed in a

Algorithm 1 Distributed Computation of $\tilde{\Phi} f$

Inputs at node n: f_n, $\mathcal{L}_{n,m}$ $\forall m$, $\{c_{k,j}\}_{j=1,2,\ldots,\eta;\ k=0,1,\ldots,M}$, and $\lambda_{\max}$

Outputs at node n: $\left\{\left(\tilde{\Phi} f\right)_{(j-1)N+n}\right\}_{j=1,2,\ldots,\eta}$

Set $\left(\overline{T}_0(\mathcal{L})f\right)_n = f_n$

Transmit f_n to all neighbors $\mathcal{N}_n := \{m : \mathcal{L}_{n,m} < 0\}$

Receive f_m from all neighbors $\mathcal{N}_n$

Compute and store

$$\left(\overline{T}_1(\mathcal{L})f\right)_n = \sum_{m\in\mathcal{N}_n\cup n} \frac{1}{\alpha}\mathcal{L}_{n,m} f_m - f_n$$

for $k = 2, \ldots, M$ **do**

  Transmit $\left(\overline{T}_{k-1}(\mathcal{L})f\right)_n$ to all neighbors $\mathcal{N}_n$

  Receive $\left(\overline{T}_{k-1}(\mathcal{L})f\right)_m$ from all neighbors $\mathcal{N}_n$

  Compute and store

$$\left(\overline{T}_k(\mathcal{L})f\right)_n = \sum_{m\in\mathcal{N}_n\cup n} \frac{2}{\alpha}\mathcal{L}_{n,m}\left(\overline{T}_{k-1}(\mathcal{L})f\right)_m - 2\left(\overline{T}_{k-1}(\mathcal{L})f\right)_n - \left(\overline{T}_{k-2}(\mathcal{L})f\right)_n$$

end for

for $j \in \{1, 2, \ldots, \eta\}$ **do**

  Output

$$\left(\tilde{\Phi} f\right)_{(j-1)N+n} = \frac{1}{2}c_{j,0} f_n + \sum_{k=1}^{M} c_{j,k}\left(\overline{T}_k(\mathcal{L})f\right)_n$$

end for

distributed manner. Let

$$a = [a_1; a_2; \ldots; a_\eta] \in \mathbb{R}^{\eta N},$$

where $a_j \in \mathbb{R}^N$. Then it is straightforward to show that

$$\left(\tilde{\Phi}^* a\right)_n = \sum_{j=1}^{\eta}\left(\frac{1}{2}c_{j,0}a_j + \sum_{k=1}^{M} c_{j,k}\overline{T}_k(\mathcal{L})a_j\right)_n. \quad (13)$$

We assume each node n starts with knowledge of $a_j(n)$ for all $j \in \{1, 2, \ldots, \eta\}$. For each $j \in \{1, 2, \ldots, \eta\}$, the distributed computation of the corresponding term on the right-hand side of (13) is done in an analogous manner to the distributed computation of $\tilde{\Phi} f$ discussed above. Since this has to be done for each j, $2M|E|$ messages, each a vector of length η, are required for every node n to compute $\left(\tilde{\Phi}^* a\right)_n$.

C. Distributed Computation of $\tilde{\Phi}^*\tilde{\Phi} f$

Using the property of Chebyshev polynomials that

$$T_k(x)T_{k'}(x) = \frac{1}{2}\left[T_{k+k'}(x) + T_{|k-k'|}(x)\right],$$

we can write (see [20] for a similar calculation)

$$\left(\tilde{\Phi}^*\tilde{\Phi} f\right)_n = \left(\frac{1}{2}d_0 f + \sum_{k=1}^{2M} d_k\overline{T}_k(\mathcal{L})f\right)_n.$$

Therefore, with each node n starting with $f(n)$ as in Section IV-A, the nodes can compute $\tilde{\Phi}^*\tilde{\Phi} f$ in a distributed manner using $4M|E|$ messages of length 1, with each node n finishing with knowledge of $\left(\tilde{\Phi}^*\tilde{\Phi} f\right)_n$.

V. Application Examples

In this section, we provide more detailed explanations of how the Chebyshev polynomial approximation of graph Fourier multipliers can be used in the context of specific distributed signal processing tasks.

A. Distributed Smoothing

Perhaps the simplest example application is distributed smoothing with the heat kernel as the graph Fourier multiplier. One way to smooth a signal $y \in \mathbb{R}^N$ is to compute $H_t y$, where, for a fixed t, $(H_t y)(n) := \sum_{\ell=0}^{N-1} e^{-t\lambda_\ell}\hat{y}(\ell)\chi_\ell(n)$. H_t clearly satisfies our definition of a graph Fourier multiplier operator (with $\eta = 1$). In the context of a centralized image smoothing application, [13] discusses in detail the *heat kernel*, H_t, and its relationship to classical Gaussian filtering. Similar to the example at the end of Section III-A, the main idea is that the multiplier $e^{-t\lambda_\ell}$ acts as a low-pass filter that attenuates the higher frequency (less smooth) components of y.

Now, to perform distributed smoothing, we just need to compute $\tilde{H}_t y$ in a distributed manner according to Algorithm 1, where $\tilde{H}_t$ is the shifted Chebyshev polynomial approximation to the graph Fourier multiplier operator H_t.

B. Distributed Regularization

Regularization is a common signal processing technique to solve ill-posed inverse problems using *a priori* information about a target signal to recover it accurately. Here we use regularization to solve the distributed denoising task discussed in Section I, starting with a noisy signal $y \in \mathbb{R}^N$ defined on a graph of N sensors. The prior belief we want to enforce is that the target signal is smooth with respect to the underlying graph topology. The class of regularization terms we consider is $f^{\mathrm{T}}\mathcal{L}^r f$ for $r \geq 1$, and the resulting regularization problem has the form

$$\operatorname*{argmin}_f \frac{\tau}{2}\|f - y\|_2^2 + f^{\mathrm{T}}\mathcal{L}^r f. \quad (14)$$

To see intuitively why incorporating such a regularization term into the objective function encourages smooth signals (with $r = 1$ as an example), note that $f^{\mathrm{T}}\mathcal{L} f = 0$ if and only if f is constant across all vertices, and, more generally

$$f^{\mathrm{T}}\mathcal{L} f = \frac{1}{2}\sum_{n\in V}\sum_{m\sim n} A_{m,n}\left[f(m) - f(n)\right]^2,$$

so $f^{\mathrm{T}}\mathcal{L} f$ is small when the signal f has similar values at neighboring vertices with large weights (i.e., it is smooth).

We now show how our novel method is useful in solving this distributed regularization problem.

***Proposition** 1:* The solution to (14) is given by Ry, where R is a graph Fourier multiplier operator of the form (5), with multiplier $g(\lambda_\ell) = \frac{\tau}{\tau + 2\lambda_\ell^r}$.[1]

Proof: The objective function in (14) is convex in f. Differentiating it with respect to f, any solution f_* to

$$\mathcal{L}^r f_* + \frac{\tau}{2}(f_* - y) = 0 \tag{15}$$

is a solution to (14).[2] Taking the graph Fourier transform of (15) yields

$$\widehat{\mathcal{L}^r f_*}(\ell) + \tfrac{\tau}{2}\left(\widehat{f}_*(\ell) - \hat{y}(\ell)\right) = 0, \quad \forall \ell \in \{0, 1, \ldots, N-1\}. \tag{16}$$

From the real, symmetric nature of $\mathcal{L}$ and the definition of the Laplacian eigenvectors ($\mathcal{L}\chi_\ell = \lambda_\ell \chi_\ell$), we have:

$$\widehat{\mathcal{L}^r f_*}(\ell) = \chi_\ell^* \mathcal{L}^r f_* = (\mathcal{L}^r \chi_\ell)^* f_* = \lambda_\ell^r \chi_\ell^* f_* = \lambda_\ell^r \widehat{f}_*(\ell). \tag{17}$$

Substituting (17) into (16) and rearranging, we have

$$\widehat{f}_*(\ell) = \frac{\tau}{\tau + 2\lambda_\ell^r}\hat{y}(\ell), \ \forall \ell \in \{0, 1, \ldots, N-1\}. \tag{18}$$

Finally, taking the inverse graph Fourier transform of (18), we have

$$f_*(n) = \sum_{\ell=0}^{N-1} \widehat{f}_*(\ell)\chi_\ell(n) = \sum_{\ell=0}^{N-1} \left[\frac{\tau}{\tau + 2\lambda_\ell^r}\right] \hat{y}(\ell)\chi_\ell(n), \quad \forall n \in \{1, 2, \ldots, N\}. \tag{19}$$

■

So, one way to do distributed denoising is to compute $\tilde{R}y$, the Chebyshev polynomial approximation of Ry, in a distributed manner via Algorithm 1. We show this now with a numerical example. We place 500 sensors randomly in the $[0,1] \times [0,1]$ square. We then construct a weighted graph according to the thresholded Gaussian kernel weighting (1) with $\sigma = 0.074$ and $\kappa = 0.600$, so that two sensor nodes are connected if their physical separation is less than 0.075. We create a smooth 500-dimensional signal with the n^{th} component given by $f_n^0 = n_x^2 + n_y^2 - 1$, where n_x and n_y are node n's x and y coordinates in $[0,1] \times [0,1]$. One instance of such a network and signal f^0 are shown in Figure 2, and the eigenvectors of the graph Laplacian are shown in Figure 3.

Next, we corrupt each component of the signal f^0 with uncorrelated additive Gaussian noise with mean zero and standard deviation 0.5. Then we apply the graph Fourier multiplier operator $\tilde{R}$, the Chebyshev polynomial approximation to R from Proposition 1, with $\tau = r = 1$. The multiplier and its Chebyshev polynomial approximations are shown in Figure 4, and the denoised signal $\tilde{R}y$ is shown in Figure 5. We repeated this entire experiment 1000 times, with a new random graph and random noise each time, and the average mean square error for the denoised signals was 0.013, as compared to 0.250 average mean square error for the noisy signals.

[1]This filter $g(\lambda_\ell)$ is the graph analog of a first-order Bessel filter from classical signal processing of functions on the real line.

[2]In the case $r = 1$, the optimality equation (15) corresponds to the optimality equation in [12, Section III-A] with $p = 2$ in that paper.

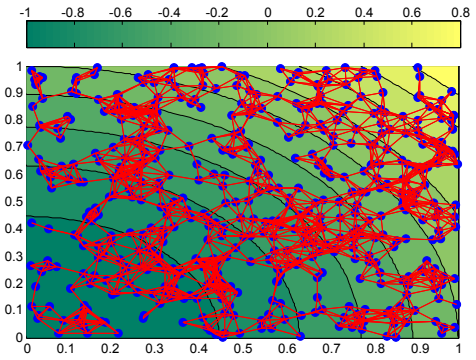

Fig. 2. A network of 500 sensors placed randomly in the $[0,1] \times [0,1]$ square. The background colors represent the values of the smooth signal f^0.

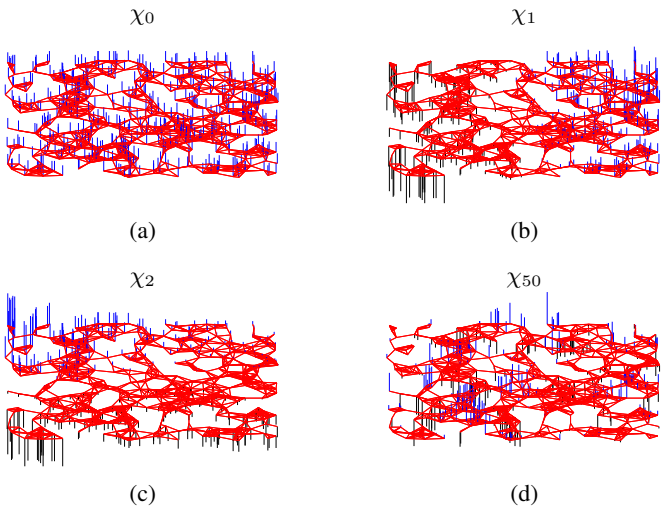

Fig. 3. Some eigenvectors of the Laplacian of the graph shown in Figure 2. The blue bars represent positive values and the black bars negative values. (a) χ_0, the constant eigenvector associated with $\lambda_0 = 0$. (b) χ_1, the *Fiedler vector* associated with the lowest strictly positive eigenvalue, nicely separates the graph into two components. (c) χ_2 is also a smooth eigenvector. (d) χ_{50} is far less smooth with some large differences across neighboring nodes.

We conclude this section by returning to the distributed binary classification task discussed in the introduction. In [9], Belkin *et al.* show that the regularizer $f^T \mathcal{L}^r f$ also works well in graph-based semi-supervised learning. One approach to distributed binary classification is to let y_n be the labels (-1 or 1) of those nodes who know their labels, and 0 otherwise. Then the nodes compute $\tilde{R}y$ in a distributed manner via Algorithm 1, and each node n sets it label to 1 if $(\tilde{R}y)_n \geq 0$ and -1 otherwise. We believe our approach to distributedly applying graph Fourier multipliers can also be used for more general learning problems, but we leave this for future work.

C. Distributed Wavelet Denoising

In this section, we consider an alternate method of distributed denoising that may be better suited to signals that are piecewise smooth on the graph, but not necessarily globally smooth. The setup is the same as in Section V-B, with a noisy signal $y \in \mathbb{R}^N$, and each sensor n observing y_n. Instead of

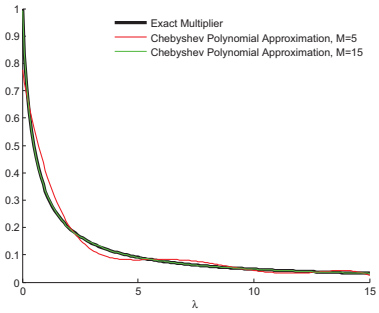

Fig. 4. The regularizing multiplier $\frac{\tau}{\tau+2\lambda_\ell^r}$ associated with the graph Fourier multiplier operator R from Proposition 1. Here, $r = \tau = 1$. Shifted Chebyshev polynomial approximations to the multiplier are shown for different values of the approximation order M.

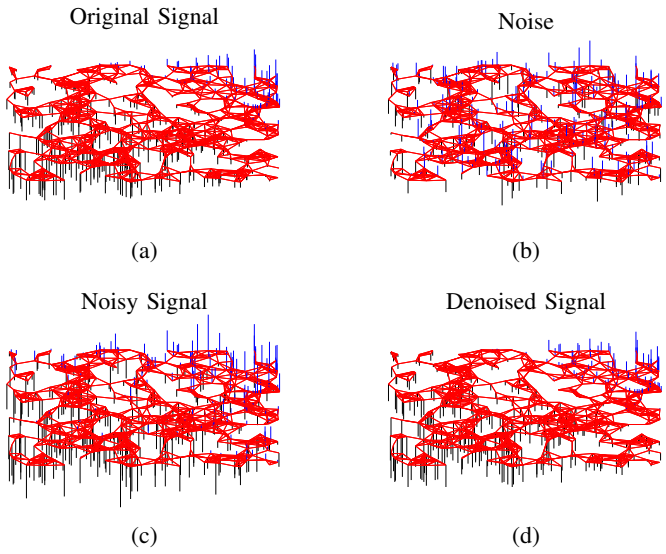

Fig. 5. A denoising example on the graph shown in Figure 2, using the regularizing multiplier shown in Figure 4. (a) The original signal $n_x^2+n_y^2-1$, where n_x and n_y are the x and y coordinates of sensor node n. (b) The additive Gaussian noise. (c) The noisy signal y. (d) The denoised signal $\tilde{R}y$.

starting with a prior that the signal is globally smooth, we start with a prior belief that the signal is sparse in the spectral graph wavelet domain [20]. The spectral graph wavelet transform, W, defined in [20], is precisely of the form of Φ in (6). Namely, it is composed of one multiplier, $h(\cdot)$, that acts as a low-pass filter to stably represent the signal's low frequency content, and J wavelet operators, defined by $g_j(\lambda_\ell) = g(t_j\lambda_\ell)$, where $\{t_j\}_{j=1,2,\ldots,J}$ is a set of scales and $g(\cdot)$ is the wavelet multiplier that acts as a band-pass filter.

The most common way to incorporate a sparse prior in a centralized setting is to regularize via a weighted version of the *least absolute shrinkage and selection operator (lasso)* [27], also called *basis pursuit denoising* [28]:

$$\underset{a}{\text{argmin}} \ \frac{1}{2}\|y - W^*a\|_2^2 + \|a\|_{1,\mu} \ , \qquad (20)$$

where $\|a\|_{1,\mu} := \sum_{i=1}^{N(J+1)} \mu_i |a_i|$. The optimization problem in (20) can be solved for example by iterative soft thresholding [29]. The initial estimate of the wavelet coefficients $a^{(0)}$ is arbitrary, and at each iteration of the soft thresholding algorithm, the update of the estimated wavelet coefficients is given by

$$a_i^{(k)} = \mathcal{S}_{\mu_i\tau}\left(\left(a^{(k-1)} + \tau W\left[y - W^*a^{(k-1)}\right]\right)_i\right), \quad i = 1,2,\ldots,N(J+1);\ k = 1,2,\ldots \qquad (21)$$

where τ is the step size and $\mathcal{S}_{\mu_i\tau}$ is the shrinkage or soft thresholding operator

$$\mathcal{S}_{\mu_i\tau}(z) := \begin{cases} 0 & , \text{ if } \mid z \mid \leq \mu_i\tau \\ z - \text{sgn}(z)\mu_i\tau & , \text{ o.w.} \end{cases} .$$

The iterative soft thresholding algorithm converges to a_*, the minimizer of (20), if $\tau < \frac{2}{\|W^*\|^2}$ [30]. The final denoised estimate of the signal is then given by W^*a_*.

We now turn to the issue of how to implement the above algorithm in a distributed fashion by sending messages between neighbors in the network. One option would be to use the distributed lasso algorithm of [19], which is a special case of the Alternating Direction Method of Multipliers [31, p. 253]. In every iteration of that algorithm, each node transmits its current estimate of *all* the wavelet coefficients to its local neighbors. With a transform the size of the spectral graph wavelet transform, this requires $2|E|$ total messages at every iteration, with each message being a vector of length $N(J+1)$. A method where the amount of communicated information does not grow with N (beyond the number of edges, $|E|$) would be highly preferable.

The Chebyshev polynomial approximation of the spectral graph wavelet transform allows us to accomplish this goal. Our approach is to approximate W by $\tilde{W}$, and use the distributed implementation of the approximate wavelet transform and its adjoint to perform iterative soft thresholding. In the first soft thresholding iteration, each node n must learn $(\tilde{W}y)_{(j-1)N+n}$ at all scales j, via Algorithm 1. These coefficients are then stored for future iterations. In the k^{th} iteration, each node n must learn the $J+1$ coefficients of $\tilde{W}\tilde{W}^*a^{(k-1)}$ centered at n, by sequentially applying the operators $\tilde{W}^*$ and $\tilde{W}$ in a distributed manner via the methods of Sections IV-B and IV-A, respectively. Finally, when a stopping criterion for the soft thresholding is satisfied, the adjoint operator $\tilde{W}^*$ is applied again in a distributed manner to the resulting coefficients $\tilde{a}_*$, and node n's denoised estimate of its signal is $\left(\tilde{W}^*\tilde{a}_*\right)_n$.

We now examine the communication requirements of this approach. Recall from Section IV that $2M|E|$ messages of length 1 are required to compute $\tilde{W}y$ in a distributed fashion. Distributed computation of $\tilde{W}\tilde{W}^*a^{(k-1)}$, the other term needed in the iterative thresholding update (21), requires $2M|E|$ messages of length $J+1$ and $2M|E|$ messages of length 1. The final application of the adjoint operator $\tilde{W}^*$ to recover the denoised signal estimates requires another $2M|E|$ messages, each a vector of length $J+1$. Therefore, the Chebyshev polynomial approximation to the spectral graph wavelet transform enables us to iteratively solve the weighted lasso in a distributed manner where the communication workload

only scales with the size of the network through $|E|$, and is otherwise independent of the network dimension N.

VI. Concluding Remarks and Future Work

We presented a novel method to distribute a class of linear operators called unions of graph Fourier multiplier operators. The main idea is to approximate the graph Fourier multipliers by Chebyshev polynomials, whose recurrence relations make them readily amenable to distributed computation in a sensor network. Key takeaways from the discussion and application examples include:

- A number of distributed signal processing tasks can be represented as distributed applications of unions of graph Fourier multiplier operators (and their adjoints) to signals on weighted graphs. Examples include distributed smoothing, denoising, and semi-supervised learning.
- The graph Fourier multiplier operators are the graph analog of filter banks, as they reshape functions' frequencies through multiplication in the Fourier domain.
- The amount of communication required to perform the distributed computations only scales with the size of the network through the number of edges of the communication graph, which is usually sparse. Therefore, the method is well suited to large-scale sensor networks.

Our ongoing work includes extending the scope and depth of our application examples. In addition to considering more applications and larger size networks, we plan a more thorough empirical comparison of the computation and communication requirements of the approach described in this paper to alternative distributed optimization methods. The second major line of ongoing work is to analyze robustness issues that arise in real networks. For instance, we would like to incorporate quantization and communication noise into the sensor network model, in order to see how these propagate when using the Chebyshev polynomial approximation approach to distributed signal processing tasks. It is also important to analyze the effects of a sensor node dropping out of the network or communicating nodes losing synchronicity to ensure that the proposed method is stable to these disturbances.

References

[1] M. Rabbat and R. Nowak, "Distributed optimization in sensor networks," in *Proc. Int. Symp. Inf. Process. Sensor Netw.*, Berkeley, CA, Apr. 2004, pp. 20–27.

[2] J. B. Predd, S. R. Kulkarni, and H. V. Poor, "Distributed learning in wireless sensor networks," *IEEE Signal Process. Mag.*, vol. 23, pp. 56–69, Jul. 2006.

[3] R. Olfati-Saber, J. Fax, and R. Murray, "Consensus and cooperation in networked multi-agent systems," *Proc. IEEE*, vol. 95, no. 1, pp. 215–233, Jan. 2007.

[4] A. G. Dimakis, S. Kar, J. M. F. Moura, M. G. Rabbat, and A. Scaglione, "Gossip algorithms for distributed signal processing," *Proc. IEEE*, vol. 98, no. 11, pp. 1847–1864, Nov. 2010.

[5] A. J. Smola and R. Kondor, "Kernels and regularization on graphs," in *Proc. Ann. Conf. Comp. Learn. Theory*, ser. Lect. Notes Comp. Sci., B. Schölkopf and M. Warmuth, Eds. Springer, 2003, pp. 144–158.

[6] D. Zhou and B. Schölkopf, "A regularization framework for learning from graph data." in *Proc. ICML Workshop Stat. Relat. Learn. and Its Connections to Other Fields*, Jul. 2004, pp. 132–137.

[7] ——, "Regularization on discrete spaces," in *Pattern Recogn.*, ser. Lect. Notes Comp. Sci., W. G. Kropatsch, R. Sablatnig, and A. Hanbury, Eds. Springer, 2005, vol. 3663, pp. 361–368.

[8] X. Zhu and Z. Ghahramani, "Semi-supervised learning using Gaussian fields and harmonic functions," in *Proc. Int. Conf. Mach. Learn.*, Washington, D.C., Aug. 2003, pp. 912–919.

[9] M. Belkin, I. Matveeva, and P. Niyogi, "Regularization and semi-supervised learning on large graphs," in *Learn. Theory*, ser. Lect. Notes Comp. Sci. Springer-Verlag, 2004, pp. 624–638.

[10] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *Adv. Neural Inf. Process. Syst.*, S. Thrun, L. Saul, and B. Schölkopf, Eds. MIT Press, 2004, pp. 321–328.

[11] S. Bougleux, A. Elmoataz, and M. Melkemi, "Discrete regularization on weighted graphs for image and mesh filtering," in *Scale Space Var. Methods Comp. Vision*, ser. Lect. Notes Comp. Sci., F. Sgallari, A. Murli, and N. Paragios, Eds. Springer, 2007, vol. 4485, pp. 128–139.

[12] A. Elmoataz, O. Lezoray, and S. Bougleux, "Nonlocal discrete regularization on weighted graphs: a framework for image and manifold processing," *IEEE Trans. Image Process.*, vol. 17, pp. 1047–1060, Jul. 2008.

[13] F. Zhang and E. R. Hancock, "Graph spectral image smoothing using the heat kernel," *Pattern Recogn.*, vol. 41, pp. 3328–3342, Nov. 2008.

[14] G. Peyré, S. Bougleux, and L. Cohen, "Non-local regularization of inverse problems," in *Proc. ECCV'08*, ser. Lect. Notes Comp. Sci., D. A. Forsyth, P. H. S. Torr, and A. Zisserman, Eds. Springer, 2008, pp. 57–68.

[15] R. Wagner, V. Delouille, and R. Baraniuk, "Distributed wavelet denoising for sensor networks," in *Proc. IEEE Int. Conf. Dec. and Contr.*, San Diego, CA, Dec. 2006, pp. 373–379.

[16] S. Barbarossa, G. Scutari, and T. Battisti, "Distributed signal subspace projection algorithms with maximum convergence rate for sensor networks with topological constraints," in *Proc. IEEE Int. Conf. Acc., Speech, and Signal Process.*, Taipei, Apr. 2009, pp. 2893–2896.

[17] C. Guestrin, P. Bodik, R. Thibaux, M. Paskin, and S. Madden, "Distributed regression: an efficient framework for modeling sensor network data," in *Proc. Int. Symp. Inf. Process. Sensor Netw.*, Berkeley, CA, Apr. 2004, pp. 1–10.

[18] J. B. Predd, S. R. Kulkarni, and H. V. Poor, "A collaborative training algorithm for distributed learning," *IEEE Trans. Inf. Theory*, vol. 55, no. 4, pp. 1856–1871, Apr. 2009.

[19] G. Mateos, J.-A. Bazerque, and G. B. Giannakis, "Distributed sparse linear regression," *IEEE Trans. Signal Process.*, vol. 58, no. 10, pp. 5262–5276, Oct. 2010.

[20] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Appl. Comput. Harmon. Anal.*, vol. 30, no. 2, pp. 129–150, Mar. 2011.

[21] F. K. Chung, *Spectral Graph Theory*. Vol. 92 of the CBMS Regional Conference Series in Mathematics, AMS Bokstore, 1997.

[22] T. Bıyıkoğlu, J. Leydold, and P. F. Stadler, *Laplacian Eigenvectors of Graphs*. Lecture Notes in Mathematics, vol. 1915, Springer, 2007.

[23] J. C. Mason and D. C. Handscomb, *Chebyshev Polynomials*. Chapman and Hall, 2003.

[24] G. M. Phillips, *Interpolation and Approximation by Polynomials*. CMS Books in Mathematics, Springer-Verlag, 2003.

[25] T. J. Rivlin, *Chebyshev Polynomials*. Wiley-Interscience, 1990.

[26] W. N. Anderson and T. D. Morley, "Eigenvalues of the Laplacian of a graph," *Linear Multilinear Algebra*, vol. 18, no. 2, pp. 141–145, 1985.

[27] R. Tibshirani, "Regression shrinkage and selection via the Lasso," *J. Royal. Statist. Soc. B*, vol. 58, no. 1, pp. 267–288, 1996.

[28] S. Chen, D. Donoho, and M. Saunders, "Atomic decomposition by basis pursuit," *SIAM J. Sci. Comp.*, vol. 20, no. 1, pp. 33–61, Aug. 1998.

[29] I. Daubechies, M. Defrise, and C. De Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Commun. Pure Appl. Math.*, vol. 57, no. 11, pp. 1413–1457, Nov. 2004.

[30] P. L. Combettes and V. R. Wajs, "Signal recovery by proximal forward-backward splitting," *Multiscale Model. Sim.*, vol. 4, no. 4, pp. 1168–1200, Nov. 2005.

[31] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Prentice-Hall, 1989.