Design of graph filters and filterbanks

Nicolas Tremblay, Paulo Gonçalves, Pierre Borgnat

Design of graph filters and filterbanks

Nicolas Tremblay[a,*], Paulo Gonçalves[a,] and Pierre Borgnat[a,†]**
[*]Univ. Grenoble Alpes, CNRS, GIPSA-lab, Grenoble, France
[**]Université Lyon, ENS de Lyon, Univ Lyon 1, CNRS, Inria, LIP & IXXI, Lyon, France
[†]Université Lyon, ENS de Lyon, Univ Lyon 1, CNRS, Laboratoire de Physique & IXXI, Lyon, France
[a]Corresponding: `nicolas.tremblay@gipsa-lab.grenoble-inp.fr`,
`paulo.goncalves@ens-lyon.fr`,
`pierre.borgnat@ens-lyon.fr`

CHAPTER OUTLINE HEAD

ABSTRACT

Basic operations in graph signal processing consist in processing signals indexed on graphs either by filtering them or by changing their domain of representation, in order to better extract or analyze the important information they contain. The aim of this chapter is to review general concepts underlying such filters and representations of graph signals. We first recall the different Graph Fourier Transforms that have been developed in the literature, and show how to introduce a notion of frequency analysis for graph signals by looking at their variations. Then, we move to the introduction of graph filters, that are defined like the classical equivalent for 1D signals or 2D images, as linear systems which operate on each frequency of a signal. Some examples of filters and of their implementations are given. Finally, as alternate representations of graph signals, we focus on multiscale transforms that are defined from filters. Continuous multiscale transforms such as spectral wavelets on graphs are reviewed, as well as the versatile approaches of filterbanks on graphs. Several variants of graph filterbanks are discussed, for structured as well as arbitrary graphs, with a focus on the central point of the choice of the decimation or aggregation operators.

Keywords: graph signal processing, graph filters, filterbanks, wavelets, multi-resolution

0.1 GRAPH FOURIER TRANSFORM AND FREQUENCIES

0.1.1 INTRODUCTION

Graph Signal Processing (GSP) has been introduced in the recent past using at least two complementary formalisms: on the one hand, the discrete signal processing on graphs [1] (see also Chapter II.1) which emphasizes the adjacency matrix as a shift operator on graphs and develops an equivalent of Discrete Signal Processing (DSP) for signals on graphs; and on the other hand, the approaches rooted in graph spectral analysis, which rely on the spectral properties of a Laplacian matrix on a graph [2, 3, 4]. Both approaches yield a harmonic analysis of graph signals via the definition of a Graph Fourier Transform : an operator projecting signals in the spectral domain of the chosen matrix. While the technical details vary, and some interpretations in the vertex domain may differ, the fundamental objective of both approaches is to decompose a signal onto components of different frequencies and to design filters that can extract or modify parts of a graph signal according to these frequencies, e.g. providing notions of low-pass, band-pass, or high-pass filters for graph signals. In this section, both approaches (along with other variations) are seen as specific instances of a general guideline for defining Graph Fourier Transform and its associated frequency analysis.

Notations.

Vectors are written in bold with small letters, and matrices in bold and capital letters. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$ be a graph with $\mathcal{V}$ the set of N nodes, $\mathcal{E}$ the set of edges, and $\mathbf{A}$ the weighted adjacency matrix in $\mathbb{R}^{N\times N}$. If $\mathbf{A}_{ij} = 0$, there is no connection from node

i to node j, otherwise, $\mathbf{A}_{ij}$ is the weight of the edge starting from i and pointing[1] to j. If an undirected edge exists between i and j, then $\mathbf{A}_{ij} = \mathbf{A}_{ji}$. We restrict ourselves to adjacency matrices with positive or null entries: $\mathbf{A}_{ij} \geq 0$. Also, the symbol $\mathbf{I}$ denotes the identity matrix (its dimension should be clear with the context), and $\boldsymbol{\delta}_i$ is a vector whose i-th entry is equal to 1 while all other entries are equal to 0. Finally, we will denote by $|\mathcal{E}|$ the number of edges in the graph.

0.1.2 GRAPH FOURIER TRANSFORM

Definition 1 (graph signal)**.** A graph signal is a vector $\mathbf{x} \in \mathbb{R}^N$ whose component x_i is considered to be defined on vertex i.

A Graph Fourier Transform (GFT) is defined via a choice of reference operator[2] admitting a spectral decomposition. Representing a graph signal in this spectral domain is interpreted as a GFT. We review the standard properties and the various definitions proposed in the literature.

Consider a matrix $\mathbf{R} \in \mathbb{R}^{N\times N}$. To be admissible as a reference operator for the graph, it is often required that for any pair of nodes $i \neq j$, $\mathbf{R}_{ij}$ and $\mathbf{R}_{ji}$ are equal to zero if i and j are not connected, as this will help for efficient implementations of filters (see Section 0.2.4). We assume furthermore that $\mathbf{R}$ is diagonalizable in $\mathbb{C}$. In fact, if $\mathbf{R}$ is not diagonalizable, one needs to consider Jordan's decomposition, which is out-of-scope of this chapter. We refer the reader to [1] and Chapter II.1 for technical details on how to handle this case. Nevertheless, in practice, we claim that it often suffices to consider only diagonalizable operators, since: *i)* diagonalizable matrices in $\mathbb{C}$ are dense in the space of matrices; and *ii)* graphs under consideration are generally measured (should they model social, sensor or biological networks,...) with some inherent noise. Thus, if one ends up unluckily with a non-diagonalizable matrix, a small perturbation within the noise level will make it diagonalizable – provided the graphs have no specific regularities that need to be kept. Still, we may assume with only a small loss of generality that the reference operator $\mathbf{R}$ has a spectral decomposition:

$$\mathbf{R} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^{-1}, \tag{0.1}$$

with $\mathbf{U}$ and $\boldsymbol{\Lambda}$ in $\mathbb{C}^{N\times N}$. The columns of $\mathbf{U}$, denoted as $\mathbf{u}_k$, are the right eigenvectors of $\mathbf{R}$, while the rows of $\mathbf{U}^{-1}$, denoted as $\mathbf{v}_k^\top$, are its left eigenvectors. $\boldsymbol{\Lambda}$ is the diagonal matrix of the eigenvalues $\{\lambda_k\}$. The GFT is defined as the transformation of a graph signal from the canonical "node" basis to its representation in the eigenvector basis:

[1] In the literature, the converse convention is sometimes chosen (e.g. in [1]), hence the $\mathbf{A}^\top$ occasionally appearing in this chapter.

[2] In the literature, this reference operator is often noted $\mathbf{S}$ for "shift". Nevertheless, the shift interpretation is essentially valid if one considers $\mathbf{S}$ to be the adjacency matrix (see discussion in Section 0.2.1). In a general setting, we prefer to denote by $\mathbf{R}$ the reference operator.

Definition 2 (Graph Fourier Transform)**.** For a given diagonalizable reference operator $\mathbf{R} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1}$ associated with a graph $\mathcal{G}$, the GFT of a graph signal $\mathbf{x} \in \mathbb{R}^N$ is:

$$\mathbb{F}_{\mathcal{G}}\,\mathbf{x} \doteq \hat{\mathbf{x}} \doteq \mathbf{U}^{-1}\mathbf{x}. \tag{0.2}$$

The GFT's coefficients are simply the projections on the left eigenvectors of $\mathbf{R}$:

$$\forall k = 1, \ldots, N \qquad (\mathbb{F}_{\mathcal{G}}\,\mathbf{x})_k = \hat{\mathbf{x}}_k = \mathbf{v}_k^T\mathbf{x}. \tag{0.3}$$

Moreover, the GFT is invertible: $\mathbf{U}\,\hat{\mathbf{x}} = \mathbf{U}\mathbf{U}^{-1}\mathbf{x} = \mathbf{x}$. While, in general, the complex Fourier modes $\mathbf{u}_k$ are not orthogonal to each other, when $\mathbf{R}$ is symmetric, the following additional properties hold true.

The special case of symmetric reference operators. If in addition to be real, $\mathbf{R}$ is also symmetric, then $\mathbf{U}$ and $\mathbf{\Lambda}$ are real matrices, and $\mathbf{U}$ may be found orthonormal, that is: $\mathbf{U}^{-1} = \mathbf{U}^\top$. In this case, $\mathbf{v}_k = \mathbf{u}_k$, the GFT of $\mathbf{x}$ is simply $\hat{\mathbf{x}} = \mathbf{U}^\top\mathbf{x}$ with coefficients $\hat{\mathbf{x}}_k = \mathbf{u}_k^\top\mathbf{x}$, and the Parseval relation holds: $||\hat{\mathbf{x}}||_2 = ||\mathbf{x}||_2$. Hereafter, when a symmetric operator $\mathbf{R}$ is encountered, one should have these properties in mind.

Finally, the interpretation of the graph Fourier modes $\mathbf{u}_k$ in terms of oscillations and frequencies will be the scope of Section 0.1.3. In the following, we list possible choices of reference operators, all diagonalizable with different $\mathbf{U}$ and $\mathbf{\Lambda}$, thus all defining different possible GFTs.

GFT for undirected graphs

Undirected graphs are characterized by symmetric adjacency matrices: $\forall(i, j)\ \mathbf{A}_{ij} = \mathbf{A}_{ji}$. This does not necessarily mean that $\mathbf{R}$ has to be chosen symmetric as well, as we will see with the example $\mathbf{R} = \mathbf{L}_{\mathbf{rw}}$. The following choices of $\mathbf{R}$ are the most common in the undirected case.

The combinatorial Laplacian [symmetric]. The first choice for $\mathbf{R}$, advocated in [2, 3, 4] is to use the graph's combinatorial Laplacian, having properties studied in [5]. It is defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$ where $\mathbf{D}$ is the diagonal matrix of nodes' strengths, defined as $\mathbf{D}_{ii} = \mathbf{d}_i = \sum_j \mathbf{A}_{ij}$. If the adjacency matrix is binary (i.e., unweighted), this strength reduces to the degree of each node. The advantages of using $\mathbf{L}$ are twofold. *i)* It is an intuitive manner to define the GFT: $\mathbf{L}$ being the discretized version of the continuous Laplacian operator which admits the Fourier modes as eigenmodes, it is fair to use by analogy the eigenvectors of $\mathbf{L}$ as graph Fourier modes. Moreover, this choice is associated with a complete theory of vector calculus (e.g., gradients) for graph signals [6] that is useful to solve partial differential equations on graphs. *ii)* $\mathbf{L}$ has well known mathematical properties [5], giving ways to characterize the graph or functions and processes on the graph (see also [7]). Most prominently, it is semi-definite positive (SDP) and its eigenvalues, being all positive or null, will serve in the following to bind eigenvectors with a notion of frequency.

The normalized Laplacian [symmetric]. A second choice for $\mathbf{R}$ is the normalized Laplacian $\mathbf{L_n} = \mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$. An interesting property of this choice of Laplacian is that all its eigenvalues lie between 0 and 2 [5].

The adjacency matrix, or deformed Laplacian [symmetric]. Another choice for $\mathbf{R}$ is the adjacency matrix[3] $\mathbf{A}^\top$, as advocated in [1]. One readily sees that the eigenbasis $\mathbf{U}$ of $\mathbf{A}^\top$ and the eigenbasis of the deformed Laplacian $\mathbf{L_d} = \mathbf{I} - \frac{\mathbf{A}^\top}{\|\mathbf{A}\|_2}$, where $\|.\|_2$ is the operator 2-norm, are the same. Therefore, the corresponding GFTs are equivalent and, for consistency in the presentation, we will use $\mathbf{R} = \mathbf{L_d}$ here.

The random walk Laplacian [not symmetric]. The random walk Laplacian is yet another Laplacian reading: $\mathbf{L_{rw}} = \mathbf{D}^{-1}\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{A}$, where $\mathbf{D}^{-1}\mathbf{A}$ serves also to describe a uniform random walk on the graph. Even though $\mathbf{L_{rw}}$ is not symmetric, we know it is diagonalizable in $\mathbb{R}$. In fact, if $\mathbf{u}_k$ is an eigenvector of $\mathbf{L_n}$ with eigenvalue λ_k, then $\mathbf{D}^{-\frac{1}{2}}\mathbf{u}_k$ is an eigenvector of $\mathbf{L_{rw}}$ with same eigenvalue. Thus, $\mathbf{L_{rw}}$ has the same eigenvalues as $\mathbf{L_n}$, and its Fourier basis $\mathbf{U}$ is real but not orthonormal.

Other possible definitions of the reference operator. For instance, the consensus operator (of the form $\mathbf{I} - \sigma\mathbf{L}$ with some suitable σ) [8], a geometric Laplacian [9], or some other deformed Laplacian one may think of, are valid alternatives.

All these operators imply a different spectral domain (different $\mathbf{U}$ and $\mathbf{\Lambda}$) and, provided one has a nice frequency interpretation (which is the object of Section 0.1.3), they all define possible GFTs. In the graph signal processing literature, the first three operators ($\mathbf{L}$, $\mathbf{L_n}$ and $\mathbf{L_d}$) are the most widely used.

GFT for directed graphs

For directed graphs, the adjacency matrix is no longer symmetric – $\mathbf{A}_{ij}$ is not necessarily equal to $\mathbf{A}_{ji}$ – which does not automatically imply that the reference operator $\mathbf{R}$ is not symmetric (e.g., the case $\mathbf{R} = \mathbf{Q}$ below). This case is of great interest in some applications where the graph is naturally directed such as hyperlink graphs (there is a directed edge between website i and website j if there is a hyperlink in website i directing to j). In directed graphs, the degree of node i is separated in its out-degree, $d_{\text{out}} = \sum_j A_{ij}$, and its in-degree, $d_{\text{in}} = \sum_j A_{ji}$.

Some straightforward approaches [not symmetric]. It is possible to readily transpose the previous notions to the directed case, choosing either $\mathbf{D_{out}}$ or $\mathbf{D_{in}}$ to replace $\mathbf{D}$ in the different formulations: e.g. $\mathbf{L} = \mathbf{D_{in}} - \mathbf{A}^\top$ as in [10], $\mathbf{L_{rw}} = \mathbf{I} - \mathbf{D_{out}}^{-1}\mathbf{A}$ as in [11], $\mathbf{L_n} = \mathbf{I} - \mathbf{D_{out}}^{-\frac{1}{2}}\mathbf{A}\mathbf{D_{out}}^{-\frac{1}{2}}$. A notable choice is to directly use $\mathbf{L_d} = \mathbf{I} - \frac{\mathbf{A}^\top}{\|\mathbf{A}\|_2}$

[3] We use $\mathbf{A}^\top$ instead of $\mathbf{A}$ for consistency purposes with [1], whose convention for directed edges in the adjacency matrix is converse to ours: what they call $\mathbf{A}$ is what we call $\mathbf{A}^\top$. Without any influence in the undirected case, it has an impact in the directed case.

as in [1] (we recall that $\mathbf{L_d}$ and $\mathbf{R} = \mathbf{A}^\top$ are equivalent for they share the same eigenvectors and thus, define the same GFT). These matrices are no longer strictly speaking Laplacians as they are no longer SDP, but one may nonetheless consider them as reference operators defining possible GFTs. Note that these definitions entail to choose (rather arbitrarily) either $\mathbf{D_{out}}$ or $\mathbf{D_{in}}$ in their formulations, with the notable exception of $\mathbf{L_d}$. In fact, $\mathbf{L_d}$ naturally generalizes to the directed case and is a classical choice of $\mathbf{R}$ in this context [1].

Chung's directed Laplacian [symmetric]. A less common approach in the graph signal processing community is the one provided by the directed Laplacians introduced by Chung [12]. To define these Laplacians, let us first introduce the random walk transition matrix (or operator) defined as $\mathbf{P} = \mathbf{D_{out}^{-1}A}$. It admits a stationary probability[4] $\boldsymbol{\pi} \in \mathbb{R}_+^N$ such that $\boldsymbol{\pi}^\top \mathbf{P} = \boldsymbol{\pi}^\top$. Writing $\mathbf{\Pi} = \mathrm{diag}(\boldsymbol{\pi})$, Chung defines the following two directed Laplacians:

$$\mathbf{Q} = \mathbf{\Pi} - \frac{\mathbf{\Pi P} + \mathbf{P}^\top \mathbf{\Pi}}{2}, \tag{0.4}$$

$$\mathbf{Q_n} = \mathbf{\Pi}^{-\frac{1}{2}} \mathbf{Q} \mathbf{\Pi}^{-\frac{1}{2}} = \mathbf{I} - \frac{\mathbf{\Pi}^{\frac{1}{2}} \mathbf{P} \mathbf{\Pi}^{-\frac{1}{2}} + \mathbf{\Pi}^{-\frac{1}{2}} \mathbf{P}^\top \mathbf{\Pi}^{\frac{1}{2}}}{2}. \tag{0.5}$$

Both the combinatorial $\mathbf{Q}$ and the normalized $\mathbf{Q_n}$ directed Laplacians verify the properties of Laplacian matrices: SDP, negative (or null) entries everywhere except on the diagonal and real symmetric. It is easy to see that (0.4) and (0.5) generalize the definitions of the undirected case since for an undirected graph, $\mathbf{\Pi} = \mathbf{D}$, $\mathbf{\Pi P} = \mathbf{A}$; hence $\mathbf{Q}$ is the combinatorial Laplacian $\mathbf{L}$, and $\mathbf{Q_n}$ is its normalized version $\mathbf{L_n}$.

Other possible definitions of the reference operator. The previous definitions of $\mathbf{L_{rw}}$ and $\mathbf{L_d}$ for undirected graphs may also be generalized to the directed Laplacian framework to obtain:

$$\mathbf{Q_{rw}} = \mathbf{I} - \frac{\mathbf{P} + \mathbf{\Pi}^{-1} \mathbf{P}^\top \mathbf{\Pi}}{2} \quad \text{and} \quad \mathbf{Q_d} = \mathbf{I} - \frac{\mathbf{\Pi P} + \mathbf{P}^\top \mathbf{\Pi}}{\|\mathbf{\Pi P} + \mathbf{P}^\top \mathbf{\Pi}\|_2}. \tag{0.6}$$

Additional notes. Other GFTs for directed graphs were proposed via the Hermitian Laplacian as introduced in [13], which generalizes $\mathbf{A}^\top$. A very different approach is to construct a Graph Fourier basis directly from an optimization scheme, requiring some notion of smoothness, or generalization of it – see [14, 15]. We will not consider this recent approach here.

0.1.3 FREQUENCIES OF GRAPH SIGNALS

To complement the notion of GFT, one needs to introduce some frequency analysis of the Fourier modes on the graph. The general way of doing so is to compute how

[4] Assuming the random walk is ergodic, i.e. irreducible and non-periodic.

fast a mode oscillates on the graph, and the tool of preference is to compute their variations across all edges of the graph. Let us first note the following facts:

- **In the undirected case**, $\mathbf{L}$ is semi-definite positive (SDP). In fact, one may write:

$$V_{\mathbf{L}}(\mathbf{x}) = \mathbf{x}^\top \mathbf{L}\mathbf{x} = \frac{1}{2} \sum_{(i,j)\in\mathcal{E}} \mathbf{A}_{ij}(x_i - x_j)^2 \geq 0. \tag{0.7}$$

  This function is also called the Dirichlet form. Similarly, $\mathbf{L_n}$ is also SDP:

$$V_{\mathbf{L_n}}(\mathbf{x}) = \mathbf{x}^\top \mathbf{L_n}\mathbf{x} = \frac{1}{2} \sum_{(i,j)\in\mathcal{E}} \mathbf{A}_{ij}\left(\frac{x_i}{\sqrt{d_i}} - \frac{x_j}{\sqrt{d_j}}\right)^2 \geq 0. \tag{0.8}$$

  As far as we know, the Dirichlet forms of $\mathbf{L_{rw}}$ and $\mathbf{L_d}$ do not have such a nice formulation as a sum of local quadratic variations over all edges of the graph. They are nevertheless SDP because: *i)* $\mathbf{L_{rw}}$ and $\mathbf{L_n}$ have the same spectrum; *ii)* the symmetry of $\mathbf{L_d}$ implies real eigenvalues, and the maximum eigenvalue of $\mathbf{A}/\|\mathbf{A}\|_2$ being 1 by definition of the norm, the minimum eigenvalue of $\mathbf{L_d}$ is 0.
- **In the directed case**, all directed Laplacians $\mathbf{Q}$, $\mathbf{Q_n}$, $\mathbf{Q_{rw}}$ and $\mathbf{Q_d}$ are SDP due to similar arguments. $\mathbf{Q}$ and $\mathbf{Q_n}$ also have Dirichlet forms in terms of a sum of local quadratic variations, e.g.:

$$V_{\mathbf{Q}}(\mathbf{x}) = \mathbf{x}^\top \mathbf{Q}\mathbf{x} = \frac{1}{2} \sum_{(i,j)\in\mathcal{E}} \pi_i \mathbf{P}_{ij}(x_i - x_j)^2 \geq 0. \tag{0.9}$$

  The other reference operators $\mathbf{L} = \mathbf{D_{in}} - \mathbf{A}^\top$, $\mathbf{L_{rw}} = \mathbf{I} - \mathbf{D_{out}}^{-1}\mathbf{A}$, $\mathbf{L_n} = \mathbf{I} - \mathbf{D_{out}}^{-\frac{1}{2}}\mathbf{A}\mathbf{D_{out}}^{-\frac{1}{2}}$, $\mathbf{L_d} = \mathbf{I} - \frac{\mathbf{A}^\top}{\|\mathbf{A}\|_2}$ are not SDP as their eigenvalues may be complex. Nevertheless, the real part of their eigenvalues is always non-negative. This is quite clear for $\mathbf{L}_d$. For $\mathbf{L} = \mathbf{D_{in}} - \mathbf{A}^\top$, as the sum of row i of $\mathbf{A}^\top$ is equal to $\mathbf{d_{in}}(i)$, Gershgorin circle theorem ensures that all eigenvalues of $\mathbf{L}$ are non-negative. For $\mathbf{L_{rw}}$: as $\mathbf{P} = \mathbf{D_{out}}^{-1}\mathbf{A}$ is a stochastic matrix, the Perron-Frobenius theorem ensures that its eigenvalues are in the disk of radius 1 in the complex plane, hence the real part of $\mathbf{L_{rw}}$'s eigenvalues are non-negative. As $\mathbf{L_{rw}}$ and $\mathbf{L_n}$ have the same set of eigenvalues, it is also true for $\mathbf{L_n}$.

To sum up, all the reference operators considered are either SDP (with real non-negative eigenvalues) or have eigenvalues whose real component is non-negative.

Definition 3 (Graph frequency)**.** Let $\mathbf{R}$ be a reference operator. If its eigenvalues are real, the generalized graph frequency ν of a graph Fourier mode $\mathbf{u}_k$ is:

$$\nu(\mathbf{u}_k) = \lambda_k \geq 0. \tag{0.10}$$

If its eigenvalues are complex, two different definitions of the generalized graph frequency ν of a graph Fourier mode $\mathbf{u}_k$ exist:

$$\nu(\mathbf{u}_k) = \operatorname{Re}(\lambda_k) \geq 0 \quad \text{or} \quad \nu(\mathbf{u}_k) = |\lambda_k| \geq 0. \tag{0.11}$$

Remarks. In the case of complex eigenvalues, it is a matter of choice whether we consider the imaginary part of the eigenvalues or not. There is no current consensus on this question. A second remark deals with the case of a multiple eigenvalue λ_k, i.e., if there are several eigenvectors associated to the same λ_k; then, only one frequency $\nu(\lambda_k)$ is defined for the associated eigenspace.

Justification: the link between frequency and variation. Two types of variation measures have been considered in the literature to show the consistency between this definition of graph frequencies and a notion of oscillation over the graph. The first one is based on the quadratic forms of the Laplacian operators. For instance, in the undirected case with the combinatorial Laplacian, Eq. (0.7) applied to any normalized Fourier mode $\mathbf{u}_k$ defined from $\mathbf{L}$ reads:

$$V_{\mathbf{L}}(\mathbf{u}_k) = \mathbf{u}_k^\top \mathbf{L}\mathbf{u}_k = \frac{1}{2}\sum_{(i,j)\in\mathcal{E}} \mathbf{A}_{ij}\,(u_k(i) - u_k(j))^2 = \lambda_k\,\|\mathbf{u}_k\|_2^2 = \lambda_k. \tag{0.12}$$

The larger the local quadratic variations of $\mathbf{u}_k$, the larger its frequency λ_k. Eqs. (0.8) and (0.9) (as well as its counterpart for $\mathbf{Q_n}$) enable to make this variation-frequency link for $\mathbf{L_n}$, $\mathbf{Q}$ and $\mathbf{Q_n}$. The second general type of variation that has been defined [16] is the total variation between a signal and its shifted version on the graph (where "shifting" a signal is understood as applying the adjacency matrix to it). For instance, in the case of $\mathbf{L_d}$, the associated variation reads[5]:

$$V_{\mathbf{L_d}}(\mathbf{x}) = \left\|\mathbf{x} - \frac{1}{|\mu_{max}|}\mathbf{A}^\top\mathbf{x}\right\|_2 = \|\mathbf{L_d}\mathbf{x}\|_2\,, \tag{0.13}$$

where μ_i designate the eigenvalues of $\mathbf{A}$ and μ_{max} the one of maximum magnitude. The variation of the graph Fourier mode $\mathbf{u}_k$ from $\mathbf{L_d}$ thus reads:

$$V_{\mathbf{L_d}}(\mathbf{u}_k) = \|\mathbf{L_d}\mathbf{u}_k\|_2 = |\lambda_k|\,\|\mathbf{u}_k\|_2 = |\lambda_k|. \tag{0.14}$$

The larger the total variation of $\mathbf{u}_k$, that is, the further is $\mathbf{u}_k$ from its shifted version along the graph, the larger its frequency[6]. For $\mathbf{L} = \mathbf{D_{in}} - \mathbf{A}^\top$, a similar approach detailed in [10] links the variation of $\mathbf{u}_k$ to its frequency $|\lambda_k|$.

It may also happen that for some operator $\mathbf{R}$, none of these two types of variations (quadratic forms or total variation) show natural. Then, one may use the variation $V_{\mathbf{R}'}$ based on another related operator $\mathbf{R}'$ to define frequencies. For instance, in [11, 17], the authors considered the random walk Laplacians $\mathbf{R} = \mathbf{P} = \mathbf{D_{out}}^{-1}\mathbf{A}$ as the reference operator to define the GFT, while the directed combinatorial Laplacian $\mathbf{R}' = \mathbf{Q}$ is used to measure the variations. With these choices, they showed that $V_{\mathbf{Q}}(\mathbf{u}_k)$ is

[5] In [16], the ℓ_1 norm is used, but the ℓ_2 norm can be used equivalently: this is a matter of how one wants to normalize the eigenvectors. In this chapter, we consider the classical Euclidean norm, hence ℓ_2.

[6] There is a direct correspondence between λ_i, the eigenvalues of $\mathbf{L_d}$, and μ_i: $\lambda_i = 1 - \mu_i/|\mu_{max}|$. We thus recover the results in [16]: the closer is μ_k from $|\mu_{max}|$ in the complex plane, the smaller the total variation of the associated Fourier mode $\mathbf{u}_k$.

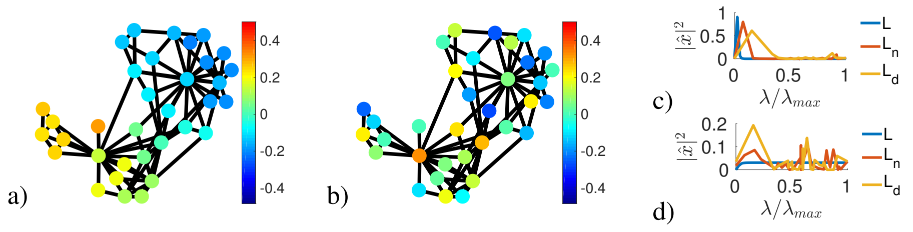

FIGURE 0.1 **Two graph signals and their GFTs.**

Plots a) and b) represent respectively, a low-frequency and a high-frequency graph signal on the binary Karate club graph [21]. Plots c) and d) are their corresponding GFTs computed for three reference operators: $\mathbf{L}$, $\mathbf{L_n}$ and $\mathbf{L_d}$ (equivalent to the GFT defined via the adjacency matrix).

equal to $\mathrm{Re}(\lambda_k)$ up to a normalization constant. Another example of such a case is in [18] where one of the reference operators to build filters is the isometric translation introduced in [19] while the variational operators are built upon the combinatorial Laplacian. Note finally that other notions of variations like the Hub authority score can be drawn from the literature. We refer the reader to [20] for details on that, as well as a complementary discussion on GFTs and their related variations.

Definition 3 associates graph frequencies only to graph Fourier modes. For an arbitrary signal, we have the following definition of its frequency analysis:

Definition 4 (Frequency analysis)**.** The frequency analysis of any graph-signal $\mathbf{x}$ on $\mathcal{G}$ is given by its components $(\mathbb{F}_{\mathcal{G}}\mathbf{x})_k$ at frequency $\nu(\mathbf{u}_k)$, as given in Definition 3.

0.1.4 IMPLEMENTATION AND ILLUSTRATION

Implementation. The implementation of the GFT requires to diagonalize $\mathbf{R}$, costing $O(N^3)$ operations in general, and $O(N^2)$ memory space to store $\mathbf{U}$. Then, applying $\mathbf{U}$ to a signal $\mathbf{x}$ to obtain $\hat{\mathbf{x}} = \mathbf{U}^{-1}\mathbf{x}$ costs $O(N^2)$ operations. These costs are prohibitive for large graphs ($N \gtrsim 10^4$ nodes). Recent works investigate how to reduce these costs, by tolerating an approximation on $\hat{\mathbf{x}}$. In the cases where $\mathbf{R}$ is symmetric, the authors in [22, 23] suggest to approximate $\mathbf{U}$ by a product of $O(N \log N)$ Givens rotations [**?**], using a truncated Jacobi algorithm. The resulting approximated fast GFT requires $O(N^2 \log^2 N)$ operations to compute the Givens rotations, and $O(N \log N)$ operations to compute the approximate GFT of $\mathbf{x}$. The difficulty in designing fast GFTs boils down to the difficulty of deciphering eigenvalues that are very close to one another. This difficulty disappears once we consider smooth filtering operations that are much easier to efficiently approximate, as we will see in the next section.

Illustrations. To illustrate the GFT and the notion of frequency, we show in Fig. 0.1

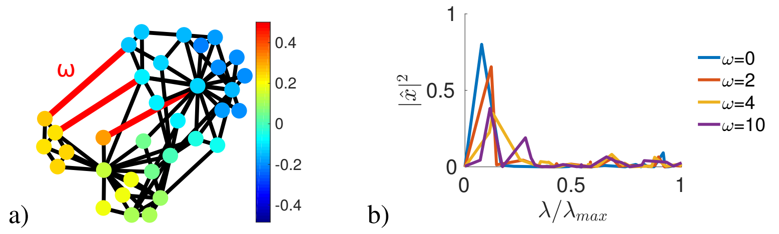

FIGURE 0.2 **Sensitivity of the GFT to the graph's topology.**

Plot a) represents the same low-frequency graph signal as in Fig. 0.1-a), but the underlying graph structure is altered by adding three edges with same weight ω (in red). Plot b) represents the variation of its GFT (here choosing $\mathbf{R} = \mathbf{L_n}$) with respect to the edges' weight ω.

two graph signals on the Karate club graph [21], corresponding to instances of a low frequency and a high frequency signal, respectively. We also show their GFTs, computed for three different choices of $\mathbf{R}$: $\mathbf{L}$, $\mathbf{L_n}$ and $\mathbf{L_d}$. We see that the choice of normalization and the choice to take explicitly the degree matrix into account or not in the definition of $\mathbf{R}$ has a quantitative impact on the GFTs. Nevertheless, qualitatively, a graph signal that varies slowly (resp. rapidly) along any path of the graph is low-frequency (resp. high-frequency). In Fig. 0.2, we show how the GFT is not only sensitive to the graph signal but also to the underlying graph structure. In fact, we observe that for a given graph signal, modifications in the graph structure (here adding three links of weight ω) induce modifications in the signal's graph Fourier transform.

0.2 GRAPH FILTERS

In this section, we assume that the reference operator $\mathbf{R}$ of the graph on which we wish to design filters is diagonalizable in $\mathbb{C}$ as in the previous section. The order of the eigenvalues and eigenvectors is chosen frequency-increasing. That is, given a choice of frequency definition (either $\nu(\lambda_k) = \mathrm{Re}(\lambda_k) \in \mathbb{R}^+$ or $\nu(\lambda_k) = |\lambda_k| \in \mathbb{R}^+$), one has $\nu(\mathbf{u}_1) \leq \nu(\mathbf{u}_2) \leq \ldots \leq \nu(\mathbf{u}_N)$.

0.2.1 DEFINITION OF GRAPH FILTERS

The reference operator $\mathbf{R}$ has an eigendecomposition as in Eq. (0.1), and it can also be written as a sum of projectors on all its eigenspaces:

$$\mathbf{R} = \sum_\lambda \lambda\, \mathbf{Pr}_\lambda, \tag{0.15}$$

where the sum is on all different eigenvalues λ and $\mathbf{Pr}_\lambda$ is the projector on the eigenspace associated with eigenvalue λ, i.e.:

$$\mathbf{Pr}_\lambda = \sum_{\lambda_k=\lambda} \mathbf{u}_k \mathbf{v}_k^\top.$$

Definition 5 (wide-sense definition of a graph filter)**.** The most general definition of a graph filter is an operator that acts separately on all the eigenspaces of $\mathbf{R}$, depending on their eigenvalue λ. Mathematically, any function

$$h : \mathbb{C} \to \mathbb{R} \tag{0.16}$$

$$\lambda \to h(\lambda). \tag{0.17}$$

defines a graph filter $\mathbf{H}$ such that

$$\mathbf{H} = \sum_\lambda h(\lambda)\, \mathbf{Pr}_\lambda = \sum_k h(\lambda_k) \mathbf{u}_k \mathbf{v}_k^\top. \tag{0.18}$$

To each eigenspace of $\mathbf{R}$ with eigenvalue λ is associated a filtering weight $h(\lambda)$ that attenuates or increases the importance of this eigenspace in the decomposition of the signal of interest. In fact, one may write the action of $\mathbf{H}$ on a signal $\mathbf{x}$ as:

$$\mathbf{Hx} = \sum_\lambda h(\lambda)\, \mathbf{Pr}_\lambda\, \mathbf{x}. \tag{0.19}$$

For any function g, let us write $g(\mathbf{\Lambda})$ as a shorthand notation for $\mathrm{diag}(g(\lambda_1)\ldots, g(\lambda_N))$. A graph filter can be written as:

$$\mathbf{H} = \mathbf{U}\, h(\mathbf{\Lambda})\, \mathbf{U}^{-1}. \tag{0.20}$$

Using functional calculus of operators, this is equivalently written as $\mathbf{H} = h(\mathbf{R})$, which calls for some interpretation remarks. In fact, this expression opens the way to interpret what is the action of a graph filter in the vertex domain. Firstly, note that applying $\mathbf{R}$ to a graph signal is in fact a *local* computation on the graph: on each node, the resulting transformed signal is a weighted sum of the values of the original signal on its (direct) neighbours. Therefore, in the node space, a filter $\mathbf{H} = h(\mathbf{R})$ can be interpreted as an operator that weights the information on the signal transmitted through edges of the graph, the same way classical filters are built on the basic operation of time-shift. This fundamental analogy shift/reference operator was first made in [1] and explains why in the GSP literature, reference operators are often called "shift operators".

Now, to illustrate the notion of graph filtering in the spectral domain, consider the graph signal $\mathbf{x} = \sum_k \alpha_k \mathbf{u}_k$ and $\mathbf{y} = \mathbf{Hx}$. The k-th Fourier component of $\mathbf{x}$ being, by construction, $\hat{\mathbf{x}}_k = \mathbf{v}_k^\top \mathbf{x} = \alpha_k$, its filtered version reads:

$$\mathbf{y} = \mathbf{Hx} = \sum_k h(\lambda_k) \alpha_k \mathbf{u}_k. \tag{0.21}$$

Hence the k-th Fourier component of the filtered signal is $\hat{\mathbf{y}}_k = h(\lambda_k)\,\alpha_k$, recalling the classical interpretation of filtering as a multiplication in the Fourier domain. We call $h(\lambda)$ the frequency response of the filter.

In many cases, it is more convenient and natural to restrict ourselves to functions h that associate the same real number to all values of $\lambda \in \{\lambda \text{ s.t. } \nu(\lambda) = \nu \in \mathbb{R}^+\}$. That is, considering any two eigenvalues λ_1 and λ_2 associated with the same frequency $\nu(\lambda_1) = \nu(\lambda_2)$, we restrict ourselves to functions h such that $h(\lambda_1) = h(\lambda_2)$. This entails the following narrowed-sense definition of a graph filter.

Definition 6 (narrowed-sense definition of a graph filter)**.** Any function

$$h : \mathbb{R}^+ \to \mathbb{R} \tag{0.22}$$

$$\nu \to h(\nu). \tag{0.23}$$

defines a narrowed-sense graph filter $\mathbf{H}$ such that

$$\mathbf{H} = \sum_{\lambda} h(\nu(\lambda))\,\mathbf{Pr}_\lambda = \mathbf{U}h(\nu(\mathbf{\Lambda}))\mathbf{U}^{-1}. \tag{0.24}$$

Remark. If the eigenvalues are real, both definitions 5 and 6 are equivalent since $\nu(\lambda) = \lambda \in \mathbb{R}^+$.

Examples of narrowed-sense filters:

- The constant filter equal to c: $h(\nu) = c$. In this case, $h(\nu(\mathbf{\Lambda})) = c\mathbf{I}$ and $\mathbf{H} = c\mathbf{I}$: all frequencies are allowed to pass, and no component is filtered out.
- The Kronecker delta in ν^*: $h(\nu) = \delta_{\nu,\nu^*}$. If there exists one (or several) eigenvalues λ of $\mathbf{R}$ such that $\nu(\lambda) = \nu^*$, then $\mathbf{H} = \sum_{\lambda \text{ s.t. } \nu(\lambda)=\nu^*} \mathbf{Pr}_\lambda$. If not, then $\mathbf{H} = \mathbf{0}$. For this filter, only the frequency ν^* is allowed to pass.
- The ideal low-pass with cut-off frequency ν_c: $h(\nu) = 1$ if $\nu \leq \nu_c$, and 0 otherwise. In this case: $\mathbf{H} = \sum_{\lambda,\text{s.t.}\nu(\lambda)\leq\nu_c} \mathbf{Pr}_\lambda$, *i.e.*, only frequencies up to ν_c are allowed to pass.
- The heat kernel $h(\nu) = \exp^{-\nu/\nu_0}$: the weight associated with ν is exponentially decreasing with the frequency ν. Actually $\mathbf{y} = \mathbf{H}\,\mathbf{x}_0$ is the solution of the graph diffusion (or heat) equation (see [2]) at time $t = 1/\nu_0$ with initial condition $\mathbf{x}_0$.

0.2.2 PROPERTIES OF GRAPH FILTERS

From now on, in order to simplify notations and concepts in this introductory chapter on graph filtering, we will restrict ourselves to symmetric reference operators, such as $\mathbf{L}$, $\mathbf{L_n}$, or $\mathbf{L_d}$ in the undirected case, or the directed Laplacians $\mathbf{Q}$ or $\mathbf{Q_n}$ in the directed case. In this case, the eigenvalues are real such that the frequency definition is straightforward $\nu(\lambda) = \lambda$, both filter definitions are equivalent such that the frequency response reads

$$h : \mathbb{R}^+ \to \mathbb{R} \tag{0.25}$$

$$\lambda \to h(\lambda), \tag{0.26}$$

and one may find a real orthonormal graph Fourier basis $\mathbf{U}$ such that $\mathbf{U}^{-1} = \mathbf{U}^\top$. A filtering operator associated with h thereby reads:

$$\mathbf{H} = \mathbf{U}h(\mathbf{\Lambda})\mathbf{U}^\top \in \mathbb{R}^{N\times N}. \tag{0.27}$$

All results presented in the following may be (carefully) generalized to the unsymmetric case.

Definition 7. We write C_p the set of finite-order polynomials in $\mathbf{R}$:

$$C_p = \left\{\mathbf{H} \text{ s.t. } \mathbf{H} = \sum_{i=0}^{n} a_i\mathbf{R}^i,\ \{a_i\}_{i=0,\dots,n} \in \mathbb{R}^{n+1}, n \in \mathbb{N}\backslash\{+\infty\}\right\}. \tag{0.28}$$

Proposition 1. *C_p is equal to the set of graph filters.*

Proof. Consider $\mathbf{H} \in C_p$. Then, defining $h(\lambda) = \sum_{i=0}^n a_i\lambda^i$, one has: $\mathbf{H} = \sum_{i=0}^n a_i\mathbf{R}^i = \mathbf{U}\sum_{i=0}^n a_i\mathbf{\Lambda}^i\mathbf{U}^\top = \mathbf{U}h(\mathbf{\Lambda})\mathbf{U}^\top$, *i.e.*, $\mathbf{H}$ is a filter. Now, consider $\mathbf{H}$ a filter, *i.e.*, there exists h such that $\mathbf{H} = \mathbf{U}h(\mathbf{\Lambda})\mathbf{U}^\top$. Consider the polynomial $\sum_{i=0}^{N-1} a_i\lambda^i$ that interpolates through all pairs $(\lambda_i, h(\lambda_i))$. The maximum degree of such a polynomial is $N-1$ as there are maximum N points to interpolate, and may be smaller if eigenvalues have multiplicity larger than one. Thereby, one may write: $\mathbf{H} = \mathbf{U}h(\mathbf{\Lambda})\mathbf{U}^\top = \mathbf{U}\sum_{i=0}^{N-1} a_i\mathbf{\Lambda}^i\mathbf{U}^\top = \sum_{i=0}^{N-1} a_i\mathbf{R}^i$. Writing $n = N-1$, this means that $\mathbf{H} \in C_p$. □

Consequence: An equivalent definition of a graph filter is a polynomial in $\mathbf{R}$.

Definition 8. We write C_d the set of all diagonal operators in the graph Fourier space:

$$C_d = \{\mathbf{H},\ \text{s.t. } \mathbf{U}^\top\mathbf{H}\mathbf{U} \text{ is diagonal}\}. \tag{0.29}$$

Proposition 2. *The set of graph filters is included in C_d. Both sets are equal iff all eigenspaces of* $\mathbf{R}$ *are simple (i.e., all eigenvalues are of multiplicity one).*

Proof. By definition of graph filters, they are included in C_d. Now, in general, an element of C_d is not necessarily a graph filter. In fact, given $\mathbf{H} \in C_d$, all diagonal entries of $\mathbf{U}^\top\mathbf{H}\mathbf{U}$ may be chosen independently, which is not the case for the diagonal entries of $h(\mathbf{\Lambda})$ corresponding to the same eigenspace. Thus, both sets are equal iff all eigenspaces are of dimension one. □

Consequence: In the case where all eigenvalues are simple, an equivalent definition of a graph filter is a diagonal matrix in the graph Fourier basis.

Definition 9. We write C_c the set of matrices that commute with $\mathbf{R}$:

$$C_c = \{\mathbf{H} \text{ s.t. } \mathbf{RH} = \mathbf{HR}\}. \tag{0.30}$$

Proposition 3. *$C_p \subseteq C_c$. The equality holds iff all eigenvalues of* $\mathbf{R}$ *are simple.*

Proof. A polynomial in $\mathbf{R}$ necessarily commutes with $\mathbf{R}$, thus $C_p \subseteq C_c$. Now, in general, an element in C_c is not necessarily in C_p. In fact, if $\mathbf{H} \in C_c$, then $\mathbf{\Lambda Y} = \mathbf{Y\Lambda}$ with $\mathbf{Y} = \mathbf{U}^\top\mathbf{HU}$. Suppose $\mathbf{\Lambda} = \lambda\mathbf{I}$. In this case, commutativity does not constrain $\mathbf{Y}$ at all, thereby $\mathbf{H}$ is not necessarily in C_p. Now, if we suppose that all eigenvalues have multiplicity one, *i.e.*, all diagonal entries of $\mathbf{\Lambda}$ are different, the only solution for $\mathbf{Y}$ is to be a diagonal matrix, *i.e.*, $C_c = C_d$. We showed in Prop. 2 that the set of graph filters is equal to C_d iff all eigenvalues have multiplicity one, therefore: $C_c = C_p$ iff all eigenvalues have multiplicity one. □

Consequence: In the case where all eigenvalues are simple, an equivalent definition of a graph filter is a linear operator that commutes with $\mathbf{R}$.

0.2.3 SOME DESIGNS OF GRAPH FILTERS

From the previous sections, it should now be clear that the frequency response of a filter $\mathbf{H}$ only alters the frequencies $\nu(\lambda)$ corresponding to the discrete set of eigenvalues of $\mathbf{R}$ (see Eq. (0.18)). More generally though, the frequency response of a graph filter can be defined over a continuous range of λ's, leading to the notion of *universal filter* design, i.e. a filter whose frequency response $h(\lambda)$ is designed for all λ's and not only adapted to the specific eigenvalues of $\mathbf{R}$. On the contrary, a *graph-dependent filter* design depends specifically on these eigenvalues.

FIR filters. From the results of Section 0.2.2, a natural class of graph filters is given in the form of Finite Impulse Response (FIR) filters, as by eq. (0.28) with a polynomial of finite order, which realizes a weighted Moving Average (MA) filtering of a signal. Also, one can design any universal filter by fitting the desired response $h(\lambda)$ with a polynomial $\sum_{i=0}^{n} a_i\lambda^i$. The larger n is, the closer the filter can approximate the desired shape. If the approximation is done only using the $h(\lambda_k)$, the design is graph-dependent, else it is universal if fitting some function $h(\lambda)$.

Let us then go back to the interpretation of $\mathbf{R}$ as a graph shift operator in the node space, (as studied or recalled for instance in [1, 24, 29, 20]), and see how it operates for FIR filters. Applied to a graph signal, the terms $\mathbf{R}^i$ in eq. (0.28), act as a i-hops local computation on the graph: on each node, the resulting filtered signal is a weighted sum of the values of the original signal lying in its i-th neighbourhood, that is, nodes attainable with a path of length i along the graph. Then, like for classical signals, FIR filters only imply a finite neighbourhood of each nodes, and this will translate in Section 0.2.4 into distributed, fast implementations of these filters. Still, FIR filters are usually poor at approximating filters with sharp changes of desired frequency response, as illustrated for instance in Fig. 1 of [24].

ARMA filters. A more versatile approximation of $h(\lambda)$ can be obtained with a rational design [24, 25, 26]:

$$h(\lambda) = \frac{\sum_{i=0}^{q} b_i \lambda^i}{1 + \sum_{i=1}^{p} a_i \lambda^i} = \frac{p_q(\lambda)}{p_p(\lambda)}. \tag{0.31}$$

Such a rational filter is called an Auto-Regressive Moving Average filter of order (p, q) and is commonly noted ARMA(p, q). Again, it is known from classical DSP that an ARMA design, being a IIR (Infinite Impulse Response) filter, is more adaptable at approximating various shapes of filters, especially with sharp changes in the frequency response. The filtering relation $\mathbf{y} = \mathbf{H}\mathbf{x}$ for ARMA filters can be written in the node domain as: $(1 + \sum_{i=1}^{p} a_i \mathbf{R}^i)\mathbf{y} = (\sum_{i=0}^{q} b_i \mathbf{R}^i)\mathbf{x}$. This ARMA filter expression will lead to the distributed implementation, discussed later in 0.2.4. For instance, for an ARMA(1,0) (i.e., an AR(1)) one will have to use: $\mathbf{y} = -a_1 \mathbf{R}\mathbf{y} + b_0 \mathbf{x}$.

Example: A first design of low-pass graph filtering is given by the simplest least-square denoising problem, where the Dirichlet form $\mathbf{x}^T\mathbf{R}\mathbf{x}$ is used as Tikhonov regularization promoting smoothness on the graph. Using the (undirected) Laplacian and assuming one observes $\mathbf{y}$, the filter is given by;

$$\mathbf{x}_* = \arg\min_{\mathbf{x}} \|\mathbf{x} - \mathbf{y}\|_2^2 + \gamma \mathbf{x}^\top \mathbf{L}\mathbf{x}. \tag{0.32}$$

The solution is then given in the spectral domain (for $\mathbf{L}$) by $(\mathbb{F}\mathbf{x}_*)_k = h_{AR(1)}(\lambda_k)(\mathbb{F}\mathbf{y})_k$ with $h_{AR(1)}(\lambda) = 1/(1 + \gamma\lambda)$. It turns out to be a (universal) AR(1) filter.

Design of coefficients. To design the coefficients of ARMA filters, the classical approach is to find the set of coefficients a_i and b_i to approximate the desired $h(\lambda)$ as a rational function. However, as recalled in [24, 25], the usual design in DSP are not easily transposed to the GSP framework because the frequency response is given in terms of the λ's, and not in terms of $j\omega$ or $e^{j\omega}$.

Henceforth, it has been studied in [24, 25] how to approximate the filter coefficients in a universal manner (i.e., with no specific reference to the graph spectrum) using a Shank's method: 1) Determine the a_i by first finding a polynomial approximation $P_h(\lambda)$ of $h(\lambda)$, and solve the system of equations $p_p(\lambda)P_h(\lambda) = p_q(\lambda)$ to identify the a_i's. Then 2) solve the least-square problem to minimize $\int_\lambda |p_q(\lambda)/p_p(\lambda) - h(\lambda)|^2 d\lambda$ w.r.t. λ to find the b_i's.

A second method is to approximate the filter response in a graph-dependent design, on the specific frequencies λ_k only. To do so, the method in [27], instead of fitting the polynomial ratio, solves the following optimization problem:

$$\min_{\mathbf{a},\mathbf{b}} \sum_{k=0}^{N-1} \left| h(\lambda_k)\left(1 + \sum_{i=1}^{p} a_i \lambda_k^i\right) - \sum_{i=0}^{q} b_i \lambda_k^i \right|^2 . \tag{0.33}$$

Using again a polynomial approximation $P_h(\lambda)$, the solution derives from the least square solution (see details in [27]).

AR filters to model random processes. We do not discuss much in this chapter random processes on graphs ; for that, see the framework to study stationary random processes on graphs in [28, 29], and Chapter II.5. Still, a remark can be done for the parametric modeling of random processes. As introduced in [1], one can model a process on a graph as the output of a graph filter, generally taken as an ARMA filter. Here, we discuss the case of AR filters. The linear prediction is written as:

$$\tilde{\mathbf{x}} = \sum_{i=1}^{p} a_i \mathbf{R}^i \mathbf{x}. \tag{0.34}$$

The coefficients of this filter can directly be obtained by numerical inversion of $\mathbf{x} = \mathbf{B}\mathbf{a}$ where $\mathbf{B} = (\mathbf{R}\mathbf{x}, \mathbf{R}^2\mathbf{x}, \dots, \mathbf{R}^p\mathbf{x})$. The solution is given by the pseudo-inverse: $\mathbf{a}^{\#} = (\mathbf{B}^\top\mathbf{B})^{-1}\mathbf{B}^\top\mathbf{x}$ which minimizes the squared error $\|\mathbf{x} - \mathbf{B}\mathbf{a}\|_2^2$, w.r.t to $\mathbf{a}$. Another possibility would be to estimate the coefficients of the AR model by means of the orthogonality principle leading to the Yule-Walker (YW) equations, as follows:

$$\mathbb{E}\left\{(\mathbf{R}^k\mathbf{x})^\top \mathbf{R}\mathbf{x}\right\} = \sum_{i=1}^{p} a_i \mathbb{E}\left\{(\mathbf{R}^i\mathbf{x})^\top \mathbf{R}^k\mathbf{x}\right\} = 0, \tag{0.35}$$

Depending on the structure of the reference operator $\mathbf{R}$, we have:

1. If $\mathbf{R}$ is symmetric ($\mathbf{R}^\top = \mathbf{R}$, as often for undirected graph), the autocorrelation function involved in this YW system is: $\rho_{\mathbf{x}}(m,n) = \mathbb{E}\{(\mathbf{R}^m\mathbf{x})^\top \mathbf{R}^n\mathbf{x}\} = \gamma_{\mathbf{x}}(m+n)$;
2. When $\mathbf{R}$ is unitary (for instance with $\mathbf{R}$ as the isometric operator from [19]), the corresponding autocorrelation will be $\rho_{\mathbf{x}}(m,n) = \mathbb{E}\{(\mathbf{R}^m\mathbf{x})^\top \mathbf{R}^n\mathbf{x}\} = \gamma_{\mathbf{x}}(n-m)$ and the usual techniques to solve the YW system can be used.

Experimentally, it was found that the isometric operator of [19] or a consensus operator are the one that offer a more exact and stable modeling [18].

0.2.4 IMPLEMENTATIONS OF GRAPH FILTERS

Given a frequency response h, how to efficiently filter a graph signal $\mathbf{x}$?

The direct approach. It consists in first diagonalizing $\mathbf{R}$ to obtain $\mathbf{U}$ and $\mathbf{\Lambda}$; then computing the filter matrix $\mathbf{H} = \mathbf{U}h(\mathbf{\Lambda})\mathbf{U}^\top$; and finally left-multiplying the graph signal $\mathbf{x}$ by $\mathbf{H}$ to obtain its filtered version. The overall computational cost of this procedure is $O(N^3)$ arithmetic operations due to the diagonalization and $O(N^2)$ memory space as the graph Fourier transform $\mathbf{U}$ is dense in general.

The polynomial approximate filtering approach. More efficiently though, we can first quickly estimate $\lambda_{\min}$ and $\lambda_{\max}$ (for instance via the power method) and second, look for a polynomial that best approximates $h(\lambda)$ on the whole interval $[\lambda_{\min}, \lambda_{\max}]$.

Let us call $\tilde{h}_i$ the coefficients of this approximate polynomial. We have:

$$\mathbf{Hx} = \mathbf{U}h(\mathbf{\Lambda})\mathbf{U}^\top\mathbf{x} \simeq \mathbf{U}\sum_{i=0}^{p}\tilde{h}_i\mathbf{\Lambda}^i\mathbf{U}^\top\mathbf{x} = \sum_{i=0}^{p}\tilde{h}_i\mathbf{R}^i\mathbf{x}. \tag{0.36}$$

The number of required arithmetic operations is $O(p|\mathcal{E}|)$, where p is a trade-off between precision and computational cost. Also, the easier is h approachable by a low-order polynomial, the better the approximation. At the same time, the larger p is, the more accurate the approximation is. The authors of [30] recommend Chebychev polynomials as an approximation basis as they are known to be optimal in the ∞-norm sense. In some circumstances, other choices can be preferred. For instance, when approximating the ideal low-pass, Chebychev polynomials yield Gibbs oscillations around the cut-off frequency that turn penalizing for smooth filters. In that case, other choices are possible [31], such as the Jackson-Chebychev polynomials that attenuate such unwanted oscillations.

The Lanczos approximate filtering approach. In [32], and based on works by [33], the authors propose an approximate filtering approach based on Lanczos iterations. Given a signal $\mathbf{x}$, the Lanczos algorithm computes an orthonormal basis $\mathbf{V}_p \in \mathbb{R}^{N\times p}$ of the Krylov subspace associated with $\mathbf{x}$: $K_p(\mathbf{L},\mathbf{x}) = \text{span}(\mathbf{x}, \mathbf{Lx}, \ldots, \mathbf{L}^{p-1}\mathbf{x})$, as well as a small tridiagonal matrix $\mathbf{H}_p \in \mathbb{R}^{p\times p}$ such that: $\mathbf{V}_p^*\mathbf{L}\mathbf{V}_p = \mathbf{H}_p$. The approximate filtering then reads:

$$\mathbf{Hx} \simeq \|\mathbf{x}\|_2\mathbf{V}_p h(\mathbf{H}_p)\boldsymbol{\delta}_1. \tag{0.37}$$

At fixed p, this approach has a typical complexity in $O(p|\mathcal{E}|)$, possibly raised to $O(p|\mathcal{E}| + Np^2)$ if a reorthonomalisation is needed to stabilize the algorithm; a cost that is comparable to that of the polynomial approximation approach. Theoretically, the quality of approximation is similar (see [32] for details). In practice however, it has been observed that if the spectrum is regularly spaced, polynomial approximations should be preferred, while the Lanczos method has an edge over others in the case of irregularly-spaced spectra. This is understandable as Krylov subspaces are also used for diagonalization purposes (see for instance, chapter 6 of [34]) and thus naturally adapt to the underlying spectrum.

Distributed implementation of ARMA filters. The ARMA filters being defined through a rational fraction, are IIR filters. Henceforth, the polynomials approach to distribute and fasten the computation are not the most efficient ones. The methods developed in [25, 26] yield a distributed implementation of ARMA filters. The first point is to remember that the rational filter of eq. (0.31) can be implemented from its partial fraction decomposition as a sum of polynomial fractions of order 1 only. Then, the distributed implementation of filtering $\mathbf{x}$ can be done by studying the 1st order recursion [25]:

$$\mathbf{y}(t+1) = c\mathbf{My}(t) + d\mathbf{x} \tag{0.38}$$

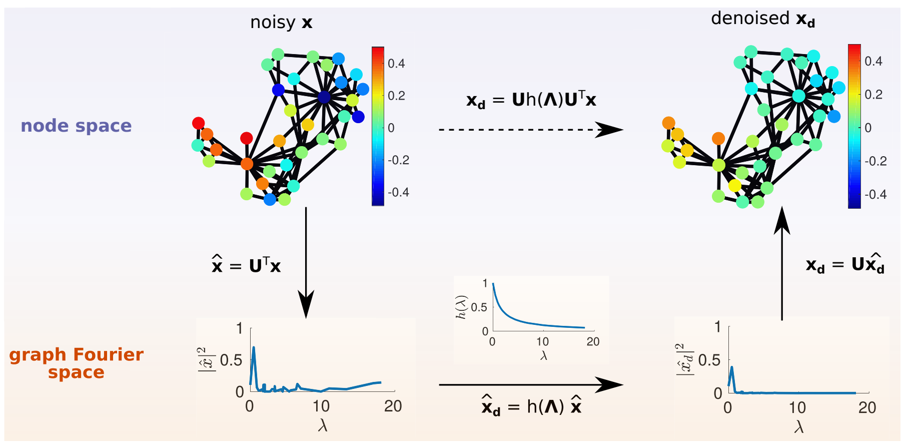
noisy x
denoised $\mathbf{x_d}$
node space
graph Fourier space
$\mathbf{x_d} = \mathbf{U}h(\mathbf{\Lambda})\mathbf{U}^T\mathbf{x}$
$\hat{\mathbf{x}} = \mathbf{U}^T\mathbf{x}$
$\mathbf{x_d} = \mathbf{U}\hat{\mathbf{x}}_\mathbf{d}$
$\hat{\mathbf{x}}_\mathbf{d} = h(\mathbf{\Lambda})\,\hat{\mathbf{x}}$
$h(\lambda)$
$|\hat{x}|^2$
$|\hat{x}_d|^2$
λ

FIGURE 0.3 **Illustration of graph filters: a denoising toy experiment**

The input signal $\mathbf{x}$ is a noisy version (additive Gaussian noise) of the low-frequency graph signal displayed in Fig. 0.1. We show here the filtering operation in the graph Fourier domain associated with $\mathbf{R} = \mathbf{L_n}$.

where $\mathbf{y}(t+1)$ is the filter output at iteration t. The operator $\mathbf{M}$ is chosen equal to $(\lambda_{\max} - \lambda_{\min})\mathbf{I} - \mathbf{R}$, with the same eigenvectors as $\mathbf{R}$, and with a minimal spectral radius that ensures good convergence properties for the recursion. The coefficients c and d are chosen in $\mathbb{C}$ so that, with $r = -d/c$ and $\rho = 1/c$, the proposed recursion reproduces the effect of the following ARMA(1,0) filter: $h(\lambda) = r/(\lambda - \rho)$. The coefficients r and ρ are the residue and the pole of the rational function, respectively.

Because $\mathbf{M}$ is local in the graph, the recursive application of eq. (0.38) is local and the algorithm is then naturally distributed on the graph, with a memory and operation at each recursion in $O(K|\mathcal{E}|)$ for K filters in parallel to compute the output of a ARMA(K, K) filter. This approach is shown in [25, 26] to converge efficiently. Moreover, in [24], the behavior of this design is also studied in time-varying settings, when the graph and the signal are possibly time-varying; it is then shown that the recursion can remain stable and usable as a distributed implementation of IIR filters.

Illustration. We show in Fig. 0.3 an example of a filtering operation on a graph signal. We consider here the case of a Tikhonov denoising (see Eq. (0.32)), i.e., with a frequency response equal to $h(\lambda) = 1/(1 + \gamma\lambda)$.

0.3 FILTERBANKS AND MULTISCALE TRANSFORMS ON GRAPHS

To process and filter signals on graphs, it is attractive to develop equivalent of multiscale transforms such as wavelets, i.e. ways to decompose a graph signal on components at different scales, or frequency ranges. The road to multiscale transforms on graphs has originally been tackled in the vertex domain [35] to analyze data on networks. Thereafter, a general design of multiscale transforms was based on the diffusion of signals on the graph structure, leading to the powerful framework of Diffusion Wavelets [36, 37]. This latter works were already based on a diffusion operators, usually a Laplacian or a random walk operator, whose powers are decomposed in order to obtain a multiscale orthonormal basis. The objective was to build a kind of equivalent of discrete wavelet transforms for graph signals.

In this chapter, we focus on two other constructions of multiscale transforms on graphs which are more related to the Graph Fourier Transform. The frequency analysis and filters described here are: 1) the method of [3] which develops an analog of continuous wavelet transform on graphs, 2) approaches that combine filters on graphs with graph decompositions through decimation (pioneered in [4] with decimation of bipartite graphs) or aggregation of nodes; in a nutshell, these methods are very close to the filter banks implementation of discrete wavelets [38].

0.3.1 CONTINUOUS MULTI-SCALE TRANSFORMS

In the following, we work with undirected graphs and $\mathbf{R} = \mathbf{L_n} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$, whose eigenvalues are contained in the interval [0, 2]. The generalization to other operators can be done using the guidelines of previous sections.

The first continuous multiscale transform based on the graph Fourier transform was introduced via the spectral graph wavelet transform [3] (using similar concepts as in diffusion polynomial frames [39]). These wavelets were defined by analogy to the classical wavelets in the following sense. Classically, a wavelet family $\{\psi_{s,\tau}(t)\}$ centered around time τ and at scale s is the translated and dilated version of a mother wavelet $\psi(t)$, generally defined as a zero-mean, square integrable function. Mathematically it is expressed for all $s \in \mathbb{R}^+$ and $\tau \in \mathbb{R}$ as:

$$\psi_{s,\tau}(t) = \frac{1}{s}\psi\left(\frac{t-\tau}{s}\right) \tag{0.39}$$

or equivalently, in the frequency domain, with $\mathbb{F}$ the continuous Fourier transform:

$$\mathbb{F}[\psi_{s,\tau}](\omega) = \mathbb{F}[\psi](s\omega)\,\mathbb{F}[\delta_\tau](\omega) = \widehat{\psi}(s\omega)\,e^{-i\omega\tau}. \tag{0.40}$$

Then, for a signal x, the wavelet coefficient at scale s and instant τ is given by the inner product $\mathbb{W}_{s,\tau}x = \langle x, \psi_{s,\tau}\rangle$.

By analogy, transposing eq. (0.40) with GFT, a spectral graph wavelet $\psi_{s,a}$ at scale s and node a reads[7]:

$$\psi_{s,a} = \mathbf{U}\, h(s\Lambda)\, \mathbf{U}^\top \boldsymbol{\delta}_a, \tag{0.41}$$

where the graph filter $h(\lambda)$ plays the role of the wavelet bandpass filter $\widehat{\psi}(\omega)$. The shifted scaled wavelet identifies to the impulse response of $h(s\lambda)$ to a Dirac localized on node a. In particular, the shape of the filter originally proposed in [3] was:

$$h^{\text{SGW}}(\lambda) = \begin{cases} \lambda_*^{-\alpha}\lambda^{\alpha} & \text{for } \lambda < \lambda_1 \\ q(\lambda) & \text{for } \lambda_1 \leq \lambda \leq \lambda_2 \\ \lambda_2^{\beta}\lambda^{-\beta} & \text{for } \lambda > \lambda_2, \end{cases}$$

with α, β, λ_1 and λ_2 four parameters, and $q(\lambda)$ the unique cubic polynomial interpolation that preserves continuity and the derivative’s continuity. Several properties on the obtained wavelets may be theoretically derived, for instance the notion of locality (the fact that wavelets’ energy is mostly contained around the node on which it is centered). Given a selection of scales $\mathcal{S} = (s_1, \ldots, s_m)$ and a graph signal $\mathbf{x}$, the signal wavelet coefficient associated with the node a and the scale $s \in \mathcal{S}$ reads: $\mathbb{W}_{s,a}\mathbf{x} = \psi_{s,a}^\top \mathbf{x}$. Then, a question that naturally arises is that of invertibility of the wavelet transform: can one recover any signal $\mathbf{x}$ from its wavelet coefficients? As defined here, the wavelet transform is not invertible as it does not take into account –due to the zero-mean constraint of the wavelets– the signal’s information associated with the null frequency, i.e., associated with the first eigenvector $\mathbf{u}_1$. To enable invertibility, one may simply add any low-pass filter $h_0(\lambda)$ to the set of filters $\{h^{\text{SGW}}(s\lambda)\}_{s\in\mathcal{S}}$ of the wavelet transform. We write ϕ_a their associated atoms:

$$\phi_a = \mathbf{U} h_0(\mathbf{\Lambda})\mathbf{U}^\top \delta_a. \tag{0.42}$$

The following theorem derives:

Theorem 1 (Theorem 5.6 in [3])**.** *Given a set of scales $\mathcal{S}$, the set of atoms $\{\{\psi_{s,a}\}_{s\in\mathcal{S}} \cup \{\phi_a\}\}_{a\in\mathcal{V}}$ forms a frame with bounds A, B given by:*

$$A = \min_{\lambda\in[0,\lambda_{max}]} G(\lambda) \tag{0.43}$$

$$B = \max_{\lambda\in[0,\lambda_{max}]} G(\lambda) \tag{0.44}$$

where $G(\lambda) = (h_0(\lambda))^2 + \sum_{s\in\mathcal{S}} (h^{\text{SGW}}(s\lambda))^2$.

In theory, invertibility is guaranteed provided that A is different from 0. Nevertheless, in practice, one should strive to design filter shapes (wavelet and low-pass filters) and to choose a set of scales such that A is as close as possible to B in order to deal with well-conditioned inverses. Doing so, we obtain the so-called tight

[7] Note that the scale parameter stays continuous, but the localization parameter is discretized to the set of nodes a of the graph.

(or snug) frames, i.e. frames such that $A = B$ (or $A \approx B$). An approach is to use classical dyadic decompositions using bandlimited filters such as in Table 1 of [40]. Another desirable property of such frames is their discriminatory power: the ability at discerning different signals only by considering their wavelet coefficients. For a filterbank to be discriminative, each filter element needs to take into account information from a similar number of eigenvalues of the Laplacian. The eigenvalues of an arbitrary graph being unevenly spaced on $[0, \lambda_{\max}]$, one needs to compute or estimate the exact density of the spectrum of the graph under consideration [41].

0.3.2 DISCRETE MULTI-SCALE TRANSFORMS

A second general way to generate multiscale transforms is via a succession of filtering and decimation operations, as in Fig. 0.4. This scheme is usually cascaded as in Fig. 0.5, and each level of the cascade represents a scale of description of the input signal. Thereby, as soon as decimation enters into the process, we talk about "discrete" multi-scale transforms, as the scale parameter can no longer be continuously varied. For details on this particular approach to multiscale transforms in classical signal processing, we refer e.g. to the book by Strang and Nguyen [38]. In the following, we directly consider the graph-based context. Let us first settle notations:

- The decimation operator may be generally defined by partitioning the set of nodes $\mathcal{V}$ into two sets $\mathcal{V}_0$ and $\mathcal{V}_1$. As this subdivision is a partition, we have $\mathcal{V}_0 \cup \mathcal{V}_1 = \mathcal{V}$ and $\mathcal{V}_0 \cap \mathcal{V}_1 = \emptyset$. Moreover, let us define $\downarrow_{\mathcal{V}_i}$ the downsampling operator associated with $\mathcal{V}_i$: given any graph signal $\mathbf{x}$, $\mathbf{y}_i = \downarrow_{\mathcal{V}_i} \mathbf{x}$ is the reduction of $\mathbf{x}$ to $\mathcal{V}_i$. We also define the upsampling operator $\uparrow_{\mathcal{V}_i} = \downarrow_{\mathcal{V}_i}^{\top}$. Given $\mathbf{y}_i$ a signal defined on $\mathcal{V}_i$, $\uparrow_{\mathcal{V}_i} \mathbf{y}_i$ is the zero-padded version of $\mathbf{y}_i$ on the whole graph. The combination of both operators reads: $\uparrow_{\mathcal{V}_i}\downarrow_{\mathcal{V}_i} = \mathrm{diag}(\mathcal{I}_{\mathcal{V}_i})$, where $\mathcal{I}_{\mathcal{V}_i}$ is the indicator function of $\mathcal{V}_i$. Moreover, we define

$$\mathbf{J} = 2 \uparrow_{\mathcal{V}_0}\downarrow_{\mathcal{V}_0} - \mathbf{I} = \mathrm{diag}(\mathcal{I}_{\mathcal{V}_0}) - \mathrm{diag}(\mathcal{I}_{\mathcal{V}_1}). \tag{0.45}$$

- We define two analysis filters: a low-pass graph filter $\mathbf{H}_0$ and a high-pass graph filter $\mathbf{H}_1$, as well as two synthesis graph filters $\mathbf{G}_0$ and $\mathbf{G}_1$. All filters are associated with their frequency responses $h_0(\lambda)$, $h_1(\lambda)$, $g_0(\lambda)$ and $g_1(\lambda)$.

The signal $\mathbf{y}_0 = \downarrow_{\mathcal{V}_0} \mathbf{H}_0\mathbf{x}$ is called the approximation of $\mathbf{x}$, whereas $\mathbf{y}_1 = \downarrow_{\mathcal{V}_1} \mathbf{H}_1\mathbf{x}$ is generally understood as the necessary details to recover $\mathbf{x}$ from its approximation.

Given the scheme of Fig. 0.4, one writes the processed signal $\tilde{\mathbf{x}}$ as:

$$\tilde{\mathbf{x}} = (\mathbf{G}_0 \uparrow_{\mathcal{V}_0}\downarrow_{\mathcal{V}_0} \mathbf{H}_0 + \mathbf{G}_1 \uparrow_{\mathcal{V}_1}\downarrow_{\mathcal{V}_1} \mathbf{H}_1)\,\mathbf{x} \tag{0.46}$$

$$= \frac{1}{2}(\mathbf{G}_0\mathbf{H}_0 + \mathbf{G}_1\mathbf{H}_1)\,\mathbf{x} + \frac{1}{2}(\mathbf{G}_0\mathbf{J}\mathbf{H}_0 - \mathbf{G}_1\mathbf{J}\mathbf{H}_1)\,\mathbf{x}. \tag{0.47}$$

When designing such discrete filterbanks, and in order to enable perfect reconstruction ($\forall \mathbf{x} \in \mathbb{R}^N \;\; \tilde{\mathbf{x}} = \mathbf{x}$), one deals with two main equations linking all four filters and

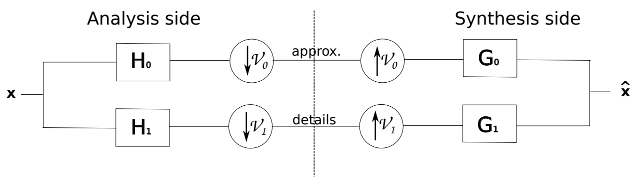

Figure 0.4 A filterbank seen as a succession of filtering and decimating operators.

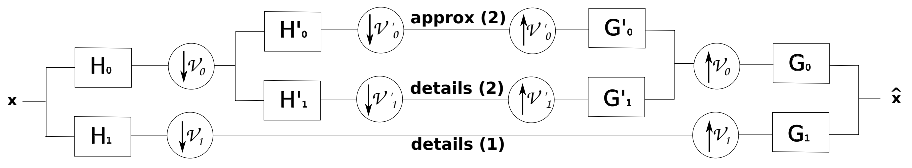

Figure 0.5 A cascaded filterbank (here two levels).

the matrix $\mathbf{J}$:

$$\mathbf{G}_0\mathbf{H}_0 + \mathbf{G}_1\mathbf{H}_1 = 2\mathbf{I} \tag{0.48}$$

$$\mathbf{G}_0\mathbf{J}\mathbf{H}_0 - \mathbf{G}_1\mathbf{J}\mathbf{H}_1 = \mathbf{0} \tag{0.49}$$

Left-multiplying by $\mathbf{U}^\top$ and right-multiplying by $\mathbf{U}$, one obtains equivalently:

$$g_0(\mathbf{\Lambda})h_0(\mathbf{\Lambda}) + g_1(\mathbf{\Lambda})h_1(\mathbf{\Lambda}) = 2\mathbf{I} \tag{0.50}$$

$$g_0(\mathbf{\Lambda})\mathbf{U}^\top\mathbf{J}\mathbf{U}h_0(\mathbf{\Lambda}) - g_1(\mathbf{\Lambda})\mathbf{U}^\top\mathbf{J}\mathbf{U}h_1(\mathbf{\Lambda}) = \mathbf{0} \tag{0.51}$$

Eq.(0.50) is purely spectral, and may be seen as a set of N equations:

$$\forall\lambda_i \quad g_0(\lambda_i)h_0(\lambda_i) + g_1(\lambda_i)h_1(\lambda_i) = 2. \tag{0.52}$$

On the other hand, Eq. (0.51) is not so simple due to the decimation operation and needs to be investigated in detail.

In 1D classical signal processing (equivalent to the undirected circle graph), the decimation operator samples one every two nodes. Moreover, given $\mathbf{x}' = \uparrow_2\downarrow_2 \mathbf{x}$, one classically has the following aliasing phenomenon (Theorem 3.3 of [38]):

$$\mathbb{F}[\mathbf{x}'](\omega) = \frac{1}{2}\left(\mathbb{F}[\mathbf{x}](\omega) + \mathbb{F}[\mathbf{x}](\omega + \pi)\right). \tag{0.53}$$

This means that the decimation operations may be explictly described in the Fourier space, which greatly simplifies calculations by enabling to write Eq. (0.51) as a purely spectral equation as well. Moreover, this also entails that the combined filtering-decimation operations may be understood as a global multiscale filter (yet with discrete scales), thereby connecting with the previous approach. Now, the central question that remains is how to mimic the filtering/decimation approach on general graphs?

In Section 0.3.2.1, it will be shown that for very well structured graphs such as bipartite graphs or m-cyclic graphs, generalizations of the decimation operator may be defined and their effect explicitly described as graph filters. Then the case of arbitrary graph is studied in 0.3.2.2 where the "one every two nodes" paradigm is not transposable directly; several approaches to do that will then be reviewed.

0.3.2.1 Filterbanks on bipartite graphs and other strongly structured graphs

Filterbanks on bipartite graphs. Bipartite graphs are graphs where the nodes are partitioned in two sets of nodes $\mathcal{A}$ and $\mathcal{B}$ such that all links of the graph connect a node in $\mathcal{A}$ with a node in $\mathcal{B}$. On bipartite graphs, the "one-every-two-node" paradigm has a natural extension: decimation ensembles are set to $\mathcal{V}_0 = \mathcal{A}$ and $\mathcal{V}_1 = \mathcal{B}$. Leveraging the fact that bipartite graphs' spectra are symmetrical[8] around the value 1, Narang and Ortega [4] show the bipartite graph spectral folding phenomenon:

$$\forall\lambda \quad \mathbf{Pr}_\lambda \mathbf{J} = \mathbf{J}\mathbf{Pr}_{2-\lambda}, \tag{0.54}$$

(with $\mathbf{Pr}_\lambda$ as in eq. 0.15). This means that for any filter $\mathbf{H} = \mathbf{U}h(\mathbf{\Lambda})\mathbf{U}^\top$ one has:

$$\mathbf{GJ} = \mathbf{U}g(\mathbf{\Lambda})\mathbf{U}^\top\mathbf{J} = \mathbf{JU}g(2\mathbf{I} - \mathbf{\Lambda})\mathbf{U}^\top. \tag{0.55}$$

Eq. (0.51) therefore boils down to a second set of N purely spectral equations:

$$\forall\lambda_i \quad g_0(2-\lambda_i)h_0(\lambda_i) - g_1(2-\lambda_i)h_1(\lambda_i) = 0. \tag{0.56}$$

Eqs. (0.52) and (0.56) give us $2N$ equations linking the $4N$ parameters of the four filters to ensure perfect reconstruction. The other $2N$ degrees of liberty are free to be used to design filterbanks with other desirable properties of filterbanks, such as (bi-)orthogonality, compact-supportness of the atoms, and of course also to adapt to the specific application for which these filters are designed.

Filterbanks on other regular structures. Extending these ideas, several authors have proposed similar approaches to define filterbanks on other regular structures such as M-block cyclic graphs [42] or circulant graphs [43, 44]. In any case, writing decimation operations exactly as graph filters requires regular structures on graphs inducing at least some regularity in the spectrum one may take advantage of. All these approaches lead to exact reconstruction procedures. However, arbitrary graphs do not have such regularities, and other approaches are required.

0.3.2.2 Filterbanks on arbitrary graphs

In order to extend the filterbanks approach to arbitrary graphs, one needs to either generalize the decimation operator, or to bypass decimation via aggregation operators. We discuss in this last part some solutions that were proposed in the literature.

[8] i.e., if λ is an eigenvalue of $\mathcal{L}$, then so is $2 - \lambda$

A complementary and thoughtful discussion on graph decimation, graph aggregation and graph reconstruction can be found in sections III and IV of [45]. For that, the two key questions are:

- how to generalize the decimation operator on arbitrary graphs? We will see that generalized decimation operators either try to mimic the classical decimation and attempt to sample "one every two nodes", or aggregate nodes to form supernodes according in general to some graph cut objective function.
- how to build the new coarser-scale graph from the decimated nodes (or aggregated supernodes)? In fact, after each decimation, if one wants to cascade the filterbank, a new coarse-grain graph has to be built in order to define the next level's graph filters. The nodes (resp. supernodes) are set thanks to decimation (resp. aggregation): but how do we link them together?

Graph decimation. The first work to generalize filterbanks on arbitrary graphs is due to Narang and Ortega [4] and consists in decomposing the graph into an edge-disjoint collection of bipartite subgraphs, and then to apply the scheme presented in Section 0.3.2.1 on each of the subgraphs. In this collection, each subgraph has the same node set, and the union of all subgraphs sums to the original graph. To perform this decomposition (which is not unique), the same authors propose a coloring-based algorithm, called Harary's decomposition. Sakiyama and Tanaka [46] also used this decomposition as one of their design's cornerstone. Unsatisfied by the NP-completeness of the coloring problem (even though heuristics exist), Nguyen and Do [47] propose another decomposition method based on maximum spanning trees.

The bipartite paradigm's main advantage comes from the fact that decimation has an explicit formulation in the graph's Fourier space, thereby enabling exact filter designs depending on the given task. In our opinion, when applied to arbitrary graphs, its main drawback comes from the non-unicity of the bipartite subgraphs decomposition, as well as the seemingly arbitrariness of such a decomposition: from a graph signal point-of-view, what is the meaning of a bipartite decomposition? Letting go of this paradigm, and slightly changing the general filterbank design presented in Section 0.3.2.1, other generalized graph decimations have been proposed. For instance, in [48], the authors propose to separate the graph in two sets $\mathcal{V}_0, \mathcal{V}_1$ according to its max cut, i.e., maximizing $\sum_{i\in\mathcal{V}_0}\sum_{j\in\mathcal{V}_1}\mathbf{W}_{ij}$. In [45], the authors suggest similarly to partition the graph into two sets according to the polarity of the last eigenvector (i.e. the eigenvector associated with the highest frequency). In [49], the authors use an original approach based on random forests to sample nodes, where they have a probabilistic version of "equally-spaced" nodes on the graph.

Graph aggregation. Another paradigm in graph reduction is graph aggregation, where, instead of *selecting* nodes as in decimation, one *aggregates* entire regions of the graph in "supernodes". In general, these methods are based on first clustering the nodes in a partition $\mathcal{P} = \{\mathcal{V}_1, \mathcal{V}_2, \dots, \mathcal{V}_J\}$. Each of these subsets will define a supernode of the coarse graph: this reduced graph thus contains J supernodes.

Once a rule is chosen to connect these supernodes together (the object of the next paragraph), the coarse graph is fully defined, and the method may be iterated to obtain a multiresolution of the initial graph's structure. All these methods differ mainly on the choice of the algorithm or the objective function to find this partition. For instance, one may find methods based on random walks [50], on short time diffusion distances [51], on the algebraic distance [52], etc. Other multiresolution approaches may also be found in [53, 54, 55]. One may also find many approaches from the network science community in the field of community detection [56, 57], and in particular multiscale community detection [58, 59, 60]. All these methods are concerned about providing a multiresolution description of the graph structure, but do not consider any graph signal. Recently, graph signal processing filterbanks have been proposed to define a multiscale representation of graph signals based on these approaches. In [61], we proposed such an approach where we define a generalized Haar filterbank: instead of averaging and differentiating over pairs of nodes as in the classical Haar filterbank, we average and differentiate over the subsets $\mathcal{V}_j$ of a partition in subgraphs. In [62], the authors propose a similar approach and define other types of filterbanks such as the hierarchical graph Laplacian eigentransform. Another similar Haar filterbank may be found in [37]. All these methods are independent of exactly which aggregation algorithm one chooses to find the partition. Let us also cite methods that provide multiresolution approaches without necessarily defining low-pass and high-pass filters explictly in the graph Fourier domain [63, 37, 64]. Finally, let us point out that these methods may be extended to graph partitions $\mathcal{P}$ containing overlaps, as in [65].

Coarse graph reconstruction. Once one decided how to decimate nodes, or how to partition them in supernodes, how should one connect these nodes together in order to form a consistent reduced graph? In order to satisfy constraints such as interlacement (coarsely speaking, that the spectrum of the reduced graph is representative of the spectrum of the initial graph) and sparsity, Shuman et al. [45] propose a Kron reduction followed by a sparsification step. The Laplacian of the reduced graph is thus defined as the Schur complement of the initial graph's Laplacian relative to the unsampled nodes. The sparsification step is performed via a sparsifier based on effective resistances by Spielman and Srivastava [66] that approximately preserves the spectrum. In [49], the authors propose another approach to the intuitive idea that the initial and coarse graph should have similar spectral properties: they look for the coarse Laplacian matrix that satisfies an intertwining relation. In [47], the authors connect nodes according to the set of nested bipartite graphs obtained by their maximum spanning tree algorithm. In aggregation methods [61, 62], there is an inherent natural way of connecting supernodes: the weight of the link between supernodes i and j is equal to the sum of the weights of the links connecting nodes in $\mathcal{V}_i$ to nodes in $\mathcal{V}_j$.

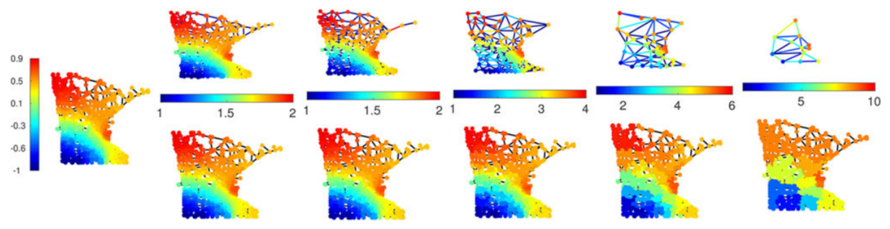

FIGURE 0.6 **An example of multiresolution analyis of a graph signal (from [61]).**

Left: original smooth graph signal (sum of the five lowest Fourier modes normalized by its maximum absolute value) defined on the Minnesota traffic graph. The vertical colorbar of this figure is valid for all graph signals represented on this figure. Top row: successive approximations of the graph signal. The horizontal colorbar on the bottom of each figure corresponds to the weights of the links of the corresponding coarsened graph. Figures which do not have a bottom horizontal colorbar represent binary graphs. Lower row: for each of the successive approximations, we represent the upsampled reconstructed graph signal obtained from the corresponding approximations.

Illustrations. We show in Fig. 0.6 an example of multiresolution analysis of a given graph signal on the Minnesota traffic graph, using a method of successive graph aggregation to compute the details and approximations at different scales. The specific method for this illustration is the one detailed in [61].

CONCLUSION

The purpose of this chapter was to introduce the reader to a basic understanding of what is a Graph Fourier Transform. We stressed how it can be generally introduced for undirected or directed graphs, by choosing a reference operator whose spectral domain will define the frequency domain for graph signals. Then, we led the reader to the more elaborated designs of graph filters and multiscale transforms on graphs. The first section is voluntarily introductory, and almost self-contained. Indeed, we endeavoured to delineate a general guideline that shows, in an original manner, that there is no major discrepancy between choosing a Laplacian, an Adjacency, or a random walk operator... as long as one chooses accordingly the appropriate notion of frequency to analyze graph signals. Then, after a proper definition of graph filters, our objective was to review the literature and to propose guidelines and pointers to the relevant results on graph filters and related multiscale transforms. This last part being written as a review, we beg for reader's indulgence, as many details are skipped, and some works are only reported here in a sketchy manner.

ACKNOWLEDGEMENTS

Work supported by the ANR grant GRAPHSIP ANR-14-CE27-0001-02 and ANR-14-CE27-0001-03, as well as the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program Investissement d?avenir.

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

[2] 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," *Signal Processing Magazine, IEEE*, vol. 30, no. 3, pp. 83–98, May 2013.

[3] D. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Applied and Computational Harmonic Analysis*, vol. 30, no. 2, pp. 129–150, 2011.

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

[5] F. Chung, *Spectral graph theory*. Amer Mathematical Society, 1997, no. 92.

[6] L. Grady and J. R. Polimeni, *Discrete Calculus: Applied Analysis on Graphs for Computational Science*. Springer, 2010.

[7] A. Barrat, M. Barthlemy, and A. Vespignani, *Dynamical processes on complex networks*. Cambridge University Press, 2008.

[8] S. Kar and J. M. F. Moura, "Consensus + innovations distributed inference over networks: cooperation and sensing in networked systems," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 99–109, May 2013.

[9] H. Rabiei, F. Richard, O. Coulon, and J. Lefevre, "Local spectral analysis of the cerebral cortex: New gyrification indices," *IEEE Transactions on Medical Imaging*, vol. 36, no. 3, pp. 838–848, March 2017.

[10] R. Singh, A. Chakraborty, and B. S. Manoj, "Graph fourier transform based on directed laplacian," in *2016 International Conference on Signal Processing and Communications (SPCOM)*, June 2016, pp. 1–5.

[11] H. Sevi, G. Rilling, and P. Borgnat, "Multiresolution analysis of functions on directed networks," in *Wavelets and Sparsity XVII*, Aug 2017.

[12] F. Chung, "Laplacians and the Cheeger inequality for directed graphs," *Annals of Combinatorics*, vol. 9, no. 1, pp. 1–19, 2005. [Online]. Available: http://link.springer.com/article/10.1007/s00026-005-0237-z

[13] G. Yu and H. Qu, "Hermitian laplacian matrix and positive of mixed graphs," *Applied Mathematics and Computation*, vol. 269, no. Supplement C, pp. 70 – 76, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0096300315009649

[14] S. Sardellitti, S. Barbarossa, and P. D. Lorenzo, "On the graph fourier transform for directed graphs," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 6, pp. 796–811, Sept 2017.

[15] R. Shafipour, A. Khodabakhsh, G. Mateos, , and E. Nikolova, "A digraph fourier transform with spread frequency components," in *Proc. of IEEE Global Conf. on Signal and Information Processing*, Nov 2017.

[16] A. Sandryhaila and J. Moura, "Discrete signal processing on graphs: Frequency analysis," *Signal Processing, IEEE Transactions on*, vol. 62, no. 12, pp. 3042–3054, June 2014.

[17] H. Sevi, G. Rilling, and P. Borgnat, "Analyse fréquentielle et filtrage sur graphes dirigés," in *26e Colloque sur le Traitement du Signal et des Images. GRETSI-2017*, Sept 2017, p. id. 220.

[18] S. Ben Alaya, P. Gonçalves, and P. Borgnat, "Linear prediction on graphs based on autoregressive models," in *Graph Signal Processing Workshop*, May 2016.

[19] B. Girault, P. Goncalves, and E. Fleury, "Translation on graphs : An isometric shift operator," *Signal Processing Letters, IEEE*, vol. 22, no. 12, Dec. 2015.

[20] A. Anis, A. Gadde, and A. Ortega, "Efficient sampling set selection for bandlimited graph

signals using graph spectral proxies," *IEEE Transactions on Signal Processing*, vol. 64, no. 14, pp. 3775–3789, July 2016.

[21] W. Zachary, "An information flow model for conflict and fission in small groups," *Journal of anthropological research*, vol. 33, no. 4, pp. 452–473, 1977.

[22] L. LeMagoarou, R. Gribonval, and N. Tremblay, "Approximate fast graph Fourier transforms via multi-layer sparse approximations," *IEEE Transactions on Signal and Information Processing over Networks*, vol. PP, no. 99, pp. 1–1, 2017.

[23] L. LeMagoarou, N. Tremblay, and R. Gribonval, "Analyzing the approximation error of the fast graph fourier transform," in *proceedings of the Asilomar Conference on Signals, Systems, and Computers*, Nov 2017.

[24] 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, Jan. 2017.

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

[26] X. Shi, H. Feng, M. Zhai, T. Yang, and B. Hu, "Infinite Impulse Response Graph Filters in Wireless Sensor Networks," *Signal Processing Letters, IEEE*, vol. 22, no. 8, pp. 1113–1117, 2015.

[27] J. Liu, E. Isufi, and G. Leus, "Autoregressive moving average graph filter design," in *6th Joint WIC/IEEE Symposium on Information Theory and Signal Processing in the Benelux*, May 2016.

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

[29] A. G. Marques, S. Segarra, G. Leus, and A. Ribeiro, "Stationary graph processes and spectral estimation," *IEEE Transactions on Signal Processing*, vol. 65, no. 22, pp. 5911–5926, Nov 2017.

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

[31] N. Tremblay, G. Puy, R. Gribonval, and P. Vandergheynst, "Compressive spectral clustering," in *Proceedings of the 33 rd International Conference on Machine Learning (ICML)*, vol. 48. JMLR: W&CP, 2016, pp. 1002–1011. [Online]. Available: http://jmlr.csail.mit.edu/proceedings/papers/v48/tremblay16.pdf

[32] A. Susnjara, N. Perraudin, D. Kressner, and P. Vandergheynst, "Accelerated filtering on graphs using Lanczos method," *arXiv preprint arXiv:1509.04537*, 2015. [Online]. Available: https://arxiv.org/abs/1509.04537

[33] E. Gallopoulos and Y. Saad, "Efficient Solution of Parabolic Equations by Krylov Approximation Methods," *SIAM Journal on Scientific and Statistical Computing*, vol. 13, no. 5, pp. 1236–1264, 1992. [Online]. Available: https://doi.org/10.1137/0913071

[34] Y. Saad, *Numerical Methods for Large Eigenvalue Problems*, 2nd ed., ser. Classics in Applied Mathematics 66. SIAM, 2011.

[35] M. Crovella and E. Kolaczyk, "Graph wavelets for spatial traffic analysis," in *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, vol. 3, 2003, pp. 1848–1857.

[36] R. Coifman and M. Maggioni, "Diffusion wavelets," *Applied and Computational Harmonic Analysis*, vol. 21, no. 1, pp. 53–94, 2006.

[37] M. Gavish, B. Nadler, and R. R. Coifman, "Multiscale wavelets on trees, graphs and high dimensional data: Theory and applications to semi supervised learning," in *Proceedings of the*

27th International Conference on Machine Learning (ICML-10), 2010, pp. 367–374.
[38] G. Strang and T. Nguyen, *Wavelets and filter banks*. SIAM, 1996.
[39] M. Maggioni and H. Mhaskar, "Diffusion polynomial frames on metric measure spaces," *Applied and Computational Harmonic Analysis*, vol. 24, no. 3, pp. 329–353, May 2008. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S106352030700070X
[40] N. Leonardi and D. Van De Ville, "Tight wavelet frames on multislice graphs," *Signal Processing, IEEE Transactions on*, vol. 61, no. 13, pp. 3357–3367, July 2013.
[41] D. Shuman, C. Wiesmeyr, N. Holighaus, and P. Vandergheynst, "Spectrum-adapted tight graph wavelet and vertex-frequency frames," *Signal Processing, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2015.
[42] O. Teke and P. P. Vaidyanathan, "Extending Classical Multirate Signal Processing Theory to Graphs ; Part I: Fundamentals," *IEEE Transactions on Signal Processing*, vol. 65, no. 2, pp. 409–422, Jan. 2017.
[43] V. N. Ekambaram, G. C. Fanti, B. Ayazifar, and K. Ramchandran, "Circulant structures and graph signal processing," in *2013 IEEE International Conference on Image Processing*, Sep. 2013, pp. 834–838.
[44] M. S. Kotzagiannidis and P. L. Dragotti, "Splines and wavelets on circulant graphs," *CoRR*, vol. abs/1603.04917, 2016. [Online]. Available: http://arxiv.org/abs/1603.04917
[45] D. I. Shuman, M. J. Faraji, and P. Vandergheynst, "A Multiscale Pyramid Transform for Graph Signals," *IEEE Transactions on Signal Processing*, vol. 64, no. 8, pp. 2119–2134, Apr. 2016.
[46] A. Sakiyama and Y. Tanaka, "Oversampled graph Laplacian matrix for graph filter banks," *Signal Processing, IEEE Transactions on*, vol. 62, no. 24, pp. 6425–6437, Dec 2014.
[47] H. Nguyen and M. Do, "Downsampling of signals on graphs via maximum spanning trees," *Signal Processing, IEEE Transactions on*, vol. 63, no. 1, pp. 182–191, Jan 2015.
[48] S. Narang and A. Ortega, "Local two-channel critically sampled filter-banks on graphs," in *Image Processing (ICIP), 2010 17th IEEE International Conference on*, Sept 2010, pp. 333–336.
[49] L. Avena, F. Castell, A. Gaudillière, and C. Mélot, "Intertwining wavelets or Multiresolution analysis on graphs through random forests," 2017.
[50] S. Lafon and A. B. Lee, "Diffusion maps and coarse-graining: a unified framework for dimensionality reduction, graph partitioning, and data set parameterization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 9, pp. 1393–1403, Sep. 2006.
[51] O. E. Livne and A. Brandt, "Lean Algebraic Multigrid (LAMG): Fast Graph Laplacian Linear Solver," *SIAM Journal on Scientific Computing*, vol. 34, no. 4, pp. B499–B522, 2012. [Online]. Available: https://doi.org/10.1137/110843563
[52] D. Ron, I. Safro, and A. Brandt, "Relaxation-based coarsening and multiscale graph organization," *Multiscale Modeling & Simulation*, vol. 9, no. 1, pp. 407–423, 2011. [Online]. Available: http://epubs.siam.org/doi/abs/10.1137/100791142
[53] I. Dhillon, Y. Guan, and B. Kulis, "Weighted Graph Cuts without Eigenvectors A Multilevel Approach," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 29, no. 11, pp. 1944–1957, Nov. 2007.
[54] G. Karypis and V. Kumar, "A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs," *SIAM Journal on Scientific Computing*, vol. 20, no. 1, pp. 359–392, 1998. [Online]. Available: http://dx.doi.org/10.1137/S1064827595287997
[55] F. Murtagh and P. Contreras, "Algorithms for hierarchical clustering: an overview," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 2, no. 1, pp. 86–97, 2012. [Online]. Available: http://dx.doi.org/10.1002/widm.53

[56] M. E. Newman, "Modularity and community structure in networks," *Proceedings of the National Academy of Sciences*, vol. 103, no. 23, pp. 8577–8582, 2006.

[57] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3-5, pp. 75–174, 2010.

[58] J. Reichardt and S. Bornholdt, "Statistical mechanics of community detection," *Phys. Rev. E*, vol. 74, no. 1, p. 016110, Jul. 2006. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevE.74.016110

[59] M. Schaub, J. Delvenne, S. Yaliraki, and M. Barahona, "Markov dynamics as a zooming lens for multiscale community detection: non clique-like communities and the field-of-view limit," *PloS one*, vol. 7, no. 2, p. e32210, 2012.

[60] N. Tremblay and P. Borgnat, "Graph Wavelets for Multiscale Community Mining," *Signal Processing, IEEE Transactions on*, vol. 62, no. 20, pp. 5227–5239, Oct. 2014.

[61] ——, "Subgraph-Based Filterbanks for Graph Signals," *IEEE Transactions on Signal Processing*, vol. 64, no. 15, pp. 3827–3840, Aug. 2016.

[62] J. Irion and N. Saito, "Applied and computational harmonic analysis on graphs and networks," in *Wavelets and Sparsity XVI*, vol. 9597, Sep. 2015, pp. 95 971F–95 971F–15. [Online]. Available: http://dx.doi.org/10.1117/12.2186921

[63] A. B. Lee, B. Nadler, and L. Wasserman, "Treelets: An Adaptive Multi-Scale Basis for Sparse Unordered Data," *The Annals of Applied Statistics*, vol. 2, no. 2, pp. 435–471, 2008. [Online]. Available: http://www.jstor.org/stable/30244207

[64] G. Mishne, R. Talmon, I. Cohen, R. R. Coifman, and Y. Kluger, "Data-Driven Tree Transforms and Metrics," *IEEE Transactions on Signal and Information Processing over Networks*, vol. PP, no. 99, pp. 1–1, 2017.

[65] A. D. Szlam, M. Maggioni, R. R. Coifman, and J. C. Bremer Jr, "Diffusion-driven multiscale analysis on manifolds and graphs: top-down and bottom-up constructions," in *Proc SPIE*, vol. 5914, 2005, p. 59141D.

[66] D. A. Spielman and N. Srivastava, "Graph Sparsification by Effective Resistances," *SIAM Journal on Computing*, vol. 40, no. 6, pp. 1913–1926, 2011. [Online]. Available: http://dx.doi.org/10.1137/080734029