Natural Graph Wavelet Dictionaries: Methods and Applications

By

HAOTIAN LI

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

APPLIED MATHEMATICS

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

Naoki Saito, Chair

Shiqian Ma

Qinglan Xia

Committee in Charge

2021

To my parents.

Contents

Natural Graph Wavelet Dictionaries: Methods and Applications

Abstract

The graph Laplacian is widely used in the graph signal processing field. When attempting to design graph wavelet transforms, people have been using its eigenvalues and eigenvectors in place of the frequencies and complex exponentials that are the backbone of the Fourier theory on Euclidean domains. However, this viewpoint could be misleading since the Laplacian eigenvalues cannot be interpreted as the frequencies of the eigenvectors on a general graph. Instead, we introduce and review several "natural" metrics of graph Laplacian eigenvectors, and propose a new way to naturally organize the eigenvectors by incorporating these metrics into a "dual" graph. We then introduce a set of novel multiscale basis transforms for graph signals fully utilizing this dual graph, rather than simply using the eigenvalue ordering. These basis dictionaries can be seen as generalizations of the classical Shannon/Meyer wavelet packet dictionary to arbitrary graphs, and do not rely on the frequency interpretation of Laplacian eigenvalues. We describe the algorithms (involving vector rotations, or orthogonalizations, or lapped orthogonal projections) to efficiently approximate and compress signals through the best-basis algorithm, and demonstrate the strengths of these basis dictionaries for graph signals on sunflower graphs and road traffic networks. Lastly, we propose a way to modify the spectral filters in the spectral graph wavelet transform by utilizing the structure of the dual graph instead of using the eigenvalue-dependent smooth functions. By doing so, we generate a redundant wavelet frame, propose a way to reduce its redundancy, and discuss its potential for applications.

Acknowledgments

First and foremost, I would like to express my deep and sincere gratitude to my advisor, Professor Naoki Saito. I have benefited immensely from working with him. He has been highly involved in my research by providing me with so many great ideas and valuable suggestions. He has also provided me with financial support, letters of recommendations, and general guidance with my graduate studies. I was very fortunate to have him as my advisor, and I really appreciate all the things he has done for me.

Along with Professor Saito, I would like to thank Professor Shiqian Ma and Professor Qinglan Xia for serving as my dissertation committee members. I am grateful to them for taking time to review my dissertation and provide helpful feedback.

I would like to thank my colleagues in our group, Alex Berrian, Jeff Irion, Yiqun Shao, and David Weber for their generous help with my research. In particular, the `MultiscaleGraphSignalTransforms.jl` implemented by Jeff Irion, Yiqun Shao and Professor Saito is very useful in my own package development, which is now incorporated as a part of `MultiscaleGraphSignalTransforms.jl` for maintenance purpose.

I would like to thank my office mates, Anthony Nguyen, Ka Wai Wong (Karry), and Jake Reschke, for their friendships and having all those great discussions with me about research, graduate school, and career paths.

I would like to thank the faculty of the math department for the beautiful lectures they delivered and the interesting seminars they organized. And I would like to thank our department staff, especially Tina Denena and Sarah Driver, who have always been there when I need help.

Last but not least, I would like to thank my mother Fenghua Wu and my fiancée Yihui Zhu for their love and support. No matter what, they have always been there for me and have been a great encouragement during my PhD studies.

Preliminary versions of parts of this dissertation (specifically, portions of Chapters 1, 2, 3, 4, 5, 6, 7 and 9) were published in papers [**9**, **51**].

This research was partially supported by the US National Science Foundation grants DMS-1418779, DMS-1819222, DMS-1912747, CCF-1934568, DMS-2012266; the US Office of Naval Research grant N00014-20-1-2381; and Russell Sage Foundation Grant 2196.

CHAPTER 1

Introduction

There is an explosion of interest and demand to analyze data sampled on graphs and networks. This has motivated development of more flexible yet mathematically sound *dictionaries* (i.e., an overcomplete collection of atoms or basis vectors) for data analysis and signal processing on graphs. Our main goal here is to build *smooth* multiscale localized basis dictionaries on an input graph, with beneficial reconstruction and sparsity properties, and to fill the "gap" left from the previous graph basis dictionaries [**33**, **34**, **35**, **36**, **77**] constructed by Prof. Saito's group as we explain below. Our approach differs from the standard literature as we fully utilize both the similarities between the nodes (through the graph adjacency matrix) and the similarities between the eigenvectors of the graph Laplacian matrix (through new nontrivial eigenvector distances).

Previous approaches to construct such graph basis dictionaries break down into two main categories. The first category partitions the nodes through recursive graph cuts to generate multiscale basis dictionaries. This includes: the Hierarchical Graph Laplacian Eigen Transform (HGLET) [**34**]; the Generalized Haar-Walsh Transform (GHWT) [**33**]; its extension, the eGHWT [**77**]; and other Haar-like graph wavelets (see, e.g., [**11**, **25**, **46**, **58**, **83**]). But their basis vectors either are nonsmooth piecewise constants or have non-overlapping supports. The second category uses spectral filters on the Laplacian (or diffusion kernel) eigenvalues to generate multiscale smooth dictionaries. This includes: the Spectral Graph Wavelet Transform (SGWT) [**29**]; Diffusion Wavelets [**13**]; extensions to spectral graph convolutional networks [**48**]. However, these dictionaries do not fully address the relationships among eigenvectors [**10**, **51**, **72**], which should be utilized for graph dictionary construction; instead, they focus on the eigenvalue distributions to organize the corresponding eigenvectors (although there are some works, e.g., [**64**, **66**, **80**], which recognized the graph structures strongly influence the eigenvector behaviors). These relationships among eigenvectors can result from eigenvector localization in different clusters,

differing scales in multi-dimensional data, etc. These notions of similarity and distance between eigenvectors, while studied in the literature [**10**, **51**, **72**], have yet to be incorporated into building localized dictionaries on graphs.

We first propose a novel way to organize the eigenvectors by incorporating these eigenvector metrics into a complete dual graph as the dual domain. We then construct three graph wavelet packet dictionaries and two graph wavelet frames based on the dual graph.

Our first graph wavelet packet dictionary, detailed in Chapter 6, fills the "gap" in the cycle of our development of the graph basis dictionaries, i.e., HGLET, GHWT, and eGHWT. It is a direct generalization of the classical wavelet packet dictionary [**54**, Chap. 8] to the graph setting: we hierarchically partition the dual domain to generate a tree-structured "subbands" each of which is an appropriate subset of the graph Laplacian eigenvectors. We also want to note the following correspondence: The HGLET [**34**] is a graph version of the Hierarchical Block Discrete Cosine Transform (DCT) dictionary [**54**, Sec. 8.3] (i.e., the non-smooth non-overlapping version of the local cosine dictionary [**15**], [**54**, Sec. 8.5]), and the former exactly reduces to the latter if the input graph is P_N, a path graph with N nodes. The former hierarchically partitions the input graph while the latter does the same (with a non-adaptive manner) on the unit interval $[0,1]$ in the *time* domain. On the other hand, the GHWT [**33**] is a graph version of the Haar-Walsh wavelet packet dictionary [**14**], [**54**, Sec. 8.1], and the former exactly reduces to the latter if the input graph is P_N. The latter hierarchically partitions the interval $[0,N)$ in the *sequency* domain while the former does the same by the graph domain partitioning plus reordering; see [**33**, **35**, **36**] for the details. Our first graph wavelet packet dictionary is a graph version of the *Shannon* wavelet packet dictionary [**54**, Sec. 8.1.2], which hierarchically partitions the interval $[0,1/2)$ (or $[0,\pi]$ depending on how one defines the Fourier transform) in the *frequency* domain. Again, the former essentially reduces to the latter if the input graph is P_N.

Our second graph wavelet packet dictionary, detailed in Chapter 7, is obtained by partitioning *both* the input graph *and* its dual domain; more precisely, we first hierarchically partition the dual domain, and then partition the input graph with constraints imposed by the dual domain partition. This approach parallels and generalizes classical time-frequency analysis, where the time domain is replaced by a general *node-domain* geometry and the frequency domain is replaced by a general *eigenvector-domain*

organization. A version of this approach of node-eigenvector organization that embeds the eigenvectors to a one-dimensional Euclidean domain has also been considered as a visualization technique for low-frequency eigenvectors on clustered graphs [**26**].

Our third graph wavelet packet dictionary, detailed in Chapter 8, is closely related to our first one. It also bipartitions the dual domain recursively but *smoothly with overlaps*, which is achieved by generalizing the smooth orthogonal projectors [**15**] for graph setting. It is a generalization of the classical *Meyer* wavelet packet dictionary [**54**, Sec. 7.2.2, 8.4.2]. Again, the former essentially reduces to the latter if the input graph is P_N.

Our first graph wavelet frame, detailed in Chapter 10, modifies the spectral filters of the SGWT by fully utilizing the information of the dual domain instead of using the eigenvalue-dependent smooth functions. Specifically, it is constructed by the set of Gaussian filters on the dual domain. Its wavelet frame vectors inherit the behavior patterns from the eigenvectors, which is useful for the graph signal feature extraction. On the other hand, our second graph wavelet frame is just a reduced version of the first, which has much less redundancy and could be used in graph signal approximation and compression.

We aim for the significance and impact of this research to be twofold. First, these results will provide the first set of graph wavelet dictionaries that adaptively scale to the local structure of the graph. This is especially important for graphs with complicated multiscale structure, whose graph Laplacians have localized eigenvectors, for example. Second, in the long term, this is a first method of systematically using the novel concept of eigenvector dual geometry [**10**, **51**, **72**]. This direction can set a path for future modification of spectral graph theory applications to incorporate dual geometry.

The structure of this dissertation is organized as follows. Chapter 2 reviews fundamentals: the basics of graphs, graph Laplacians and graph Fourier transform, graph wavelet transforms and frames that were proposed previously, as well as graph wavelet packets and the best-basis algorithm. Chapter 3 reviews the issue of viewing the eigenvalues as frequencies on general graphs. Chapter 4 introduces and reviews the nontrivial metrics of graph Laplacian eigenvectors, and studies their relationships. Chapter 5 presents the dual geometry/eigenvector-domain of an input graph with numerical experiments. Chapter 6 presents a natural graph wavelet packet dictionary constructed through hierarchical partition of the eigenvector-domain. Chapter 7 presents a second version of a natural graph wavelet packet

Symbol	Usual Meaning
N	the number of nodes/eigenvectors in a graph or the length of the vector
M	the number of edges in a graph
j	the scale/level index
k	the subgraph index
l	an index for the eigenvectors or the basis vectors, $l = 0 : N-1$
x	a location/node index, $x = 1 : N$
i	a generic index variable
d	the non-trivial eigenvector distance
G	an input/primal graph
V	the node set of G
$V_k^{(j)}$	the node set of a subgraph
$G^\star$	the dual graph of G
$V^\star$	the set of the eigenvectors or the node set of $G^\star$
$V_k^{\star(j)}$	the subset of $V^\star$
ϕ_l	a Laplacian eigenvector
$\phi_{k,l}^{(j)}$	an HGLET basis vector
$\psi_{k,l}^{(j)}$	a NGWP basis vector
$\psi_{l,x}$	a NGWF vector
$\overline{\psi}_{l,x}$	a rNGWF vector

TABLE 1.1. List of symbols we consistently use throughout this dissertation.

dictionary constructed through a pair of hierarchical partitions, one on the input graph and one on its dual domain. Chapter 8 presents a third version of natural graph wavelet packet dictionary constructed through hierarchical partition of the eigenvector-domain but smoothly with overlaps. In Chapter 9, we demonstrate the usefulness of our proposed graph wavelet packet dictionaries in graph signal approximation using numerical experiments. In Chapter 10, we present the natural graph wavelet frame and its reduced version, and show their potentials for applications. We conclude with discussing our findings gained through these numerical experiments and near-future projects for further improvements of our dictionaries.

Throughout this dissertation, we make an effort to be consistent with our notation. Table 1.1 shows what we typically use and their corresponding meaning.

Accompanying this dissertation is the `NGWP.jl` package [**50**], which is written in `Julia` [**4**] and is now merged into the `MultiscaleGraphSignalTransforms.jl` package [**32**]. It includes code scripts for reproducing many of the figures and tables in this dissertation, which we list as below. We note that most of the figures are generated by the `Plots.jl` package [**6**].

1.1. List of Reproducible Figures and Tables

Figures

Tables

CHAPTER 2

Background

2.1. Basics of Graph Theory

In this section, we cover some basics of graph theory and introduce our notation that will be used throughout this dissertation.

A graph $G = (V, E)$ consists of a set of nodes (or vertices) $V = V(G) = \{v_1, v_2, \ldots, v_N\}$[1], where $N := |V(G)|$; and a set of edges $E = E(G) = \{e_1, e_2, \ldots, e_M\}$ where each e_k connects two nodes, say, i and j, and $M := |E(G)|$. If $M, N < \infty$, then G is a finite graph. If any $e_k \in E(G)$ does not specify a direction, then G is undirected. If any two nodes $i, j \in V(G)$ are connected by a sequence of head-tail edges, then G is connected. Furthermore, if G does not have any loops (an edge connecting a node to itself) or multiple edges (more than one edge connecting a pair of nodes), then G is a simple graph. In this dissertation, we only consider *finite undirected connected simple* graphs. We use $\boldsymbol{f} = [f(1), \ldots, f(N)]^\top \in \mathbb{R}^N$ to denote a graph signal on G, and we define $\mathbf{1} := [1, \ldots, 1]^\top \in \mathbb{R}^N$.

We now discuss several matrices associated with graphs. The information in both V and E is captured by the *edge weight matrix* $W = W(G) \in \mathbb{R}^{N \times N}$, where $W_{ij} \geq 0$ is the edge weight between nodes i and j. In an unweighted graph, this is restricted to be either 0 or 1, depending on whether nodes i and j are adjacent, and we may refer to $W(G)$ as an *adjacency matrix*. In a weighted graph, W_{ij} indicates the *affinity* between nodes i and j. In either case, since G is undirected, $W(G)$ is a symmetric matrix. We then define the *incidence matrix* $Q(G) = [\boldsymbol{q}_1 | \cdots | \boldsymbol{q}_M] \in \mathbb{R}^{N \times M}$ where $\boldsymbol{q}_k$ indicates the head and tail of the kth edge $e_k \in E$. However, we note that we need to orient each edge of G in an arbitrary manner to form a directed graph temporarily in order to construct its incidence matrix. For example, suppose e_k joins nodes i and j, then we can set either $(Q_{ik}, Q_{jk}) = (-\sqrt{W_{ij}}, \sqrt{W_{ij}})$ or $(\sqrt{W_{ij}}, -\sqrt{W_{ij}})$. Of course, we set $Q_{lk} = 0$ for $l \neq i, j$. Note that $Q(G)^\top$ can be viewed as the *graph gradient operator* ∇_G. Next, we define the *degree matrix* $D(G)$ as the diagonal matrix with entries $D_{ii} = \sum_j W_{ij}$. With this in place, we are able to

[1] For simplicity, we typically associate each node with its index and write i in place of v_i.

define the *(unnormalized) Laplacian* matrix, *random-walk normalized Laplacian* matrix, and *symmetric normalized Laplacian* matrix, respectively, as

$$\begin{aligned} L(G) &:= D(G) - W(G) \\ L_{\mathrm{rw}}(G) &:= D(G)^{-1} L(G) \\ L_{\mathrm{sym}}(G) &:= D(G)^{-1/2} L(G) D(G)^{-1/2}. \end{aligned} \tag{2.1}$$

We use $0 = \lambda_0 \leq \lambda_1 \leq \ldots \leq \lambda_{N-1}$ to denote the sorted Laplacian eigenvalues and $\phi_0, \phi_1, \ldots, \phi_{N-1}$ to denote their corresponding eigenvectors with unit length, where the specific Laplacian matrix to which they refer will be clear from either contexts or subscripts.

REMARK 2.1.1. *Note that the eigenvectors are unique up to sign flips provided that all eigenvalues are simple. We standardize the sign of ϕ_l as follows: 1) if* $\max(\phi_l) < -\min(\phi_l)$*, we flip the sign of ϕ_l; 2) if* $\max(\phi_l) = -\min(\phi_l)$*, we adjust the sign of ϕ_l to ensure its first non-zero entry is positive.*

REMARK 2.1.2. *If the multiplicity of an eigenvalue is greater than* 1*, then the choice of corresponding eigenvectors is not unique even up to sign flips. One can apply the varimax rotation algorithm (i.e., Algorithm 2) discussed in Chapter 6 on the corresponding eigenspace to ensure the sparsest eigenvectors are selected. However, this case did not happen among the numerical experiments of this dissertation.*

Denoting $\Phi := [\phi_0|\cdots|\phi_{N-1}]$ and $\Lambda := \mathrm{diag}([\lambda_0, \ldots, \lambda_{N-1}])$, the eigendecomposition of $L(G)$ can be written as $L(G) = \Phi\Lambda\Phi^\top$. Similarly, denoting $\Phi_{\mathrm{sym}} := \left[\phi_0^{\mathrm{sym}}|\cdots|\phi_{N-1}^{\mathrm{sym}}\right]$ and $\Lambda_{\mathrm{sym}} := \mathrm{diag}(\left[\lambda_0^{\mathrm{sym}}, \ldots, \lambda_{N-1}^{\mathrm{sym}}\right])$, the eigendecomposition of $L_{\mathrm{sym}}(G)$ can be written as $L_{\mathrm{sym}}(G) = \Phi_{\mathrm{sym}}\Lambda_{\mathrm{sym}}\Phi_{\mathrm{sym}}^\top$. Both L and L_{sym} are symmetric matrices and therefore their eigenvectors form orthonormal bases (ONBs) for $\mathbb{R}^N$.

Since we only consider connected graphs here, we have $0 = \lambda_0 \lneq \lambda_1$, and $\phi_0 = \mathbf{1}/\sqrt{N}$, which is called the direct current component vector or the *DC vector* for short; $0 = \lambda_0^{\mathrm{sym}} \lneq \lambda_1^{\mathrm{sym}}$, and $\phi_0^{\mathrm{sym}}(i) = \sqrt{D_{ii}/\sum_{j=1}^N D_{jj}}$. The second smallest eigenvalue λ_1 is called the *algebraic connectivity* of G and the corresponding eigenvector ϕ_1 is called the *Fiedler vector* of G. The Fiedler vector plays an important role in graph partitioning and spectral clustering; see, e.g., [**87**], which suggests the use of the Fiedler vector of $L_{\mathrm{rw}}(G)$ for spectral clustering over that of the other Laplacian matrices. Furthermore, for $l = 1 : N-1$, we have $\mathbf{1}^\top \phi_l = 0$, i.e., $\sum_{x=1}^N \phi_l(x) = 0$, due to $\langle \phi_0, \phi_l \rangle = 0$, while $\sum_{x=1}^N \phi_l^{\mathrm{sym}}(x) \neq 0$ since ϕ_0^{sym} is not a constant vector.

For a *weighted* graph $G(V,E,W)$, one may also be interested in studying the connectivity of the graph. We denote the *unweighted version of the adjacency matrix* and the *unweighted version of the incidence matrix*, respectively, as

$$\tilde{W}(G) := \operatorname{sign}.(W(G)) \in \mathbb{R}^{N\times N}$$

$$\tilde{Q}(G) = [\tilde{\boldsymbol{q}}_1|\cdots|\tilde{\boldsymbol{q}}_M] := \operatorname{sign}.(Q(G)) \in \mathbb{R}^{N\times M}$$

where $\operatorname{sign}.(\cdot)$ applies the sign function in the entrywise manner to its argument. We then define the *unweighted version* of the *degree matrix* $\tilde{D}(G)$ as the diagonal matrix with entries $\tilde{D}_{ii} = \sum_j \tilde{W}_{ij}$ and the the *unweighted (unnormalized) Laplacian matrix* as $\tilde{L}(G) := \tilde{D}(G) - \tilde{W}(G)$. Denoting $\tilde{\Phi} := [\tilde{\boldsymbol{\phi}}_0|\cdots|\tilde{\boldsymbol{\phi}}_{N-1}]$ and $\tilde{\Lambda} := \operatorname{diag}([\tilde{\lambda}_0,\ldots,\tilde{\lambda}_{N-1}])$, the eigendecomposition of $\tilde{L}(G)$ can be written as $\tilde{L}(G) = \tilde{\Phi}\tilde{\Lambda}\tilde{\Phi}^\mathsf{T}$. Since G is connected, $\tilde{\lambda}_0 = \lambda_0 = 0$ and $\tilde{\boldsymbol{\phi}}_0 = \boldsymbol{\phi}_0 = \mathbf{1}/\sqrt{N}$.

As an important example and for future reference, let us consider the Laplacian eigenpairs of an unweighted path graph P_N, which is also discussed earlier in [**35**,**60**,**71**,**73**,**74**]. In this case, the eigenvectors of $L(P_N)$ are exactly the *DCT Type II* basis vectors (used in the JPEG standard) [**82**]:

$$\lambda_l = \lambda_{l;N} := 4\sin^2\left(\frac{\pi l}{2N}\right),\ \ \boldsymbol{\phi}_l(x) = \boldsymbol{\phi}_{l;N}(x) := a_{l;N}\cos\left(\frac{\pi l}{N}\left(x-\frac{1}{2}\right)\right), \tag{2.2}$$

where $l = 0:N-1$, $x = 1:N$, and $a_{l;N}$ is a normalization constant to have $\|\boldsymbol{\phi}_{l;N}\|_2 = 1$. It is clear that the eigenvalue is a monotonically increasing function of the *frequency*, which is the eigenvalue index l divided by 2 in this case.

2.2. Graph Fourier Transform and Graph Wavelet Transforms

The Fourier transform is a classical tool in harmonic analysis. The graph Laplacian eigenvectors are often viewed as the generalized Fourier modes on graphs. Therefore, for any graph signal $\boldsymbol{f} \in \mathbb{R}^N$ and coefficient vector $\boldsymbol{g} \in \mathbb{R}^N$, the *graph Fourier transform* and *inverse graph Fourier transform* [**79**] are defined by

$$\mathscr{F}_G(\boldsymbol{f}) := \Phi^\mathsf{T}\cdot\boldsymbol{f} \in \mathbb{R}^N \quad \text{and} \quad \mathscr{F}_G^{-1}(\boldsymbol{g}) := \Phi\cdot\boldsymbol{g} \in \mathbb{R}^N. \tag{2.3}$$

Since Φ is an orthogonal matrix, it is not hard to see that $\mathscr{F}_G^{-1}\circ\mathscr{F}_G = I_N$. Thus, we can use $\mathscr{F}_G$ as an analysis operator and $\mathscr{F}_G^{-1}$ as a synthesis operator for graph harmonic analysis.

We now briefly review graph wavelet transforms and frames; see, e.g., [**62**, **79**] for more information. Translation and dilation are two important operators for classical wavelet construction. However, unlike $\mathbb{R}^d$ ($d \in \mathbb{N}$) or its finite and discretized lattice graph $P_{N_1} \times \cdots \times P_{N_d}$, we cannot assume the underlying graph has self-symmetric structure in general, i.e., its interior nodes may not always have the same neighborhood structure. Therefore, it is difficult to construct graph wavelet bases or frames by translating and dilating a single mother wavelet function of a fixed shape, e.g., the Mexican hat mother wavelet in $\mathbb{R}$, because the graph structure varies at different locations. Instead, some researchers, e.g., Hammond et al. [**29**], constructed wavelet frames by the Spectral Graph Wavelet Transform (SGWT), i.e., by shifting smooth graph spectral filters to be centered at different nodes. Let us take the SGWT frame as an example to summarize a general framework of building wavelet frames as follows:

$$\boldsymbol{\psi}_{j,x}^{\mathrm{SGWT}} := \overbrace{\Phi F_j \Phi^{\mathsf{T}}}^{\text{Filtering}} \boldsymbol{\delta}_x \quad \text{for } j = 0 : J \text{ and } x = 1 : N, \tag{2.4}$$

where the index j stands for different scale of spectral filtering (the greater j, the finer the scale, and $J \in \mathbb{N}$ represents the finest scale specified by the user), the index x represents the center location of the wavelet, $\boldsymbol{\delta}_x$ is the standard basis vector centered at node x, and the diagonal matrices $F_j \in \mathbb{R}^{N \times N}$ are the so-called *spectral graph filters* and usually defined by $(F_0)_{i,i} = h(\lambda_{i-1})$ and $(F_j)_{i,i} = g(s_j \lambda_{i-1})$ for $i = 1 : N$, $j = 1 : J$. Here, h is a scaling function (which mainly deals with the small eigenvalues), while g is a graph wavelet generating kernel. For example, the kernel proposed in [**29**] can be approximated by the Chebyshev polynomial and lead to a fast algorithm. Note that $\{s_j\}_{j=1:J}$ are dilation parameters.

Furthermore, one can show that as long as the generalized partition of unity

$$A \cdot I_N \leq \sum_{j=0}^{J} F_j \leq B \cdot I_N, \qquad 0 < A \leq B \tag{2.5}$$

holds, $\left\{\boldsymbol{\psi}_{j,x}^{\mathrm{SGWT}}\right\}_{j=0:J;x=1:N}$ forms a *graph wavelet frame*, which can be used to decompose and recover any given graph signals [**29**].

However, one important drawback of the above method is that the construction of the spectral filters F_j solely depends on the eigenvalue distribution (except some flexibility in choosing the filter pair (h, g), and the dilation parameters $\{s_j\}_{j=1:J}$) and does *not* reflect how the eigenvectors *behave*. For simple graphs such as P_N and C_N (a cycle graph with N nodes), the graph Laplacian eigenvectors are global

sinusoids whose frequencies can be simply read off from the corresponding eigenvalues, as discussed in Section 2.1. Hence, the usual *Littlewood-Paley wavelet theory* (see, e.g., [**17**, Sec. 4.2], [**37**, Sec. 2.4]) applies for those simple graphs. Unfortunately, the graph Laplacian eigenvectors of a general graph — even if it is ever so slightly more complicated than P_N and C_N — can behave in a much more complicated or unexpected manner than those of P_N or C_N, as discussed in [**10**, **35**, **51**, **60**, **71**, **72**, **73**, **74**] and Chapter 3.

2.3. Graph Wavelet Packets and Best-Basis Algorithm

Instead of building the graph wavelet dictionary by graph wavelet frames using spectral filters as summarized in Section 2.2, one could also accomplish it by generalizing the classical wavelet packets to graphs. The classical wavelet packet decomposition (or dictionary construction) of a 1D discrete signal is obtained by passing it through a full binary tree of filters (each node of the tree represents either low-pass filtered or high-pass filtered versions of the coefficients entering that node followed by the subsampling operation) to get a set of binary-tree-structured coefficients [**16**], [**54**, Sec. 8.1]. This basis dictionary for an input signal of length N has up to $N(1+\log_2 N)$ basis vectors (hence clearly redundant), yet contains more than 1.5^N searchable orthonormal bases (ONBs) [**16**, **85**].

For the purpose of efficient signal approximation, the *best-basis algorithm* originally proposed by Coifman and Wickerhauser [**16**] can find the most desirable ONB (and the expansion coefficients of the input signal) for a given task among such an immense number of ONBs. The best-basis algorithm requires a user-specified cost function, e.g., the ℓ^p-norm ($0 < p \leq 1$) of the expansion coefficients for sparse signal approximation, and the basis search starts at the bottom level of the dictionary and proceeds upwards, comparing the cost of the coefficients at the children nodes to the cost of the coefficients at their parents nodes. This best-basis search procedure only costs $O(N)$ operations provided that the expansion coefficients of the input signal have already been computed.

In order to generalize the classical wavelet packets to the graph setting, however, there are two main difficulties: 1) the concept of the frequency domain of a given graph is not well-defined (see Chapter 3 for the details); and 2) the relation between the Laplacian eigenvectors and sample locations are much more subtle on general graphs. For 1), we propose to construct a *dual graph*[2] $G^\star$ of the input graph G and view it as the natural spectral domain of G, and use any graph partition method to hierarchically

[2] Our definition of a dual graph is different from the standard definition in the graph theory; see Remark 5.1.1 for the details.

bipartition $G^\star$ instead of building low and high pass filters like the classical case. This can be viewed as the *generalized Littlewood-Paley theory*. For 2), we propose a node-eigenvector organization algorithm called the *pair-clustering algorithm*, which implicitly provides a downsampling process on graphs; see Chapter 7 for the details.

CHAPTER 3

Motivating Examples

In order to concretely demonstrate the problem of using eigenvalues to organize the corresponding eigenvectors, we examine the following two examples that were also discussed in [**10**, **51**, **72**]: the 2D lattice graph and the neuronal dendritic tree.

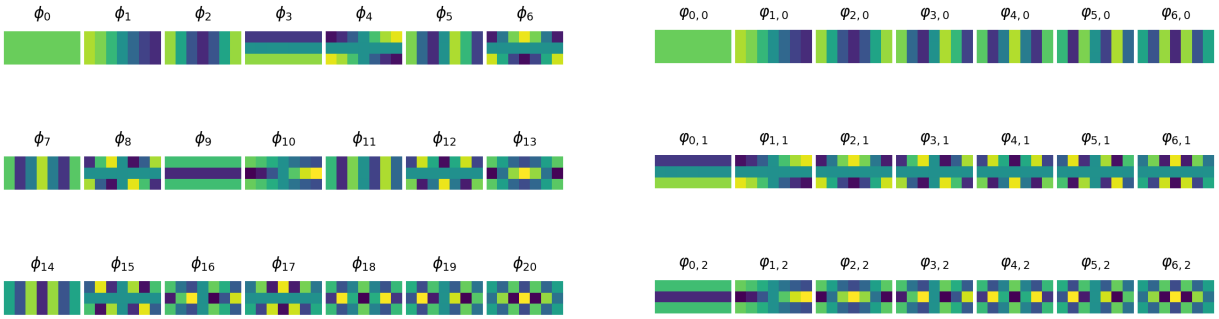
3.1. Lattice Graph

Let us consider a thin rectangle in $\mathbb{R}^2$, and suppose that this rectangle is discretized as $P_{N_x} \times P_{N_y}$ ($N_x > N_y > 1$). The Laplacian eigenpairs of this lattice graph can be easily derived from Eq. (2.2) as:

$$\lambda_l = \lambda_{(l_x,l_y)} := \lambda_{l_x;N_x} + \lambda_{l_y;N_y}$$

$$\phi_l(x,y) = \varphi_{l_x,l_y}(x,y) := \phi_{l_x;N_x}(x) \cdot \phi_{l_y;N_y}(y)$$

where $l = 0 : N_x N_y - 1$; $l_x = 0 : N_x - 1$, $l_y = 0 : N_y - 1$, $x = 1 : N_x$, and $y = 1 : N_y$. As always, let $\{\lambda_l\}_{l=0:N_x N_y-1}$ be ordered in the nondecreasing manner. Figure 3.1a shows the corresponding eigenvectors ordered in this manner (with $N_x = 7$, $N_y = 3$). Note that the layout of 3×7 grid of subplots is for the page saving purpose: the layout of 1×21 grid of subplots would be more natural if we use only the eigenvalue size for eigenvector ordering. For such a 2D lattice graph, the smallest eigenvalue is still $\lambda_0 = \lambda_{(0,0)} = 0$, and the corresponding eigenvector is constant. The second smallest eigenvalue λ_1 is $\lambda_{(1,0)} = 4\sin^2(\pi/2N_x)$, since $\pi/2N_x < \pi/2N_y$, and its eigenvector has one oscillation (i.e., half period) in the x-direction. But, how about λ_2? Even for such a simple situation there are two possibilities for λ_2, depending on N_x and N_y. If $N_x > 2N_y$, then $\lambda_2 = \lambda_{(2,0)} < \lambda_{(0,1)}$. On the other hand, if $N_y < N_x < 2N_y$, then $\lambda_2 = \lambda_{(0,1)} < \lambda_{(2,0)}$. More generally, if $KN_y < N_x < (K+1)N_y$ for some $K \in \mathbb{N}$, then $\lambda_l = \lambda_{(l,0)} = 4\sin^2(l\pi/2N_x)$ for $l = 0, \ldots, K$. Yet we have $\lambda_{K+1} = \lambda_{(0,1)} = 4\sin^2(\pi/2N_y)$ and λ_{K+2} is equal to either $\lambda_{(K+1,0)} = 4\sin^2((K+1)\pi/2N_x)$ or $\lambda_{(1,1)} = 4[\sin^2(\pi/2N_x) + \sin^2(\pi/2N_y)]$ depending on N_x and N_y. Clearly, the mapping between l and (l_x, l_y) is quite nontrivial, and moreover, the eigenpair computation does not tell us how to map from l



(a) Nondecreasing eigenvalue ordering

(b) Natural frequency ordering

FIGURE 3.1. Laplacian eigenvectors of $P_7 \times P_3$ ordered sequentially in terms of nondecreasing eigenvalues (a); those ordered in terms of their natural horizontal/vertical frequencies (b). The color scheme called *viridis* [**68**] is used to represent the amplitude of eigenvectors ranging from deep violet (negative) to teal (zero) to yellow (positive).

to (l_x, l_y). In Figure 3.1a, one can see this behavior with $K = 2$, i.e., notice that $\phi_2 (\equiv \varphi_{2,0})$ has two oscillation in the x-direction and no oscillation in the y-direction whereas $\phi_3 (\equiv \varphi_{0,1})$ has no oscillation in the x-direction and one oscillation in the y-direction. In other words, all of a sudden the eigenvalue of a completely different type of oscillation *sneaks into* the eigenvalue sequence. Hence, on a general graph, by simply looking at its Laplacian eigenvalue sequence $\{\lambda_l\}_{l=0,1,\ldots}$, it is *almost impossible to organize the eigenvectors into physically meaningful dyadic blocks and follow the Littlewood-Paley approach* unless the underlying graph is of very simple nature, e.g., P_N or C_N. Therefore, it will be problematic to design graph wavelets by using spectral filters built solely upon eigenvalues and we need to find a way to distinguish eigenvector behaviors.

What we really want to do is to *organize* those eigenvectors based on their natural frequencies or their behaviors, as shown in Figure 3.1b instead of Figure 3.1a, without explicitly knowing the mapping from l to (l_x, l_y) in this example.

3.2. Neuronal Dendritic Tree

Figure 3.2 shows a dendritic tree of a retinal ganglion cell (RGC) of a mouse, which is referred to as the RGC #100 (see [**73**] for the details of this RGC tree of a mouse). This graph $G(V, E, W)$, is in fact a tree (i.e., acyclic connected graph), and has $N = 1154$ nodes and $M = 1153$ edges, and its edge weights are assigned as the inverse of the Euclidean distance between two adjacent nodes. If we order the eigenvectors of its

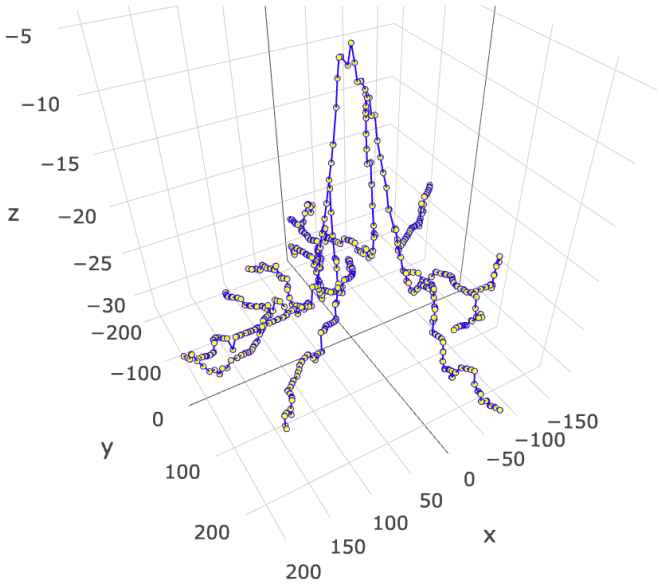


FIGURE 3.2. The dendritic tree of RGC #100 graph in $\mathbb{R}^3$.

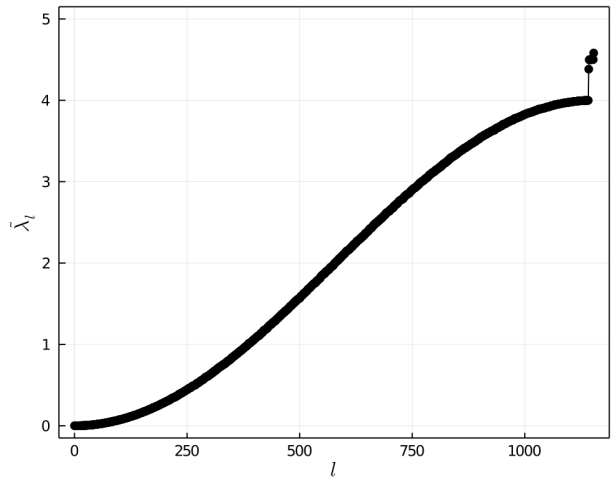
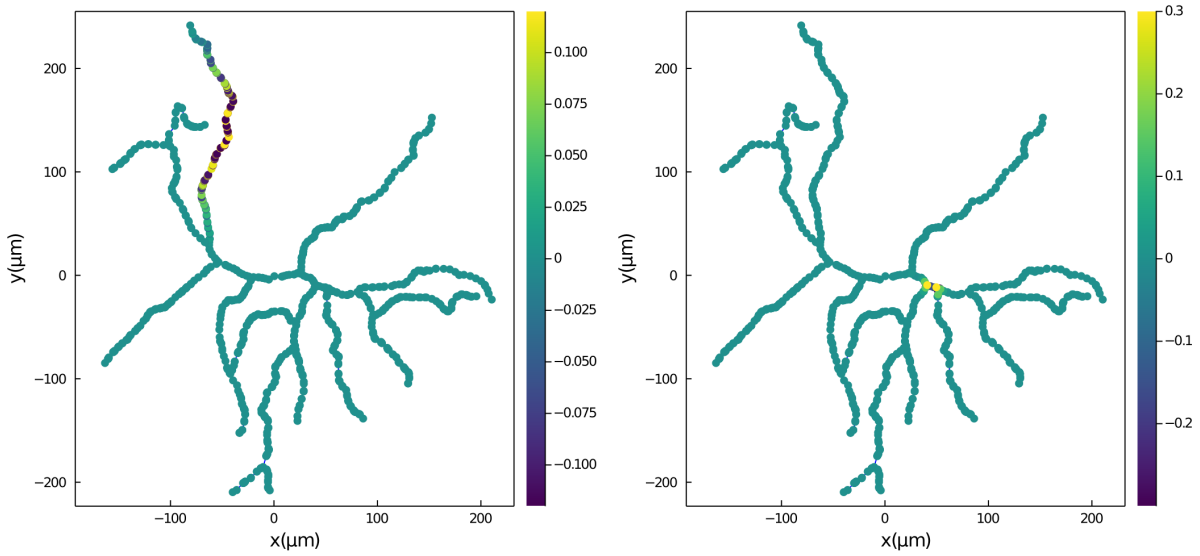


FIGURE 3.3. Eigenvalues of the *unweighted* graph Laplacian of the RGC #100.

unweighted graph Laplacian (i.e., $\tilde{L}(G)$) based on the size of corresponding eigenvalues, we encounter a peculiar phenomenon around $\tilde{\lambda} = 4$ [**73**].

The eigenvalue distribution, as shown in Figure 3.3, has a nice bell-shape curve starting at $\tilde{\lambda}_0 = 0$, and then around the eigenvalue 4, there is a sudden jump, followed by several eigenvalues greater than 4. As discussed in [**73**], this is a kind of *phase transition* phenomenon: the eigenvectors corresponding to



(a) $\tilde{\phi}_{1141}$ corresponding to $\tilde{\lambda}_{1141} = 3.9994$

(b) $\tilde{\phi}_{1142}$ corresponding to $\tilde{\lambda}_{1142} = 4.3829$

FIGURE 3.4. Two consecutive graph Laplacian eigenvectors of RGC #100 (2D plan view).

the eigenvalues below 4 are *semi-global oscillations* (like *Fourier cosines*) over the entire dendrites or one of the dendrite arbors (e.g., Figure 3.4a) while those corresponding to the eigenvalues above 4 are much more *localized* (like *wavelets*) around *junctions/bifurcation nodes* (e.g., Figure 3.4b).

Even though we can detect the above *phase transition* phenomenon by monitoring the eigenvalues, it is still impossible to tell how the behaviors of the eigenvectors change based solely on the eigenvalues. Therefore, it is important to define and compute quantitative similarity or difference between its eigenvectors for a general graph. Then, it is more natural to use these eigenvector metrics for the eigenvector organization instead of using the eigenvalues. However, we cannot use the usual ℓ^2-distances among them since they all have the same value $\sqrt{2}$ due to their orthonormality. So a natural question is: how can we *quantify the similarity/difference between the eigenvectors*?

CHAPTER 4

Metrics of Graph Laplacian Eigenvectors

To answer the question in the previous chapter, we introduce five non-trivial *behavioral* metrics of the graph Laplacian eigenvectors in a chronological order based on the time they are invented in this chapter. We also investigate the relationships among them.

4.1. Ramified Optimal Transport (ROT) Distance

The ramified optimal transport (ROT) distance [**72**] is a non-trivial eigenvector metric introduced by Saito to qualify the *behavioral difference* between the eigenvectors on graphs.

4.1.1. ROT Distance Between Probability Mass Functions (pmfs). The ROT theory [**89**] is a branch of the more general optimal transport theory [**44**, **65**]. It studies transporting the "mass" between two probability measures along the ramified transport paths, and has been used as a tool to analyze the optimal *branching* structures, e.g., the veins on a leaf, animal cardiovascular/circulatory systems, communication networks, etc. Given an *undirected* graph $G = (V, E, W)$, let $\mathrm{Path}(\boldsymbol{p}, \boldsymbol{q}) \subset G$ be all possible transport paths from one pmf $\boldsymbol{p}$ to another $\boldsymbol{q}$ without cycles, i.e., each $P \in \mathrm{Path}(\boldsymbol{p}, \boldsymbol{q})$ is a weighted acyclic *directed* graph whose edge weights (> 0) satisfy *the Kirchhoff law (or the balance equation)* at each node in $V(G)$.

$$\tilde{\tilde{Q}}\boldsymbol{\tau} = \boldsymbol{q} - \boldsymbol{p}, \qquad \boldsymbol{\tau} \in \mathbb{R}^{2M}_{\geq 0}, \tag{4.1}$$

where $\tilde{\tilde{Q}} = \tilde{Q}(\tilde{\tilde{G}}) := [\tilde{Q}(G)| - \tilde{Q}(G)] \in \mathbb{R}^{N \times 2M}$ is the *(unweighted) incidence matrix* of the *bidirected graph* $\tilde{\tilde{G}}$ generated from the undirected original graph G, i.e., each edge in $E(G)$ becomes two directed edges in $E(\tilde{\tilde{G}})$, so that the probability "mass" can move in either directions. The reason of using the *unweighted incidence matrix* instead of the weighted one is because at this step we only care about the possibilities of how the masses can be moved through edges in $E(\tilde{\tilde{G}})$, not how easy or hard they can be moved over different edges. Note that any $\boldsymbol{\tau}$ satisfying Eq. (4.1) represents a transportation path (or plan) from $\boldsymbol{p}$ to

$\boldsymbol{q}$, and there may be multiple solutions. Hence, we define the cost of a transport path $P \in \text{Path}(\boldsymbol{p}, \boldsymbol{q})$ as:

$$\boldsymbol{M}_\alpha(P) := \sum_{e \in E(P)} \tau(e)^\alpha \,\text{length}(e), \quad \alpha \in [0, 1].$$

where length(e) is the "length" of the edge $e \in E(P)$, which may be the Euclidean distance between the two nodes associated with e, or the *inverse* of the original edge weight of the input graph G if the original edge weight represents the affinity between those two nodes. And length(e) quantifies the difficulty of moving masses through e. Now, we can define the *minimum transportation cost*

$$d_{\text{ROT}}(\boldsymbol{p}, \boldsymbol{q}; \alpha) := \min_{P \in \text{Path}(\boldsymbol{p}, \boldsymbol{q})} \boldsymbol{M}_\alpha(P). \tag{4.2}$$

Xia proved in [**89**] that this is a *metric* on the space of pmfs and of homogeneous of degree α; and moreover he derived numerical algorithms to generate the α-optimal path for a given pair $(\boldsymbol{p}, \boldsymbol{q})$. In particular, when $\alpha = 1$, $d_{\text{ROT}}(\boldsymbol{p}, \boldsymbol{q}; \alpha = 1) = W_1(\boldsymbol{p}, \boldsymbol{q})$, where W_1 is the 1$^{\text{st}}$ Wasserstein distance [**65**] (a.k.a. the Earth Mover's Distance (EMD)) on graphs.

In practice, we propose to solve the following *Linear Programming* (LP) problem (see, e.g., [**63**]):

$$\min_{\boldsymbol{\tau} \in \mathbb{R}^{2M}_{\geq 0}} \|\boldsymbol{\tau}\|_1 \quad \text{subject to:} \quad \tilde{\tilde{Q}}\boldsymbol{\tau} = \boldsymbol{q} - \boldsymbol{p}, \tag{4.3}$$

to obtain one of the sparsest solutions of Eq. (4.1). We then compute the ROT distance by

$$d_{\text{ROT}}(\boldsymbol{p}, \boldsymbol{q}; \alpha) = \sum_{e \in E(P)} \tau(e)^\alpha \,\text{length}(e).$$

In our `NGWP.jl` package [**50**], we used the `JuMP.jl` optimization package [**19**] to solve Eq. (4.3).

4.1.2. ROT Distance Between Eigenvectors. There are two ways to define the *ROT distance* between the eigenvectors. The first way, as discussed in [**72**], is by: 1) converting the eigenvectors to pmfs based on a specific rule, denoted by pmf($\cdot$), and 2) computing *the minimum ROT cost* between the corresponding pmfs.

There are two methods that we typically use to convert an eigenvector $\boldsymbol{\phi}_l$ to a pmf. The first method is based on the fact that each eigenvector $\boldsymbol{\phi}_l$ has the unit length, we can turn it to a pmf by simply taking

entrywise squares:

$$\mathrm{pmf}^{(1)}\left(\phi_l\right)(x) := \phi_l(x)^2, \quad l = 0 : N-1 \text{ and } x = 1 : N.$$

Although this method is straightforward, we lose the sign information of the eigenvectors when taking squares. Another method inspired by [**21**] is to apply the *exponential function* on each entry followed by a normalization step, which carries the sign or phase information of an eigenvector:

$$\mathrm{pmf}^{(2)}\left(\phi_l\right)(x) := \frac{\exp(\phi_l(x))}{\sum_{y=1}^{N}\exp(\phi_l(y))}, \quad l = 0 : N-1 \text{ and } x = 1 : N. \tag{4.4}$$

After we convert the eigenvectors ϕ_i and ϕ_j to pmfs, we can define *their ROT distance* accordingly by

$$d_{\mathrm{ROT}}^{(1)}\left(\phi_i, \phi_j; \alpha\right) := d_{\mathrm{ROT}}\left(\mathrm{pmf}(\phi_i), \mathrm{pmf}(\phi_j); \alpha\right). \tag{4.5}$$

However, no matter which method we choose for the pmf conversion, we lose some information about the *behaviour* of the original eigenvectors. Fortunately, Eq. (4.2) is also well-defined for any vector measures $\boldsymbol{p}, \boldsymbol{q} \in \mathbb{R}^N$, as long as $\boldsymbol{p}$ and $\boldsymbol{q}$ have the same total mass, i.e., $\sum_{x=1}^{N} p(x) = \sum_{x=1}^{N} q(x)$. Considering $\sum_{x=1}^{N} \phi_l(x) = 0$ for $l = 1 : N-1$, so if we replace the DC vector ϕ_0 by $\mathbf{0} \in \mathbb{R}^N$, all the eigenvectors share the same total mass 0. Besides, ϕ_0 and $\mathbf{0}$ can be considered as two graph signals that behave the same on G, i.e., being flat or a constant. Therefore, we introduce the second way to define *the ROT distance* between eigenvectors as follows.

$$d_{\mathrm{ROT}}^{(2)}\left(\phi_i, \phi_j; \alpha\right) := \begin{cases} d_{\mathrm{ROT}}\left(\mathbf{0}, \phi_j; \alpha\right), & i = 0, \\ d_{\mathrm{ROT}}\left(\phi_i, \mathbf{0}; \alpha\right), & j = 0, \\ d_{\mathrm{ROT}}\left(\phi_i, \phi_j; \alpha\right), & i, j = 1 : N-1. \end{cases} \tag{4.6}$$

4.2. Simplified ROT (sROT) Distance for Trees

The high computational cost of solving Eq. (4.3) motivates us to speed up or simplify the ROT metric. If the underlying graph is a tree, we develop a computationally efficient simplified ROT (sROT) metric. Notice that there are only three types of nodes in a tree: terminal nodes (degree 1); internal nodes (degree 2); and junction nodes (degree greater than 2). When we consider using the pmfs of the eigenvectors in

$d^{(1)}_{\mathrm{ROT}}$ of Eq. (4.5) for the ordering and organization purposes, we are mainly analyzing how the probability masses are distributed on different branches (consisting of terminal and internal nodes) and junctions.

Therefore, let us decompose the tree into branches and junctions as follows. We first find all the junction nodes by checking the degree of each node. Then, we use the junction nodes to chop the tree into several branches and junctions (i.e., subgraphs). In particular, the junction nodes represent a bunch of one-node subgraphs. After this process, we get K disconnected subgraphs $G_k = (V_k, E_k)$ $(k = 0 : K-1)$ including all the branches and junctions of the tree. We also get a graph of these subgraphs, denoted as $G^s = (V^s, E^s)$, which describes how the subgraphs connected with each other. Specifically, $V^s = \{G_0, G_1, \ldots, G_{K-1}\}$ and edges in E^s connect V^s based on the connectivity between branches and junctions in the original tree. Moreover, one can use the centroid of nodes in V_k to represent the location of $G_k \in V^s$. Intuitively speaking, we compress the branches of G as one node of G^s, and G^s can be viewed as the simplified version of G. The above procedures form a further simplified version of the tree simplification algorithm introduced in [**75**]. An example of this G^s construction is given in Section 5.3.2.

Then, instead of computing the d_{ROT} directly between $\mathrm{pmf}(\phi_i)$ and $\mathrm{pmf}(\phi_j)$ on G, we add a preprocessing step to compress their probability masses into subgraphs. In other words, we compute the mass of $\mathrm{pmf}(\phi_l)$ $(l = 0 : N-1)$ in each subgraph $G_k = (V_k, E_k)$ $(k = 0 : K-1)$: 1) if G_k is a junction one-node subgraph (i.e., $V_k = \{v\}$), we just preserve the value of $\mathrm{pmf}(\phi_l)$ at node v; and 2) if G_k is a branch-type subgraph, we sum the mass of $\mathrm{pmf}(\phi_l)$ over the branch nodes V_k. We end up getting a K-dim vector $\boldsymbol{\theta}_l$ for each $\mathrm{pmf}(\phi_l)$, i.e.,

$$\theta_l(k) := \sum_{x \in V_k} \mathrm{pmf}(\phi_l)(x), \quad l = 0 : N-1 \text{ and } k = 0 : K-1.$$

Denote $\Theta := [\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_{N-1}] \in \mathbb{R}^{K \times N} (K \leq N)$. Note that each $\boldsymbol{\theta}_l$ can be also viewed as a K-dim pmf representation of ϕ_l. Thus, the computational cost of the ROT distance between $\boldsymbol{\theta}_i$ and $\boldsymbol{\theta}_j$ on G^s is significantly lower than the original ROT distance between $\mathrm{pmf}(\phi_i)$ and $\mathrm{pmf}(\phi_j)$ on G if the original tree contains long branches. Finally, the *sROT distance* between ϕ_i and ϕ_j is defined as follows.

$$d_{\mathrm{sROT}}(\phi_i, \phi_j; \alpha) := d_{\mathrm{ROT}}(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j; \alpha) \tag{4.7}$$

4.3. Hadamard (HAD) Product Affinity and Distance

The *Hadamard product affinity* [**10**] defines a similarity between the graph Laplacian eigenvectors in terms of average local correlations. Let ϕ_i, ϕ_j be eigenvectors of the graph Laplacian $L(G)$. It was shown in [**81**] that for all $x \in V$,

$$e^{-tL}(\phi_i \phi_j)(x) = \sum_{y \in V} p_t(x,y)(\phi_i(y) - \phi_i(x))(\phi_j(y) - \phi_j(x)),$$

where $\phi_i \phi_j$ is the Hadamard (i.e., entrywise) product of ϕ_i and ϕ_j, $p_t(x,y)$ is the (x,y)-th entry of e^{-tL}, and t satisfies

$$e^{-t\lambda_i} + e^{-t\lambda_j} = 1. \tag{4.8}$$

This implies that, rather than computing the local correlation between eigenvectors for all $x \in V$, one can simply compute the Hadamard product between the eigenvectors and apply the heat kernel matrix e^{-tL} at time t satisfying (4.8). When $e^{-tL}(\phi_i \phi_j)$ is similar in magnitude to $\phi_i \phi_j$, this implies the local correlation between ϕ_i and ϕ_j is large. This yields a natural similarity measure between eigenvectors

$$a_{\mathrm{HAD}}(\phi_i, \phi_j) := \begin{cases} 1, & \lambda_i = \lambda_j = 0, \\ \dfrac{\left\| e^{-tL}(\phi_i \phi_j) \right\|_2}{\|\phi_i \phi_j\|_2}, & \text{otherwise.} \end{cases} \tag{4.9}$$

Based on the above definition, it can be easily shown $0 \le a_{\mathrm{HAD}}(\phi_i, \phi_j) \le 1$, for $i, j = 0 : N-1$. For the case of $\lambda_i = \lambda_j = 0$, Eq. (4.8) does not hold and t is unsolvable, so we define $a_{\mathrm{HAD}}(\phi_0, \phi_0)$ independently by the maximum affinity value 1. Intuitively, a_{HAD} measures the fraction of $\phi_i \phi_j$ that projects into the graph Laplacian eigenvectors with smaller eigenvalues (i.e., more global low-frequency eigenvectors). We define the *Hadamard (HAD) product distance* by

$$d_{\mathrm{HAD}}(\phi_i, \phi_j) := \begin{cases} 0, & \lambda_i = \lambda_j, \\ \sqrt{1 - \left(a_{\mathrm{HAD}}(\phi_i, \phi_j)\right)^2}, & \lambda_i \neq \lambda_j. \end{cases} \tag{4.10}$$

We note that this "distance" does not satisfy the identity of discernible of the axioms of metric, yet it still quantifies the behavioral difference between the eigenvectors as shown in [**10**] and Section 5.3.1.

4.4. Difference of Absolute Gradient (DAG) Distance

Another effective way to measure "behavioral" difference between the eigenvectors is by the so-called *Difference of Absolute Gradient (DAG) pseudometric* [**51**]. The advantages of the DAG pseudometric are: 1) its computational efficiency compared to the other eigenvector metrics; 2) its superior performance for grid-like graphs; and 3) its close relationship with the Hadamard-product affinity proposed in [**10**]. This DAG distance is introduced as below.

Instead of the usual ℓ^2-distance, we use the *absolute gradient* of each eigenvector as its feature vector describing its behavior. More precisely, let $Q(G) \in \mathbb{R}^{N\times M}$ be the *incidence matrix* of an input graph $G(V,E,W)$. Based on the definition of $Q(G)$ in Section 2.1, it is easy to see that $Q(G)\,Q(G)^{\mathsf{T}} = L(G)$.

We now define the *DAG distance* between ϕ_i and ϕ_j by

$$d_{\mathrm{DAG}}(\phi_i,\phi_j) := \left\| |\nabla_G|\phi_i - |\nabla_G|\phi_j \right\|_2 \quad \text{where } |\nabla_G|\phi := \mathrm{abs}.\left(Q(G)^{\mathsf{T}}\phi\right) \in \mathbb{R}^M_{\geq 0}, \tag{4.11}$$

where $\mathrm{abs}.(\cdot)$ applies the absolute value in the entrywise manner to its argument. We note that $|\nabla_G|\phi$, the absolute gradient of an eigenvector ϕ, is invariant with respect to: 1) sign flip, i.e., $|\nabla_G|\phi \equiv |\nabla_G|(-\phi)$ and 2) choice of sign of each column (i.e., edge orientation) of the incidence matrix $Q(G)$. We also note that this quantity is not a metric but a *pseudometric* because the identity of discernible of the axioms of metric is not satisfied. In order to see the meaning of this quantity, let us analyze its square as follows.

$$\begin{aligned} d_{\mathrm{DAG}}(\phi_i,\phi_j)^2 &= \left\langle |\nabla_G|\phi_i - |\nabla_G|\phi_j, |\nabla_G|\phi_i - |\nabla_G|\phi_j \right\rangle_E \\ &= \langle |\nabla_G|\phi_i, |\nabla_G|\phi_i\rangle_E + \left\langle |\nabla_G|\phi_j, |\nabla_G|\phi_j \right\rangle_E - 2\left\langle |\nabla_G|\phi_i, |\nabla_G|\phi_j \right\rangle_E \\ &= \lambda_i + \lambda_j - \sum_{x\in V}\sum_{y\sim x} |\phi_i(x)-\phi_i(y)|\cdot|\phi_j(x)-\phi_j(y)| \quad \text{thanks to } Q(G)\,Q(G)^{\mathsf{T}} = L(G) \end{aligned} \tag{4.12}$$

where $\langle\cdot,\cdot\rangle_E$ is the inner product over edges. The last term of the formula can be viewed as *a global average of absolute local correlation* between eigenvectors. In this sense, this quantity is related to the *Hadamard-product affinity* between eigenvectors discussed in Section 4.3. Note that the computational cost is $O(M)$ for each $d_{\mathrm{DAG}}(\cdot,\cdot)$ evaluation provided that the eigenvectors have already been computed.

4.5. Time-Step Diffusion (TSD) Distance

4.5.1. TSD Metric on graphs. Given $G = (V, E, W)$, consider the governing heat diffusion ODE system[1] on the graph, which describes the evolution of the graph signal $\boldsymbol{f}_0 \in \mathbb{R}^N$:

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{f}(t) + \tilde{L}(G)\cdot\boldsymbol{f}(t) = \mathbf{0} \quad t \geq 0, \qquad \boldsymbol{f}(0) = \boldsymbol{f}_0 \in \mathbb{R}^N. \tag{4.13}$$

Since the *(unweighted) graph Laplacian* (i.e., $\tilde{L}(G)$) eigenvectors $\{\tilde{\boldsymbol{\phi}}_0, \tilde{\boldsymbol{\phi}}_1, \cdots, \tilde{\boldsymbol{\phi}}_{N-1}\}$ form an ONB of $\mathbb{R}^N$, we have:

$$\boldsymbol{f}(t) = \sum_{l=0}^{N-1} \langle \boldsymbol{f}_0, \tilde{\boldsymbol{\phi}}_l \rangle \mathrm{e}^{-\tilde{\lambda}_l t}\tilde{\boldsymbol{\phi}}_l. \tag{4.14}$$

At a time $T > 0$, we define the following TSD functional:

$$K_{\mathrm{TSD}}(\boldsymbol{f}_0, T) := \int_0^T \|\tilde{\nabla}_G \boldsymbol{f}(t)\|_{1,\mathrm{w}}\mathrm{d}t, \tag{4.15}$$

where $\tilde{\nabla}_G \boldsymbol{f} = \tilde{Q}(G)^{\mathsf{T}}\boldsymbol{f} \in \mathbb{R}^M$ is the *(unweighted) graph gradient* of $\boldsymbol{f}$ and $\|\cdot\|_{1,\mathrm{w}} : \mathbb{R}^M \mapsto \mathbb{R}_{\geq 0}$ is a weighted ℓ^1-norm,

$$\|\boldsymbol{x}\|_{1,\mathrm{w}} = \sum_{k=1}^{M} |x(k)| \cdot \mathrm{length}(e_k), \quad \text{for } \boldsymbol{x} \in \mathbb{R}^M,$$

in which $\mathrm{length}(e_k)$ is the "length" of the kth edge in $E(G)$, which may be the Euclidean distance between the two nodes associated with e_k, or the *inverse* of the original edge weight of the input graph G if the original edge weight represents the affinity between those two nodes. This functional can be viewed as the cost (or effort) to "flatten" the initial graph signal $\boldsymbol{f}_0$ via diffusion process up to the time T, and also as the time-accumulated "anisotropic total variation norm" [**30**] of $\boldsymbol{f}_0$. Clearly, $K_{\mathrm{TSD}}(\boldsymbol{f}_0, T)$ is an increasing

[1] Here we use the *unweighted version* of the graph Laplacian matrix $\tilde{L}(G)$ to study the heat diffusion based solely on the connectivity of the graph. Our reasoning is the same as in the case when we considered the ROT balance equation Eq. (4.1).

function w.r.t. T. Moreover, we can show that $\lim_{T\to\infty} K_{\mathrm{TSD}}(\boldsymbol{f}_0, T) < \infty$ for any $\boldsymbol{f}_0 \in \mathbb{R}^N$:

$$\begin{aligned}
\lim_{T\to\infty} K_{\mathrm{TSD}}(\boldsymbol{f}_0, T) &= \int_0^\infty \left\| \tilde{\nabla}_G \sum_{l=0}^{N-1} \langle \boldsymbol{f}_0, \tilde{\boldsymbol{\phi}}_l \rangle e^{-\tilde{\lambda}_l t} \tilde{\boldsymbol{\phi}}_l \right\|_{1,\mathrm{w}} \mathrm{d}t \\
&\le \int_0^\infty \sum_{l=0}^{N-1} \left\| \langle \boldsymbol{f}_0, \tilde{\boldsymbol{\phi}}_l \rangle e^{-\tilde{\lambda}_l t} \tilde{\nabla}_G \tilde{\boldsymbol{\phi}}_l \right\|_{1,\mathrm{w}} \mathrm{d}t \\
&= \int_0^\infty \sum_{l=1}^{N-1} \left\| \langle \boldsymbol{f}_0, \tilde{\boldsymbol{\phi}}_l \rangle e^{-\tilde{\lambda}_l t} \tilde{\nabla}_G \tilde{\boldsymbol{\phi}}_l \right\|_{1,\mathrm{w}} \mathrm{d}t \quad \text{thanks to } \tilde{\nabla}_G \tilde{\boldsymbol{\phi}}_0 = \mathbf{0} \\
&= \sum_{l=1}^{N-1} \int_0^\infty \left\| \langle \boldsymbol{f}_0, \tilde{\boldsymbol{\phi}}_l \rangle e^{-\tilde{\lambda}_l t} \tilde{\nabla}_G \tilde{\boldsymbol{\phi}}_l \right\|_{1,\mathrm{w}} \mathrm{d}t \\
&= \sum_{l=1}^{N-1} \left\| \langle \boldsymbol{f}_0, \tilde{\boldsymbol{\phi}}_l \rangle \tilde{\nabla}_G \tilde{\boldsymbol{\phi}}_l \right\|_{1,\mathrm{w}} \int_0^\infty e^{-\tilde{\lambda}_l t} \mathrm{d}t \\
&= \sum_{l=1}^{N-1} \frac{|\langle \boldsymbol{f}_0, \tilde{\boldsymbol{\phi}}_l \rangle|}{\tilde{\lambda}_l} \left\| \tilde{\nabla}_G \tilde{\boldsymbol{\phi}}_l \right\|_{1,\mathrm{w}} < \infty.
\end{aligned}$$

Now, $K_{\mathrm{TSD}}(\cdot, T)$ is well defined for $T \in \mathbb{R}_{>0} \cup \{\infty\}$. With this in place, we can further show the following lemma.

LEMMA 4.5.1. *Given $G(V,E)$ and $T \in \mathbb{R}_{>0} \cup \{\infty\}$, $K_{\mathrm{TSD}}(\cdot, T)$ is a norm on $\mathbb{R}_0^N := \{\boldsymbol{f} \in \mathbb{R}^N | \mathbf{1}^\top \boldsymbol{f} = 0\}$, i.e., $\left(\mathbb{R}_0^N, K_{\mathrm{TSD}}(\cdot, T)\right)$ is a normed vector space. Thus, we get a metric vector space $\left(\mathbb{R}_0^N, d_{\mathrm{TSD}}(\cdot,\cdot; T)\right)$ by defining*

$$d_{\mathrm{TSD}}(\boldsymbol{f}, \boldsymbol{g}; T) := K_{\mathrm{TSD}}(\boldsymbol{g} - \boldsymbol{f}, T), \qquad \boldsymbol{f}, \boldsymbol{g} \in \mathbb{R}_0^N.$$

See Appendix A.1 for the proof.

After setting the input signal $\boldsymbol{f}_0 = \boldsymbol{\phi}_j - \boldsymbol{\phi}_i$, we can define the *TSD distance* between eigenvectors[2] at time $T \in \mathbb{R}_{>0} \cup \{\infty\}$ by

$$d_{\mathrm{TSD}}(\boldsymbol{\phi}_i, \boldsymbol{\phi}_j; T) := K_{\mathrm{TSD}}(\boldsymbol{f}_0, T). \tag{4.16}$$

In practice, we used the `QuadGK.jl` [**39**] to compute the numerical integration of Eq. (4.15) and consequently the approximated TSD distance in Eq. (4.16).

[2] Since $\boldsymbol{\phi}_0 \notin \mathbb{R}_0^N$, $\boldsymbol{\phi}_0$ is replaced by $\mathbf{0} \in \mathbb{R}_0^N$ when computing the TSD distance.

4.5.2. TSD metric on a compact Riemannian manifold $\mathscr{M}$. We can also define the TSD metric on a compact Riemannian manifold $\mathscr{M}$ where the heat diffusion system in Eq. (4.14) can be defined based on the Laplace-Beltrami operator Δ.

We consider the heat diffusion on $\mathscr{M}$ with Neumann boundary conditions

$$\begin{cases} \frac{\partial}{\partial t} f(x,t) = \Delta f(x,t), \quad x \in \mathscr{M}, t \geq 0 \\ \frac{\partial}{\partial \boldsymbol{n}} f(x,t) = 0, \quad x \in \partial\mathscr{M}, t \geq 0 \\ f(x,0) = f_0, \end{cases} \tag{4.17}$$

where $\boldsymbol{n}$ is the normal direction at boundary $\partial\mathscr{M}$ and the initial signal

$$f_0 \in L_0^2(\mathscr{M}) := \left\{ f \in L^2(\mathscr{M}) \,|\, \int_{\mathscr{M}} f(x) \mathrm{d}\mu(x) = 0 \right\}.$$

Then, the TSD functional on $\mathscr{M}$ can be defined by:

$$K_{\mathrm{TSD}}(f_0, T) := \int_0^T \int_{\mathscr{M}} |\nabla_x f(x,t)| \mathrm{d}\mu(x) \mathrm{d}t,$$

where $\mathrm{d}\mu$ is the measure on $\mathscr{M}$.

We denote the eigenfunctions of Δ in Eq. (4.17) as ϕ_l with $\|\phi_l\|_2 = 1$, $l \in \{0\} \cup \mathbb{N}$. The following theorem shows an upper bound of this functional on $\mathscr{M}$.

THEOREM 4.5.2. *For $T \in \mathbb{R}^N_{>0} \cup \{\infty\}$ and $f_0 \in L_0^2(\mathscr{M})$,*

$$K_{\mathrm{TSD}}(f_0, T) \leq \sum_{l=1}^{\infty} \frac{1}{\sqrt{\lambda_l}} |\hat{f}_0(l)| \cdot \sqrt{\mathrm{Vol}(\mathscr{M})},$$

where λ_l's are the positive eigenvalues of Laplace-Beltrami operator, $\hat{f}_0(l) := \langle \phi_l, f_0 \rangle$ are the Fourier coefficients of f_0 and $\mathrm{Vol}(\mathscr{M}) := \int_{\mathscr{M}} \mathrm{d}\mu(x)$.

See Appendix A.2 for the proof.

Therefore, *the TSD distance with parameter T* on $L_0^2(\mathscr{M})$ between eigenfunctions[3] ϕ_i and ϕ_j is well defined by

$$d_{\mathrm{TSD}}(\phi_i,\phi_j;T) := K_{\mathrm{TSD}}(\phi_j-\phi_i,T) \le \left(\frac{1}{\sqrt{\lambda_i}}+\frac{1}{\sqrt{\lambda_j}}\right)\sqrt{\mathrm{Vol}(\mathscr{M})} < \infty.$$

4.6. Relationship Between ROT and TSD

The motivation of designing the TSD metric is to construct a time-evolving optimal transport-like metric. Actually, we have the following theorem for $T=\infty$.

THEOREM 4.6.1. *Given a graph $G=G(V,E,W)$ and two associated vector measures $\boldsymbol{p},\boldsymbol{q}$ with the same total mass, i.e., $\sum_{x=1}^N p(x)=\sum_{x=1}^N q(x)$, we have*

$$d_{\mathrm{ROT}}(\boldsymbol{p},\boldsymbol{q};\alpha=1) \le K_{\mathrm{TSD}}(\boldsymbol{q}-\boldsymbol{p},T=\infty) \le C(G)\cdot d_{\mathrm{ROT}}(\boldsymbol{p},\boldsymbol{q};\alpha=1),$$

$$\textit{i.e.,}\quad W_1(\boldsymbol{p},\boldsymbol{q}) \le K_{\mathrm{TSD}}(\boldsymbol{q}-\boldsymbol{p},T=\infty) \le C(G)\cdot W_1(\boldsymbol{p},\boldsymbol{q}), \tag{4.18}$$

where $C(G)$ is a constant depending on the graph G.

The following lemma proves the first inequality in Eq. (4.18).

LEMMA 4.6.2. *Given $G(V,E)$ and two vector measures $\boldsymbol{p},\boldsymbol{q}\in\mathbb{R}^N$ with the same total mass, i.e.,*

$$\sum_{x=1}^N p(x) = \sum_{x=1}^N q(x).$$

The following inequality holds

$$W_1(\boldsymbol{p},\boldsymbol{q}) \le K_{\mathrm{TSD}}(\boldsymbol{q}-\boldsymbol{p},T=\infty).$$

PROOF. If we can show that K_{TSD} with $T=\infty$ provides a transport plan $\boldsymbol{\tau}_{\mathrm{TSD}}$ that satisfies the balance equation Eq. (4.1), then the above inequality holds automatically based on the definition of the ROT cost

[3] except the DC component ϕ_0

in Eq. (4.2). In other words, we want to find a $\boldsymbol{\tau}_{\mathrm{TSD}} \in \mathbb{R}^{2M}_{\geq 0}$ such that

$$\tilde{\tilde{Q}}\,\boldsymbol{\tau}_{\mathrm{TSD}} = \boldsymbol{q} - \boldsymbol{p},$$

where $\tilde{\tilde{Q}} = [\tilde{Q}|-\tilde{Q}] = [\tilde{\tilde{\boldsymbol{q}}}_1|\cdots|\tilde{\tilde{\boldsymbol{q}}}_{2M}] \in \mathbb{R}^{N\times 2M}$ is the *unweighted incidence matrix* of the *bidirected graph* $\tilde{\tilde{G}}$[4], and $\tilde{Q} = \tilde{Q}(G)$ for short. Based on the definition of K_{TSD}, we perform the following derivations.

$$\begin{aligned}
K_{\mathrm{TSD}}(\boldsymbol{q}-\boldsymbol{p}, T=\infty) &= \int_0^\infty \|\tilde{Q}^{\mathsf{T}}\cdot \boldsymbol{f}(t)\|_{1,\mathrm{w}}\mathrm{d}t \\
&= \sum_{k=1}^{M}\left(\int_0^\infty |\tilde{\boldsymbol{q}}_k^{\mathsf{T}}\cdot\boldsymbol{f}(t)|\mathrm{d}t\right)\cdot \mathrm{length}(e_k) \\
&\geq \sum_{k=1}^{M}\left|\int_0^\infty \tilde{\boldsymbol{q}}_k^{\mathsf{T}}\cdot\boldsymbol{f}(t)\mathrm{d}t\right|\cdot \mathrm{length}(e_k) \\
&= \sum_{k=1}^{2M}\left(\int_0^\infty \tilde{\tilde{\boldsymbol{q}}}_k^{\mathsf{T}}\cdot\boldsymbol{f}(t)\mathrm{d}t\right)_+\cdot \mathrm{length}(\tilde{\tilde{e}}_k),
\end{aligned}$$

where $(\cdot)_+$ turns negative values to zero, e_k is the k-th edge in $E(G)$ and $\tilde{\tilde{e}}_k$ is the k-th edge in $E(\tilde{\tilde{G}})$. Then, a natural guess for $\boldsymbol{\tau}_{\mathrm{TSD}}$ would be

$$\tau_{\mathrm{TSD}}(k) := \left(\int_0^\infty \tilde{\tilde{\boldsymbol{q}}}_k^{\mathsf{T}}\cdot\boldsymbol{f}(t)\mathrm{d}t\right)_+, \quad \text{for } k = 1,2,\ldots,2M.$$

Now, let us verify the defined $\boldsymbol{\tau}_{\mathrm{TSD}}$ satisfies the balance equation Eq. (4.1).

$$\tilde{\tilde{Q}}\,\boldsymbol{\tau}_{\mathrm{TSD}} = \sum_{k=1}^{2M}\tilde{\tilde{\boldsymbol{q}}}_k\cdot\left(\int_0^\infty \tilde{\tilde{\boldsymbol{q}}}_k^{\mathsf{T}}\cdot\boldsymbol{f}(t)\mathrm{d}t\right)_+ \overset{\text{(a)}}{=} \int_0^\infty \tilde{L}(G)\cdot\boldsymbol{f}(t)\mathrm{d}t = \int_0^\infty -\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{f}(t)\mathrm{d}t = \boldsymbol{f}(0)-\boldsymbol{f}(\infty) \overset{\text{(b)}}{=} \boldsymbol{f}(0) = \boldsymbol{q}-\boldsymbol{p}.$$

The equality (a) holds is because without loss of generality, we can assume $\int_0^\infty \tilde{\tilde{\boldsymbol{q}}}_k^{\mathsf{T}}\cdot\boldsymbol{f}(t)\mathrm{d}t \geq 0$ for $k = 1:M$, which implies $\int_0^\infty \tilde{\tilde{\boldsymbol{q}}}_k^{\mathsf{T}}\cdot\boldsymbol{f}(t)\mathrm{d}t = -\int_0^\infty \tilde{\tilde{\boldsymbol{q}}}_{k-M}^{\mathsf{T}}\cdot\boldsymbol{f}(t)\mathrm{d}t \leq 0$ for $k = M+1:2M$. Therefore,

$$\sum_{k=1}^{2M}\tilde{\tilde{\boldsymbol{q}}}_k\cdot\left(\int_0^\infty \tilde{\tilde{\boldsymbol{q}}}_k^{\mathsf{T}}\cdot\boldsymbol{f}(t)\mathrm{d}t\right)_+ = \sum_{k=1}^{M}\tilde{\tilde{\boldsymbol{q}}}_k\cdot\int_0^\infty \tilde{\tilde{\boldsymbol{q}}}_k^{\mathsf{T}}\cdot\boldsymbol{f}(t)\mathrm{d}t = \int_0^\infty \tilde{Q}\tilde{Q}^{\mathsf{T}}\cdot\boldsymbol{f}(t)\mathrm{d}t = \int_0^\infty \tilde{L}(G)\cdot\boldsymbol{f}(t)\mathrm{d}t.$$

The equality (b) holds is because $\boldsymbol{p}$ and $\boldsymbol{q}$ have the same total mass such that $\langle\tilde{\phi}_0, \boldsymbol{q}-\boldsymbol{p}\rangle = 0$, which implies $\boldsymbol{f}(\infty) = \boldsymbol{0} \in \mathbb{R}^N$. Hence, $\boldsymbol{\tau}_{\mathrm{TSD}}$ represents a transportation plan on $\tilde{\tilde{G}}$ and the inequality holds. □

On the other hand, the following lemma shows the second inequality of Eq. (4.18).

[4] which is defined under Eq. (4.1).

LEMMA 4.6.3. *Given a finite graph $G(V,E)$ and two vector measures $\boldsymbol{p},\boldsymbol{q} \in \mathbb{R}^N$ with the same total mass, i.e.,*

$$\sum_{x=1}^{N} p(x) = \sum_{x=1}^{N} q(x).$$

The following inequality holds

$$K_{\mathrm{TSD}}(\boldsymbol{q}-\boldsymbol{p}, T=\infty) \le C(G) \cdot \|\boldsymbol{q}-\boldsymbol{p}\|_{W_1},$$

where $\|\boldsymbol{q}-\boldsymbol{p}\|_{W_1} := W_1(\boldsymbol{p},\boldsymbol{q})$.

This lemma can be proved by the well-known fact: *all norms are equivalent in finite vector spaces*. With that, we concluded our proof of Theorem 4.6.1. Note that how one can properly estimate the lower bound of the constant $C(G)$ based on the properties of G itself remains as a question.

Furthermore, the underlying domain in Eq. (4.18) could also be a compact Riemannian manifold $\mathscr{M}$ instead of a graph. Unfortunately, the optimal transportation plan of W_1 on $\mathscr{M}$ generally cannot be expressed in an explicit form [**49**], so it is hard to prove the inequalities in Eq. (4.18). However, if the underlying manifold is $\mathscr{M} = [0,2\pi]$ and input measures are probability density functions, (where the explicit expression of W_1 is known [**49**]), we can show the first inequality of Eq. (4.18) on $\mathscr{M}$ by the following theorem.

THEOREM 4.6.4. *Given two probability density functions p, q on $[0,2\pi]$,*

$$W_1(p,q) \le K_{\mathrm{TSD}}(q-p, T=\infty).$$

See Appendix A.3 for the proof.

Another perspective to study the inequalities in Eq. (4.18) is by numerical simulations. For convenience, we define the ratio ρ between $K_{\mathrm{TSD}}(\boldsymbol{q}-\boldsymbol{p}, T=\infty)$ and $W_1(\boldsymbol{p},\boldsymbol{q})$ as

$$\rho(\boldsymbol{p},\boldsymbol{q}) := \begin{cases} \dfrac{K_{\mathrm{TSD}}(\boldsymbol{q}-\boldsymbol{p}, T=\infty)}{W_1(\boldsymbol{p},\boldsymbol{q})}, & \text{if } \boldsymbol{p} \neq \boldsymbol{q}, \\ 1, & \text{if } \boldsymbol{p} = \boldsymbol{q}. \end{cases}$$

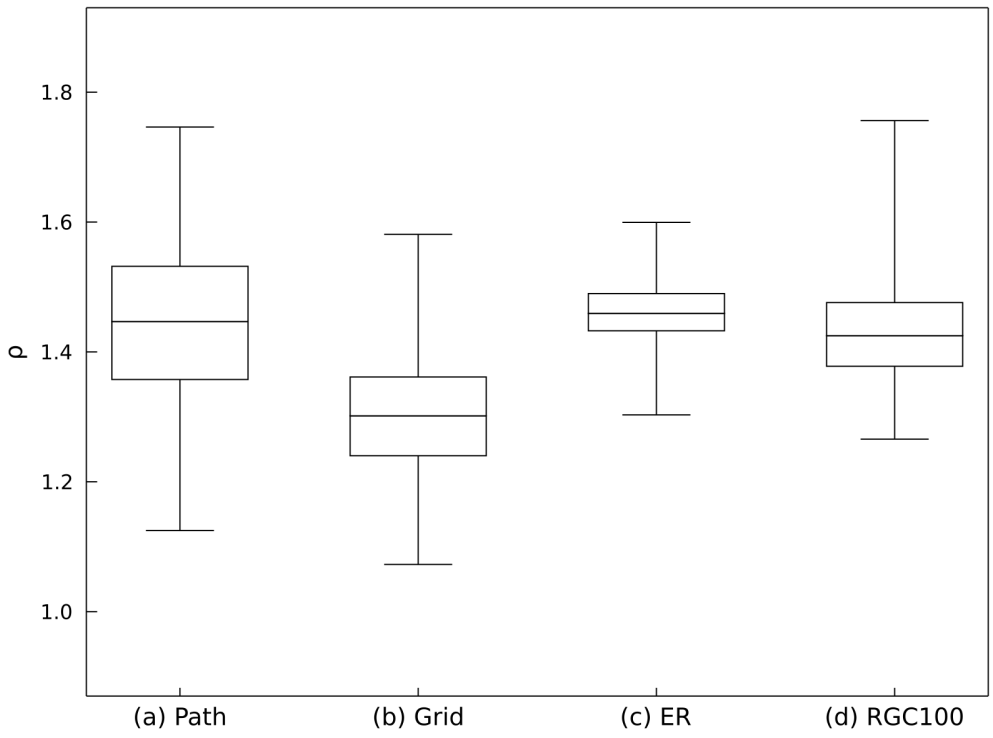


FIGURE 4.1. Boxplots of ρ in four different graphs.

The numerical experiments are performed on four different graphs: (a) the *unweighted* path graph P_{64} ($N = 64,\ M = 63$); (b) the *unweighted* grid lattice graph $P_7 \times P_3$ ($N = 21,\ M = 32$), as seen in Figure 3.1; (c) an *unweighted* connected Erdős Rényi (ER) random graph [**22**] ($N = 50,\ M = 200$); and (d) the *weighted* RGC #100 graph ($N = 1154,\ M = 1153$), as seen in Figure 3.2.

We create 500 pairs of pmfs $(\boldsymbol{p}, \boldsymbol{q})$ randomly on each $G(V,E)$: 1) sample each entry in $\boldsymbol{p}, \boldsymbol{q}$ independently from the standard uniform distribution $U(0,1)$, i.e., $p(x), q(x) \overset{i.i.d.}{\sim} U(0,1)$ for $x = 1:N$. 2) normalize the obtained vectors $\boldsymbol{p}$ and $\boldsymbol{q}$ respectively such that $\sum_{x=1}^{N} p(x) = \sum_{x=1}^{N} q(x) = 1$.

Finally, the ratio ρ is computed on each graph 500 times and the four corresponding boxplots are displayed in Figure 4.1. We see that $\rho \geq 1$ for all four graphs no matter if the graph is *weighted or unweighted*, which verifies the result in Lemma 4.6.2. Based on the numerical simulation results, the estimated lower bound of $C(G)$, i.e., the max of ρ, equals to: (a) 1.7463 for P_{64}; (b) 1.5810 for $P_7 \times P_3$; (c) 1.5995 for the Erdős Rényi random graph; and (d) 1.7562 for the *weighted* RGC #100 graph.

4.7. Relationship Between DAG and TSD

Beside the relationship between ROT and TSD, the TSD metric is also related to the *DAG pseudometric*. Given an *unweighted* graph $G = G(V,E)$ and a *small* $T > 0$, we have the following approximation of d_{TSD} based on the continuity of the function inside the integral of Eq. (4.15).

$$d_{\mathrm{TSD}}(\boldsymbol{\phi}_i, \boldsymbol{\phi}_j; T) = K_{\mathrm{TSD}}(\boldsymbol{\phi}_j - \boldsymbol{\phi}_i, T) \approx T \left\| \nabla_G (\boldsymbol{\phi}_j - \boldsymbol{\phi}_i) \right\|_1 = T \left\| \nabla_G \boldsymbol{\phi}_i - \nabla_G \boldsymbol{\phi}_j \right\|_1, \quad T = o(1),$$

where we also used the facts $\tilde{\nabla}_G \equiv \nabla_G$ and $\|\cdot\|_{1,\mathrm{w}} \equiv \|\cdot\|_1$ when G is unweighted. Moreover, this approximated quantity is related to d_{DAG}[5] by the following inequality.

$$d_{\mathrm{DAG}}(\boldsymbol{\phi}_i, \boldsymbol{\phi}_j) = \left\| |\nabla_G| \boldsymbol{\phi}_i - |\nabla_G| \boldsymbol{\phi}_j \right\|_2 \le \left\| \nabla_G \boldsymbol{\phi}_i - \nabla_G \boldsymbol{\phi}_j \right\|_2 \le \left\| \nabla_G \boldsymbol{\phi}_i - \nabla_G \boldsymbol{\phi}_j \right\|_1.$$

The last inequality holds is because for any $\boldsymbol{f} \in \mathbb{R}^N$,

$$\|\boldsymbol{f}\|_1^2 = \big(|f(1)| + |f(2)| + \cdots + |f(N)|\big)^2 \ge f(1)^2 + f(2)^2 + \cdots + f(N)^2 = \|\boldsymbol{f}\|_2^2.$$

Therefore,

$$d_{\mathrm{TSD}}(\boldsymbol{\phi}_i, \boldsymbol{\phi}_j; T) \gtrsim T \cdot d_{\mathrm{DAG}}(\boldsymbol{\phi}_i, \boldsymbol{\phi}_j), \quad T = o(1). \tag{4.19}$$

Hence, the TSD metric in some sense bridges the DAG *pseudometric* with $T = o(1)$ and the ROT *metric* with $T = \infty$ on *unweighted* graphs.

[5] Note that $\boldsymbol{\phi}_0$ is replaced by $\mathbf{0} \in \mathbb{R}^N$ before feeding into the TSD metric, but this does not have any effect on the DAG pseudometric since $|\nabla_G| \boldsymbol{\phi}_0 = |\nabla_G| \mathbf{0} = \mathbf{0} \in \mathbb{R}^M$.

CHAPTER 5

Natural Organization of Graph Laplacian Eigenvectors

The metrics introduced in the previous chapter are summarized in Table 5.1. With these non-trivial metrics in place, the next questions are: 1) how can we *use these metrics to naturally organize the eigenvectors*? 2) how can we *visualize the organization in a low dimensional Euclidean space, i.e.,* $\mathbb{R}^2$ *or* $\mathbb{R}^3$? In this chapter, we focus on answering these two questions.

5.1. Dual Graph

Given a graph $G = G(V,E,W)$ with $|V| = N$ and an eigenvector distance d in Table 5.1, we build a *dual graph* $G^\star = G^\star(V^\star, E^\star, W^\star)$ by viewing the eigenvectors as its nodes, $V^\star = \{\phi_0, \ldots, \phi_{N-1}\}$, and the nontrivial affinity between eigenvector pairs as its edge weights,

$$W^\star_{ij} := 1/d(\phi_{i-1}, \phi_{j-1}), \quad i,j = 1:N.$$

We note that one can use the alternative and popular Gaussian affinity, i.e., $\exp(-d(\phi_{i-1},\phi_{j-1})^2/\epsilon)$. This affinity, however, requires a user to select an appropriate scale parameter $\epsilon > 0$, which is not a trivial task

[1] EMD is short for the Earth Mover's Distance, i.e., the ROT distance with $\alpha = 1$.

metrics	input arguments	parameter	notes
$d^{(1)}_{\mathrm{ROT}}$	$\mathrm{pmf}^{(1)}(\phi_i)$, $\mathrm{pmf}^{(1)}(\phi_j)$	$\alpha \in [0,1]$	entrywise square
$d^{(1)}_{\mathrm{ROT}}$	$\mathrm{pmf}^{(2)}(\phi_i)$, $\mathrm{pmf}^{(2)}(\phi_j)$	$\alpha \in [0,1]$	entrywise exponentiation
$d^{(2)}_{\mathrm{ROT}}$	ϕ_i, ϕ_j (ϕ_0 replaced by $\mathbf{0}$)	$\alpha \in [0,1]$	measures with total mass 0
d_{sROT}	$\mathrm{pmf}^{(1)}(\phi_i)$, $\mathrm{pmf}^{(1)}(\phi_j)$	$\alpha \in [0,1]$	only work on trees
d_{sROT}	$\mathrm{pmf}^{(2)}(\phi_i)$, $\mathrm{pmf}^{(2)}(\phi_j)$	$\alpha \in [0,1]$	only work on trees
d_{HAD}	ϕ_i, ϕ_j	—	related to d_{DAG}
d_{DAG}	ϕ_i, ϕ_j	—	fast computation
d_{TSD}	ϕ_i, ϕ_j (ϕ_0 replaced by $\mathbf{0}$)	$T \in \mathbb{R}_{>0} \cup \{\infty\}$	time-evolving EMD[1]-like

TABLE 5.1. List of all metrics introduced in Chapter 4.

as explained in [**53**], for example. Moreover, our edge weights using the inverse distances tend to connect the eigenvectors more globally compared to the Gaussian affinity with a fixed bandwidth. Using $G^\star$, which is a complete graph, for representing the graph spectral domain and studying relations between the eigenvectors is clearly more *natural* and effective than simply using the eigenvalue magnitudes, as [**10**, **51**, **72**] hinted at.

REMARK 5.1.1. *Our definition of the "dual graph" $G^\star$ of a given (primal) graph G is a graph representing the dual geometry/eigenvector domain of the primal graph; in particular, it is not related to the graph-theoretic notion of dual graph (see, e.g., [**27**, Sec. 1.8]), and does not satisfy the equivalence of the double dual graph and the primal graph. Our definition is also different from that of Leus et al. [**47**] who defined the dual graph of a primal graph by first assuming that the eigenbasis of a graph shift operator (e.g., the adjacency matrix) of the dual graph is the transpose of the eigenbasis of the graph shift operator of the primal graph. Furthermore, their definition does not work for graph Laplacian matrices.*

REMARK 5.1.2. *As known from the discrete 1D Fourier transform [**23**, Sec. 7.6], the dual graph of P_N can be viewed as itself. Therefore, we directly use P_N as the dual graph of P_N, i.e., $P_N^\star = P_N$, without computing the non-trivial metrics listed in Table 5.1 for all the numerical experiments in this dissertation.*

We denote the *dual graph signal* $\boldsymbol{g} = [g(1), \ldots, g(N)]^\mathsf{T} \in \mathbb{R}^N$ as a graph signal on $G^\star$, whose support node set is $V^\star$ and whose entry $g(l+1)$ is associated with the eigenvector $\boldsymbol{\phi}_l$ for $l = 0 : N-1$. We say that $\boldsymbol{g}$ is supported on $V_0^\star \subset V^\star$, if $g(l+1) \neq 0$ for $\boldsymbol{\phi}_l \in V_0^\star$ and $g(l+1) = 0$ for $\boldsymbol{\phi}_l \in V^\star \setminus V_0^\star$.

5.2. Classical Multidimensional Scaling (MDS)

Even though the eigenvectors in $V^\star$ can be naturally organized by the dual graph $G^\star$, we still cannot visualize the arrangement and get a better understanding of the intrinsic structure provided by the underlying metric. Fortunately, if some notion of distances between points is given, there are a number of ways to embed points into a low dimensional Euclidean space while preserving the inter-point distance as much as possible [**1**, **5**, **12**, **40**, **76**]. Throughout this dissertation, we use the classical Multidimensional Scaling (MDS) method [**5**, Chap. 12] to embed the eigenvectors into $\mathbb{R}^2$ or $\mathbb{R}^3$ for visualization purpose.

Given $G(V,E,W)$ and an eigenvector distance d in Table 5.1, we assemble the distance matrix $\Delta \in \mathbb{R}^{N\times N}$ by the mutual distance between the eigenvectors

$$\Delta_{ij} := d(\phi_{i-1}, \phi_{j-1}), \quad i,j = 1:N. \tag{5.1}$$

Clearly, Δ is symmetric and the diagonal elements of Δ are zeros. Then, we input the distance matrix Δ along with an embedding dimension s (e.g., $s = 2,3$) to the classical MDS algorithm, i.e., Algorithm 1, and we get the coordinate matrix of classical scaling $X \in \mathbb{R}^{N\times s}$, whose $(l+1)$-th row represents the embedding position of ϕ_l in $\mathbb{R}^s$ ($l = 0:N-1$).

Algorithm 1: The Classical Multidimensional Scaling (MDS) Algorithm [**5**, Chap. 12]

Input: A distance matrix $\Delta \in \mathbb{R}^{N\times N}$ and the dimension of the embedding space s (e.g., $s = 2,3$).
Output: The coordinate matrix of classical scaling $X \in \mathbb{R}^{N\times s}$.

1. Compute the matrix of squared distance $\mathrm{sqr.}(\Delta)$, where $\mathrm{sqr.}(\cdot)$ applies the square function in the entrywise manner to its argument.
2. Apply double centering to $\mathrm{sqr.}(\Delta)$ to get the Gram matrix B:
$$B = -\frac{1}{2} C\,\mathrm{sqr.}(\Delta) C,$$
where $C := I - \frac{1}{N}\mathbf{1}\cdot\mathbf{1}^\top$ is the so called *centering matrix*.
3. Compute the eigendecomposition of $B = U\Lambda U^\top$.
4. Let the diagonal matrix of the largest s eigenvalues *greater than zero* be $\Lambda_s \in \mathbb{R}^{s\times s}$ and the corresponding eigenvector matrix be $U_s \in \mathbb{R}^{N\times s}$. Then, the coordinate matrix of classical scaling is given by
$$X = U_s \Lambda_s^{1/2}.$$
Note that if the selected eigenvalues in Λ_s are simple, X is unique up to reflections over i-th coordinate ($i = 1:s$). Otherwise, if multiple eigenvalues are selected, X is also unique up to orthogonal rotations in the corresponding eigensubspaces.

In fact, Gower (1966) [**28**] proved that the coordinate matrix X computed in Algorithm 1 minimizes the cost function $c_{\mathrm{MDS}} : \mathbb{R}^{N\times s} \mapsto \mathbb{R}_{\geq 0}$,

$$c_{\mathrm{MDS}}(X) := \left\| XX^\top - B \right\|_F = \left\| XX^\top + \frac{1}{2} C\,\mathrm{sqr.}(\Delta) C \right\|_F,$$

where $\|\cdot\|_F$ is the Frobenius norm, and the Gram matrix B and the centering matrix C are as defined in Algorithm 1. Furthermore, the dimensions are *nested* for the classical MDS. It means that, for example, the first two columns of $X \in \mathbb{R}^{N\times 3}$ obtained by 3D MDS are the same as the two columns of $X \in \mathbb{R}^{N\times 2}$ obtained by 2D MDS.

5.3. Experimental Results

After we set up the concept of the *dual graph* $G^\star(V^\star, E^\star, W^\star)$ and the procedure of the *classical MDS algorithm*, we can visualize $V^\star$ in $\mathbb{R}^s$ ($s = 2,3$)[2] and examine the performance of various eigenvector metrics in Table 5.1. In this section, we run numerical experiments on two graphs: the 2D lattice $P_7 \times P_3$ (as seen in Figure 3.1) and a simple synthetic dendritic tree. On each graph, we show and explore the classical MDS embedding results[3] based on the metrics chosen from Table 5.1.

REMARK 5.3.1. *In the following experiments, if the chosen metric is associated with the parameter α or T, we try out $\alpha = 0.1, 0.2, 0.5, 1$ or $T = 0.1, 1, 10, \infty$, and select the one with the clearest embedding structure (i.e., not congested and good for visualization).*

5.3.1. 2D Lattice $P_7 \times P_3$. We have introduced the lattice graph $P_7 \times P_3$ in Section 3.1 as a motivational example. As demonstrated in Figure 3.1, we want to order the Laplacian eigenvectors in terms of their natural horizontal/vertical frequencies instead of nondecreasing eigenvalues. Let us see how the eigenvector metrics in Table 5.1 organize the dual graph's nodes $V^\star$ in $\mathbb{R}^2$ via the classical MDS.

Figure 5.1 shows the embedding of these 21 eigenvectors into $\mathbb{R}^2$ via the $d^{(1)}_{\mathrm{ROT}} \circ \mathrm{pmf}^{(1)}$ metric (Eq. (4.5)) with $\alpha = 0.5$ [**72**]. It somewhat reveals the two-dimensional ordering of the eigenvectors, but the eigenvectors with even or odd oscillations in either x (horizontal axis) or y (vertical axis) direction are embedded in a symmetric pattern around the DC vector $\varphi_{0,0}(= \phi_0)$. As explained in the paper [**72**], this phenomenon is due to the use of *squared eigenvectors* as pmfs and the facts

$$\varphi_{l_x,l_y}(x,y)^2 + \varphi_{N_x-l_x,l_y}(x,y)^2 = a^2_{l_x;N_x}\phi_{l_y;N_y}(y)^2, \quad l_x = 1:N_x-1, l_y = 0:N_y-1, x = 1:N_x, y = 1:N_y,$$

$$\varphi_{l_x,l_y}(x,y)^2 + \varphi_{l_x,N_y-l_y}(x,y)^2 = a^2_{l_y;N_y}\phi_{l_x;N_x}(x)^2, \quad l_x = 0:N_x-1, l_y = 1:N_y-1, x = 1:N_x, y = 1:N_y,$$

where $N_x = 7$, $N_y = 3$, and $a_{l_x;N_x}$, $\phi_{l_x;N_x}$, $a_{l_y;N_y}$, $\phi_{l_y;N_y}$ are as defined in Eq. (2.2).

Figure 5.2 shows the embedding of the 21 eigenvectors into $\mathbb{R}^2$ via the $d^{(1)}_{\mathrm{ROT}} \circ \mathrm{pmf}^{(2)}$ metric (Eq. (4.5)) with $\alpha = 0.1$. The eigenvectors are nicely grouped into four clusters around the DC vector $\varphi_{0,0}$: 1) eigenvectors only oscillate along y-axis (i.e., φ_{0,l_y}, $l_y = 1:N_y-1$); 2) eigenvectors only oscillate along x-axis (i.e., $\varphi_{l_x,0}$, $l_x = 1:N_x-1$); 3) eigenvectors oscillate in both directions but only having one oscillation

[2] In general, when a distance matrix is given for the classical MDS, we cannot assume the best embedding dimension a priori. But for visualization purposes, we choose $s = 2$ or 3.

[3] Without explicitly saying, the MDS results in this dissertation are unique up to reflections over the i-th coordinate ($i = 1:s$).

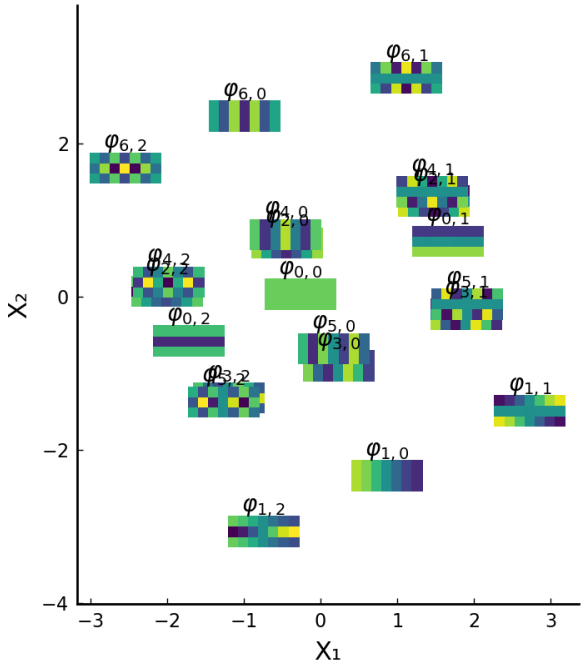


FIGURE 5.1. Embedding of the Laplacian eigenvectors of $P_7 \times P_3$ into $\mathbb{R}^2$ via $d^{(1)}_{\mathrm{ROT}} \circ \mathrm{pmf}^{(1)}$ ($\alpha = 0.5$) and the classical MDS.

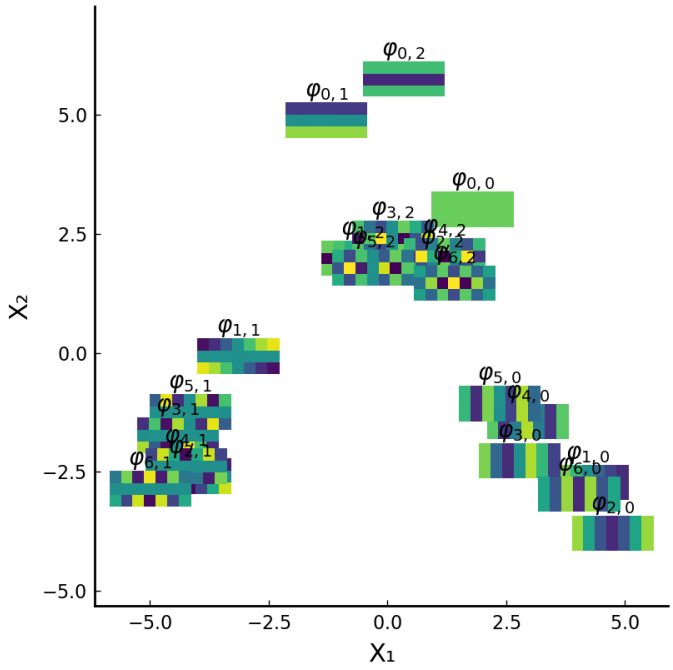


FIGURE 5.2. Embedding of the Laplacian eigenvectors of $P_7 \times P_3$ into $\mathbb{R}^2$ via $d^{(1)}_{\mathrm{ROT}} \circ \mathrm{pmf}^{(2)}$ ($\alpha = 0.1$) and the classical MDS.

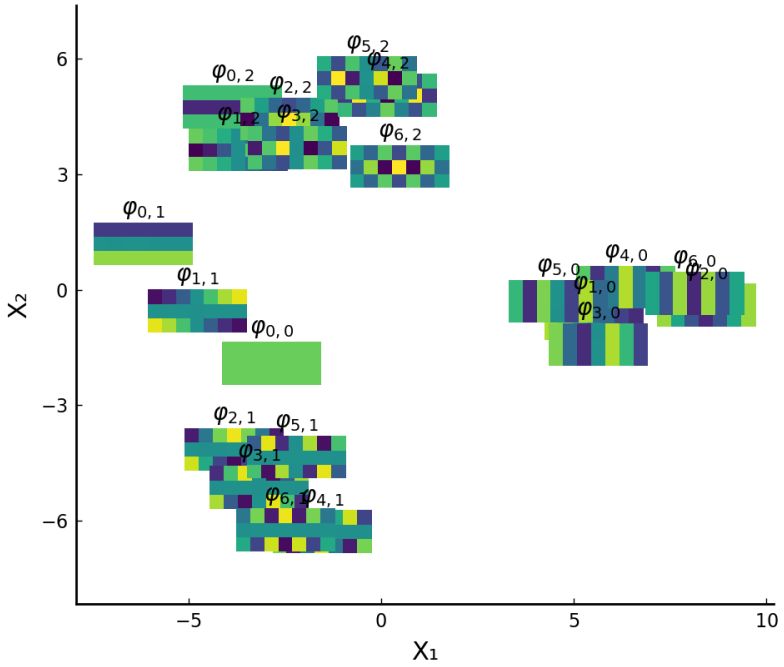


FIGURE 5.3. Embedding of the Laplacian eigenvectors of $P_7 \times P_3$ into $\mathbb{R}^2$ via $d^{(2)}_{\mathrm{ROT}}$ ($\alpha = 0.1$) and the classical MDS.

along y-axis (i.e., $\varphi_{l_x,1}$, $l_x = 1 : N_x - 1$); and 4) eigenvectors oscillate in both directions but having two oscillations along y-axis (i.e., $\varphi_{l_x,2}$, $l_x = 1 : N_x - 1$). Within each cluster, however, there is no clear ordering of these eigenvectors in terms of their horizontal frequencies.

Figure 5.3 shows the embedding of the 21 eigenvectors into $\mathbb{R}^2$ via the $d^{(2)}_{\mathrm{ROT}}$ metric (Eq. (4.6)) with $\alpha = 0.1$. As we can see, it shows a very similar arrangement of the eigenvectors compared to the previous case: $d^{(1)}_{\mathrm{ROT}} \circ \mathrm{pmf}^{(2)}$. This is due to the use of exponential function in $\mathrm{pmf}^{(2)}$ as defined in Eq. (4.4). Based on the Taylor expansion of the exponential function, i.e., $\exp(x) = 1 + x + o(x^2)$,

$$\exp(\phi_l(x)) \approx 1 + \phi_l(x), \quad l = 0 : N-1 \text{ and } x = 1 : N.$$

So if we ignore the normalization factor in $\mathrm{pmf}^{(2)}$, we are approximately computing the ROT distance between a set of input vector measures $\{\mathbf{1} + \phi_l\}_{l=0:N-1}$[4]. This would generate exactly the same result as the $d^{(2)}_{\mathrm{ROT}}$ metric. In other words, the normalization factor in $\mathrm{pmf}^{(2)}$ and the approximation in the Taylor expansion slightly differentiate the outcomes obtained by the metrics $d^{(1)}_{\mathrm{ROT}} \circ \mathrm{pmf}^{(2)}$ and $d^{(2)}_{\mathrm{ROT}}$.

[4]As before, we replace ϕ_0 by $\mathbf{0} \in \mathbb{R}^N$, so that all input measures share the same total mass N.

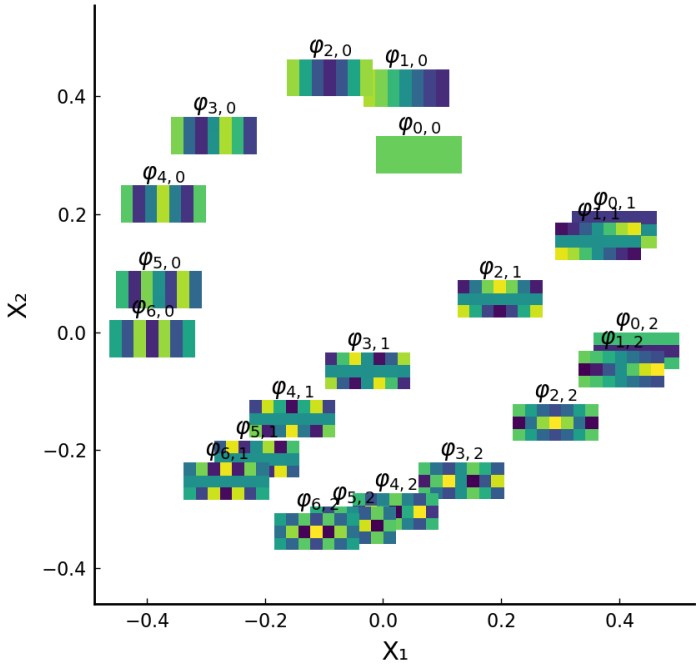


FIGURE 5.4. Embedding of the Laplacian eigenvectors of $P_7 \times P_3$ into $\mathbb{R}^2$ via d_{HAD} and the classical MDS.

Figure 5.4 shows the embedding of the 21 eigenvectors into $\mathbb{R}^2$ via d_{HAD} (Eq. (4.10)). It nicely reveals the two dimensional ordering of the eigenvectors, but the positions slightly deviate from a 2D lattice. We note that: 1) the authors of [**10**] use the *Hadamard product affinity* (i.e., a_{HAD} in Eq. (4.9)) instead of the *Hadamard product distance* (i.e., d_{HAD} in Eq. (4.10)) to organize the eigenvectors of a lattice graph; and 2) the authors apply another embedding technique instead of the classical MDS. Yet, the outcomes look similar. It shows our definition of d_{HAD} in Eq. (4.10) is reasonable.

Figure 5.5 shows the embedding of the 21 eigenvectors into $\mathbb{R}^2$ via d_{DAG} (Eq. (4.11)). It clearly reveals the natural two-dimensional organization of the eigenvectors, and is similar to a rotated version of Fig. 3.1b. Moreover, the embedding result is somewhat similar with the one via d_{HAD}. This is mainly because of the relation between d_{DAG} and a_{HAD} we derived in Eq. (4.12).

Figure 5.6 shows the embedding of the 21 eigenvectors into $\mathbb{R}^2$ via d_{TSD} metric (Eq. (4.16)) with $T = 0.1$. Like the results via d_{HAD} and d_{DAG}, it also reveals the two-dimensional ordering of the eigenvectors. However, the eigenvectors having no oscillation along y-axis (i.e., $\varphi_{l_x,0}$, $l_x = 0 : N_x - 1$) are sandwiched by the eigenvectors having one oscillation along y-axis (i.e., $\varphi_{l_x,1}$, $l_x = 0 : N_x - 1$) and the eigenvectors having two oscillations along y-axis (i.e., $\varphi_{l_x,2}$, $l_x = 0 : N_x - 1$), which is slightly different with the organization

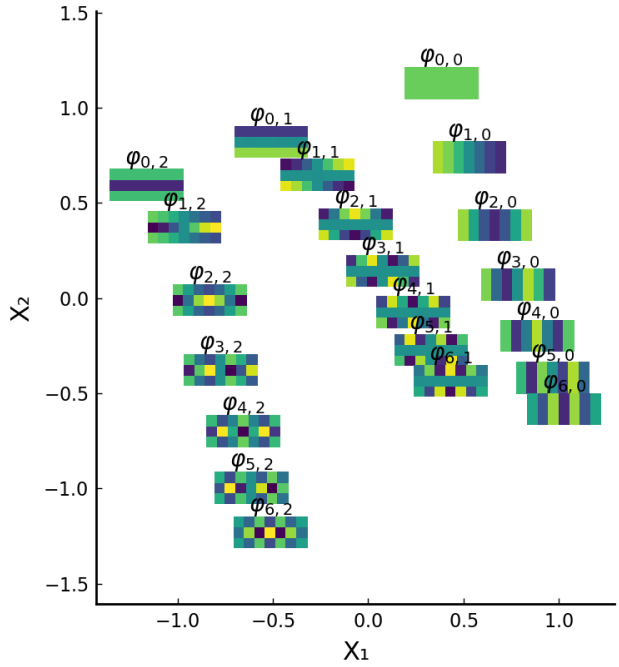


FIGURE 5.5. Embedding of the Laplacian eigenvectors of $P_7 \times P_3$ into $\mathbb{R}^2$ via d_{DAG} and the classical MDS.

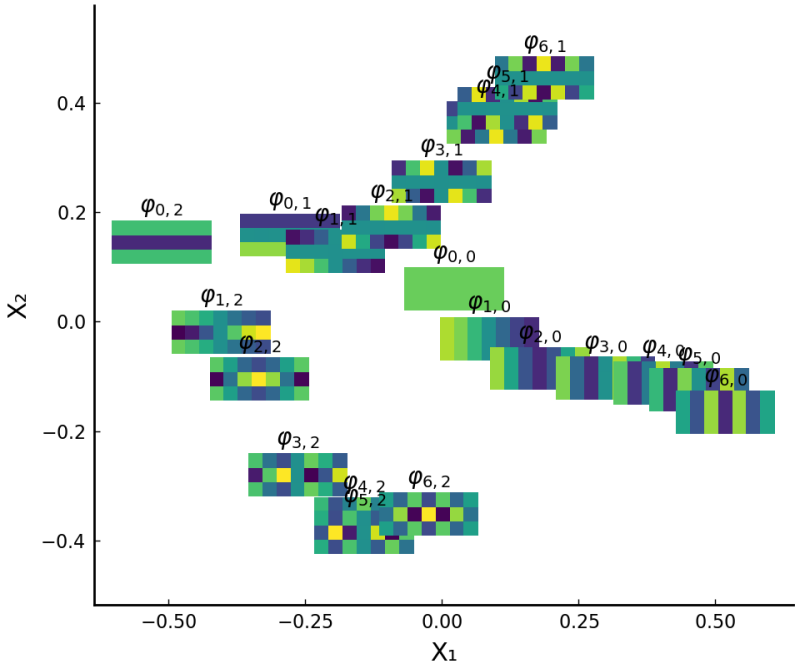


FIGURE 5.6. Embedding of the Laplacian eigenvectors of $P_7 \times P_3$ into $\mathbb{R}^2$ via d_{TSD} ($T = 0.1$) and the classical MDS.

described in Fig. 3.1b. On the other hand, note that the chosen parameter T is very small in this example. Therefore, the resemblance between the classical MDS results via d_{DAG} and d_{TSD} ($T = 0.1$) somewhat verifies the relation we discussed in Section 4.7.

Figure 5.7 shows the affinity between the embedding results via $d^{(2)}_{\mathrm{ROT}}$ ($\alpha = 1$) and d_{TSD} ($T = \infty$)[5]. We can see that everything centers around the DC vector $\varphi_{0,0}$: 1) low frequency[6] eigenvectors (e.g., $\varphi_{1,0}$, $\varphi_{0,1}$, and $\varphi_{2,0}$) are far apart from the center; and 2) high frequency eigenvectors are congested around the center. These phenomena are due to the fact that it is harder to flatten the vector measures of low frequency eigenvectors by either ROT or TSD comparing to the high frequency ones, because their positive measures and negative measures are more isolated across the graph. This similarity between the two embedding results shows the close relation between ROT and TSD we discussed in Section 4.6.

Furthermore, d_{TSD} is actually a time-dependent metric. In other words, we can get an embedding of these eigenvectors via d_{TSD} for any given $T \in \mathbb{R}_{>0} \cup \{\infty\}$. To examine the time evolution of the embedding via d_{TSD}, we generated a movie[7] to show how the embedding results evolve from Figure 5.6 ($T = 0.1$) to Figure 5.7b ($T = \infty$).

5.3.2. A Simple Synthetic Dendritic Tree. Since the d_{sROT} metric only works on trees, we create an *unweighted* tree $G = G(V, E)$ with $N = 100$ and $M = 99$ as shown in Figure 5.8a. Unlike the 2D lattice case, we do not have any clear priori information of the eigenvector behaviors on a general tree. Thus, we first study this particular tree and its Laplacian eigenvectors as follows. In Figure 5.8a, there are five regions that we are interested in: 1) the top left branch $G_0(V_0, E_0)$ (marked by pink); 2) the bottom left branch $G_1(V_1, E_1)$ (marked by orange); 3) the bottom right branch $G_2(V_2, E_2)$ (marked by green); 4) the top right branch $G_3(V_3, E_3)$ (marked by yellow); and 5) the four junctions (marked by red). We define and compute the energy score $\mathrm{ES}(\cdot, \cdot)$ of each eigenvector on each branch as

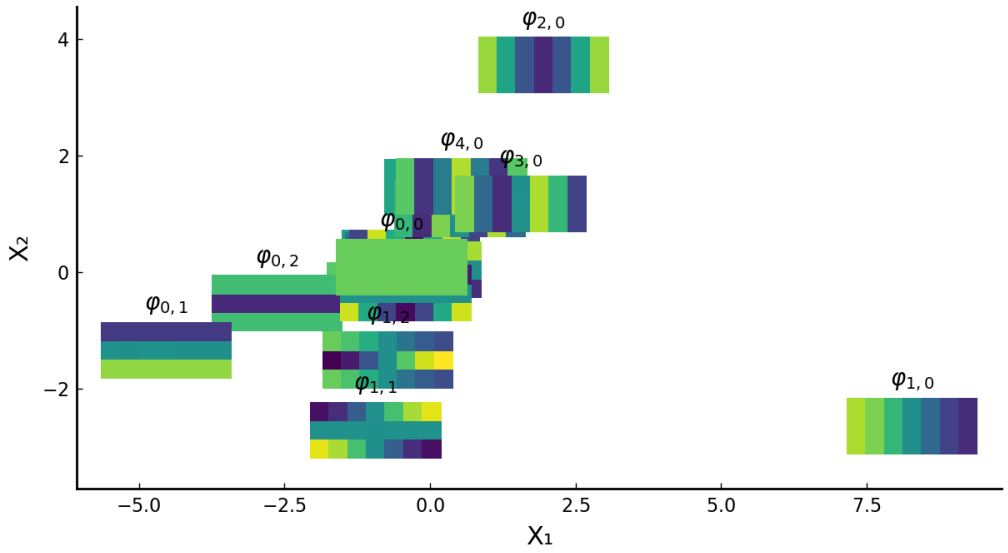
$$\mathrm{ES}(\phi_l, V_k) := \sum_{x \in V_k} \phi_l^2(x) \in [0, 1], \quad l = 0 : N-1 \text{ and } k = 0 : 3.$$

We then select the eigenvectors that mainly concentrate on the k-th branch, i.e., $\mathrm{ES}(\phi_l, V_k) > 0.5$, $k = 0 : 3$. By doing so, we get four groups of eigenvectors and we use the same four colors, i.e., pink, orange, green,

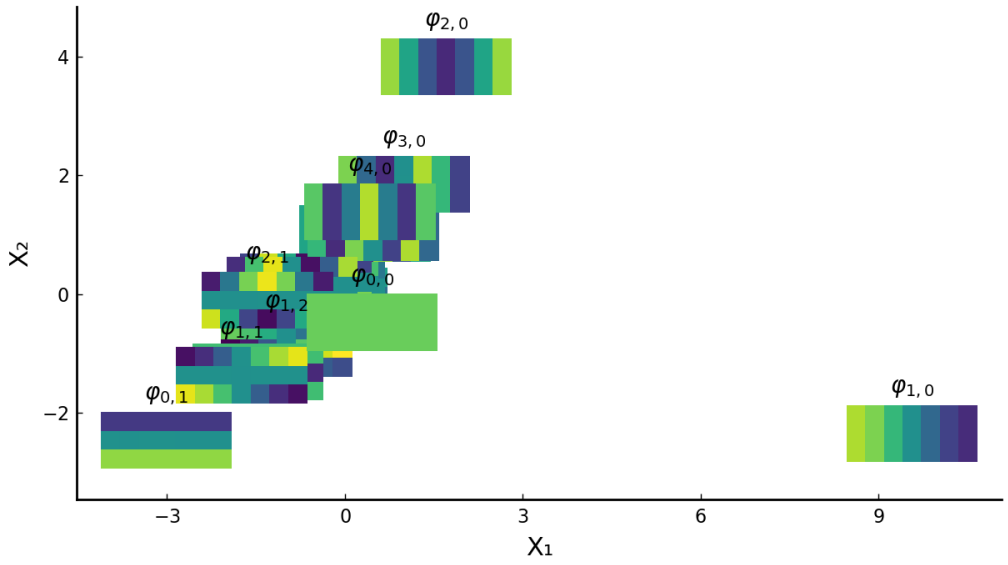
[5] In numerical simulations, we cannot really reach $T = \infty$, we stop at some T when $e^{-\lambda_1 T}$ is less than or equal to a predefined tolerance threshold, e.g., here we use the default $\mathrm{tol} = 10^{-5}$, which amounts to $T = 58.13$.

[6] Here frequency means the magnitude of the eigenvalue.

[7] `https://github.com/UCD4IDS/NGWP.jl/blob/master/dissertation/gifs/Grid7x3_MDS_TSD_T01_inf.gif`

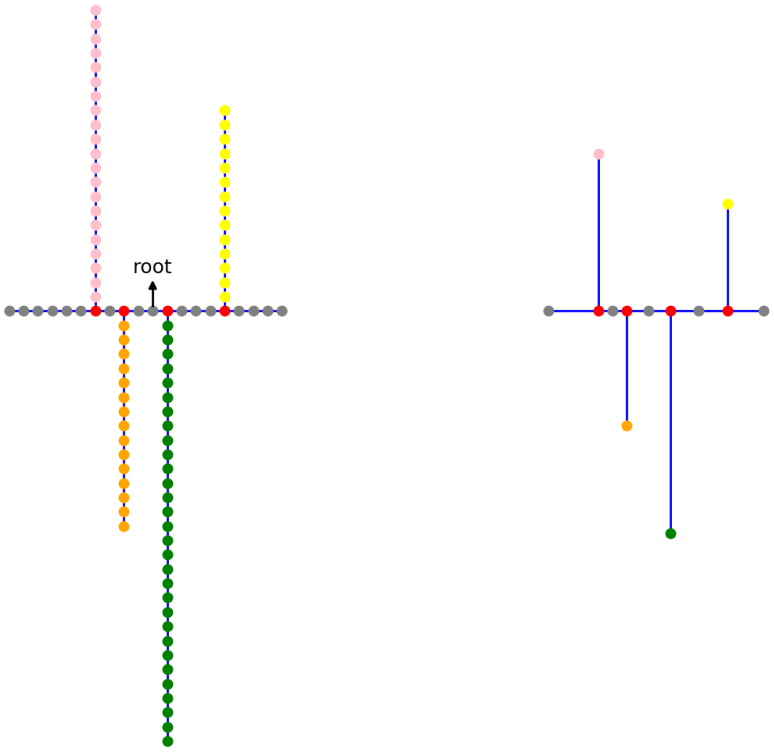


(a) $d^{(2)}_{\mathrm{ROT}}$ $(\alpha = 1)$



(b) d_{TSD} $(T = \infty)$

FIGURE 5.7. Comparison between the classical MDS results of the Laplacian eigenvectors of $P_7 \times P_3$ via $d^{(2)}_{\mathrm{ROT}}$ $(\alpha = 1)$ and d_{TSD} $(T = \infty)$.



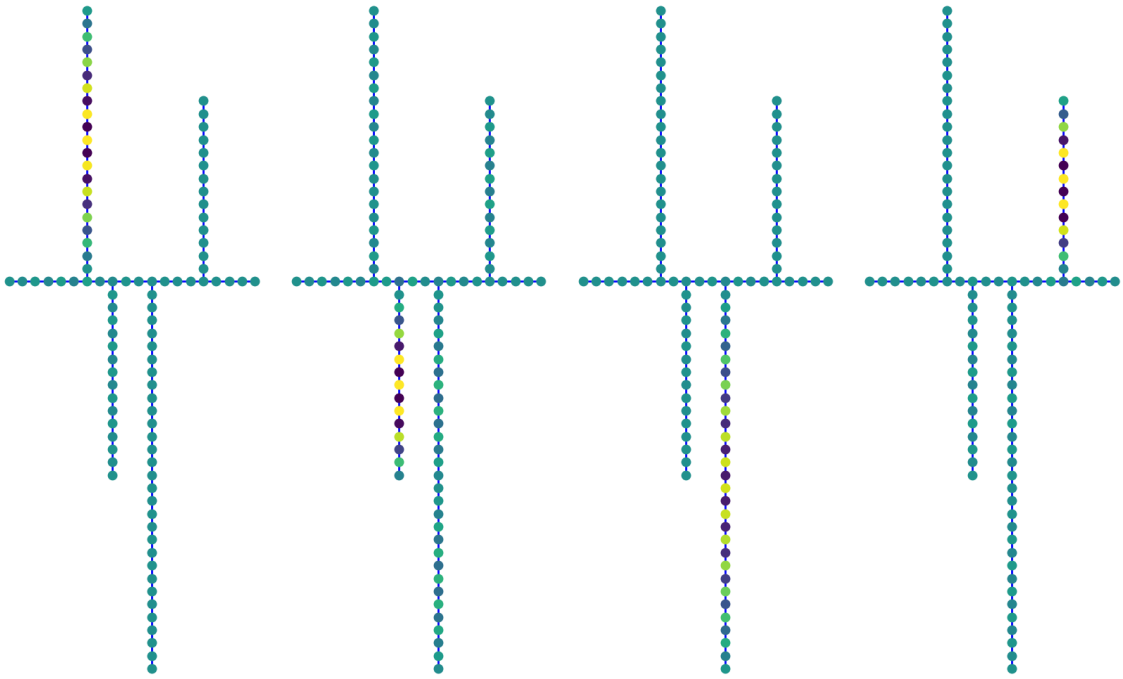
(a) an *unweighted* tree (b) the simplified version

FIGURE 5.8. A synthetic dendritic tree and its simplified version.

and yellow, to represent them respectively. Figure 5.9 shows examples of such eigenvectors of the four groups. We end up getting: 1) 18 eigenvectors for the *pink*-group; 2) nine eigenvectors for the *orange*-group; 3) 23 eigenvectors for the *green*-group; and 4) seven eigenvectors for the *yellow*-group.

On the other hand, the *phase transition* phenomenon as discussed in Section 3.2 and [**60**, **73**, **75**] also happens on this tree. Figure 5.10 shows the eigenvalue distribution of the tree. As we can see, there is also a sudden jump around the eigenvalue 4. In particular, there are four eigenvalues greater than 4 after the jump. The corresponding eigenvectors, as shown in Figure 5.11, are localized around the junctions (i.e, the red nodes marked in Figure 5.8a). Note that these four eigenvectors do not belong to any group we mentioned earlier. We denote them as the *red*-group. Now, we have five eigenvector groups of interest in total. The eigenvectors within each group can be considered as having *similar behavior* on the tree.

Figure 5.12a and Figure 5.12b show the classical MDS embedding of the 100 eigenvectors into $\mathbb{R}^3$ via $d_{\mathrm{ROT}}^{(1)} \circ \mathrm{pmf}^{(1)}$ ($\alpha = 1$) and $d_{\mathrm{sROT}} \circ \mathrm{pmf}^{(1)}$ ($\alpha = 1$) respectively. The meanings of the colors used in the



(a) ϕ_{94} (pink-group) (b) ϕ_{92} (orange-group) (c) ϕ_{95} (green-group) (d) ϕ_{91} (yellow-group)

FIGURE 5.9. Examples of four eigenvector groups that concentrated on V_0, V_1, V_2 and V_3, respectively. The eigenvector amplitudes within $(-0.3, 0.3)$ are mapped to the viridis colormap.

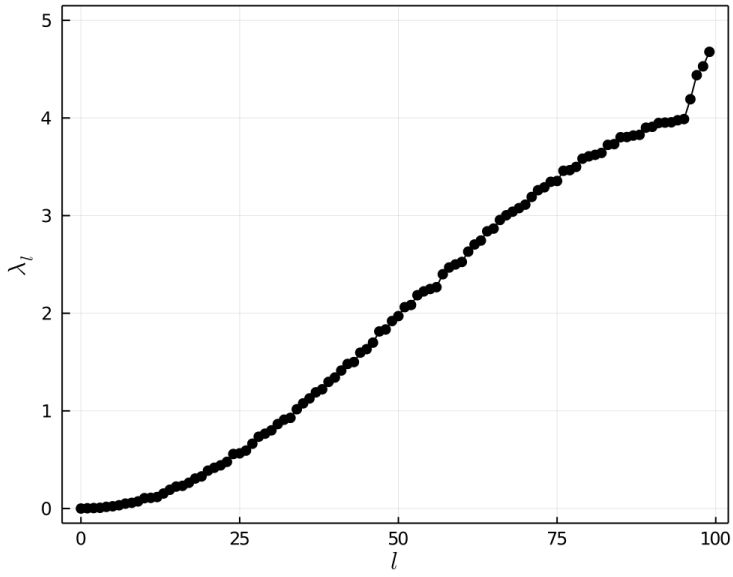
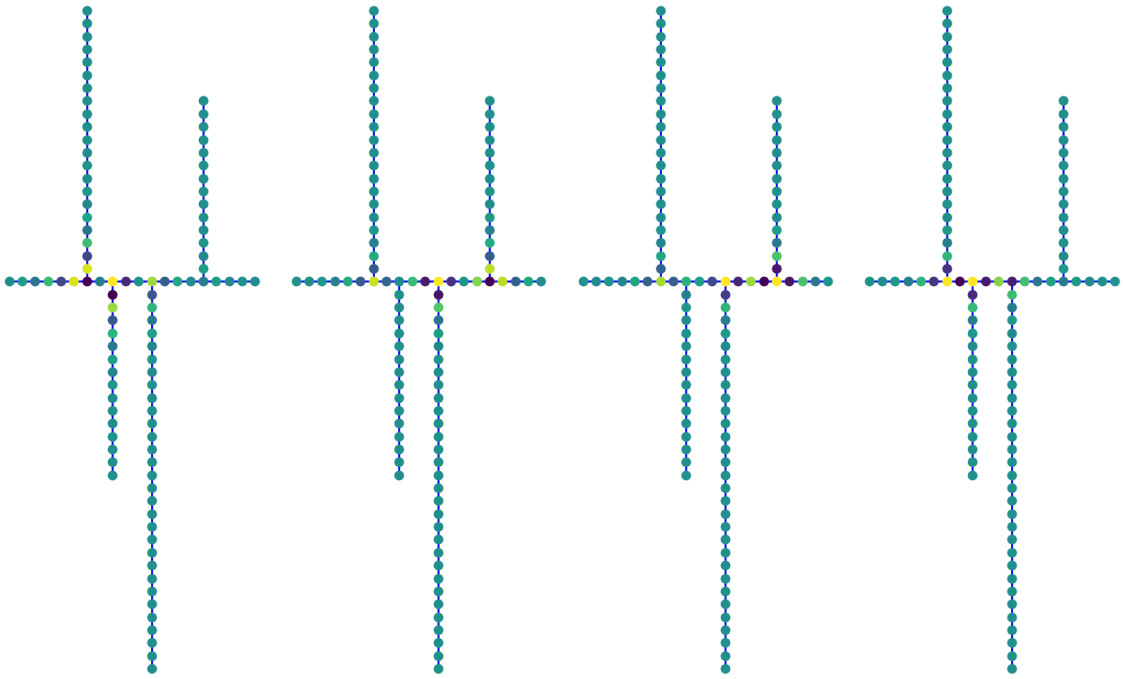


FIGURE 5.10. Eigenvalues of the graph Laplacian of the tree in Figure 5.8a.

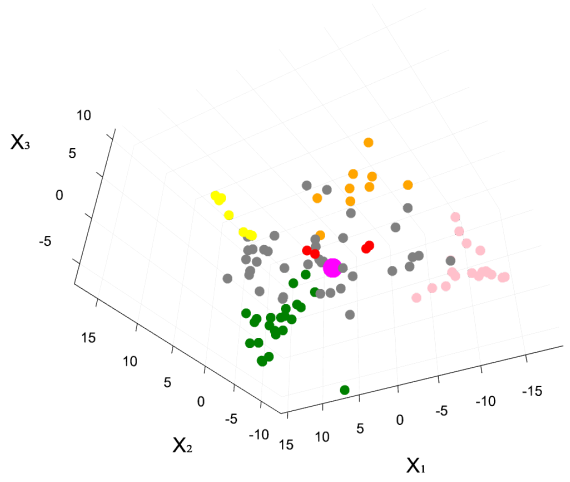


(a) ϕ_{96} ($\lambda_{96} = 4.1922$) (b) ϕ_{97} ($\lambda_{97} = 4.4384$) (c) ϕ_{98} ($\lambda_{98} = 4.5289$) (d) ϕ_{99} ($\lambda_{99} = 4.6776$)

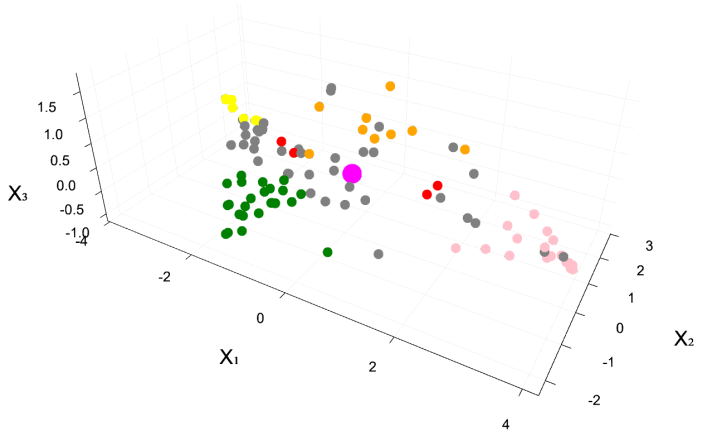
FIGURE 5.11. The four eigenvectors whose eigenvalues are great than 4. The eigenvector amplitudes within $(-0.3, 0.3)$ are mapped to the viridis colormap.

figures are described as follows: 1) the big magenta circle represents the DC vector ϕ_0; 2) the 18 pink circles represent the *pink*-group whose eigenvectors are mainly concentrated on V_0; 3) the nine orange circles represent the *orange*-group whose eigenvectors are mainly concentrated on V_1; 4) the 23 green circles represent the *green*-group whose eigenvectors are mainly concentrated on V_2; 5) the seven yellow circles represent the *yellow*-group whose eigenvectors are mainly concentrated on V_3; 6) the four red circles represent the *red*-group whose eigenvectors are localized around the junctions; and 7) the rest grey circles represent the eigenvectors that we are not interested in.

We can see from both of the classical MDS embedding results that all eigenvectors are centered around the DC vector and the eigenvectors within each color-group are nicely clustered. If we interpret or view the DC vector as the root node of a tree, the arrangement of the eigenvectors in the embedding space look like a "dual" tree in some sense. It means that the eigenvectors concentrated on the branches are farther away from the DC vector and get clustered into four corners just like the four branches, while the four eigenvectors localized on the junctions are closer to the DC vector and get allocated to the two



(a) $d^{(1)}_{\mathrm{ROT}} \circ \mathrm{pmf}^{(1)}$ $(\alpha = 1)$



(b) $d_{\mathrm{sROT}} \circ \mathrm{pmf}^{(1)}$ $(\alpha = 1)$

FIGURE 5.12. Comparison between the classical MDS results of the Laplacian eigenvectors of the tree (Figure 5.8a) via $d^{(1)}_{\mathrm{ROT}} \circ \mathrm{pmf}^{(1)}$ $(\alpha = 1)$ and $d_{\mathrm{sROT}} \circ \mathrm{pmf}^{(1)}$ $(\alpha = 1)$.

sides of the DC vector (two for each side) just like the positions of the four junctions in the original tree. These embedding results via the ROT metric and the sROT metric naturally organize the eigenvectors of the tree and help us get a better understanding of the *dual* domain. On the contrary, if we use the eigenvalues to organize the eigenvectors, we will group, for example, ϕ_{94}, ϕ_{95} and ϕ_{96} together even though their behaviors on the tree are very different, i.e., ϕ_{94} (Figure 5.9a) has a semi-global oscillation structure on V_0; ϕ_{95} (Figure 5.9c) has a semi-global oscillation structure on V_2; and ϕ_{96} (Figure 5.11a) is extremely localized on the junctions.

Also, the embedding results via the two metrics are similar, but the d_{sROT} is more computationally efficient than d_{ROT}. Figure 5.8b shows the simplified version of Figure 5.8a. The simplified tree is constructed as described in Section 4.2. It only has $K = 13$ nodes, which represent the four interested branches, the four junctions and the five disinterested branches, respectively. Computing the ROT distance on the simplified tree is much faster than on the original tree which has $N = 100$ nodes. Hence, d_{sROT} is a good replacement of $d^{(1)}_{\text{ROT}}$ for organizing the eigenvectors on trees.

We also examined the case of $d^{(1)}_{\text{ROT}} \circ \text{pmf}^{(2)}$ vs. $d^{(1)}_{\text{sROT}} \circ \text{pmf}^{(2)}$, which gave us more congested embedding results around $\varphi_{0,0}$.

5.4. Summary

We have presented the classical MDS embedding results on $P_7 \times P_3$ and a simple synthetic dendritic tree demonstrating the effectiveness of using the metrics (as introduced in Chapter 4) to organize the graph Laplacian eigenvectors (i.e., the nodes of the dual graph). In general, we are interested in two types of eigenvector behavior patterns on graphs: 1) global and directional (or oriented) oscillation pattern; and 2) energy localization/concentration pattern. The first pattern characterizes how the eigenvector globally oscillate on the graphs, e.g., the DCT type II eigenvectors of P_N where the oscillations pattern are completely characterized by the eigenvalues; or the eigenvectors of $P_{N_1} \times \cdots \times P_{N_d}$ ($d \geq 2$) where the oscillation patterns can be characterized by different directional frequencies. On the other hand, the second pattern describes node locations where the eigenvectors are more active, e.g., the eigenvectors of trees may be concentrated on the junctions or may have a semi-global oscillation structure on a certain branch. Some metrics we described in Chapter 4 (summarized in Table 5.1) work well to discriminate the first pattern while the others work better detecting the second pattern. It is a hard but important

question to ask which metric is preferable for eigenvector organization on a given graph. In the following, we discuss some empirical observations on different type of graphs.

For Cartesian product graphs, the ROT metric does not perform very well for detecting the oscillation patterns of the eigenvectors in general. On the contrary, the DAG and the HAD pseudometrics reveal the directional oscillation patterns of the eigenvectors quite well. For trees, however, the sROT and the ROT metrics are great at detecting the energy localization of the eigenvectors. For the TSD metric with small $T > 0$, perceptually, it is good at oscillation detection because of its similarity to the DAG pseudometric as discussed in Section 4.7. On the other hand, for the TSD with large T (or $T = \infty$), it is good at detecting energy concentration because of the similarity to the ROT metric (with $\alpha = 1$) as discussed in Section 4.6. However, it is hard to tell which T is the "best" for a given input graph.

Although these metrics provide natural and interpretable organizations of the eigenvectors, there remain several questions that need to be answered:

- Q1: What is the best way to turn a $\phi_l \in \mathbb{R}^N$ to a pmf in $d^{(1)}_{\mathrm{ROT}}$ or d_{sROT}?
- Q2: How do we choose the best parameter $\alpha \in [0,1]$ in the ROT metrics?
- Q3: How do we choose the best parameter $T \in \mathbb{R}_{>0} \cup \{\infty\}$ in the TSD metric?
- Q4: Is it possible to design fast algorithms to compute the ROT and the TSD metrics? If so, how?
- Q5: Besides the two types of eigenvector behavior patterns on graphs we discussed above, are there other patterns that are also interesting and need to be addressed?
- Q6: How can we design an auto-adaptive and cost efficient metric that is good for all types of eigenvector behavior detection?
- Q7: How can we develop the true Littlewood-Paley theory on graphs and most "natural" wavelets on graphs once the above embedding is done or the dual graph is constructed by the eigenvector distances?

Q1 to Q6 are left for the future research, while the last question Q7 is answered in the following chapters.

CHAPTER 6

Natural Graph Wavelet Packets using *Varimax Rotations*

In this chapter, we propose one of our natural graph wavelet packet dictionary constructions solely based on hierarchical bipartitioning of $G^\star = G^\star(V^\star, E^\star, W^\star)$. Basic steps to generate such a graph wavelet packet dictionary for $G = G(V, E, W)$ are quite straightforward:

Step 1: *Bipartition the dual graph $G^\star$ recursively* via any method, e.g., spectral graph bipartition using the *Fiedler vectors*;

Step 2: *Generate wavelet packet vectors* using the eigenvectors belonging to each subgraph of $G^\star$ that are *well localized on G*.

Note that Step 1 corresponds to bipartitioning the frequency band of an input signal using the *characteristic functions* in the classical setting. Hence, our graph wavelet packet dictionary constructed as above can be viewed as a graph version of the *Shannon* wavelet packet dictionary [**54**, Sec. 8.1.2]. We now describe the details of each step of our graph wavelet packet dictionary construction below.

6.1. Hierarchical Bipartitioning of $G^\star$

Let $V_0^{\star(0)} := V^\star$ be the node set of the dual graph $G^\star$, which is simply the set of the eigenvectors of the unnormalized graph Laplacian matrix $L(G)$. Suppose we get the *hierarchical bipartition tree* of $V_0^{\star(0)}$ as shown in Figure 6.1, where we use K^j to denote the number of sets of nodes on level j of the tree, and use $k = 0 : K^j - 1$ to index these sets. Hence, each $V_k^{\star(j)}$ contains an appropriate subset of the eigenvectors of $L(G)$. As we mentioned earlier, any graph bipartitioning method can be used to generate this hierarchical bipartition tree of $G^\star$. Typically, we use the Fiedler vector of the random-walk normalized graph Laplacian matrix L_{rw} (see Eq. (2.1)) of each subgraph of $G^\star$ throughout this dissertation, whose use is preferred over that of L or L_{sym} as von Luxburg discussed [**87**].

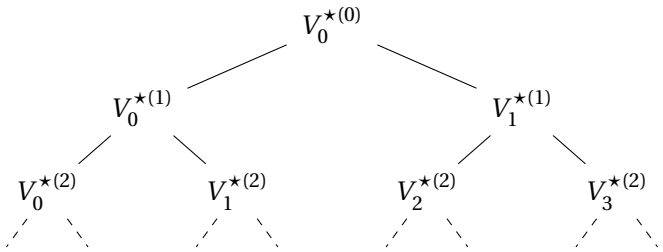


FIGURE 6.1. The hierarchical bipartition tree of the dual graph nodes $V^\star \equiv V_0^{\star(0)}$, which corresponds to the frequency domain bipartitioning used in the classical wavelet packet dictionary.

REMARK 6.1.1. *We recursively apply the above bipartition algorithm until we reach $j = j_{\max} > 0$, where each $V_k^{\star(j_{\max})}$, $0 \le k < K^{j_{\max}} = N$, contains a single eigenvector. Note that the previous graph basis dictionaries, i.e., HGLET [**34**], GHWT [**33**], and eGHWT [**77**], also constructed such "full" hierarchical bipartition trees in the primal (input graph) domain, not in the dual domain. Note also that during the hierarchical bipartition procedure, some $V_k^{\star(j)}$ may become a singleton before reaching $j = j_{\max}$. If this happens, such a subset is copied to the next lower level $j+1$. See [**36**] for the detailed explanation of such situations. We can also stop the recursion at some level $J (< j_{\max})$, of course. Below, we denote $j_{\max}$ as the deepest possible level at which every subset becomes a singleton for the first time whereas we denote $J (\le j_{\max})$ as a more general deepest level specified by a user.*

As an example, Figure 6.2 demonstrates the above strategy for the 2D lattice graph discussed in Section 3.1 and Section 5.3.1 accompanied with the DAG pseudometric d_{DAG}, whose dual domain geometry together with the graph Laplacian eigenvectors belonging to $V_0^{\star(0)}$ was displayed in Figure 5.5. The thick red line indicates the first split of $V_0^{\star(0)}$, i.e., all the eigenvectors above this red line belong to $V_0^{\star(1)}$ while those below it belong to $V_1^{\star(1)}$. Then, our hierarchical bipartition algorithm further splits them into $\left\{V_0^{\star(2)}, V_1^{\star(2)}\right\}$ and $\left\{V_2^{\star(2)}, V_3^{\star(2)}\right\}$, respectively. This two-level bipartition pattern is quite reasonable and natural considering the fact that the size of the original rectangle is 7×3, both of which are odd integers.

6.2. Localization on G via Varimax Rotation

For realizing Step 2 of the above basic algorithm, we propose to use the *varimax rotation* on the eigenvectors in $V_k^{\star(j)}$ for each j and k. Let $\Phi_k^{(j)} \in \mathbb{R}^{N \times N_k^j}$ be a matrix whose columns are the eigenvectors belonging to $V_k^{\star(j)}$. A varimax rotation is an orthogonal rotation, originally proposed by Kaiser [**42**] and

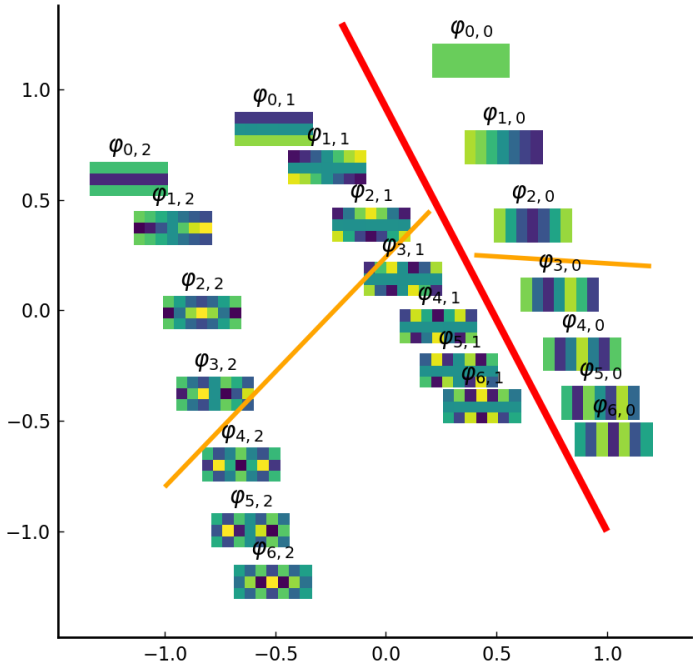


FIGURE 6.2. The result of the hierarchical bipartition algorithm applied to the dual geometry of the 2D lattice graph $P_7 \times P_3$ via d_{DAG} shown in Figure 5.5 with $J = 2$. The thick red line indicates the bipartition at $j = 1$ while the orange lines indicate those at $j = 2$.

often used in *factor analysis* (see, e.g., [**57**, Chap. 11]), to maximize the variances of energy distribution (or a scaled version of the *kurtosis*) of the input column vectors, which can also be interpreted as the *approximate entropy minimization of the distribution of the eigenvector components* [**70**, Sec. 3.2].

Thanks to the orthonormality of columns of $\Phi_k^{(j)}$, this is equivalent to finding an orthogonal rotation that maximizes the overall *4th order moments*, i.e.,

$$\Psi_k^{(j)} := \Phi_k^{(j)} \cdot R_k^{(j)}, \quad \text{where } R_k^{(j)} = \arg \max_{R \in \mathrm{SO}(N_k^j)} \sum_{x=1}^{N} \sum_{y=1}^{N_k^j} \left[\left(\Phi_k^{(j)} \cdot R \right)^4 \right]_{x,y}. \tag{6.1}$$

The algorithm of varimax rotation we adopted and used in this article is the so-called Basic Singular Value (BSV) algorithm proposed by Jennrich [**38**]. Algorithm 2 describes the details. Jennrich also showed that under general conditions, this algorithm converges to a stationary point from any initial estimate. The BSV algorithm seems to be the standard varimax rotation algorithm available in many packages, e.g., MATLAB ®[1], R, etc. The procedures of solving $R_k^{(j)}$ in Eq. (6.1) can be summarized as follows:

(0) Initialize an orthogonal rotation matrix R.

[1] MATLAB is a registered trademark of The MathWorks, Inc.

Algorithm 2: The Varimax Rotation Algorithm

Input: Full column rank input matrix whose columns to be rotated $A \in \mathbb{R}^{N\times K}$ ($K \leq N$); maximum number of iteration steps maxit (default: 1000); relative tolerance tol (default: 1e-12)

Output: Rotated matrix $B \in \mathbb{R}^{N\times K}$

```
B = A                                                  // initialize the output matrix
S = 0                                                  // initialize the nuclear norm
for i = 1 : maxit do
    S_0 = S
    [U, Σ, V] = svd(A^T · (N · cube.(B) − B · diag(B^T · B)))      // where cube.(·) applies
                  // the cube function in the entrywise manner to its argument
    R = U · V^*                                  // update the orthogonal rotation matrix
    S = trace(Σ)
    B = A · R                                                // update the rotated matrix
    if |S − S_0|/S < tol then
        break                                    // stop when S does not change much
    end
end
return B
```

(1) Compute $\mathrm{d}f/\mathrm{d}R$, where f is the objective function defined in Eq. (6.1), and $\mathrm{d}f/\mathrm{d}R$ is the matrix of partial derivatives of f at R.

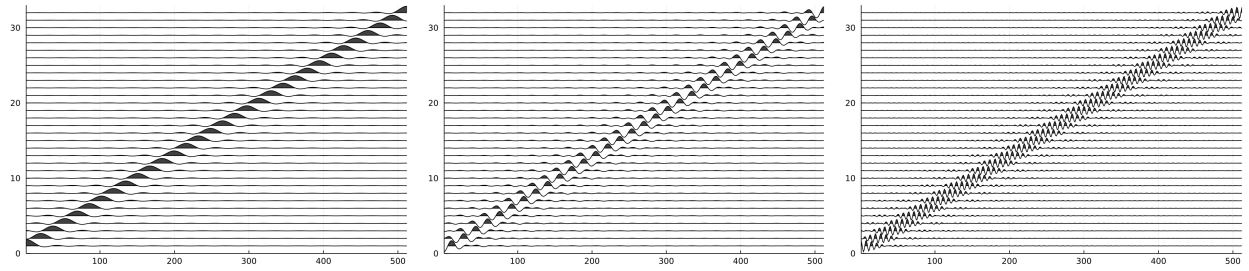
(2) Find the singular value decomposition $U\Sigma V^*$ of $\mathrm{d}f/\mathrm{d}R$.

(3) Replace R by UV^* and go to (1) or stop.

(4) Set $R_k^{(j)} := R$.

The column vectors of obtained $\Psi_k^{(j)}$ are *more "localized" in the primal domain G* than those of $\Phi_k^{(j)}$. This type of localization is important since the graph Laplacian eigenvectors in $\Phi_k^{(j)}$ are of *global* nature in general. We also note that the column vectors of $\Psi_k^{(j)}$ are orthogonal to those of $\Psi_{k'}^{(j')}$ as long as the latter is neither a direct ancestor nor a direct descendant of the former. Hence, Steps 1 and 2 mentioned in the beginning of this chapter truly generate the graph wavelet packet dictionary for an input graph. We refer to this graph wavelet packet dictionary $\left\{\Psi_k^{(j)}\right\}_{j=0:J;\,k=0:K^j-1}$ generated by this algorithm as the *Varimax Natural Graph Wavelet Packet* (VM-NGWP) dictionary.

One can run the *best-basis algorithm* of Coifman-Wickerhauser [**16**] on this dictionary to extract the ONB most suitable for a task at hand (e.g., an efficient graph signal approximation) once an appropriate cost function is specified (e.g., the ℓ^p-norm minimization, $0 < p \leq 1$). Note also that it is easy to extract a graph Shannon wavelet basis from this dictionary by specifying the appropriate dual graph nodes, i.e.,



(a) Father wavelet vectors $\Psi_0^{(4)}$ (b) Mother wavelet vectors $\Psi_1^{(4)}$ (c) Wavelet packet vectors $\Psi_4^{(4)}$

FIGURE 6.3. Some of the Shannon wavelet packet vectors on P_{512}.

$\Psi_1^{(1)}, \Psi_1^{(2)}, \ldots, \Psi_1^{(J)}$, and the father wavelet vectors $\Psi_0^{(J)}$ where $J(\leq j_{\max})$ is the user-specified deepest level of the hierarchical bipartition tree. We point out that the meaning of the level index j in our NGWP dictionaries is different from that in the general graph wavelet frames (2.4) discussed in Section 2.2: in our NGWP dictionaries, a smaller j corresponds to a finer and more localized (in the primal graph domain) basis vector in $V_k^{\star(j)}$.

Let us now demonstrate that our algorithm actually generates the classical *Shannon* wavelet packets dictionary [**54**, Sec. 8.1.2] when an input graph is the simple path P_N. Note that the varimax rotation algorithm does not necessarily sort the vectors as shown in Figure 6.3 because the minimization in Eq. (6.1) is the same modulo to any permutation of the columns and any sign flip of each column. In other words, to produce Figure 6.3, we carefully applied sign flip to some of the columns, and sorted the whole columns so that each subfigure simply shows translations of the corresponding wavelet packet vectors.

Let us also demonstrate how some VM-NGWP basis vectors of the 2D lattice graph $P_7 \times P_3$ look like. Figure 6.4 shows such VM-NGWP basis vectors with $J = 2$. Those basis vectors are placed at the same locations as the graph Laplacian eigenvectors in the dual domain shown in Figure 6.2 for the demonstration purpose. It is quite clear that those VM-NGWP basis vectors are more localized in the primal graph domain than those graph Laplacian eigenvectors shown in Figure 6.2. We note that we determined the index l in $\psi_{k,l}$ for each k in such a way that the main features of the VM-NGWP basis vectors translates nicely in the horizontal and vertical directions, and some sign flips were applied as in the case of the 1D Shannon wavelet packets shown in Figure 6.3. As one can see, like the classical wavelet packet vectors on a rectangle, $\{\psi_{0,l}\}_{l=0:2}$, are the father wavelets and clearly function as local averaging operators along the horizontal direction, while $\{\psi_{1,l}\}_{l=0:3}$, work as localized first order differential operators along

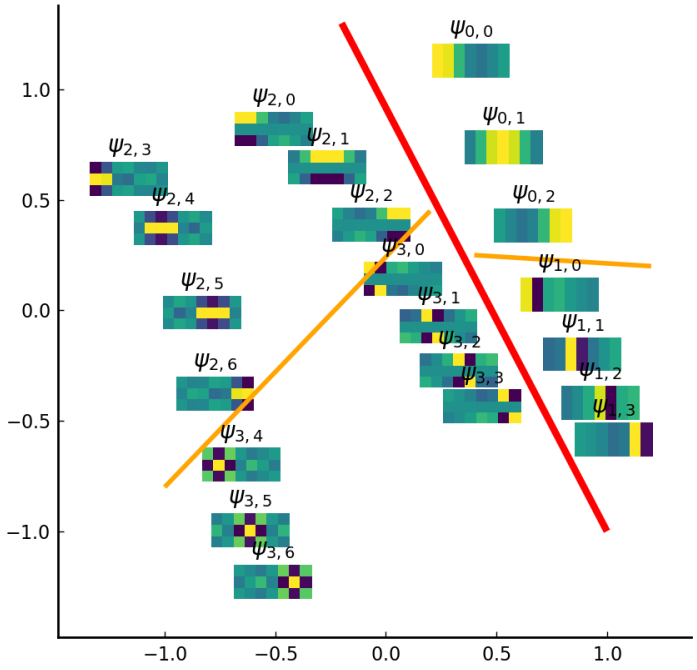


FIGURE 6.4. The VM-NGWP basis vectors of the 2D lattice graph $P_7 \times P_3$ computed by the varimax rotations in the hierarchically partitioned dual domain shown in Figure 6.2. Note that the column vectors of the basis matrix $\Psi_k^{(2)}$ are denoted as $\psi_{k,l}$, $l = 0, 1, \ldots$, in this figure instead of $\psi_{k,l}^{(2)}$ for simplicity.

the horizontal direction. On the other hand, $\{\psi_{2,l}\}_{l=0:2}$ work as localized first order differential operators along the vertical direction; $\{\psi_{2,l}\}_{l=3:6}$ work as localized second order differential operators along the vertical direction; $\{\psi_{3,l}\}_{l=0:3}$ work as localized mixed differential operators; and finally, $\{\psi_{3,l}\}_{l=4:6}$ work as localized Laplacian operators.

6.3. Computational Complexity

The varimax rotation algorithm (i.e., Algorithm 2) is of iterative nature and is an example of the BSV algorithms [**38**]: for each iteration at the dual node set $V_k^{\star(j)}$, it requires computing the full Singular Value Decomposition (SVD) of a matrix of size $N_k^j \times N_k^j$ representing a gradient of the objective function, which itself is computed by multiplying matrices of sizes $N_k^j \times N$ and $N \times N_k^j$. The convergence is checked with respect to the relative error between the current and previous gradient estimates measured in the nuclear norm (i.e., the sum of the singular values). For our numerical experiments in Chapter 9, we set the maximum iteration as 1000 and the error tolerance as 10^{-12}. Therefore, to generate $\Psi_k^{(j)}$ for each

(j,k), the computational cost in the worst case scenario is $O\left(c\cdot(N_k^j)^3+N\cdot(N_k^j)^2\right)$ where $c=1000$ and the first term accounts for the SVD computation and the second does for the matrix multiplication. For a perfectly balanced and fully developed bipartition tree with $N=2^{j_{\max}}$, we have $K^j=2^j$ and $N_k^j=2^{j_{\max}-j}$, $j=0:j_{\max}$, $k=0:2^j-1$. Hence we have:

$$\sum_{k=0}^{2^j-1}(N_k^j)^2=\sum_{k=0}^{2^j-1}2^{2(j_{\max}-j)}=2^{2j_{\max}-2j}\cdot 2^j=N^2\cdot 2^{-j}, \tag{6.2}$$

and

$$\sum_{k=0}^{2^j-1}(N_k^j)^3=\sum_{k=0}^{2^j-1}2^{3(j_{\max}-j)}=2^{3j_{\max}-3j}\cdot 2^j=N^3\cdot 2^{-2j}.$$

Note that at the bottom level $j=j_{\max}$, each node is a leaf containing only one eigenvector, and there is no need to do any rotation estimation and computation. Note also that at the root level $j=0$, the columns of $\Phi_0^{(0)}$ span the whole $\mathbb{R}^N$, and we know that the varimax rotation turns $\Phi_0^{(0)}$ into the identity matrix (or its permuted version). Hence, we do not need to run the varimax rotation algorithm on the root node. Finally, summing the cost $O\left(c\cdot(N_k^j)^3+N\cdot(N_k^j)^2\right)$ from $j=1$ to $j_{\max}-1$, the total worst case computational cost becomes $O((1+c/3)N^3-2N^2-4c/3N)$. So after all, it is an $O(N^3)$ algorithm. In practice, the convergence is often achieved with less than 1000 iterations at each node except possibly for the nodes with small j where N_k^j is large. For example, when computing the VM-NGWP dictionary for the path graph P_{512} ($j_{\max}=9$) shown in Figure 6.3, the average number of iterations over all the dual graph nodes $\left\{V_k^{\star(j)}\right\}_{j=0:9;\,k=0:2^j-1}$ was 68.42 with the standard deviation 98.09.

6.4. L_{sym} Version of the *VM-NGWP*

So far, we use the unnormalized graph Laplacian eigenvectors of $L(G)$ to construct the VM-NGWP dictionary. In parallel, we can also use the symmetric normalized graph Laplacian eigenvectors of $L_{\text{sym}}(G)$ as $V^\star$ to construct another version of the VM-NGWP dictionary.[2] However, note that ϕ_0^{sym} is no longer a constant vector; furthermore, ϕ_l^{sym} and $\phi_{l'}^{\text{sym}}$ ($l\neq l'$) do not share the same total mass in general. Hence, the metrics $d_{\text{ROT}}^{(2)}$ and d_{TSD} in Table 5.1 cannot be used to construct $W^\star$. The remaining *viable* metrics (or pseudometrics) are $d_{\text{ROT}}^{(1)}$, d_{sROT} (for trees), d_{HAD}, and d_{DAG}. After the L_{sym} version of $G^\star$ is built, we continue to construct the L_{sym} *version of the Varimax Natural Graph Wavelet Packet* (VM-NGWP-L_{sym}) dictionary by following the same procedures as discussed in Section 6.1 and Section 6.2.

[2] We cannot use the random-walk normalized graph Laplacian eigenvectors of $L_{\text{rw}}(G)$, since they do not form an ONB for $\mathbb{R}^N$.

CHAPTER 7

Natural Graph Wavelet Packets using *Pair-Clustering*

Another way to construct a natural graph wavelet packet dictionary is to mimic the convolution and subsampling strategy of the classical wavelet packet dictionary construction: form a binary tree of spectral filters in the dual domain via $\left\{V_k^{\star(j)}\right\}_{j=0:J;k=0:K^j-1}$ and then perform the filtering/downsampling process based on the relations between the sampling points (primal nodes) and the eigenvectors of $L(G)$. In order to fully utilize such relations, we look for a coordinated pair of partitions on G and $G^\star$, which is realized by our *pair-clustering* algorithm described below. We will first describe the one-level pair-clustering algorithm and then proceed to the hierarchical version.

7.1. One-Level Pair-Clustering

Suppose we partition the dual graph $G^\star$ into $K \geq 2$ clusters using any method including the spectral clustering [**87**] as we used in the previous chapter. Let $V_0^\star, \ldots, V_{K-1}^\star$ be those mutually disjoint K clusters of the nodes $V^\star$, i.e., $V^\star = \bigsqcup_{k=0}^{K-1} V_k^\star$, which is also often written as $\bigoplus_{k=0}^{K-1} V_k^\star$. Denote the cardinality of each cluster as $N_k := |V_k^\star|$, $k = 0 : K-1$, and we clearly have $\sum_{k=0}^{K-1} N_k = N$. Then, we also partition the primal graph nodes V into mutually disjoint K clusters, $V_0, \ldots, V_{K-1}$ with the constraint that $|V_k| = |V_k^\star| = N_k$, $k = 0 : K-1$, and the members of V_k and $V_k^\star$ are as "closely related" as possible. The purpose of partitioning V is to select appropriate primal graph nodes as sampling points around which the graph wavelet packet vectors using the information on $V_k^\star$ are localized. With a slight abuse of notation, let V also represent a collection of the standard basis vectors in $\mathbb{R}^N$, i.e., $V := \{\boldsymbol{\delta}_1, \ldots, \boldsymbol{\delta}_N\}$, where $\boldsymbol{\delta}_x(x) = 1$ and 0 otherwise. In order to formalize this constrained clustering of V, we define the *affinity measure* α_{PC} between V_k and $V_k^\star$ as follows:

$$\alpha_{\mathrm{PC}}(V_k, V_k^\star) := \sum_{\boldsymbol{\delta} \in V_k, \boldsymbol{\phi} \in V_k^\star} |\langle \boldsymbol{\delta}, \boldsymbol{\phi} \rangle|^2, \tag{7.1}$$

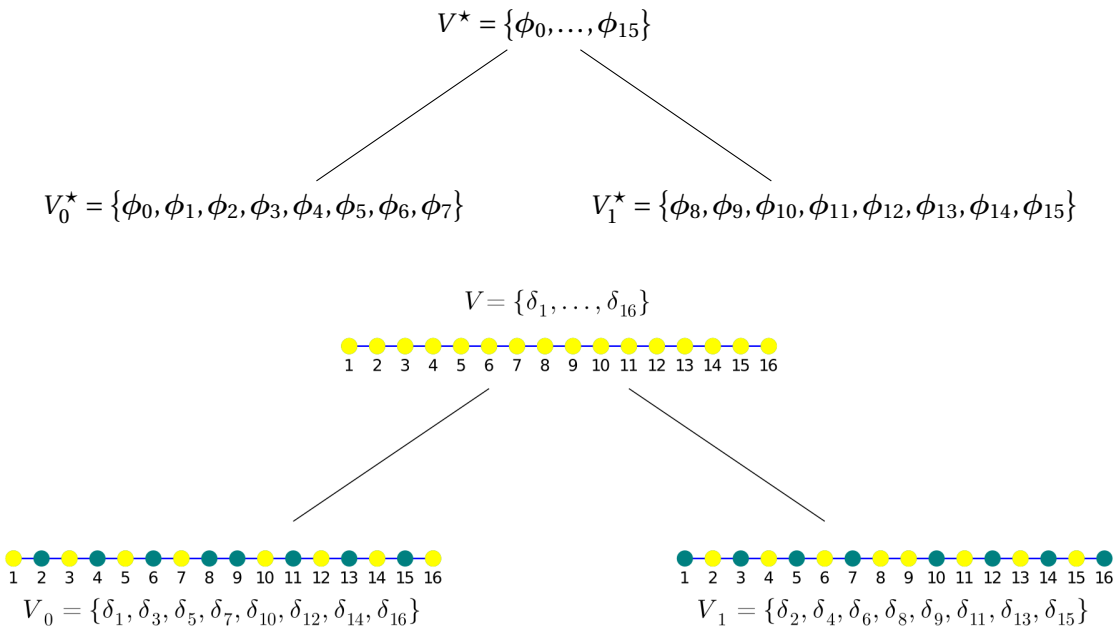


FIGURE 7.1. One-level pair-clustering ($K = 2$) result of P_{16}.

where $\langle \cdot, \cdot \rangle$ is the standard inner product in $\mathbb{R}^N$. In particular,

$$\alpha_{\mathrm{PC}}(V, V^\star) = \sum_{\delta \in V, \phi \in V^\star} |\langle \delta, \phi \rangle|^2 = \sum_{\phi \in V^\star} \|\phi\|^2 = N.$$

Denote the feasible partition set as

$$\mathscr{U}(V; N_0, \ldots, N_{K-1}) := \left\{ (V_0, \ldots, V_{K-1}) \,\middle|\, \bigsqcup_{k=0}^{K-1} V_k = V; |V_k| = N_k, k = 0 : K-1 \right\}.$$

Now we need to solve the following optimization problem for a given partition of $V^\star = \bigsqcup_{k=0}^{K-1} V_k^\star$:

$$(V_0, \ldots, V_{K-1}) = \arg \max_{(V_0, \ldots, V_{K-1}) \in \mathscr{U}(V; N_0, \ldots, N_{K-1})} \sum_{k=0}^{K-1} \alpha_{\mathrm{PC}}(V_k, V_k^\star). \tag{7.2}$$

This is a discrete optimization problem. In general, it is not easy to find the global optimal solution except for the case of $K = 2$. When $K = 2$, we can find the desired partition of V by the following greedy algorithm:

1) Compute score(δ) := $\alpha_{\mathrm{PC}}(\{\delta\}, V_0^\star) - \alpha_{\mathrm{PC}}(\{\delta\}, V_1^\star)$ for each $\delta \in V$.
2) Select N_0 δ's in V that give the largest N_0 values of score$(\cdot)$, set them as V_0, and set $V_1 = V \setminus V_0$.

Figure 7.1, as an example, shows the one-level pair-clustering result ($K = 2$) of P_{16}, given $V^\star = \{\phi_l\}_{l=0:15}$ is evenly partitioned into $V_0^\star = \{\phi_l\}_{l=0:7}$ and $V_1^\star = \{\phi_l\}_{l=8:15}$. The standard basis vectors V is then evenly partitioned into two sets as shown in the figure, i.e., $V_0 = \{\boldsymbol{\delta}_1, \boldsymbol{\delta}_3, \boldsymbol{\delta}_5, \boldsymbol{\delta}_7, \boldsymbol{\delta}_{10}, \boldsymbol{\delta}_{12}, \boldsymbol{\delta}_{14}, \boldsymbol{\delta}_{16}\}$ and $V_1 = \{\boldsymbol{\delta}_2, \boldsymbol{\delta}_4, \boldsymbol{\delta}_6, \boldsymbol{\delta}_8, \boldsymbol{\delta}_9, \boldsymbol{\delta}_{11}, \boldsymbol{\delta}_{13}, \boldsymbol{\delta}_{15}\}$. In other words, the indices of standard basis vectors in V_0 are odd up to the middle and switch to even till the end, while the indices of standard basis vectors in V_1 does the other way around. Actually, we have the following theorem for the one-level pair-clustering ($K = 2$) result on P_N.

THEOREM 7.1.1. *Assume the (unnormalized) graph Laplacian eigenvectors of the unweighted path graph P_N as defined in Eq.* (2.2) *is partitioned into $K = 2$ clusters, i.e.,*

$$V_0^\star = \{\phi_l\}_{l=0:\lfloor \frac{N-1}{2} \rfloor} \quad \textit{and} \quad V_1^\star = \{\phi_l\}_{l=\lfloor \frac{N-1}{2} \rfloor + 1:N-1}.$$

Then, the solutions of Eq. (7.2) *would be*

1. *If N is even, i.e., $N = 2m$, and m is also even, i.e., $m = 2n$, then $V_0 = \{\boldsymbol{\delta}_{2x-1}\}_{x=1:n} \cup \{\boldsymbol{\delta}_{2x}\}_{x=n+1:m}$ and $V_1 = V \setminus V_0$.*
2. *If N is even, i.e., $N = 2m$, but m is odd, i.e., $m = 2n-1$, then there are two solutions of Eq.* (7.2)*:*
   (a) *$V_0 = \{\boldsymbol{\delta}_{2x-1}\}_{x=1:n} \cup \{\boldsymbol{\delta}_{2x}\}_{x=n+1:m}$ and $V_1 = V \setminus V_0$.*
   (b) *$V_0 = \{\boldsymbol{\delta}_{2x-1}\}_{x=1:n-1} \cup \{\boldsymbol{\delta}_{2x}\}_{x=n:m}$ and $V_1 = V \setminus V_0$.*
3. *If N is odd, i.e., $N = 2m-1$, then $V_0 = \{\boldsymbol{\delta}_{2x-1}\}_{x=1:m}$ and $V_1 = V \setminus V_0 = \{\boldsymbol{\delta}_{2x}\}_{x=1:m-1}$.*

See Appendix A.4 for the proof.

On the other hand, when $K > 2$, we can find a *local optimum* by the similar strategy:

1) Compute the values $\alpha_{\mathrm{PC}}(\{\boldsymbol{\delta}\}, V_0^\star)$ for each $\boldsymbol{\delta} \in V$.
2) Select N_0 $\boldsymbol{\delta}$'s giving the largest N_0 values, and set them as V_0.
3) Compute the values $\alpha_{\mathrm{PC}}(\{\boldsymbol{\delta}\}, V_1^\star)$ for each $\boldsymbol{\delta} \in V \setminus V_0$, select N_1 $\boldsymbol{\delta}$'s giving the largest N_1 values, and set them as V_1.
4) Repeat the above process to produce $V_2, \ldots, V_{K-1}$.

While this greedy strategy does not reach the global optimum of Eq. (7.2), we find that empirically the algorithm attains a reasonably large value of the objective function. We note that our one-level pair-clustering problem is a particular example of the so-called *submodular welfare problem* [**88**] with cardinality constraints; however, we will not pursue this direction for a general $K > 2$ with the one-level pair clustering. Rather, we will apply it with $K = 2$ in a hierarchical manner, which will be discussed next.

7.2. Hierarchical Pair-Clustering

In order to build a multiscale graph wavelet packet dictionary, we develop a hierarchical (i.e., multi-level) version of the pair-clustering algorithm. First, let us assume that the hierarchical bipartition tree of $V^\star$ is already computed using the same algorithm discussed in Section 6.1. We now begin with level $j = 0$ where $V_0^{(0)}$ is simply $V = \{\boldsymbol{\delta}_1, \boldsymbol{\delta}_2, \cdots, \boldsymbol{\delta}_N\}$ and $V_0^{\star(0)}$ is $V^\star = \{\phi_0, \phi_1, \cdots, \phi_{N-1}\}$. Then, we perform one-level pair-clustering algorithm ($K = 2$) to get $\left(V_0^{\star(1)}, V_0^{(1)}\right)$ and then $\left(V_1^{\star(1)}, V_1^{(1)}\right)$. We iterate the above process to generate paired clusters $\left(V_k^{\star(j)}, V_k^{(j)}\right)$, $j = 0 : J$, $k = 0 : K^j - 1$. Note that the hierarchical pair-clustering algorithm ensures nestedness in both the primal node domain V and the dual/eigenvector domain $V^\star$.

7.3. Generating the NGWP Dictionary

Once we generate two hierarchical bipartition trees $\left\{V_k^{(j)}\right\}$ and $\left\{V_k^{\star(j)}\right\}$, we can proceed to generate the NGWP vectors $\left\{\Psi_k^{(j)}\right\}$ that are necessary to form an NGWP dictionary. For each $\boldsymbol{\delta}_l \in V_k^{(j)}$, we first compute the orthogonal projection of $\boldsymbol{\delta}_l$ onto the span of $V_k^{\star(j)}$, i.e., span$\left(\Phi_k^{(j)}\right)$ where $\Phi_k^{(j)}$ are those eigenvectors of $L(G)$ belonging to $V_k^{\star(j)}$. Unfortunately, $\Phi_k^{(j)}\left(\Phi_k^{(j)}\right)^\mathsf{T}\boldsymbol{\delta}_l$ and $\Phi_k^{(j)}\left(\Phi_k^{(j)}\right)^\mathsf{T}\boldsymbol{\delta}_{l'}$ are not mutually orthogonal for $\boldsymbol{\delta}_l, \boldsymbol{\delta}_{l'} \in V_k^{(j)}$ in general. Hence, we need to perform orthogonalization of the vectors $\left\{\Phi_k^{(j)}\left(\Phi_k^{(j)}\right)^\mathsf{T}\boldsymbol{\delta}_l\right\}_l$. We use the *modified Gram-Schmidt with $\ell^p (0 < p < 2)$ pivoting orthogonalization* (MGSLp) [**13**] to generate the orthonormal graph wavelet packet vectors associated with $V_k^{\star(j)}$ (and hence also $V_k^{(j)}$).

In particular, we implemented a simplified version (i.e., Algorithm 3) of the modified Gram-Schmidt with mixed ℓ^2-ℓ^p $(0 < p < 2)$ pivoting algorithm in [**13**]. Our version skips the step of computing the largest ℓ^2 norm and picking the parameter λ (a notation used in [**13**]) to increase the numerical stability. Instead, we directly set up a tolerance parameter, i.e., tol in Algorithm 3, for the robustness. On the other hand, we keep the ℓ^p $(0 < p < 2)$ pivoting portion in MGS (i.e., always perform the orthogonalization

process of the vector with minimum ℓ^p-norm in the candidate pool), which nicely preserves the sparsity of the obtained wavelet-like vectors after the orthogonalization process.

Algorithm 3: Modified Gram-Schmidt Orthogonalization with ℓ^p pivoting (MGSLp)

Input: List of unit vectors $v = [v_1, \ldots, v_m] \in \mathbb{R}^{N\times m}$; norm parameter $0 < p < 2$ (default: 1); error tolerance tol (default: 1e-12)
Output: List of orthonormal vectors $q = [q_1, \ldots, q_r] \in \mathbb{R}^{N\times r}$ where $r = \text{rank}(v)$

```
q = ∅                                                  // initialize the output list
w = [‖v_1‖_p, ..., ‖v_m‖_p]
for i = 1 : m do
    k = i − 1 + findmin(w)                             // find the minimum ℓ^p-norm index
    swap(v_i, v_k)                                     // pivoting
    if ‖v_i‖_2 < tol then
        break                                          // check linear dependency
    ṽ = v_i / ‖v_i‖_2
    w = ∅                                              // re-initialize the ℓ^p-norm vector
    for j = i + 1 : m do
        v_j = v_j − (ṽ^T · v_j) ṽ
        w ← w ∪ {‖v_j‖_p}
    q ← q ∪ {ṽ}
return q
```

This MGSLp algorithm tends to generate localized orthonormal vectors because the ℓ^p-norm[1] pivoting promotes sparsity. We refer to the graph wavelet packet dictionary $\left\{\Psi_k^{(j)}\right\}_{j=0:J;\,k=0:K^j-1}$ generated by this algorithm as the *Pair-Clustering Natural Graph Wavelet Packet* (PC-NGWP) dictionary.

Let us now briefly discuss the performance of the PC-NGWP dictionary on the same examples in Section 6.2, i.e., P_{512} and $P_7 \times P_3$, without displaying figures to save pages. We essentially obtained the similar wavelet packet vectors in both cases as those shown in Figures 6.3 and 6.4 using the VM-NGWP dictionaries; yet they are not exactly the same: the localization of those PC-NGWP vectors in the primal node domain is worse (e.g., with larger sidelobes) than that of the VM-NGWP vectors mainly due to the MGSLp orthogonalization procedure (even if it promoted sparsity).

[1] We typically set $p = 1$ here, and in fact, that setting was used in all the numerical experiments with the PC-NGWP dictionary in this dissertation.

7.4. Computational Complexity

At each $V_k^{\star(j)}$ of the hierarchical bipartition tree of the dual graph $G^\star$, the orthogonal projection of the standard basis vectors in $V_k^{(j)}$ onto span$\left(\Phi_k^{(j)}\right)$ and the MGSLp procedure are the two main computational burdens for our PC-NGWP dictionary construction. Specifically, the orthogonal projection costs $O\left(N\cdot(N_k^j)^2+N\cdot N_k^j\right)$ and the MGSLp costs $O\left(2N\cdot(N_k^j)^2\right)$. Hence, the dominating cost for this procedure is $O\left(3N\cdot(N_k^j)^2\right)$ for each (j,k). And we need to sum up this cost on all the tree nodes. Let us analyze the special case of the perfectly balanced and fully developed bipartition tree with $N=2^{j_{\max}}$ as we did for the VM-NGWP in Section 6.3. In this case, the bipartition tree has $1+j_{\max}$ levels, and $N_k^j=2^{j_{\max}-j}$, $k=0:2^j-1$. So, for the jth level, using Eq. (6.2), we have $O(3N^3\cdot 2^{-j})$. Finally, by summing this from $j=1$ to $j_{\max}-1$ (again, no computation is needed at the root and the bottom levels), the total cost for PC-NGWP dictionary construction in this ideal case is: $O(3N^3\cdot(1-2/N))\approx O(3N^3)$. So, it still requires $O(N^3)$ operations; the difference from that of the VM-NGWP is the constants, i.e., 3 (PC-NGWP) vs $1+1000/3\approx 334$ (the worst case VM-NGWP).

7.5. L_{sym} Version of the *PC-NGWP*

As explained in Section 6.4, we can also construct the L_{sym} version of $G^\star=G^\star(V^\star,E^\star,W^\star)$ via one of the *viable* metrics in Table 5.1, i.e., $d_{\text{ROT}}^{(1)}$, d_{sROT} (for trees), d_{HAD}, and d_{DAG}. After that, we can build the L_{sym} *version of the Pair-Clustering Natural Graph Wavelet Packet* (PC-NGWP-L_{sym}) dictionary by following the same procedures in Section 7.2 and Section 7.3.

CHAPTER 8

Natural Graph Wavelet Packets using *Lapped Orthogonal Projections*

In Chapters 6 and 7, we constructed the natural graph wavelet packets based on hierarchical bipartitioning of the dual graph $G^\star = G^\star(V^\star, E^\star, W^\star)$. Specifically, we used the sign of the Fiedler vectors[1] to bipartition the dual graph recursively. Those partitions are the so-called *hard* bipartitions, i.e., each node is only assigned to one cluster or the other. Moreover, since the eigenvectors (i.e., $V^\star$) form an ONB of $\mathbb{R}^N$, the hard bipartition of $V_k^{\star(j)}$ nicely yields two subspaces $\text{span}\left(V_{k'}^{\star(j+1)}\right)$ and $\text{span}\left(V_{k'+1}^{\star(j+1)}\right)$ such that

$$\text{span}\left(V_{k'}^{\star(j+1)}\right) \bigoplus \text{span}\left(V_{k'+1}^{\star(j+1)}\right) = \text{span}\left(V_k^{\star(j)}\right) \quad \text{and} \quad \text{span}\left(V_{k'}^{\star(j+1)}\right) \perp \text{span}\left(V_{k'+1}^{\star(j+1)}\right),$$

where $V_k^{\star(j)}$ contains at least two eigenvectors. Therefore, any dual graph signal[2] $\boldsymbol{g}$ supported on $V_k^{\star(j)}$ can be sharply split into two pieces, each of which is supported on either $V_{k'}^{\star(j+1)}$ or $V_{k'+1}^{\star(j+1)}$, by orthogonal projections (or simply restrictions in this case). This orthogonal splitting property is essential in the following natural graph wavelet packet constructions. But the question is whether we can bipartition $V_k^{\star(j)}$ in a *soft or lapped* way by allowing some spillovers across the cutoff boundary between $V_{k'}^{\star(j+1)}$ and $V_{k'+1}^{\star(j+1)}$ such that their associated supporting subspaces still satisfy the orthogonal splitting property while the dual graph signal on $V_k^{\star(j)}$ is split in a more *gentle or smooth* manner near the cutoff boundary.

Recall the local cosine basis dictionary [**15**], [**54**, Sec. 8.4.3] is built based on such desired lapped partitions via the *orthogonal folding/unfolding operators* and the *smooth orthogonal projectors* on the time domain (i.e., $\mathbb{R}$). Thus, we propose a strategy in this chapter to generalize these operators to the general graph settings. Our strategy is a variant of the method discussed in [**83**] and it is easier to implement with little cost.

[1] Without explicit saying, the Fiedler vector used in bipartitioning is associated with the random-walk normalized graph Laplacian matrix in Eq. (2.1). Based on the context, it could be the Laplacian of the primal graph G or the dual graph $G^\star$.
[2] See the end of Section 5.1 for the definitions of the dual graph signal and its support.

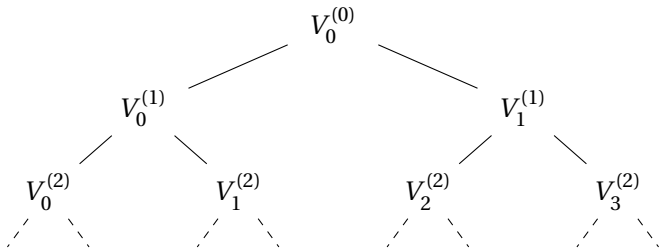


FIGURE 8.1. The hierarchical bipartition tree of the primal graph nodes $V \equiv V_0^{(0)}$ via the Fiedler vectors, which is used in the HGLET construction.

The HGLET [**34**] is a graph version of the Hierarchical Block DCT dictionary, which is based on the hierarchical (hard) bipartition tree of a primal graph G. By applying the orthogonal unfolding operators on the *primal graph* to the HGLET basis vectors, we can obtain a *lapped* version of HGLET.

On the other hand, the VM-NGWP dictionary can be viewed as a graph version of the *Shannon* wavelet packet dictionary [**54**, Sec. 8.1.2], which is based on the hard bipartition tree of $V^\star$ as shown in Figure 6.1. Bipartitioning $G^\star$ recursively but *smoothly with overlaps* via the smooth orthogonal projectors on the *dual graph*, we can analogously build a graph version of the *Meyer* wavelet packet dictionary [**54**, Sec. 7.2.2, 8.4.2].

Since our goal is to construct such lapped bipartitions in a hierarchical manner, we first introduce some multiscale notation of graphs. On the dual graph $G^\star$, we continue using the notation introduced in Section 6.1, i.e., $\left\{V_k^{\star(j)}\right\}_{j=0:J;k=0:K^j-1}$. On the primal graph G, we introduce the notation by removing the "$\star$" as follows. Let $V_0^{(0)} := V$ be the node set of the primal graph G. Suppose we get the hierarchical hard bipartition tree of $V_0^{(0)}$ as shown in Figure 8.1. Then, the hierarchical subgraphs can be represented by $\left\{G_k^{(j)}\right\}_{j=0:J;k=0:K^j-1}$ with the corresponding node sets $\left\{V_k^{(j)}\right\}_{j=0:J;k=0:K^j-1}$. Note that, unlike in Section 7.2, the bipartitions on the primal graph are achieved by using the Fiedler vectors of each subgraph of G instead of the pair-clustering algorithm in Eq. (7.2).

Now, let us review the smooth orthogonal projector on the unweighted path graph P_N, where we closely follow [**69**] and introduce some important notation and concepts for later use in the general graph case.

8.1. Smooth Orthogonal Projector on P_N

Given a graph signal $\boldsymbol{f} \in \mathbb{R}^N$ on $P_N(V,E)$, we can easily split the signal brutally into two pieces by the restriction operator (or matrix) $\boldsymbol{\chi}_{V_k} \in \mathbb{R}^{N\times N}$ each of which is supported on V_k, where V_k is the node set of the subgraph P_{N_k} for $k = 0,1$ such that V_0, V_1 are disjoint and $V_0 \bigcup V_1 = V$, and $\boldsymbol{\chi}_{V_k} \in \mathbb{R}^{N\times N}$ is a diagonal matrix with $\left(\boldsymbol{\chi}_{V_k}\right)_{x,x} = 1$ if $x \in V_k$, $= 0$ otherwise (for $x = 1:N$). With a slight abuse of notation as we did in Chapter 7, let V also represents a collection of the standard basis vectors in $\mathbb{R}^N$, i.e., $V := \{\boldsymbol{\delta}_x\}_{x\in V}$, and in parallel $V_k := \{\boldsymbol{\delta}_x\}_{x\in V_k}$. Then such brutal splitting by the restriction operator $\boldsymbol{\chi}_{V_k}$ is equivalent to projecting the signal $\boldsymbol{f} \in \mathbb{R}^N$ onto the subspace $\mathrm{span}(V_k)$ $(k = 0,1)$ thanks to the fact $\mathrm{span}(V_0) \bigoplus \mathrm{span}(V_1) = \mathrm{span}(V) = \mathbb{R}^N$ and $\mathrm{span}(V_0) \perp \mathrm{span}(V_1)$.

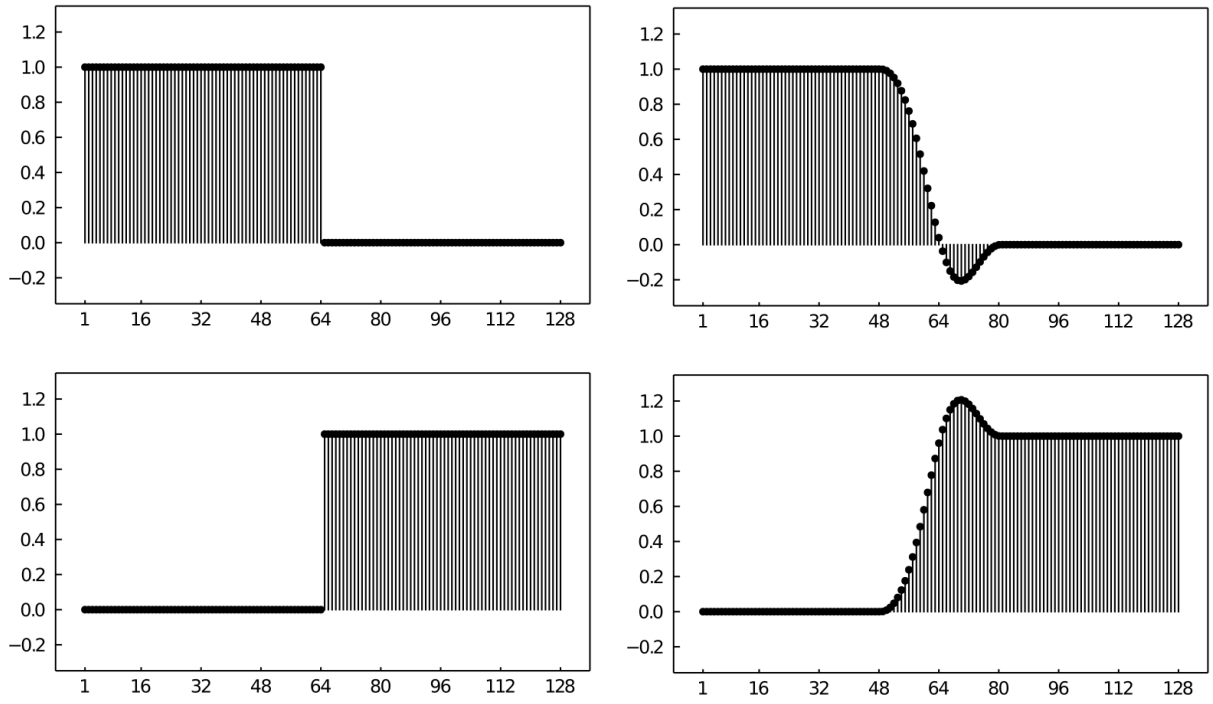
On the other hand, instead of such hard splitting, we would like to split the signal $\boldsymbol{f}$ *smoothly* into two pieces each of which is supported on V_k (with some overlaps). We define two subspaces Ω_k associated with V_k $(k = 0,1)$ such that $\Omega_0 \bigoplus \Omega_1 = \mathbb{R}^N$ and $\Omega_0 \perp \Omega_1$. To achieve that, Coifman and Meyer [**15**] introduced the following *smooth orthogonal projector*[3] from $\mathbb{R}^N$ into $\Omega_k = P_{V_k}\left(\mathbb{R}^N\right)$ (also as discussed in [**69**, **83**]):

$$P_{V_k}\boldsymbol{f} := U^{\mathsf{T}}\boldsymbol{\chi}_{V_k}U\boldsymbol{f}, \qquad k = 0,1. \tag{8.1}$$

It consists of three operators: 1) the *orthogonal folding operator* $U \in \mathbb{R}^{N\times N}$; 2) the restriction operator $\boldsymbol{\chi}_{V_k} \in \mathbb{R}^{N\times N}$; and 3) the *orthogonal unfolding operator* $U^{\mathsf{T}} \in \mathbb{R}^{N\times N}$, i.e., the transpose of the orthogonal folding operator. P_{V_k} works as a smooth version of $\boldsymbol{\chi}_{V_k}$. For example, Figure 8.2 shows the bipartitions of a constant signal $\boldsymbol{f} \equiv \mathbf{1}$ into $V_0 = \{\boldsymbol{\delta}_x\}_{x=1:64}$ and $V_1 = \{\boldsymbol{\delta}_x\}_{x=65:128}$ on P_{128}, which are done by $\boldsymbol{\chi}_{V_k}$ (hard) and P_{V_k} (soft), respectively. Note that the negative dent appeared in the top figure of Figure 8.2b is necessary for the orthogonality between P_{V_0} and P_{V_1}.

Before moving on to the definition of U, we first introduce the *action region* $(\beta-\eta, \beta+\eta) \subset (1,N) \subset \mathbb{R}$, where the *cutoff boundary* $\beta \in (1,N)$ and the *action region bandwidth* $\eta \in \left(0, \min\left(\beta-1, N-\beta\right)\right]$. In particular, $(\beta-\eta,\beta)$ is the *negative action region* and $(\beta,\beta+\eta)$ is the *positive action region*. We denote the node set of P_N within the negative action region by $R^- := V \cap (\beta-\eta,\beta)$ and the node set of P_N within the positive action region by $R^+ := V \cap (\beta,\beta+\eta)$. With these concepts in place, we can define the set of *reflection triples* $\left\{(v_i^-, v_i^+, r_i)\right\}_{i=1:N_p}$, where 1) v_i^- is the i-th closest node to β in R^-; 2) v_i^+ is the i-th

[3] They actually introduced a more broadly concept on $L^2(\mathbb{R})$, but we only focus on the discrete setting in this dissertation.



(a) Hard bipartition by χ_{V_k} on P_{128}.

(b) Soft (Lapped) bipartition by P_{V_k} on P_{128}.

FIGURE 8.2. Splitting the constant signal into $V_0 = \{\boldsymbol{\delta}_x\}_{x=1:64}$ and $V_1 = \{\boldsymbol{\delta}_x\}_{x=65:128}$, by the restriction operators (a), and by the smooth orthogonal projectors (b).

closest node to β in R^+; 3) v_i^- and v_i^+ are a *reflection pair* about β such that $\beta - v_i^- = v_i^+ - \beta$, and their *reflection radius* is defined by $r_i := \beta - v_i^- = v_i^+ - \beta$; and 4) the total number of reflection pairs within the action region is given by $N_p := \min\left(|R^-|, |R^+|\right)$. Note that $0 < r_i < \eta$ for $i = 1 : N_p$, and $r_1 < r_2 < \cdots < r_{N_p}$.

Then, the orthogonal folding operator $U := U(s, \beta, \eta) \in \mathbb{R}^{N \times N}$ associated with the action region $(\beta - \eta, \beta + \eta)$ and the set of reflection triples $\left\{(v_i^-, v_i^+, r_i)\right\}_{i=1:N_p}$ is defined by modifying the identity matrix I_N as below

$$
\begin{aligned}
U_{v_i^-, v_i^-} &:= s\left(\frac{r_i}{\eta}\right), \qquad U_{v_i^-, v_i^+} := -s\left(-\frac{r_i}{\eta}\right), \\
U_{v_i^+, v_i^-} &:= s\left(-\frac{r_i}{\eta}\right), \qquad U_{v_i^+, v_i^+} := s\left(\frac{r_i}{\eta}\right).
\end{aligned} \tag{8.2}
$$

Note that if the boundary β happens to be a node of P_N, the diagonal entry $U_{\beta,\beta} = 1$. The function $s(t)$ above is the so-called *rising cutoff function*, which is a smooth version of the Heaviside step function

i	1	2	3	4	5	6	7	8
v_i^-	64	63	62	61	60	59	58	57
v_i^+	65	66	67	68	69	70	71	72
r_i	0.5	1.5	2.5	3.5	4.5	5.5	6.5	7.5
i	9	10	11	12	13	14	15	16
v_i^-	56	55	54	53	52	51	50	49
v_i^+	73	74	75	76	77	78	79	80
r_i	8.5	9.5	10.5	11.5	12.5	13.5	14.5	15.5

TABLE 8.1. The reflection triples on P_{128} with $\beta = 64.5$ and $\eta = 16$.

such that

$$|s(t)|^2 + |s(-t)|^2 = 1 \quad \text{for all } t \in \mathbb{R}, \quad \text{and} \quad s(t) = \begin{cases} 0, & \text{if } t \le -1, \\ \\ 1, & \text{if } t \ge 1. \end{cases}$$

As mentioned in [**69**], a typical example of $s \in C^1(\mathbb{R})$ is the following iterated sine function[4]:

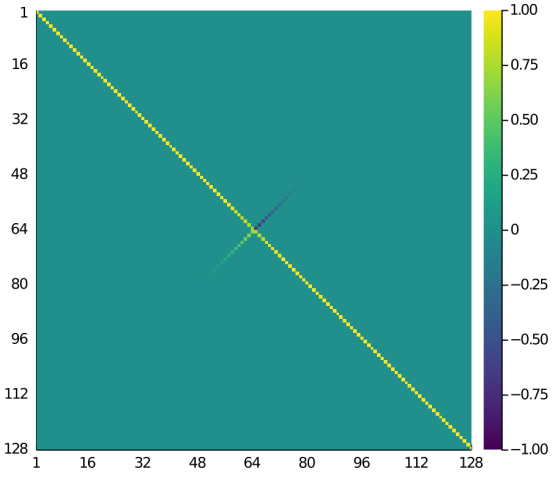
$$s(t) = \begin{cases} 0, & \text{if } t \le -1, \\ \\ \sin[\frac{\pi}{4}(1+\sin\frac{\pi}{2}t)], & \text{if } |t| < 1, \\ \\ 1, & \text{if } t \ge 1. \end{cases}$$

We can easily verify that U is an orthogonal matrix, i.e., $U^{\mathsf{T}}U = I_N$.

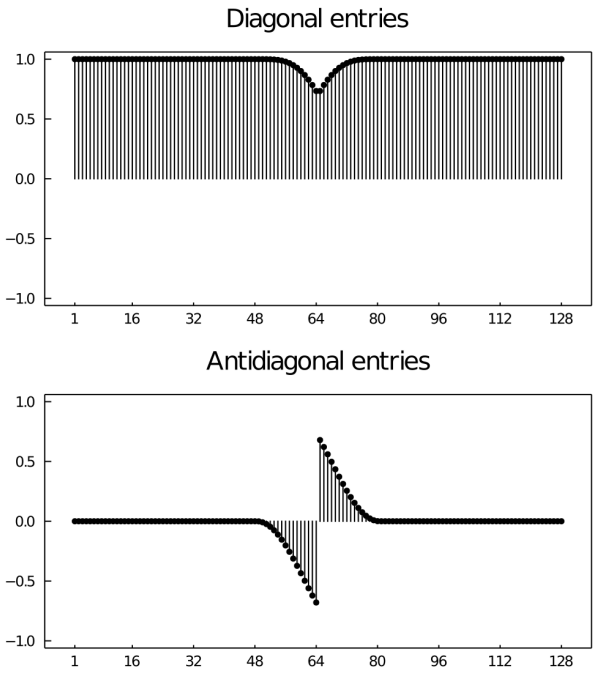
Now, let us continue the example of P_{128} with $V_0 = \{\boldsymbol{\delta}_x\}_{x=1:64}$ and $V_1 = \{\boldsymbol{\delta}_x\}_{x=65:128}$ by setting $\beta = 64.5$ and $\eta = 16$. Then, the action region is $(48.5, 80.5)$ and the reflection triples are listed in Table 8.1. The entries of the orthogonal folding operator U are as shown in Figure 8.3. As one can see from the figures, the orthogonal folding operator U makes a signal *locally odd* for the negative action region $(\beta-\eta, \beta)$ and *locally even* for the positive action region $(\beta, \beta+\eta)$ [**69**].

In summary, the procedures of generating such desired lapped bipartition subspaces Ω_k of $\text{span}(V) = \mathbb{R}^N$ are: 1) setting the cutoff boundary β, e.g., $\beta = \frac{N+1}{2}$, and setting the action region bandwidth η, e.g.,

[4] which will be used in all the numerical experiments throughout this dissertation.



(a) Heatmap plot of U



(b) Diagonal entries and antidiagonal entries of U

FIGURE 8.3. The orthogonal folding operator $U(s,\beta,\eta)$ on P_{128} with $\beta = 64.5$ and $\eta = 16$.

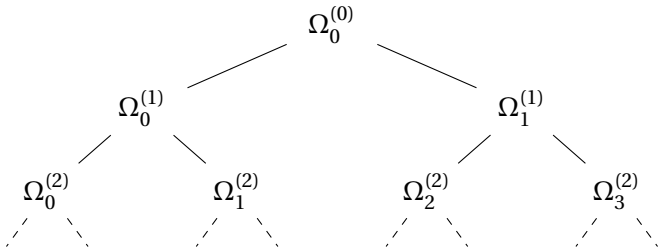
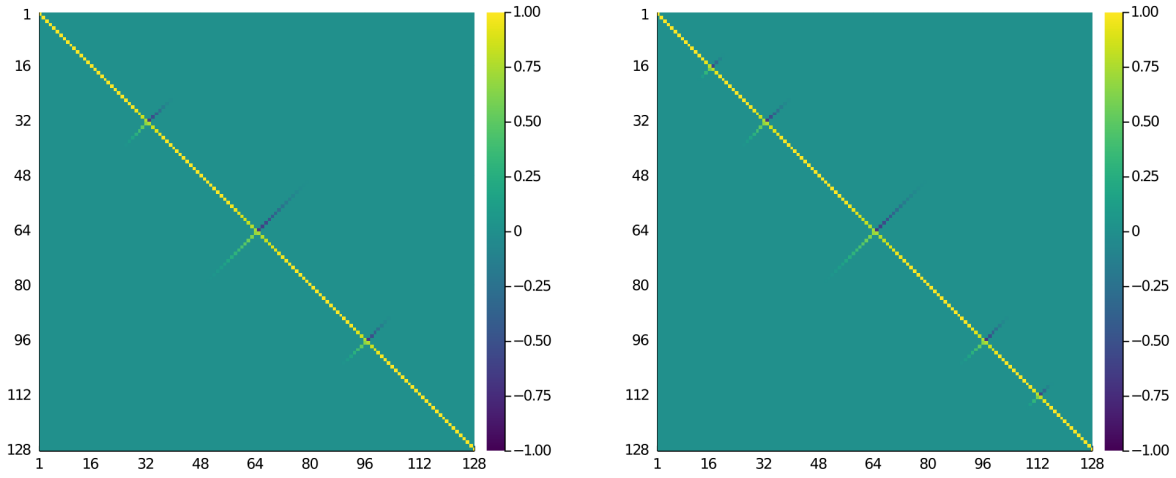


FIGURE 8.4. The hierarchical bipartition subspaces of the primal graph supporting space $\mathrm{span}(V) \equiv \Omega_0^{(0)} \equiv \mathbb{R}^N$, which is the soft/lapped version of Figure 8.1.

$\eta = \frac{1}{8}N$; 2) finding the set of reflection triples $\{(v_i^-, v_i^+, r_i)\}_{i=1:N_p}$ within the action region; 3) assembling the orthogonal folding operator U; 4) computing the smooth orthogonal projector P_{V_k} based on Eq. (8.1); and 5) projecting $\mathbb{R}^N$ into the subspace $\Omega_k = P_{V_k}(\mathbb{R}^N)$ (for $k = 0, 1$).

The above procedures also can be done recursively, yielding a hierarchical family of subspaces $\Omega_k^{(j)}$ of $\mathrm{span}(V) = \mathbb{R}^N$, as shown in Figure 8.4. Note that any two subspaces $\Omega_{k_1}^{(j)}$ and $\Omega_{k_2}^{(j)}$ at the same level are



(a) Heatmap plot of $U^{(2)}$

(b) Heatmap plot of $U^{(3)}$

FIGURE 8.5. Orthogonal folding operator $U^{(j)}$ on P_{128} with $j = 2$ (a) and $j = 3$ (b). The cutoff boundaries are set to be the middle point of each subgraph $G_k^{(j)}$ and the action region bandwidths $\eta^{(j)} = \frac{1}{8}\left|V_k^{(j)}\right|$.

orthogonal to each other. Moreover, if $\dim\left(\Omega_k^{(j)}\right) > 1$ and $\Omega_{k'}^{(j+1)}$, $\Omega_{k'+1}^{(j+1)}$ are the two children of $\Omega_k^{(j)}$,

$$\Omega_k^{(j)} = \Omega_{k'}^{(j+1)} \bigoplus \Omega_{k'+1}^{(j+1)}, \qquad j = 0 : J-1 \text{ and } k = 0 : K^j - 1.$$

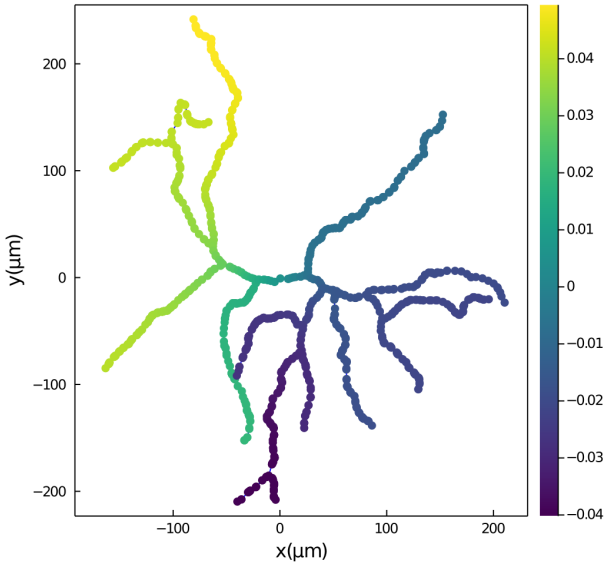
Each $\Omega_k^{(j)}$ is associated with: 1) the supporting node set $V_k^{(j)}$; 2) the j-th level orthogonal folding operator $U^{(j)}$ with the j-th level action region bandwidth $\eta^{(j)}$; and 3) the corresponding smooth orthogonal projector $P_{V_k^{(j)}}$. In particular, $\Omega_k^{(j)} = P_{V_k^{(j)}}\left(\mathbb{R}^N\right)$. Therefore, the previous one-level notation $V \equiv V_0^{(0)}$, $V_k \equiv V_k^{(1)}$, $U \equiv U^{(1)}$ and $P_{V_k} \equiv P_{V_k^{(1)}}$ (for $k = 0, 1$). Note that it becomes problematic when the action regions of different levels start to overlap, which is also discussed in [**83**]. It means that one node could be the reflections of multiple nodes at different levels. These one-to-many mappings break the orthogonality of the folding operator $U^{(j)}$. As a simple remedy to this issue, we keep the ancestor's one-to-one reflection correspondence if a node has multiple reflection pairs across levels. In other words, $U^{(j+1)}$ is built upon $U^{(j)}$ without changing its modified columns and rows from $U^{(j-1)}$, for $j = 1 : J$ (where we denote $U^{(0)} \equiv I_N$). Figure 8.5 shows how $U^{(j)}$ on P_{128} looks like at levels $j = 2$ and $j = 3$. As one can see from Figure 8.5b, the entries $U_{x,x}^{(3)}$ and $U_{x,97-x}^{(3)}$ (for $x = 45 : 52$) do not get updated since the nodes within the positive action region of $V_1^{(2)}$, i.e., $x = \{49, 50, 51, 52\}$, have been paired with the reflection points in the positive action region of $V_0^{(0)}$, i.e., $x = \{80, 79, 78, 77\}$, as listed in Table 8.1. We note that this

simple remedy may result in oversized or discontinuous spillovers at $V_k^{(j)}$ with large j, nevertheless it can easily implement and generalize to the general graph settings. Alternative remedies can be found in [**2**, **3**, **83**], but they are either hard to generalize or having an expensive computational cost. For instance, the authors of [**83**] repeatedly use eigendecomposition of $N \times N$ matrices to tackle such issue, which is obviously more expensive than our approach.

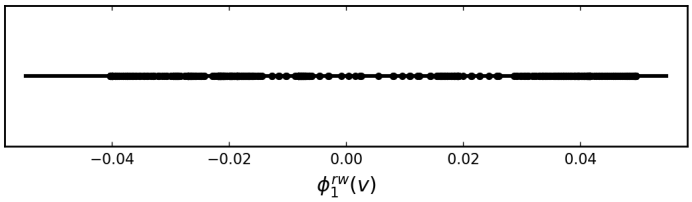
8.2. Smooth Orthogonal Projector on General Graphs

The key of constructing the smooth orthogonal projector is the orthogonal folding operator, and in turn the key of assembling the orthogonal folding operator is to find the proper set of reflection triples $\{(v_i^-, v_i^+, r_i)\}_{i=1:N_p}$. To generalize the smooth orthogonal projector to the general graph settings, we utilize the Fiedler vector to define the reflection triples on graphs.

Given $G(V, E, W)$, the Fiedler vector ϕ_1^{rw} of $L_{\text{rw}}(G)$ provides an embedding of the graph nodes into $\mathbb{R}$ such that the affinities between the nodes in G are best preserved in the 1D embedding space in the sense of minimizing certain loss function as discussed in [**1**]. In other words, we can view each coordinate of ϕ_1^{rw}, i.e., $\phi_1^{\text{rw}}(x)$ $(x = 1 : N)$, as an 1D representation of the node x. As mentioned in Chapters 6 and 7, the spectral clustering theory [**87**] suggests us the value *zero* is a good choice for the cutoff if we want to bipartition the nodes *sharply* into two pieces. Denote the resulting bipartitions $V_0 := \{v \in V \,|\, \phi_1^{\text{rw}}(v) > 0\}$ and $V_1 := V \setminus V_0$. The question is how can we do it in a *soft or smooth* manner by allowing some overlaps. Naturally, we can set the lapped region, i.e., the action region $(\beta - \eta, \beta + \eta)$, to be $\left(-\epsilon \cdot \|\phi_1^{\text{rw}}\|_\infty, \epsilon \cdot \|\phi_1^{\text{rw}}\|_\infty\right)$ in the 1D embedding space, where the cutoff boundary $\beta = 0$, and the action region bandwidth $\eta = \epsilon \cdot \|\phi_1^{\text{rw}}\|_\infty$, and $\epsilon \in (0, 1)$ is the *relative action region bandwidth* that measures the fraction of the action region relative to the whole embedding region of the graph nodes. We use the medium size $\epsilon = 0.3$ as an example consistently in all numerical experiments of this dissertation. Then, we can pair the nodes whose 1D representations are within the negative action region $R_\epsilon^- := \{v^- \in V \,|\, \phi_1^{\text{rw}}(v^-) \in \left(-\epsilon \cdot \|\phi_1^{\text{rw}}\|_\infty, 0\right]\}$ and the nodes whose 1D representations are within the positive action region $R_\epsilon^+ := \{v^+ \in V \,|\, \phi_1^{\text{rw}}(v^+) \in \left(0, \epsilon \cdot \|\phi_1^{\text{rw}}\|_\infty\right)\}$ in the following manner: we start from the cutoff boundary $\beta = 0$ and go both directions simultaneously, and pair the nodes in R_ϵ^- and R_ϵ^+ along the way until the nodes within either R_ϵ^- or R_ϵ^+ are exhausted. In practice, we carry out the following operations: 1) sort the nodes in R_ϵ^- based on the value of $\phi_1^{\text{rw}}(v)$ in decreasing order and denote the



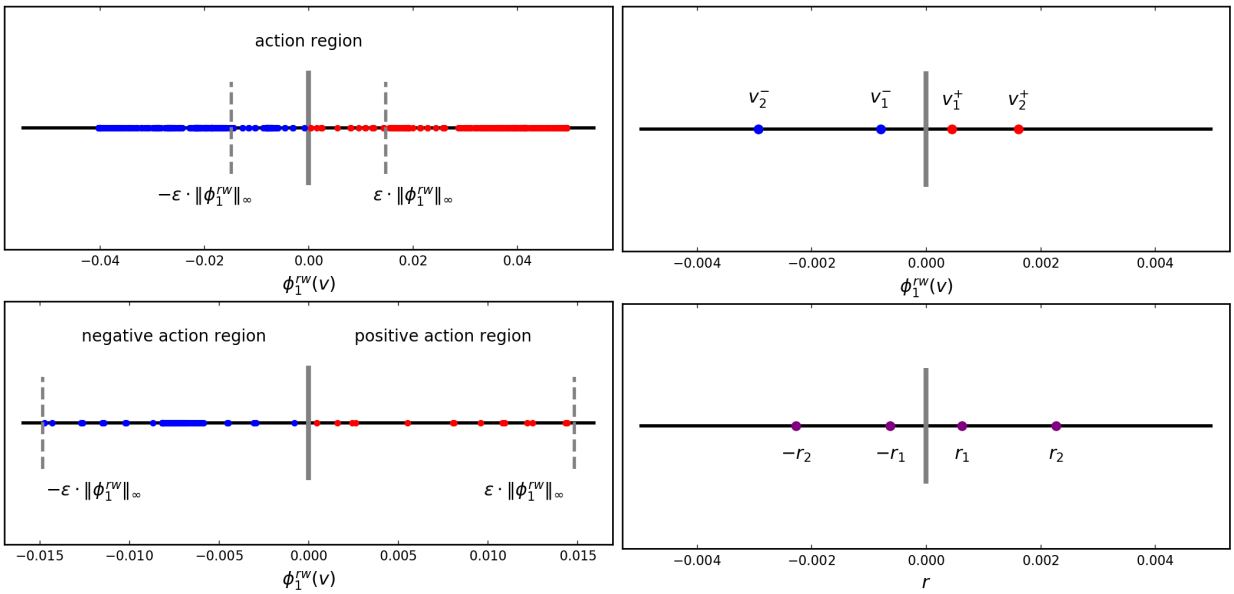
(a) ϕ_1^{rw} on the RGC #100



(b) 1D representation of the RGC #100

FIGURE 8.6. 1D embedding of the nodes of the RGC #100 via ϕ_1^{rw}.

sorted result as $\{v_i^-\}_{i=1:|R_\epsilon^-|}$; 2) sort the nodes in R_ϵ^+ based on the value of $\phi_1^{\mathrm{rw}}(v)$ in increasing order and denote the sorted result as $\{v_i^+\}_{i=1:|R_\epsilon^+|}$; and 3) put together the first N_p pairs $\{(v_i^-, v_i^+)\}_{i=1:N_p}$, where $N_p := \min\left(|R_\epsilon^-|, |R_\epsilon^+|\right)$. If there is a tie while sorting, we use higher dimensional embeddings and sort the nodes with respect to their embedding coordinate values in the lexicographic order. For example, if $\phi_1^{\mathrm{rw}}(v_1^+) = \phi_1^{\mathrm{rw}}(v_2^+)$, we then sort v_1^+ and v_2^+ w.r.t. their corresponding two dimensional embedding coordinates $\left(\phi_1^{\mathrm{rw}}(v_i^+), \phi_2^{\mathrm{rw}}(v_i^+)\right)$ $(i = 1:2)$. After we found the paired nodes, the next step is to define a proper reflection radius r_i for each pair (v_i^-, v_i^+). Normally, $\beta - \phi_1^{\mathrm{rw}}(v_i^-) \neq \phi_1^{\mathrm{rw}}(v_i^+) - \beta$ in the case of general graphs. Therefore, the definition of r_i in P_N cannot be directly adopted. Instead, we redefine the reflection radius of the paired nodes (v_i^-, v_i^+) as $r_i := \frac{1}{2}\left(\phi_1^{\mathrm{rw}}(v_i^+) - \phi_1^{\mathrm{rw}}(v_i^-)\right)$, i.e., half of the distance between v_i^- and v_i^+ in the 1D embedding space. For example, Figure 8.6 shows the 1D embedding result of the



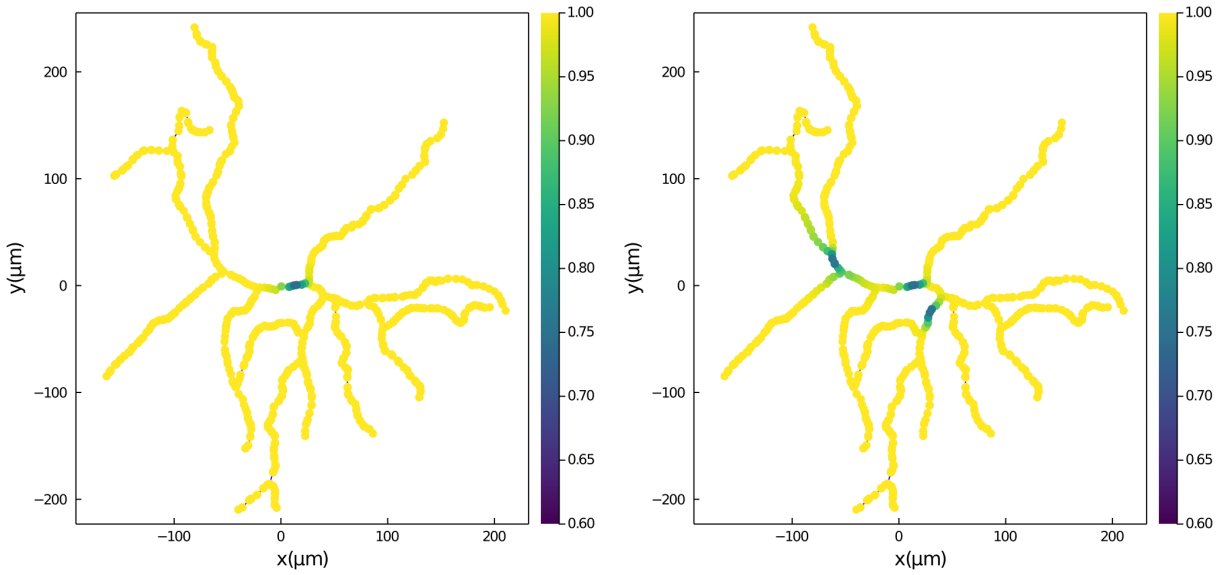
(a) Finding the action region via ϕ_1^{rw} ($\epsilon = 0.3$)

(b) Finding the (first two) reflection triples

FIGURE 8.7. Locating the positive and negative action regions on the 1D embedding of the RGC #100 (a); and finding the (first two) reflection triples, i.e., (v_i^-, v_i^+, r_i) $(i = 1,2)$, near the cutoff boundary $\beta = 0$ (b).

RGC #100 graph discussed in Section 3.2 via the Fiedler vector ϕ_1^{rw}. Figure 8.7 further demonstrates how we find the reflection triples based on the above procedures.

With the reflection triples $\{(v_i^-, v_i^+, r_i)\}_{i=1:N_p}$ in place, we can assemble the orthogonal folding operator U on graphs by making the same modifications of I_N in Eq. (8.2). Consequently, the smooth orthogonal projector on graphs P_{V_k} $(k = 0,1)$ can be derived based on Eq. (8.1). Moreover, the subspaces Ω_k $(k = 0,1)$ associated with V_k (with spillovers) can be obtained by $\Omega_k = P_{V_k}(\mathbb{R}^N)$. So far, we have demonstrated the construction of the smooth orthogonal projector on graphs in the one-level setting. Of course, it can be done recursively as in Section 8.1, yielding: 1) a set of multiscale orthogonal folding operators $\left\{U^{(j)}\right\}_{j=1:J}$ with the uniform relative action bandwidth ϵ; 2) a hierarchical tree of smooth orthogonal projectors $\left\{P_{V_j^{(k)}}\right\}_{j=0:J;k=0:K^j-1}$; and 3) a hierarchical tree of subspaces $\left\{\Omega_k^{(j)}\right\}_{j=0:J;k=0:K^j-1}$ as shown in Figure 8.4. As an example, Figure 8.8 shows the diagonal entries of the resulting orthogonal folding operator $U^{(j)}$ on the RGC #100 graph with $j = 1$ and $j = 2$, which is in parallel with the P_N case in the top figure of Figure 8.3b.



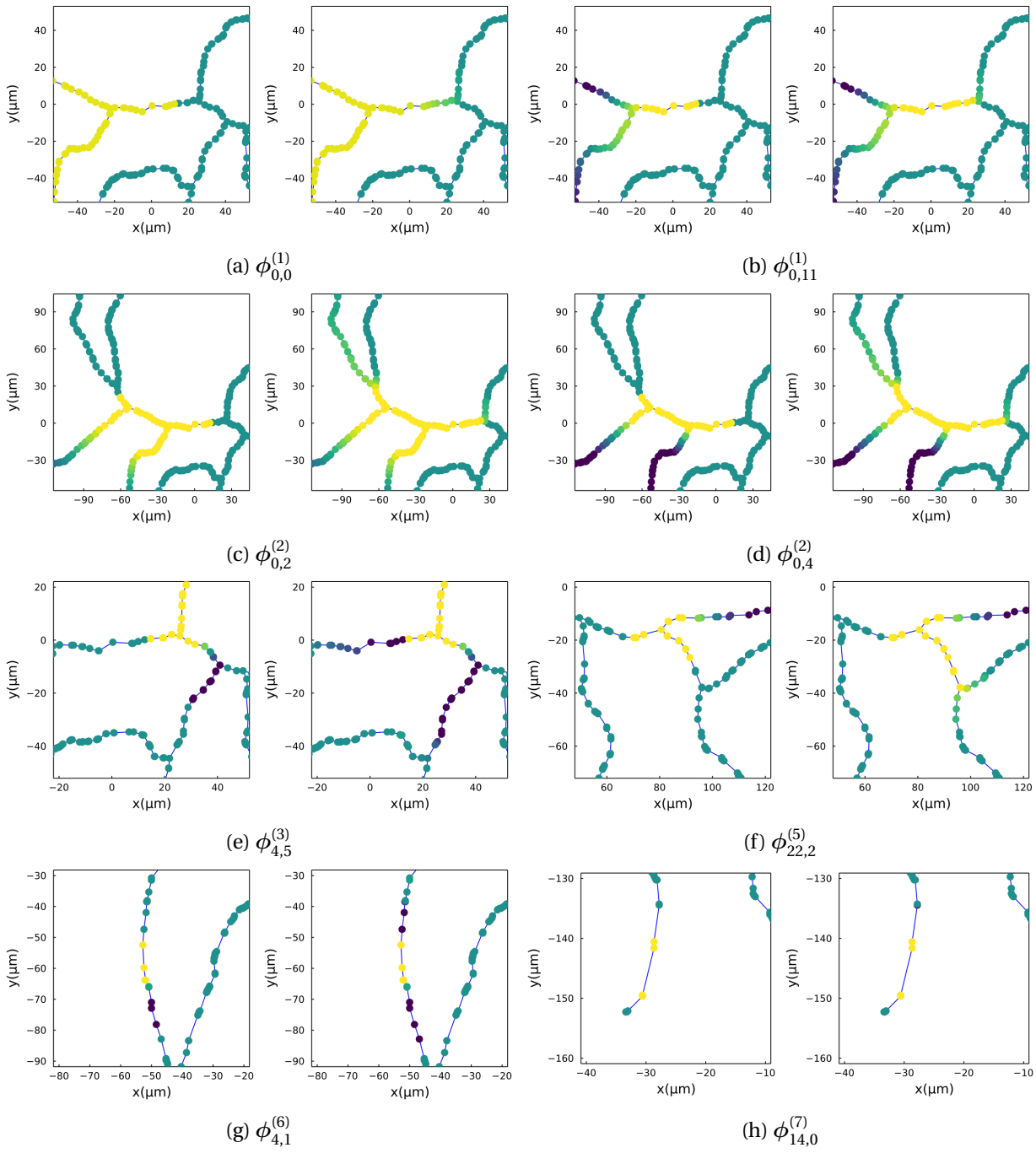
(a) diag$\left(U^{(1)}\right)$ (b) diag$\left(U^{(2)}\right)$

FIGURE 8.8. The diagonal entries of the orthogonal folding operator $U^{(j)}$ ($\epsilon = 0.3$) on the RGC #100 at levels $j = 1$ and $j = 2$.

8.3. Lapped HGLET

One of direct applications of such generalized operators on primal graphs is to get a *lapped* version of HGLET. As briefly mentioned in the beginning of this chapter, the HGLET [**34**] is a graph version of the Hierarchical Block DCT dictionary and is based on the hierarchical bipartition tree of a primal graph G. We use $\phi_{k,l}^{(j)} \in \mathbb{R}^N$ to denote one of its basis vectors, which is essentially the l-th[5] eigenvector of $L(G_k^{(j)})$ extended by zeros outside of its support $V_k^{(j)}$. Like in the case of the Hierarchical Block DCT, the hard bipartitions of the underlying domain G used in HGLET cause discontinuities of $\phi_{k,l}^{(j)}$ near the cutoff boundaries. As a remedy, we can utilize the orthogonal unfolding operators $\left(U^{(j)}\right)^{\mathsf{T}}$ to eliminate the discontinuities and get the smoothed HGLET basis vectors with spillovers across the cutoff boundaries, i.e., $\left\{\left(U^{(j)}\right)^{\mathsf{T}} \phi_{k,l}^{(j)}\right\}$. Note that both $\left(U^{(j)}\right)^{\mathsf{T}}$ and $P_{V_j^{(k)}}$ have the smoothing effect for the HGLET basis vectors near the cutoff boundaries. However, $P_{V_j^{(k)}}$ are not orthogonal matrices. To ensure the orthogonality of the basis vectors, we use $\left(U^{(j)}\right)^{\mathsf{T}}$. This is in parallel with the way Coifman and Meyer defined the local cosine functions in [**15**]. We denote such obtained dictionary as the *Lapped Hierarchical Graph Laplacian Eigen Transform* (LP-HGLET) dictionary.

[5] with respect to the magnitude of the eigenvalues in non-decreasing order



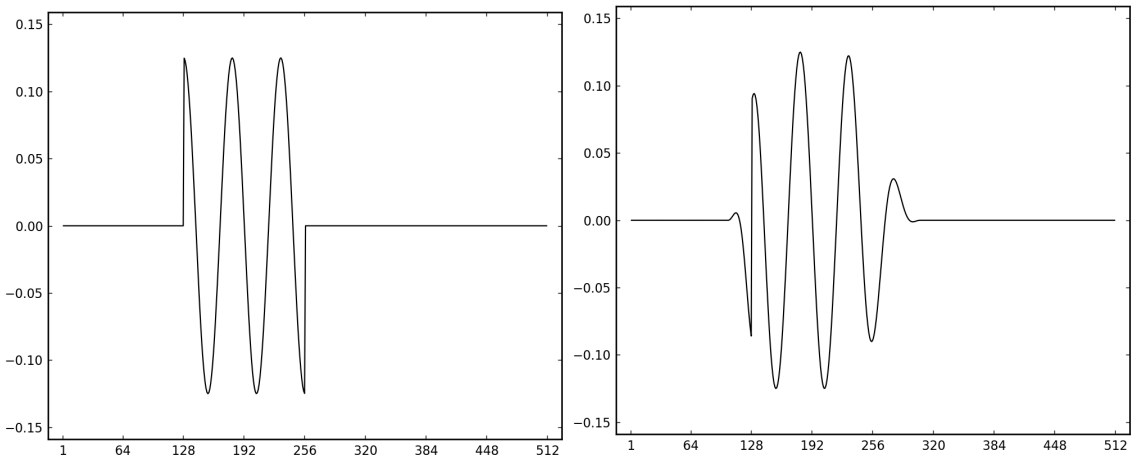
(a) $\phi^{(1)}_{0,0}$ (b) $\phi^{(1)}_{0,11}$

(c) $\phi^{(2)}_{0,2}$ (d) $\phi^{(2)}_{0,4}$

(e) $\phi^{(3)}_{4,5}$ (f) $\phi^{(5)}_{22,2}$

(g) $\phi^{(6)}_{4,1}$ (h) $\phi^{(7)}_{14,0}$

FIGURE 8.9. HGLET vs. LP-HGLET on the RGC #100 (zoomed in on the cutoff boundaries). In each subfigure, the left is an HGLET basis vector and the right is its lapped version by applying $\left(U^{(j)}\right)^{\mathsf{T}}$. The basis vector amplitudes within $(-0.05, 0.05)$ are mapped to the viridis colormap.



(a) $\phi_{1,5}^{(2)}$ (b) $\left(U^{(2)}\right)^{\mathsf{T}} \phi_{1,5}^{(2)}$

FIGURE 8.10. An HGLET basis vector vs. its lapped version on P_{512}.

Figure 8.9 shows the comparison between some of the HGLET basis vectors and their lapped version on the RGC #100, where $U^{(j)}$ is constructed with $\epsilon = 0.3$. As we can see, the basis vectors in the shallow levels (i.e., smaller j) are nicely smoothed over the cutoff boundaries (i.e., Figure 8.9a, b, c, d), while the smoothing effects of the basis vectors in the deeper levels are not so ideal (i.e., Figure 8.9e, f, g, h). This is due to the shrinkage of the action regions as we go deeper: 1) the action regions start intersecting with the ones used in the shallower levels, such that the intersected nodes are ignored when assembling the orthogonal folding operators as discussed in Section 8.1; and 2) the subgraphs have fewer nodes so that the action regions become smaller or even become empty sets.

Figure 8.10 compares an HGLET basis vector, i.e., $\phi_{1,5}^{(2)}$, with its lapped version $\left(U^{(2)}\right)^{\mathsf{T}} \phi_{1,5}^{(2)}$ on P_{512}, where $U^{(2)}$ is constructed with $\epsilon = 0.3$. Note that the right end of $V_1^{(2)} = 129:256$ is essentially lapped by the action region on level $j = 1$, and the left end of $V_1^{(2)}$ is lapped by one of the action regions on level $j = 2$, which has a smaller bandwidth. Therefore, we can see that the lapped basis vector is smoother on the right end of $V_1^{(2)}$ compared to the left end of $V_1^{(2)}$.

The LP-HGLET dictionary is associated with the parameter $\epsilon \in (0,1)$. When $\epsilon = 0$, it degenerates to the HGLET dictionary since the orthogonal unfolding operators reduce to I_N. Note that the HGLET basis vectors can also choose the extended eigenvectors of L_{rw} and L_{sym}. Yet, the eigenvectors of L_{rw} are not orthogonal to each other, so the Lapped HGLET also applies to HGLET with L_{sym} but not L_{rw}. Furthermore, the HGLET best basis algorithm [**31**, **34**] can be easily adapted to the LP-HGLET without

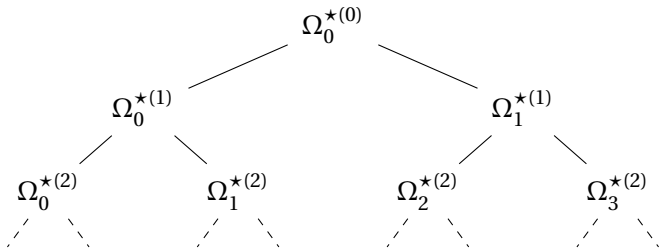


FIGURE 8.11. The hierarchical bipartition subspaces of the dual graph supporting space $\text{span}(V^\star) \equiv \Omega_0^{\star(0)} \equiv \mathbb{R}^N$, which is the soft/lapped version of Figure 6.1.

further modifications. This is because of the same hierarchical bipartition structures of Figure 8.1 and Figure 8.4.

8.4. Construction of the Lapped Natural Graph Wavelet Packet

We now focus on the application of the smooth orthogonal projectors on a dual graph $G^\star$. To be clear, we introduce another set of symbols associated with the smooth orthogonal projectors on dual graphs. Figure 8.11 shows the hierarchical soft/lapped bipartition subspaces $\Omega_k^{\star(j)}$ of $\text{span}(V^\star) = \mathbb{R}^N$, whose definition is explicitly described under Eq. (8.4). Note that any two subspaces $\Omega_{k_1}^{\star(j)}$ and $\Omega_{k_2}^{\star(j)}$ at the same level are orthogonal to each other. Moreover, if $\dim\left(\Omega_k^{\star(j)}\right) > 1$ and $\Omega_{k'}^{\star(j+1)}, \Omega_{k'+1}^{\star(j+1)}$ are the two children of $\Omega_k^{\star(j)}$,

$$\Omega_k^{\star(j)} = \Omega_{k'}^{\star(j+1)} \bigoplus \Omega_{k'+1}^{\star(j+1)}, \qquad j = 0 : J-1 \text{ and } k = 0 : K^j - 1.$$

Each $\Omega_k^{\star(j)}$ is associated with: 1) the supporting set of eigenvectors $V_k^{\star(j)}$; 2) the j-th level dual graph orthogonal folding operator $U^{(j)}$ with the uniform relative action region bandwidth ϵ; and 3) the corresponding smooth orthogonal projector $P_{V_k^{\star(j)}}$.

Recall the construction of the VM-NGWP dictionary can be summarized as below:

$$\Phi_k^{(j)} = \Phi \chi_{V_k^{\star(j)}} I_k^{(j)}, \qquad \Psi_k^{(j)} = \text{varimax}\left(\Phi_k^{(j)}\right), \tag{8.3}$$

where the dual graph restriction operator $\chi_{V_k^{\star(j)}} \in \mathbb{R}^{N\times N}$ is a diagonal matrix with $\left(\chi_{V_k^{\star(j)}}\right)_{l+1,l+1} = 1$ if $\phi_l \in V_k^{\star(j)}$, $= 0$ otherwise (for $l = 0 : N-1$), and $I_k^{(j)} \in \mathbb{R}^{N\times N_k^j}$ ($N_k^j := \left|V_k^{\star(j)}\right|$) is the *slicing operator* and it can be simply obtained by removing all the zero columns of $\chi_{V_k^{\star(j)}}$. The column space of $\Phi_k^{(j)}$ is the

subspace span$\left(V_k^{\star(j)}\right)$ and the columns of $\Phi_k^{(j)}$ form an ONB of span$\left(V_k^{\star(j)}\right)$. By performing the varimax rotation on $\Phi_k^{(j)}$, we obtain $\Psi_k^{(j)}$ whose columns are more "localized" in the primal domain G.

Likely, we can build a graph version of the Meyer wavelet packet dictionary as follows. We replace the dual graph restriction operator $\chi_{V_k^{\star(j)}}$ in Eq. (8.3) by the smooth orthogonal projector $P_{V_k^{\star(j)}}$ and get the matrix $Y_k^{(j)} \in \mathbb{R}^{N\times N_k^j}$.

$$Y_k^{(j)} := \Phi P_{V_k^{\star(j)}} I_k^{(j)}, \qquad P_{V_k^{\star(j)}} = \left(U^{(j)}\right)^{\mathsf{T}} \chi_{V_k^{\star(j)}} U^{(j)}. \tag{8.4}$$

The column space of $Y_k^{(j)}$ is $\Omega_k^{\star(j)}$, but the column vectors of $Y_k^{(j)}$ do not form an ONB of $\Omega_k^{\star(j)}$. Thus, we first perform the modified Gram-Schmidt (MGS) algorithm (i.e., Algorithm 3 without the pivoting step) to find an ONB of $\Omega_k^{\star(j)}$. Then, we apply the varimax rotation (i.e., Algorithm 2) on the obtained ONB to get the graph wavelet packet vectors, which are more "localized" in the primal domain G.

$$\Psi_k^{(j)} = \text{varimax}\left(\text{MGS}\left(Y_k^{(j)}\right)\right). \tag{8.5}$$

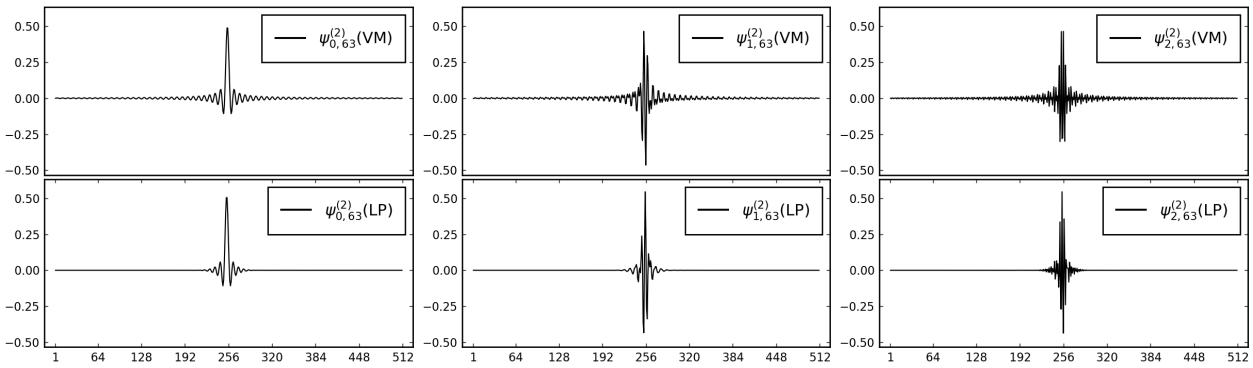
We refer to this graph wavelet packet dictionary $\left\{\Psi_k^{(j)}\right\}_{j=0:J;\, k=0:K^j-1}$ generated by the above procedures as the *Lapped Natural Graph Wavelet Packet* (LP-NGWP) dictionary. Since all the dual graph orthogonal folding operators used in the construction depend on a relative action region bandwidth ϵ, the LP-NGWP dictionary also depends on such a parameter $\epsilon \in (0,1)$. Furthermore, the LP-NGWP dictionary with $\epsilon = 0$ degenerates to the VM-NGWP dictionary, because $U^{(j)} \equiv I_N$ and $P_{V_k^{\star(j)}} \equiv \chi_{V_k^{\star(j)}}$ when $\epsilon = 0$.

Figure 8.12 shows the comparison between the VM-NGWP basis vectors and the LP-NGWP ($\epsilon = 0.3$) basis vectors on P_{512}. As one can see, the basis vectors of LP-NGWP are more compactly supported and localized in the primal domain than those of VM-NGWP. Moreover, we can define the following two quantities to measure the basis vector's main support width and degree of localization on P_{512}.

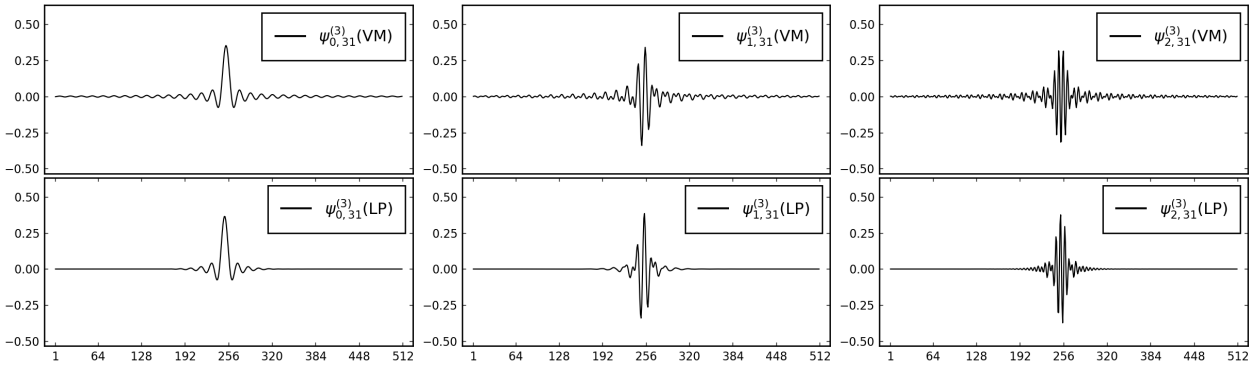
$$\text{main support width of } \psi_{k,l}^{(j)} := \min\left\{x_r - x_l + 1 \,\middle|\, 1 \le x_l \le x_r \le 512, \text{ s.t. } \max_{x=1:512, x\notin[x_l,x_r]} \left|\psi_{k,l}^{(j)}(x)\right| < 0.01\right\}$$

$$\text{sidelobe attenuation of } \psi_{k,l}^{(j)} := \frac{\text{the second largest local maximum value of } \psi_{k,l}^{(j)}}{\text{the maximum value of } \psi_{k,l}^{(j)}}$$

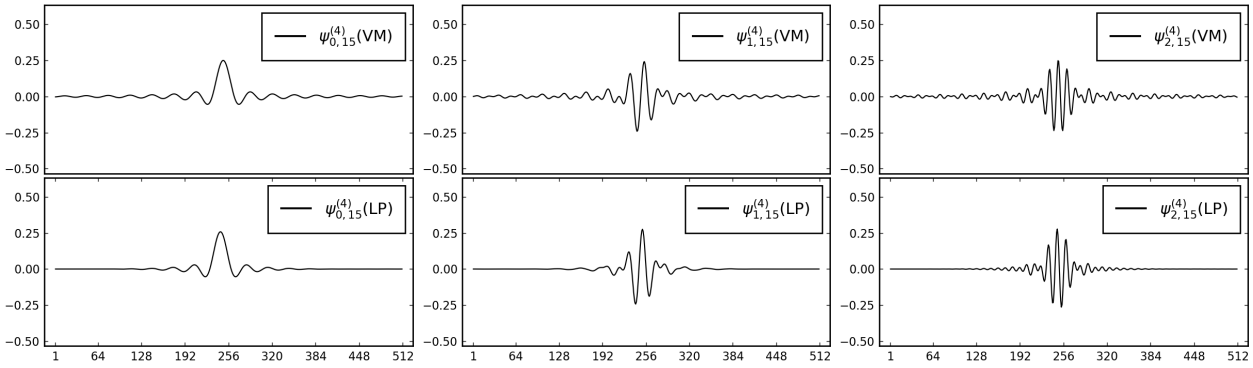
Table 8.2 shows these two quantities of the basis vectors in Figure 8.12, which also indicates the basis vectors of LP-NGWP have shorter supports and more localized structures.



(a) $\psi^{(2)}_{0,63}$ (b) $\psi^{(2)}_{1,63}$ (c) $\psi^{(2)}_{2,63}$



(d) $\psi^{(3)}_{0,31}$ (e) $\psi^{(3)}_{1,31}$ (f) $\psi^{(3)}_{2,31}$



(g) $\psi^{(4)}_{0,15}$ (h) $\psi^{(4)}_{1,15}$ (i) $\psi^{(4)}_{2,15}$

FIGURE 8.12. Comparison between the NGWP basis vectors on P_{512} (VM-NGWP vs. LP-NGWP ($\epsilon = 0.3$)).

basis vector	main support width	sidelobe attenuation
$\psi^{(2)}_{0,63}$ (VM)	118	0.1266
$\psi^{(2)}_{0,63}$ (LP)	**44**	**0.1156**
$\psi^{(2)}_{1,63}$ (VM)	218	0.6374
$\psi^{(2)}_{1,63}$ (LP)	**47**	**0.4377**
$\psi^{(2)}_{2,63}$ (VM)	248	0.9969
$\psi^{(2)}_{2,63}$ (LP)	**47**	**0.6540**
$\psi^{(3)}_{0,31}$ (VM)	170	0.1286
$\psi^{(3)}_{0,31}$ (LP)	**86**	**0.1114**
$\psi^{(3)}_{1,31}$ (VM)	306	0.6638
$\psi^{(3)}_{1,31}$ (LP)	**79**	**0.4396**
$\psi^{(3)}_{2,31}$ (VM)	302	0.9941
$\psi^{(3)}_{2,31}$ (LP)	**83**	**0.7850**
$\psi^{(4)}_{0,15}$ (VM)	245	0.1285
$\psi^{(4)}_{0,15}$ (LP)	**144**	**0.1127**
$\psi^{(4)}_{1,15}$ (VM)	411	0.6633
$\psi^{(4)}_{1,15}$ (LP)	**142**	**0.4330**
$\psi^{(4)}_{2,15}$ (VM)	414	0.7723
$\psi^{(4)}_{2,15}$ (LP)	**150**	**0.7371**

TABLE 8.2. Quantitative measurements of localization of the NGWP basis vectors in Figure 8.12.

8.5. Computational Complexity

At each $\Omega_k^{\star(j)}$ of the hierarchical bipartition tree of span$\left(V^\star\right)$ in Figure 8.11, there are two main computational burdens for our LP-NGWP dictionary construction: 1) the matrix multiplications when assembling the matrix $Y_k^{(j)}$ in Eq. (8.4) cost $O\left(2N^2\cdot N_k^j\right)$; and 2) performing the MGS algorithm followed by the varimax algorithm in Eq. (8.5) costs $O\left(3N\cdot(N_k^j)^2+c\cdot(N_k^j)^3\right)$ where $c=1000$; see Section 6.3 for the details of the cost of the varimax algorithm. Therefore, the total cost at each (j,k) is

$$O\left(2N^2\cdot N_k^j+3N\cdot(N_k^j)^2+c\cdot(N_k^j)^3\right).$$

And we need to sum up this cost on all the tree nodes. Let us analyze the special case of the perfectly balanced and fully developed bipartition tree with $N=2^{j_{\max}}$ as we did for VM-NGWP in Section 6.3 and PC-NGWP in Section 7.4. In this case, the bipartition tree has $1+j_{\max}$ levels, and $N_k^j=2^{j_{\max}-j}$, $k=0:2^j-1$.

Thus, at each level j, we have:

$$\sum_{k=0}^{2^j-1} 2N^2 \cdot N_k^j + 3N \cdot (N_k^j)^2 + c \cdot (N_k^j)^3 = 2N^3 + 3N^3 \cdot 2^{-j} + cN^3 \cdot 2^{-2j}.$$

Finally, by summing this from $j = 1$ to $j_{\max}$ (no computation is needed at the root),

$$\sum_{j=1}^{j_{\max}} N^3(2 + 3 \cdot 2^{-j} + c \cdot 2^{-2j}) = 2N^3 \cdot \log_2 N + (3 + \frac{c}{3}) \cdot N^3 - 3 \cdot N^2 - \frac{c}{3} \cdot N,$$

the total cost for LP-NGWP dictionary construction in this ideal case is $O\left(N^3 \log_2 N\right)$, which is slightly more expensive than the $O\left(N^3\right)$ cost of VM-NGWP and PC-NGWP.

8.6. L_{sym} Version of the *LP-NGWP*

As explained in Section 6.4 and Section 7.5, we can also construct the L_{sym} version of $G^\star(V^\star, E^\star, W^\star)$ via one of the *viable* metrics in Table 5.1, i.e., $d_{\text{ROT}}^{(1)}$, d_{sROT} (for trees), d_{HAD}, and d_{DAG}. Then, we can build the L_{sym} *version of the Lapped Natural Graph Wavelet Packet* (LP-NGWP-L_{sym}) dictionary by following the same procedures in Section 8.4. Note that the LP-NGWP-L_{sym} dictionary is also associated with $\epsilon \in (0, 1)$ and it reduces to the VM-NGWP-L_{sym} dictionary if $\epsilon = 0$.

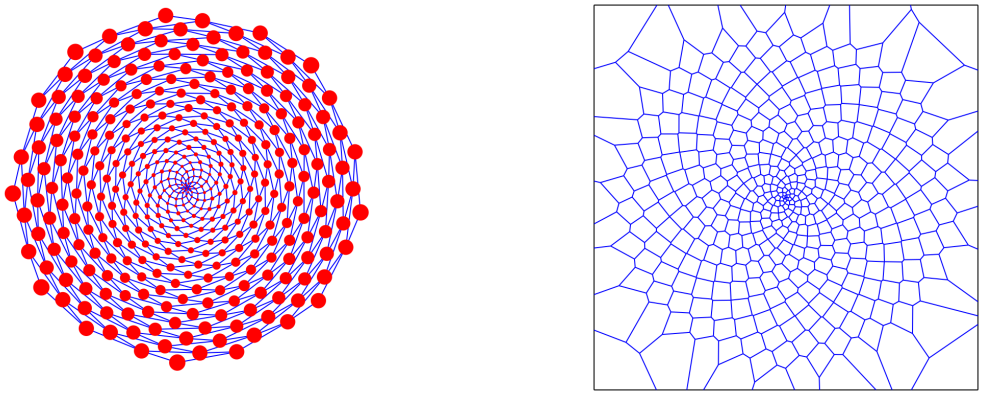
CHAPTER 9

Graph Signal Approximation via Natural Graph Wavelet Packets

In this chapter, we demonstrate the usefulness of our proposed NGWP dictionaries in efficient approximation of graph signals on two graphs, and compare the performance with the other previously proposed methods: the global eigenbasis of $L(G)$; the global eigenbasis of $L_{\mathrm{sym}}(G)$; the HGLET best basis [**34**]; the LP-HGLET best basis (Section 8.3); and the eGHWT best basis [**77**]. Note that the LP-HGLET, the LP-NGWP and the LP-NGWP-L_{sym} are associated with $\epsilon = 0.3$ consistently in this chapter's experiments. Note also that, although the HGLET best basis can choose three different types of graph Laplacian eigenvectors of L, L_{rw}, and L_{sym} at each subgraph (see Eq. (2.1)), we only use L and L_{sym} at each subgraph in order to compare its performance in a fair manner with the NGWP dictionaries that are based on the eigenvectors of $L(G)$ and $L_{\mathrm{sym}}(G)$ respectively. And likewise, we use the lapped eigenvectors of L and L_{sym} at each subgraph for the LP-HGLET best basis. Next, we use the ℓ^1-norm minimization as the best-basis selection criterion for all the best bases in our experiments. The edge weights of the dual graph $G^\star$ are the reciprocals of the DAG pseudometric (i.e., d_{DAG}) between the corresponding eigenvectors of $L(G)$ as defined in Eq. (4.11). For a given graph G and a graph signal $\boldsymbol{f}$ defined on it, we decompose $\boldsymbol{f}$ into those dictionaries and select those bases first. Then, to measure the approximation performance, we sort the expansion coefficients in non-increasing order of their magnitude, and use the top k most significant terms to approximate $\boldsymbol{f}$ where k starts from 0 up to about 50% of the total number of terms, or more precisely, $\lfloor 0.5N \rfloor + 1$. All of the approximation performance is measured by the relative ℓ^2 approximation error with respect to the fraction of coefficients retained, which we denote FCR for simplicity.

9.1. Sunflower Graph Signals Sampled on Images

We consider the so-called "sunflower" graph shown in Figure 9.1a. This particular graph has 400 nodes and each edge weight is set as the reciprocal of the Euclidean distance between the endpoints of that edge. Consistently counting the number of spirals in such a sunflower graph gives rise to the *Fibonacci numbers*: $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \ldots$; see Figure 9.1a. We also note that the majority of nodes



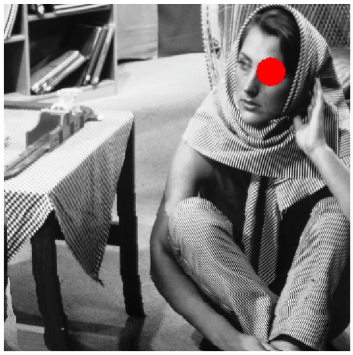
(a) Sunflower graph (b) Voronoi tessellation

FIGURE 9.1. (a) Sunflower graph ($N = 400$); node radii vary for visualization purpose; (b) its Voronoi tessellation.

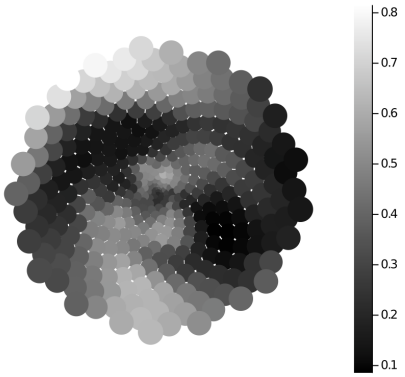
(374 among 400) have degree[1] 4 while there are eight nodes with degree 2, 17 nodes with degree 3, and the central node has the greatest degree 9. See, e.g., [**55**, **86**] and our code `SunFlowerGraph.jl` in [**50**], for algorithms to construct such sunflower grids and graphs. We can also view such a distribution of nodes as a simple model of the distribution of *photoreceptors in mammalian visual systems* due to cell generation and growth; see, e.g., [**67**, Chap. 9]. Such a viewpoint motivates us the following sampling scheme: 1) overlay the sunflower graph on several parts of the standard Barbara image; 2) construct the Voronoi tessellation of the bounding square region with the nodes of the sunflower graph as its seeds as shown in Figure 9.1b; 3) compute the average pixel value within each Voronoi cell; and 4) assign that average pixel value to the corresponding seed/node[2]. See [**90**] for more about the relationship between the Voronoi tessellation and the sunflower graph. We also note that for generating the Voronoi tessellation, we used the following open source Julia packages developed by the JuliaGeometry team [**84**]: `VoronoiDelaunay.jl`; `VoronoiCells.jl`; and `GeometricalPredicates.jl`. For our numerical experiments, we sampled two different regions: her left eye and pants, where quite different image features are represented, i.e., a piecewise-smooth image containing oriented edges and a textured image with directional oscillatory patterns, respectively.

[1]Here we are talking about the unweighted node degree, i.e., the diagonal entry of $\tilde{D}(G)$.

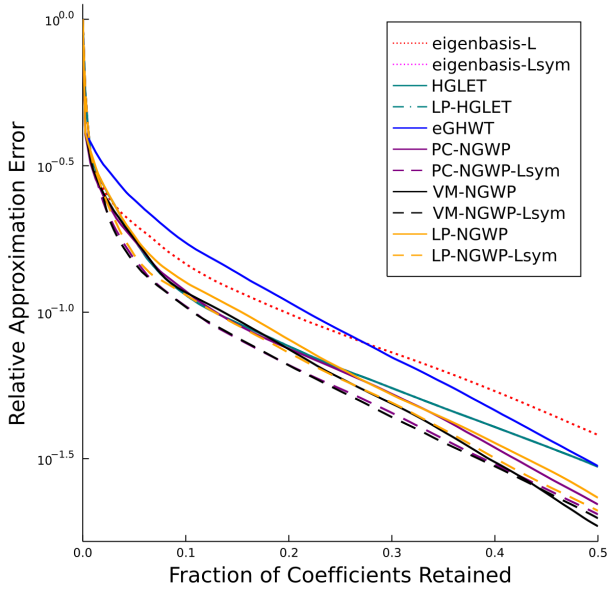
[2]If a Voronoi cell does not contain any original image pixels (which occurs at some tiny cells around the center), we bilinearly interpolate the pixel value at the node location using the nearest image pixel values.



(a) The sunflower graph overlaid on Barbara's left eye



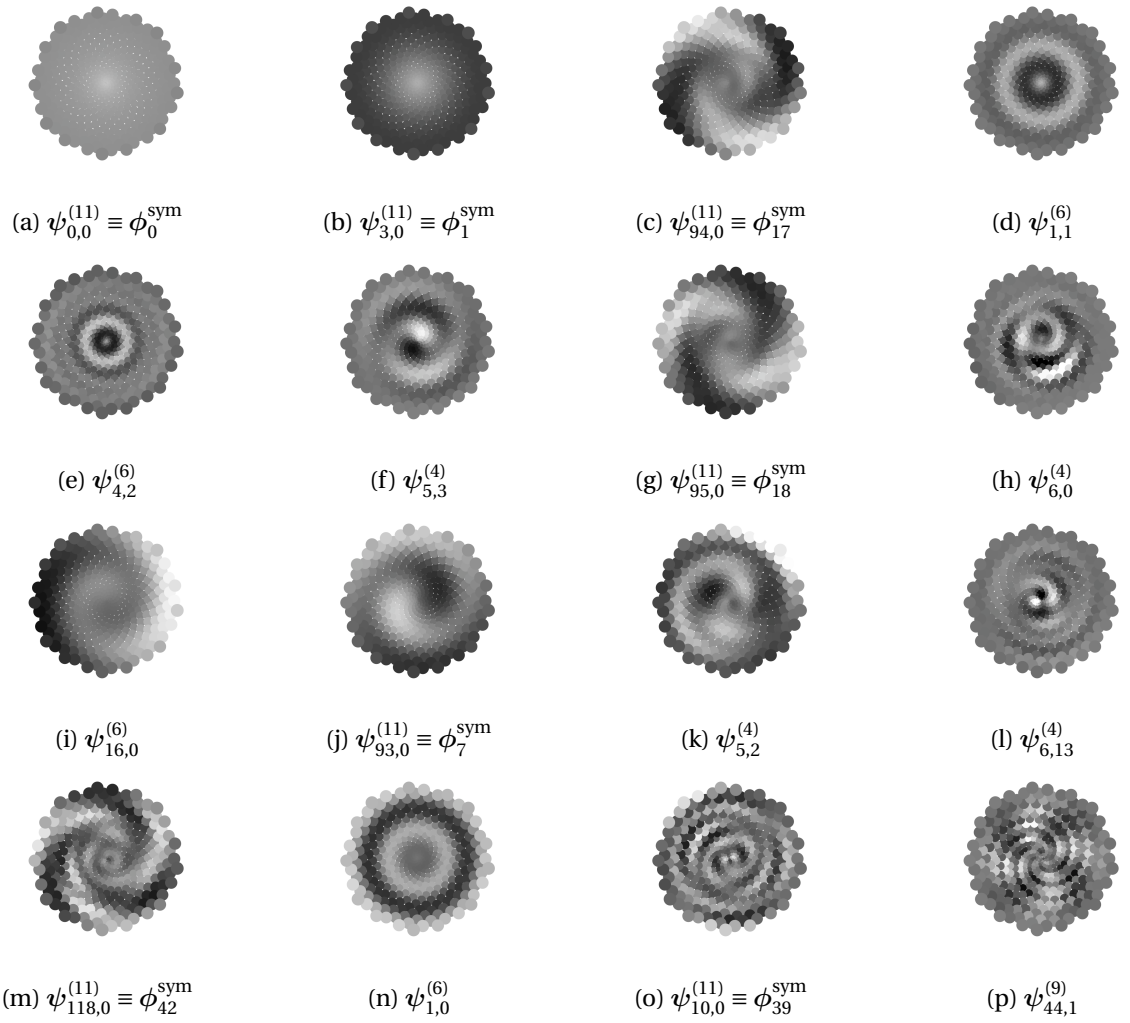
(b) Barbara's left eye as an input graph signal



(c) Approximation performance of various methods

FIGURE 9.2. Barbara's left eye region sampled on the sunflower graph nodes (a) as a graph signal (b); the relative ℓ^2 approximation errors by various methods (c).

First, let us discuss our approximation experiments on Barbara's eye graph signal, which are shown in Figure 9.2. From Figure 9.2c, we observe the following: 1) the L_{sym} version of NGWP best bases performed best closely followed by the regular L version of NGWP best bases; 2) the HGLET best basis and the LP-HGLET best basis were the next best performers, both of which chose the global eigenbasis of $L_{\text{sym}}(G)$, and they worked quite well particularly up to $FCR \approx 0.25$; 3) the global eigenbasis of $L(G)$ worked relatively well up to $FCR \approx 0.27$; and 4) the eGHWT best basis only performed fair enough in the range $FCR \gtrapprox 0.27$. These observations can be attributed to the fact that this Barbara's eye graph signal is



(a) $\psi_{0,0}^{(11)} \equiv \phi_0^{\text{sym}}$ (b) $\psi_{3,0}^{(11)} \equiv \phi_1^{\text{sym}}$ (c) $\psi_{94,0}^{(11)} \equiv \phi_{17}^{\text{sym}}$ (d) $\psi_{1,1}^{(6)}$

(e) $\psi_{4,2}^{(6)}$ (f) $\psi_{5,3}^{(4)}$ (g) $\psi_{95,0}^{(11)} \equiv \phi_{18}^{\text{sym}}$ (h) $\psi_{6,0}^{(4)}$

(i) $\psi_{16,0}^{(6)}$ (j) $\psi_{93,0}^{(11)} \equiv \phi_7^{\text{sym}}$ (k) $\psi_{5,2}^{(4)}$ (l) $\psi_{6,13}^{(4)}$

(m) $\psi_{118,0}^{(11)} \equiv \phi_{42}^{\text{sym}}$ (n) $\psi_{1,0}^{(6)}$ (o) $\psi_{10,0}^{(11)} \equiv \phi_{39}^{\text{sym}}$ (p) $\psi_{44,1}^{(9)}$

FIGURE 9.3. Sixteen most significant VM-NGWP-L_{sym} best basis vectors for Barbara's eye. The basis vector amplitudes within $(-0.15, 0.15)$ are mapped to the grayscale colormap.

not of piecewise-*constant* nature; rather, it is a *locally smooth* graph signal. Hence, the NGWP dictionaries containing smooth *and* more localized basis vectors made a difference in performance compared to the global graph Laplacian eigenbases, the HGLET best basis, the LP-HGLET best basis and the eGHWT best basis.

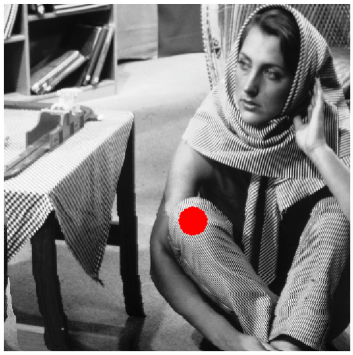
In order to examine what kind of basis vectors were chosen as the best basis to approximate this Barbara's eye signal, we display the 16 most significant VM-NGWP-L_{sym} best basis vectors in Figure 9.3. The other versions of the NGWP best bases vectors are relatively similar; hence they are not shown here.

We note that many of these top basis vectors essentially work as oriented edge detectors for Barbara's eye. For example, $\boldsymbol{\psi}_{1,1}^{(6)}$ (Figure 9.3d), $\boldsymbol{\psi}_{4,2}^{(6)}$ (Figure 9.3e) and $\boldsymbol{\psi}_{1,0}^{(6)}$ (Figure 9.3n) try to capture her eyelid while $\boldsymbol{\psi}_{5,3}^{(4)}$ (Figure 9.3f), $\boldsymbol{\psi}_{6,0}^{(4)}$ (Figure 9.3h), and $\boldsymbol{\psi}_{6,13}^{(4)}$ (Figure 9.3l) do the same for her iris and sclera. The other basis vectors take care of shading and peripheral features of her eye region. We also note that seven among these top 16 best basis vectors are the global eigenvectors of $L_{\mathrm{sym}}(G)$; see Figure 9.3a, b, c, g, j, m, o.

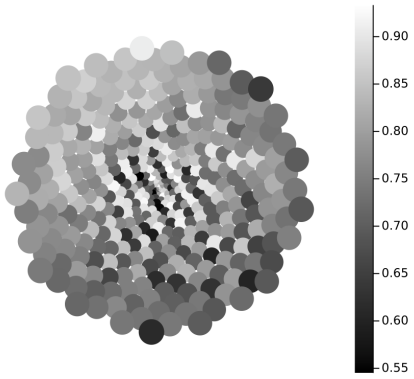
Now, let us discuss our second approximation experiments: Barbara's pants region as an input graph signal as shown in Figure 9.4. The nature of this graph signal is completely different from the eye region: it is dominated by directional oscillatory patterns of her pants. From Figure 9.4c, we observe the following: 1) the regular L version of NGWP best bases and the eGHWT best basis performed very well and competitively; the L version of NGWP best bases performed better than the eGHWT best basis up to $FCR \approx 0.2$ while the latter outperformed all the others for $FCR \gtrapprox 0.2$; 2) there is a gap in performance between those four bases and the rest: the L_{sym} version of NGWP best bases; the HGLET best basis; the LP-HGLET best basis; and the global Laplacian eigenbases. Note that both the HGLET and the LP-HGLET choose the global eigenbasis of $L(G)$. The poor performance of the L_{sym} version of NGWP best bases is mainly because the important $\boldsymbol{\phi}_0^{\mathrm{sym}}$ (see Figure 9.3a) contained in the basis dictionaries is nonconstant, and it does not work well comparing to $\boldsymbol{\phi}_0$ in capturing the average of this oscillatory signal. We knew that the eGHWT is known to be quite efficient in capturing oscillating patterns as shown by Shao and Saito for the graph setting [**77**] and by Lindberg and Villemoes for the classical non-graph setting [**52**]. Hence, it is a good thing to observe that our L version of the NGWP dictionaries are competitive with the eGHWT for this type of textured signal.

Figure 9.5 shows the 16 most significant VM-NGWP best basis vectors for approximating Barbara's pants signal. We note that the majority of these basis vectors are of high-frequency nature than those for the eye signal shown in Figure 9.3, which reflect the oscillating anisotropic patterns of her pants. The basis vectors $\boldsymbol{\psi}_{3,1}^{(6)}$ (Figure 9.5a), $\boldsymbol{\psi}_{9,0}^{(11)}$ (Figure 9.5j), and $\boldsymbol{\psi}_{1,0}^{(11)}$ (Figure 9.5p) take care of shading and peripheral features[3] in this region while the other basis vectors extract oscillatory patterns of various scales. We also note that four among these top 16 best basis vectors are the global graph Laplacian eigenvectors; see Figure 9.5h, j, k, p.

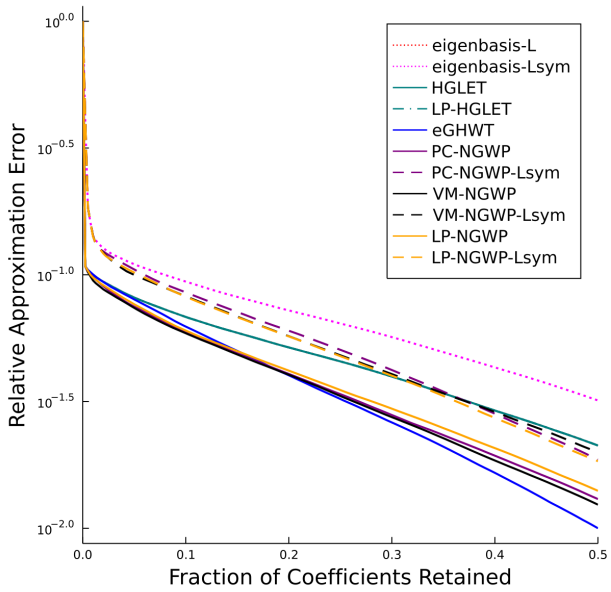
[3]Since the Voronoi cells used in the signal sampling are larger in the peripheral region, more pixel values got averaged result in a different feature compared to the center.



(a) The sunflower graph overlaid on Barbara's pants



(b) Barbara's pants region as an input graph signal



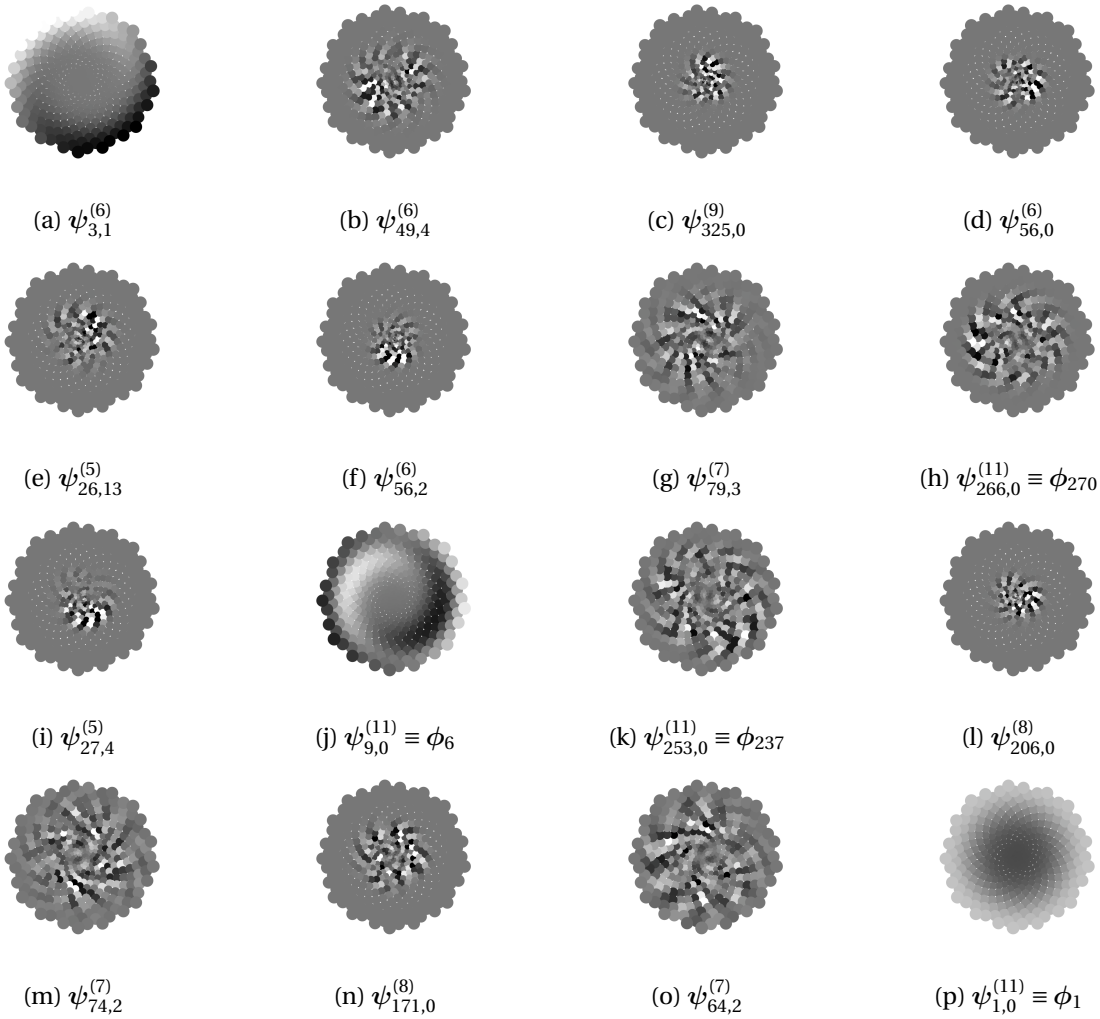
(c) Approximation performance of various methods

FIGURE 9.4. Barbara's pants region sampled on the sunflower graph nodes (a) as a graph signal (b); the relative ℓ^2 approximation errors by various methods (c).

9.2. Toronto Street Network

We obtained the street network data of the City of Toronto from its open data portal[4]. Using the street names and intersection coordinates included in the dataset, we construct the graph representing the street network there with $N = 2275$ nodes and $M = 3381$ edges. Figure 9.6a displays this graph. As

[4]URL: `https://open.toronto.ca/dataset/traffic-signal-vehicle-and-pedestrian-volumes`



(a) $\psi^{(6)}_{3,1}$ (b) $\psi^{(6)}_{49,4}$ (c) $\psi^{(9)}_{325,0}$ (d) $\psi^{(6)}_{56,0}$

(e) $\psi^{(5)}_{26,13}$ (f) $\psi^{(6)}_{56,2}$ (g) $\psi^{(7)}_{79,3}$ (h) $\psi^{(11)}_{266,0} \equiv \phi_{270}$

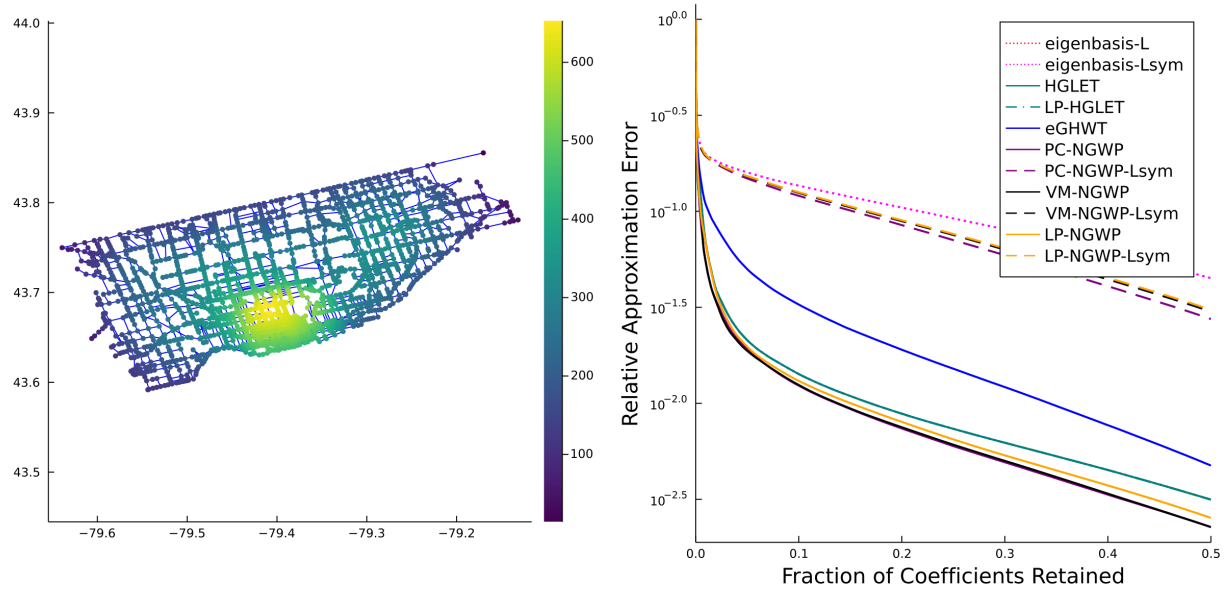
(i) $\psi^{(5)}_{27,4}$ (j) $\psi^{(11)}_{9,0} \equiv \phi_{6}$ (k) $\psi^{(11)}_{253,0} \equiv \phi_{237}$ (l) $\psi^{(8)}_{206,0}$

(m) $\psi^{(7)}_{74,2}$ (n) $\psi^{(8)}_{171,0}$ (o) $\psi^{(7)}_{64,2}$ (p) $\psi^{(11)}_{1,0} \equiv \phi_{1}$

FIGURE 9.5. Sixteen most significant VM-NGWP best basis vectors (the DC vector not shown) for Barbara's pants. The basis vector amplitudes within $(-0.15, 0.15)$ are mapped to the grayscale colormap.

before, each edge weight was set as the reciprocal of the Euclidean distance between the endpoints of that edge.

We analyze two graph signals on this street network: 1) spatial distribution of the street intersections and 2) pedestrian volume measured at each intersection. The first graph signal was constructed by counting the number of the nodes within the disk of radius 4.7 km centered at each node. In other words, this is a smooth version of histogram of the distribution of street intersections computed with the overlapping circular bins of equal size. The longest edge length measured in the Euclidean distance among



(a) A smooth spatial distribution of the street intersections

(b) Approximation performance of various methods

FIGURE 9.6. A graph signal representing the smooth spatial distribution of the street intersections on the Toronto street network (a). The horizontal and vertical axes of this plot represent the longitude and latitude geo-coordinates of this area, respectively. The results of our approximation experiments (b).

all these 3381 edges was chosen as this radius of this disk, which is located at the northeast corner of this graph as one can easily see in Figure 9.6a. The second graph signal is the most recent 8 peak-hour pedestrian volume counts collected at intersections (i.e., nodes in this graph) where there are traffic signals. The dataset was collected between the hours of 7:30 am and 6:00 pm, over the period of 03/22/2004–02/28/2018.

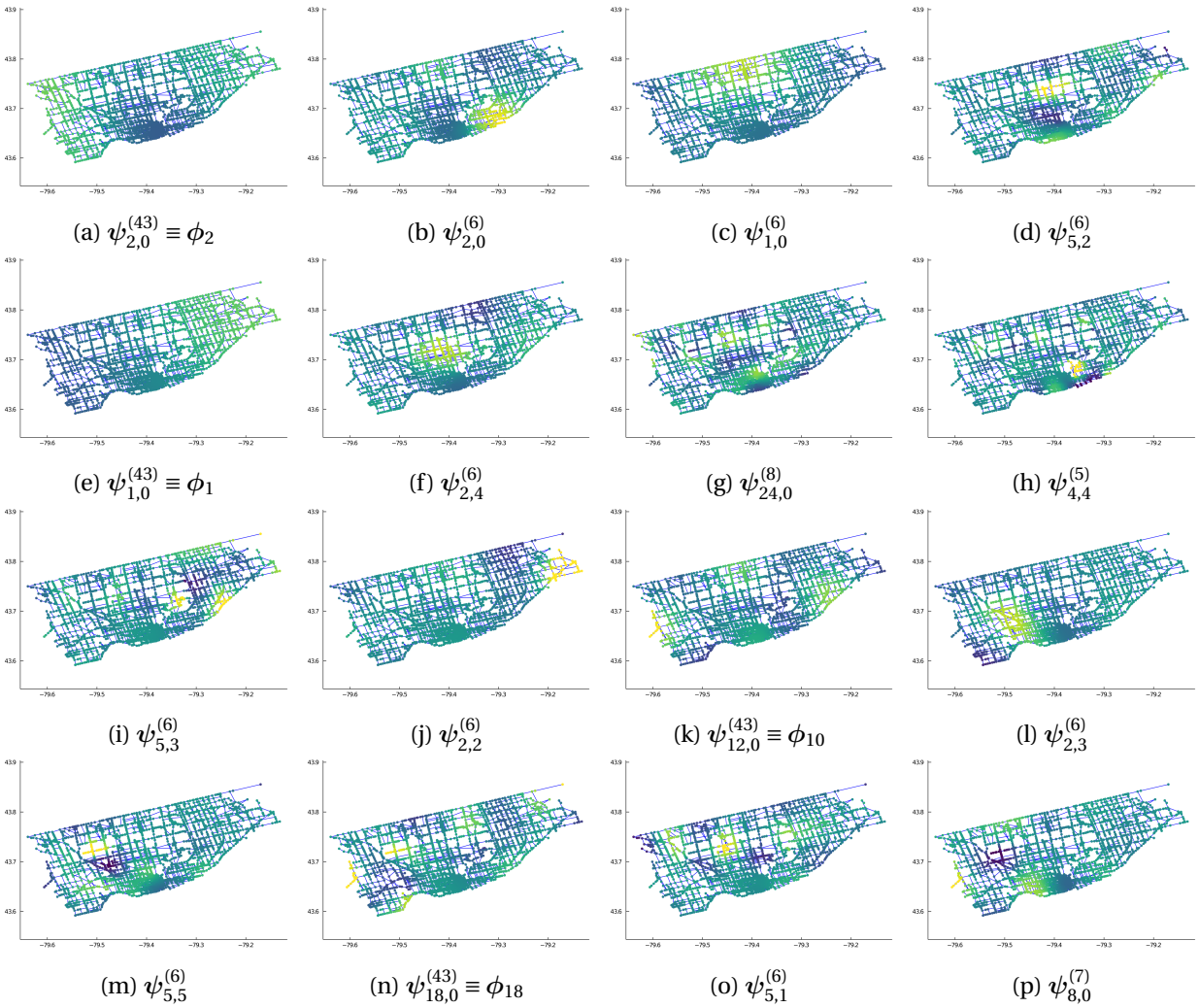
From Figure 9.6b, we observe that qualitative behaviors of these error curves (except the L_{sym} version bases) are relatively similar to those of Barbara's eye signal shown in Figure 9.2c. More precisely, 1) the L version of NGWP best bases outperformed all the others, and the difference between the VM-NGWP and the PC-NGWP is negligible, and the LP-NGWP is followed closely; 2) the HGLET best basis and the LP-HGLET best basis chose the global eigenbasis of $L(G)$, which worked quite well following the L version of NGWP best bases; 3) the eGHWT did not perform well; and 4) the L_{sym} version of NGWP best bases and the global eigenbasis of $L_{\mathrm{sym}}(G)$ performed very badly mainly due to the following reasons. Unlike in the sunflower graph, the nodes in the Toronto street network are sampled in a quite non-uniform manner

such that the variance of the node degrees[5] is quite large. Specifically, the average node degree of the Toronto street network is 924.06 with the standard deviation 579.44, while the average node degree of the sunflower graph is 47.69 with the standard deviation 68.50. Since $L_{\text{sym}}(G)$ is obtained by normalizing $L(G)$ with those node degrees in Eq. (2.1), its eigenvectors lose the global smoothness. In particular, they do not have a constant-like basis vector in their dictionaries to approximate this smooth graph signal.

In order to examine what kind of basis vectors were chosen to approximate this smooth histogram of street intersections, we display the most important 16 VM-NGWP best basis vectors in Figure 9.7. We note that these top basis vectors exhibit different spatial scales. The basis vectors with levels $j = 5$ and $j = 6$ are relatively localized to specific regions of Toronto. For example, $\psi_{4,4}^{(5)}$ (Figure 9.7h) tries to differentiate the eastern neighbor of the dense downtown region along the north-south direction while $\psi_{2,0}^{(6)}$ (Figure 9.7b) tries to do the same along the east-west direction. $\psi_{2,2}^{(6)}$ (Figure 9.7j) tries to differentiate the intersection density around the northeast region of Toronto. On the other hand, there are coarse scale basis vectors with $j = j_{\max} = 43$, which are in fact the global eigenvectors of $L(G)$, i.e., Figure 9.7a, e, k, n. It is not surprising that these coarse scale basis vectors were selected as a part of the VM-NGWP best basis considering that the global eigenbasis of $L(G)$ performed quite well on this graph signal as shown in Figure 9.6b.

Now, let us analyze the pedestrian volume data measured at the street intersections as shown in Figure 9.8a, which is highly localized around the specific part of the downtown region (the dense region in the lower middle section) of the street graph. Figure 9.8b shows the approximation errors of various methods. From Figure 9.8b, we observe the following: 1) the eGHWT best basis clearly outperformed all the other methods; 2) the HGLET best basis followed the eGHWT best basis; 3) the LP-HGLET best basis followed the HGLET best basis; 4) the NGWP best bases were the next best performers in the following order: the VM-NGWP, the VM-NGWP-L_{sym}, the LP-NGWP, the LP-NGWP-L_{sym}, the PC-NGWP-L_{sym}, and the PC-NGWP; and 5) the global Laplacian eigenbases were the worst performers. Considering the non-smooth and highly localized nature of the input signal, it is not surprising that the global bases did not perform well and that the non-smooth local bases (the eGHWT) and the basis vectors whose supports

[5] Here we refer to the weighted node degrees, i.e., the diagonal entries of the degree matrix $D(G)$.



(a) $\psi_{2,0}^{(43)} \equiv \phi_2$ (b) $\psi_{2,0}^{(6)}$ (c) $\psi_{1,0}^{(6)}$ (d) $\psi_{5,2}^{(6)}$

(e) $\psi_{1,0}^{(43)} \equiv \phi_1$ (f) $\psi_{2,4}^{(6)}$ (g) $\psi_{24,0}^{(8)}$ (h) $\psi_{4,4}^{(5)}$

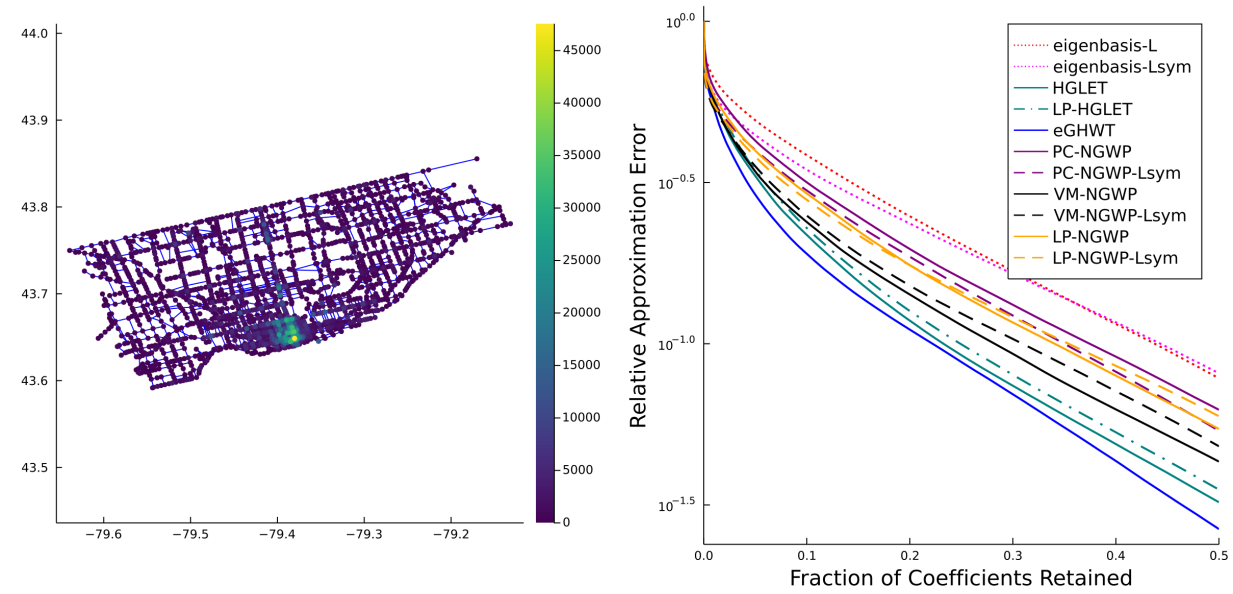
(i) $\psi_{5,3}^{(6)}$ (j) $\psi_{2,2}^{(6)}$ (k) $\psi_{12,0}^{(43)} \equiv \phi_{10}$ (l) $\psi_{2,3}^{(6)}$

(m) $\psi_{5,5}^{(6)}$ (n) $\psi_{18,0}^{(43)} \equiv \phi_{18}$ (o) $\psi_{5,1}^{(6)}$ (p) $\psi_{8,0}^{(7)}$

FIGURE 9.7. Sixteen most significant VM-NGWP best basis vectors (the DC vector not shown) for street intersection density data on the Toronto street map. The basis vector amplitudes within $(-0.075, 0.075)$ are mapped to the viridis colormap.

strictly or softly follow the partition pattern of the primal graph (i.e., the HGLET best basis or the LP-HGLET best basis) had an edge over the NGWP best bases that contain smooth basis vectors whose supports are not controlled in the primal graph.

In order to examine the performance difference among the regular L version of the NGWP best bases, we display their 16 most significant basis vectors in Figure 9.9 (VM), Figure 9.10 (PC), and Figure 9.11 (LP), respectively. The relationship among the L_{sym} version of NGWP best bases is relatively similar,

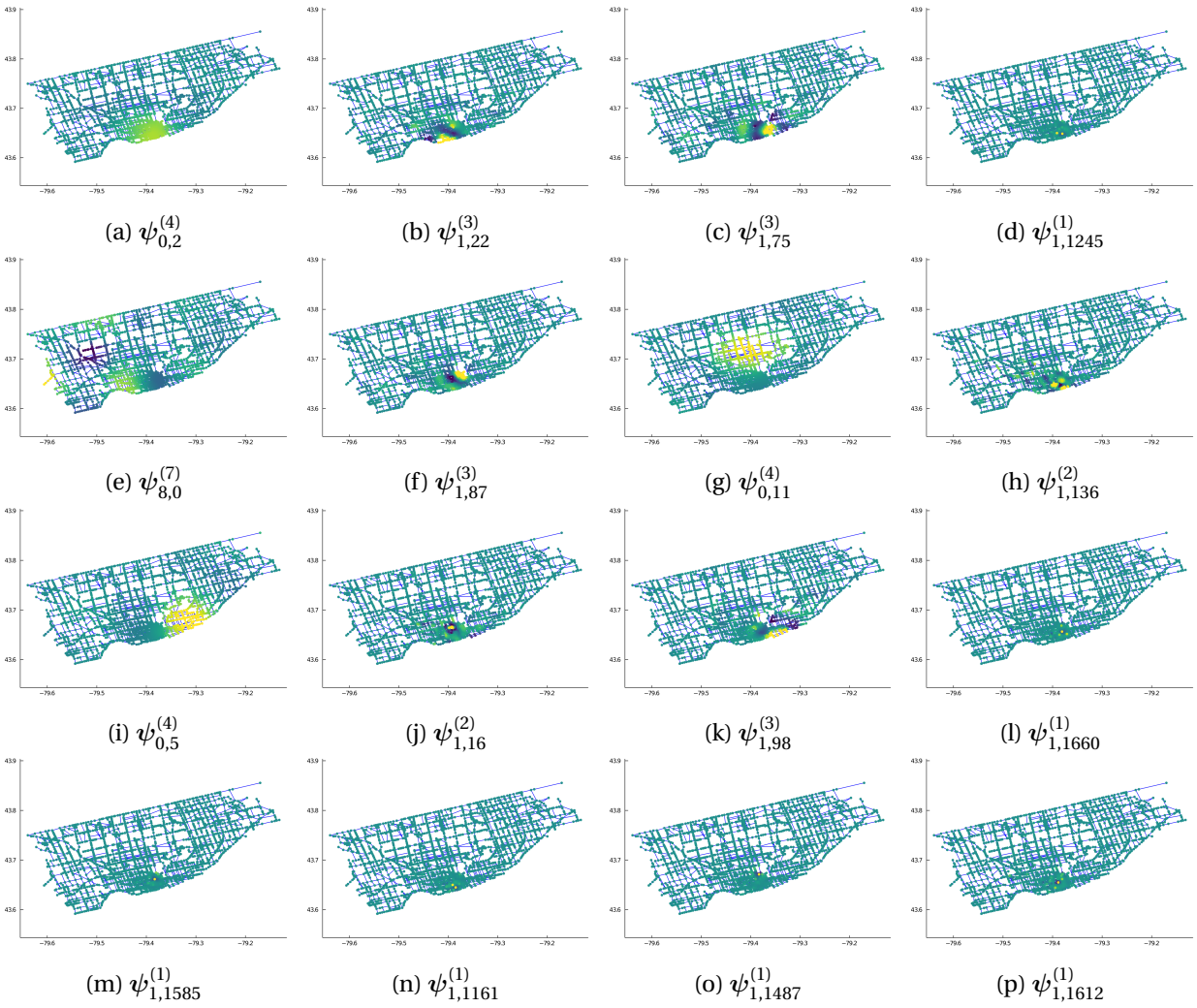


(a) Pedestrian volume data measured at the street intersections

(b) Approximation performance of various methods

FIGURE 9.8. The pedestrian volume graph signal on the Toronto street network (a); the results of our approximation experiments (b).

thus their top basis vectors are not shown here. Firstly, we note that the top VM-NGWP best basis vectors exhibit local to intermediate spatial scales. The basis vectors with $j = 1$ (Figure 9.9d, l, m, n, o, p) are highly localized at certain nodes within the dense downtown region while the basis vectors with $j = 2,3,4$ (Figure 9.9b, c, f, g, h, i, j, k) try to characterize the pedestrian volume within the downtown region and its neighbors as oriented edge detectors. The top basis vector $\psi_{0,2}^{(4)}$ in Figure 9.9a works as a local averaging operator around the downtown region. Secondly, the top PC-NGWP best basis vectors are more localized than those of the VM-NGWP best basis vectors. As one can see from Figure 9.10, there are neither medium nor coarse scale basis vectors in these top 16 basis vectors. The reason behind these performance difference between the VM-NGWP and the PC-NGWP is the following. The VM-NGWP best basis for this graph signal turned out to be "almost" the graph Shannon wavelet basis with the deepest level $J = 4$, i.e., the basis for the union of the following subspaces: $V_0^{\star(4)}$, $V_8^{\star(7)}$, $V_{18}^{\star(43)}(= \{\phi_{18}\})$, $V_{19}^{\star(43)}(= \{\phi_{19}\})$, $V_5^{\star(6)}$, $V_3^{\star(5)}$, $V_1^{\star(3)}$, $V_1^{\star(2)}$, and $V_1^{\star(1)}$. Note that $V_8^{\star(7)} \cup V_{18}^{\star(43)} \cup V_{19}^{\star(43)} \cup V_5^{\star(6)} \cup V_3^{\star(5)} = V_1^{\star(4)}$, hence this is "almost" the graph Shannon wavelet basis with $J = 4$, which is the basis for the union $V_0^{\star(4)} \cup V_1^{\star(4)} \cup V_1^{\star(3)} \cup V_1^{\star(2)} \cup V_1^{\star(1)}$. Hence, this VM-NGWP best basis should behave similarly to that



(a) $\psi^{(4)}_{0,2}$ (b) $\psi^{(3)}_{1,22}$ (c) $\psi^{(3)}_{1,75}$ (d) $\psi^{(1)}_{1,1245}$

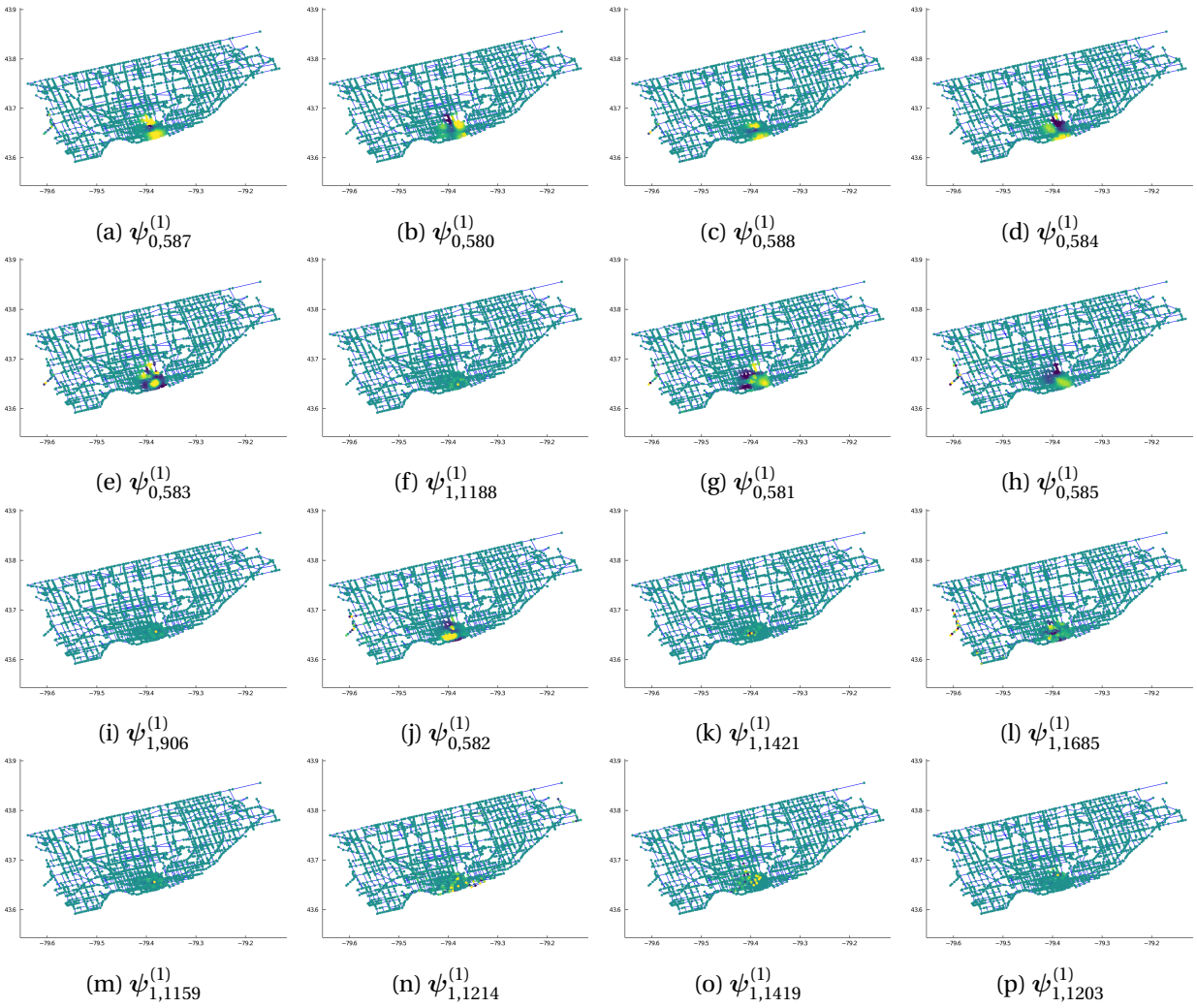
(e) $\psi^{(7)}_{8,0}$ (f) $\psi^{(3)}_{1,87}$ (g) $\psi^{(4)}_{0,11}$ (h) $\psi^{(2)}_{1,136}$

(i) $\psi^{(4)}_{0,5}$ (j) $\psi^{(2)}_{1,16}$ (k) $\psi^{(3)}_{1,98}$ (l) $\psi^{(1)}_{1,1660}$

(m) $\psi^{(1)}_{1,1585}$ (n) $\psi^{(1)}_{1,1161}$ (o) $\psi^{(1)}_{1,1487}$ (p) $\psi^{(1)}_{1,1612}$

FIGURE 9.9. Sixteen most significant VM-NGWP best basis vectors for pedestrian volume data on the Toronto street map. The basis vector amplitudes within $(-0.075, 0.075)$ are mapped to the viridis colormap.

graph Shannon wavelet basis (except the mother wavelet vectors at level $j = 4$). In particular, it does not contain oscillatory basis vectors of large scale that are not really necessary to approximate this highly localized pedestrian volume data. On the other hand, the PC-NGWP best basis turned out to be the graph Shannon wavelet basis with $J = 1$, i.e., the basis for the subspaces $V_0^{\star(1)}$ and $V_1^{\star(1)}$. Since the pedestrian volume data is quite non-smooth and localized, δ-like basis vectors with scale $j = 1$ in the PC-NGWP dictionary tend to generate sparser coefficients, i.e., having a small number of large magnitude coefficients with many negligible ones. Therefore, the best basis algorithm with ℓ^1-norm ends up favoring



(a) $\psi^{(1)}_{0,587}$ (b) $\psi^{(1)}_{0,580}$ (c) $\psi^{(1)}_{0,588}$ (d) $\psi^{(1)}_{0,584}$

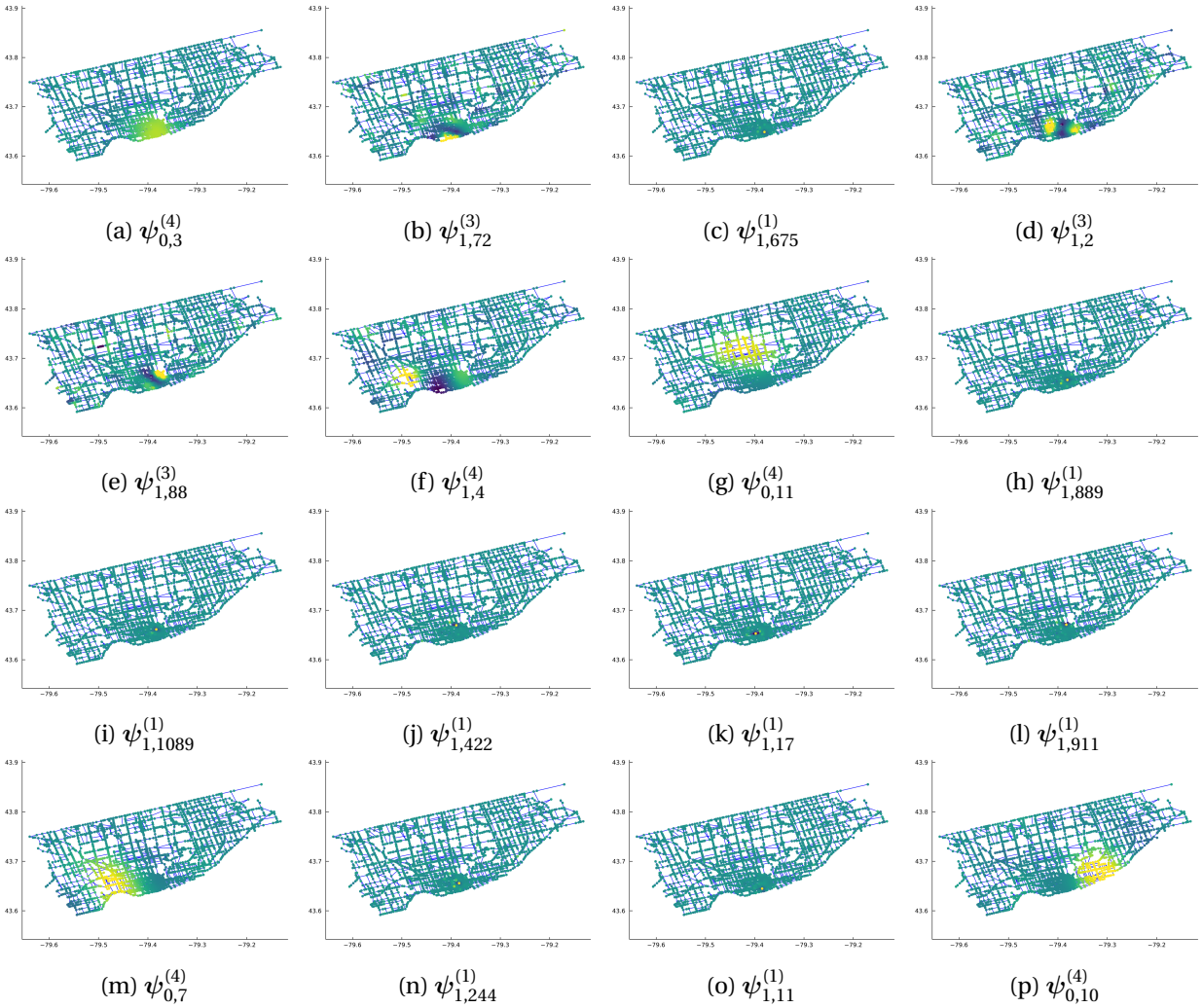
(e) $\psi^{(1)}_{0,583}$ (f) $\psi^{(1)}_{1,1188}$ (g) $\psi^{(1)}_{0,581}$ (h) $\psi^{(1)}_{0,585}$

(i) $\psi^{(1)}_{1,906}$ (j) $\psi^{(1)}_{0,582}$ (k) $\psi^{(1)}_{1,1421}$ (l) $\psi^{(1)}_{1,1685}$

(m) $\psi^{(1)}_{1,1159}$ (n) $\psi^{(1)}_{1,1214}$ (o) $\psi^{(1)}_{1,1419}$ (p) $\psi^{(1)}_{1,1203}$

FIGURE 9.10. Sixteen most significant PC-NGWP best basis vectors for pedestrian volume data on the Toronto street map. The basis vector amplitudes within $(-0.075, 0.075)$ are mapped to the viridis colormap.

those fine scale basis vectors in the PC-NGWP best basis for this graph signal. Lastly, the top LP-NGWP best basis vectors exhibit similar scale structures with those in the VM-NGWP. In fact, as one can see from Figure 9.11, there are several selected top LP-NGWP basis vectors that also appeared in Figure 9.9. Also, we note that those common basis vectors are at levels $j = 1, 3, 4$. Yet, the level $j = 2$ basis vectors that nicely characterized the local structure of the pedestrian volume within the downtown region in Figure 9.9 are not shown. Considering the LP-NGWP dictionary actually reduces to the VM-NGWP dictionary with $\epsilon = 0$, this phenomenon could be due to the inappropriate choice of ϵ, i.e., the choice of



(a) $\psi^{(4)}_{0,3}$ (b) $\psi^{(3)}_{1,72}$ (c) $\psi^{(1)}_{1,675}$ (d) $\psi^{(3)}_{1,2}$

(e) $\psi^{(3)}_{1,88}$ (f) $\psi^{(4)}_{1,4}$ (g) $\psi^{(4)}_{0,11}$ (h) $\psi^{(1)}_{1,889}$

(i) $\psi^{(1)}_{1,1089}$ (j) $\psi^{(1)}_{1,422}$ (k) $\psi^{(1)}_{1,17}$ (l) $\psi^{(1)}_{1,911}$

(m) $\psi^{(4)}_{0,7}$ (n) $\psi^{(1)}_{1,244}$ (o) $\psi^{(1)}_{1,11}$ (p) $\psi^{(4)}_{0,10}$

FIGURE 9.11. Sixteen most significant LP-NGWP best basis vectors for pedestrian volume data on the Toronto street map. The basis vector amplitudes within $(-0.075, 0.075)$ are mapped to the viridis colormap.

$\epsilon = 0.3$ was too large, such that the LP-NGWP basis vectors did not perform well in approximating this highly localized signal. We also note that the uniform choice of ϵ across all levels might not be the best for graph signal approximations. So, how can we appropriately *and* adaptively set ϵ at each level in the LP-NGWP dictionary forms one of the future research projects.

CHAPTER 10

Natural Graph Wavelet Frames

Given $G = (V, E, W)$ with $|V| = N$, its Laplacian eigenvectors[1] (i.e., $\{\phi_l\}_{l=0:N-1}$), *and* an eigenvector distance d from Table 5.1, we constructed the dual graph $G^\star(V^\star, E^\star, W^\star)$ in Chapter 5. In Chapters 6, 7 and 8, we introduced three methods of building the natural graph wavelet packet dictionaries. In this chapter, we utilize the eigenvector distance d and mimic the framework of building the SGWT frames in Eq. (2.4) to construct the *Natural Graph Wavelet Frame* (NGWF). We then propose an algorithm to reduce its redundancy and get the *reduced Natural Graph Wavelet Frame* (rNGWF). Finally, we show several numerical experiments to demonstrate their usefulness.

10.1. Natural Spectral Graph Filters

The spectral graph filters as introduced in Section 2.2, i.e., the diagonal matrices $F_j \in \mathbb{R}^{N\times N}$, are built by the smooth function pair (h, g) of the eigenvalues accompanied with a set of dilation parameters $\{s_j\}_{j=1:J}$ [**29**],

$$(F_0)_{i,i} = h(\lambda_{i-1}) \quad \text{and} \quad (F_j)_{i,i} = g(s_j\lambda_{i-1}), \qquad i = 1:N,$$

where the index j stands for different scale of spectral filtering (the greater j, the finer the scale, and $J \in \mathbb{N}$ represents the finest scale specified by the user). As we emphasised throughout this dissertation (e.g., see Chapter 3), it is not reliable to organize the graph Laplacian eigenvectors by the corresponding eigenvalues. Therefore, one may potentially face serious problems using such smooth function pair (h, g) and a set of dilation parameters $\{s_j\}_{j=1:J}$ over the eigenvalue distribution to assemble the spectral graph filters. Instead, we create a set of *natural spectral graph filters* $\{\mathcal{F}_l\}_{l=0:N-1}$[2] centered at each node of $V^\star$ (i.e., ϕ_l) via the eigenvector distance d. The diagonal entries of the diagonal matrix $\mathcal{F}_l \in \mathbb{R}^{N\times N}$ are

[1] We can choose either the eigenvectors of $L(G)$ or the eigenvectors of $L_{\text{sym}}(G)$, since they both form an ONB for $\mathbb{R}^N$. However, we mainly use the former for demonstration purpose as before.

[2] Here we discard the subscript j, because the term "scale" is not well-defined on $G^\star$. Instead, we use the subscript l to represent the dual graph node $\phi_l \in V^\star$ that $\mathcal{F}_l$ centers on.

defined by

$$(\mathcal{F}_l)_{i+1,i+1} := \frac{\exp\left(-(d(\boldsymbol{\phi}_i,\boldsymbol{\phi}_l)/\sigma)^2\right)}{\sum_{i'=0}^{N-1}\exp\left(-(d(\boldsymbol{\phi}_{i'},\boldsymbol{\phi}_l)/\sigma)^2\right)}, \qquad i=0:N-1 \text{ and } l=0:N-1, \tag{10.1}$$

where σ is a scale parameter for the distances and $\boldsymbol{\phi}_l$ is the *centered* eigenvector of $\mathcal{F}_l$. In particular, we define σ as follows.

$$\sigma := c\cdot d_{\max}, \qquad \exists c>0 \text{ and } d_{\max} := \max_{\boldsymbol{\phi}_i,\boldsymbol{\phi}_j\in V^\star} d(\boldsymbol{\phi}_i,\boldsymbol{\phi}_j). \tag{10.2}$$

For convenience, we denote $\boldsymbol{\mu}_l \in \mathbb{R}^N_{>0}$ by the diagonal entries of $\mathcal{F}_l$, i.e., $\boldsymbol{\mu}_l = \mathrm{diag}(\mathcal{F}_l)$ for $l = 0:N-1$. Note that $\sum_{i=0}^{N-1}\mu_l(i+1) = 1$ thanks to the normalization factor in Eq. (10.1). Thus, $\{\boldsymbol{\mu}_l\}_{l=0:N-1}$ can be viewed as a set of pmfs on the dual graph $G^\star$. Moreover, $\{\boldsymbol{\mu}_l\}_{l=0:N-1}$ can be also viewed as a set of *Gaussian window functions* (with different window width parameter σ) on $G^\star$ centered at each node in $V^\star = \{\boldsymbol{\phi}_l\}_{l=0:N-1}$. Note also that the corresponding Gaussian kernels use the eigenvector distance d instead of the Euclidean distance in the dual domain in general, except for the case when the input graph is a path P_N or a cycle C_N that we utilize their special properties as follows.

Let us take P_N as an example. Based on Remark 5.1.2, $P_N^\star = P_N$. Therefore, the ground truth metric between the *DCT type-II* basis vectors is defined by the Euclidean distance in the dual domain.

$$d_{\mathrm{DCT}}(\boldsymbol{\phi}_i,\boldsymbol{\phi}_j) := |i-j|, \qquad i,j=0:N-1.$$

For instance, when $N = 128$ and $l = 63$, Figure 10.1 displays the diagonal entries of the natural spectral graph filters $\mathcal{F}_{63}$, i.e., $\boldsymbol{\mu}_{63} = \mathrm{diag}(\mathcal{F}_{63})$, which is built by the distance $d = d_{\mathrm{DCT}}$ along with three different values of scale parameter σ on the dual graph $P_{128}^\star$. As we can see from this figure, these three $\boldsymbol{\mu}_{63}$ are truly the Gaussian window functions on $P_{128}^\star$. In addition, $\boldsymbol{\mu}_{63}$ with a larger σ corresponds to a larger window width Gaussian function centered at $\boldsymbol{\phi}_{63}$.

On the other hand, if the underlying graph G is not P_N or C_N, we choose other non-trivial *behavioral* eigenvector distance d from Table 5.1. By doing so, the Gaussian pmf $\boldsymbol{\mu}_l$ on $G^\star$ is essentially windowing the eigenvectors in $V^\star$ that have similar "behaviors" as $\boldsymbol{\phi}_l$.

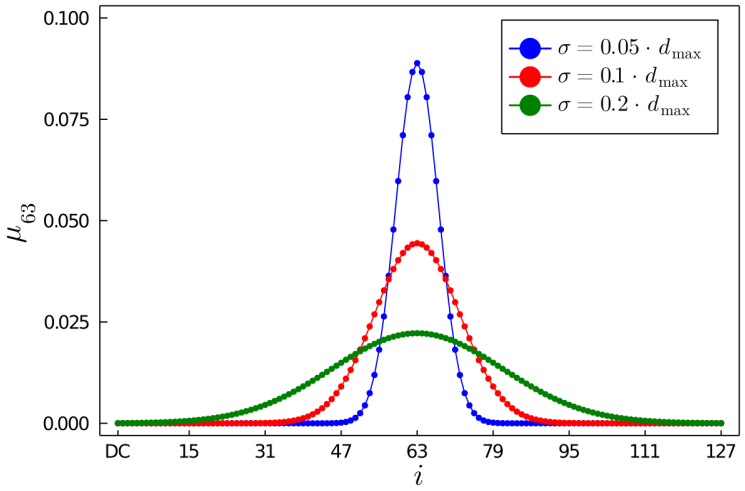


FIGURE 10.1. μ_{63} ($d = d_{\mathrm{DCT}}$) with three different σ on $P^{\star}_{128}$.

10.2. Generating Redundant Natural Graph Wavelet Frame

With the set of natural spectral graph filters $\{\mathcal{F}_l\}_{l=0:N-1}$ in place, we generate the *Natural Graph Wavelet Frame* (NGWF) vectors based on the general framework provided by Eq. (2.4).

$$\boldsymbol{\psi}_{l,x} := \Phi \mathcal{F}_l \Phi^{\mathsf{T}} \boldsymbol{\delta}_x, \qquad l = 0 : N-1 \text{ and } x = 1 : N. \tag{10.3}$$

Note that $\boldsymbol{\psi}_{l,x}$ is a wavelet vector that centers on $\phi_l \in V^{\star}$ in the dual graph and concentrates on $x \in V$ in the primal graph. The obtained $\{\boldsymbol{\psi}_{l,x}\}_{l=0:N-1;x=1:N}$ contains N^2 number of wavelet vectors in total and it forms a redundant *graph wavelet frame* for $\mathbb{R}^N$ because of the following theorem.

THEOREM 10.2.1. *The natural spectral graph filters* $\{\mathcal{F}_l\}_{l=0:N-1}$ *built by an eigenvector distance* d *satisfies the generalized partition of unity in Eq.* (2.5). *In particular,*

$$\frac{1}{N} \cdot I_N < \sum_{l=0}^{N-1} \mathcal{F}_l < \frac{N+1}{2} \cdot I_N.$$

See Appendix A.5 for the proof.

For convenience, we introduce a *sequential index set* Γ of the NGWF.

$$\Gamma := \left\{ \gamma_i \,|\, \gamma_i = (l,x), i = N \cdot l + x, l = 0 : N-1, x = 1 : N \right\}, \qquad |\Gamma| = N^2.$$

We then can represent the NGWF as $\left\{\boldsymbol{\psi}_{\gamma_i}\right\}_{\gamma_i \in \Gamma}$.

Given a graph signal $\boldsymbol{f} \in \mathbb{R}^N$, we denote the *natural graph wavelet frame operator* $\mathscr{U} : \mathbb{R}^N \mapsto \mathbb{R}^{N^2}$ as

$$\mathscr{U}\boldsymbol{f}(i) = \langle \boldsymbol{\psi}_{\gamma_i}, \boldsymbol{f} \rangle, \quad \gamma_i \in \Gamma,$$

e.g., see [**17**, Chap. 3]. With a slight abuse of notation, let $\mathscr{U}$ also represent *the corresponding matrix* of the frame operator, i.e., $\mathscr{U} \in \mathbb{R}^{N^2 \times N}$.

Applying $(\mathscr{U}^\top \mathscr{U})^{-1}$ to the NGWF vectors, we get a new set of vectors denoted by $\{\widetilde{\boldsymbol{\psi}}_{\gamma_i}\}_{\gamma_i \in \Gamma}$,

$$\widetilde{\boldsymbol{\psi}}_{\gamma_i} = (\mathscr{U}^\top \mathscr{U})^{-1} \boldsymbol{\psi}_{\gamma_i}.$$

By the Proposition 3.2.3 in [**17**, Chap. 3], we know $\{\widetilde{\boldsymbol{\psi}}_{\gamma_i}\}_{\gamma_i \in \Gamma}$ is also a wavelet frame and we call it the *dual natural graph wavelet frame*. We use $\mathscr{V}$ to denote its corresponding frame operator and we call it the *dual natural graph wavelet frame operator*, which is defined by $\mathscr{V} := \mathscr{U}(\mathscr{U}^\top \mathscr{U})^{-1} \in \mathbb{R}^{N^2 \times N}$.

Now, one can decompose the graph signal $\boldsymbol{f}$ to get the expansion coefficient vector $\boldsymbol{\alpha} \in \mathbb{R}^{N^2}$ by the *NGWF transform*,

$$\boldsymbol{\alpha} = \mathscr{U}\boldsymbol{f} \in \mathbb{R}^{N^2},$$

and recover it by the *inverse NGWF transform*,

$$\mathscr{V}^\top \boldsymbol{\alpha} = \left(\mathscr{U}(\mathscr{U}^\top \mathscr{U})^{-1}\right)^\top \mathscr{U}\boldsymbol{f} = (\mathscr{U}^\top \mathscr{U})^{-1} \mathscr{U}^\top \mathscr{U}\boldsymbol{f} = \boldsymbol{f},$$

which is equivalent to

$$\sum_{\gamma_i \in \Gamma} \alpha(i) \widetilde{\boldsymbol{\psi}}_{\gamma_i} = \sum_{\gamma_i \in \Gamma} \langle \boldsymbol{\psi}_{\gamma_i}, \boldsymbol{f} \rangle \widetilde{\boldsymbol{\psi}}_{\gamma_i} = \boldsymbol{f}. \tag{10.4}$$

Eq. (10.4) provides a way to approximate the input graph signal via a frame. See Algorithm 4 for the details.

On the other hand, the NGWF transform can be interpreted as follows. Let us first reshape the expansion coefficient vector $\boldsymbol{\alpha} \in \mathbb{R}^{N^2}$ into a 2D manner:

$$\alpha_2(l, x) := \langle \boldsymbol{\psi}_{l,x}, \boldsymbol{f} \rangle = \boldsymbol{\delta}_x^\top \Phi \mathcal{F}_l \Phi^\top \boldsymbol{f}, \qquad l = 0 : N-1 \text{ and } x = 1 : N.$$

Algorithm 4: Graph Signal Approximation via a Frame

Input: A graph signal $\boldsymbol{f} \in \mathbb{R}^N$; a set of frame vectors $\{\boldsymbol{\psi}_{\gamma_i}\}_{\gamma_i \in \Gamma}$; the corresponding set of dual frame vectors $\{\widetilde{\boldsymbol{\psi}}_{\gamma_i}\}_{\gamma_i \in \Gamma}$; and the total number of kept coefficients $K_p \in [1, |\Gamma|]$ (default: $\lfloor 0.5 \cdot |\Gamma| \rfloor$).

Output: The relative approximation error vector relerror $\in \mathbb{R}^{K_p+1}$.

relerror = [1.0] // initialize the relative approximation error vector

$\boldsymbol{\alpha}$ = zeros($|\Gamma|$) // initialize the expansion coefficient vector

for $i = 1 : |\Gamma|$ **do**

  $\alpha(i) = \langle \boldsymbol{\psi}_{\gamma_i}, \boldsymbol{f} \rangle$

end

ind = sortperm(abs.($\boldsymbol{\alpha}$)) // get a permutation vector of indices of abs.($\boldsymbol{\alpha}$) // that puts it in decreasing sorted order

$\boldsymbol{f}_{\text{approx}}$ = zeros(N) // initialize the approximated graph signal

for $j = 1 : K_p$ **do**

  i = ind(j)

  $\boldsymbol{f}_{\text{approx}} = \boldsymbol{f}_{\text{approx}} + \alpha(\gamma_i)\, \widetilde{\boldsymbol{\psi}}_{\gamma_i}$ // update $\boldsymbol{f}_{\text{approx}}$

  relerror $\leftarrow$ relerror $\cup \{\|\boldsymbol{f} - \boldsymbol{f}_{\text{approx}}\|_2 / \|\boldsymbol{f}\|_2\}$

end

return relerror

If we look at the right hand side of the above formula closely, there are four steps involved in the process of the NGWF transform:

Step 1: Perform the graph Fourier transform of $\boldsymbol{f}$ to get its graph Fourier coefficient vector $\boldsymbol{g} \in \mathbb{R}^N$, i.e., $\boldsymbol{g} = \Phi^\top \boldsymbol{f}$. Note that $\boldsymbol{g}$ can be also viewed as a graph signal on $G^\star$.

Step 2: Windowing $\boldsymbol{g}$ by the Gaussian filter $\mathcal{F}_l$ (with the scale parameter σ) to get a well-localized (in terms of the eigenvector distance d) slice of $\boldsymbol{g}$ on $G^\star$, i.e., $\mathcal{F}_l \boldsymbol{g} =: \boldsymbol{g}_w$.

Step 3: Perform the inverse graph Fourier transform of the windowed signal to get a partially recovered graph signal $\boldsymbol{f}_w$ on G, i.e., $\Phi \boldsymbol{g}_w =: \boldsymbol{f}_w$.

Step 4: Extract the x-th entry of $\boldsymbol{f}_w$ to get the NGWF transform expansion coefficient $\alpha_2(l, x)$, i.e., $\alpha_2(l, x) = \boldsymbol{\delta}_x^\top \boldsymbol{f}_w$.

Hence, we can view the NGWF transform as the graph Fourier transform followed by the *windowed* inverse graph Fourier transform. The two dimensional structure of the set of expansion coefficients $\{\alpha_2(l, x)\}_{l=0:N-1; x=1:N}$ reveals the information of the graph signal $\boldsymbol{f}$ in both G and $G^\star$. We denote the expansion coefficient matrix of the NGWF transform as $A \in \mathbb{R}^{N \times N}$ with $A_{l+1,x} := \alpha_2(l, x)$, for $l = 0 : N-1$ and $x = 1 : N$. Moreover, we call the heatmap plot of the matrix abs.(A) as the *NGWF spectrogram*.

10.3. Reducing the Redundancy of Natural Graph Wavelet Frame

Comparing to the Spectral Graph Wavelet Transform [**29**] (SGWT) frame $\left\{\psi_{j,x}^{\mathrm{SGWT}}\right\}_{j=0:J;x=1:N}$, the NGWF $\{\boldsymbol{\psi}_{\gamma_i}\}_{\gamma_i\in\Gamma}$ contains too many frame vectors. This is not very efficient for graph signal approximation and we should reduce its redundancy. To achieve that, we change two places when generating the NGWF in Eq. (10.3).

1. After constructing the set of natural spectral graph filters $\{\mathcal{F}_l\}_{l=0:N-1}$ by the eigenvector metric d, we zero out some of the diagonal entries of $\mathcal{F}_l$, i.e., the entries of $\boldsymbol{\mu}_l$, if they are smaller than a preset threshold. By doing so, we get a more compactly supported set of filters $\left\{\overline{\mathcal{F}}_l\right\}_{l=0:N-1}$.
2. For each $l = 0:N-1$, the matrix $\overline{H}_l := \Phi\overline{\mathcal{F}}_l\Phi^{\mathsf{T}}$ is *rank deficient* depending on the preset threshold in Item 1 above. Therefore, we use the *pivoted QR* algorithm to select a subset of nodes for subsampling.

We denote the resulting wavelet frame as $\left\{\overline{\boldsymbol{\psi}}_\gamma\right\}_{\gamma\in\overline{\Gamma}}$ where $\overline{\Gamma}\subset\Gamma$ and $|\overline{\Gamma}|\ll|\Gamma| = N^2$, and we call it the *reduced Natural Graph Wavelet Frame* (rNGWF). See Algorithm 5 for the details. Note that the rNGWF may contain global Laplacian eigenvectors because of the two changes mentioned above. If an eigenvector $\boldsymbol{\phi}_l$ is far from all the others in terms of the eigenvector distance d, we could obtain $\mathrm{diag}\left(\overline{\mathcal{F}}_l\right) = \mathrm{const}\cdot\boldsymbol{\delta}_{l+1}$ by the cut-off procedure in the first change. Then, the obtained $\overline{H}_l$ is clearly a rank one matrix. Therefore, the pivoted QR algorithm used in the second change will generate one wavelet vector from $\overline{H}_l$, which is $\mathrm{const}\cdot\boldsymbol{\phi}_l$.

10.4. Computational Complexity

The algorithm of generating the NGWF is of iterative nature. Assume the graph Laplacian eigenvectors $\Phi\in\mathbb{R}^{N\times N}$ and their pairwise eigenvector distances are provided. For $l = 0:N-1$, the costs within each iteration are: 1) $O(N)$ for assembling the natural spectral graph filter $\mathcal{F}_l$; 2) $O(N^3)$ for computing the matrix $H_l = \Phi\mathcal{F}_l\Phi^{\mathsf{T}}$ for future use; and 3) $O(N^2)$ for getting the NGWF vector $\boldsymbol{\psi}_{l,x} = H_l\,\boldsymbol{\delta}_x$ for $x = 1:N$ since multiplying $\boldsymbol{\delta}_x$ is equivalent to selecting the x-th column of H_l. Hence, the total cost of generating the NGWF is $O\left(N\cdot(N+N^3+N^2)\right) = O(N^4)$.

On the other hand, generating the rNGWF algorithm (i.e., Algorithm 5) just adds two additional steps in each iteration of l in above algorithm: 1) $O(N)$ for the cut-off procedure of $\mathcal{F}_l$ and 2) $O(N^3)$ for the

Algorithm 5: Generating the reduced Natural Graph Wavelet Frame (rNGWF)

Input: The graph Laplacian eigenvectors $\Phi \in \mathbb{R}^{N\times N}$; an eigenvector metric d; a scale parameter $\sigma > 0$ (default: $0.2 \cdot d_{\max}$, where $d_{\max}$ is defined in Eq. (10.2)) and a threshold $\text{thres} \in (0,1)$ (default: 0.15).

Output: The rNGWF $\overline{\Psi} = \left\{\overline{\psi}_\gamma\right\}_{\gamma\in\overline{\Gamma}}$.

$\overline{\Psi} = \emptyset$ // initialize the output rNGWF

$\overline{\Gamma} = \emptyset$ // initialize the rNGWF index set

for $l = 0 : N-1$ **do**

  $\overline{\boldsymbol{\mu}}_l = \text{zeros(N)}$ // initialize the filter vector

  for $i = 1 : N$ **do**

    $\overline{\mu}_l(i) = \exp\left(-(d(\phi_{i-1}, \phi_l)/\sigma)^2\right)$

  end

  $\overline{\boldsymbol{\mu}}_l = \overline{\boldsymbol{\mu}}_l / \sum_{i=1}^{N} \overline{\mu}_l(i)$ // normalize the filter vector

  for $i = 1 : N$ **do**

    if $\overline{\mu}_l(i) \le \text{thres} \cdot \max\left(\overline{\boldsymbol{\mu}}_l\right)$ **then**

      $\overline{\mu}_l(i) = 0$ // zero out the small entries

    end

  end

  $\overline{\mathcal{F}_l} = \text{diag}\left(\overline{\boldsymbol{\mu}}_l\right)$

  $\overline{H}_l = \Phi \overline{\mathcal{F}}_l \Phi^{\mathsf{T}}$

  $Q, R, \Pi = \text{pivotedQR}\left(\overline{H}_l\right)$ // apply the pivoted QR algorithm s.t. $\overline{H}_l \Pi = QR$

  $r = \text{rank}\left(\overline{H}_l\right)$ // compute the numerical rank of $\overline{H}_l$ by SVD

  // s.t., $\sigma_r\left(\overline{H}_l\right) > (10^4 \epsilon_{\text{mach}}) \cdot \sigma_1\left(\overline{H}_l\right) \ge \sigma_{r+1}\left(\overline{H}_l\right)$,

  // where ϵ_{mach} is the machine epsilon.

  for $i = 1 : r$ **do**

    $\boldsymbol{\delta}_x = \Pi[:, i]$ // select the i-th column of the permutation matrix Π

    $\gamma = (l, x)$

    $\overline{\Gamma} \leftarrow \overline{\Gamma} \cup \{\gamma\}$

    $\overline{\psi}_\gamma = \overline{H}_l \boldsymbol{\delta}_x$

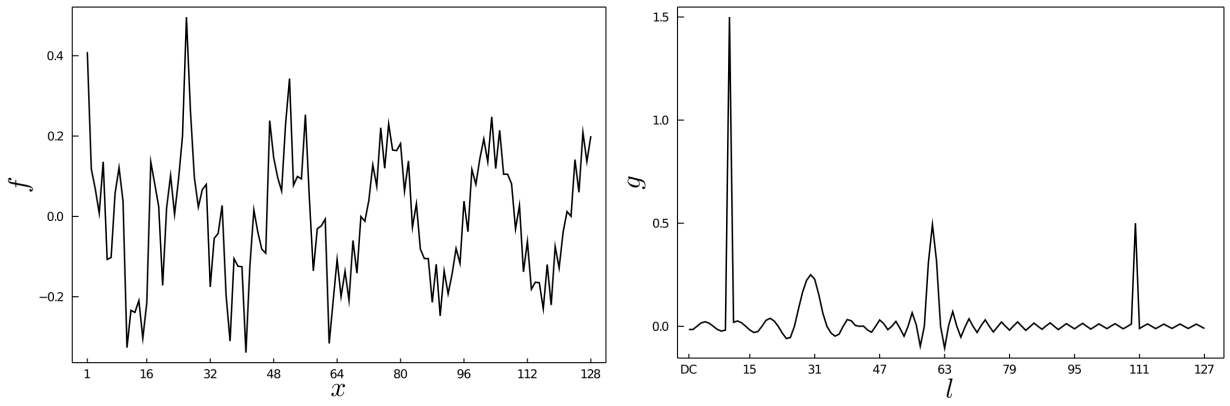
    $\overline{\Psi} \leftarrow \overline{\Psi} \cup \left\{\overline{\psi}_\gamma\right\}$

  end

end

return $\overline{\Psi}$

pivoted QR algorithm and computing the numerical rank of $\overline{H}_l \in \mathbb{R}^{N\times N}$ since they essentially use the SVD of an $N \times N$ matrix. Hence, the total operations of generating the rNGWF is also $O(N^4)$.



(a) $\boldsymbol{f}$ on P_{128}. (b) $\boldsymbol{g} = \Phi^{\mathsf{T}}\boldsymbol{f}$ on $P^{\star}_{128}$.

FIGURE 10.2. (a) The graph signal $\boldsymbol{f}$ as defined in Eq. (10.5); (b) its graph Fourier transform coefficient $\boldsymbol{g}$.

10.5. Numerical Experiments

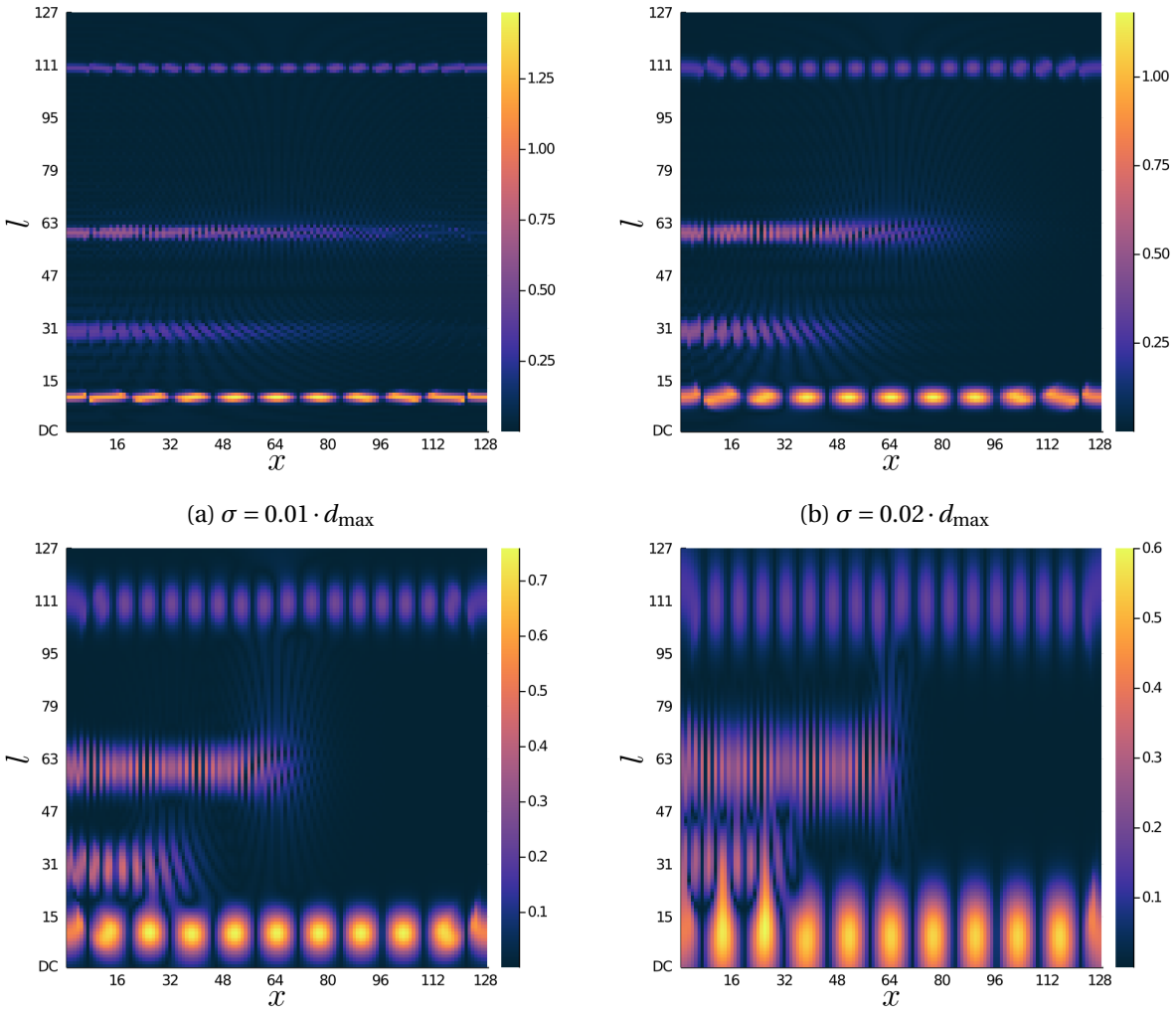
Here we show four experimental results of NGWF and rNGWF on three graphs: P_{128}, $P_{23} \times P_{22}$[3] and the synthetic dendritic tree introduced in Section 5.3.2.

10.5.1. NGWF Spectrograms on P_{128}. We first examine the results of the NGWF spectrograms on P_{128} (where $d = d_{\mathrm{DCT}}$ is used as the eigenvector metric). As an example, we create a graph signal $\boldsymbol{f} \in \mathbb{R}^{128}$ as follows:

$$f(x) = \begin{cases} 1.5\phi_{10}(x) + \phi_{30}(x) + \phi_{60}(x) + 0.5\phi_{110}(x), & x = 1:32, \\ 1.5\phi_{10}(x) + \phi_{60}(x) + 0.5\phi_{110}(x), & x = 33:64, \\ 1.5\phi_{10}(x) + 0.5\phi_{110}(x), & x = 65:128, \end{cases} \tag{10.5}$$

where ϕ_l ($l = 10, 30, 60, 110$) are the DCT type-II basis vectors as given in Eq. (2.2). Figure 10.2a shows how the graph signal $\boldsymbol{f}$ looks like on P_{128}, while Figure 10.2b shows how the graph Fourier transform coefficient vector $\boldsymbol{g}$ looks like on $P^{\star}_{128}$. This graph signal $\boldsymbol{f}$ contains a strong low-frequency sinusoid $1.5\phi_{10}$ and a weak high-frequency sinusoid $0.5\phi_{110}$ over the entire path. It also contains the sinusoid ϕ_{60} over the first half of the path and the sinusoid ϕ_{30} over the first quarter of the path. Consequently, its graph Fourier transform coefficient vector $\boldsymbol{g}$ contains four peaks at ϕ_l ($l = 10, 30, 60, 110$). Given a

[3]We choose the peculiar numbers $N_x = 23$ and $N_y = 22$ because they are relatively prime to each other so that no multiple eigenvalues of the graph Laplacian exist. See Remarks 2.1.1, 2.1.2 and Section 3.1 for the details.



(a) $\sigma = 0.01 \cdot d_{\max}$

(b) $\sigma = 0.02 \cdot d_{\max}$

(c) $\sigma = 0.05 \cdot d_{\max}$

(d) $\sigma = 0.1 \cdot d_{\max}$

FIGURE 10.3. The NGWF spectrograms of $\boldsymbol{f}$, i.e., abs.(A) as computed in Eq. (10.6), with four different values of the scale parameter σ displayed in heatmap plots.

scale parameter σ of the natural spectral graph filters, we can compute the NGWF expansion coefficient matrix $A \in \mathbb{R}^{128\times 128}$ with,

$$A_{l+1,x} = \langle \boldsymbol{\psi}_{l,x}, \boldsymbol{f} \rangle, \quad l = 0:127 \text{ and } x = 1:128. \tag{10.6}$$

Figure 10.3 shows four NGWF spectrograms, i.e., abs.(A) generated by $\{\boldsymbol{\psi}_{\gamma_i}\}_{\gamma_i \in \Gamma}$ with four different values of the scale parameter σ. Overall, they showed the four frequency components of $\boldsymbol{f}$ (i.e., the four peaks of $\boldsymbol{g}$ on $P^{\star}_{128}$) and their corresponding active regions on P_{128} with different resolutions. As σ gets

larger, the NGWF spectrograms show more accurate and localized information about the active region of each frequency component of $\boldsymbol{f}$ on P_{128}; while as σ gets smaller, the NGWF spectrograms show more accurate and localized information about the four peaks of $\boldsymbol{g}$ on $P^{\star}_{128}$. This phenomenon is due to the *Heisenberg uncertainty principle* (e.g., see [**23**, Sec. 7.3]). In particular, let us consider two extreme cases of σ:

Case 1: $\sigma = 0$. Then, $\boldsymbol{\mu}_l = \boldsymbol{\delta}_{l+1}$ for $l = 0:127$.

$$\boldsymbol{\psi}_{l,x} = \Phi\,\mathrm{diag}(\boldsymbol{\delta}_{l+1})\,\Phi^{\mathsf{T}}\,\boldsymbol{\delta}_x = \langle \boldsymbol{\phi}_l, \boldsymbol{\delta}_x \rangle\,\boldsymbol{\phi}_l.$$

Therefore, $\boldsymbol{\psi}_{l,x}$ is essentially the eigenvector $\boldsymbol{\phi}_l$ multiplied by the constant $\boldsymbol{\phi}_l(x)$ for $x = 1:128$. The NGWF transform of $\boldsymbol{f}$ becomes

$$A_{l+1,x} = \langle \boldsymbol{\phi}_l, \boldsymbol{\delta}_x \rangle \cdot \langle \boldsymbol{\phi}_l, \boldsymbol{f} \rangle = \boldsymbol{\phi}_l(x) \cdot g(l+1), \qquad l = 0:127 \text{ and } x = 1:128,$$

i.e., the x-th column of A is $\boldsymbol{\phi}_l(x) \cdot \boldsymbol{g}$. Hence, A does not contain any explicit information about $\boldsymbol{f}$ on P_{128}.

Case 2: $\sigma = \infty$. Then, $\boldsymbol{\mu}_l = \mathbf{1}/N$ (i.e., the uniform pmf) for $l = 0:127$.

$$\boldsymbol{\psi}_{l,x} = \frac{1}{N}\Phi\, I_N\, \Phi^{\mathsf{T}}\,\boldsymbol{\delta}_x = \frac{1}{N}\boldsymbol{\delta}_x.$$

Therefore, $\boldsymbol{\psi}_{l,x}$ is essentially the standard basis vector $\boldsymbol{\delta}_x$ divided by the constant N for $x = 1:128$. The NGWF transform of $\boldsymbol{f}$ becomes

$$A_{l+1,x} = \frac{1}{N}\,\langle \boldsymbol{\delta}_x, \boldsymbol{f} \rangle = \frac{1}{N} f(x), \qquad l = 0:127 \text{ and } x = 1:128,$$

i.e., the $(l+1)$-th row of A is $\frac{1}{N} \cdot \boldsymbol{f}$. Hence, A does not contain any explicit information about $\boldsymbol{g}$ on $P^{\star}_{128}$.

Thus, there is a trade-off of the explicit information A contained between $\boldsymbol{f}$ on P_{128} and $\boldsymbol{g}$ on $P^{\star}_{128}$ by selecting different σ. Especially, we can see a very nice and interpretable NGWF spectrogram of $\boldsymbol{f}$ from Figure 10.3c ($\sigma = 0.05 \cdot d_{\max}$). It roughly reveals the four peaks of $\boldsymbol{g}$ on $P^{\star}_{128}$ with their corresponding strength, i.e., $1.5\boldsymbol{\phi}_{10}$, $\boldsymbol{\phi}_{30}$, $\boldsymbol{\phi}_{60}$ and $0.5\boldsymbol{\phi}_{110}$, and it also approximately reveals the interval duration for

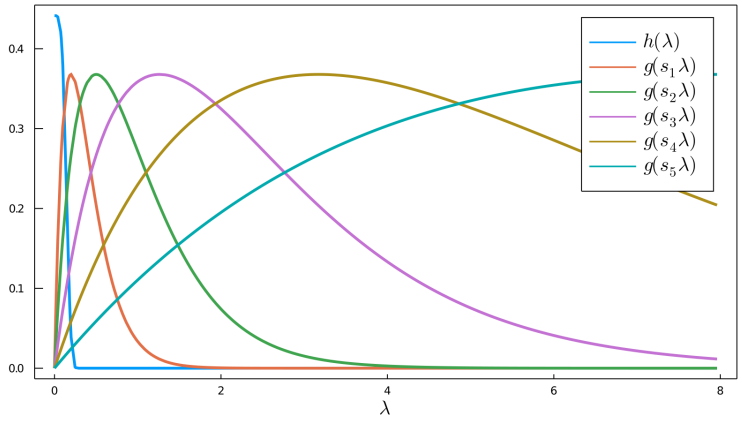


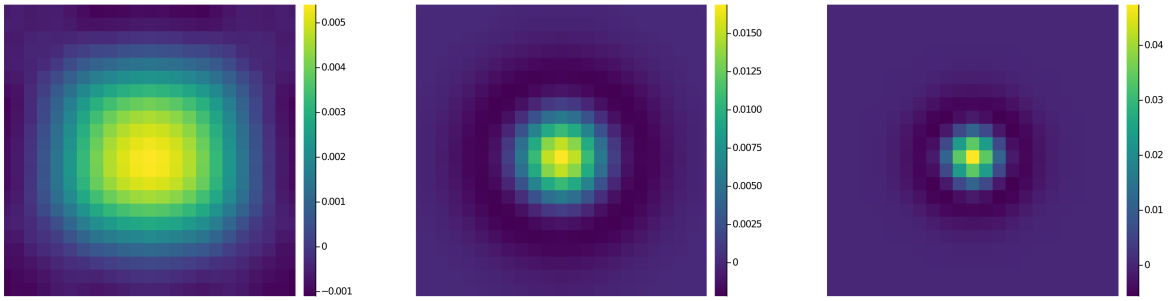
FIGURE 10.4. The SGWT filter bank on $P_{23} \times P_{22}$.

each frequency component on P_{128}, i.e., 1) $\phi_{30}(x)$ is active for $x = 1:32$; 2) $\phi_{60}(x)$ is active for $x = 1:64$; and 3) $\phi_{10}(x)$ and $\phi_{110}(x)$ are active for the whole path.

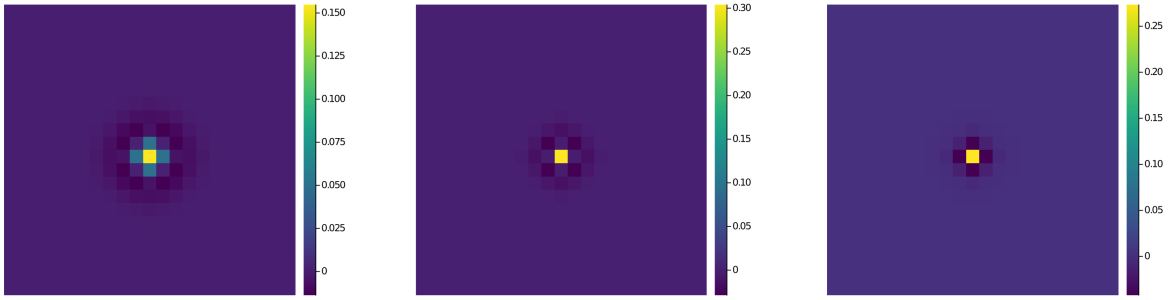
10.5.2. Anisotropic Wavelets on $P_{23} \times P_{22}$. Having shown the NGWF spectrograms with $d = d_{\mathrm{DCT}}$ on a path graph, we now present the use of NGWF with the non-trivial eigenvector distance $d = d_{\mathrm{DAG}}$ on a lattice graph $P_{23} \times P_{22}$ ($N = 506$). As discussed in Section 5.4, the DAG pseudometric is very good at detecting the directional oscillation patterns of the eigenvectors. In particular, it perfectly recovers the horizontal/vertical oscillation patterns of the eigenvectors on lattice graphs (e.g., see Figure 5.5). We will examine the performance of the frame vectors generated by the NGWF transform by comparing it with the SGWT frame vectors.

In Figure 10.4, we use the `PyGSP` [**18**] package to generate the SGWT filter bank (i.e., $\{F_j\}_{j=0:J}$ with $J = 5$) over the eigenvalues of $L(P_{23} \times P_{22})$. See Section 2.2 for the details. We then use the package to construct the corresponding SGWT frame $\left\{\psi^{\mathrm{SGWT}}_{j,x}\right\}_{j=0:5;x=1:506}$ on $P_{23} \times P_{22}$. For demonstration purpose, we select $x = 242$ (in the middle of the grids) and display $\psi^{\mathrm{SGWT}}_{j,242}$ for $j = 0:5$ in Figure 10.5. As we can see from the figure, there are six scales (from coarse to fine) of the SGWT frame vectors localized at the node $x = 242$, but they do not have any anisotropic oscillatory patterns.

On the other hand, we use d_{DAG} and $\sigma = 0.2 \cdot d_{\max}$ to build the set of natural spectral graph filters $\{\mathcal{F}_l\}_{l=0:505}$ and consequently the NGWF vectors $\{\psi_{l,x}\}_{l=0:505;x=1:506}$. Figure 10.6d, e, f show three examples of $\psi_{l,242}$ with $l = 10, 15, 29$. We can see that these frame vectors have the same anisotropic oscillatory patterns inherited from their centered eigenvectors, i.e., ϕ_{10} (Figure 10.6a), ϕ_{15} (Figure 10.6b), and ϕ_{29}



(a) $\psi^{\mathrm{SGWT}}_{0,242}$ (b) $\psi^{\mathrm{SGWT}}_{1,242}$ (c) $\psi^{\mathrm{SGWT}}_{2,242}$

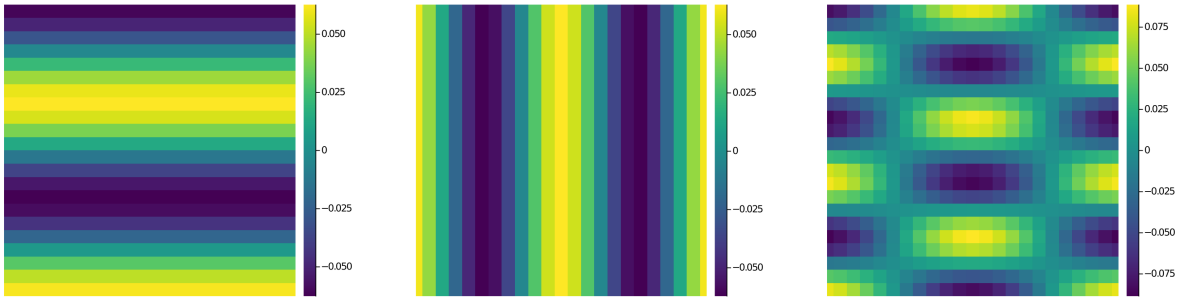


(d) $\psi^{\mathrm{SGWT}}_{3,242}$ (e) $\psi^{\mathrm{SGWT}}_{4,242}$ (f) $\psi^{\mathrm{SGWT}}_{5,242}$

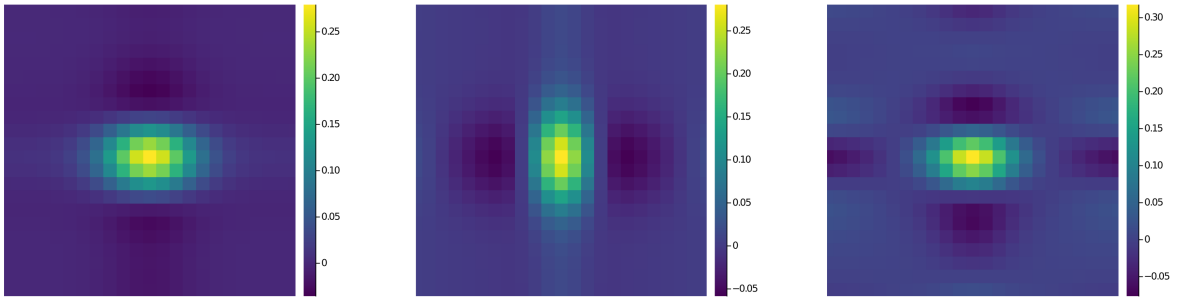
FIGURE 10.5. The SGWT frame vectors $\psi^{\mathrm{SGWT}}_{j,x}$ on $P_{23} \times P_{22}$ with $j = 0 : J$ ($J = 5$) and $x = 242$ (brightest yellow point), which are built by the six spectral graph filters shown in Figure 10.4, respectively.

(Figure 10.6c), respectively. For instance, since ϕ_{10} slowly oscillates only in the vertical direction, d_{DAG} puts the eigenvectors in $V^\star$ that also have only slow vertical oscillations closer to ϕ_{10}. Then, the natural spectral graph filter $\mathcal{F}_{10}$ selects this feature by windowing out the neighborhood of ϕ_l on $G^\star$ via the Gaussian window function with $\sigma = 0.2 \cdot d_{\max}$. Finally, we get the NGWF (vertical) anisotropic wavelet vector $\psi_{10,242} = \Phi \mathcal{F}_{10} \Phi^\top \delta_{242}$. Meanwhile, it is the same story for the case of $\psi_{15,242}$ with ϕ_{15} (horizontal) and the case of $\psi_{29,242}$ with ϕ_{29} (mixed in both directions).

10.5.3. Graph Signal Approximation via Frames on $P_{23} \times P_{22}$. These kinds of anisotropic NGWF vectors on $P_{23} \times P_{22}$ could provide an edge comparing to the SGWT in graph signal approximation. We continue our experiments on $P_{23} \times P_{22}$ by creating a graph signal as follows. First, we select a hand-written digit image "3" from the MNIST dataset [**45**]. It is a 28 by 28 grey scale image. We then overlay $P_{23} \times P_{22}$ in the center of the image such that each node collapses at a pixel, and we take the normalized

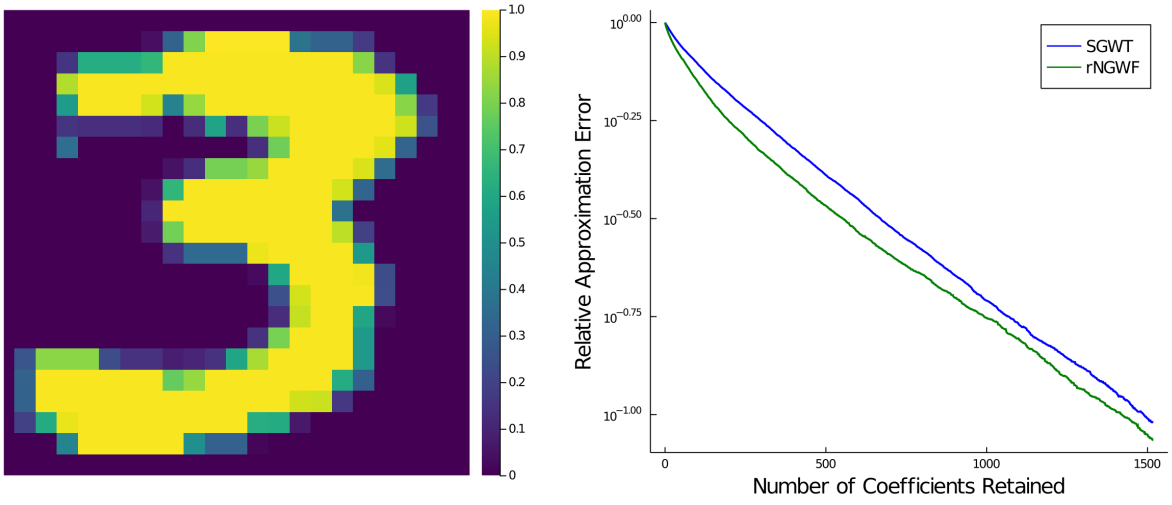


(a) ϕ_{10} (b) ϕ_{15} (c) ϕ_{29}



(d) $\psi_{10,242}$ (e) $\psi_{15,242}$ (f) $\psi_{29,242}$

FIGURE 10.6. Three NGWF vectors (d) (e) (f) on $P_{23} \times P_{22}$ localized at the node $x = 242$ (brightest yellow point), which are built by $\mathcal{F}_l$ ($d = d_{\mathrm{DAG}}$ and $\sigma = 0.2 \cdot d_{\max}$) centered at ϕ_l with $l = 10$ (a), $l = 15$ (b), $l = 29$ (c), respectively.



(a) A hand-written digit “3” graph signal. (b) Approximation performance of two frames.

FIGURE 10.7. The hand-written digit “3” on $P_{23} \times P_{22}$ (a); the results of our approximation experiments (b).

pixel values as the graph signal on $P_{23} \times P_{22}$. Figure 10.7a shows the resulting graph signal in a heatmap plot.

As discussed in Section 10.3, there are too many frame vectors in NGWF compared to the SGWT, i.e., $N^2 = 256036$ vs. $(J+1)N = 3036$. Therefore, we use Algorithm 5 (with default parameters) to get the rNGWF $\left\{\overline{\psi}_\gamma\right\}_{\gamma\in\bar{\Gamma}}$, which only has $|\bar{\Gamma}| = 2986$ number of frame vectors in total. Considering the rNGWF and the SGWT frame have nearly the same number of frame vectors, it is fair to use Algorithm 4 to compare their graph signal approximation performance. Figure 10.7b shows the approximation experiment results on the hand-written digit "3" graph signal, where the approximation performance is measured by the relative ℓ^2 approximation error with respect to the number of coefficients retained (denote as NCR for simplicity). Specifically, the NCR in the experiments starts from 0 up to $3N = 1518$. From Figure 10.7b, we observe that the rNGWF clearly outperformed the SGWT frame, and especially with a big lead up to $NCR \approx 500$.

In order to examine the performance difference between the SGWT frame and the rNGWF, we display their 100 most significant frame vectors in Figure 10.8 and Figure 10.9, respectively. We note that the top SGWT frame vectors are generally coarse scales, and they try to capture the main body of the digit "3". Yet, there is one fine scale SGWT wavelet vector marked by the red box in Figure 10.8, which captures the details of the digit in the bottom left corner. On the other hand, most of wavelets among the top rNGWF vectors also try to capture the main body of the digit "3", while some of them exhibit horizontal/vertical anisotropic patterns (marked by red boxes in Figure 10.9) and try to capture the edges of the digit "3". These anisotropic wavelets make the rNGWF perform better than the SGWT frame in the digit "3" signal approximation experiment. We also note that there are five global graph Laplacian eigenvectors (marked by orange boxes in Figure 10.9) appeared among the top 100 rNGWF vectors. This phenomenon is due to the two changes when generating the rNGWF as explained in Section 10.3.

10.5.4. Graph Signal Approximation via Frames on a Synthetic Dendritic Tree. Next, we consider the *unweighted* synthetic dendritic tree graph (see Figure 5.8a), whose eigenvectors are carefully studied in Section 5.3.2. It has $N = 100$ nodes and $M = 99$ edges. Figure 10.10a shows a synthetic graph signal on this tree, which is composed of four delta impulses at four junctions, and four sinusoids with different oscillatory frequencies on four branches, and zeros elsewhere.

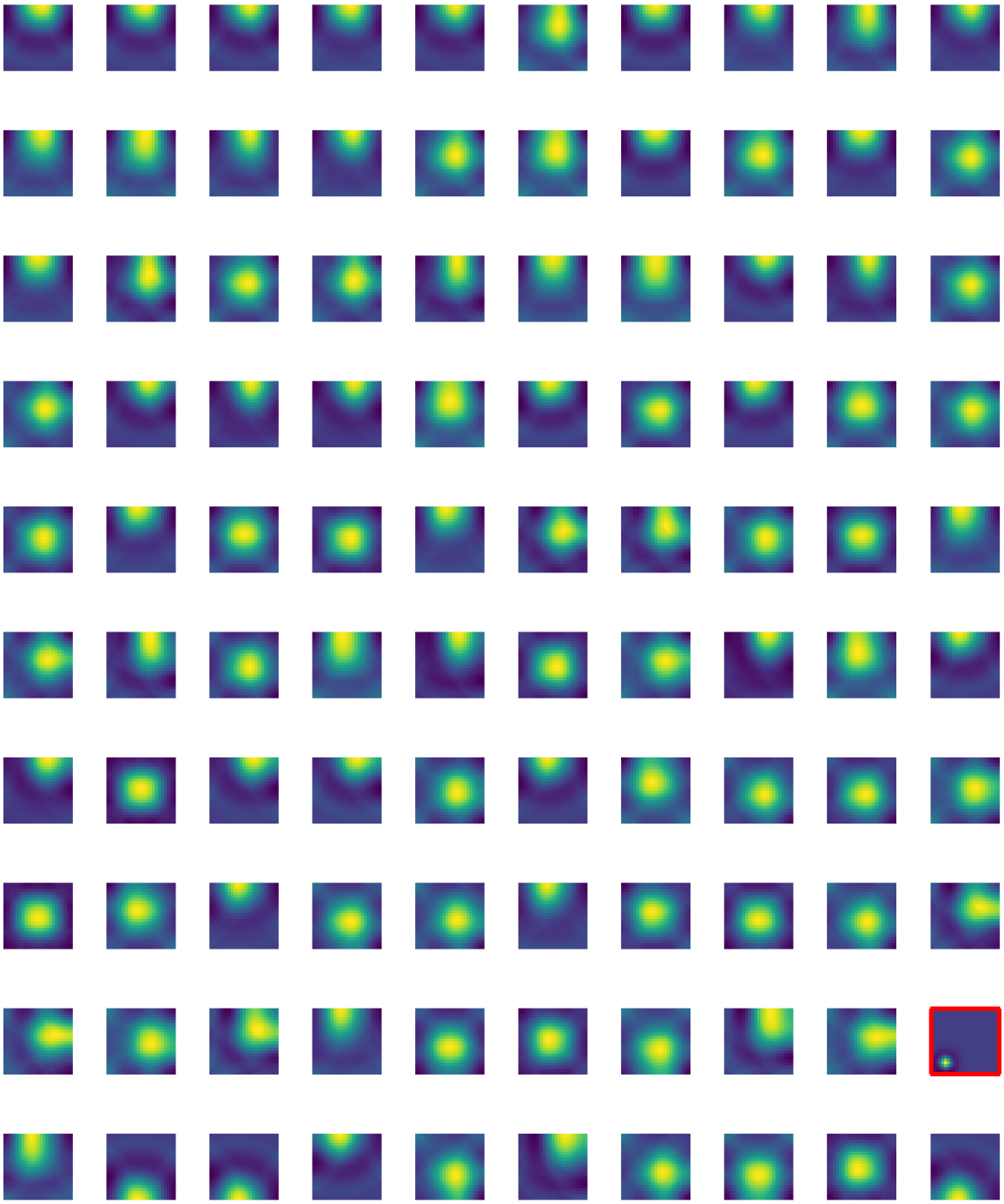


FIGURE 10.8. Top 100 SGWT frame vectors used in the hand-written digit “3” approximation.

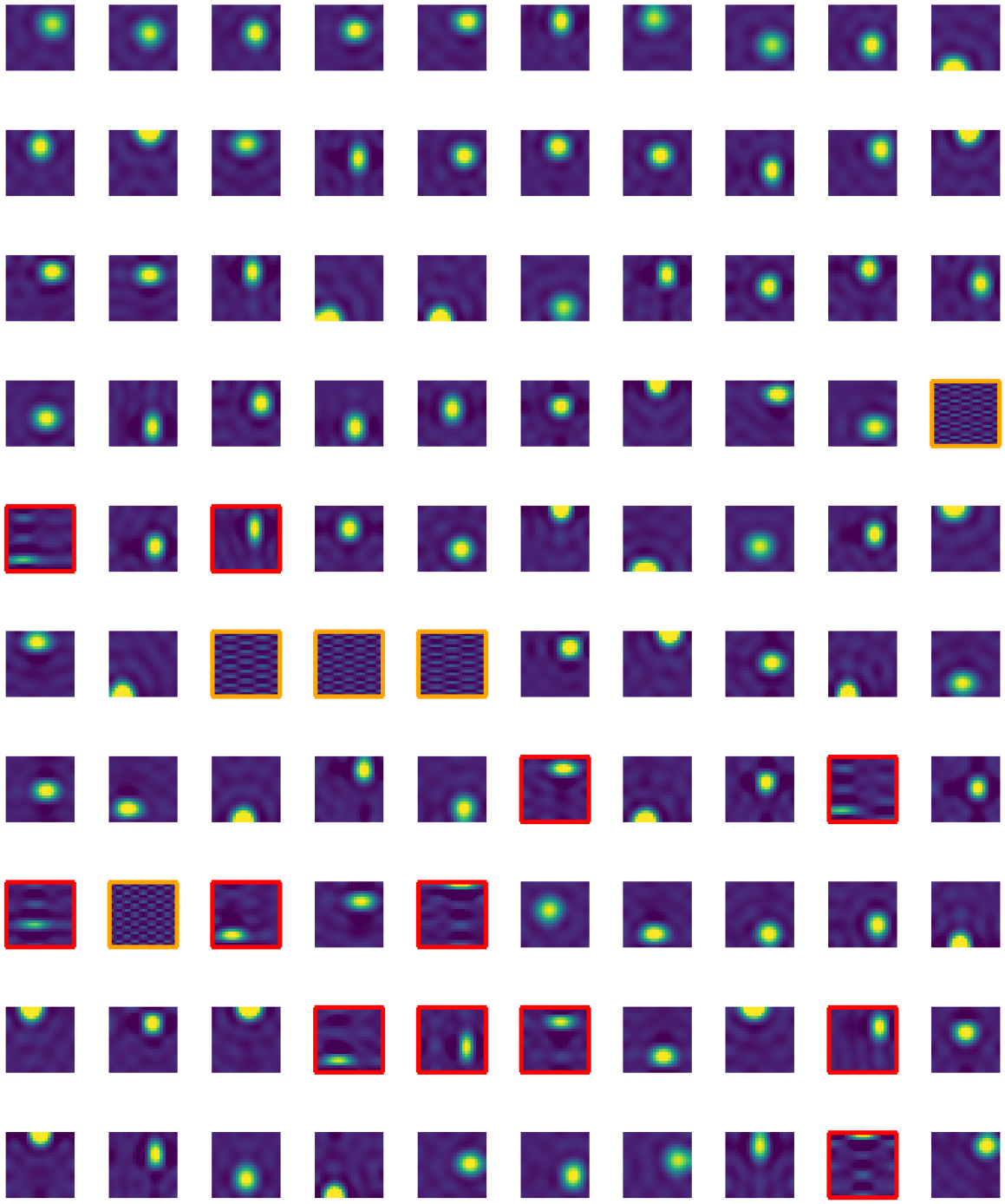
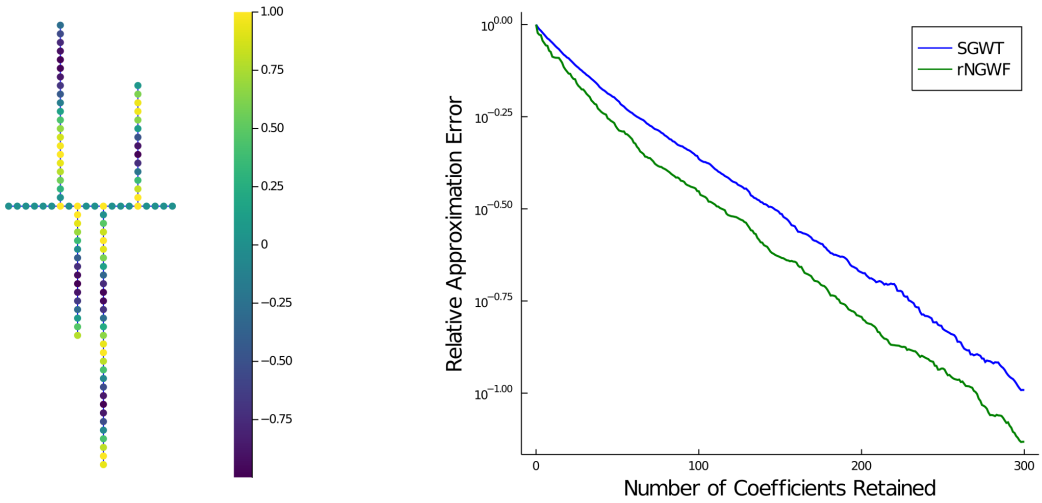


FIGURE 10.9. Top 100 rNGWF vectors used in the hand-written digit “3” approximation.



(a) A synthetic tree signal.

(b) Approximation performance of two frames.

FIGURE 10.10. A synthetic signal on the tree (a); the results of our approximation experiments (b).

We use $d_{\mathrm{ROT}}^{(1)} \circ \mathrm{pmf}^{(1)}$ ($\alpha = 1$) and $\sigma = 0.1 \cdot d_{\max}$ to construct the set of natural spectral graph filters $\{\mathcal{F}_l\}_{l=0:N-1}$. They can be viewed as the Gaussian filters on the 3D MDS embedding space of the eigenvectors as shown in Figure 5.12a. We then get the rNGWF $\left\{\overline{\psi}_\gamma\right\}_{\gamma\in\overline{\Gamma}}$ via Algorithm 5 with the default threshold 0.15. The obtained rNGWF has $|\overline{\Gamma}| = 632$ frame vectors, which is much less than the number of frame vectors contained in the NGWF ($|\Gamma| = N^2 = 10000$). On the other hand, we use the same set of eigenvalue-dependent smooth functions in Figure 10.4 to construct the SGWT frame $\left\{\psi_{j,x}^{\mathrm{SGWT}}\right\}_{j=0:J;x=1:N}$ (with $J = 5$) on the tree, which contains $(J+1)\cdot N = 600$ frame vectors in total. Considering the rNGWF and the SGWT frame have about the same number of frame vectors, it is reasonable to compare their graph signal approximation performance via Algorithm 4. Figure 10.10b shows the approximation results of the synthetic tree signal. The NCR in the experiments goes from 0 to $3N = 300$. We observe that the rNGWF plainly beat the SGWT frame. Besides that, we also observe that the approximation curves in this figure are jagged compared to the ones in Chapter 9 (e.g., see Figure 9.2c). This phenomenon is because these curves are generated by frames not orthonormal bases. Since the frame vectors are not always orthogonal to each other, we cannot guarantee that the relerror in Algorithm 4 is monotonically decreasing.

In order to examine the performance difference between the SGWT frame and the rNGWF, we display their top 21 frame vectors in Figure 10.11 and Figure 10.12, respectively. The top SGWT frame vectors are

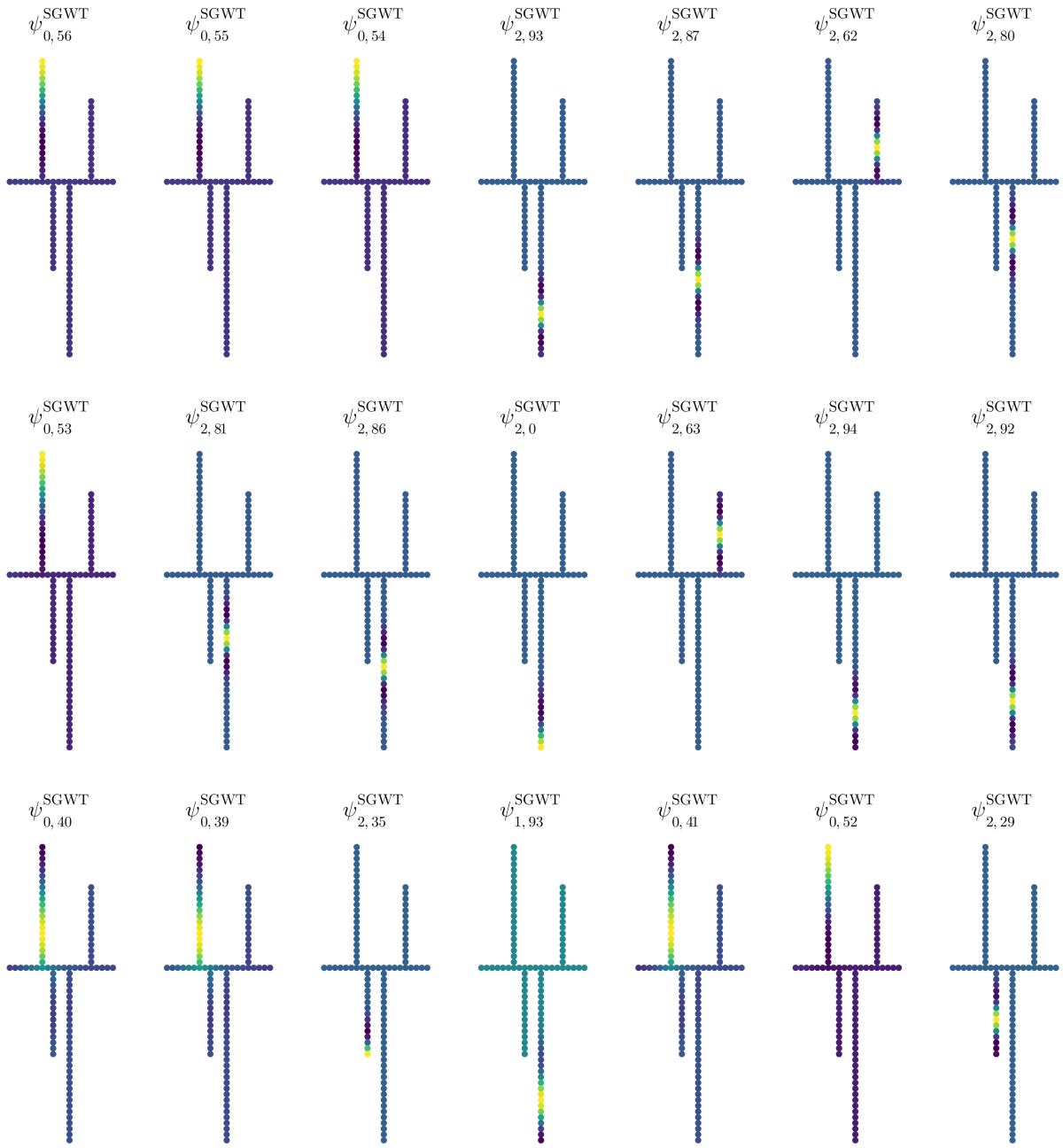


FIGURE 10.11. Top 21 SGWT frame vectors used in the synthetic tree signal approximation.

coarse scales ($j = 0:2$), and try to characterize the oscillations of the synthetic signal on the branches. On the other hand, the top rNGWF vectors exhibit more rich structures that try to capture both the oscillations on branches and delta impulses at junctions. These behaviors are inherited from their centered

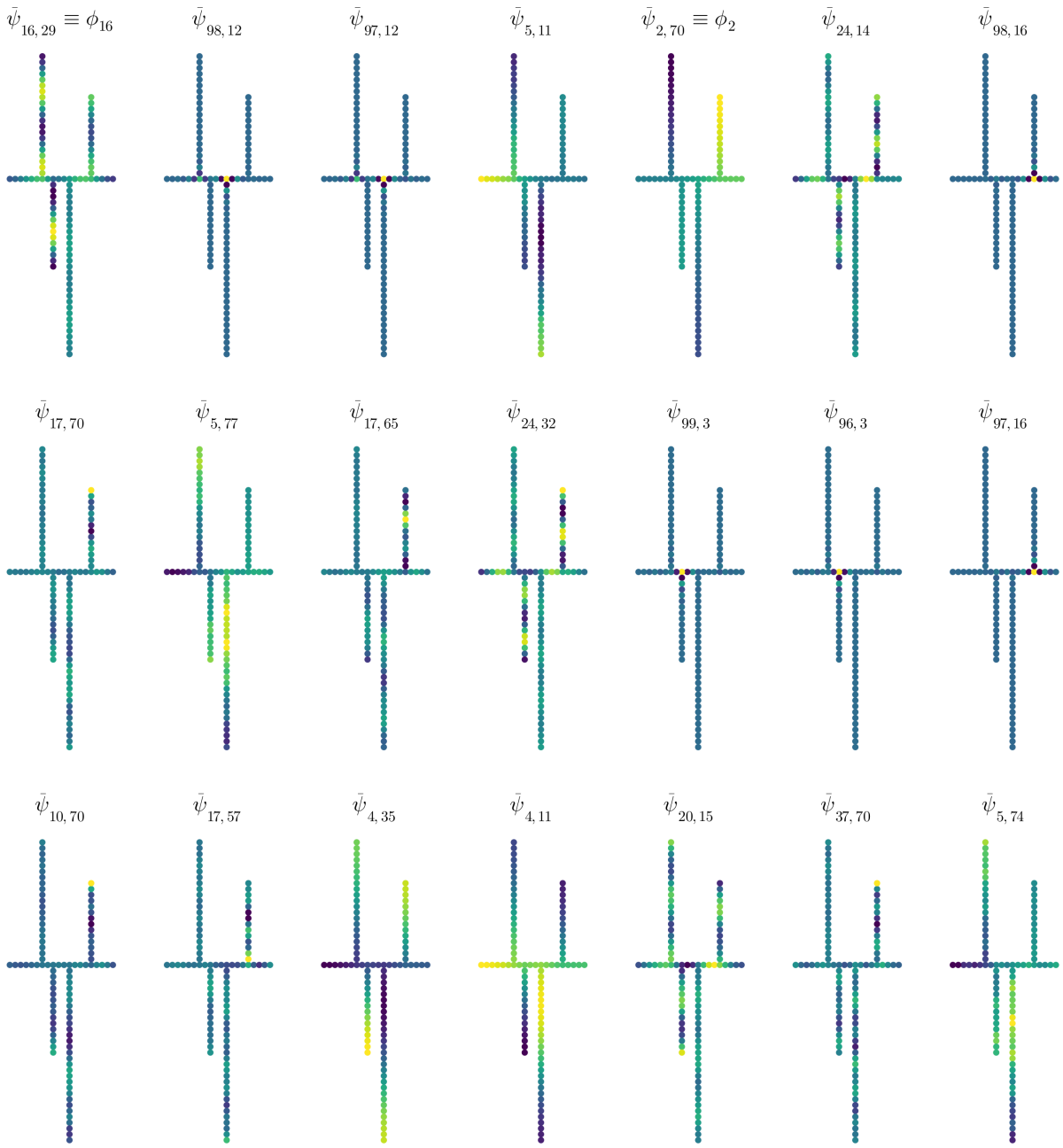


FIGURE 10.12. Top 21 rNGWF vectors used in the synthetic tree signal approximation.

eigenvectors with the ROT distance. Note that there are two eigenvectors selected among the rNGWF vectors, i.e., ϕ_2 and ϕ_{16}. Figure 10.13 shows the centered eigenvectors of the top rNGWF vectors for reference. The ROT distance, as discussed in Section 5.4, is good at detecting the energy localization

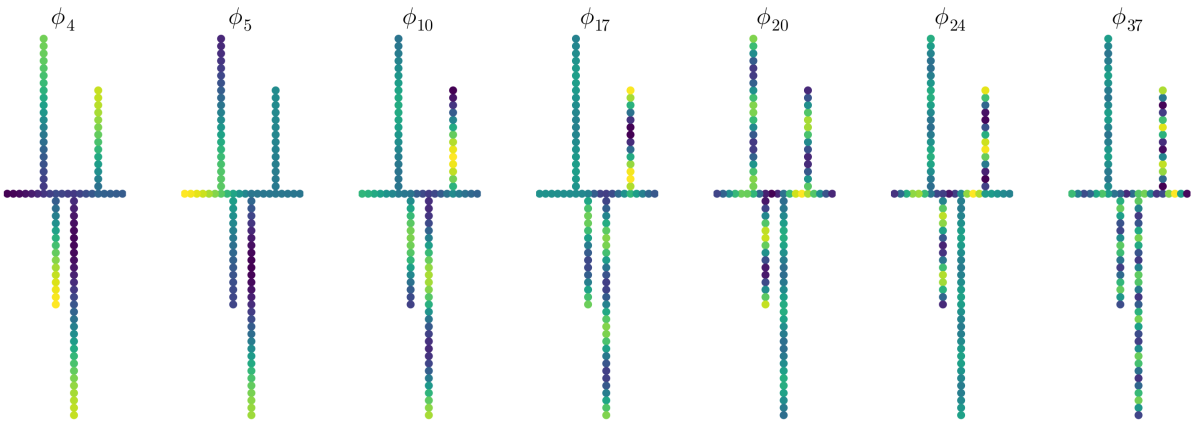


FIGURE 10.13. The centered eigenvectors of the top 21 rNGWF vectors in Figure 10.12. See Figure 5.11 for the eigenvectors localized at junctions, i.e., ϕ_l ($l = 96:99$).

patterns of the eigenvectors. As one can see from Figure 10.12, Figure 10.13 and Figure 5.11, the rNGWF vectors behave similarly to their centered eigenvectors in terms of energy distribution on the tree, but they are more "localized" in the primal domain G. These rich structures of rNGWF vectors make it more efficient to approximate this synthetic tree signal comparing to the SGWT frame.

10.6. Summary

We first introduced the natural spectral graph filters by utilizing the eigenvector distance d, and we then built the NGWF by following the framework of building the SGWT frames in Eq. (2.4). To reduce its redundancy, we presented an algorithm to get its reduced version, i.e., the rNGWF. We also demonstrated how to use them for graph signal analysis and synthesis. We displayed the 2D expansion coefficients of the NGWF transform of an artificial graph signal on P_{128}, i.e., the NGWF spectrogram. We also showcased the NGWF vectors built by d_{DAG} on $P_{23} \times P_{22}$ and compared them with the SGWT frame vectors, and we observed the anisotropic wavelets among the NGWF vectors. At last, we showed the power of the rNGWF by comparing its performance with the SGWT frame in two graph signal approximation experiments: 1) a hand-written digit "3" signal on $P_{23} \times P_{22}$ and 2) a synthetic signal on the synthetic dendritic tree.

CHAPTER 11

Conclusion

In this dissertation, we have introduced and reviewed several non-trivial metrics of graph Laplacian eigenvectors to characterize the eigenvectors behavioral patterns on an input graph G, and used these metrics to naturally organize the dual domain of G by building a complete dual graph $G^\star$. We used the classical MDS technique to visualize those eigenvector organizations in $\mathbb{R}^2$ or $\mathbb{R}^3$, and presented the experimental results in Section 5.3.

We then proposed three methods to construct graph wavelet packet dictionaries that fully utilize the dual graph: the VM-NGWP, the PC-NGWP and the LP-NGWP dictionaries. Then, using two different graph signals on each of the two different graphs, we compared their performance in approximating a given graph signal against the multiscale graph basis dictionaries previously designed by Prof. Saito's group, such as the HGLET and eGHWT dictionaries; and the LP-HGLET dictionary developed in Section 8.3. Our proposed dictionaries outperformed the others on locally smooth graph signals, and performed reasonably well for a graph signal sampled on an image containing oriented anisotropic texture patterns. On the other hand, our new dictionaries were beaten by the eGHWT on the non-smooth and localized graph signal. The potential reasons for such a behavior are the facts that 1) the VM-NGWP and the PC-NGWP dictionaries are a direct generalization of the "Shannon" wavelet packet dictionaries, i.e., their spectral (i.e., dual) domain support is localized and well-controlled while the primal domain support is not compact; and 2) although the LP-NGWP dictionary is a generalization of the "Meyer" wavelet packet dictionary whose basis vectors are more localized on primal domain, it does not have the appropriate control of how much the basis vectors are localized on primal domain at different scales.

We also note that the VM-NGWP and the PC-NGWP dictionaries performed generally better than the LP-NGWP dictionary for the graph signals we have examined in Chapter 9. This could be due to the inappropriate choice of ϵ of the LP-NGWP dictionary, considering it actually reduces to the VM-NGWP dictionary with $\epsilon = 0$. Thus, one important question left in the LP-NGWP dictionary is: how do we choose

appropriate sizes of the action regions at different levels in order to get the best multiscale LP-NGWP basis vectors in graph signal approximations? Also, the VM-NGWP dictionary performed generally better than the PC-NGWP dictionary. One of the possible reasons is the use of the explicit localization procedure, i.e., the varimax rotation, in the former; the latter allows one to try to "pinpoint" a particular primal node where the basis vector should concentrate, but the MGSLp procedure unfortunately shuffles and slightly delocalizes the basis vectors after orthogonalization. On the other hand, the differences in their computational costs are not significant: the VM-NGWP and PC-NGWP dictionaries cost $O(N^3)$ operations with a different constant, while the LP-NGWP dictionary costs $O\left(N^3 \log_2 N\right)$ operations. Hence, it is important to investigate how to reduce the computational complexity in the three cases. One such possibility is to use only the first N_0 graph Laplacian eigenvectors with $N_0 \ll N$. Clearly, one cannot represent a given graph signal precisely with N_0 eigenvectors, but this scenario may be acceptable for certain applications including graph signal clustering, classification, and regression. Of course, it is of our interest to investigate whether we can come up with faster versions of the varimax rotation algorithm and the MGSLp algorithm, which forms one of the future research projects.

Next, we proposed the natural graph wavelet frame (NGWF) and its reduced version (rNGWF) by utilizing the dual graph and the framework of the Spectral Graph Wavelet Transform (SGWT). Then, we compared the performance of rNGWF and the SGWT in approximating a hand-written digit "3" signal on a lattice graph and a synthetic signal on a synthetic dendritic tree. The rNGWF clearly beats the SGWT, and its frame vectors present more rich structures, which are inherited from the global graph Laplacian eigenvectors via the natural graph spectral filters with the non-trivial eigenvector metrics. On the other hand, the computational cost of building the NGWF and the rNGWF are both $O(N^4)$ operations. So, reducing the computational complexity is also an important task for both cases. Note that, although one can use the NGWF or the rNGWF to approximate graph signals, they are generally not as efficient as the NGWP dictionaries, because the frames tend to need more vectors with a higher computational cost.

Although it is not explicitly included in this dissertation, a major contribution of this work is the accompanying software package [**50**][1]. We have developed a framework to build the complete dual graph by computing the non-trivial distances between the eigenvectors, and consequently construct the NGWP dictionaries and NGWFs for graph signal processing. Along with the software we provide scripts

[1] which is now merged into [**32**].

for reproducing figures from our articles [**9**,**51**] and this dissertation. It is our hope that interested readers will download the software themselves, recreate our figures, and possibly conduct their own experiments with it.

Finally, we would like to emphasize that the natural dual domain $G^\star$ can be used in applications beyond dictionary construction. Such applications include: graph cuts and spectral clustering [**61**, **78**] to move beyond noted limitations to using the first few eigenvectors [**59**]; graph visualization and embeddings [**1**] to represent embeddings with lower distortion [**41**] as was done in [**43**]; and anomaly detection through spectral methods [**20**, **56**] by going beyond the first few eigenvectors [**7**, **8**]. It is interesting to investigate going beyond the initial set of eigenvectors with small eigenvalues in a method informed by $G^\star$, and the effect of $G^\star$ on such methods.

APPENDIX A

Supporting Proofs

A.1. Proof of Lemma 4.5.1

PROOF. Let us prove $K_{\mathrm{TSD}}(\cdot, T)$ is a norm on $\mathbb{R}_0^N$ by verifying that it satisfies the following four properties. Denote $\tilde{Q} = \tilde{Q}(G)$ as the unweighted incidence matrix of G.

(1) (Non-negativity) $K_{\mathrm{TSD}}(\boldsymbol{f}_0, T) \geq 0$, because of the non-negativity of the weighted ℓ^1-norm inside the integral.

(2) (Identity of discernible) First, if $\boldsymbol{f} = \mathbf{0} \in \mathbb{R}_0^N$, then it is easy to see that $K_{\mathrm{TSD}}(\mathbf{0}, T) = 0$.

On the other hand, if $K_{\mathrm{TSD}}(\boldsymbol{f}, T) = 0$,

$$K_{\mathrm{TSD}}(\boldsymbol{f}, T) = \int_0^T \left\| \tilde{Q}^{\mathsf{T}} \sum_{l=0}^{N-1} \langle \boldsymbol{f}, \tilde{\boldsymbol{\phi}}_l \rangle e^{-\tilde{\lambda}_l t} \tilde{\boldsymbol{\phi}}_l \right\|_{1,\mathrm{w}} \mathrm{d}t = 0$$

$$\begin{aligned}
\Longrightarrow \quad & \left\| \tilde{Q}^{\mathsf{T}} \sum_{l=0}^{N-1} \langle \boldsymbol{f}, \tilde{\boldsymbol{\phi}}_l \rangle e^{-\tilde{\lambda}_l t} \tilde{\boldsymbol{\phi}}_l \right\|_{1,\mathrm{w}} = 0 \quad \text{for any } t \in [0, T) \\
\Longrightarrow \quad & \tilde{Q}^{\mathsf{T}} \sum_{l=0}^{N-1} \langle \boldsymbol{f}, \tilde{\boldsymbol{\phi}}_l \rangle e^{-\tilde{\lambda}_l t} \tilde{\boldsymbol{\phi}}_l = \mathbf{0} \quad \text{for any } t \in [0, T) \\
\Longrightarrow \quad & \tilde{Q}^{\mathsf{T}} \sum_{l=1}^{N-1} \langle \boldsymbol{f}, \tilde{\boldsymbol{\phi}}_l \rangle e^{-\tilde{\lambda}_l t} \tilde{\boldsymbol{\phi}}_l = \mathbf{0} \quad \text{thanks to } \tilde{Q}^{\mathsf{T}} \tilde{\boldsymbol{\phi}}_0 = \mathbf{0} \in \mathbb{R}^M \\
\Longrightarrow \quad & \tilde{Q}^{\mathsf{T}} \begin{bmatrix} \tilde{\boldsymbol{\phi}}_1 & \tilde{\boldsymbol{\phi}}_2 & \cdots & \tilde{\boldsymbol{\phi}}_{N-1} \end{bmatrix} \cdot \begin{bmatrix} \langle \boldsymbol{f}, \tilde{\boldsymbol{\phi}}_1 \rangle e^{-\tilde{\lambda}_1 t} \\ \langle \boldsymbol{f}, \tilde{\boldsymbol{\phi}}_2 \rangle e^{-\tilde{\lambda}_2 t} \\ \vdots \\ \langle \boldsymbol{f}, \tilde{\boldsymbol{\phi}}_{N-1} \rangle e^{-\tilde{\lambda}_{N-1} t} \end{bmatrix} = \mathbf{0}.
\end{aligned}$$

Since the G is connected (implies $M \geq N-1$), it is easy to show that $\mathrm{rank}(\tilde{Q}) = N-1$ [**24**]. Denoting $A := \tilde{Q}^{\mathsf{T}}[\tilde{\boldsymbol{\phi}}_1, \tilde{\boldsymbol{\phi}}_2, \cdots, \tilde{\boldsymbol{\phi}}_{N-1}] \in \mathbb{R}^{M \times (N-1)}$, A is full column rank, i.e., $\mathrm{rank}(A) = N-1$. Therefore,

it implies $\langle \boldsymbol{f}, \tilde{\phi}_l \rangle e^{-\tilde{\lambda}_l t} = 0$, i.e., $\langle \boldsymbol{f}, \tilde{\phi}_l \rangle = 0$ for $l = 1 : N-1$. Since $\boldsymbol{f} \in \mathbb{R}_0^N$, we also have $\langle \boldsymbol{f}, \tilde{\phi}_0 \rangle = 0$. Hence, $\boldsymbol{f} = \sum_{l=0}^{N-1} \langle \boldsymbol{f}, \tilde{\phi}_l \rangle \tilde{\phi}_l = \mathbf{0} \in \mathbb{R}_0^N$.

(3) (Absolutely homogeneous) For $\alpha \in \mathbb{R}$ and $\boldsymbol{f} \in \mathbb{R}_0^N$, $K_{\mathrm{TSD}}(\alpha \boldsymbol{f}, T) = |\alpha| K_{\mathrm{TSD}}(\boldsymbol{f}, T)$.

(4) (Triangle inequality) For any $\boldsymbol{f}, \boldsymbol{g} \in \mathbb{R}_0^N$,

$$\begin{aligned} K_{\mathrm{TSD}}(\boldsymbol{f}+\boldsymbol{g}, T) &= \int_0^T \left\| \tilde{Q}^{\mathsf{T}} \sum_{l=0}^{N-1} \langle \boldsymbol{f}+\boldsymbol{g}, \tilde{\phi}_l \rangle e^{-\tilde{\lambda}_l t} \tilde{\phi}_l \right\|_{1,\mathrm{w}} \mathrm{d}t \\ &\le \int_0^T \left\| \tilde{Q}^{\mathsf{T}} \sum_{l=0}^{N-1} \langle \boldsymbol{f}, \tilde{\phi}_l \rangle e^{-\tilde{\lambda}_l t} \tilde{\phi}_l \right\|_{1,\mathrm{w}} + \left\| \tilde{Q}^{\mathsf{T}} \sum_{l=0}^{N-1} \langle \boldsymbol{g}, \tilde{\phi}_l \rangle e^{-\tilde{\lambda}_l t} \tilde{\phi}_l \right\|_{1,\mathrm{w}} \mathrm{d}t \\ &= K_{\mathrm{TSD}}(\boldsymbol{f}, T) + K_{\mathrm{TSD}}(\boldsymbol{g}, T). \end{aligned}$$

Therefore, $K_{\mathrm{TSD}}(\cdot, T)$ is a norm on $\mathbb{R}_0^N$ and $(\mathbb{R}_0^N, K_{\mathrm{TSD}}(\cdot, T))$ is a normed vector space.

□

A.2. Proof of Theorem 4.5.2

PROOF.

$$\begin{aligned} K_{\mathrm{TSD}}\left(f_0, T\right) &= \int_0^T \int_{\mathscr{M}} \left| \nabla_x f(x,t) \right| \mathrm{d}x \mathrm{d}t \\ &= \int_0^T \int_{\mathscr{M}} \left| \nabla_x \sum_{l=0}^{\infty} \langle \phi_l, f_0 \rangle e^{-\lambda_l t} \phi_l(x) \right| \mathrm{d}x \mathrm{d}t \\ \left[\phi_0 \equiv \text{ a constant}\right] &= \int_0^T \int_{\mathscr{M}} \left| \nabla_x \sum_{l=1}^{\infty} \langle \phi_l, f_0 \rangle e^{-\lambda_l t} \phi_l(x) \right| \mathrm{d}x \mathrm{d}t \\ \left[\text{by triangle inequality}\right] &\le \int_0^T \int_{\mathscr{M}} \sum_{l=1}^{\infty} \left| \langle \phi_l, f_0 \rangle e^{-\lambda_l t} \nabla \phi_l(x) \right| \mathrm{d}x \mathrm{d}t \\ \left[\text{by Fubini theorem}\right] &= \sum_{l=1}^{\infty} \int_0^T e^{-\lambda_l t} \mathrm{d}t \cdot \int_{\mathscr{M}} \left| \langle \phi_l, f_0 \rangle \nabla \phi_l(x) \right| \mathrm{d}x \\ \left[\text{as } T \to \infty\right] &\le \sum_{l=1}^{\infty} \frac{1}{\lambda_l} |\hat{f}_0(l)| \cdot \int_{\mathscr{M}} \left| \nabla \phi_l(x) \right| \mathrm{d}x \\ \left[\text{by Cauchy–Schwarz inequality}\right] &\le \sum_{l=1}^{\infty} \frac{1}{\lambda_l} |\hat{f}_0(l)| \cdot \|\nabla \phi_l\|_2 \cdot \sqrt{\mathrm{Vol}(\mathscr{M})} \\ \left[\text{since } \|\nabla \phi_l\|_2 = \sqrt{\lambda_l} \text{ by Green's formula}\right] &= \sum_{l=1}^{\infty} \frac{1}{\sqrt{\lambda_l}} |\hat{f}_0(l)| \cdot \sqrt{\mathrm{Vol}(\mathscr{M})}. \end{aligned}$$

□

A.3. Proof of Theorem 4.6.4

The heat diffusion PDE with the Neumann boundary condition on $[0,2\pi]$:

$$\begin{cases} \frac{\partial}{\partial t} f(x,t) - \frac{\partial^2}{\partial x^2} f(x,t) = 0, \quad x \in [0,2\pi] \\ f(x,0) = f_0(x) = q(x) - p(x) \in L_0^2([0,2\pi]) \\ \frac{\partial}{\partial x} f(0,t) = \frac{\partial}{\partial x} f(2\pi,t) = 0, \quad t \geq 0 \end{cases}$$

and its general solution using Laplacian eigenfunctions ϕ_l:

$$f(x,t) = \sum_{l=0}^{\infty} \langle \phi_l, f_0 \rangle e^{-\lambda_l t} \phi_l(x) \qquad \text{in which } \phi_l(x) = \begin{cases} \frac{1}{\sqrt{2\pi}}, & l = 0, \\ \frac{1}{\sqrt{\pi}} \cos\left(\frac{l}{2}x\right), & l \geq 1. \end{cases} \text{ and } \lambda_l = \frac{1}{4} l^2.$$

Now, the functional K_{TSD} in Eq. (4.15) becomes

$$K_{\text{TSD}}(f_0, T) = \int_0^T \int_0^{2\pi} \left| \frac{\partial}{\partial x} f(x,t) \right| \mathrm{d}x \mathrm{d}t.$$

PROOF.

$$\begin{aligned}
K_{\text{TSD}}(q-p, \infty) &= \int_0^{\infty} \int_0^{2\pi} \left| \sum_{l=0}^{\infty} \langle \phi_l, q-p \rangle e^{-\lambda_l t} \phi_l'(x) \right| \mathrm{d}x \mathrm{d}t \\
\left[\text{since } \phi_0' \equiv 0\right] &= \int_0^{\infty} \int_0^{2\pi} \left| \sum_{l=1}^{\infty} \langle \phi_l, q-p \rangle e^{-\lambda_l t} \phi_l'(x) \right| \mathrm{d}x \mathrm{d}t \\
\left[\text{by Fubini theorem}\right] &= \int_0^{2\pi} \int_0^{\infty} \left| \sum_{l=1}^{\infty} \langle \phi_l, q-p \rangle e^{-\lambda_l t} \phi_l'(x) \right| \mathrm{d}t \mathrm{d}x \\
\left[\text{by triangle inequality}\right] &\geq \int_0^{2\pi} \left| \int_0^{\infty} \sum_{l=1}^{\infty} \langle \phi_l, q-p \rangle e^{-\lambda_l t} \phi_l'(x) \mathrm{d}t \right| \mathrm{d}x \\
&= \int_0^{2\pi} \left| \sum_{l=1}^{\infty} \langle \phi_l, q-p \rangle \frac{1}{\lambda_l} \phi_l'(x) \right| \mathrm{d}x \\
\left[\text{since } -\phi_l'' = \lambda_l \phi_l\right] &= \int_0^{2\pi} \left| \sum_{l=1}^{\infty} \langle \phi_l, q-p \rangle \int_0^x \phi_l(s) \mathrm{d}s \right| \mathrm{d}x \\
&= \int_0^{2\pi} \left| \int_0^x (q(s) - p(s)) \mathrm{d}s \right| \mathrm{d}x = W_1(p,q).
\end{aligned}$$

For the last equation, we used the explicit formula of W_1 in $\mathbb{R}$ as shown in [**49**]. □

A.4. Proof of Theorem 7.1.1

PROOF. The graph Laplacian eigenvectors $\phi_l = \phi_{l;N}$ as defined in Eq. (2.2) are

$$\phi_l(x) = \begin{cases} \frac{1}{\sqrt{N}}, & l = 0, \\ \sqrt{\frac{2}{N}} \cos\left(\frac{\pi l}{N}\left(x - \frac{1}{2}\right)\right), & l = 1 : N-1. \end{cases}$$

Our goal is to solve Eq. (7.2) based on the greedy algorithm, i.e.,

1) compute score $(\boldsymbol{\delta}_x) = \alpha_{\mathrm{PC}}(\{\boldsymbol{\delta}_x\}, V_0^\star) - \alpha_{\mathrm{PC}}(\{\boldsymbol{\delta}_x\}, V_1^\star) = 2\alpha_{\mathrm{PC}}(\{\boldsymbol{\delta}_x\}, V_0^\star) - 1$, in which the last equality is due to the fact $\alpha_{\mathrm{PC}}(\{\boldsymbol{\delta}_x\}, V_0^\star) + \alpha_{\mathrm{PC}}(\{\boldsymbol{\delta}_x\}, V_1^\star) = \alpha_{\mathrm{PC}}(\{\boldsymbol{\delta}_x\}, V^\star) = \sum_{l=0}^{N-1} \phi_l(x)^2 = 1$, for $x = 1 : N$.
2) select $N_0 (= \lfloor \frac{N+1}{2} \rfloor)$ $\boldsymbol{\delta}_x$'s that give the largest N_0 values of score$(\cdot)$.

Also, the following trigonometric identity is repeatedly used in our proofs.

$$\sum_{l=1}^{m-1} \cos^2(l\pi\theta) = \sum_{l=1}^{m-1} \frac{1 + \cos(2l\pi\theta)}{2} = \frac{1}{4}\left(\frac{\sin((2m-1)\pi\theta)}{\sin(\pi\theta)} + 2m - 3\right).$$

Now, let us prove the theorem under two cases: 1) N is even and 2) N is odd.

Case 1: $\underline{N = 2m}$. In this case, $N_0 = \lfloor \frac{N+1}{2} \rfloor = m$.

$$\begin{aligned} \alpha_{\mathrm{PC}}(\{\boldsymbol{\delta}_x\}, V_0^\star) &= \frac{1}{N} + \frac{2}{N} \sum_{l=1}^{m-1} \cos^2\left(\frac{\pi l}{N}\left(x - \frac{1}{2}\right)\right) \\ \left[\text{let } \theta := \frac{x - \frac{1}{2}}{N}\right] &= \frac{1}{N} + \frac{2}{N} \sum_{l=1}^{m-1} \cos^2(l\pi\theta) \\ &= \frac{1}{N} + \frac{1}{2N}\left(\frac{\sin(N-1)\pi\theta}{\sin\pi\theta} + N - 3\right) \\ &= \frac{1}{2} + \frac{1}{2N}\left(\frac{\sin(N-1)\pi\theta}{\sin\pi\theta} - 1\right). \end{aligned}$$

Therefore, for $x = 1:N$,

$$\begin{aligned}
\text{score}(\boldsymbol{\delta}_x) &= \frac{1}{N}\left(\frac{\sin(N-1)\pi\theta}{\sin\pi\theta} - 1\right) \\
&= \frac{1}{N}\left(\frac{\sin(N\pi\theta)\cos\pi\theta - \cos(N\pi\theta)\sin\pi\theta}{\sin\pi\theta} - 1\right) \\
&= \frac{1}{N}\left(\sin(N\pi\theta)\cot\pi\theta - 1\right) \\
&= \frac{1}{N}\left(\sin\left(\pi\left(x-\frac{1}{2}\right)\right)\cot\left(\frac{\pi}{N}\left(x-\frac{1}{2}\right)\right) - 1\right) \\
&= \frac{1}{N}\left(-\cos(\pi x)\cot\left(\frac{\pi}{N}\left(x-\frac{1}{2}\right)\right) - 1\right) \\
&= \frac{1}{N}\left((-1)^{x-1}\cot\left(\frac{\pi}{N}\left(x-\frac{1}{2}\right)\right) - 1\right).
\end{aligned}$$

Note that: 1) score$(\cdot)$ is symmetric based on the above formula, i.e., score$(\boldsymbol{\delta}_x)$ = score$(\boldsymbol{\delta}_{N+1-x})$, so we only need to analyze score$(\boldsymbol{\delta}_x)$, for $x = 1:\lfloor\frac{N+1}{2}\rfloor = 1:m$; 2) $\cot(\cdot)$ is a positive and decreasing function over the interval $(0,\pi/2]$, so the series $\left\{\text{score}(\boldsymbol{\delta}_x) + \frac{1}{N}\right\}_{x=1:m}$ is an alternating series and its terms monotonically decrease in magnitude; and 3) the indices of the largest m terms in $\left\{\text{score}(\boldsymbol{\delta}_x) + \frac{1}{N}\right\}_{x=1:N}$ are the same as the indices of the largest m terms in $\{\text{score}(\boldsymbol{\delta}_x)\}_{x=1:N}$.

If m is even, i.e., $\underline{m = 2n}$, then the m $\boldsymbol{\delta}_x$'s that have the largest m values of score$(\cdot)$ are the combination of the n $\boldsymbol{\delta}_x$'s that have the largest n terms in the series $\left\{\text{score}(\boldsymbol{\delta}_x) + \frac{1}{N}\right\}_{x=1:m}$ and the n $\boldsymbol{\delta}_x$'s that have the largest n terms in the symmetric series $\left\{\text{score}(\boldsymbol{\delta}_x) + \frac{1}{N}\right\}_{x=m+1:N}$, i.e.,

$$V_0 = \{\boldsymbol{\delta}_{2x-1}\}_{x=1:n} \cup \{\boldsymbol{\delta}_{2x}\}_{x=n+1:m}.$$

If m is odd, i.e., $\underline{m = 2n-1}$, then the $m-1$ $\boldsymbol{\delta}_x$'s that have the largest $m-1$ values of score$(\cdot)$ are the combination of the $n-1$ $\boldsymbol{\delta}_x$'s that have the largest $n-1$ terms in the series $\left\{\text{score}(\boldsymbol{\delta}_x) + \frac{1}{N}\right\}_{x=1:m}$ and the $n-1$ $\boldsymbol{\delta}_x$'s that have the largest $n-1$ terms in the (symmetric) series $\left\{\text{score}(\boldsymbol{\delta}_x) + \frac{1}{N}\right\}_{x=m+1:N}$. We also need a last term, i.e., the m-th largest term in $\left\{\text{score}(\boldsymbol{\delta}_x) + \frac{1}{N}\right\}_{x=1:N}$. However, there are two m-th largest terms in the series, i.e.,

$$\text{score}(\boldsymbol{\delta}_m) + \frac{1}{N} = \text{score}(\boldsymbol{\delta}_{m+1}) + \frac{1}{N}.$$

We have to choose one of them and put it into V_0, so the two possibilities of the m $\boldsymbol{\delta}_x$'s that have the largest m values of score$(\cdot)$ are

$$V_0 = \{\boldsymbol{\delta}_{2x-1}\}_{x=1:n} \cup \{\boldsymbol{\delta}_{2x}\}_{x=n+1:m} \quad \text{and} \quad V_0 = \{\boldsymbol{\delta}_{2x-1}\}_{x=1:n-1} \cup \{\boldsymbol{\delta}_{2x}\}_{x=n:m}.$$

At last, we set $V_1 = V \setminus V_0$ in each case.

Case 2: $\underline{N = 2m-1}$. In this case, $N_0 = \lfloor \frac{N+1}{2} \rfloor = m$.

$$\begin{aligned}
\alpha_{\mathrm{PC}}(\{\boldsymbol{\delta}_x\}, V_0^\star) &= \frac{1}{N} + \frac{2}{N}\sum_{l=1}^{m-1} \cos^2\left(\frac{\pi l}{N}\left(x - \frac{1}{2}\right)\right) \\
\left[\text{let } \theta := \frac{x-\frac{1}{2}}{N}\right] &= \frac{1}{N} + \frac{2}{N}\sum_{l=1}^{m-1} \cos^2(l\pi\theta) \\
&= \frac{1}{N} + \frac{1}{2N}\left(\frac{\sin(N\pi\theta)}{\sin\pi\theta} + N - 2\right) \\
&= \frac{1}{N} + \frac{1}{2N}\left(\frac{-\cos(\pi x)}{\sin\pi\theta} + N - 2\right) \\
&= \frac{1}{2} + \frac{1}{2N}(-1)^{x-1}\csc\left(\frac{\pi}{N}\left(x - \frac{1}{2}\right)\right).
\end{aligned}$$

Therefore, for $x = 1:N$,

$$\text{score}(\boldsymbol{\delta}_x) = \frac{1}{N}(-1)^{x-1}\csc\left(\frac{\pi}{N}\left(x - \frac{1}{2}\right)\right).$$

Again, score$(\cdot)$ is symmetric based on the formula, i.e., score$(\boldsymbol{\delta}_x)$ = score$(\boldsymbol{\delta}_{N+1-x})$, so we only need to analyze it for $x = 1:m$. Meanwhile, csc$(\cdot)$ is also a positive and decreasing function over the interval $(0, \pi/2]$, so the series $\{\text{score}(\boldsymbol{\delta}_x)\}_{x=1:m}$ is an alternating series and its terms monotonically decrease in magnitude. Using the similar arguments in Case 1, i.e., split into $m = 2n$ and $m = 2n-1$ two subcases and so forth, we end up getting m $\boldsymbol{\delta}_x$'s that have the largest m values of score$(\cdot)$ in both subcases:

$$V_0 = \{\boldsymbol{\delta}_{2x-1}\}_{x=1:m}.$$

Finally, we set $V_1 = V \setminus V_0 = \{\boldsymbol{\delta}_{2x}\}_{x=1:m-1}$.

□

A.5. Proof of Theorem 10.2.1

PROOF. Since $\mathcal{F}_l = \mathrm{diag}(\boldsymbol{\mu}_l)$, it is equivalent to show

$$\frac{1}{N}\mathbf{1} < \sum_{l=0}^{N-1} \boldsymbol{\mu}_l < \frac{N+1}{2}\mathbf{1}.$$

In other words, we need to show the above inequalities hold for each entry of $\boldsymbol{\mu}_l$, i.e.,

$$\frac{1}{N} < \sum_{l=0}^{N-1} \mu_l(i+1) < \frac{N+1}{2}, \qquad i = 0 : N-1. \tag{A.1}$$

For each $\boldsymbol{\mu}_l$ ($l = 0 : N-1$), its entries satisfy $\sum_{i=0}^{N-1} \mu_l(i+1) = 1$ with $\mu_l(i+1) > 0$ and

$$l = \underset{i=0:N-1}{\arg\max}\, \mu_l(i+1) = \underset{i=0:N-1}{\arg\max} \frac{\exp\left(-(d(\phi_i, \phi_l)/\sigma)^2\right)}{\sum_{i'=0}^{N-1} \exp\left(-(d(\phi_{i'}, \phi_l)/\sigma)^2\right)},$$

because $d(\phi_l, \phi_l) = 0$. Now, the first inequality in Eq. (A.1) can be derived as follows.

$$\sum_{l=0}^{N-1} \mu_l(i+1) > \mu_l(l+1) > \frac{1}{N} \sum_{i=0}^{N-1} \mu_l(i+1) = \frac{1}{N}.$$

On the other hand, we can show the second inequality in Eq. (A.1) as below.

$$\begin{aligned}
\sum_{l=0}^{N-1} \mu_l(i+1) &= \mu_i(i+1) + \sum_{\substack{l=0:N-1 \\ l \neq i}} \mu_l(i+1) \\
&< 1 + \sum_{\substack{l=0:N-1 \\ l \neq i}} \frac{1}{2}(\mu_l(i+1) + \mu_l(l+1)) \\
&< 1 + \sum_{\substack{l=0:N-1 \\ l \neq i}} \frac{1}{2} \sum_{i'=0}^{N-1} \mu_l(i'+1) \\
&= \frac{N+1}{2}.
\end{aligned}$$

□

Bibliography

[1] M. BELKIN AND P. NIYOGI, *Laplacian eigenmaps for dimensionality reduction and data representation*, Neural Comput., 15 (2003), pp. 1373–1396.

[2] R. BERNARDINI AND J. KOVAČEVIĆ, *Designing local orthogonal bases on finite groups. I. Abelian case*, J. Fourier Anal. Appl., 6 (2000), pp. 1–23.

[3] ———, *Designing local orthogonal bases on finite groups. II. Nonabelian case*, J. Fourier Anal. Appl., 6 (2000), pp. 207–231.

[4] J. BEZANSON, A. EDELMAN, S. KARPINSKI, AND V. B. SHAH, *Julia: A fresh approach to numerical computing*, SIAM review, 59 (2017), pp. 65–98.

[5] I. BORG AND P. J. F. GROENEN, *Modern Multidimensional Scaling: Theory and Applications*, Springer, New York, 2nd ed., 2005.

[6] T. BRELOFF, *JuliaPlots: Plots.jl*. `https://github.com/JuliaPlots/Plots.jl`, 2021.

[7] X. CHENG AND G. MISHNE, *Spectral embedding norm: Looking deep into the spectrum of the graph Laplacian*, SIAM J. Imaging Sci., 13 (2020), pp. 1015–1048.

[8] A. CLONINGER AND W. CZAJA, *Eigenvector localization on data-dependent graphs*, in 2015 International Conference on Sampling Theory and Applications (SampTA), IEEE, 2015, pp. 608–612.

[9] A. CLONINGER, H. LI, AND N. SAITO, *Natural graph wavelet packet dictionaries*, J. Fourier Anal. Appl., 27 (2021). DOI:10.1007/s00041-021-09832-3.

[10] A. CLONINGER AND S. STEINERBERGER, *On the dual geometry of Laplacian eigenfunctions*, Experimental Mathematics, (2018). DOI:10.1080/10586458.2018.1538911.

[11] R. R. COIFMAN AND M. GAVISH, *Harmonic analysis of digital data bases*, in Wavelets and Multiscale Analysis: Theory and Applications, J. Cohen and A. I. Zayed, eds., Applied and Numerical Harmonic Analysis, Boston, MA, 2011, Birkhäuser, pp. 161–197.

[12] R. R. COIFMAN AND S. LAFON, *Diffusion maps*, Appl. Comput. Harmon. Anal., 21 (2006), pp. 5–30.

[13] R. R. COIFMAN AND M. MAGGIONI, *Diffusion wavelets*, Appl. Comput. Harmon. Anal., 21 (2006), pp. 53–94.

[14] R. R. COIFMAN AND Y. MEYER, *Nouvelles bases orthonormées de $L^2(\mathbb{R})$ ayant la structure du systèm de Walsh*, preprint, Dept. of Mathematics, Yale University, New Haven, CT, Aug. 1989.

[15] ———, *Remarques sur l'analyse de Fourier à fenêtre*, C. R. Acad. Sci. Paris Sér. I Math., 312 (1991), pp. 259–261.

[16] R. R. COIFMAN AND M. V. WICKERHAUSER, *Entropy-based algorithms for best basis selection*, IEEE Trans. Inform. Theory, 38 (1992), pp. 713–718.

[17] I. DAUBECHIES, *Ten Lectures on Wavelets*, vol. 61 of CBMS-NSF Regional Conference Series in Applied Mathematics, SIAM, Philadelphia, PA, USA, 1992.

[18] M. DEFFERRARD, L. MARTIN, R. PENA, AND N. PERRAUDIN, *PyGSP: Graph signal processing in Python*, 2017. DOI:10.5281/zenodo.1003157.

[19] I. DUNNING, J. HUCHETTE, AND M. LUBIN, *Jump: A modeling language for mathematical optimization*, SIAM Review, 59 (2017), pp. 295–320.

[20] H. E. EGILMEZ AND A. ORTEGA, *Spectral anomaly detection using graph-based filtering for wireless sensor networks*, in 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2014, pp. 1085–1089.

[21] B. ENGQUIST AND Y. YANG, *Seismic inversion and the data normalization for optimal transport*, Methods Appl. Anal., 26 (2019), pp. 133–147.

[22] P. ERDŐS AND A. RÉNYI, *On random graphs. I*, Publ. Math. Debrecen, 6 (1959), pp. 290–297.

[23] G. B. FOLLAND, *Fourier Analysis and Its Applications*, The Wadsworth & Brooks/Cole Mathematics Series, Wadsworth & Brooks/Cole Advanced Books & Software, Pacific Grove, CA, 1992.

[24] L. R. FOULDS, *Graph Theory Applications*, Universitext, Springer-Verlag, New York, 1992.

[25] 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 Proc. 27th Intern. Conf. Machine Learning, J. Fürnkranz and T. Joachims, eds., Haifa, Israel, 2010, Omnipress, pp. 367–374.

[26] B. GIRAULT AND A. ORTEGA, *What's in a frequency: New tools for graph Fourier transform visualization*, 2019.

[27] C. GODSIL AND G. ROYLE, *Algebraic Graph Theory*, vol. 207 of Graduate Texts in Mathematics, Springer, New York, 2001.

[28] J. C. GOWER, *Some distance properties of latent root and vector methods used in multivariate analysis*, Biometrika, 53 (1966), pp. 325–338.

[29] D. K. HAMMOND, P. VANDERGHEYNST, AND R. GRIBONVAL, *Wavelets on graphs via spectral graph theory*, Appl. Comput. Harmon. Anal., 30 (2011), pp. 129–150.

[30] H.BIRKHOLZ, *A unifying approach to isotropic and anisotropic total variation denoising models*, J. Comput. Appl. Math., 235 (2011), pp. 2502–2514.

[31] J. IRION, *Multiscale Transforms for Signals on Graphs: Methods and Applications*, Ph.D. dissertation, University of California, Davis, 2015.

[32] J. IRION, H. LI, N. SAITO, AND Y. SHAO, *MultiscaleGraphSignalTransforms.jl*. `https://github.com/UCD4IDS/MultiscaleGraphSignalTransforms.jl`, 2021.

[33] J. IRION AND N. SAITO, *The generalized Haar-Walsh transform*, in Proc. 2014 IEEE Workshop on Statistical Signal Processing, 2014, pp. 472–475.

[34] ———, *Hierarchical graph Laplacian eigen transforms*, JSIAM Lett., 6 (2014), pp. 21–24.

[35] ———, *Applied and computational harmonic analysis on graphs and networks*, in Wavelets and Sparsity XVI, Proc. SPIE 9597, M. Papadakis, V. K. Goyal, and D. Van De Ville, eds., 2015. Paper # 95971F.

[36] ———, *Efficient approximation and denoising of graph signals using the multiscale basis dictionaries*, IEEE Trans. Signal and Inform. Process. Netw., 3 (2017), pp. 607–616.

[37] S. JAFFARD, Y. MEYER, AND R. D. RYAN, *Wavelets: Tools for Science & Technology*, SIAM, Philadelphia, PA, 2001.

[38] R. I. JENNRICH, *A simple general procedure for orthogonal rotation*, Psychometrika, 66 (2001), pp. 289–306.

[39] S. G. JOHNSON, *QuadGK.jl: Gauss–Kronrod integration in Julia*. `https://github.com/JuliaMath/QuadGK.jl`, 2020.

[40] I. T. JOLLIFFE, *Principal Component Analysis*, Springer Series in Statistics, Springer-Verlag, New York, second ed., 2002.

[41] P. W. JONES, M. MAGGIONI, AND R. SCHUL, *Manifold parametrizations by eigenfunctions of the Laplacian and heat kernels*, Proc. Natl. Acad. Sci. USA, 105 (2008), pp. 1803–1808.

[42] H. F. KAISER, *The varimax criterion for analytic rotation in factor analysis*, Psychometrika, 23 (1958), pp. 187–200.

[43] D. KOHLI, A. CLONINGER, AND G. MISHNE, *LDLE: Low distortion local eigenmaps*, 2021.

[44] S. KOLOURI, S. R. PARK, M. THORPE, D. SLEPCEV, AND G. K. ROHDE, *Optimal mass transport: Signal processing and machine-learning applications*, IEEE Signal Processing Magazine, 34 (2017), pp. 43–59.

[45] Y. LECUN AND C. CORTES, *MNIST handwritten digit database*. `http://yann.lecun.com/exdb/mnist`, 2010.

[46] A. LEE, B. NADLER, AND L. WASSERMAN, *Treelets—an adaptive multi-scale basis for sparse unordered data*, Ann. Appl. Stat., 2 (2008), pp. 435–471.

[47] G. LEUS, S. SEGARRA, A. RIBEIRO, AND A. G. MARQUES, *The dual graph shift operator: Identifying the support of the frequency domain*, 2017.

[48] R. LEVIE, F. MONTI, X. BRESSON, AND M. M. BRONSTEIN, *CayleyNets: Graph convolutional neural networks with complex rational spectral filters*, IEEE Trans. Signal Process., 67 (2018), pp. 97–109.

[49] E. LEVINA AND P. BICKEL, *The earth mover's distance is the Mallows distance: Some insights from statistics*, in Proc. 8th IEEE International Conference on Computer Vision (ICCV), vol. 2, 2001, pp. 251–256.

[50] H. LI, *NGWP.jl*. `https://github.com/UCD4IDS/NGWP.jl`, 2021.

[51] H. LI AND N. SAITO, *Metrics of graph Laplacian eigenvectors*, in Wavelets and Sparsity XVIII, Proc. SPIE 11138, D. Van De Ville, M. Papadakis, and Y. M. Lu, eds., 2019. Paper #111381K.

[52] M. LINDBERG AND L. F. VILLEMOES, *Image compression with adaptive Haar-Walsh tilings*, in Wavelet Applications in Signal and Image Processing VIII, Proc. SPIE 4119, A. Aldroubi, A. F. Laine, and M. A. Unser, eds., 2000, pp. 911–921.

[53] O. LINDENBAUM, M. SALHOV, A. YEREDOR, AND A. AVERBUCH, *Gaussian bandwidth selection for manifold learning and classification*, Data Min. Knowl. Discov., 34 (2020), pp. 1676–1712.

[54] S. MALLAT, *A Wavelet Tour of Signal Processing*, Academic Press, Burlington, MA, third ed., 2009.

[55] A. M. MATHAI AND T. A. DAVIS, *Constructing the sunflower head*, Math. Biosci., 20 (1974), pp. 117–133.

[56] G. MISHNE AND I. COHEN, *Multiscale anomaly detection using diffusion maps*, IEEE Journal of Selected Topics in Signal Processing, 7 (2012), pp. 111–123.

[57] S. A. MULAIK, *Foundations of Factor Analysis*, CRC Press, 2nd ed., 2010.

[58] F. MURTAGH, *The Haar wavelet transform of a dendrogram*, J. Classification, 24 (2007), pp. 3–32.

[59] B. NADLER AND M. GALUN, *Fundamental limitations of spectral clustering*, in Advances in Neural Information Processing Systems, 2007, pp. 1017–1024.

[60] Y. NAKATSUKASA, N. SAITO, AND E. WOEI, *Mysteries around the graph Laplacian eigenvalue* 4, Linear Algebra Appl., 438 (2013), pp. 3231–3246.

[61] A. NG, M. JORDAN, AND Y. WEISS, *On spectral clustering: Analysis and an algorithm*, Advances in Neural Information Processing Systems, 14 (2001), pp. 849–856.

[62] A. ORTEGA, P. FROSSARD, J. KOVAČEVIĆ, J. M. F. MOURA, AND P. VANDERGHEYNST, *Graph signal processing: Overview, challenges, and applications*, Proc. IEEE, 106 (2018), pp. 808–828.

[63] C. H. PAPADIMITRIOU AND K. STEIGLITZ, *Combinatorial Optimization: Algorithms and Complexity*, Dover Publications, Inc., Mineola, NY, 1998. Corrected reprint of the 1982 original.

[64] N. PERRAUDIN, B. RICAUD, D. I. SHUMAN, AND P. VANDERGHEYNST, *Global and local uncertainty principles for signals on graphs*, APSIPA Trans. Signal and Image Process., 7 (2018). E3.

[65] G. PEYRÉ AND M. CUTURI, *Computational optimal transport: With applications to data science*, Foundations and Trends® in Machine Learning, 11 (2019), pp. 355–607.

[66] B. RICAUD, D. I. SHUMAN, AND P. VANDERGHEYNST, *On the sparsity of wavelet coefficients for signals on graphs*, in Wavelets and Sparsity XV, Proc. SPIE 8858, D. V. D. Ville, V. K. Goyal, and M. Papadakis, eds., 2013. Paper # 88581L.

[67] R. W. RODIECK, *The First Steps in Seeing*, Sinauer Associates, Inc., Sunderland, MA, 1998.

[68] B. RUDIS, N. ROSS, AND S. GARNIER, *The viridis color palettes*. `https://cran.r-project.org/web/packages/viridis/vignettes/intro-to-viridis.html`, 2018.

[69] N. SAITO, *Local Fourier dictionary: A natural tool for data analysis*, in Wavelet Applications in Signal and Image Processing VII, M. A. Unser, A. Aldroubi, and A. F. Laine, eds., vol. 3813, International Society for Optics and Photonics, SPIE, 1999, pp. 610 – 624.

[70] ———, *The generalized spike process, sparsity, and statistical independence*, in Modern Signal Processing, D. Rockmore and J. D. Healy, eds., vol. 46 of MSRI Publications, Cambridge University Press, 2004, pp. 317–340.

[71] ———, *Applied harmonic analysis on graphs and networks*, Bull. Jpn. Soc. Ind. Appl. Math., 25 (2015), pp. 6–15. in Japanese.

[72] ———, *How can we naturally order and organize graph Laplacian eigenvectors?*, in Proc. 2018 IEEE Workshop on Statistical Signal Processing, 2018, pp. 483–487.

[73] N. SAITO AND E. WOEI, *Analysis of neuronal dendrite patterns using eigenvalues of graph Laplacians*, JSIAM Lett., 1 (2009), pp. 13–16. Invited paper.

[74] ———, *On the phase transition phenomenon of graph Laplacian eigenfunctions on trees*, RIMS Kôkyûroku, 1743 (2011), pp. 77–90.

[75] ———, *Tree simplification and the 'plateaux' phenomenon of graph Laplacian eigenvalues*, Linear Algebra Appl., 481 (2015), pp. 263–279.

[76] L. K. SAUL AND S. T. ROWEIS, *Think globally, fit locally: Unsupervised learning of low dimensional manifolds*, J. Mach. Learn. Res., 4 (2004), pp. 119–155.

[77] Y. SHAO AND N. SAITO, *The extended generalized Haar-Walsh transform and applications*, in Wavelets and Sparsity XVIII, Proc. SPIE 11138, D. Van De Ville, M. Papadakis, and Y. M. Lu, eds., 2019. Paper #111380C.

[78] J. SHI AND J. MALIK, *Normalized cuts and image segmentation*, IEEE Trans. Pattern Anal. Machine Intell., 22 (2000), pp. 888–905.

[79] D. I. SHUMAN, S. K. NARANG, P. FROSSARD, A. ORTEGA, AND P. VANDERGHEYNST, *The emerging field of signal processing on graphs*, IEEE Signal Processing Magazine, 30 (2013), pp. 83–98.

[80] D. I. SHUMAN, B. RICAUD, AND P. VANDERGHEYNST, *Vertex-frequency analysis on graphs*, Appl. Comput. Harmon. Anal., 40 (2016), pp. 260–291.

[81] S. STEINERBERGER, *On the spectral resolution of products of Laplacian eigenfunctions*, J. Spectr. Theory, 9 (2019), pp. 1367–1384.

[82] G. STRANG, *The discrete cosine transform*, SIAM Review, 41 (1999), pp. 135–147.

[83] 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 Wavelets XI, Proc. SPIE 5914, M. Papadakis, A. F. Laine, and M. A. Unser, eds., 2005. Paper # 59141D.

[84] THE JULIAGEOMETRY TEAM, *JuliaGeometry: Computational Geometry with Julia*. `https://github.com/JuliaGeometry`, 2021.

[85] C. M. THIELE AND L. F. VILLEMOES, *A fast algorithm for adapted time-frequency tilings*, Appl. Comput. Harmon. Anal., 3 (1996), pp. 91–99.

[86] H. VOGEL, *A better way to construct the sunflower head*, Math. Biosci., 44 (1979), pp. 179–189.

[87] U. VON LUXBURG, *A tutorial on spectral clustering*, Stat. Comput., 17 (2007), pp. 395–416.

[88] J. VONDRÁK, *Optimal approximation for the submodular welfare problem in the value oracle model*, in STOC'08, ACM, New York, 2008, pp. 67–74.

[89] Q. XIA, *Motivations, ideas and applications of ramified optimal transportation*, ESAIM Math. Model. Numer. Anal., 49 (2015), pp. 1791–1832. Special Issue – Optimal Transport.

[90] Y. YAMAGISHI AND T. SUSHIDA, *Archimedean Voronoi spiral tilings*, J. Phys. A: Math. Theor., 51 (2018), p. 045203.