## Data Analytics on Graphs

## Other titles in

## Computational Optimal Transport Gabriel Peyre and Marco Cuturi

 ISBN: 978-1-68083-550-2An Introduction to Deep Reinforcement Learning Vincent Francois-Lavet, Peter Henderson, Riashat Islam, Marc G. Bellemare and Joelle Pineau ISBN: 978-1-68083-538-0

An Introduction to Wishart Matrix Moments
Adrian N. Bishop, Pierre Del Moral and Angele Niclas
ISBN: 978-1-68083-506-9

A Tutorial on Thompson Sampling
Daniel J. Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband and Zheng Wen
ISBN: 978-1-68083-470-3

## Data Analytics on Graphs

Ljubiša Stanković
University of Montenegro
Montenegro
ljubisa@ucg.ac.me

## Danilo Mandic

Imperial College London
UK
d.mandic@imperial.ac.uk

Miloš Daković
University of Montenegro
Montenegro
milos@ucg.ac.me

## Miloš Brajović

University of Montenegro
Montenegro
milosb@ucg.ac.me

Bruno Scalzo

Imperial College London
UK
bruno.scalzo-dees12@imperial.ac.uk

## Shengxi Li

Imperial College London
UK
shengxi.li17@imperial.ac.uk

## Anthony G. Constantinides

Imperial College London
UK
a.constantinides@imperial.ac.uk

Published, sold and distributed by: now Publishers Inc.<br>PO Box 1024<br>Hanover, MA 02339<br>United States<br>Tel. +1-781-985-4510<br>www.nowpublishers.com<br>sales@nowpublishers.com<br>Outside North America:<br>now Publishers Inc.<br>PO Box 179<br>2600 AD Delft<br>The Netherlands<br>Tel. +31 -6-51115274

The preferred citation for this publication is
L. Stanković, D. Mandic, M. Daković, M. Brajović, B. Scalzo and A. G. Constantinides.

Data Analytics on Graphs. , vol. 13, no. 1, pp. 1-157, 2020.
ISBN: 978-1-68083-982-1
© 2021 L. Stanković, D. Mandic, M. Daković, M. Brajović, B. Scalzo and A. G. Constantinides

[^0]
# Volume 13, Issue 1, 2020 <br> Editorial Board 

## Editor-in-Chief

Michael Jordan<br>University of California, Berkeley<br>United States

## Editors

| Peter Bartlett | Aapo Hyvarinen | Luc de Raedt |
| :--- | :--- | :--- |
| UC Berkeley | Helsinki IIT | KU Leuven |
| Yoshua Bengio | Leslie Pack Kaelbling | Christian Robert |
| Université de Montréal | MIT | Paris-Dauphine |
| Avrim Blum | Michael Kearns | Sunita Sarawagi |
| Toyota Technological | UPenn | IIT Bombay |
| Institute | Daphne Koller | Robert Schapire |
| Craig Boutilier | Stanford University | Microsoft Research |
| University of Toronto | John Lafferty | Bernhard Schoelkopf |
| Stephen Boyd | Yale | Max Planck Institute |
| Stanford University | Michael Littman | Richard Sutton |
| Carla Brodley | Brown University | University of Alberta |
| Northeastern University | Gabor Lugosi | Larry Wasserman |
| Inderjit Dhillon | Pompeu Fabra | CMU |
| Texas at Austin | David Madigan | Bin Yu |
| Jerome Friedman | Columbia University | UC Berkeley |
| Stanford University | Pascal Massart |  |
| Kenji Fukumizu | Université de Paris-Sud |  |
| ISM | Andrew McCallum |  |
| Zoubin Ghahramani | University of |  |
| Cambridge University | Massachusetts Amherst |  |
| David Heckerman | Marina Meila |  |
| Amazon | University of Washington |  |
| Tom Heskes | Andrew Moore |  |
| Radboud University | CMU |  |
| Geoffrey Hinton | John Platt |  |
| University of Toronto | Microsoft Research |  |

## Editorial Scope

## Topics

Information for Librarians

## Contents

I Graphs and Spectra on Graphs ..... 1
1 Introduction ..... 3
2 Graph Definitions and Properties ..... 7
2.1 Basic Definitions ..... 7
2.2 Some Frequently Used Graph Topologies ..... 14
2.3 Properties of Graphs and Associated Matrices ..... 19
3 Spectral Decomposition of Graph Matrices ..... 29
3.1 Eigenvalue Decomposition of the Adjacency Matrix ..... 29
3.2 Spectral Graph Theory ..... 32
3.3 Eigenvalue Decomposition of the Graph Laplacian ..... 39
4 Vertex Clustering and Mapping ..... 49
4.1 Clustering Based on Graph Topology ..... 50
4.2 Spectral Methods for Graph Clustering ..... 58
4.3 Spectral Clustering Implementation ..... 75
4.4 Vertex Dimensionality Reduction Using the Laplacian Eigenmaps ..... 91
4.5 Pseudo-Inverse of Graph Laplacian-Based Mappings ..... 104
4.6 Summary of Embedding Mappings ..... 116
5 Graph Sampling Strategies ..... 119
5.1 Graph Down-Sampling Strategies ..... 119
5.2 Graph Sparsification ..... 121
5.3 Graph Coarsening ..... 127
5.4 Kron Reduction of Graphs ..... 134
6 Conclusion ..... 135
II Signals on Graphs ..... 137
7 Introduction ..... 139
8 Problem Statement: An Illustrative Example ..... 143
9 Signals and Systems on Graphs ..... 155
9.1 Adjacency Matrix and Graph Signal Shift ..... 158
9.2 Systems Based on Graph Shifted Signals ..... 160
9.3 Graph Fourier Transform (GFT), Adjacency Matrix Based Definition ..... 163
9.4 System on a Graph in the GFT Domain ..... 164
9.5 Graph Signal Filtering in the Spectral Domain of the Adjacency Matrix ..... 166
9.6 Graph Fourier Transform Based on the Laplacian ..... 178
9.7 Ordering and Filtering in the Laplacian Spectral Domain ..... 180
9.8 Systems on a Graph Defined Using the Graph Laplacian ..... 184
9.9 Convolution of Signals on a Graph ..... 186
9.10 The $z$-Transform of a Signal on a Graph ..... 188
9.11 Shift Operator in the Spectral Domain ..... 192
9.12 Parseval's Theorem on a Graph ..... 193
9.13 Optimal Denoising ..... 193
9.14 Summary of Shift Operators for Systems on a Graph ..... 197
10 Subsampling, Compressed Sensing, and Reconstruction ..... 199
10.1 Subsampling of Bandlimited Graph Signals ..... 199
10.2 Subsampling of Sparse Graph Signals ..... 203
10.3 Measurements as Linear Combinations of Samples ..... 211
10.4 Aggregate Sampling ..... 212
10.5 Random Sampling with Optimal Strategy ..... 215
11 Filter Bank on a Graph ..... 221
12 Time-Varying Signals on Graphs ..... 231
12.1 Diffusion on Graph and Low Pass Filtering ..... 232
12.2 Taubin's $\alpha-\beta$ Algorithm ..... 233
13 Random Graph Signal Processing ..... 237
13.1 Review of WSS and Related Properties for Random Signals in Standard Time Domain ..... 238
13.2 Adjacency Matrix Based Definition of GWSS ..... 239
13.3 Wiener Filter on a Graph ..... 240
13.4 Spectral Domain Shift Based Definition of GWSS ..... 241
13.5 Isometric Shift Operator ..... 242
14 Vertex-Frequency Representations ..... 243
14.1 Localized Graph Fourier Transform (LGFT) ..... 245
14.2 Inversion of the LGFT ..... 283
14.3 Uncertainty Principle for Graph Signals ..... 288
14.4 Graph Spectrogram and Frames ..... 290
14.5 Vertex-Frequency Energy Distributions ..... 294
15 Conclusion ..... 305
III Machine Learning on Graphs, from Graph Topology to Ap- plications ..... 307
16 Introduction ..... 309
17 Geometrically Defined Graph Topologies ..... 313
18 Graph Topology Based on Signal Similarity ..... 323
19 Learning of Graph Laplacian from Data ..... 331
19.1 Imposing Sparsity on the Connectivity Matrix ..... 334
19.2 Smoothness Constrained Learning of Graph Laplacian ..... 339
19.3 Graph Topology Estimation with the Graph Laplacian Energy Condition ..... 342
19.4 Learning of Generalized Laplacian-Graphical LASSO ..... 343
19.5 Graph Topology Learning Based on the Eigenvectors ..... 352
20 From Newton Minimization to Graphical LASSO, via LASSO ..... 363
20.1 Newton Method ..... 363
20.2 Standard LASSO ..... 365
20.3 Graphical LASSO ..... 368
21 Physically Well Defined Graphs ..... 375
21.1 Resistive Electrical Circuits ..... 375
21.2 Heat Transfer ..... 383
21.3 Spring-Mass Systems ..... 383
21.4 Social Networks and Linked Pages ..... 385
21.5 PageRank ..... 386
21.6 Random Walk ..... 389
21.7 Hitting and Commute Time ..... 393
21.8 Relating Gaussian Random Signal to Electric Circuits ..... 397
22 Graph Learning from Data and External Sources ..... 399
23 Random Signal Simulation on Graphs ..... 405
24 Summary of Graph Learning from Data Using Probabilistic Generative Models ..... 409
24.1 Basic Gaussian Models ..... 410
24.2 Gaussian Graphical Model ..... 412
24.3 Diffusion Models ..... 419
25 Graph Neural Networks ..... 425
25.1 Basic Graph Elements Related to GCNs ..... 427
25.2 Gradient Descent as a Diffusion Process ..... 430
25.3 Label Propagation as a Diffusion Process with External Sources ..... 431
25.4 GNNs of a Recurrent Style ..... 433
25.5 Spatial GCNs via Localization of Graphs ..... 436
25.6 Spectral GCNs via Graph Fourier Transform ..... 438
25.7 Link Prediction via Graph Neural Nets ..... 444
26 Tensor Representation of Lattice-Structured Graphs ..... 453
26.1 Tensorization of Graph Signals in High-Dimensional Spaces ..... 454
26.2 Tensor Decomposition ..... 455
26.3 Connectivity of a Tensor ..... 457
26.4 DFT of a Tensor ..... 460
26.5 Unstructured Graphs ..... 461
26.6 Tensor Representation of Multi-Relational Graphs ..... 462
26.7 Multi-Graph Tensor Networks ..... 464
27 Metro Traffic Modeling Through Graphs ..... 471
27.1 Traffic Centrality as a Graph-Theoretic Measure ..... 472
27.2 Modeling Commuter Population from Net Passenger Flow ..... 474
28 Portfolio Cuts ..... 481
28.1 Structure of Market Graph ..... 483
28.2 Minimum Cut Based Vertex Clustering ..... 484
28.3 Spectral Bisection Based Minimum Cut ..... 486
28.4 Repeated Portfolio Cuts ..... 488
28.5 Graph Asset Allocation Schemes ..... 490
28.6 Numerical Example ..... 491
29 Conclusion ..... 495
Appendices ..... 497
A Power Method for Eigenanalysis ..... 499
B Algorithm for Graph Laplacian Eigenmaps ..... 503
C Other Graph Laplacian Forms ..... 505
C. 1 Graph Laplacian for Directed Graphs ..... 505
C. 2 Signed Graphs and Signed Graph Laplacian ..... 507
C. 3 Graph p-Laplacian ..... 508
Acknowledgments ..... 511
References ..... 513
Index ..... 540

## Part I

## Graphs and Spectra on Graphs

## Introduction

Data analytics on graphs is a multidisciplinary research area, of which the roots can be traced back to the 1970s (Afrati and Constantinides, 1978; Christofides, 1975; Morris et al., 1986), one that is witnessing significant rapid growth. The recent developments, in response to the requirements posed by radically new classes of data sources, typically embark upon the classical results on "static" graph topology optimization, to treat graphs as irregular data domains, which make it possible to address completely new paradigms of "information processing on graphs" and "signal processing on graphs". This has already resulted in advanced and physically meaningful solutions in manifold applications (Grady and Polimeni, 2010; Jordan, 1998; Krim and Hamza, 2015; Marques et al., 2017; Ray, 2012). For example, while the emerging areas of Graph Machine Learning (GML) and Graph Signal Processing (GSP) do comprise the classic methods of optimization of graphs themselves (Bapat, 1996; Bunse-Gerstner and Gragg, 1988; Fujiwara, 1995; Grebenkov and Nguyen, 2013; Jordan et al., 2004; Maheswari and Maheswari, 2016; O'Rourke et al., 2016), significant progress has been made towards redefining basic data analysis objectives (spectral estimation, probabilistic inference, filtering, dimensionality reduction,
clustering, statistical learning), to make them amenable for direct estimation of signals on graphs (Chen et al., 2014; Ekambaram, 2014; Gavili and Zhang, 2017; Hamon et al., 2016b; Moura, 2018; Sandryhaila and Moura, 2013, 2014a,b; Shuman et al., 2013; Vetterli et al., 2014; Wainwright et al., 2008). Indeed, this is a necessity in numerous practical scenarios where the signal domain is not designated by equidistant instants in time or a regular grid in a space or a transform domain. Examples include modern Data Analytics for e.g., social network modeling or in smart grid - data domains which are typically irregular and, in some cases, not even related to the notions of time or space, where ideally, the data sensing domain should also reflect domainspecific properties of the considered system/network; for example, in social or web related networks, the sensing points and their connectivity may be related to specific individuals, objectives, or topics, and their relations, whereby the processing on irregular domains requires the consideration of data properties other than time or space relationships. In addition, even for the data sensed in well-defined time and space domains, the new contextual and semantic-related relations between the sensing points, introduced through graphs, promise to equip problem definition with physical relevance, and consequently provide new insights into analysis and can lead to enhanced data processing results.

In applications which admit the definition of the data domain as a graph (such as social networks, power grids, vehicular networks, and brain connectivity), the role of classic temporal/spatial sampling points is assumed by graph vertices - the nodes - where the data values are observed, while the edges between vertices designate the existence and nature of vertex connections (directionality, strength). In this way, graphs are perfectly well equipped to exploit the fundamental relations among both the measured data and the underlying graph topology; this inherent ability to incorporate physically relevant data properties has made GSP and GML key technologies in the emerging field of Big Data Analytics (BDA). Indeed, in applications defined on irregular data domains, Graph Data Analytics (GDA) has been shown to offer a quantum step forward from the classical time (or space) series analyses (Brouwer and Haemers, 2012; Cvetković and Doob, 1985; Cvetković
and Gutman 2011; Cvetković et al., 1980; Chung, 1997; Jones, 2013; Mejia et al., 2017; Stanković et al., 2017b, 2019), including the following aspects.

- Graph-based data processing approaches can be applied not only to technological, biological, and social networks, but also they can lead to both improvements of the existing and even to the creation of radically new methods in classical signal processing and machine learning (Dong et al., 2012; Hamon et al., 2016a; Horaud, 2009; Lu et al., 2014; Masoumi and Hamza, 2017; Masoumi et al., 2016; Stanković et al., 2017a, 2018).
- The involvement of graphs makes it possible for the classical sensing domains of time and space (which may be represented as a linear or circular graph) to be structured in a more advanced way, e.g., by considering the connectivity of sensing points from a signal similarity or sensor association point of view.

The first step in graph data analytics is to decide on the properties of the graph as a new signal/information domain. However, while the data sensing points (graph vertices) may be well-defined by the application itself, that is not the case with their connectivity (graph edges), where:

- In the case of the various computer, social, road, transportation and electrical networks, the vertex connectivity is often naturally defined, resulting in an exact underlying graph topology.
- In many other cases, the data domain definition in a graph form becomes part of the problem definition itself, as is the case with, e.g., graphs for sensor networks, in finance or smart cities. In such cases, a vertex connectivity scheme needs to be determined based on the properties of the sensing positions or from the acquired data, as e.g., in the estimation of the temperature field in meteorology (Stankovic et al., 2019a).

This additional aspect of the definition of an appropriate graph structure is of crucial importance for a meaningful and efficient application of the GML and GSP approaches.

With that in mind, this monograph was written in response to the urgent need of multidisciplinary data analytics communities for a seamless and rigorous transition from classical data analytics to the corresponding paradigms which operate directly on irregular graph domains. To this end, we start our approach from a review of basic definitions of graphs and their properties, followed by a physical intuition and step-by-step introduction of graph spectral analysis (eigen-analysis). Particular emphasis is on eigendecomposition of graph matrices, an area which serves as a basis for mathematical formalisms in graph signal and information processing. As an example of the ability of GML and GSP to generalize standard methodologies for graphs, we elaborate in a step-by-step way the introduction of Graph Discrete Fourier Transform (GDFT), and show that it simplifies into standard Discrete Fourier Transform (DFT) for directed circular graphs; this also exemplifies the generic nature of graph approaches. Finally, spectral vertex analysis and spectral graph segmentation are used as the basis for understanding relations among distinct but physically meaningful regions in graphs; this is demonstrated through examples of regional infrastructure modeling, brain connectivity, clustering, and dimensionality reduction.

## 2

## Graph Definitions and Properties

Graph theory has been established for almost three centuries as a branch in mathematics, and has become a staple methodology in science and engineering areas including chemistry, operational research, electrical and civil engineering, social networks, and computer sciences. The beginning of graph theory applications in electrical engineering can be traced back to the mid-19th century with the introduction of Kirchoff's laws. Fast forward two centuries or so, the analytics of data acquired on graphs has become a rapidly developing research paradigm in Signal Processing and Machine Learning (Grady and Polimeni, 2010; Krim and Hamza, 2015; Marques et al., 2017; Ray, 2012).

### 2.1 Basic Definitions

Definition: A graph $\mathcal{G}=\{\mathcal{V}, \mathcal{B}\}$ is defined as a set of vertices, $\mathcal{V}$, which are connected by a set of edges, $\mathcal{B} \subset \mathcal{V} \times \mathcal{V}$, where the symbol $\times$ denotes a direct product operator.

Examples of graph topologies with $N=8$ vertices, with

$$
\mathcal{V}=\{0,1,2,3,4,5,6,7\}
$$



Figure 2.1: Basic graph structures. (a) Undirected graph and (b) Directed graph.
are presented in Figure 2.1, along with the corresponding edges. The vertices are usually depicted as points (circles) and the edges as lines that connect the vertices. More formally, a line between the vertices $m$ and $n$ indicates the existence of an edge between vertices $m$ and $n$, that is, $(m, n) \in \mathcal{B}$, so that, for example, the graph from Figure 2.1(b) can be described as

$$
\begin{gathered}
\mathcal{V}=\{0,1,2,3,4,5,6,7\} \\
\mathcal{B} \subset\{0,1,2,3,4,5,6,7\} \times\{0,1,2,3,4,5,6,7\} \\
\mathcal{B}=\{(0,1),(1,2),(2,0),(2,3),(2,4),(2,7),(3,0), \\
(4,1),(4,2),(4,5),(5,7),(6,3),(6,7),(7,2),(7,6)\} .
\end{gathered}
$$

Regarding the directionality of vertex connections, a graph can be undirected and directed, as illustrated respectively in Figures 2.1(a) and (b).

Definition: A graph is undirected if the edge connecting a vertex $m$ to a vertex $n$ also connects the vertex $n$ to the vertex $m$, for all $m$ and $n$.

In other words, for an undirected graph, if $(n, m) \in \mathcal{B}$ then also $(m, n) \in \mathcal{B}$, as in the case, for example, with edges $(1,2)$ and $(2,1)$ in Figure 2.1(a). For directed graphs, in general, this property does not hold, as shown in Figure 2.1(b). Observe, for example, that the edge $(2,1)$ does not exist, although the edge $(1,2)$ connects vertices 1 and 2. Therefore, undirected graphs can be considered as a special case of directed graphs.

For a given set of vertices and edges, a graph can be formally represented by its adjacency matrix, A, which describes the vertex connectivity; for $N$ vertices $\mathbf{A}$ is an $N \times N$ matrix.

Definition: The elements $A_{m n}$ of the adjacency matrix $\mathbf{A}$ assume values $A_{m n} \in\{0,1\}$. The value $A_{m n}=0$ is assigned if the vertices $m$ and $n$ are not connected with an edge, and $A_{m n}=1$ if these vertices are connected, that is

$$
A_{m n} \stackrel{\text { def }}{=} \begin{cases}1, & \text { if }(m, n) \in \mathcal{B} \\ 0, & \text { if }(m, n) \notin \mathcal{B} .\end{cases}
$$

Therefore, the respective adjacency matrices, $\mathbf{A}_{\text {un }}$ and $\mathbf{A}_{\text {dir }}$, for the undirected and directed graphs from Figures 2.1(a) and (b) are given by

$$
\mathbf{A}_{\mathrm{un}}=\begin{gather*}
0  \tag{2.1}\\
1 \\
2 \\
3 \\
4 \\
5 \\
6 \\
7
\end{gather*}\left[\begin{array}{cccccccc}
0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 0
\end{array}\right]
$$

$$
\mathbf{A}_{\text {dir }}=\begin{gather*}
0  \tag{2.2}\\
1 \\
2 \\
3 \\
4 \\
5 \\
6 \\
7
\end{gather*}\left[\begin{array}{llllllll}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 0
\end{array}\right] .
$$

Adjacency matrices not only fully reflect the structure arising from the topology of data acquisition, but also they admit analysis through linear algebra, and can be sparse, or exhibit some other interesting and useful matrix properties.
Remark 1: The adjacency matrix of an undirected graph is symmetric, that is,

$$
\mathbf{A}=\mathbf{A}^{T}
$$

Since a graph is fully determined by its adjacency matrix, defined over a given set of vertices, any change in vertex ordering will cause the corresponding changes in the adjacency matrix.
Remark 2: Observe that a vertex indexing scheme does not change the graph itself (graphs are isomorphic domains), so that the relation between adjacency matrices of the original and renumerated graphs, $\mathbf{A}_{1}$ and $\mathbf{A}_{2}$ respectively, is straightforwardly defined using an appropriate permutation matrix, $\mathbf{P}$, in the form

$$
\begin{equation*}
\mathbf{A}_{2}=\mathbf{P} \mathbf{A}_{1} \mathbf{P}^{T} \tag{2.3}
\end{equation*}
$$

Recall that each row and each column of a permutation matrix has exactly one nonzero element equal to unity.

In general, in the context of an application the edges can also convey information about a relative importance about the vertices they interconnect, through a weighted graph.
Remark 3: The set of weights, $\mathcal{W}$, corresponds morphologically to the set of edges, $\mathcal{B}$, so that a weighted graph represents a generic extension of an unweighted graph. It is commonly assumed that edge


Figure 2.2: Example of a weighted graph.
weights are nonnegative real numbers; therefore, if weight 0 is associated with a nonexisting edge, then the graph can be described by a weight matrix, $\mathbf{W}$, similar to the description by the adjacency matrix $\mathbf{A}$.

Definition: A nonzero element in the weight matrix $\mathbf{W}, W_{m n} \in \mathcal{W}$, designates both an edge between the vertices $m$ and $n$ and the corresponding weight. The value $W_{m n}=0$ indicates no edge connecting the vertices $m$ and $n$. The elements of a weight matrix are nonnegative real numbers.

Figure 2.2 shows an example of a weighted undirected graph, with the corresponding weight matrix given by

$$
\mathbf{W}=\begin{array}{r}
0  \tag{2.4}\\
1 \\
2 \\
3 \\
4 \\
5 \\
6 \\
7
\end{array}\left[\begin{array}{rrrrrrrr}
0 & 0.23 & 0.74 & 0.24 & 0 & 0 & 0 & 0 \\
0.23 & 0 & 0.35 & 0 & 0.23 & 0 & 0 & 0 \\
0.74 & 0.35 & 0 & 0.26 & 0.24 & 0 & 0 & 0 \\
0.24 & 0 & 0.26 & 0 & 0 & 0 & 0.32 & 0 \\
0 & 0.23 & 0.24 & 0 & 0 & 0.51 & 0 & 0.14 \\
0 & 0 & 0 & 0 & 0.51 & 0 & 0 & 0.15 \\
0 & 0 & 0 & 0.32 & 0 & 0 & 0 & 0.32 \\
0 & 0 & 0 & 0 & 0.14 & 0.15 & 0.32 & 0 \\
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7
\end{array}\right] .
$$

In this sense, the adjacency matrix, $\mathbf{A}$, can be considered as a special case of the weight matrix, $\mathbf{W}$, whereby all nonzero weights are equal to unity. It then follows that the weight matrix of undirected graphs is also symmetric

$$
\begin{equation*}
\mathbf{W}=\mathbf{W}^{T} \tag{2.5}
\end{equation*}
$$

while, in general, for directed graphs this property does not hold.

Definition: A degree matrix, $\mathbf{D}$, of an undirected graph is a diagonal matrix with elements, $D_{m m}$, which are equal to the sum of weights of all edges connected to the vertex $m$, that is, the sum of elements in the $m$-th row of the weight matrix, $\mathbf{W}$,

$$
D_{m m} \stackrel{\text { def }}{=} \sum_{n=0}^{N-1} W_{m n} .
$$

Remark 4: For an unweighted and undirected graph, the value of the element $D_{m m}$ is equal to the number of edges connected to the $m$-th vertex.

The degree matrices for directed graphs will be consider in the Appendix on the Laplacian of directed graphs.

Vertex degree centrality. The degree centrality of a vertex is defined as the number of vertices connected to the considered vertex with a single edge, and in this way it models the importance of a given vertex. For undirected and unweighted graphs, the vertex degree centrality of a vertex $m$ is equal to the element, $D_{m m}$, of the degree matrix.
Example 1: For the undirected weighted graph from Figure 2.2, the degree matrix is given by

$$
\mathbf{D}=\begin{gather*}
0  \tag{2.6}\\
1 \\
2 \\
3 \\
4 \\
5 \\
6 \\
7
\end{gather*}\left[\begin{array}{cccccccc}
1.21 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.81 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1.59 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.82 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1.12 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.66 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0.64 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.61 \\
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7
\end{array}\right] .
$$

Another important descriptor of graph connectivity is the graph Laplacian matrix, $\mathbf{L}$, which combines the weight matrix and the degree matrix.

Definition: The graph Laplacian matrix is defined as

$$
\begin{equation*}
\mathbf{L} \stackrel{\text { def }}{=} \mathbf{D}-\mathbf{W} \tag{2.7}
\end{equation*}
$$

where $\mathbf{W}$ is the weight matrix and $\mathbf{D}$ the diagonal degree matrix with elements $D_{m m}=\sum_{n} W_{m n}$. The elements of a Laplacian matrix are therefore nonnegative real numbers at the diagonal positions, and nonpositive real numbers at the off-diagonal positions.

For an undirected graph, the Laplacian matrix is symmetric, that is, $\mathbf{L}=\mathbf{L}^{T}$. For example, the graph Laplacian for the weighted graph from Figure 2.2 is given by

$$
\mathbf{L}=\left[\begin{array}{rrrrrrrr}
1.21 & -0.23 & -0.74 & -0.24 & 0 & 0 & 0 & 0  \tag{2.8}\\
-0.23 & 0.81 & -0.35 & 0 & -0.23 & 0 & 0 & 0 \\
-0.74 & -0.35 & 1.59 & -0.26 & -0.24 & 0 & 0 & 0 \\
-0.24 & 0 & -0.26 & 0.82 & 0 & 0 & -0.32 & 0 \\
0 & -0.23 & -0.24 & 0 & 1.12 & -0.51 & 0 & -0.14 \\
0 & 0 & 0 & 0 & -0.51 & 0.66 & 0 & -0.15 \\
0 & 0 & 0 & -0.32 & 0 & 0 & 0.64 & -0.32 \\
0 & 0 & 0 & 0 & -0.14 & -0.15 & -0.32 & 0.61
\end{array}\right] .
$$

For practical reasons, it is often advantageous to use the normalized Laplacian, defined as

$$
\begin{equation*}
\mathbf{L}_{N} \stackrel{\text { def }}{=} \mathbf{D}^{-1 / 2}(\mathbf{D}-\mathbf{W}) \mathbf{D}^{-1 / 2}=\mathbf{I}-\mathbf{D}^{-1 / 2} \mathbf{W} \mathbf{D}^{-1 / 2} \tag{2.9}
\end{equation*}
$$

Remark 5: For undirected graphs, the normalized Laplacian matrix is symmetric, and has all diagonal values equal to 1 , with its trace equal to the number of vertices $N$.

Other interesting properties, obtained through Laplacian normalization, shall be described later in the various application contexts.

One more form of the graph Laplacian is the so called random-walk Laplacian, defined as

$$
\begin{equation*}
\mathbf{L}_{R W} \stackrel{\text { def }}{=} \mathbf{D}^{-1} \mathbf{L}=\mathbf{I}-\mathbf{D}^{-1} \mathbf{W} . \tag{2.10}
\end{equation*}
$$

The random-walk graph Laplacian is rarely used, since it has lost the symmetry property of the original graph Laplacian for undirected graphs, $\mathbf{L}_{R W} \neq \mathbf{L}_{R W}^{T}$.
Vertex-weighted graphs. Most of the applications of graph theory are based on edge-weighted graphs, where edge-weighting is designated by the weight matrix, $\mathbf{W}$. Note that weighting can be also introduced into graphs based on vertex-weighted approaches (although rather
rarely), whereby a weight is assigned to each vertex of a graph. To this end, we can use a diagonal matrix, $\mathbf{V}$, to define the vertex weights $v_{i}$, $i=0,1, \ldots, N-1$, with one possible (Chung and Langlands, 1996) version of the vertex-weighted graph Laplacian, given by

$$
\begin{equation*}
\mathbf{L}_{V} \stackrel{\text { def }}{=} \mathbf{V}^{1 / 2} \mathbf{L V}^{1 / 2} \tag{2.11}
\end{equation*}
$$

Observe that for $\mathbf{V}=\mathbf{D}^{-1}$, the vertex-weighted graph Laplacian in (2.11) reduces to the standard edge-weighted normalized graph Laplacian in (2.9).

### 2.2 Some Frequently Used Graph Topologies

When dealing with graphs, it is useful to introduce a taxonomy of graph topologies, as follows.

1. Complete graph. A graph is complete if there exists an edge between every pair of its vertices. Therefore, the adjacency matrix of a complete graph has elements $A_{m n}=1$ for all $m \neq n$, and $A_{m m}=0$, that is, no self-connections are present. Figure 2.3(a) gives an example of a complete graph.
2. Bipartite graph. A graph for which the vertices, $\mathcal{V}$, can be partitioned into two disjoint subsets, $\mathcal{E}$ and $\mathcal{H}$, whereby $\mathcal{V}=\mathcal{E} \cup \mathcal{H}$ and $\mathcal{E} \cap \mathcal{H}=\emptyset$, such that there are no edges between the vertices within the same subset $\mathcal{E}$ or $\mathcal{H}$, is referred to as a bipartite graph. Figure 2.3(b) gives an example of a bipartite undirected graph with $\mathcal{E}=\{0,1,2\}$ and $\mathcal{H}=\{3,4,5,6\}$, whereby all edges designate only connections between the sets $\mathcal{E}$ and $\mathcal{H}$. Observe also that the graph in Figure 2.3(b) is a complete bipartite graph, since all possible edges between the sets $\mathcal{E}$ and $\mathcal{H}$ are present.

For convenience of mathematical formalism, if vertex ordering is performed in a such way that all vertices belonging to $\mathcal{E}$ are indexed before the vertices belonging to $\mathcal{H}$, then the resulting adjacency matrix can be written in a block form

$$
\mathbf{A}=\left[\begin{array}{cc}
\mathbf{0} & \mathbf{A}_{\mathcal{E}}  \tag{2.12}\\
\mathbf{A}_{\mathcal{H E}} & \mathbf{0}
\end{array}\right]
$$


(a) Complete graph

(c) Regular graph

(b) Bipartite graph

(d) Star graph

(e) Circular graph

(f) Path graph

(g) Directed circular graph

(h) Directed path graph

Figure 2.3: Special graph topologies. (a) Complete graph with 8 vertices. (b) Complete bipartite graph. (c) Regular graph whereby each vertex is connected to 4 vertices. (d) Star graph. (e) Circular graph. (f) Path graph. (g) Directed circular graph. (h) Directed path graph.
where the submatrices $\mathbf{A}_{\mathcal{E H}}$ and $\mathbf{A}_{\mathcal{H E}}$ define the respective connections between the vertices belonging to the disjoint sets $\mathcal{E}$ and $\mathcal{H}$. Observe that for an undirected bipartite graph, $\mathbf{A}_{\mathcal{E}}=\mathbf{A}_{\mathcal{H} \mathcal{E}}^{T}$. Bipartite graphs are also referred to as Kuratowski graphs, denoted by $K_{N_{\mathcal{E}}, N_{\mathcal{H}}}$, where $N_{\mathcal{E}}$ and $N_{\mathcal{H}}$ are the respective numbers of vertices in the sets $\mathcal{E}$ and $\mathcal{H}$. It is important to mention that a complete bipartite graph with three vertices in each of the sets, $\mathcal{H}$ and $\mathcal{E}$, is referred to as the first Kuratowski graph, denoted by $K_{3,3}$, which may be used to define conditions for a graph to be planar (more detail is given in the sequel).
Multipartite graph. A generalization of the concept of bipartite graph is a multipartite ( $M$-partite) graph for which the vertices are partitioned into $M$ subsets, whereby each edge connects only vertices that belong to different subsets.
3. Regular graph. An unweighted graph is said to be regular (or $\mathcal{J}$ regular) if all its vertices exhibit the same degree of connectivity, $\mathcal{J}$, which is defined as the number of edges connected to each vertex. An example of a regular graph with $\mathcal{J}=4$ is given in Figure 2.3(c). From (2.7) and (2.9), the Laplacian and the normalized Laplacian of a $\mathcal{J}$-regular graph are

$$
\begin{equation*}
\mathbf{L}=\mathcal{J} \mathbf{I}-\mathbf{A} \quad \text { and } \quad \mathbf{L}_{N}=\mathbf{I}-\frac{1}{\mathcal{J}} \mathbf{A} . \tag{2.13}
\end{equation*}
$$

4. Planar graph. A graph that can be drawn on a two-dimensional plane without the crossing of any of its edges is called planar.
For example, if the edges $(0,2),(2,4),(4,6)$, and $(6,0)$ in the regular graph from Figure 2.3(c) are plotted as arches outside the circle defined by the vertices, all instances of edge crossing will be avoided and such graph presentation will be planar. The graphs shown in Figures 2.3(d)-(h) are examples of planar graphs.
5. Star graph. This type of graph has one central vertex that is connected to all other vertices, with no other edges present. An example of star graph is given in Figure 2.3(d). Observe that a star graph can be considered as a special case of a complete
bipartite graph, with only one vertex in the first set, $\mathcal{E}$. The vertex degree centrality for the central vertex of a star graph with $N$ vertices is therefore $N-1$.
6. Circular (ring) graph. A graph is said to be circular if the degree of its every vertex is $\mathcal{J}=2$. This graph is also a regular graph with $\mathcal{J}=2$. An example of a circular graph with 8 vertices is given in Figure 2.3(e).
7. Path graph. A series of connected vertices defines a path graph, whereby the first and the last vertex are of connectivity degree $\mathcal{J}=1$, while all other vertices are of the connectivity degree $\mathcal{J}=2$. An example of a path graph with 5 vertices is presented in Figure 2.3(f).
8. Directed circular graph. A directed graph is said to be circular if each vertex is related to only one predecessor vertex and only one successor vertex. An example of a directed circular graph with 8 vertices is given in Figure 2.3(g), with the adjacency matrix

$$
\mathbf{A}=\begin{align*}
& 0  \tag{2.14}\\
& 1 \\
& 2 \\
& 3 \\
& 4 \\
& 5 \\
& 6 \\
& 7
\end{align*}\left[\begin{array}{llllllll}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7
\end{array}\right] .
$$

Remark 6: The adjacency matrix of any directed or undirected circular graph is a circulant matrix.
9. Directed path graph. A directed path graph consists of a series of vertices connected in only one direction, whereby the first and the last vertex do not have a respective predecessor or successor.

An example of a directed path graph with 5 vertices is presented in Figure 2.3(h).

Remark 7: Path and circular graphs (directed and undirected) are of particular interest in Data Analytics, since their domain properties correspond to classical time or space domains. Therefore, any graph signal processing or machine learning paradigm which is developed for path and circular graphs is equivalent to its corresponding standard time and/or spatial domain paradigm.
10. Erdös-Renyi graph model. This is an $N$-vertex graph model, denoted by $\mathcal{G}(N, p)$ and introduced by Gilbert, which is formed in such a way that the presence of an edge between any two vertices $m$ and $n$ is designated with a probability $p$. Since the number of edges in a complete graph is $N(N-1) / 2$, the expected number of edges in this graph model is $p N(N-1) / 2$. A variant of this model, denoted by $\mathcal{G}(N, M)$, is obtained when exactly $M$ randomly chosen edges are used in a graph with $N$ vertices.
These two closely related graph models are commonly used within probabilistic approaches, for example, to demonstrate that a certain property holds for almost all graphs.
11. Stochastic block graph model. Here, the $N$ vertices of a graph are grouped into $K$ communities, each comprising sets of vertices that behave similarly (we shall later refer to these groups as clusters of vertices). The vertices are then randomly connected with edges, typically with denser connections within one community, than between the different communities. The probabilities of the existence of an edge connection between the community $i$ and the community $k$ are denoted by $p_{i k}$, where $i, k=1,2, \ldots, K$ are the community indices; this means that commonly $p_{k k}>p_{i k}$ for $i \neq k$. If $p_{i k}$ is constant, then this model reduces to a special case of the Erdös-Renyi model, since there is no inter-community preference on the probability for edge existence.
12. Preferential attachment model. In graphs that model realworld social and other networks, it is not uncommon that the
number of vertices (representing the users) increases over time. Consider a graph with $N$ vertices, and assume that a new, $(N+1)$ th vertex, is added. In the preferential attachment graph model, this new vertex, $(N+1)$, is connected with other vertices, $n$, with a probability proportional to their degrees, $p_{n}=$ $D_{n n} / \sum_{m=1}^{N} D_{m m}$, calculated before the new vertex is added. In this way, the more connected vertices accumulate more new edges (connections).

### 2.3 Properties of Graphs and Associated Matrices

The notions from graph analysis that are most relevant to the processing of data on graphs are as follows.
$M_{1}$ : Symmetry: For an undirected graph, the matrices $\mathbf{A}, \mathbf{W}$, and $\mathbf{L}$ are all symmetric.
$M_{2}$ : A walk between a vertex $m$ and a vertex $n$ is a connected sequence of edges and vertices that begins at the vertex $m$ and ends at the vertex $n$. Edges and vertices can be included in a walk more than once. There is also more than one walk between vertices $m$ and $n$. The length of a walk is equal to the number of included edges in unweighted graphs. The number of walks of length $K$, between a vertex $m$ and a vertex $n$, is equal to the value of the $m n$-th element of the matrix $\mathbf{A}^{K}$, which can be proved through mathematical induction, as follows (Duncan, 2004).
(i) The elements, $A_{m n}$, of the adjacency matrix $\mathbf{A}$, by definition, indicate the existence of a walk of length $K=1$ (an edge, in this case) between the vertices $m$ and $n$ in a graph.
(ii) Assume that the elements of matrix $\mathbf{A}^{K-1}$ are equal to the number of walks of length $K-1$, between two arbitrary vertices $m$ and $n$.
(iii) The number of walks of length $K$ between two vertices, $m$ and $n$, is then equal to the number of all walks of length $K-1$, between the vertex $m$ and an intermediate vertex $s, s \in \mathcal{V}$, which itself is indicated by the element at the position $m s$ of the matrix
$\mathbf{A}^{K-1}$, according to (ii), for all $s$ for which there is an edge from vertex $s$ to the destination vertex $n$. If an edge between the intermediate vertex $s$ and the final vertex $n$ exists, then $A_{s n}=1$. This means that the number of walks of length $K$ between the vertices $m$ and $n$ is obtained as the inner product of the $m$-th row of $\mathbf{A}^{K-1}$ with the $n$-th column in $\mathbf{A}$, to yield the element $m n$ of matrix $\mathbf{A}^{K-1} \mathbf{A}=\mathbf{A}^{K}$.

Example 2: Consider the vertex 0 and the vertex 4 in the graph from Figure 2.4, and only the walks of length $K=2$. The adjacency matrix for this graph is given in (2.1). There are two such walks ( $0 \rightarrow 1 \rightarrow 4$ and $0 \rightarrow 2 \rightarrow 4$ ), so that the element $A_{04}^{2}$ in the first row and the fifth column of matrix $\mathbf{A}^{2}$, is equal to 2 , as designated in bold font in the matrix $\mathbf{A}^{2}$ below,

$$
\mathbf{A}^{2}=\begin{align*}
& 0  \tag{2.15}\\
& 1 \\
& 2 \\
& 3 \\
& 4 \\
& 5 \\
& 6 \\
& 7
\end{align*}\left[\begin{array}{llllllll}
3 & 1 & 2 & 1 & \mathbf{2} & 0 & 1 & 0 \\
1 & 3 & 2 & 2 & 1 & 1 & 0 & 1 \\
2 & 2 & 4 & 1 & 1 & 1 & 1 & 1 \\
1 & 2 & 1 & 3 & 1 & 0 & 0 & 1 \\
2 & 1 & 1 & 1 & 4 & 1 & 1 & 1 \\
0 & 1 & 1 & 0 & 1 & 2 & 1 & 1 \\
1 & 0 & 1 & 0 & 1 & 1 & 2 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 & 3
\end{array}\right],
$$

thus indicating $K=2$ walks between these vertices.
$M_{3}$ : The number of walks of length not higher than $K$, between the vertices $m$ and $n$, is given by the $m n$-th element of the matrix

$$
\begin{equation*}
\mathbf{B}_{K}=\mathbf{A}+\mathbf{A}^{2}+\cdots+\mathbf{A}^{K}, \tag{2.16}
\end{equation*}
$$

that is, by a value in its $m$-th row and $n$-th column. In other words, the total number of walks is equal to the sum of all walks, which are individually modeled by $\mathbf{A}^{k}, k=1,2, \ldots, K$, as stated in property $M_{2}$.
$M_{4}$ : The $K$-neighborhood of a vertex is defined as a set of vertices that are reachable from this vertex in walks whose length is up to $K$.


Figure 2.4: Walks of length $K=2$ from vertex 0 to vertex 4 (thick blue and brown lines).

For a vertex $m$, based on the property $M_{3}$, the $K$-neighborhood is designated by the positions and the numbers of non-zero elements in the $m$-th row of matrix $\mathbf{B}_{K}$ in (2.16). The $K$-neighborhoods of vertex 0 for $K=1$ and $K=2$ are illustrated in Figure 2.5.
$M_{5}$ : A path is a special kind of walk whereby each vertex can be included only once, whereby the number of edges included in a path is referred to as the path cardinality or path length, while the path weight is defined as the sum of weights along these edges.

An Euler path is a graph path that uses every edge of a graph exactly once. An Euler path for an unweighted graph does exist if and only if at most two of its vertices are of an odd degree. An Euler path which starts and ends at the same vertex is referred to as an Euler circuit, and it exists if and only if the degree of every vertex is even.

A Hamiltonian path is a graph path between two vertices of a graph that visits each vertex in a graph exactly once, while a cycle that uses every vertex in a graph exactly once is called a Hamiltonian cycle.
$M_{6}$ : The distance, $r_{m n}$, between two vertices $m$ and $n$ in an unweighed graph is equal to the minimum path length between these two vertices. For example, for the graph in Figure 2.4, the distance between vertex 1 and vertex 5 is $r_{15}=2$.

(a)

(a)

Figure 2.5: The $K$-neighborhoods of vertex 0 for the graph from Figure 2.4, where: (a) $K=1$ and (b) $K=2$. The neighboring vertices are shaded.
$M_{7}$ : The diameter, $d$, of a graph is equal to the largest distance (number of edges) between all pairs of its vertices, that is, $d=$ $\max _{m, n \in \mathcal{V}} r_{m n}$. For example, the diameter of a complete graph is $d=1$, while the diameter of the graph in Figure 2.4 is $d=3$, with one of the longest paths being $6 \rightarrow 3 \rightarrow 2 \rightarrow 1$.
$M_{8}$ : Vertex closeness centrality. The farness (remoteness) of a vertex is equal the sum of its distances to all other vertices, $f_{n}=\sum_{m \neq n} r_{n m}$. The vertex closeness is defined then as an inverse to the farness, $c_{n}=1 / f_{n}$, and can be interpreted as a measure of how long it will take for data to sequentially shift from the considered vertex to all other vertices. For example, the vertex farness and closeness
for the vertices $n=2$ and $n=5$ in Figure 2.1(a) are respectively $f_{2}=10, f_{5}=14$, and $c_{2}=0.1, c_{5}=0.071$.
$M_{9}$ : Vertex or edge betweenness. Vertex/edge betweenness of a vertex $n$ or edge $(m, n)$ is equal to the number of times that this vertex/edge acts as a bridge along the shortest paths between any other two vertices.
$M_{10}$ : Spanning tree and minimum spanning tree. The spanning tree of a graph is a subgraph that is tree-shaped and connects all its vertices together. A tree does not have cycles and cannot be disconnected. The cost of the spanning tree represents the sum of the weights of all edges in the tree. The minimum spanning tree is a spanning tree for which the cost is minimum among all possible spanning trees in a graph. Spanning trees are typically used in graph clustering analysis.

In the classical literature on graph theory, it is commonly assumed that the values of edge weights in weighted graphs are proportional to the standard vertex distance, $r_{m n}$. However, this is not the case in data analytics on graphs, where the edge weights are typically defined as a function of vertex distance, for example, through a Gaussian kernel, $W_{m n} \sim \exp \left(-r_{m n}^{2}\right)$, or some other data similarity metric. The cost function to minimize for the Minimum Spanning Tree (MST) can then be defined as a log-sum of distances, $r_{m n}=-2 \ln W_{m n}$. A spanning tree for the graph from Figure 2.2 is shown in Figure 2.6. The cost for this spanning tree, calculated as a sum of all distances (log-weights), $r_{m n}$, is 15.67 .
$M_{11}$ : An undirected graph is called connected if there exists a walk between each pair of its vertices.
$M_{12}$ : If the graph is not connected, then it consists of two or more disjoint but locally connected subgraphs (graph components). Back to mathematical formalism, such disjoint graphs impose a blockdiagonal form on the adjacency matrix, A, and the Laplacian, $\mathbf{L}$. For $M$ disjoint components (subgraphs) of a graph, these matrices


Figure 2.6: Concept of the spanning tree for graphs. (a) A spanning tree for the unweighted graph from Figure 2.1(a). (b) A spanning tree for the weighted graph from Figure 2.2, designated by thick blue edges. The graph edges in thin blue lines are not included in this spanning tree.
take the form

$$
\begin{align*}
& \mathbf{A}=\left[\begin{array}{cccc}
\mathbf{A}_{1} & \mathbf{0} & \cdots & \mathbf{0} \\
\mathbf{0} & \mathbf{A}_{2} & \cdots & \mathbf{0} \\
\vdots & \vdots & \ddots & \vdots \\
\mathbf{0} & \mathbf{0} & \cdots & \mathbf{A}_{M}
\end{array}\right]  \tag{2.17}\\
& \mathbf{L}=\left[\begin{array}{cccc}
\mathbf{L}_{1} & \mathbf{0} & \cdots & \mathbf{0} \\
\mathbf{0} & \mathbf{L}_{2} & \cdots & \mathbf{0} \\
\vdots & \vdots & \ddots & \vdots \\
\mathbf{0} & \mathbf{0} & \cdots & \mathbf{L}_{M}
\end{array}\right] . \tag{2.18}
\end{align*}
$$

Note that this block diagonal form is obtained only if the vertex numbering follows the subgraph structure.


Figure 2.7: A disconnected graph which consists of two sub-graphs.

Example 3: Consider a graph derived from Figure 2.1(a) by removing some edges, as shown in Figure 2.7. The adjacency matrix for this graph is given by

$$
\mathbf{A}=\begin{gather*}
0  \tag{2.19}\\
1 \\
2 \\
3 \\
4 \\
5 \\
6 \\
7
\end{gather*}\left[\begin{array}{cccc:cccc}
0 & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 & 0 & 0 \\
\mathbf{1} & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 \\
\mathbf{1} & \mathbf{1} & 0 & \mathbf{1} & 0 & 0 & 0 & 0 \\
\mathbf{1} & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 \\
\hdashline 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & \mathbf{1} \\
0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & \mathbf{1} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} \\
0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0
\end{array}\right]
$$

with the corresponding Laplacian in the form

$$
\mathbf{L}=\left[\begin{array}{rrrr:rrrr}
3 & -1 & -1 & -1 & 0 & 0 & 0 & 0  \tag{2.20}\\
-1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 \\
-1 & -1 & 3 & -1 & 0 & 0 & 0 & 0 \\
-1 & 0 & -1 & 2 & 0 & 0 & 0 & 0 \\
\hdashline 0 & 0 & 0 & 0 & 2 & -1 & 0 & -1 \\
0 & 0 & 0 & 0 & -1 & 2 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\
0 & 0 & 0 & 0 & -1 & -1 & -1 & 3
\end{array}\right] .
$$

Observe that, as elaborated above, these matrices are in a blockdiagonal form with the two constituent blocks clearly separated.

Therefore, for an isolated vertex in a graph, the corresponding row and column of the matrices $\mathbf{A}$ and $\mathbf{L}$ will be zero-valued.
$M_{13}$ : For two graphs defined on the same set of vertices, with the corresponding adjacency matrices $\mathbf{A}_{1}$ and $\mathbf{A}_{2}$, the summation operator produces a new graph, for which the adjacency matrix is given by

$$
\mathbf{A}=\mathbf{A}_{1}+\mathbf{A}_{2}
$$

To maintain the binary values in the resultant adjacency matrix, $A_{m n} \in\{0,1\}$, a logical (Boolean) summation rule, e.g., $1+1=1$, may be used for matrix addition. In this monograph, the arithmetic summation rule is assumed in data analytics algorithms, as for example, in Equation (2.16) in property $M_{3}$.
$M_{14}$ : The Kronecker (tensor) product of two disjoint graphs $\mathcal{G}_{1}=$ $\left(\mathcal{V}_{1}, \mathcal{B}_{1}\right)$ and $\mathcal{G}_{2}=\left(\mathcal{V}_{2}, \mathcal{B}_{2}\right)$ yields a new graph $\mathcal{G}=(\mathcal{V}, \mathcal{B})$ where $\mathcal{V}=\mathcal{V}_{1} \times \mathcal{V}_{2}$ is a direct product of the sets $\mathcal{V}_{1}$ and $\mathcal{V}_{2}$, and $\left(\left(n_{1}, m_{1}\right),\left(n_{2}, m_{2}\right)\right) \in \mathcal{B}$ only if $\left(n_{1}, n_{2}\right) \in \mathcal{B}_{1}$ and $\left(m_{1}, m_{2}\right) \in \mathcal{B}_{2}$.
The adjacency matrix $\mathbf{A}$ of the resulting graph $\mathcal{G}$ is then equal to the Kronecker product of the individual adjacency matrices $\mathbf{A}_{1}$ and $\mathbf{A}_{2}$, that is

$$
\mathbf{A}=\mathbf{A}_{1} \otimes \mathbf{A}_{2}
$$

An illustration of the Kronecker product for two simple graphs is given in Figure 2.8.
$M_{15}$ : The Cartesian product (graph product) of two disjoint graphs $\mathcal{G}_{1}=$ $\left(\mathcal{V}_{1}, \mathcal{B}_{1}\right)$ and $\mathcal{G}_{2}=\left(\mathcal{V}_{2}, \mathcal{B}_{2}\right)$ gives a new graph $\mathcal{G}=\mathcal{G}_{1} \square \mathcal{G}_{2}=(\mathcal{V}, \mathcal{B})$, where $\mathcal{V}=\mathcal{V}_{1} \times \mathcal{V}_{2}$ is a direct product of the sets $\mathcal{V}_{1}$ and $\mathcal{V}_{2}$, and $\left(\left(m_{1}, n_{1}\right),\left(m_{2}, n_{2}\right)\right) \in \mathcal{B}$, only if

$$
\begin{aligned}
m_{1} & =m_{2} \quad \text { and } \quad\left(n_{1}, n_{2}\right) \in \mathcal{B}_{2} \quad \text { or } \\
n_{1} & =n_{2} \quad \text { and } \quad\left(m_{1}, m_{2}\right) \in \mathcal{B}_{1} .
\end{aligned}
$$

The adjacency matrix of a Cartesian product of two graphs is then given by the Kronecker sum

$$
\mathbf{A}=\mathbf{A}_{1} \otimes \mathbf{I}_{N_{2}}+\mathbf{I}_{N_{1}} \otimes \mathbf{A}_{2} \stackrel{\text { def }}{=} \mathbf{A}_{1} \oplus \mathbf{A}_{2}
$$



Figure 2.8: Kronecker (tensor) product of two graphs.
where $\mathbf{A}_{1}$ and $\mathbf{A}_{2}$ are the respective adjacency matrices of graphs $\mathcal{G}_{1}, \mathcal{G}_{2}$, while $N_{1}$ and $N_{2}$ are the corresponding numbers of vertices in $\mathcal{G}_{1}$ and $\mathcal{G}_{2}$, with $\mathbf{I}_{N_{1}}$ and $\mathbf{I}_{N_{2}}$ being the identity matrices of orders $N_{1}$ and $N_{2}$. The Cartesian product of two simple graphs is illustrated in Figure 2.9. Notice that a Cartesian product of two


Figure 2.9: Cartesian product of two graphs.
graphs that reside in a two-dimensional space can be considered as a three-dimensional structure of vertices and edges ( $c f$. tensors Saito et al., 2018).

## 3

## Spectral Decomposition of Graph Matrices

As a prerequisite for the optimization and data analytics on graphs, we next introduce several intrinsic connections between standard linear algebraic tools and graph topology (Bapat, 1996; Brouwer and Haemers, 2012; Chung, 1997; Cvetković et al., 1980; Fujiwara, 1995; Jones, 2013; Maheswari and Maheswari, 2016; O'Rourke et al., 2016).

### 3.1 Eigenvalue Decomposition of the Adjacency Matrix

Like any other general matrix, graph description matrices can be analyzed using eigenvalue decomposition. In this sense, a column vector $\mathbf{u}$ is an eigenvector of the adjacency matrix $\mathbf{A}$ if

$$
\begin{equation*}
\mathbf{A} \mathbf{u}=\lambda \mathbf{u} \tag{3.1}
\end{equation*}
$$

where the constant $\lambda$, that corresponds to the eigenvector $\mathbf{u}$, is called the eigenvalue.

The above relation can be equally written as $(\mathbf{A}-\lambda \mathbf{I}) \mathbf{u}=\mathbf{0}$, and a nontrivial solution for $\mathbf{u}$ does exist if

$$
\operatorname{det}|\mathbf{A}-\lambda \mathbf{I}|=0 .
$$

In other words, the problem turns into that of finding zeros of $\operatorname{det}|\mathbf{A}-\lambda \mathbf{I}|$ as roots of a polynomial in $\lambda$, called the characteristic polynomial of
matrix $\mathbf{A}$, which is given by

$$
\begin{equation*}
P(\lambda)=\operatorname{det}|\mathbf{A}-\lambda \mathbf{I}|=\lambda^{N}+c_{1} \lambda^{N-1}+c_{2} \lambda^{N-2}+\cdots+c_{N} \tag{3.2}
\end{equation*}
$$

Remark 8: The order of the characteristic polynomial of graphs has the physical meaning of the number of vertices, $N$, within a graph while the eigenvalues represent the roots of the characteristic polynomial, that is, $P(\lambda)=0$.

In general, for a graph with $N$ vertices, its adjacency matrix has $N$ eigenvalues, $\lambda_{0}, \lambda_{1}, \ldots, \lambda_{N-1}$. Some eigenvalues may also be repeated, which indicates that zeros of algebraic multiplicity higher than one exist in the characteristic polynomial. The total number of roots of a characteristic polynomial, including their multiplicities, must be equal to its degree, $N$, whereby

- the algebraic multiplicity of an eigenvalue, $\lambda_{k}$, is equal to its multiplicity when considered as a root of the characteristic polynomial;
- the geometric multiplicity of an eigenvalue, $\lambda_{k}$, represents the number of linearly independent eigenvectors that can be associated with this eigenvalue.

The geometric multiplicity of an eigenvalue is always equal or lower than its algebraic multiplicity.

Denote the distinct eigenvalues in (3.2) by $\mu_{1}, \mu_{2}, \ldots, \mu_{N_{m}}$, and their corresponding algebraic multiplicities by $p_{1}, p_{2}, \ldots, p_{N_{m}}$, where $p_{1}+p_{2}+$ $\cdots+p_{N_{m}}=N$ is equal to the order of the considered matrix/polynomial and $N_{m} \leq N$ is the number of distinct eigenvalues. The characteristic polynomial can now be rewritten in the form

$$
P(\lambda)=\left(\lambda-\mu_{1}\right)^{p_{1}}\left(\lambda-\mu_{2}\right)^{p_{2}} \cdots\left(\lambda-\mu_{N_{m}}\right)^{p_{N_{m}}} .
$$

Definition: The minimal polynomial of the considered adjacency matrix, $\mathbf{A}$, is obtained from its characteristic polynomial by reducing the algebraic multiplicities of all eigenvalues to unity, and has the form

$$
P_{\min }(\lambda)=\left(\lambda-\mu_{1}\right)\left(\lambda-\mu_{2}\right) \cdots\left(\lambda-\mu_{N_{m}}\right)
$$

### 3.1. 1 Properties of the Characteristic and Minimal Polynomial

$P_{1}$ : The degree of the characteristic polynomial is equal to the number of vertices in the considered graph.
$P_{2}$ : For $\lambda=0, P(0)=\operatorname{det}(\mathbf{A})=-\lambda_{0}\left(-\lambda_{1}\right) \cdots\left(-\lambda_{N-1}\right)$.
$P_{3}$ : The sum of all the eigenvalues is equal to the sum of the diagonal elements of the adjacency matrix, $\mathbf{A}$, that is, its trace, $\operatorname{tr}\{\mathbf{A}\}$. For the characteristic polynomial of the adjacency matrix, $P(\lambda)$, this means that the value of $c_{1}$ in (3.2) is $c_{1}=\operatorname{tr}\{\mathbf{A}\}=0$.
$P_{4}$ : The coefficient $c_{2}$ in $P(\lambda)$ in (3.2) is equal to the number of edges multiplied by -1 .

This property, together with $P_{3}$, follows from the Faddeev-LeVerrier algorithm to calculate the coefficients of the characteristic polynomial of a square matrix, $\mathbf{A}$, as $c_{1}=-\operatorname{tr}\{\mathbf{A}\}, c_{2}=$ $-\frac{1}{2}\left(\operatorname{tr}\left\{\mathbf{A}^{2}\right\}-(\operatorname{tr}\{\mathbf{A}\})^{2}\right)$, and so on. Since $\operatorname{tr}\{\mathbf{A}\}=0$ and the diagonal elements of $\mathbf{A}^{2}$ are equal to the number of edges connected to each vertex (vertex degree), the total number of edges is equal to $\operatorname{tr}\left\{\mathbf{A}^{2}\right\} / 2=-c_{2}$.
$P_{5}$ : The degree of the minimal polynomial, $N_{m}$, is strictly larger than the graph diameter, $d$.

Example 4: Consider a connected graph with $N$ vertices and only two distinct eigenvalues, $\lambda_{0}$ and $\lambda_{1}$. The order of minimal polynomial is then $N_{m}=2$, while the diameter of this graph is $d=1$, which indicates a complete graph.

Example 5: For the graph from Figure 2.1(a), the characteristic polynomial of its adjacency matrix, $\mathbf{A}$, defined in (2.1), is given by

$$
P(\lambda)=\lambda^{8}-12 \lambda^{6}-8 \lambda^{5}+36 \lambda^{4}+36 \lambda^{3}-22 \lambda^{2}-32 \lambda-8,
$$

with the eigenvalues

$$
\lambda \in\{-2,-1.741,-1.285,-0.677,-0.411,1.114,1.809,3.190\}
$$

With all the eigenvalues different, the minimal polynomial is equal to the characteristic polynomial, $P_{\min }(\lambda)=P(\lambda)$.

Example 6: The adjacency matrix for the disconnected graph from Figure 2.7 is given in (2.19), and its characteristic polynomial has the form

$$
P(\lambda)=\lambda^{8}-9 \lambda^{6}-6 \lambda^{5}+21 \lambda^{4}+26 \lambda^{3}+3 \lambda^{2}-4 \lambda
$$

with the eigenvalues

$$
\lambda \in\{-1.5616,-1.4812,-1,-1,0,0.3111,2.1701,2.5616\}
$$

Observe that the eigenvalue $\lambda=-1$ is of multiplicity higher than 1 (multiplicity of 2 ), so that the corresponding minimal polynomial becomes

$$
P_{\min }(\lambda)=\lambda^{7}-\lambda^{6}-8 \lambda^{5}+2 \lambda^{4}+19 \lambda^{3}+7 \lambda^{2}-4 \lambda
$$

Although this graph is disconnected, the largest eigenvalue of its adjacency matrix, $\lambda_{\max }=2.5616$, is of multiplicity 1 . Relation between the graph connectivity and the multiplicity of eigenvalues will be discussed later.

### 3.2 Spectral Graph Theory

If all the eigenvalues of $\mathbf{A}$ are distinct (of algebraic multiplicity 1 ), then the $N$ equations in the eigenvalue problem in (3.1), that is, $\mathbf{A} \mathbf{u}_{k}=\lambda_{k} \mathbf{u}_{k}$, $k=0,1, \ldots, N-1$, can be written in a compact form as one matrix equation with respect to the adjacency matrix, as

$$
\mathbf{A U}=\mathbf{U} \mathbf{\Lambda}
$$

or

$$
\begin{equation*}
\mathbf{A}=\mathbf{U} \mathbf{\Lambda} \mathbf{U}^{-1} \tag{3.3}
\end{equation*}
$$

where $\boldsymbol{\Lambda}=\operatorname{diag}\left(\lambda_{0}, \lambda_{1}, \ldots, \lambda_{N-1}\right)$ is the diagonal matrix with the eigenvalues on its diagonal and $\mathbf{U}$ is a matrix composed of the eigenvectors, $\mathbf{u}_{k}$, as its columns. Since the eigenvectors, $\mathbf{u}$, are obtained by solving a homogeneous system of equations, defined by (3.1) and in the form $(\mathbf{A}-\lambda \mathbf{I}) \mathbf{u}=\mathbf{0}$, one element of the eigenvector $\mathbf{u}$ can be arbitrarily
chosen. The common choice is to enforce unit energy, $\left\|\mathbf{u}_{k}\right\|_{2}^{2}=1$, for every $k=0,1, \ldots, N-1$.

Remark 9: For an undirected graph, the adjacency matrix $\mathbf{A}$ is symmetric, that is $\mathbf{A}=\mathbf{A}^{T}$. Any symmetric matrix (i) has real-valued eigenvalues; (ii) is diagonalizable; and (iii) has orthogonal eigenvectors, and hence

$$
\mathbf{U}^{-1}=\mathbf{U}^{T}
$$

Remark 10: For directed graphs, in general, $\mathbf{A} \neq \mathbf{A}^{T}$.
Recall that a square matrix is diagonalizable if all its eigenvalues are distinct (this condition is sufficient, but not necessary) or if the algebraic multiplicity of each eigenvalue is equal to its geometrical multiplicity.

For some directed graphs, the eigenvalues of their adjacency matrix may be with algebraic multiplicity higher than one, and the matrix $\mathbf{A}$ may not be diagonalizable. In such cases, the algebraic multiplicity of the considered eigenvalue is higher than its geometric multiplicity and the Jordan normal form may be used in decomposition.

Definition: The set of the eigenvalues of an adjacency matrix is called the graph adjacency spectrum.

Remark 11: The spectral theory of graphs studies properties of graphs through the eigenvalues and eigenvectors of their associated adjacency and graph Laplacian matrices.
Example 7: For the graph presented in Figure 2.1(a), the graph adjacency spectrum is given by $\lambda \in\{-2,-1.741,-1.285,-0.677,-0.411$, $1.114,1.809,3.190\}$, and is shown in Figure 3.1(top).

Example 8: The vertices of the graph presented in Figure 2.1(a) are randomly reordered, as shown in Figure 3.2. Observe that the graph adjacency spectrum, given in the same figure, retains the same values, with vertex indices of the eigenvectors reordered in the same way as the graph vertices, while the eigenvalues (spectra) retain the same order as in the original graph in Figure 3.1. By a simple inspection we see that, for example, the eigenvector elements at the vertex index position $n=0$ in Figure 3.1 are now at the vertex index position $n=3$ in all eigenvectors in Figure 3.2.


Figure 3.1: Eigenvalues, $\lambda_{k}$, for spectral indices (eigenvalue numbers) $k=$ $0,1, \ldots, N-1$, and elements of the corresponding eigenvectors, $u_{k}(n)$, as a function of the vertex index $n=0,1, \ldots, N-1$, for the adjacency matrix, $\mathbf{A}$, of the undirected graph presented in Figure 2.1(a). The distinct eigenvectors are shown both on the vertex index axis, $n$, (left) and on the graph itself (right).


Figure 3.2: Eigenvalues, $\lambda_{k}$, for spectral indices (eigenvalue numbers) $k=$ $0,1, \ldots, N-1$, and elements of the corresponding eigenvectors, $u_{k}(n)$, as a function of the vertex index $n=0,1, \ldots, N-1$, for the adjacency matrix, $\mathbf{A}$, of the undirected graph presented in Figure 2.1(a) with index reordering according to the scheme $[0,1,2,3,4,5,6,7] \rightarrow[3,2,4,5,1,0,6,7]$. The distinct eigenvectors are shown both on the vertex index axis, $n$, (left) and on the graph itself (right). Compare with the results for the original vertex ordering in Figure 3.1.

Remark 12: A unique feature of graphs is that vertex reindexing does not alter the eigenvalues of the adjacency matrix, while the corresponding eigenvectors of the reindexed adjacency matrix contain the same elements as the original eigenvectors, but reordered according to the vertex renumbering. This follows from the properties of the permutation matrix, as in Equation (2.3).

### 3.2.1 The DFT Basis Functions as a Special Case of Eigenvectors of the Adjacency Matrix

For continuity with standard spectral analysis, we shall first consider directed circular graphs, as this graph topology encodes the standard time and space domains.

Eigenvalue decomposition for the directed circular graph in Figure $2.3(\mathrm{~g})$, assuming $N$ vertices, follows from the definition $\mathbf{A u}_{k}=$ $\lambda_{k} \mathbf{u}_{k}$, and the form of the adjacency matrix in (2.14). Then, the elements of vector $\mathbf{A} \mathbf{u}_{k}$ are $u_{k}(n-1)$, as effectively matrix $\mathbf{A}$ here represents a shift operator, while the elements of vector $\lambda_{k} \mathbf{u}_{k}$ are $\lambda_{k} u_{k}(n)$, to give

$$
\begin{equation*}
u_{k}(n-1)=\lambda_{k} u_{k}(n) \tag{3.4}
\end{equation*}
$$

where $u_{k}(n)$ are the elements of the eigenvector $\mathbf{u}_{k}$ for given vertex indices $n=0,1, \ldots, N-1$, and $k$ is the index of an eigenvector, $k=$ $0,1, \ldots, N-1$. This is a first-order linear difference equation, whose general form for a discrete signal $x(n)$ is $x(n)=a x(n-1)$, for which the solution is

$$
\begin{equation*}
u_{k}(n)=\frac{1}{\sqrt{N}} e^{j 2 \pi n k / N} \quad \text { and } \quad \lambda_{k}=e^{-j 2 \pi k / N} \tag{3.5}
\end{equation*}
$$

with $k=0,1, \ldots, N-1$. It is straightforward to verify that this solution satisfies the difference equation (3.4). Since the considered graph is circular, the eigenvectors also exhibit circular behavior, that is, $u_{k}(n)=$ $u_{k}(n+N)$. For convenience, a unit energy condition is used to find the constants within the general solution of this first-order linear difference equation. Observe that the eigenvectors in (3.5) correspond exactly to the standard harmonic basis functions in DFT.

Remark 13: Classic DFT analysis may be obtained as a special case of the graph spectral analysis in (3.5), when considering directed circular
graphs. Observe that for circular graphs, the adjacency matrix plays the role of a shift operator, as seen in (3.4), with the elements of $\mathrm{Au}_{k}$ equal to $u_{k}(n-1)$. This property will be used to define the shift operator on a graph in the following sections.

### 3.2.2 Decomposition of Graph Product Adjacency Matrices

We have already seen in Figures 2.8 and 2.9 that complex graphs, for example those with a three-dimensional vertex space, may be obtained as a Kronecker (tensor) product or a Cartesian (graph) product of two disjoint graphs $\mathcal{G}_{1}$ and $\mathcal{G}_{2}$. Their respective adjacency matrices, $\mathbf{A}_{1}$ and $\mathbf{A}_{2}$, are correspondingly combined into the adjacency matrices of the Kronecker graph product, $\mathbf{A}_{\otimes}=\mathbf{A}_{1} \otimes \mathbf{A}_{2}$ and the Cartesian graph product, $\mathbf{A}_{\oplus}=\mathbf{A}_{1} \oplus \mathbf{A}_{2}$, as described in properties $M_{14}$ and $M_{15}$.

Graph Kronecker product. For the eigendecomposition of the Kronecker product of matrices $\mathbf{A}_{1}$ and $\mathbf{A}_{2}$, the following holds

$$
\begin{aligned}
\mathbf{A}_{\otimes} & =\mathbf{A}_{1} \otimes \mathbf{A}_{2}=\left(\mathbf{U}_{1} \boldsymbol{\Lambda}_{1} \mathbf{U}_{1}^{H}\right) \otimes\left(\mathbf{U}_{2} \boldsymbol{\Lambda}_{2} \mathbf{U}_{2}^{H}\right) \\
& =\left(\mathbf{U}_{1} \otimes \mathbf{U}_{2}\right)\left(\mathbf{\Lambda}_{1} \otimes \boldsymbol{\Lambda}_{2}\right)\left(\mathbf{U}_{1} \otimes \mathbf{U}_{2}\right)^{H}
\end{aligned}
$$

or in other words, the eigenvectors of the adjacency matrix of the Kronecker product of graphs are obtained by a Kronecker product of the eigenvectors of the adjacency matrices of individual graphs, as $\mathbf{u}_{k+l N_{1}}=\mathbf{u}_{k}^{\left(A_{1}\right)} \otimes \mathbf{u}_{l}^{\left(A_{2}\right)}, k=0,1,2, \ldots, N_{1}-1, l=0,1,2, \ldots, N_{2}-1$.
Remark 14: The eigenvectors of the individual graph adjacency matrices, $\mathbf{u}_{k}^{\left(A_{1}\right)}$ and $\mathbf{u}_{k}^{\left(A_{2}\right)}$, are of much lower dimensionality than those of the adjacency matrix of the resulting graph Kronecker product. This property can be used to reduce computational complexity when analyzing data observed on this kind of graph. Recall that the eigenvalues of the resulting graph adjacency matrix are equal to the product of the eigenvalues of adjacency matrices of the constituent graphs, $\mathcal{G}_{2}$ and $\mathcal{G}_{2}$, that is,

$$
\lambda_{k+l N_{1}}=\lambda_{k}^{\left(A_{1}\right)} \lambda_{l}^{\left(A_{2}\right)}
$$

Graph Cartesian product. The eigendecomposition of the adjacency matrix of the Cartesian product of graphs, whose respective adjacency
matrices are $\mathbf{A}_{1}$ and $\mathbf{A}_{2}$, is of the form

$$
\begin{equation*}
\mathbf{A}_{\oplus}=\mathbf{A}_{1} \oplus \mathbf{A}_{2}=\left(\mathbf{U}_{1} \otimes \mathbf{U}_{2}\right)\left(\boldsymbol{\Lambda}_{1} \oplus \boldsymbol{\Lambda}_{2}\right)\left(\mathbf{U}_{1} \otimes \mathbf{U}_{2}\right)^{H} \tag{3.6}
\end{equation*}
$$

with $\mathbf{u}_{k}=\mathbf{u}_{k}^{\left(A_{1}\right)} \otimes \mathbf{u}_{k}^{\left(A_{2}\right)}$ and $\lambda_{k+l N_{1}}=\lambda_{k}^{\left(A_{1}\right)}+\lambda_{l}^{\left(A_{2}\right)}, k=0,1,2, \ldots$, $N_{1}-1, l=0,1,2, \ldots, N_{2}-1$ (Barik et al., 2015).
Remark 15: The Kronecker product and the Cartesian product of graphs share the same eigenvectors, while their spectra (eigenvalues) are different.
Example 9: The basis functions of classic two-dimensional (image) 2D-DFT follow from the spectral analysis of a Cartesian graph product which is obtained as a product the circular directed graph from Figure 2.3 with itself. Since from (3.5), the eigenvector elements of each graph are $u_{k}(n)=e^{j 2 \pi n k / N} / \sqrt{N}$, then the elements of the resulting basis functions are given by

$$
u_{k+l N}(m+n N)=\frac{1}{N} e^{j 2 \pi m k / N} e^{j 2 \pi n l / N}
$$

for $k=0,1, \ldots, N-1, l=0,1, \ldots, N-1, m=0,1, \ldots, N-1$, and $n=0,1, \ldots, N-1$. Figure 3.3 illustrates the Cartesian product of two circular undirected graphs with $N_{1}=N_{2}=8$.

Remark 16: Cartesian products of graphs may be used for multidimensional extensions of vertex spaces and graph data domains, whereby


Figure 3.3: Graph Cartesian product of two planar circular unweighted graphs, with $N=8$ vertices, produces a three-dimensional torus topology.
the resulting eigenvectors (basis functions) can be efficiently calculated using the eigenvectors of the original graphs, which are of lower dimensionality.

### 3.2.3 Decomposition of Matrix Powers and Polynomials

From the eigendecomposition of the adjacency matrix $\mathbf{A}$ in (3.3), eigenvalue decomposition of the squared adjacency matrix, $\mathbf{A} \mathbf{A}=\mathbf{A}^{2}$, is given by

$$
\mathbf{A}^{2}=\mathbf{U} \boldsymbol{\Lambda} \mathbf{U}^{-1} \mathbf{U} \boldsymbol{\Lambda} \mathbf{U}^{-1}=\mathbf{U} \boldsymbol{\Lambda}^{2} \mathbf{U}^{-1}
$$

under the assumption that $\mathbf{U}^{-1}$ exists. For an arbitrary natural number, $m$, the above result generalizes straightforwardly to

$$
\begin{equation*}
\mathbf{A}^{m}=\mathbf{U} \mathbf{\Lambda}^{m} \mathbf{U}^{-1} \tag{3.7}
\end{equation*}
$$

Further, for any matrix function, $f(\mathbf{A})$, that can be written in a polynomial form, given by

$$
f(\mathbf{A})=h_{0} \mathbf{A}^{0}+h_{1} \mathbf{A}^{1}+h_{2} \mathbf{A}^{2}+\cdots+h_{N-1} \mathbf{A}^{N-1}
$$

its eigenvalue decomposition is, in general, given by

$$
f(\mathbf{A})=\mathbf{U} f(\boldsymbol{\Lambda}) \mathbf{U}^{-1}
$$

This is self-evident from the properties of eigendecomposition of matrix powers, defined in (3.7), and the linearity of the matrix multiplication operator, $\mathbf{U}\left(h_{0} \mathbf{A}^{0}+h_{1} \mathbf{A}^{1}+h_{2} \mathbf{A}^{2}+\cdots+h_{N-1} \mathbf{A}^{N-1}\right) \mathbf{U}^{-1}$.

### 3.3 Eigenvalue Decomposition of the Graph Laplacian

Spectral analysis for graphs can also be performed based on the graph Laplacian, $\mathbf{L}$, defined in (2.7). For convenience, we here adopt the same notation for the eigenvalues and eigenvectors of the graph Laplacian, as we did for the adjacency matrix $\mathbf{A}$, although the respective eigenvalues and eigenvectors are not directly related. The Laplacian of an undirected graph can be therefore written as

$$
\mathbf{L}=\mathbf{U} \boldsymbol{\Lambda} \mathbf{U}^{T} \quad \text { or } \quad \mathbf{L} \mathbf{U}=\mathbf{U} \mathbf{\Lambda}
$$

where $\boldsymbol{\Lambda}=\operatorname{diag}\left(\lambda_{0}, \lambda_{1}, \ldots, \lambda_{N-1}\right)$ is a diagonal matrix of Laplacian eigenvalues and $\mathbf{U}$ is the orthonormal matrix of its eigenvectors (in columns), with $\mathbf{U}^{-1}=\mathbf{U}^{T}$. Note that the Laplacian of an undirected graph is always diagonalizable, since its matrix $\mathbf{L}$ is real and symmetric.

Then, every eigenvector, $\mathbf{u}_{k}, k=0,1, \ldots, N-1$, of a graph Laplacian, $\mathbf{L}$, satisfies

$$
\begin{equation*}
\mathbf{L u}_{k}=\lambda_{k} \mathbf{u}_{k} \tag{3.8}
\end{equation*}
$$

Definition: The set of the eigenvalues, $\lambda_{k}, k=0,1, \ldots, N-1$, of the graph Laplacian is referred to as the graph spectrum or graph Laplacian spectrum ( $c f$. graph adjacency spectrum based on $\mathbf{A}$ ).
Example 10: The Laplacian spectrum of the undirected graph from Figure 2.2, is given by

$$
\lambda \in\{0,0.29,0.34,0.79,1.03,1.31,1.49,2.21\},
$$

and shown in Figure 3.4, along with the corresponding eigenvectors. The Laplacian spectrum of the disconnected graph from Figure 3.5, is given by

$$
\lambda \in\{0,0,0.22,0.53,0.86,1.07,1.16,2.03\},
$$

and is illustrated in Figure 3.6. The disconnected nature of this graph is indicated by the zero-valued eigenvalue of algebraic multiplicity 2 , that is, $\lambda_{0}=\lambda_{1}=0$.

Remark 17: Observe that when graph-component (sub-graph) based vertex indexing is employed, then even though the respective graph spectra for the connected graph in Figure 3.4 and the disconnected graph Figure 3.6 are similar, for a given spectral index, the eigenvectors of a disconnected graph take nonzero values on only one of the individual disconnected graph components.

### 3.3.1 Properties of Laplacian Eigenvalue Decomposition

$L_{1}$ : The graph Laplacian matrix is defined in (2.7) in such a way that the sum of elements in its each row (column) is zero. As a consequence, this enforces the inner products of every row of $\mathbf{L}$ with any constant vector, $\mathbf{u}$, to be zero-valued, that is,


Figure 3.4: Eigenvalues, $\lambda_{k}$, for spectral indices (eigenvalue number) $k=$ $0,1, \ldots, N-1$, and elements of the corresponding eigenvectors, $u_{k}(n)$, as a function of the vertex index $n=0,1, \ldots, N-1$, for the Laplacian matrix, $\mathbf{L}$, of the undirected graph presented in Figure 2.2. The distinct eigenvectors are shown both on the vertex index axis, $n$, (left) and on the graph itself (right). A comparison with the eigenvectors of the adjacency matrix in Figure 3.1, shows that for the adjacency matrix the smoothest eigenvector corresponds to the largest eigenvalue, while for the graph Laplacian the smoothest eigenvector corresponds to the smallest eigenvalue, $\lambda_{0}$.


Figure 3.5: A disconnected weighted graph which consists of two sub-graphs.
$\mathbf{L u}=\mathbf{0}=0 \cdot \mathbf{u}$, for any constant vector $\mathbf{u}$. This means that at least one eigenvalue of the Laplacian is zero, $\lambda_{0}=0$, and its corresponding constant unit energy eigenvector is given by $\mathbf{u}_{0}=[1,1, \ldots, 1]^{T} / \sqrt{N}=\mathbf{1} / \sqrt{N}$.
$L_{2}$ : The multiplicity of the eigenvalue $\lambda_{0}=0$ of the graph Laplacian is equal to the number of connected components (connected subgraphs) in the corresponding graph.
This property follows from the fact that the Laplacian matrix of disconnected graphs can be written in a block diagonal form, as in (2.18). The set of eigenvectors of a block-diagonal matrix is obtained by grouping together the sets of eigenvectors of individual block submatrices. Since each subgraph of a disconnected graph behaves as an independent graph, then for each subgraph $\lambda_{0}=0$ is the eigenvalue of the corresponding block Laplacian submatrix, according to property $L_{1}$. Therefore, the multiplicity of the eigenvalue $\lambda_{0}=0$ corresponds to the number of disjoint components (subgraphs) within a graph.

This property does not hold for the adjacency matrix, since there are no common eigenvalues in the adjacency matrices for the blocks (subgraphs) or arbitrary graphs, like in the case of $\lambda_{0}=0$ for the graph Laplacian matrix and any graph. In this sense, the graph Laplacian matrix carries more physical meaning than the corresponding adjacency matrix.


Figure 3.6: Eigenvalues, $\lambda_{k}$, for spectral indices (eigenvalue number) $k=$ $0,1, \ldots, N-1$, and elements of the corresponding eigenvectors, $u_{k}(n)$, as a function of the vertex index $n=0,1, \ldots, N-1$, for the Laplacian matrix, $\mathbf{L}$, of the undirected graph presented in Figure 3.5. The distinct eigenvectors are shown both on the vertex index axis, $n$, (left) and on the graph itself (right). This graph is characterized with the zero eigenvalue of algebraic multiplicity 2 , that is, $\lambda_{0}=\lambda_{1}=0$. Observe that for every spectral index, $k$, the corresponding eigenvectors take nonzero values on only one of the disconnected graph components.

Remark 18: If $\lambda_{0}=\lambda_{1}=0$, then the graph is not connected. If $\lambda_{2}>0$, then there are exactly two individually connected but globally disconnected components in this graph. If $\lambda_{1} \neq 0$ then this eigenvalue may be used to describe the so called algebraic connectivity of a graph, whereby very small values of $\lambda_{1}$ indicate that the graph is weakly connected. This can be used as an indicator of the possibility of graph segmentation, as elaborated in Section 4.2.3.
$L_{3}$ : As with any other matrix, the sum of the eigenvalues of the Laplacian matrix is equal to its trace. For the normalized Laplacian, the sum of its eigenvalues is equal to the number of vertices, $N$, if there are no isolated vertices.
$L_{4}$ : The coefficient, $c_{N}$, in the characteristic polynomial of the graph Laplacian matrix

$$
P(\lambda)=\operatorname{det}|\mathbf{L}-\lambda \mathbf{I}|=\lambda^{N}+c_{1} \lambda^{N-1}+\cdots+c_{N-1} \lambda+c_{N}
$$

is equal to 0 , since $\lambda=0$ is an eigenvalue for the Laplacian matrix. For unweighted graphs, the coefficient $c_{1}$ is equal to the number of edges multiplied by -2 . This is straightforward to show following the relations from property $P_{4}$ which state that $c_{1}=-\operatorname{tr}\{\mathbf{L}\}$. For unweighted graphs, the diagonal elements of the Laplacian are equal to the corresponding vertex degrees (number of edges). Therefore, the number of edges in an unweighted graph is equal to $-c_{1} / 2$.

Example 11: The characteristic polynomial of the Laplacian for the graph from Figure 2.1(a) is given by

$$
\begin{aligned}
P(\lambda)= & \lambda^{8}-24 \lambda^{7}+238 \lambda^{6}-1256 \lambda^{5}+3777 \lambda^{4} \\
& -6400 \lambda^{3}+5584 \lambda^{2}-1920 \lambda
\end{aligned}
$$

with the eigenvalues $\lambda \in\{0,1,1.4384,3,4,4,5,5.5616\}$. Observe that the eigenvalue $\lambda=4$ is of multiplicity higher than one. The minimal polynomial therefore becomes $P_{\min }(\lambda)=\lambda^{7}-20 \lambda^{6}+$ $158 \lambda^{5}-624 \lambda^{4}+1281 \lambda^{3}-1276 \lambda^{2}+480 \lambda$.

For the disconnected graph in Figure 2.7, the characteristic polynomial of the Laplacian is given by

$$
P(\lambda)=\lambda^{8}-18 \lambda^{7}+131 \lambda^{6}-490 \lambda^{5}+984 \lambda^{4}-992 \lambda^{3}+384 \lambda^{2},
$$

with the eigenvalues $\lambda \in\{0,0,1,2,3,4,4,4\}$. The eigenvalue $\lambda=0$ is of algebraic multiplicity 2 and the eigenvalue $\lambda=4$ of algebraic multiplicity 3 , so that the minimal polynomial takes the form

$$
P_{\min }(\lambda)=\lambda^{5}-10 \lambda^{4}+35 \lambda^{3}-50 \lambda^{2}+24 \lambda .
$$

Since the eigenvalue $\lambda=0$ is of algebraic multiplicity 2 , property $L_{2}$ indicates that this graph is disconnected, with two disjoint sub-graphs as its constituent components.
$L_{5}$ : Graphs with identical spectra are called isospectral or cospectral graphs. However, isospectral graphs are not necessary isomorphic, and construction of isospectral graphs that are not isomorphic is an important topic in graph theory.

Remark 19: A complete graph is uniquely determined by its Laplacian spectrum (Van Dam and Haemers, 2003). The Laplacian spectrum of a complete unweighted graph, with $N$ vertices, is $\lambda_{k} \in\{0, N, N, \ldots, N\}$. Therefore, two complete isospectral graphs are also isomorphic.
$L_{6}$ : For a $\mathcal{J}$-regular graph, as in Figure 2.3(c), the eigenvectors of the graph Laplacian and the adjacency matrices are identical, with the following relation for the eigenvalues,

$$
\lambda_{k}^{(L)}=\mathcal{J}-\lambda_{k}^{(A)}
$$

where the superscript $L$ designates the Laplacian and superscript $A$ the corresponding adjacency matrix. This follows directly from $\mathbf{U}^{T} \mathbf{L} \mathbf{U}=\mathbf{U}^{T}(\mathcal{J} \mathbf{I}-\mathbf{A}) \mathbf{U}$.
$L_{7}$ : The eigenvalues of the normalized graph Laplacian, $\mathbf{L}_{N}=\mathbf{I}-$ $\mathbf{D}^{-1 / 2} \mathbf{A} \mathbf{D}^{-1 / 2}$, are nonnegative and upper-bounded by

$$
0 \leq \lambda \leq 2
$$

The equality for the upper bound holds if and only if the graph is a bipartite graph, as in Figure 2.3(b). This will be proven within the next property.
$L_{8}$ : The eigenvalues and eigenvectors of the normalized Laplacian of a bipartite graph, with the disjoint sets of vertices $\mathcal{E}$ and $\mathcal{H}$, satisfy the relation, referred to as the graph spectrum folding, given by

$$
\begin{gather*}
\lambda_{k}=2-\lambda_{N-k}  \tag{3.9}\\
\mathbf{u}_{k}=\left[\begin{array}{c}
\mathbf{u}_{\mathcal{E}} \\
\mathbf{u}_{\mathcal{H}}
\end{array}\right] \quad \text { and } \quad \mathbf{u}_{N-k}=\left[\begin{array}{c}
\mathbf{u}_{\mathcal{E}} \\
-\mathbf{u}_{\mathcal{H}}
\end{array}\right] \tag{3.10}
\end{gather*}
$$

where $\mathbf{u}_{k}$ designates the $k$-th eigenvector of a bipartite graph, $\mathbf{u}_{\mathcal{E}}$ is its part indexed on the first set of vertices, $\mathcal{E}$, while $\mathbf{u}_{\mathcal{H}}$ is the part of the eigenvector $\mathbf{u}_{k}$ indexed on the second set of vertices, $\mathcal{H}$. In order to prove this property, we shall write the adjacency and the normalized Laplacian matrices of an undirected bipartite graph in their block forms

$$
\mathbf{A}=\left[\begin{array}{cc}
\mathbf{0} & \mathbf{A}_{\mathcal{E H}} \\
\mathbf{A}_{\mathcal{E} \mathcal{H}}^{T} & \mathbf{0}
\end{array}\right] \quad \text { and } \quad \mathbf{L}_{N}=\left[\begin{array}{cc}
\mathbf{I} & \mathbf{L}_{\mathcal{E H}} \\
\mathbf{L}_{\mathcal{E H}}^{T} & \mathbf{I}
\end{array}\right]
$$

The eigenvalue relation, $\mathbf{L}_{N} \mathbf{u}_{k}=\lambda_{k} \mathbf{u}_{k}$, can now be evaluated as

$$
\mathbf{L}_{N} \mathbf{u}_{k}=\left[\begin{array}{l}
\mathbf{u}_{\mathcal{E}}+\mathbf{L}_{\mathcal{E}} \mathbf{u}_{\mathcal{H}} \\
\mathbf{L}_{\mathcal{E} \mathcal{H}}^{T} \mathbf{u}_{\mathcal{E}}+\mathbf{u}_{\mathcal{H}}
\end{array}\right]=\lambda_{k}\left[\begin{array}{c}
\mathbf{u}_{\mathcal{E}} \\
\mathbf{u}_{\mathcal{H}}
\end{array}\right]
$$

From there, we have $\mathbf{u}_{\mathcal{E}}+\mathbf{L}_{\mathcal{E} \mathcal{H}} \mathbf{u}_{\mathcal{H}}=\lambda_{k} \mathbf{u}_{\mathcal{E}}$ and $\mathbf{L}_{\mathcal{E} \mathcal{H}}^{T} \mathbf{u}_{\mathcal{E}}+\mathbf{u}_{\mathcal{H}}=$ $\lambda_{k} \mathbf{u}_{\mathcal{H}}$, resulting in $\mathbf{L}_{\mathcal{E} \mathcal{H}} \mathbf{u}_{\mathcal{H}}=\left(\lambda_{k}-1\right) \mathbf{u}_{\mathcal{E}}$ and $\mathbf{L}_{\mathcal{E} \mathcal{H}}^{T} \mathbf{u}_{\mathcal{E}}=\left(\lambda_{k}-1\right) \mathbf{u}_{\mathcal{H}}$, to finally yield

$$
\mathbf{L}_{N}\left[\begin{array}{c}
\mathbf{u}_{\mathcal{E}} \\
-\mathbf{u}_{\mathcal{H}}
\end{array}\right]=\left(2-\lambda_{k}\right)\left[\begin{array}{c}
\mathbf{u}_{\mathcal{E}} \\
-\mathbf{u}_{\mathcal{H}}
\end{array}\right]
$$

This completes the proof.
Since for the graph Laplacian $\lambda_{0}=0$ always holds (see the property $L_{1}$ ), from $\lambda_{k}=2-\lambda_{N-k}$ in (3.9), it then follows that the largest eigenvalue is $\lambda_{N}=2$, which also proves the property $L_{7}$ for a bipartite graph.

### 3.3.2 Fourier Analysis as a Special Case of the Laplacian Spectrum

Consider the undirected circular graph from Figure 2.3(e). Then, from the property $L_{1}$, the eigendecomposition relation for the Laplacian of this graph, $\mathbf{L u}=\lambda \mathbf{u}$, admits a simple form

$$
\begin{equation*}
-u(n-1)+2 u(n)-u(n+1)=\lambda u(n) . \tag{3.11}
\end{equation*}
$$

This is straightforward to show by inspecting the Laplacian for the undirected circular graph from Figure 2.3(e), with $N=8$ vertices for which the eigenvalue analysis is based on

$$
\mathbf{L u}=\left[\begin{array}{rrrrrrrr}
2 & -1 & 0 & 0 & 0 & 0 & 0 & -1  \tag{3.12}\\
-1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 \\
-1 & 0 & 0 & 0 & 0 & 0 & -1 & 2
\end{array}\right]\left[\begin{array}{l}
u(0) \\
u(1) \\
u(2) \\
u(3) \\
u(4) \\
u(5) \\
u(6) \\
u(7)
\end{array}\right] .
$$

This directly gives the term $-u(n-1)+2 u(n)-u(n+1)$, while a simple inspection of the values $u(0)$ and $u(N)$ illustrates the circular nature of the eigenvectors; see also Remark 6. The solution to the second order difference equation in (3.11) is $u_{k}(n)=\cos \left(\frac{2 \pi k n}{N}+\phi_{k}\right)$, with $\lambda_{k}=2\left(1-\cos \left(\frac{2 \pi k}{N}\right)\right)$. Obviously, for every eigenvalue, $\lambda_{k}$ (except for $\lambda_{0}$ and for the last eigenvalue, $\lambda_{N-1}$, for an even $N$ ), we can choose to have two orthogonal eigenvectors with, for example, $\phi_{k}=0$ and $\phi_{k}=\pi / 2$. This means that most of the eigenvalues are of algebraic multiplicity 2 , i.e., $\lambda_{1}=\lambda_{2}, \lambda_{3}=\lambda_{4}$, and so on. This eigenvalue multiplicity of two can be formally expressed as

$$
\lambda_{k}= \begin{cases}2-2 \cos (\pi(k+1) / N), & \text { for odd } k=1,3,5, \ldots  \tag{3.13}\\ 2-2 \cos (\pi k / N), & \text { for even } k=2,4,6, \ldots\end{cases}
$$

For an odd $N, \lambda_{N-2}=\lambda_{N-1}$, whereas for an even $N$ we have $\lambda_{N-1}=2$ which is of algebraic multiplicity 1 .

The corresponding eigenvectors $\mathbf{u}_{0}, \mathbf{u}_{1}, \ldots, \mathbf{u}_{N-1}$, then have the form

$$
u_{k}(n)= \begin{cases}\sin (\pi(k+1) n / N), & \text { for odd } k, k<N-1  \tag{3.14}\\ \cos (\pi k n / N), & \text { for even } k \\ \cos (\pi n), & \text { for odd } k, k=N-1\end{cases}
$$

where $k=0,1, \ldots, N-1$ and $n=0,1, \ldots, N-1$.
Relation between graph Fourier analysis based on the graph adjacency matrix and graph Laplacian matrix. Recall that an arbitrary linear combination of eigenvectors $\mathbf{u}_{2 k-1}$ and $\mathbf{u}_{2 k}, 1 \leq k<$ $N / 2$, is also an eigenvector since the corresponding eigenvalues are equal (in this case both their algebraic and geometric multiplicities are equal to 2, see Equation (3.13)). With this in mind, we can rewrite the full set of eigenvectors in an alternative compact form, given by

$$
u_{k}(n)= \begin{cases}1, & \text { for } k=0 \\ e^{j \pi(k+1) n / N}=e^{j 2 \pi r n / N}, & \text { for odd } k=2 r-1, k<N-1 \\ e^{-j 2 \pi k n / N}=e^{-j 2 \pi r n / N}, & \text { for even } k=2 r, k>0 \\ \cos (\pi n), & \text { for odd } k, k=N-1\end{cases}
$$

where $j^{2}=-1$. The above eigenvectors are assumed to be normalized. It is now clear that, as desired, this set of eigenvectors is also orthonormal, so that the individual eigenvectors, $\mathbf{u}_{k}$, correspond to the harmonic basis functions within the standard temporal/spatial DFT obtained by the directed circular graph adjacency matrix decomposition.

If the vertices correspond to the pixels of a two-dimensional $N \times N$ image in a stacked-column representation, then the edge weights for a given vertex, $n$, are $w_{m n}=1, m \in\{n-N, n-1, n+1, n+N\}$, while the degree of every vertex, $n$, is equal to 4 . The corresponding graph Laplacian now becomes a discrete approximation of second-order partial derivatives, and is used as a standard tool in image processing for edge detection, while two-dimensional Fourier analysis can be defined using eigenvalue decomposition of this Laplacian. Notice that the Laplacians in the graph Cartesian product exhibit similar relations to those for the adjacency matrix in Equation (3.6) and Figure 3.3.

## 4

## Vertex Clustering and Mapping

An important task for data analytics on graphs is to identify groups of vertices which exhibit similar behavior, referred to as vertex clustering. This is of particular importance in machine learning for data on irregular domains, while vertex clustering also represents a basis for collaborative data processing. Spectral domain analysis for vertex clustering may be performed based on several measures appropriate to the task at hand including the graph Laplacian, normalized graph Laplacian, generalized Laplacian eigenvectors, principal component analysis of the graph Laplacian, commute time (effective resistance) spectral vectors, the diffusion spectral vectors or other factors.

Definition: Vertex clustering is a type of graph learning which aims to group together vertices from the set $\mathcal{V}$ into multiple disjoint subsets, $\mathcal{V}_{i}$, called clusters. Vertices which are clustered into a subset of vertices, $\mathcal{V}_{i}$, are expected to exhibit a larger degree of within-cluster mutual similarity (in some sense) than with the vertices in other subsets, $\mathcal{V}_{j}$, $j \neq i$.

While the clustering of graph vertices refers to the process of identifying and arranging the vertices of a graph into nonoverlapping vertex subsets, with data in each subset expected to exhibit relative similarity
in some sense, the segmentation of a graph refers to its partitioning into nonoverlapping graph segments (components).

The notion of vertex similarity metrics and their use to accordingly cluster the vertices into sets, $\mathcal{V}_{i}$, of "related" vertices in graphs, has been a focus of significant research effort in machine learning and pattern recognition; this has resulted in a number of established vertex similarity measures and corresponding methods for graph clustering (Schaeffer, 2007b). These can be considered within two main categories (i) clustering based on graph topology and (ii) spectral (eigenvector-based) methods for graph clustering.

Notice that in traditional clustering, a vertex is assigned to one cluster only. The type of clustering where a vertex may belong to more than one cluster is referred to as fuzzy clustering (Mordeson and Nair, 2012; Schaeffer, 2007b), an approach that is not yet widely accepted in the context of graphs.

### 4.1 Clustering Based on Graph Topology

Among many such existing methods, the most popular ones are based on:

- Finding the minimum set of edges whose removal would disconnect a graph in some "optimal" or "least disturbance" way (minimum cut based clustering).
- Designing clusters within a graph based on the disconnection of vertices or edges which belong to the highest numbers of shortest paths in the graph (vertex betweenness and edge betweenness based clustering).
- The minimum spanning tree of a graph has been a basis for a number of widely used clustering methods (Kleinberg and Tardos, 2006; Morris et al., 1986).
- Analysis of highly connected subgraphs (HCS) (Khuller, 1998) has also been used for graph clustering.
- Finally, graph data analysis may be used for machine learned graph clustering, like for example, the $k$-means based clustering methods (Dhillon et al., 2004; Jain, 2010).


### 4.1.1 Minimum Graph Cut

We shall first briefly review the notion of graph cuts, as spectral methods for graph clustering may be introduced and interpreted based on the analysis and approximation of the (graph topology-based) minimum cut clustering.

Definition: Consider an undirected graph which is defined by a set of vertices, $\mathcal{V}$, and the corresponding set of edge weights, $\mathcal{W}$. Assume next that the vertices are grouped into $k=2$ disjoint subsets of vertices, $\mathcal{E} \subset \mathcal{V}$ and $\mathcal{H} \subset \mathcal{V}$, with $\mathcal{E} \cup \mathcal{H}=\mathcal{V}$ and $\mathcal{E} \cap \mathcal{H}=\emptyset$. A cut of this graph, for the given subsets of vertices, $\mathcal{E}$ and $\mathcal{H}$, is equal to a sum of all weights that correspond to the edges which connect the vertices between the subsets, $\mathcal{E}$ and $\mathcal{H}$, that is

$$
\operatorname{Cut}(\mathcal{E}, \mathcal{H})=\sum_{\substack{m \in \mathcal{E} \\ n \in \mathcal{H}}} W_{m n} .
$$

Remark 20: For clarity, we shall focus on the case with $k=2$ disjoint subsets of vertices. However, the analysis can be straightforwardly generalized to $k \geq 2$ disjoint subsets of vertices and the corresponding minimum $k$-cuts.
Example 12: Consider the graph in Figure 2.2, and the sets of vertices $\mathcal{E}=\{0,1,2,3\}$ and $\mathcal{H}=\{4,5,6,7\}$, shown in Figure 4.1. Its cut into the two components (sub-graphs), $\mathcal{E}$ and $\mathcal{H}$, involves the weights of all edges which exist between these two sets, that is, $\operatorname{Cut}(\mathcal{E}, \mathcal{H})=$ $0.32+0.24+0.23=0.79$. Such edges are shown by thin red lines in Figure 4.1.
Definition: A cut which exhibits the minimum value of the sum of weights between the disjoint subsets $\mathcal{E}$ and $\mathcal{H}$, considering all possible divisions of the set of vertices, $\mathcal{V}$, is referred to as the minimum cut. Finding the minimum cut of a graph in this way is a combinatorial problem.
Remark 21: The number of all possible combinations to split an even number of vertices, $N$, into two disjoint subsets is given by

$$
C=\binom{N}{1}+\binom{N}{2}+\cdots+\binom{N}{N / 2-1}+\binom{N}{N / 2} / 2 .
$$



Figure 4.1: A cut for the weighted graph from Figure 2.2, with the disjoint subsets of vertices defined by $\mathcal{E}=\{0,1,2,3\}$ and $\mathcal{H}=\{4,5,6,7\}$. The edges between the sets $\mathcal{E}$ and $\mathcal{H}$ are designated by thin red lines. The $\operatorname{cut}, \operatorname{Cut}(\mathcal{E}, \mathcal{H})$, is equal to the sum of the weights that connect sets $\mathcal{E}$ and $\mathcal{H}$, and has the value $\operatorname{Cut}(\mathcal{E}, \mathcal{H})=$ $0.32+0.24+0.23=0.79$.

To depict the computational burden associated with this "brute force" graph cut approach, even for a relatively small graph with $N=50$ vertices, the number of combinations to split the vertices into two subsets is $C=5.6 \cdot 10^{14}$.
Example 13: The minimum cut for the graph from Figure 4.1 is

$$
\operatorname{Cut}(\mathcal{E}, \mathcal{H})=0.32+0.14+0.15=0.61
$$

for $\mathcal{E}=\{0,1,2,3,4,5\}$ and $\mathcal{H}=\{6,7\}$. This can be confirmed by considering all $\binom{8}{1}+\binom{8}{2}+\binom{8}{3}+\binom{8}{4} / 2=127$ possible cuts, that is, all combinations of the subsets $\mathcal{E}$ and $\mathcal{H}$ for this small size graph or by using, for example, the Stoer-Wagner algorithm (Stoer and Wagner, 1997).

### 4.1.2 Maximum-Flow Minimum-Cut Approach

This approach to the minimum cut problem employs the framework of flow networks.
Definition: A flow network is a directed graph with an arbitrary number of vertices, $N \geq 3$, but which involves two given vertices (nodes) called the source vertex, $s$, and the sink vertex, $t$, whereby the capacity
of edges (arcs) is defined by their weights. The flow (of information, water, traffic, ...) through an edge cannot exceed its capacity (the value of edge weight). For any vertex in the graph the sum of all input flows is equal to the sum of all its output flows (except for the source and sink vertices).

Problem formulation. The maximum-flow minimum-cut solution to the graph partitioning aims to find the maximum value of flow that can be passed through the graph (network flow) from the source vertex, $s$, to the sink vertex, $t$. The solution is based on the max-flow min-cut theorem which states that the maximum flow through a graph from a given source vertex, $s$, to a given sink vertex, $t$, is equal to the minimum cut, that is, the minimum sum of those edge weights (capacities) which, if removed, would disconnect the source, $s$ from the sink, $t$ (minimum cut capacity) (Kleinberg and Tardos, 2006; Kron, 1963). Physical interpretation of this theorem is obvious, since the maximum flow is naturally defined by the graph flow bottleneck between the source and sink vertices. The capacity of the bottleneck (maximum possible flow) will then be equal to the minimum capacity (weight values) of the edges which, if removed, would disconnect the graph into two parts, one containing vertex $s$ and the other containing vertex $t$. Therefore, the problem of maximum flow is equivalent to the minimum cut (capacity) problem, under the assumption that the considered vertices, $s$ and $t$, must belong to different disjoint subsets of vertices $\mathcal{E}$ and $\mathcal{H}$. This kind of cut, with predefined vertices $s$ and $t$, is called the $(s, t)$ cut.

Remark 22: In general, if the source and sink vertices are not given, the maximum flow algorithm should be repeated for all combinations of the source and sink vertices in order to find the minimum cut of a graph.

The most widely used approach to solve the minimum-cut maximumflow problem is the Ford-Fulkerson method (Kleinberg and Tardos, 2006; Kron, 1963).
Example 14: Consider the weighted graph from Figure 2.2, with the assumed source and sink vertices, $s=0$ and $t=6$, as shown in Figure 4.2(a). The Ford-Fulkerson method is based on the analysis of paths and the corresponding flows between the source and sink vertex.


Figure 4.2: Principle of the maximum flow minimum cut method. (a) The weighted graph from Figure 2.2, with the assumed source vertex $s=0$ and sink vertex $t=6$, and a path between these two vertices for which the maximum flow is equal to the minimum capacity (weight) along this path, $W_{57}=0.15$. This maximum flow value, $W_{57}=0.15$, is then subtracted from all the original edge capacities (weights) to yield the new residual edge capacities (weights) which are shown in red. (b) The final edge capacities (weights) after the maximum flows are subtracted for all paths $0 \rightarrow 3 \rightarrow 6$, $0 \rightarrow 2 \rightarrow 4 \rightarrow 7 \rightarrow 6$, and $0 \rightarrow 2 \rightarrow 3 \rightarrow 6$, between vertices $s=0$ and $t=6$, with the resulting minimum cut now crossing only the zero-capacity (zero-weight) edges with its value equal to the sum of their initial capacities (weights), shown in Panel (a) in black. (c) A directed form of the undirected graph from (a), with the same path and the residual capacities (weights) given for both directions.

One such possible path between $s$ and $t, 0 \rightarrow 1 \rightarrow 4 \rightarrow 5 \rightarrow 7 \rightarrow 6$, is designated by the thick line in Figure 4.2(a). Recall that the maximum flow, for a path connecting the vertices $s=0$ and $t=6$, is restricted by the minimum capacity (equal to the minimum weight) along the considered path. For the considered path $0 \rightarrow 1 \rightarrow 4 \rightarrow 5 \rightarrow 7 \rightarrow 6$ the maximum flow from $s=0$ to $t=6$ is therefore equal to

$$
\max _{0 \rightarrow 1 \rightarrow 4 \rightarrow 5 \rightarrow 7 \rightarrow 6}=\min \{0.23,0.23,0.51,0.15,0.32\}=0.15,
$$

since the minimum weight along this path is that connecting vertices 5 and $7, W_{57}=0.15$. The value of this maximum flow is then subtracted from each capacity (weight) in the considered path, with the new residual edge capacities (weights) designated in red in the residual graph in Figure $4.2(\mathrm{a})$. The same procedure is repeated for the remaining possible paths $0 \rightarrow 3 \rightarrow 6,0 \rightarrow 2 \rightarrow 4 \rightarrow 7 \rightarrow 6$, and $0 \rightarrow 2 \rightarrow 3 \rightarrow 6$, with appropriate corrections to the capacities (edge weights) after consideration of each path. The final residual form of the graph, after zero-capacity edges are obtained in such a way that no new path with nonzero flow from $s$ to $t$ can be defined, is given in Figure 4.2(b). For example, if we consider the path $0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 6$ (or any other path), in the residual graph, then its maximum flow would be 0 , since the residual weight in the edge $3 \rightarrow 6$ is equal to 0 . The minimum cut has now been obtained as that which separates the sink vertex, $t=6$, and its neighborhood from the the source vertex, $s=0$, through the remaining zero-capacity (zero-weight) edges. This cut is shown in Figure 4.2(b), and separates the vertices $\mathcal{H}=\{6,7\}$ from the rest of vertices by cutting the edges connecting vertices $3 \rightarrow 6,4 \rightarrow 7$, and $5 \rightarrow 7$. The original total weights of these edges are $\operatorname{Cut}(\mathcal{E}, \mathcal{H})=0.32+0.14+0.15=0.61$.

We have so far considered an undirected graph, but since the Ford-Fulkerson algorithm is typically applied to directed graphs, notice that an undirected graph can be considered as a directed graph with every edge being split into a pair of edges having the same weight (capacity), but with opposite directions. After an edge is used in one direction (for example, edge 5-7 in Figure 4.2(a)) with a flow equal to its maximum capacity of 0.15 in the considered direction, the other flow direction (sister edge) becomes 0.30, as shown in Figure 4.2(c). The edge with opposite direction could be used (up the algebraic sum
of flows in both directions being equal to the total edge capacity) to form another path (if possible) from the source to the sink vertex. More specifically, the capacity of an edge (from the pair) in the assumed direction is reduced by the same value of the considered flow, while the capacity of the opposite-direction edge (from the same pair) is increased by the same flow, and can be used to send the flow in reverse direction if needed. All residual capacities for the path from Figure $4.2(\mathrm{a})$ are given in Figure 4.2 (c). For clarity, the edge weights which had not been changed by this flow are not shown in Figure 4.2(c).

## Ratio Minimum Cut

A number of optimization approaches may be employed to enforce some desired properties on graph clusters. One such approach is the ratio minimum cut, which is commonly used in graph theory, and is introduced by penalizing the value of $\operatorname{Cut}(\mathcal{E}, \mathcal{H})$ by an additional term (cost) to enforce the subsets $\mathcal{E}$ and $\mathcal{H}$ to be simultaneously as large as possible. An obvious form of the ratio cut is given by Hagen and Kahng (1992b)

$$
\begin{equation*}
\operatorname{Cut} N(\mathcal{E}, \mathcal{H})=\left(\frac{1}{N_{\mathcal{E}}}+\frac{1}{N_{\mathcal{H}}}\right) \sum_{\substack{m \in \mathcal{E} \\ n \in \mathcal{H}}} W_{m n} \tag{4.1}
\end{equation*}
$$

where $N_{\mathcal{E}}$ and $N_{\mathcal{H}}$ are the respective numbers of vertices in the sets $\mathcal{E}$ and $\mathcal{H}$. Since $N_{\mathcal{E}}+N_{\mathcal{H}}=N$, the term $\frac{1}{N_{\mathcal{E}}}+\frac{1}{N_{\mathcal{H}}}$ reaches its minimum for $N_{\mathcal{E}}=N_{\mathcal{H}}=N / 2$.
Example 15: Consider again Example 12, and the graph from Figure 4.1. For the sets of vertices, $\mathcal{E}=\{0,1,2,3\}$ and $\mathcal{H}=\{4,5,6,7\}$, the ratio cut is calculated as $\operatorname{Cut} N(\mathcal{E}, \mathcal{H})=(1 / 4+1 / 4) 0.79=0.395$. This cut also represents the minimum ratio cut for this graph; this can be confirmed by checking all possible cut combinations of $\mathcal{E}$ and $\mathcal{H}$ in this (small) graph. Figure 4.3 illustrates the clustering of vertices according to the minimum ratio cut. Notice, however, that in general the minimum cut and the minimum ratio cut do not produce the same vertex clustering into $\mathcal{E}$ and $\mathcal{H}$.
Graph separability. Relevant to this section, the minimum cut value admits a physical interpretation as a measure of graph separability.


Figure 4.3: A clustering scheme based on the minimum ratio cut of the vertices in the graph from Figure 2.2 into two vertex clusters, $\mathcal{E}=\{0,1,2,3\}$ and $\mathcal{H}=\{4,5,6,7\}$. This cut corresponds to the arbitrarily chosen cut presented in Figure 4.1.

An ideal separability is possible if the minimum cut is equal to zero, meaning that there are no edges between subsets $\mathcal{E}$ and $\mathcal{H}$. In Example 15 , the minimum cut value was $\operatorname{Cut} N(\mathcal{E}, \mathcal{H})=0.395$, which is not close to 0 , and indicates that the segmentation of this graph into two subgraphs would not yield a close approximation of the original graph.

### 4.1.3 Volume Normalized Minimum Cut

A more general form of the normalized cut may also involve vertex weights when designing the size of subsets $\mathcal{E}$ and $\mathcal{H}$. By defining, respectively, the volumes of these sets as $V_{\mathcal{E}}=\sum_{n \in \mathcal{E}} D_{n n}$ and $V_{\mathcal{H}}=\sum_{n \in \mathcal{H}} D_{n n}$, and using these volumes instead of the numbers of vertices $N_{\mathcal{E}}$ and $N_{\mathcal{H}}$ in the definition of the ratio cut in (28.7), we arrive at Shi and Malik (2000)

$$
\begin{equation*}
\operatorname{Cut} V(\mathcal{E}, \mathcal{H})=\left(\frac{1}{V_{\mathcal{E}}}+\frac{1}{V_{\mathcal{H}}}\right) \sum_{\substack{m \in \mathcal{E} \\ n \in \mathcal{H}}} W_{m n}, \tag{4.2}
\end{equation*}
$$

where $D_{n n}=\sum_{m \in \mathcal{V}} W_{m n}$ is the degree of a vertex $n$. The vertices with a higher degree, $D_{n n}$, are considered as structurally more important than the vertices with lower degrees.

The above discussion shows that finding the normalized minimum cut is also a combinatorial problem, for which an approximative spectralbased solution will be discussed later in this section.

### 4.1.4 Other Forms of the Normalized Cut

In addition to the two presented forms of the normalized cut, based on the number of vertices and volume, other frequently used forms are:

1. The sparsity of a cut which is defined as

$$
\begin{equation*}
\rho(\mathcal{E})=\frac{1}{N_{\mathcal{E}} N_{\mathcal{V}-\mathcal{E}}} \sum_{\substack{m \in \mathcal{E} \\ n \in \mathcal{V}-\mathcal{E}}} W_{m n} \tag{4.3}
\end{equation*}
$$

where $\mathcal{V}-\mathcal{E}$ is the set difference of $\mathcal{V}$ and $\mathcal{E}$. The sparsity of a cut, $\rho(\mathcal{E})$, is related to the ratio cut as $N \rho(\mathcal{E})=\operatorname{Cut} N(\mathcal{E}, \mathcal{H})$, since $\mathcal{H}=\mathcal{V}-\mathcal{E}$ and $N_{\mathcal{E}}+N_{\mathcal{V}-\mathcal{E}}=N$. The sparsity of a graph is then defined as the minimum sparsity of a cut. It then follows that the cut which exhibits minimum sparsity and the minimum ratio cut in (28.7) produce the same set $\mathcal{E}$.
2. The edge expansion of a subset, $\mathcal{E} \subset \mathcal{V}$, is defined by

$$
\begin{equation*}
\alpha(\mathcal{E})=\frac{1}{N_{\mathcal{E}}} \sum_{\substack{m \in \mathcal{E} \\ n \in \mathcal{V}-\mathcal{E}}} W_{m n} \tag{4.4}
\end{equation*}
$$

with $N_{\mathcal{E}} \leq N / 2$. Observe a close relation of edge expansion to the ratio cut in (28.7).
3. The Cheeger ratio of a subset, $\mathcal{E} \subset \mathcal{V}$, is defined as

$$
\begin{equation*}
\phi(\mathcal{E})=\frac{1}{\min \left\{V_{\mathcal{E}}, V_{\mathcal{V}-\mathcal{E}}\right\}} \sum_{\substack{m \in \mathcal{E} \\ n \in \mathcal{V}-\mathcal{E}}} W_{m n} \tag{4.5}
\end{equation*}
$$

The minimum value of $\phi(\mathcal{E})$ is denoted by $\phi(\mathcal{V})$ and called the Cheeger constant or conductance of a graph (Mohar, 1989). This form is closely related to the volume normalized cut in (4.2).

### 4.2 Spectral Methods for Graph Clustering

This class of methods is a modern alternative to the classical direct graph topology analysis, whereby vertex clustering is based on the eigenvectors of the graph Laplacian. Practical spectral methods for
graph clustering typically employ several smoothest eigenvectors of the graph Laplacian.

Simplified algorithms for vertex clustering may even employ only one eigenvector, namely the second (Fiedler, 1973b) eigenvector of the graph Laplacian, $\mathbf{u}_{1}$, to yield a quasi-optimal clustering or partitioning scheme on a graph. These are proven to be efficient in a range of applications, including data processing on graphs, machine learning, and computer vision (Malik et al., 2001). Despite their simplicity, such algorithms are typically quite accurate, and a number of studies show that graph clustering and cuts based on the second eigenvector, $\mathbf{u}_{1}$, give a good approximation to the optimal cut (Ng et al., 2002; Spielman and Teng, 2007b; Von Luxburg, 2007). Using more than one smooth eigenvector in graph clustering and partitioning will increase the number of degrees of freedom to consequently yield more physically meaningful clustering, when required for practical applications in data analytics.

For an enhanced insight we shall next review the smoothness index, before introducing the notions of graph spectral vectors and their distance, followed by the notions of similarity and clustering of vertices.

### 4.2.1 Smoothness of Eigenvectors on Graphs

Definition: The smoothness of an eigenvector, $\mathbf{u}_{k}$, is introduced through its quadratic Laplacian form, $\mathbf{u}_{k}^{T} \mathbf{L} \mathbf{u}_{k}$, with the smoothness index equal to the corresponding eigenvalue, $\lambda_{k}$, that is

$$
\begin{equation*}
\mathbf{u}_{k}^{T}\left(\mathbf{L} \mathbf{u}_{k}\right)=\mathbf{u}_{k}^{T}\left(\lambda_{k} \mathbf{u}_{k}\right)=\lambda_{k} . \tag{4.6}
\end{equation*}
$$

To demonstrate physical intuition behind the use of quadratic form, $\mathbf{u}_{k}^{T} \mathbf{L} \mathbf{u}_{k}$, as a smoothness metric of $\mathbf{u}_{k}$, consider

$$
\mathbf{u}_{k}^{T} \mathbf{L} \mathbf{u}_{k}=\mathbf{u}_{k}^{T}(\mathbf{D}-\mathbf{W}) \mathbf{u}_{k}
$$

Then, an $n$-th element of the vector $\mathbf{L} \mathbf{u}_{k}$ is given by

$$
\sum_{m=0}^{N-1} W_{n m} u_{k}(n)-\sum_{m=0}^{N-1} W_{n m} u_{k}(m)
$$

since $D_{n n}=\sum_{m=0}^{N-1} W_{n m}$. Therefore,

$$
\begin{align*}
\mathbf{u}_{k}^{T} \mathbf{L} \mathbf{u}_{k} & =\sum_{m=0}^{N-1} u_{k}(m) \sum_{n=0}^{N-1} W_{m n}\left(u_{k}(m)-u_{k}(n)\right) \\
& =\sum_{m=0}^{N-1} \sum_{n=0}^{N-1} W_{m n}\left(u_{k}^{2}(m)-u_{k}(m) u_{k}(n)\right) . \tag{4.7}
\end{align*}
$$

Owing to the symmetry of the weight matrix, $\mathbf{W}$ (as shown in (2.5)), we can use $W_{n m}=W_{m n}$ to replace the full summation of $u_{k}^{2}(n)$ over $m$ and $n$ with a half of the summations for both $u_{k}^{2}(m)$ and $u_{k}^{2}(n)$, over all $m$ and $n$. The same applies for the term $u(m) u(n)$. With that, we can write

$$
\begin{align*}
\mathbf{u}_{k}^{T} \mathbf{L} \mathbf{u}_{k}= & \frac{1}{2} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} W_{m n}\left(u_{k}^{2}(m)-u_{k}(m) u_{k}(n)\right) \\
& +\frac{1}{2} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} W_{m n}\left(u_{k}^{2}(n)-u_{k}(n) u_{k}(m)\right) \\
= & \frac{1}{2} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} W_{m n}\left(u_{k}(n)-u_{k}(m)\right)^{2} \geq 0 \tag{4.8}
\end{align*}
$$

Obviously, a small $\mathbf{u}_{k}^{T} \mathbf{L} \mathbf{u}_{k}=\lambda_{k}$ implies that all terms $W_{n m}\left(u_{k}(n)-\right.$ $\left.u_{k}(m)\right)^{2} \leq 2 \lambda_{k}$ are also small, thus indicating close values of $u_{k}(m)$ and $u_{k}(n)$ for vertices $m$ and $n$ with significant connections, $W_{m n}$. The eigenvectors corresponding to a small $\lambda_{k}$ are therefore slow-varying and smooth on a graph.
Example 16: An exemplar of eigenvectors with a small, a moderate and a large smoothness index, $\lambda_{k}$, is given on the three graphs in Figure 4.4.

In order to illustrate the interpretation of the smoothness index in classical time-domain data processing, the time-domain form of the eigenvectors/basis functions in the real-valued Fourier analysis (3.14) is also shown in Figure 4.4 (middle). In this case, the basis functions can be considered as the eigenvectors of a directed circular graph, where the vertices assume the role of time instants.

Observe that in all three graphs the smooth eigenvectors, $\mathbf{u}_{0}$ and $\mathbf{u}_{1}$, have similar elements on the neighboring vertices (in the case of a path graph - time instants), and thus may be considered as smooth data on


Figure 4.4: Illustration of the concept of smoothness of the graph Laplacian eigenvectors for three different graphs: The graph from Figure 2.2 (left), a path graph corresponding to classic temporal data analysis (middle), and an example of a more complex graph with $N=64$ vertices (right). (a) Constant eigenvector, $u_{0}(n)$, shown on the three considered graphs. This is the smoothest possible eigenvector for which the smoothness index is $\lambda_{0}=0$. (b) Slow-varying Fiedler eigenvector (the smoothest eigenvector whose elements are not constant), $u_{1}(n)$, for the three graphs considered. (c) Fast-varying eigenvectors, for $k=5$ (left), and $k=30$ (middle and right). Graph vertices are denoted by black circles, and the values of elements of the eigenvectors, $u_{k}(n)$, by red lines, for $n=0,1, \ldots, N-1$. The smoothness index, $\lambda_{k}$, is also given for each case.
the corresponding graph domains. Such similarity does not hold for the fast-varying eigenvectors, $\mathbf{u}_{5}$ (left of Figure 4.4) and $\mathbf{u}_{30}$ (middle and right of Figure 4.4), which exhibit a much higher smoothness index.

Remark 23: The eigenvector of the graph Laplacian which corresponds to $\lambda_{0}=0$ is constant (maximally smooth for any vertex ordering) and is therefore not appropriate as a template for vertex ordering. The next smoothest eigenvector is $\mathbf{u}_{1}$, which corresponds to the eigenvalue $\lambda_{1}$.

Ordering of vertices for smoothest Fiedler vector. It is natural to order vertices within a graph in such a way so that the presentation of
the sequence of elements of the smoothest eigenvector, $\mathbf{u}_{1}$, as a function of the vertex index, $n$, is also maximally smooth. This can be achieved by sorting (rank ordering) the elements of the Fiedler vector, $\mathbf{u}_{1}$, in a nondecreasing order. Recall from Remark 12 that the isomorphic nature of graphs means that the reindexing of vertices does not change any graph property. The new order of graph vertices in the sorted $\mathbf{u}_{1}$ then corresponds to the smoothest sequence of elements of this vector along the vertex index line.

A unique feature of graphs, which renders them indispensable in modern data analytics on irregular domains, is that the ordering of vertices of a graph can be arbitrary, an important difference from classical data analytics where the ordering is inherently sequential and fixed (Stankovic et al., 2019a). Therefore, in general, any change in data ordering (indexing) would cause significant changes in the results of classical methods, while when it comes to graphs, owing to their topological invariance (as shown in Figures 3.1 and 3.2 in the previous section), reordering of vertices would automatically imply the corresponding reordering of indices within each eigenvector, with no implication on the analysis results. However, the presentation of data sensed at the graph vertices, along a line of vertex indices, as in Figure 3.1(left), a common case for practical reasons, would benefit from an appropriate vertex ordering. Notice that vertex ordering in a graph is just a one-dimensional simplification of an important paradigm in graph analysis, known as graph clustering (Dong et al., 2012; Horaud, 2009; Hamon et al., 2016a; Lu et al., 2014; Masoumi and Hamza, 2017; Masoumi et al., 2016; Mejia et al., 2017).

### 4.2.2 Spectral Space and Spectral Similarity of Vertices

For a graph with $N$ vertices, the orthogonal eigenvectors of its graph Laplacian form the basis of an $N$-dimensional space, called the spectral space. In this way, the elements, $u_{k}(n)$, of every eigenvector $\mathbf{u}_{k}$, $k=0,1,2, \ldots, N-1$, are assigned to the corresponding vertices, $n=0,1,2, \ldots, N-1$, as shown in Figure 4.5(a). This, in turn, means that a set of elements, $u_{0}(n), u_{1}(n), u_{2}(n), \ldots, u_{N-1}(n)$, is assigned to every vertex $n$, as shown in Figure $4.5(\mathrm{~b})$. For every vertex, $n$, we can


Figure 4.5: Illustration of spectral vectors for the graph from Figure 2.2, with $N=8$ vertices. For an intuitive analogy with the classical Discrete Fourier Transform, notice that the complex harmonic basis functions within the DFT would play the role of eigenvectors in graph spectral representation, $\mathbf{u}_{k}, k=0,1, \ldots, 7$. Then, the spectral vectors, $\mathbf{q}_{n}, n=0,1, \ldots, 7$, would be analogous to the basis functions of the inverse Discrete Fourier transform (excluding the first constant element).
then group these elements into an N -dimensional spectral vector

$$
\mathbf{q}_{n} \stackrel{\text { def }}{=}\left[u_{0}(n), u_{1}(n), \ldots, u_{N-1}(n)\right]
$$

which is associated with the vertex $n$. Since the elements of the first eigenvector, $\mathbf{u}_{0}$, are constant, they do not convey any spectral difference to the graph vertices. Therefore, the elements of $\mathbf{u}_{0}$ are commonly
omitted from the spectral vector for vertex $n$, to yield

$$
\begin{equation*}
\mathbf{q}_{n}=\left[u_{1}(n), \ldots, u_{N-1}(n)\right], \tag{4.9}
\end{equation*}
$$

as illustrated in Figure 4.5(b).
Vertex dimensionality in the spectral space. Now that we have associated a unique spectral vector $\mathbf{q}_{n}$ in (4.9), to every vertex $n=$ $0,1, \ldots, N-1$, it is important to note that this ( $N-1$ )-dimensional representation of every vertex in a graph (whereby the orthogonal graph Laplacian eigenvectors, $\mathbf{u}_{1}, \mathbf{u}_{2}, \ldots, \mathbf{u}_{N-1}$, serve as a basis of that representation) does not affect the graph itself; this just means that the additional degrees of freedom introduced through spectral vectors facilitate more sophisticated and efficient graph analysis. For example, we may now talk about vertex similarity in the spectral space, or about the spectrum based graph cut, segmentation, and vertex clustering.

An analogy with classical signal processing would be to assign a vector of harmonic basis function values at a time instant (vertex) $n$, to "describe" this instant, that is, to assign the $n$-th column of the Discrete Fourier transform matrix to the instant $n$. This intuition is illustrated in Figures 4.5(a) and (b).

The spectral vectors shall next be used to define spectral similarity of vertices.

Definition: Two vertices, $m$ and $n$, are called spectrally similar if their distance in the spectral space is within a small predefined threshold. The spectral similarity between vertices $m$ and $n$ is typically measured through the Euclidean norm of their spectral space distance, given by

$$
d_{m n} \stackrel{\text { def }}{=}\left\|\mathbf{q}_{m}-\mathbf{q}_{n}\right\|_{2}
$$

Spectral manifold. Once graph is characterized by the original ( $N-1$ )-dimensional spectral vectors, the so obtained vertex positions in spectral vertex representation may reside near some well defined surface (commonly a hyperplane) called a spectral manifold which is of a reduced dimensionality $M<(N-1)$. The aim of spectral vertex mapping is then to map each spectral vertex representation from the original $N$-dimensional spectral vector space to a new spectral manifold which lies in a reduced $M$-dimensional spectral space, to a position closest
to its original $(N-1)$-dimensional spectral position. This principle is related to the Principal Component Analysis (PCA) method, and this relation will be discussed later in this section. An analogy with classical Discrete Fourier Transform analysis would imply a restriction of the spectral analysis from the space of $N$ harmonics to the reduced space of the $M$ slowest-varying harmonics (excluding the constant one).

These spectral dimensionality reduction considerations suggest to restrict the definition of spectral similarity to only a few lower-order (smooth) eigenvectors in the spectral space of reduced dimensionality. For example, if the spectral similarity is restricted to the two smoothest eigenvectors, $\mathbf{u}_{1}$ and $\mathbf{u}_{2}$ (omitting the constant $\mathbf{u}_{0}$ ), then the spectral vector for a vertex $n$ would become

$$
\mathbf{q}_{n}=\left[u_{1}(n), u_{2}(n)\right]
$$

as illustrated in Figures $4.5(\mathrm{c})$ and $4.6(\mathrm{a})$. If for two vertices, $m$ and $n$, the values of $u_{1}(m)$ are close to $u_{1}(n)$ and the values of $u_{2}(m)$ are close to $u_{2}(n)$, then these two vertices are said to be spectrally similar, that is, they exhibit a small spectral distance, $d_{m n}=\left\|\mathbf{q}_{m}-\mathbf{q}_{n}\right\|_{2}$.

Finally, the simplest spectral description uses only one (smoothest nonconstant) eigenvector to describe the spectral content of a vertex, so that the spectral vector reduces to a spectral scalar

$$
\mathbf{q}_{n}=\left[q_{n}\right]=\left[u_{1}(n)\right]
$$

whereby the so reduced spectral space is a one-dimensional line.
Example 17: The two-dimensional and three-dimensional spectral vectors, $\mathbf{q}_{n}=\left[u_{1}(n), u_{2}(n)\right]$ and $\mathbf{q}_{n}=\left[u_{1}(n), u_{2}(n), u_{3}(n)\right]$, of the graph from Figure 2.2 are shown in Figure 4.6 , for $n=2$ and $n=6$.
Spectral embedding. The mapping from the reduced dimensionality spectral space back onto the original vertices is referred to as Spectral embedding.

We can proceed in two ways with the reduced spectral vertex space representation: (i) to assign the reduced dimension spectral vectors to the original vertex positions, for example, in the form of vertex coloring, as a basis for subsequent vertex clustering (Section 4.2.3), or (ii) to achieve new vertex positioning in the reduced dimensionality space of


Figure 4.6: Illustration of the spectral vectors, $\mathbf{q}_{n}=\left[u_{1}(n), u_{2}(n)\right]$ and $\mathbf{q}_{n}=$ $\left[u_{1}(n), u_{2}(n), u_{3}(n)\right]$, for the Laplacian matrix of the graph in Figure 2.2. (a) Twodimensional spectral vectors, $\mathbf{q}_{2}=\left[u_{1}(2), u_{2}(2)\right]$ and $\mathbf{q}_{6}=\left[u_{1}(6), u_{2}(6)\right]$. (b) Threedimensional spectral vectors, $\mathbf{q}_{2}=\left[u_{1}(2), u_{2}(2), u_{3}(2)\right]$ and $\mathbf{q}_{6}=\left[u_{1}(6), u_{2}(6), u_{3}(6)\right]$. For clarity, the spectral vectors are shown on both the vertex index axis and directly on graph.
eigenvectors (reduced spectral space), using eigenmaps (Section 4.4). Both yield similar information and can be considered as two sides of the same coin (Belkin and Niyogi, 2003). For visualization purposes, we will use colors of the RGB system to represent the spectral vector values in a reduced (one, two, or three) dimensional spectral space. Vertices at the original graph positions will be colored according to the spectral vector values.

### 4.2.3 Indicator Vector

Remark 21 shows that the combinatorial approach to minimum cut problem is computationally infeasible, as even for a graph with only 50 vertices we have $5.6 \cdot 10^{14}$ such potential cuts.

To break this Curse of Dimensionality it would be very convenient to relate the problem of the minimization of the ratio cut in (28.7) and (4.2) to that of eigenanalysis of graph Laplacian. To this end, we shall introduce the notion of indicator vector $\mathbf{x}$ on a graph, for which the elements take subgraph-wise constant values within each disjoint subset (cluster) of vertices, with these constants taking different values for different clusters of vertices (subset-wise constant vector). While this does not immediately reduce the computational burden (the same number of combinations remains as in the brute force method), the elements of $\mathbf{x}$ now uniquely reflect the assumed cut of the graph into disjoint subsets $\mathcal{E}, \mathcal{H} \subset \mathcal{V}$.

Establishing a further link with only the smoothest eigenvector of the graph Laplacian will convert the original, computationally intractable, combinatorial minimum cut problem into a manageable algebraic eigenvalue problem, for which the computation complexity is of the $\mathcal{O}\left(N^{3}\right)$ order. By casting the problem into the linear algebra framework, complexity of calculation can be additionally reduced through efficient eigenanalysis methods, such as the Power Method which sequentially computes the desired number of largest eigenvalues and the corresponding eigenvectors, at an affordable $\mathcal{O}\left(N^{2}\right)$ computations per iteration, as shown in the appendix.

However, unlike the indicator vector, $\mathbf{x}$, the smoothest eigenvector (corresponding to the smallest nonzero eigenvalue) of graph Laplacian
is not subset-wise constant, and such solution would be approximate, but computationally feasible.
Remark 24: The concept of indicator vector can be introduced through the analysis with an ideal minimum cut of a graph, given by

$$
\operatorname{Cut}(\mathcal{E}, \mathcal{H})=\sum_{\substack{m \in \mathcal{E} \\ n \in \mathcal{H}}} W_{m n}=0
$$

that is, when considering an already disjoint graph for which $\operatorname{Cut}(\mathcal{E}, \mathcal{H})=0$ indicates that there exist no edges between the subsets $\mathcal{E}$ and $\mathcal{H}$, that is, $W_{m n}=0$ for $m \in \mathcal{E}$, and $n \in \mathcal{H}$. Obviously, this ideal case can be solved without resorting to the combinatorial approach, since this graph is already in the form of two disconnected subgraphs, defined by the sets of vertices $\mathcal{E}$ and $\mathcal{H}$. For such a disconnected graph, the second eigenvalue of the graph Laplacian is $\lambda_{1}=0$, as established by the graph Laplacian property $L_{2}$. When $\lambda_{1}=0$, then

$$
2 \mathbf{u}_{1}^{T} \mathbf{L} \mathbf{u}_{1}=\sum_{m=0}^{N-1} \sum_{n=0}^{N-1} W_{m n}\left(u_{1}(n)-u_{1}(m)\right)^{2}=2 \lambda_{1}=0
$$

which follows from (4.6) and (4.8). Since all terms in the last sum are nonnegative, this implies that they must be zero-valued, that is, the eigenvector $\mathbf{u}_{1}$ is subset-wise constant, with $u_{1}(n)=u_{1}(m)=c_{1}$ for $m, n \in \mathcal{E}$ and $u_{1}(n)=u_{1}(m)=c_{2}$ for $m, n, \in \mathcal{H}$. Since the eigenvector $\mathbf{u}_{1}$ is orthogonal to the constant eigenvector $\mathbf{u}_{0}$, then $\sum_{n=0}^{N-1} u_{1}(n)=0$. A possible solution for $u_{1}(n)$, that satisfies the subset-wise constant form and has zero mean, is $u_{1}(n)=c_{1}=1 / N_{\mathcal{E}}$ for $n \in \mathcal{E}$ and $u_{1}(n)=$ $c_{2}=-1 / N_{\mathcal{H}}$ for $n \in \mathcal{H}$. We can conclude that the problem of finding an ideal minimum cut can indeed be solved by introducing an indicator vector $\mathbf{x}=\mathbf{u}_{1}$, such that $x(n)=1 / N_{\mathcal{E}}$ for $n \in \mathcal{E}$ and $x(n)=-1 / N_{\mathcal{H}}$ for $n \in \mathcal{H}$. The membership of a vertex, $n$, to either the subset $\mathcal{E}$ or $\mathcal{H}$ of the ideal minimum cut is therefore uniquely defined by the sign of indicator vector $\mathbf{x}=\mathbf{u}_{1}$. This form of $\mathbf{x}$ is not normalized to unit energy, as its scaling by any constant would not influence solution for vertex clustering into subsets $\mathcal{E}$ or $\mathcal{H}$.

For a general graph, and following the above reasoning, we here consider two specific subset-wise constant forms of the indicator vector, $\mathbf{x}$, which are based on
(i) the number of vertices in disjoint subgraphs,

$$
x(n)= \begin{cases}\frac{1}{N_{\mathcal{E}}}, & \text { for } n \in \mathcal{E}  \tag{4.10}\\ -\frac{1}{N_{\mathcal{H}}}, & \text { for } n \in \mathcal{H}\end{cases}
$$

where $N_{\mathcal{E}}$ is the number of vertices in $\mathcal{E}$, and $N_{\mathcal{H}}$ is the number of vertices in $\mathcal{H}$, and
(ii) the volumes of the disjoint subgraphs,

$$
x(n)= \begin{cases}\frac{1}{V_{\mathcal{E}}}, & \text { for } n \in \mathcal{E}  \tag{4.11}\\ -\frac{1}{V_{\mathcal{H}}}, & \text { for } n \in \mathcal{H}\end{cases}
$$

where the volumes of the sets, $V_{\mathcal{E}}$ and $V_{\mathcal{H}}$, are defined as the sums of all vertex degrees, $D_{n n}$, in the corresponding subsets, $V_{\mathcal{E}}=\sum_{n \in \mathcal{E}} D_{n n}$ and $V_{\mathcal{H}}=\sum_{n \in \mathcal{H}} D_{n n}$.

Before proceeding further with the analysis of these two forms of indicator vector (in the next two remarks), it is important to note that if we can find the vector $\mathbf{x}$ which minimizes the ratio cut, $\operatorname{Cut} N(\mathcal{E}, \mathcal{H})$ in (28.7), then the elements of vector $\mathbf{x}$ (their $\operatorname{signs,~} \operatorname{sign}(x(n))=1$ for $n \in \mathcal{E}$ and $\operatorname{sign}(x(n))=-1$ for $n \in \mathcal{H})$ may be used to decide whether to associate a vertex, $n$, to either the set $\mathcal{E}$ or $\mathcal{H}$ of the minimum ratio cut.
Remark 25: The ratio cut, $\operatorname{Cut} N(\mathcal{E}, \mathcal{H})$, defined in (28.7), for the indicator vector $\mathbf{x}$ with the elements $x(n)=1 / N_{\mathcal{E}}$ for $n \in \mathcal{E}$ and $x(n)=-1 / N_{\mathcal{H}}$ for $n \in \mathcal{H}$, is equal to the Rayleigh quotient of the matrix $\mathbf{L}$ and vector $\mathbf{x}$, that is

$$
\begin{equation*}
\operatorname{Cut} N(\mathcal{E}, \mathcal{H})=\frac{\mathbf{x}^{T} \mathbf{L} \mathbf{x}}{\mathbf{x}^{T} \mathbf{x}} \tag{4.12}
\end{equation*}
$$

To prove this relation we rewrite (4.8) as

$$
\begin{equation*}
\mathbf{x}^{T} \mathbf{L} \mathbf{x}=\frac{1}{2} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} W_{m n}(x(n)-x(m))^{2} \tag{4.13}
\end{equation*}
$$

For all vertices $m$ and $n$ within the same subgraph, that is, such that $m \in \mathcal{E}$ and $n \in \mathcal{E}$, the elements of vector $\mathbf{x}$ are therefore the same
and equal to $x(m)=x(n)=1 / N_{\mathcal{E}}$. In turn, this means that the terms $(x(n)-x(m))^{2}$ in (4.13) are zero-valued. The same holds for any two vertices belonging to the set $\mathcal{H}$, that is, for $m \in \mathcal{H}$ and $n \in \mathcal{H}$. Therefore, only the terms corresponding to the edges which define the cut, when $m \in \mathcal{E}$ and $n \in \mathcal{H}$, and vice versa, remain in the sum, and they are constant and equal to $(x(n)-x(m))^{2}=\left(1 / N_{\mathcal{E}}-\left(-1 / N_{\mathcal{H}}\right)\right)^{2}$, to yield

$$
\begin{align*}
\mathbf{x}^{T} \mathbf{L} \mathbf{x} & =\left(\frac{1}{N_{\mathcal{E}}}+\frac{1}{N_{\mathcal{H}}}\right)^{2} \sum_{\substack{m \in \mathcal{E} \\
n \in \mathcal{H}}} W_{m n} \\
& =\left(\frac{1}{N_{\mathcal{E}}}+\frac{1}{N_{\mathcal{H}}}\right) \operatorname{Cut} N(\mathcal{E}, \mathcal{H}) \tag{4.14}
\end{align*}
$$

where the ratio cut, $\operatorname{Cut} N(\mathcal{E}, \mathcal{H})$, is defined in (28.7). Finally, from the energy of the indicator vector, $\mathbf{x}^{T} \mathbf{x}=e_{x}^{2}$,

$$
\begin{equation*}
\mathbf{x}^{T} \mathbf{x}=\|\mathbf{x}\|_{2}^{2}=e_{x}^{2}=\frac{N_{\mathcal{E}}}{N_{\mathcal{E}}^{2}}+\frac{N_{\mathcal{H}}}{N_{\mathcal{H}}^{2}}=\frac{1}{N_{\mathcal{E}}}+\frac{1}{N_{\mathcal{H}}} \tag{4.15}
\end{equation*}
$$

which proves (28.9).
The same analysis holds if the indicator vector is normalized to unit energy, whereby $x(n)=1 /\left(N_{\mathcal{E}} e_{x}\right)$ for $n \in \mathcal{E}$ and $x(n)=-1 /\left(N_{\mathcal{H}} e_{x}\right)$ for $n \in \mathcal{H}$, with $e_{x}$ defined in (4.15) as $e_{x}=\|\mathbf{x}\|_{2}$.

We can therefore conclude that the indicator vector, $\mathbf{x}$, which solves the problem of minimization of the ratio cut, is also a solution to (28.9). This minimization problem, for the unit energy form of the indicator vector the elements of which are the minimization variables, can also be written as

$$
\begin{equation*}
\min _{\mathbf{x}}\left\{\mathbf{x}^{T} \mathbf{L} \mathbf{x}\right\} \text { subject to } \mathbf{x}^{T} \mathbf{x}=1 \tag{4.16}
\end{equation*}
$$

In general, this is again a combinatorial problem, since all possible combinations of subsets of vertices, $\mathcal{E}$ and $\mathcal{H}$, together with the corresponding indicator vectors, $\mathbf{x}$, must be considered.

For a moment we shall put aside the very specific (subset-wise constant) form of the indicator vector and consider the general minimization problem in (28.11). This problem can be solved using the method of Lagrange multipliers, with the corresponding cost function

$$
\mathcal{L}(\mathbf{x})=\mathbf{x}^{T} \mathbf{L} \mathbf{x}-\lambda\left(\mathbf{x}^{T} \mathbf{x}-1\right)
$$

From $\partial \mathcal{L}(\mathbf{x}) / \partial \mathbf{x}^{T}=\mathbf{0}$, it follows that $\mathbf{L} \mathbf{x}=\lambda \mathbf{x}$, which is precisely the eigenvalue/eigenvector relation for the graph Laplacian $\mathbf{L}$, the solution of which is $\lambda=\lambda_{k}$ and $\mathbf{x}=\mathbf{u}_{k}$, for $k=0,1, \ldots, N-1$. In other words, upon replacing $\mathbf{x}$ by $\mathbf{u}_{k}$ in the term $\min \left\{\mathbf{x}^{T} \mathbf{L} \mathbf{x}\right\}$ above, we obtain $\min _{k}\left\{\mathbf{u}_{k}^{T} \mathbf{L} \mathbf{u}_{k}\right\}=\min _{k}\left\{\lambda_{k}\right\}$. After neglecting the trivial solution $\lambda_{0}=0$, since it produces a constant eigenvector $\mathbf{u}_{0}$, we next arrive at $\min _{k}\left\{\lambda_{k}\right\}=\lambda_{1}$ and $\mathbf{x}=\mathbf{u}_{1}$. Note that this solution yields a general form of vector $\mathbf{x}$ that minimizes (28.9). However, such a form does not necessarily correspond to a subset-wise constant indicator vector, $\mathbf{x}$. The fact that the trivial solution (constant vector $\mathbf{x}$ ) is neglected, is commonly written as an additional constraint in (28.11), of the form (see also Part III, Section 13.3),

$$
\mathbf{x}^{T} \mathbf{1}=\mathbf{0}
$$

### 4.2.4 Bounds on the Minimum Cut

In general, the subset-wise constant indicator vector, $\mathbf{x}$, may be written as a linear combination of the graph Laplacian eigenvectors, $\mathbf{u}_{k}, k=$ $1,2, \ldots, N-1$, to give

$$
\begin{equation*}
\mathbf{x}=\alpha_{1} \mathbf{u}_{1}+\alpha_{2} \mathbf{u}_{2}+\cdots+\alpha_{N-1} \mathbf{u}_{N-1} \tag{4.17}
\end{equation*}
$$

This kind of vector expansion onto the set of eigenvectors shall be addressed in more detail in Part II of this monograph. Note that the constant eigenvector $\mathbf{u}_{0}$ is omitted since the indicator vector is zero-mean by definition (orthogonal to a constant vector). The calculation of coefficients $\alpha_{i}$ would require the indicator vector (that is, the sets $\mathcal{E}$ and $\mathcal{H}$ ) to be known, leading again to the combinatorial problem of vertex set partitioning. It is interesting to note that the quadratic form of indicator vector, $\mathbf{x}$, given by (4.17) is equal to $\mathbf{x}^{T} \mathbf{L} \mathbf{x}=$ $\alpha_{1}^{2} \lambda_{1}+\alpha_{2}^{2} \lambda_{2}+\cdots+\alpha_{N-1}^{2} \lambda_{N-1}$, and that it assumes the minimum value for $\alpha_{1}=1, \alpha_{2}=\cdots=\alpha_{N-1}=0$, that is, when $\mathbf{x}=\mathbf{u}_{1}$, which corresponds to imposing the normalized energy condition, $\mathbf{x}^{T} \mathbf{x}=\alpha_{1}^{2}+\alpha_{2}^{2}+\cdots+$ $\alpha_{N-1}^{2}=1$. In other words, we now arrive at a physically meaningful bound

$$
\lambda_{1} \leq \mathbf{x}^{T} \mathbf{L} \mathbf{x}=C u t N(\mathcal{E}, \mathcal{H})
$$

Observe that this inequality corresponds to the lower Cheeger bound for the minimum ratio cut in (28.7).

Remark 26: If the space of approximative solutions for the indicator vector, $\mathbf{x}$, is relaxed to allow for vectors that are not subset-wise constant (while omitting the constant eigenvector of the graph Laplacian, $\mathbf{u}_{0}$ ), the approximative solution becomes $\mathbf{x}=\mathbf{u}_{1}$ (as previously shown and illustrated in Example (18)). The above analysis indicates that this solution is quasi-optimal, however, despite its simplicity, the graph cut based on only the second graph Laplacian eigenvector, $\mathbf{u}_{1}$, typically produces a good approximation to the optimal (minimum ratio) cut.

It has been shown that the value of the true ratio minimum cut in (28.7), when the indicator vector $\mathbf{x}$ is subset-wise constant, is bounded on both sides (upper and lower) with the constants which are proportional to the smallest nonzero eigenvalue, $\mathbf{u}_{1}^{T} \mathbf{L} \mathbf{u}_{1}=\lambda_{1}$, of the graph Laplacian. The simplest form of these bounds (Cheeger's bounds) for the cut defined by (4.5), has the form (alon1986eigenvalues; Chung, 2005, chung2007four; Trevisan, 2013)

$$
\begin{equation*}
\frac{\lambda_{1}}{2} \leq \phi(\mathcal{V}) \stackrel{\text { def }}{=} \min _{\mathcal{E} \subset \mathcal{V}}\{\phi(\mathcal{E})\} \leq \sqrt{2 \lambda_{1}} \tag{4.18}
\end{equation*}
$$

This shows that the eigenvalue $\lambda_{1}$ is also a good measure of a graph separability and consequently the quality of spectral clustering in the sense of a minimum Cheeger's ratio cut. The value of the minimum Cheeger's ratio cut of a graph (also referred to as Cheeger's constant, conductivity, or isoperimetric number of a graph) may also be considered as a numerical measure of the presence of a "bottleneck" in a graph.

### 4.2.5 Indicator Vector for Normalized Graph Laplacian

We shall now address the cut based on normalized graph Laplacian, in light of the above analysis.

Remark 27: The volume normalized cut, $\operatorname{Cut} V(\mathcal{E}, \mathcal{H})$, defined in (4.2), is equal to

$$
\begin{equation*}
\operatorname{Cut} V(\mathcal{E}, \mathcal{H})=\frac{\mathbf{x}^{T} \mathbf{L} \mathbf{x}}{\mathbf{x}^{T} \mathbf{D} \mathbf{x}}, \tag{4.19}
\end{equation*}
$$

where the corresponding, subset-wise constant, indicator vector has the values $x(n)=1 / V_{\mathcal{E}}$ for $n \in \mathcal{E}$ and $x(n)=-1 / V_{\mathcal{H}}$ for $n \in \mathcal{H}$, while the volumes of the sets, $V_{\mathcal{E}}$ and $V_{\mathcal{H}}$, are defined in (4.2).

The proof is identical to that given in Remark 25. For the normalized indicator vector, we have $\mathbf{x}^{T} \mathbf{D} \mathbf{x}=1$, so that the minimization problem in (4.19), for finding the elements of $\mathbf{x}$, reduces to

$$
\begin{equation*}
\min \left\{\mathbf{x}^{T} \mathbf{L} \mathbf{x}\right\} \text { subject to } \mathbf{x}^{T} \mathbf{D} \mathbf{x}=1 \tag{4.20}
\end{equation*}
$$

If the solution space is restricted to the space of generalized eigenvectors of the graph Laplacian, defined by

$$
\mathbf{L} \mathbf{u}_{k}=\lambda_{k} \mathbf{D} \mathbf{u}_{k}
$$

then the solution to (28.16) becomes

$$
\mathbf{x}=\mathbf{u}_{1}
$$

where $\mathbf{u}_{1}$ is the generalized eigenvector of the graph Laplacian that corresponds to the lowest nonzero eigenvalue. The fact that the trivial solution (constant vector $\mathbf{x}$ ) is avoided as a solution, can be written in the form of an additional constraint, $\mathbf{x}^{T} \mathbf{1}=\mathbf{0}$, in (28.16).

The eigenvectors of the normalized Laplacian, $\mathbf{L}_{N}=\mathbf{D}^{-1 / 2} \mathbf{L} \mathbf{D}^{-1 / 2}$, may also be used in optimal cut approximations since the minimization problem in (4.19) can be rewritten using the normalized Laplacian through a change of the variable, to yield

$$
\mathbf{x}=\mathbf{D}^{-1 / 2} \mathbf{y}
$$

which allows us to arrive at the following form Ng et al. (2002)

$$
\begin{gather*}
\min \left\{\mathbf{y}^{T} \mathbf{D}^{-1 / 2} \mathbf{L} \mathbf{D}^{-1 / 2} \mathbf{y}\right\}=\min \left\{\mathbf{y}^{T} \mathbf{L}_{N} \mathbf{y}\right\} \\
\text { subject to } \mathbf{y}^{T} \mathbf{y}=1 \tag{4.21}
\end{gather*}
$$

If the space of solutions to this minimization problem is relaxed to the eigenvectors, $\mathbf{v}_{k}$, of the normalized graph Laplacian, $\mathbf{L}_{N}$, then $\mathbf{y}=\mathbf{v}_{1}$. For more detail on the various forms of the eigenvalues and eigenvectors of graph Laplacian, we refer to Table 4.1.

It is obvious now from (28.16) and (4.21) that the relation of the form $\mathbf{x}=\mathbf{D}^{-1 / 2} \mathbf{y}$ also holds for the corresponding eigenvectors of the normalized graph Laplacian, $\mathbf{v}_{k}$, and the generalized eigenvectors of the
Table 4.1: Summary of graph embedding mappings. The Graph Laplacian mapping, the Generalized eigenvectors of the Laplacian mapping, the Normalized Laplacian mapping, the Commute time mapping, the Diffusion mapping, and the Cumulative diffusion mapping.

| Mapping | Eigen-Analysis Relation | Reduced Dimensionality Spectral Vector |
| :--- | :--- | :--- |
| Graph Laplacian mapping | $\mathbf{L u} \mathbf{u}_{k}=\lambda_{k} \mathbf{u}_{k}$ | $\mathbf{q}_{n}=\left[u_{1}(n), u(2), \ldots, u_{M}(n)\right]$ |
| Generalized eigenvectors | $\mathbf{L u} \mathbf{u}_{k}=\lambda_{k} \mathbf{D} \mathbf{u}_{k}$ | $\mathbf{q}_{n}=\left[u_{1}(n), u(2), \ldots, u_{M}(n)\right]$ |
| $\quad$ of Laplacian mapping | $\left(\mathbf{D}^{-1 / 2} \mathbf{L D}^{-1 / 2}\right) \mathbf{u}_{k}=\lambda_{k} \mathbf{u}_{k}$ | $\mathbf{q}_{n}=\left[u_{1}(n), u(2), \ldots, u_{M}(n)\right]$ |
| Normalized Laplacian mapping | $\mathbf{L u}_{k}=\lambda_{k} \mathbf{u}_{k}$ | $\mathbf{q}_{n}=\left[\frac{u_{1}(n)}{\sqrt{\lambda_{1}}}, \frac{u_{2}(n)}{\sqrt{\lambda_{2}}}, \ldots, \frac{u_{M}(n)}{\sqrt{\lambda_{M}}}\right]$ |
| Commute time mapping | $\mathbf{L u} \mathbf{u}_{k}=\lambda_{k} \mathbf{D} \mathbf{u}_{k}$ | $\mathbf{q}_{n}=\left[u_{1}(n)\left(1-\lambda_{1}\right)^{t}, \ldots, u_{M}(n)\left(1-\lambda_{M}\right)^{t}\right]$ |
| Diffusion (random walk) mapping | $\mathbf{q}_{n}=\left[\frac{u_{1}(n)}{\lambda_{1}}, \frac{u_{2}(n)}{\lambda_{2}}, \ldots, \frac{u_{M}(n)}{\lambda_{M}}\right]$ |  |
| Cumulative diffusion mapping | $\mathbf{L u}=\lambda_{k} \mathbf{D} \mathbf{u}_{k}$ |  |

Laplacian, $\mathbf{v}_{k}$, that is,

$$
\mathbf{u}_{k}=\mathbf{D}^{-1 / 2} \mathbf{v}_{k}
$$

It is important to note that, in general, clustering results based on the three forms of eigenvectors:
(i) the smoothest graph Laplacian eigenvector,
(ii) the smoothest generalized eigenvector of the Laplacian, and
(iii) the smoothest eigenvector of the normalized Laplacian,
are different. While the method (i) favors the clustering into subsets with (almost) equal number of vertices, the methods (ii) and (iii) favor subsets with (almost) equal volumes (defined as sums of the vertex degrees in the subsets). Also note that the methods (i) and (ii) approximate the indicator vector in different eigenvector subspaces. All three methods will produce the same clustering result for unweighted regular graphs, for which the volumes of subsets are proportional to the number of their corresponding vertices, while the eigenvectors for all the three Laplacian forms are the same in regular graphs, as shown in (2.13).
Generalized eigenvectors of the graph Laplacian and eigenvectors of the normalized Laplacian. Recall that the matrix $\mathbf{D}^{-1 / 2}$ is of a diagonal form, and with positive elements. Then, the solution to (28.16) which is equal to the generalized eigenvector of the graph Laplacian, and the solution to (4.21) which is equal to the eigenvector of the normalized Laplacian, are related as $\operatorname{sign}(\mathbf{y})=\operatorname{sign}(\mathbf{x})$ or $\operatorname{sign}\left(\mathbf{v}_{1}\right)=\operatorname{sign}\left(\mathbf{u}_{1}\right)$. This indicates that if the sign of the corresponding eigenvector is used for the minimum cut approximation (clustering), both results are the same.

### 4.3 Spectral Clustering Implementation

Spectral clustering is most conveniently implemented using only lowdimensional spectral vectors, with the simplest case when only a onedimensional spectral vector is used as indicator vector. More degrees of freedom can be achieved by clustering schemes which use two or three Laplacian eigenvectors, as discussed next.

### 4.3.1 Clustering Based on Only One (Fiedler) Eigenvector

From the analysis in the previous section, we can conclude that only the smoothest eigenvector, $\mathbf{u}_{1}$, can produce a good (quasi-optimal) approximation to the problem of minimum ratio cut graph clustering into two subsets of vertices, $\mathcal{E}$ and $\mathcal{H}$. Within the concept of spectral vectors, presented in Section 4.2.2, this indicates that the simplest form of spectral vector, $\mathbf{q}_{n}=u_{1}(n)$, based on just one (the smoothest) Fiedler eigenvector, $\mathbf{u}_{1}$, can be used for efficient spectral vertex clustering. Since the spectral vector $\mathbf{q}_{n}=u_{1}(n)$ is used as an approximative solution to the indicator vector within the minimum ratio cut definition, its values may be normalized. One such normalization

$$
\begin{equation*}
\mathbf{y}_{n}=\mathbf{q}_{n} /\left\|\mathbf{q}_{n}\right\|_{2} \tag{4.22}
\end{equation*}
$$

yields a two-level form of the spectral vector

$$
\mathbf{y}_{n}=\left[u_{1}(n) /\left\|u_{1}(n)\right\|_{2}\right]=\left[\operatorname{sign}\left(u_{1}(n)\right)\right]
$$

and represents a step before clustering, as proposed in Ng et al. (2002). This is justified based on the original form of the indicator vector, whose sign indicates the vertex association to either subset $\mathcal{E}$ or $\mathcal{H}$. For illustrative representation of the normalized spectral vector, we may use a simple two-level colormap and assign one of two colors to each vertex. Such a simple algorithm for clustering is given in Algorithm 1 (for an algorithm with more options for clustering and representation see the appendix (Algorithm 9) and Remarks 30 and 33).
Example 18: Consider the graph from Figure 2.2 and its Laplacian eigenvector, $\mathbf{u}_{1}$, from Figure 3.4. The elements of this single eigenvector, $\mathbf{u}_{1}$, are used to encode the vertex colormap, as shown in Figure $4.7(\mathrm{a})$. Here, the minimum element of $\mathbf{u}_{1}$ was used to select the red color (vertex 7 ), while the white color at vertex 0 was designated by the maximum value of this eigenvector. Despite its simplicity, this scheme immediately allows us to threshold $\mathbf{u}_{1}$ and identify two possible graph clusters, $\{0,1,2,3\}$, and $\{4,5,6,7\}$, as illustrated in Figure $4.7(\mathrm{~b})$. The same result would be obtained if the sign of $\mathbf{u}_{1}$ was used to color the vertices, and this would correspond to the minimum ratio cut clustering in Figure 4.3.

Algorithm 1. Clustering using the graph Laplacian.

## Input:

- Graph vertices $\mathcal{V}=\{0,1, \ldots, N-1\}$
- Graph Laplacian L

1: $[\mathbf{U}, \boldsymbol{\Lambda}] \leftarrow \operatorname{eig}(\mathbf{L})$
2: $y_{n} \leftarrow U(2, n)$
3: $\mathcal{E} \leftarrow\left\{n \mid y_{n}>0\right\}, \quad \mathcal{H} \leftarrow\left\{n \mid y_{n} \leq 0\right\}$

## Output:

- Vertex clusters $\mathcal{E}$ and $\mathcal{H}$


Figure 4.7: Vertex coloring for the graph from Figure 2.2, with its spectrum shown in Figure 3.4. (a) The eigenvector, $\mathbf{u}_{1}$, of the Laplacian matrix of this graph, given in (2.8), is normalized and is used to define the red color intensity levels within the colormap for every vertex. For this example, $\mathbf{u}_{1}=$ $[0.42,0.38,0.35,0.15,-0.088,-0.34,-0.35,-0.54]^{T}$. The largest element of this eigenvector is $u_{1}(0)=0.42$ at vertex 0 , which indicates that this vertex should be colored by the lowest red intensity (white), while the smallest element is $u_{1}(7)=-0.54$, so that vertex 7 is colored with the strongest red color intensity. (b) Simplified two-level coloring based on the sign of the elements of eigenvector $\mathbf{u}_{1}$.

The true indicator vector, $\mathbf{x}$, for the minimum ratio cut of this graph is presented in Figure 4.8(a). This vector is obtained by checking all the 127 possible cut combinations of $\mathcal{E}$ and $\mathcal{H}$ in this small graph, together with the corresponding $x(n)$. The signs of the elements of this vector indicate the way for optimal clustering into the subsets $\mathcal{E}=\{0,1,2,3\}$ and $\mathcal{H}=\{4,5,6,7\}$, while the minimum cut value is


Figure 4.8: Principle of the minimum ratio cut based clustering and its spectral (graph Laplacian eigenvector) based approximation; all vectors are plotted against the vertex index $n$. (a) The ideal indicator vector for a minimum ratio cut, $\operatorname{Cut} N(\mathcal{E}, \mathcal{H})$, normalized to unit energy. (b) The graph Laplacian eigenvector, $\mathbf{u}_{1}$. (c) The generalized eigenvector of the Laplacian, $\mathbf{u}_{1}$. (d) The eigenvector of the normalized Laplacian, $\mathbf{v}_{1}$. The eigenvectors in (c) and (d) are related as $\mathbf{u}_{1}=\mathbf{D}^{-1 / 2} \mathbf{v}_{1}$. In this case, the signs of the indicator vector and the eigenvectors, $\operatorname{sign}(\mathbf{x}), \operatorname{sign}\left(\mathbf{u}_{1}\right)$, and $\operatorname{sign}\left(\mathbf{v}_{1}\right)$ are the same in all the four vectors. The signs of these vectors then all may be used to define the minimum ratio cut based clustering into $\mathcal{E}$ and $\mathcal{H}$, that is, the association of a vertex, $n$, to either the subset $\mathcal{E}$ or subset $\mathcal{H}$.
$\operatorname{Cut} N(\mathcal{E}, \mathcal{H})=\mathbf{x}^{T} \mathbf{L} \mathbf{x}=0.395$. Figure $4.8(\mathrm{~b})$ shows an approximation of the indicator vector within the space of the graph Laplacian eigenvector, $\mathbf{u}_{1}$. The quadratic form of the eigenvector, $\mathbf{u}_{1}$, is equal to $\mathbf{u}_{1}^{T} \mathbf{L} \mathbf{u}_{1}=$ $\lambda_{1}=0.286$. As shown in (4.17), note that the true indicator vector, $\mathbf{x}$, can be decomposed into the set of all graph Laplacian eigenvectors, $\mathbf{u}_{k}$, and written as their linear combination.

The generalized Laplacian eigenvector, $\mathbf{u}_{1}=[0.37,0.24,0.32,0.13$, $-0.31,-0.56,-0.34,-0.58]$, which is an approximation of the indicator vector for the minimum volume normalized cut in (4.2), is presented in Figure 4.8(c). In this case, the generalized eigenvector indicates the same clustering subsets, $\mathcal{E}=\{0,1,2,3\}$ and $\mathcal{H}=\{4,5,6,7\}$. The eigenvector of the normalized Laplacian, $\mathbf{v}_{1}$, is shown in Figure 4.8(d).

Example 19: Consider the graph from Figure 2.2, with the weight matrix, $\mathbf{W}$, in (2.4), and the graph Laplacian eigenvector $\mathbf{u}_{1}$ (shown in Figure 3.4, Figure 4.4(b) (left), and Figure 4.8(b)). When this eigenvector is thresholded to only two intensity levels, $\operatorname{sign}\left(\mathbf{u}_{1}\right)$, two graph clusters are obtained, as shown in Figure 4.7(b). In an ideal case, these clusters may even be considered as independent graphs (graph segmentation being the strongest form of clustering); this can be achieved by redefining the weights as $W_{n m}=0$, if $m$ and $n$ are in different clusters, and $W_{n m}=W_{n m}$ otherwise ( Ng et al., 2002), for the corresponding disconnected (segmented) graph, whose weight matrix, $\hat{\mathbf{W}}$, is given by

$$
\hat{\mathbf{W}}=\begin{array}{r}
0  \tag{4.23}\\
1 \\
2 \\
3 \\
4 \\
5 \\
6 \\
7
\end{array}\left[\begin{array}{rrrrrrrr}
0 & 0.23 & 0.74 & 0.24 & 0 & 0 & 0 & 0 \\
0.23 & 0 & 0.35 & 0 & 0 & 0 & 0 & 0 \\
0.74 & 0.35 & 0 & 0.26 & 0 & 0 & 0 & 0 \\
0.24 & 0 & 0.26 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.51 & 0 & 0.14 \\
0 & 0 & 0 & 0 & 0.51 & 0 & 0 & 0.15 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.32 \\
0 & 0 & 0 & 0 & 0.14 & 0.15 & 0.32 & 0
\end{array}\right] .
$$

### 4.3.2 "Closeness" of the Segmented and Original Graphs

The issue of how "close" the behavior of the weight matrix of the segmented graph, $\hat{\mathbf{W}}$, in (4.23) (and the corresponding $\hat{\mathbf{L}}$ ) is to the
original $\mathbf{W}$ and $\mathbf{L}$, in (2.4) and (2.8), is usually considered within matrix perturbation theory.

It can be shown that a good measure of the "closeness" is the socalled eigenvalue gap, $\delta=\lambda_{2}-\lambda_{1}, \mathrm{Ng}$ et al. (2002), that is the difference between the eigenvalue $\lambda_{1}$ associated with the eigenvector $\mathbf{u}_{1}$, which is used for segmentation, and the next eigenvalue, $\lambda_{2}$, in the graph spectrum of the normalized graph Laplacian (for additional explanation see Example 23). For the obvious reason of analyzing the eigenvalue gap at an appropriate scale, we suggest to consider the relative eigenvalue gap

$$
\begin{equation*}
\delta_{r}=\frac{\lambda_{2}-\lambda_{1}}{\lambda_{2}}=1-\frac{\lambda_{1}}{\lambda_{2}} \tag{4.24}
\end{equation*}
$$

The relative eigenvalue gap value range is within the interval $0 \leq \delta_{r} \leq 1$, since the eigenvalues are nonnegative real-valued numbers sorted into a nondecreasing order. The value of this gap may be considered as large if it is close to the maximum eigengap value, $\delta_{r}=1$.
Example 20: The Laplacian eigenvalues for the graph in Figure 4.7 are $\lambda \in\{0,0.29,0.34,0.79,1.03,1.31,1.49,2.21\}$, with the relative eigenvalue gap, $\delta_{r}=\left(\lambda_{2}-\lambda_{1}\right) / \lambda_{2}=0.15$, which is not large and indicates that the segmentation in Example 19 is not "close".

As an illustration, consider three hypothetical but practically relevant scenarios: (i) $\lambda_{2}=0$ and $\lambda_{3}=1$, (ii) $\lambda_{2}=0$ and $\lambda_{3}=\varepsilon$, (iii) $\lambda_{2}=1$ and $\lambda_{3}=1+\varepsilon$, where $\varepsilon$ is small positive number and close to 0 . According to Remark 18, the graph in case (i) consists of exactly two disconnected components, and the subsequent clustering and segmentation is appropriate, with $\delta_{r}=1$. For the case (ii), the graph consists of more than two almost disconnected components and the clustering in two sets can be performed in various ways, with $\delta_{r}=1 / \varepsilon$. Finally, in the last scenario the relative gap is very small, $\delta_{r}=\varepsilon$, thus indicating that the behavior of the segmented graph is not "close" to the original graph, that is, $\hat{\mathbf{L}}$ is not "close" to $\mathbf{L}$, and thus any segmentation into two disconnected subgraphs would produce inadequate results.

Remark 28: The thresholding of elements of the Fiedler vector, $\mathbf{u}_{1}$, of the normalized graph Laplacian, $\mathbf{L}_{N}=\mathbf{D}^{-1 / 2} \mathbf{L} \mathbf{D}^{-1 / 2}$, performed in order to cluster the graph is referred to as the Shi-Malik algorithm (Shi and Malik, 2000; Weiss, 1999). Note that similar results would have been
obtained if clustering was based on the thresholding of elements of the smoothest eigenvector corresponding to the second largest eigenvalue of the normalized weight matrix, $\mathbf{W}_{N}=\mathbf{D}^{-1 / 2} \mathbf{W D}^{-1 / 2}$ (Perona-Freeman algorithm (Perona and Freeman, 1998; Weiss, 1999)). This becomes clear after recalling that the relation between the normalized weight and graph Laplacian matrices is given by

$$
\begin{align*}
& \mathbf{L}_{N}=\mathbf{D}^{-1 / 2} \mathbf{L} \mathbf{D}^{-1 / 2}=\mathbf{I}-\mathbf{D}^{-1 / 2} \mathbf{W} \mathbf{D}^{-1 / 2} \\
& \mathbf{L}_{N}=\mathbf{I}-\mathbf{W} \tag{4.25}
\end{align*}
$$

The eigenvalues of these two matrices are therefore related as $\lambda_{k}^{\left(L_{N}\right)}=$ $1-\lambda_{k}^{\left(W_{N}\right)}$, while they share the same eigenvectors.

## Clustering Based on More Than One Eigenvector

More complex clustering schemes can be achieved when using more than one Laplacian eigenvector. In turn, vertices with similar values of several slow-varying eigenvectors, $\mathbf{u}_{k}$, would exhibit high spectral similarity.

The concept of using more than one eigenvector in vertex clustering and possible subsequent graph segmentation was first introduced by Scott and Longuet-Higgins (1990). They used $k$ eigenvectors of the weight matrix $\mathbf{W}$ to form a new $N \times k$ matrix $\mathbf{V}$, for which a further row normalization was performed. Vertex clustering is then performed based on the elements of the matrix $\mathbf{V V}^{T}$.

For the normalized weight matrix, $\mathbf{W}_{N}$, the Scott and LonguetHiggins algorithm reduces to the corresponding analysis with $k$ eigenvectors of the normalized graph Laplacian, $\mathbf{L}_{N}$. Since $\mathbf{W}_{N}$ and $\mathbf{L}_{N}$ are related by (4.25), they thus have the same eigenvectors.

Example 21: Consider two independent ratio cuts of a graph, where the first cut splits the graph into the sets of vertices $\mathcal{E}_{1}$ and $\mathcal{H}_{1}$, and the second cut further splits all vertices into the sets $\mathcal{E}_{2}$ and $\mathcal{H}_{2}$, and define this two-level cut as

$$
\begin{equation*}
\operatorname{CutN2}\left(\mathcal{E}_{1}, \mathcal{H}_{1}, \mathcal{E}_{2}, \mathcal{H}_{2}\right)=\operatorname{Cut} N\left(\mathcal{E}_{1}, \mathcal{H}_{1}\right)+\operatorname{Cut} N\left(\mathcal{E}_{2}, \mathcal{H}_{2}\right) \tag{4.26}
\end{equation*}
$$

where both $\operatorname{Cut} N\left(\mathcal{E}_{i}, \mathcal{H}_{i}\right), i=1,2$, are defined by (28.7).

If we now introduce two indicator vectors, $\mathbf{x}_{1}$ and $\mathbf{x}_{2}$, for the two respective cuts, then, from (28.9) we may write

$$
\begin{equation*}
\operatorname{CutN2}\left(\mathcal{E}_{1}, \mathcal{H}_{1}, \mathcal{E}_{2}, \mathcal{H}_{2}\right)=\frac{\mathbf{x}_{1}^{T} \mathbf{L} \mathbf{x}_{1}}{\mathbf{x}_{1}^{T} \mathbf{x}_{1}}+\frac{\mathbf{x}_{2}^{T} \mathbf{L} \mathbf{x}_{2}}{\mathbf{x}_{2}^{T} \mathbf{x}_{2}} \tag{4.27}
\end{equation*}
$$

As mentioned earlier, finding the indicator vectors, $\mathbf{x}_{1}$ and $\mathbf{x}_{2}$, which minimize (4.27) is a combinatorial problem. However, if the space of solutions for the indicator vectors is now relaxed from the subset-wise constant form to the space spanned by the eigenvectors of the graph Laplacian, then the approximative minimum value of the two cuts, $\operatorname{CutN2}\left(\mathcal{E}_{1}, \mathcal{H}_{1}, \mathcal{E}_{2}, \mathcal{H}_{2}\right)$, is obtained for $\mathbf{x}_{1}=\mathbf{u}_{1}$ and $\mathbf{x}_{2}=\mathbf{u}_{2}$, since $\mathbf{u}_{1}$ and $\mathbf{u}_{2}$ are maximally smooth but not constant (for the proof see (4.31)-(4.32) and for the illustration see Example 22).

For the case of two independent cuts, for convenience, we may form the indicator $N \times 2$ matrix $\mathbf{Y}=\left[\mathbf{x}_{1}, \mathbf{x}_{2}\right]$, so that the corresponding matrix of the solution (within the graph Laplacian eigenvectors space) to the two ratio cuts minimization problem, has the form

$$
\mathbf{Q}=\left[\mathbf{u}_{1}, \mathbf{u}_{2}\right] .
$$

The rows of this matrix, $\mathbf{q}_{n}=\left[u_{1}(n), u_{2}(n)\right]$, are the spectral vectors which are assigned to each vertex, $n$.

The same reasoning can be followed for the cases of three or more independent cuts, to obtain an $N \times M$ indicator matrix $\mathbf{Y}=$ $\left[\mathbf{x}_{1}, \mathbf{x}_{2}, \ldots, \mathbf{x}_{M}\right]$ with the corresponding eigenvector approximation, $\mathbf{Q}$, the rows of which are the spectral vectors $\mathbf{q}_{n}=\left[u_{1}(n), u_{2}(n), \ldots, u_{M}(n)\right]$.

Remark 29: Graph clustering in the spectral domain may be performed by assigning the spectral vector,

$$
\mathbf{q}_{n}=\left[u_{1}(n), \ldots, u_{M}(n)\right]
$$

in (4.9), to each vertex, $n$, and subsequently by grouping the vertices with similar spectral vectors into the corresponding clusters (Belkin and Niyogi, 2003; Ng et al., 2002).

Low dimensional spectral vectors (up to $M=3$ ) can be represented by color coordinates of, for example, standard $R G B$ coloring system. To this


Figure 4.9: Spectral vertex clustering schemes for the graph from Figure 4.4. (a) The eigenvector, $\mathbf{u}_{1}$, of the Laplacian matrix (plotted in red lines on vertices designated by black dots) is first normalized and is then used to designate (b) a two-level blue colormap intensity (through its signs) for every vertex (blue-white circles). (c) The eigenvector, $\mathbf{u}_{2}$, of the Laplacian matrix is normalized and is then used to provide (d) a two-level green colormap intensity for every vertex. (e) The eigenvector, $\mathbf{u}_{3}$, of the Laplacian matrix is normalized and used as (f) a two-level red colormap intensity for every vertex. (g) Clustering based on the combination of the eigenvectors $\mathbf{u}_{1}$ and $\mathbf{u}_{2}$. (h) Clustering based on the combination of the eigenvectors $\mathbf{u}_{1}, \mathbf{u}_{2}$, and $\mathbf{u}_{3}$. Observe an increase in degrees of freedom with the number of eigenvectors used; this is reflected in the number of detected clusters, starting from two clusters in (b) and (d), via four clusters in (g), to 8 clusters in (h).
end, it is common to use different vertex colors, which represent different spectral vectors, for the visualization of spectral domain clustering.

Example 22: Figure 4.9 illustrates several spectral vector clustering schemes for the graph in Figure 4.4 (right), based on the three smoothest eigenvectors $\mathbf{u}_{1}, \mathbf{u}_{2}$, and $\mathbf{u}_{3}$. Clustering based on the eigenvector $\mathbf{u}_{1}$, with $\mathbf{q}_{n}=\left[u_{1}(n)\right]$, is shown in Figure $4.9(\mathrm{~b})$, clustering using the eigenvector $\mathbf{u}_{2}$ only, with $\mathbf{q}_{n}=\left[u_{2}(n)\right]$, is shown in Figure $4.9(\mathrm{~d})$, while Figure $4.9(\mathrm{e})$ illustrates the clustering based on the eigenvectors $\mathbf{u}_{3}$, when $\mathbf{q}_{n}=\left[u_{3}(n)\right]$. Clustering based on the combination of the two smoothest eigenvectors $\mathbf{u}_{1}$, and $\mathbf{u}_{2}$, with spectral vectors $\mathbf{q}_{n}=\left[u_{1}(n), u_{2}(n)\right]$, is shown in Figure $4.9(\mathrm{~g})$, while Figure $4.9(\mathrm{~h})$ illustrates clustering based on the three smoothest eigenvectors, $\mathbf{u}_{1}, \mathbf{u}_{2}$, and $\mathbf{u}_{3}$, whereby the spectral vector $\mathbf{q}_{n}=\left[u_{1}(n), u_{2}(n), u_{3}(n)\right]$. In all cases, two-level colormaps were used for each eigenvector. The smallest eigenvalues were $\lambda_{0}=0, \lambda_{1}=0.0286$, $\lambda_{2}=0.0358, \lambda_{3}=0.0899, \lambda_{4}=0.104$, and $\lambda_{5}=0.167$, so that the largest relative gap was obtained when $\mathbf{u}_{1}$ and $\mathbf{u}_{2}$ were used for clustering, with the corresponding eigenvalue gap of $\delta_{r}=1-\lambda_{2} / \lambda_{3}=0.6$.

Remark 30: $\boldsymbol{k}$-means algorithm. The above clustering schemes are based on the quantized levels of spectral vectors. These can be refined using the $k$-means algorithm, that is, through postprocessing in the form of unsupervised learning and in the following way.
(i) After an initial vertex clustering is performed by grouping the vertices into $\mathcal{V}_{i}, i=1,2, \ldots, k$ nonoverlapping vertex subsets, a new spectral vector centroid, $\mathbf{c}_{i}$, is calculated as

$$
\mathbf{c}_{i}=\operatorname{mean}_{n \in \mathcal{V}_{i}}\left\{\mathbf{q}_{n}\right\}
$$

for each cluster of vertices $\mathcal{V}_{i}$.
(ii) Every vertex, $n$, is then reassigned to its nearest (most similar) spectral domain centroid, $i$, where the spectral distance (spectral similarity) is calculated as $\left\|\mathbf{q}_{n}-\mathbf{c}_{i}\right\|_{2}$.

This two-step algorithm is iterated until no vertex changes clusters. Finally, all vertices in one cluster are colored based on the corresponding common spectral vector $\mathbf{c}_{i}$ (or visually, a color representing $\mathbf{c}_{i}$ ).

Clustering refinement using the $k$-means algorithm is illustrated later in Example 29.

Example 23: Graphs represent quite a general mathematical formalism, and we will here provide only one possible physical interpretation of graph clustering. Assume that each vertex represents one out of the set of $N$ images, which exhibit both common elements and individual differences. If the edge weights are calculated so as to represent mutual similarities between these images, then spectral vertex analysis can be interpreted as follows. If the set is complete and with very high similarity among all vertices, then $W_{m n}=1$, and $\lambda_{0}=0, \lambda_{1}=$ $N, \lambda_{2}=N, \ldots, \lambda_{N-1}=N$, as shown in Remark 19. The relative eigenvalue gap is then $\delta_{r}=\left(\lambda_{2}-\lambda_{1}\right) / \lambda_{2}=0$ and the segmentation is not possible.

Assume now that the considered set of images consists of two connected subsets with the respective numbers of $N_{1}$ and $N_{2} \geq N_{1}$ of very similar photos within each subset. In this case, the graph consists of two complete components (sub-graphs). According to Remarks 18 and 19, the graph Laplacian eigenvalues are now $\lambda_{0}=0, \lambda_{1}=0, \lambda_{2}=$ $N_{1}, \ldots, \lambda_{N_{1}}=N_{1}, \lambda_{N_{1}+1}=N_{2}, \ldots, \lambda_{N-1}=N_{2}$. Then, this graph may be well segmented into two components (sub-graphs) since the relative eigenvalue gap is now large, $\delta_{r}=\left(\lambda_{2}-\lambda_{1}\right) / \lambda_{2}=1$. Therefore, this case can be used for collaborative data processing within each of these subsets. The analysis can be continued and further refined for cases with more than one eigenvector and more than two subsets of vertices. Note that segmentation represents a "hard-thresholding" operation of cutting the connections between vertices in different subsets, while clustering refers to just a grouping of vertices, which exhibit some similarity, into subsets, while keeping their mutual connections.

Example 24: For enhanced intuition, we next consider a real-world dataset with eight images, shown in Figure 4.10. The connectivity weights were calculated using the structural similarity index (SSIM), with an appropriate threshold (Wang et al., 2003). The so obtained


Figure 4.10: A graph representation of a set of the real-world images which exhibit an almost constant background but different head orientation, which moves gradually from the left profile (bottom left) to the right profile (top right). The images serve as vertices, while the edges and the corresponding weight matrix are defined through the squared structural similarity index (SSIM) between images, with $W_{m n}=$ $\operatorname{SSIM}_{T}^{2}(m, n)$, and hard thresholded at 0.28 to account for the contribution of the background to the similarity index, that is, $\operatorname{SSIM}_{T}(m, n)=\operatorname{hard}(\operatorname{SSIM}(m, n), 0.53)$.
weight matrix, $\mathbf{W}$, is given by

$$
\mathbf{W}=\begin{array}{r}
0  \tag{4.28}\\
1 \\
2 \\
3 \\
4 \\
5 \\
6 \\
7
\end{array}\left[\begin{array}{rrrrrrrr}
0 & 0.49 & 0.33 & 0.29 & 0.31 & 0 & 0 & 0 \\
0.49 & 0 & 0.32 & 0 & 0.30 & 0 & 0 & 0.29 \\
0.33 & 0.32 & 0 & 0.37 & 0.30 & 0 & 0 & 0 \\
0.29 & 0 & 0.37 & 0 & 0.31 & 0 & 0 & 0 \\
0.31 & 0.30 & 0.30 & 0.31 & 0 & 0.31 & 0.30 & 0.29 \\
0 & 0 & 0 & 0 & 0.31 & 0 & 0.40 & 0.48 \\
0 & 0 & 0 & 0 & 0.30 & 0.40 & 0 & 0.64 \\
0 & 0.29 & 0 & 0 & 0.29 & 0.48 & 0.64 & 0 \\
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7
\end{array}\right] .
$$

The standard graph form for this real-world scenario in Figure 4.10 is shown in Figure 4.11, together with the corresponding image/vertex indexing. Notice the almost constant background in all eight images


Figure 4.11: Graph topology for the real-world images from Figure 4.10.
(the photos were taken in the wild by a "hand-held device"), and that the only differences between the images are in that the model gradually moved her head position from the left profile (bottom left) to the right profile (top right). Therefore, the two frontal face positions, at vertices $n=4$ and $n=0$, exhibit higher vertex degrees than the other head orientations, which exemplifies physical meaningfulness of graph representations. The normalized spectral vectors for this graph, $\mathbf{q}_{n}=\left[u_{1}(n)\right] /\left\|\left[u_{1}(n)\right]\right\|_{2}$ and $\mathbf{q}_{n}=\left[u_{1}(n), u_{2}(n)\right] /\left\|\left[u_{1}(n), u_{2}(n)\right]\right\|_{2}$ were obtained as the generalized eigenvectors of the graph Laplacian, and were used to define the coloring scheme for the graph clustering in Figure 4.12. Recall that similar vertex colors indicate spectral similarity of the images assigned to the corresponding vertices.

The eigenvalues of the graph Laplacian for this example are $\lambda_{k} \in\{0$, $0.42,1.12,1.63,1.68,1.89,2.31,2.42\}$. The largest relative eigenvalue gap is therefore between the eigenvalues $\lambda_{1}=0.42$ and $\lambda_{2}=1.12$, and indicates that the best clustering will be obtained in a one-dimensional spectral space (with clusters shown in Figure 4.12(a)). However, the value of such cut would be large, $\operatorname{Cut}(\{0,1,2,3,4\},\{5,6,7\})=1.19$, while the value of the ratio cut,

$$
\operatorname{Cut} N(\{0,1,2,3,4\},\{5,6,7\})=0.63 \sim \lambda_{1}=0.42,
$$



Figure 4.12: Graph clustering structure for the images from Figure 4.10. (a) Vertices are clustered (colored) using the row-normalized spectral Fiedler eigenvector to give the spectral vector $\mathbf{u}_{1}, \mathbf{q}_{n}=\left[u_{1}(n)\right] /\left\|\left[u_{1}(n)\right]\right\|_{2}$. (b) Clustering scheme whereby spectral values of vertices are calculated using the two smoothest eigenvectors, $\mathbf{q}_{n}=\left[u_{1}(n), u_{2}(n)\right]$, which are then employed to designate the colormap for the vertices. Recall that the so obtained similar vertex colors indicate spectral similarity of the images from Figure 4.10.
indicates that the connections between these two clusters are too significant for a segmented graph to produce a "close" approximation of the original graph with only two components (disconnected subgraphs). Given the gradual change in head orientation, this again conforms with physical intuition, and the subsequent clustering based on two smoothest eigenvectors, $\mathbf{u}_{1}$ and $\mathbf{u}_{2}$, yields three meaningful clusters of vertices corresponding to the "left head orientation" (red), "frontal head orientation" (two shades of pink), and "right head orientation" (yellow).

Example 25: Minnesota roadmap graph. Three eigenvectors of the graph Laplacian matrix, $\mathbf{u}_{2}, \mathbf{u}_{3}$, and $\mathbf{u}_{4}$, were used as the coloring templates to represent the spectral similarity and clustering in the benchmark Minnesota roadmap graph, shown in Figure 4.13. The eigenvectors $\mathbf{u}_{0}$ and $\mathbf{u}_{1}$ were omitted, since their corresponding eigenvalues are $\lambda_{0}=\lambda_{1}=0$ (due to an isolated vertex in the graph data which behaves as a graph component, see Remark 18). The full (nonquantized) colormap scale was used to color the vertices (that is, represent threedimensional spectral vectors). As elaborated above, regions where the vertices visually assume similar colors are also spectrally similar, and with similar behavior of the corresponding slow-varying eigenvectors.


Figure 4.13: Vertex coloring in the benchmark Minnesota road-map graph using the three smoothest Laplacian eigenvectors $\left\{\mathbf{u}_{2}, \mathbf{u}_{3}, \mathbf{u}_{4}\right\}$, as coordinates in the standard RGB coloring system (a three-dimensional spectral space with the spectral vector $\mathbf{q}_{n}=\left[u_{2}(n), u_{3}(n), u_{4}(n)\right]$ for every vertex, $\left.n\right)$. The vertices with similar colors are therefore also considered spectrally similar. Observe three different clusters, characterized by the shades of predominantly red, green, and blue color, that correspond to intensities defined by the eigenvectors $u_{2}(n), u_{3}(n)$, and $u_{4}(n)$.

Example 26: Brain connectivity graph. Figure 4.14 shows the benchmark Brain Atlas connectivity graph (Mijalkov et al., 2017, Rubinov and Sporns, 2010), for which the data is given in two matrices: "Coactivation matrix", $\hat{\mathbf{W}}$, and "Coordinate matrix". The "Coordinate matrix" contains the vertex coordinates in a three-dimensional Euclidean space, whereby the coordinate of a vertex $n$ is defined by the $n$-th row of the "Coordinate matrix", that is, $\left[x_{n}, y_{n}, z_{n}\right]$.

In our analysis, the graph weight matrix, $\mathbf{W}$, was empirically formed by:
(i) thresholding the "Coactivation matrix", $\hat{\mathbf{W}}$, to preserve only the strongest connections within this brain atlas, for example, those greater than $0.1 \max \left\{\hat{W}_{m n}\right\}$, as recommended in Rubinov and Sporns (2010);
(ii) only the edges between the vertices $m$ and $n$, whose Euclidean distance satisfies $d_{m n} \leq 20$ are kept in the graph representation.


Figure 4.14: Brain atlas (top) and its graph (bottom), with vertex coloring based on the three smoothest generalized eigenvectors, $\mathbf{u}_{1}, \mathbf{u}_{2}$, and $\mathbf{u}_{3}$, of graph Laplacian. The spectral vector, $\mathbf{q}_{n}=\left[u_{1}(n), u_{2}(n), u_{3}(n)\right]$ is employed as the coordinates in the RGB coloring scheme (Mijalkov et al., 2017; Rubinov and Sporns, 2010).

The elements, $W_{m n}$, of the brain graph weight matrix, $\mathbf{W}$, are therefore obtained from the corresponding elements, $\hat{W}_{m n}$, of the "Coactivation matrix" as

$$
W_{m n}= \begin{cases}\hat{W}_{m n}, & \text { if } \hat{W}_{m n}>0.1 \max \left\{\hat{W}_{m n}\right\} \text { and } d_{m n} \leq 20  \tag{4.29}\\ 0, & \text { elsewhere }\end{cases}
$$

The brain connectivity graph with the so defined weight matrix, $\mathbf{W}$, is shown in Figure 4.14(bottom).

The three smoothest generalized eigenvectors, $\mathbf{u}_{1}, \mathbf{u}_{2}$ and $\mathbf{u}_{3}$, of the corresponding graph Laplacian matrix, $\mathbf{L}=\mathbf{D}-\mathbf{W}$, were next used to define the spectral vectors

$$
\mathbf{q}_{n}=\left[u_{1}(n), u_{2}(n), u_{3}(n)\right]
$$

for each vertex, $n=0,1, \ldots, N-1$. The elements of this spectral vector, $\mathbf{q}_{n}$, were then used to designate the corresponding RGB coordinates for the coloring of the vertices of the brain graph, as shown in Figure 4.14.

### 4.4 Vertex Dimensionality Reduction Using the Laplacian Eigenmaps

We have seen that graph clustering can be used for collaborative processing on the set of data which is represented by the vertices within a cluster. In general, any form of the presentation of a graph and its corresponding vertices, that employs the eigenvectors of the graph Laplacian may be considered as a Laplacian eigenmap. The idea which underpins eigenmap-based approaches presented here is to employ spectral vectors, $\mathbf{q}_{n}$, to define the new positions of the original vertices in such a "transform-domain" space so that spectrally similar vertices appear spatially closer than in the original vertex space.
Remark 31: The Laplacian eigenmaps may be employed for vertex dimensionality reduction, while at the same time preserving the local properties and natural connections within the original graph (Belkin and Niyogi, 2003).

Consider a vertex $n, n=0,1, \ldots, N-1$, which resides in an $L$-dimensional space $\mathbb{R}^{L}$, at the position defined by the $L$-dimensional
vector $\mathbf{r}_{n}$. A spectral vector for vertex $n$ is then defined in a new lowerdimensional ( $M$-dimensional) space, with $M<N$, by keeping the $M$ smoothest eigenvectors of graph Laplacian, $\mathbf{u}_{0}, \mathbf{u}_{1}, \ldots, \mathbf{u}_{M}$. Upon omitting the constant eigenvector, $\mathbf{u}_{0}$, this gives the new basis designated by the spectral vector

$$
\begin{equation*}
\mathbf{q}_{n}=\left[u_{1}(n), \ldots, u_{M}(n)\right] \tag{4.30}
\end{equation*}
$$

Since $M<L$, this provides the desired dimensionality reduction of the vertex space. The concepts of spectral vector-based vertex dimensionality reduction, and physical meaning associated with the spectral vector space representation are illustrated in the next example.

Example 27: Vertex dimensionality reduction. Consider a set of $N=70$ students and their marks in 40 lecture courses. Every student can be considered as a vertex located in the original $L=40$ dimensional space at the position $\mathbf{r}_{n}$, where $r_{n}(k)$ is a mark for the $n$-th student at $k$-th course. Assume that the marks are within the set $\{2,3,4,5\}$ and that some students have affinity to certain subsets of courses (for example, social sciences, natural sciences and skills). This set-up can be represented in a tabular $(70 \times 40)$ compact matrix form as in Figure $4.15(\mathrm{a})$, where the columns contain the marks for every student (the marks are color coded).

The average marks per student and per course are shown in Figures 4.15 (b) and (c). Observe the limitations of this representation, as for example, the average marks cannot be used to determine student affinities to the subsets of their courses.

We can now create a graph representation by connecting with edges students with similar marks. In our example, the edge weights were determined through a distance in the 40-dimensional feature (marks) space, as

$$
W_{m n}= \begin{cases}e^{-\left\|\mathbf{r}_{m}-\mathbf{r}_{n}\right\|_{2}^{2} / 70}, & \text { for }\left\|\mathbf{r}_{m}-\mathbf{r}_{n}\right\|_{2} \geq 7 \\ 0, & \text { otherwise }\end{cases}
$$

With the so obtained connectivity, this graph is presented in Figure $4.15(\mathrm{~d})$, whereby the vertices (students) are randomly positioned in a plane and connected with edges. We shall now calculate the


Figure 4.15: Illustration of spectral dimensionality reduction through an example of exam marks for a cohort of students. (a) Each of the 70 columns (students) represents a 40-dimensional vector with student marks. Therefore the dimensionality of the original representation space is $L=40$. (b) Average mark per student. (c) Average mark per course. (d) Two-dimensional graph representation of the matrix in (a), where the individual students are represented by randomly positioned vertices in the plane. To perform vertex (student) dimensionality reduction we can use spectral vectors to reduce their original $L=40$ dimensional representation space to (e) $M=3$, (f) $M=2$, and (g) $M=1$ dimensional spectral representation spaces. (h) Vertices from path graph (g) positioned on a circle (by connecting the ends of the line) which allows us to also show the edges.
normalized Laplacian eigenvectors and remap the vertices according to the three-dimensional, two-dimensional and one-dimensional spectral vectors, $\mathbf{q}_{n}$, defined by (4.30) that is, for $M=3, M=2$, and $M=1$. In this way, the vertex dimensionality is reduced from the original $L=40$ to a much lower $M \ll L$. The corresponding graph representations are respectively shown in Figures $4.15(\mathrm{e})-(\mathrm{g})$. For $M=2$ and $M=3$ we can now clearly divide students into the three affinity groups (designated by the red, blue, and black). Although the obtained groups (clusters) are logically ordered even in the one-dimensional case in Figure 4.15(g), observe that we cannot use $M=1$ for precise grouping since there is no enough gap between the groups. However, even in this case, if we re-cast the vertices on a circle instead on a line (by connecting two ends of a line), and draw the connecting edges (the same edges as in Figures $4.15(\mathrm{~d})-(\mathrm{f}))$ we can see the benefit of a graph representation even after such a radical dimensionality reduction.

The dimensionality reduction principle can also be demonstrated based on Example 24, whereby each vertex is a $640 \times 480$ RGB color image which can be represented as a vector in the $L=640 \times 480 \times 3=$ 921600 dimensional space. Indeed, using spectral vectors with $M=2$, this graph can be presented in a two-dimensional space as in Figure 4.10.

Within the Laplacian eigenmaps method, we may use any of the three forms of graph Laplacian eigenvectors introduced in Section 4.2.3. The relations among these three presentations are explained in Section 4.2.3 and Table 4.1. A unified algorithm for all three variants of the Laplacian eigenmaps, and the corresponding clustering methods, is given in Algorithm 9 in the appendix.

Remark 32: The Laplacian eigenmaps are optimal in the sense that they minimize an objective function which penalizes for the distance between the neighboring vertices in the spectral space. This ensures that if the vertices at the positions $\mathbf{r}_{m}$ and $\mathbf{r}_{n}$ in the original high-dimensional L-dimensional space are "close" in the sense of some data association metric, then they will also be close in the Euclidean sense in the reduced $M$-dimensional spectral space, where their positions are defined by the corresponding spectral vectors, $\mathbf{q}_{m}$ and $\mathbf{q}_{n}$.

### 4.4.1 Euclidean Distances in the Space of Spectral Vectors

We shall prove the "distance preserving" property of the above spectral mapping in an inductive way. Assume that a graph is connected, i.e., $\lambda_{1} \neq 0$. The derivation is based on the quadratic form in (4.8)

$$
\mathbf{u}_{k}^{T} \mathbf{L} \mathbf{u}_{k}=\frac{1}{2} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1}\left(u_{k}(m)-u_{k}(n)\right)^{2} W_{m n}
$$

which states that $\mathbf{u}_{k}^{T} \mathbf{L} \mathbf{u}_{k}$ is equal to the weighted sum of squared Eu clidean distances between the elements of the $m$-th and $n$-th eigenvector at vertices $m$ and $n$, for all $m$ and $n$. Recall that $\mathbf{u}_{k}^{T} \mathbf{L} \mathbf{u}_{k}$ is also equal to $\lambda_{k}$, by definition (see the elaboration after (4.6)).

Single-dimensional case. To reduce the original $L$-dimensional vertex space to a single-dimensional path graph with vertex coordinates $\mathbf{q}_{n}=u_{k}(n)$, the minimum sum of the weighted squared distances between the vertices $m$ and $n$, that is

$$
\begin{aligned}
& \frac{1}{2} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1}\|\mathbf{q}(m)-\mathbf{q}(n)\|_{2}^{2} W_{m n} \\
& \quad=\frac{1}{2} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1}\left(u_{k}(m)-u_{k}(n)\right)^{2} W_{m n}=\lambda_{k}
\end{aligned}
$$

will be obtained with the new positions of vertices, designated by $\mathbf{q}_{n}=\left[u_{1}(n)\right]$, and for $k=1$, since $\min _{k, \lambda_{k} \neq 0}\left\{\lambda_{k}\right\}=\lambda_{1}$ is the smallest nonzero eigenvalue.

Two-dimensional case. If we desire to reduce the original $L$-dimensional vertex representation space to a two-dimensional spectral space, designated by $\mathbf{q}_{n}=\left[u_{k}(n), u_{l}(n)\right]$ and defined through any two eigenvectors of the graph Laplacian, $\mathbf{u}_{k}$ and $\mathbf{u}_{l}$, then the minimum sum of the weighted squared distances between all vertices, $m$ and $n$, given by

$$
\begin{aligned}
& \frac{1}{2} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1}\left\|\mathbf{q}_{m}-\mathbf{q}_{n}\right\|_{2}^{2} W_{m n} \\
& \quad=\frac{1}{2} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1}\left(u_{k}(m)-u_{k}(n)\right)^{2} W_{m n}
\end{aligned}
$$

$$
\begin{align*}
& +\frac{1}{2} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1}\left(u_{l}(m)-u_{l}(n)\right)^{2} W_{m n} \\
= & \mathbf{u}_{k}^{T} \mathbf{L} \mathbf{u}_{k}+\mathbf{u}_{l}^{T} \mathbf{L} \mathbf{u}_{l}=\lambda_{k}+\lambda_{l} \tag{4.31}
\end{align*}
$$

will be obtained with the new spectral positions, $\mathbf{q}_{n}=\left[u_{k}(n), u_{l}(n)\right]$, such that $\mathbf{q}_{n}=\left[u_{1}(n), u_{2}(n)\right]$, since

$$
\begin{equation*}
\min _{k, l, k \neq l, k l \neq 0}\left\{\lambda_{k}+\lambda_{l}\right\}=\lambda_{1}+\lambda_{2} \tag{4.32}
\end{equation*}
$$

for nonzero $k$ and $l$, and keeping in mind that $\lambda_{1} \leq \lambda_{2} \leq \lambda_{3} \leq \cdots \leq$ $\lambda_{N-1}$. The same reasoning holds for new three- and higher-dimensional spectral representation spaces for the vertices, which yields (4.30) as the optimal vertex positions in the reduced $M$-dimensional vertex space.

The same relations hold for both the generalized eigenvectors of the Laplacian, defined by $\mathbf{L} \mathbf{u}_{k}=\lambda_{k} \mathbf{D} \mathbf{u}_{k}$, and the eigenvectors of the normalized Laplacian, defined by $\mathbf{D}^{-1 / 2} \mathbf{L} \mathbf{D}^{-1 / 2} \mathbf{v}_{k}=\lambda_{k} \mathbf{v}_{k}$. The only difference is in their respective normalization conditions, $\mathbf{u}_{k}^{T} \mathbf{D} \mathbf{u}_{k}$ and $\mathbf{v}_{k}^{T} \mathbf{v}_{k}$. The relation between the eigenvectors of the normalized graph Lapla$\operatorname{cian}, \mathbf{v}_{k}$, and the generalized eigenvectors of the graph Laplacian, $\mathbf{u}_{k}$, in the form $\mathbf{u}_{k}=\mathbf{D}^{-1 / 2} \mathbf{v}_{k}$, follows from their definitions (see Remark 27). Since the elements $u_{1}(n)$ and $u_{2}(n)$ are obtained by multiplying the elements $v_{1}(n)$ and $v_{2}(n)$ by the same value, $1 / D_{n n}$, that is, $\left[u_{1}(n), u_{2}(n)\right]=\left[v_{1}(n), v_{2}(n)\right] / D_{n n}$, their normalized forms of $\mathbf{u}_{k}$ and $\mathbf{v}_{k}$ are identical,

$$
\frac{\mathbf{q}_{n}}{\left\|\mathbf{q}_{n}\right\|_{2}}=\frac{\left[u_{1}(n), u_{2}(n)\right]}{\left\|\left[u_{1}(n), u_{2}(n)\right]\right\|_{2}}=\frac{\left[v_{1}(n), v_{2}(n)\right]}{\left\|\left[v_{1}(n), v_{2}(n)\right]\right\|_{2}}
$$

### 4.4.2 Examples of Graph Analysis in the Spectral Space

Example 28: The graph from Figure 2.2, where the vertices reside in a two-dimensional plane, is shown in Figure 4.16(a), while Figure 4.16(b) illustrates the same graph but represented in a reduced single-dimensional vertex space (a line). The vertex positions on the line are defined by the spectral vector, $\mathbf{q}_{n}=\left[u_{1}(n)\right]$, with $\mathbf{u}_{1}=[0.42$, $0.38,0.35,0.15,-0.088,-0.34,-0.35,-0.54]^{T}$.


Figure 4.16: Principle of vertex dimensionality reduction based on the spectral vectors. (a) The weighted graph from Figure 2.2 with its vertices in a two-dimensional space. (b) The graph from (a) with its vertices located along a line (one-dimensional vertex space), whereby the positions on the line are defined by the one-dimensional spectral vector, $\mathbf{q}_{n}=\left[u_{1}(n)\right]$, with $\mathbf{u}_{1}=[0.42,0.38,0.35,0.15,-0.088,-0.34,-0.35$, $-0.54]^{T}$. Observe that this dimensionality reduction method may be used for clustering, based on the vertex position on the line.

Remark 33: After the vertices are reordered according to the Fiedler eigenvector, $\mathbf{u}_{1}$, Example 28 indicates the possibility of clustering refinement through a recalculation of ratio cuts. For the set of vertices $\mathcal{V}=\{0,1,2, \ldots, N-1\}$, Figure 4.16(b) illustrates their ordering along a line, with the new order $\left\{v_{1}, v_{2}, \ldots, v_{N}\right\}=\{7,6,5,4,3,2,1,0\}$. Instead of using the sign of $\mathbf{u}_{1}$ to cluster the vertices, we can recalculate the ratio cuts, $\operatorname{Cut} N\left(\mathcal{E}_{p}, \mathcal{H}_{p}\right)$, with this sequential vertex order, where $\mathcal{E}_{p}=$ $\left\{v_{1}, v_{2}, \ldots, v_{p}\right\}$ and $\mathcal{H}_{p}=\left\{v_{p+1}, v_{p+2}, \ldots, v_{N}\right\}$, for $p=1,2, \ldots, N-1$. The estimation of the minimum ratio cut then becomes

$$
\left(\mathcal{E}_{p}, \mathcal{H}_{p}\right)=\arg \min _{p}\left\{C u t N\left(\mathcal{E}_{p}, \mathcal{H}_{p}\right)\right\}
$$

This approximation of the Cheeger's cut can also be written using the thresholding of the eigenvector $\mathbf{u}_{1}$, by a threshold $t$, as

$$
\begin{equation*}
\phi^{*}(\mathcal{V})=\min _{t}\left\{\frac{1}{\min \left\{N_{\mathcal{E}_{t}}, N_{\mathcal{V}-\mathcal{E}_{t}}\right\}} \sum_{\substack{m \in \mathcal{E}_{\mathcal{E}} \\ n \in \mathcal{V}-\mathcal{E}_{t}}} W_{m n}\right\}, \tag{4.33}
\end{equation*}
$$

where the vertex $n$ belongs to $\mathcal{E}_{t}$ if $u_{1}(n)>t$.
This method is computationally efficient since only ( $N-1$ ) cuts, $\operatorname{Cut} N\left(\mathcal{E}_{p}, \mathcal{H}_{p}\right)$, need to be calculated. In addition, the cuts $\operatorname{Cut} N\left(\mathcal{E}_{p}, \mathcal{H}_{p}\right)$ can be calculated recursively, using the previous $\operatorname{Cut} N\left(\mathcal{E}_{p-1}, \mathcal{H}_{p-1}\right)$ and the connectivity parameters (degree, $D_{p p}$, and weights, $W_{p m}$ ) of vertex $p$. Any normalized cut form presented in Section 4.1 can also be used instead of $\operatorname{Cut} N\left(\mathcal{E}_{p}, \mathcal{H}_{p}\right)$. When the Cheeger ratio, defined in (4.5), is used in this minimization, then an upper bound on the cut can be obtained as Trevisan (2013)

$$
\begin{equation*}
\min _{p}\left\{\phi\left(\mathcal{E}_{p}\right)\right\} \leq \sqrt{2 \lambda_{1}} \leq 2 \sqrt{\phi(\mathcal{V})}, \tag{4.34}
\end{equation*}
$$

where $\phi(\mathcal{V})$ denotes the combinatorial (true) minimum cut, with bounds given in (4.18).
Example 29: We shall now revisit the graph in Figure 4.9 and examine the clustering schemes based on: (i) standard Laplacian eigenvectors (Figure 4.17), (ii) generalized eigenvectors of graph Laplacian (Figure 4.18), and (iii) eigenvectors of the normalized Laplacian (Figure 4.19). Figure 4.17(b) illustrates Laplacian eigenmaps based dimensionality reduction for the graph from Figure 4.9(g), with the two eigenvectors, $\mathbf{u}_{1}$ and $\mathbf{u}_{2}$, serving as new vertex coordinates, and using the same vertex coloring scheme as in Figure 4.9(g). While both the original and the new vertex space are two-dimensional, we can clearly see that in the new vertex space the vertices belonging to the same clusters are also spatially closer, which is both physically meaningful and exemplifies the practical value of the eigenmaps. Figure 4.17(c) is similar to Figure 4.17 (b) but is presented using the normalized spectral space coordinates, $\mathbf{q}_{n}=\left[u_{1}(n), u_{2}(n)\right] /\left\|\left[u_{1}(n), u_{2}(n)\right]\right\|_{2}$. In Figure $4.17(\mathrm{~d})$ the clusters are refined using the $k$-means algorithm, as per Remark 30. The same representations are repeated and shown in


Figure 4.17: Principle of Laplacian eigenmaps and clustering based on the eigenvectors of the graph Laplacian, $\mathbf{L}$. (a) The original graph from Figure 4.9, with the spectral vector $\mathbf{q}_{n}=\left[u_{1}(n), u_{2}(n)\right]$, defined by the graph Laplacian eigenvectors $\left\{\mathbf{u}_{1}, \mathbf{u}_{2}\right\}$, which is used to cluster (color) the vertices. (b) Two-dimensional vertex positions obtained through Laplacian eigenmaps, with the spectral vector $\mathbf{q}_{n}=\left[u_{1}(n), u_{2}(n)\right]$ serving as the vertex coordinates (the 2D Laplacian eigenmap). While both the original and this new vertex space are two-dimensional, the new eigenmaps-based space is advantageous in that it emphasizes vertex spectral similarity in a spatial way (physical closeness of spectrally similar vertices). (c) The graph from (b) but produced using normalized spectral space coordinates $\mathbf{q}_{n}=\left[u_{1}(n), u_{2}(n)\right] /\left\|\left[u_{1}(n), u_{2}(n)\right]\right\|_{2}$, as in (4.22). (d) The graph from (c) with clusters refined using the $k$-means algorithm, as per Remark 30. The centroids of clusters are designated by squares of the same color. The complexity of graph presentation is also significantly reduced through eigenmaps, with most of the edges between strongly connected vertices being very short and located along a circle.


Figure 4.18: Principle of Laplacian eigenmaps and clustering based on the generalized eigenvectors of the graph Laplacian, obtained as a solution to $\mathbf{L} \mathbf{u}_{k}=\lambda_{k} \mathbf{D u} \mathbf{u}_{k}$. Vertex coloring was produced using the same procedure as in Figure 4.17.

Figures 4.18(a)-(d) for the representation based on the generalized eigenvectors of the graph Laplacian, obtained as a solution to $\mathbf{L u} \mathbf{u}_{k}=\lambda_{k} \mathbf{D} \mathbf{u}_{k}$. Finally, in Figures 4.19(a)-(d), the Laplacian eigenmaps and clustering are produced based on the eigenvectors of the normalized graph Laplacian, $\mathbf{L}_{N}=\mathbf{D}^{-1 / 2} \mathbf{L} \mathbf{D}^{-1 / 2}$. As expected, the eigenmaps obtained using the generalized Laplacian eigenvectors, in Figure 4.19(b), and the eigenvectors of the normalized Laplacian, in Figure 4.18(b), are different; however, they reduce to the same eigenmaps after spectral


Figure 4.19: Principle of Laplacian eigenmaps and clustering based on the eigenvectors of the normalized graph Laplacian, $\mathbf{L}_{N}=\mathbf{D}^{-1 / 2} \mathbf{L D}^{-1 / 2}$. Vertex coloring was performed using the same procedure as in Figure 4.17. The eigenvectors of the normalized graph Laplacian, $\mathbf{v}_{k}$, are related to the generalized eigenvectors of the graph Laplacian, $\mathbf{u}_{k}$, through $\mathbf{u}_{k}=\mathbf{D}^{-1 / 2} \mathbf{v}_{k}$, as stated in Remark 27. This means that the signs of these two eigenvectors are the same, $\operatorname{sign}\left(\mathbf{u}_{k}\right)=\operatorname{sign}\left(\mathbf{v}_{k}\right)$. Since in order to obtain $u_{1}(n)$ and $u_{2}(n)$, the elements $v_{1}(n)$ and $v_{2}(n)$ are multiplied by the same value, $1 / D_{n n}$, then $\left[u_{1}(n), u_{2}(n)\right] /\left\|\left[u_{1}(n), u_{2}(n)\right]\right\|_{2}=\left[v_{1}(n), v_{2}(n)\right] /\left\|\left[v_{1}(n), v_{2}(n)\right]\right\|_{2}$, thus yielding the same graph forms in (c) and (d) in both this figure and in Figure 4.18.
vector normalization, as shown Figure 4.19(c) and Figure 4.18(c). After the $k$-means based clustering refinement was applied, in all three cases two vertices switched their initial color (cluster), as shown in Figures 4.17(d), 4.18(d), and 4.19(d).

Observe that the eigenmaps obtained with the normalized forms of the generalized eigenvectors of the Laplacian and the eigenvectors of the normalized Laplacian are the same, and in this case their clustering performances are similar to those based on the eigenmaps produced with the eigenvectors of the original Laplacian.

Remark 34: In general, an independent quantization of two smoothest eigenvectors of the graph Laplacian, $\mathbf{u}_{1}$ and $\mathbf{u}_{2}$, will produce four clusters. However, that will not be the case if we analyze the graph with an almost ideal eigenvalue gap (unit value) between $\lambda_{2}$ and $\lambda_{3}$. In other words, when the gap $\delta_{r}=1-\lambda_{2} / \lambda_{3}$ tends to 1 , that is, $\lambda_{2} \rightarrow 0$ and $\lambda_{1}<\lambda_{2} \rightarrow 0$, then this case corresponds to a graph with exactly three disjoint subgraph components, with vertices belonging to the disjoint sets $\mathcal{E}, \mathcal{H}$, and $\mathcal{K}$. Without loss of generality, assume $N_{\mathcal{E}}>N_{\mathcal{H}}>N_{\mathcal{K}}$. The minimum ratio cut, $\operatorname{Cut} N(\mathcal{E}, \mathcal{H} \cup \mathcal{K})$ is then obtained with the first indicator vector $x_{1}(n)=c_{11}$ for $n \in \mathcal{E}$ and $x_{1}(n)=c_{12}$ for $n \in \mathcal{H} \cup \mathcal{K}$. The second indicator vector will produce the next minimum ratio cut, $\operatorname{Cut} N(\mathcal{E} \cup \mathcal{K}, \mathcal{H})$ with $x_{2}(n)=c_{21}$ for $n \in \mathcal{E} \cup \mathcal{K}$ and $x_{2}(n)=c_{22}$ for $n \in \mathcal{H}$. Following the same analysis as in the case of one indicator vector and the cut of graph into two disjoint subsets of vertices, we can immediately conclude that the two smoothest eigenvectors, $\mathbf{u}_{1}$ and $\mathbf{u}_{2}$, which correspond to $\lambda_{2} \rightarrow 0$ and $\lambda_{1} \rightarrow 0$, can be used to form an indicator matrix $\mathbf{Y}=\left[\mathbf{x}_{1}, \mathbf{x}_{2}\right]$, so that the corresponding matrix of the solution (within the graph Laplacian eigenvector space) to the minimization problem of two ratio cuts, has the form $\left[\operatorname{sign}\left(\mathbf{u}_{1}\right), \operatorname{sign}\left(\mathbf{u}_{2}\right)\right]$. The elements of these indicator vectors, $\left[\operatorname{sign}\left(u_{1}(n)\right), \operatorname{sign}\left(u_{2}(n)\right)\right]$, have therefore a subset-wise constant vector form, assuming exactly three different vector values that correspond to individual disjoint sets $\mathcal{E}, \mathcal{H}$, and $\mathcal{K}$.

This procedure can be generalized up to every individual vertex becoming a cluster (no clustering). To characterize $N$ independent disjoint sets we will need $(N-1)$ spectral vectors, if the constant eigenvector, $\mathbf{u}_{0}$, is omitted.
Example 30: The two-dimensional Laplacian eigenmap for the benchmark Minnesota roadmap graph (with $M=2$ ) is given in Figure 4.20. In this new space, the spectral vectors $\mathbf{q}_{n}=\left[u_{2}(n), u_{3}(n)\right]$, are used as


Figure 4.20: Laplacian eigenmaps for the Minnesota road-map graph, produced based on the new two-dimensional spectral vertex positions defined by the Laplacian eigenvectors $\left\{\mathbf{u}_{2}, \mathbf{u}_{3}\right\}$ as the vertex coordinates (the 2D Laplacian eigenmap).
the coordinates of the new vertex positions. Here, two vertices with similar slow-varying eigenvectors are located close to one another in the new coordinate system defined by $\mathbf{u}_{2}$ and $\mathbf{u}_{3}$. This illustrates that the eigenmaps can be considered as a basis for "scale-wise" graph representation.

Example 31: The Laplacian eigenmaps of the Brain Atlas graph from Figure 4.14, whose original vertex locations reside in an $L=3$ dimensional space, is presented in a new reduced $M=2$ dimensional space which is defined based on the two smoothest eigenvectors, $\mathbf{u}_{1}$
and $\mathbf{u}_{2}$. This example of vertex dimensionality reduction, with new vertex locations but with the original edges kept, is shown in Figure 4.21.

The generalized eigenvectors of the graph Laplacian, $\mathbf{u}_{k}$, for $k=$ $1,2,3,4,5,6$, are shown in Figure 4.22(a) using the standard colormap in both the original three-dimensional and the reduced two-dimensional space, as shown in Figure 4.22(b).

Example 32: Vertices of a three-dimensional Swiss roll graph are shown in Figure 4.24(a). The vertex locations in this original $L=3$ dimensional space are calculated as $x_{n}=\alpha_{n} \cos \left(\alpha_{n}\right) /(4 \pi), y_{n}=\beta_{n}$, and $z_{n}=\alpha_{n} \sin \left(\alpha_{n}\right) /(4 \pi), n=0,1,2, \ldots, N-1$, with $\alpha_{n}$ randomly taking values between $\pi$ and $4 \pi$, and $\beta_{n}$ from -1.5 to 1.5. The edge weights are calculated using $W_{m n}=\exp \left(-d_{m n}^{2} /\left(2 \kappa^{2}\right)\right)$, where $d_{m n}$ is the square Euclidean distance between the vertices $m$ and $n$, and $W_{m n}=0$ if $d_{m n} \geq 0.15$ with $\kappa=0.1$. The resulting three-dimensional Swiss roll graph is shown in Figure 4.24(b), while Figure 4.24(c) shows the same graph but with vertices colored (clustered) using the normalized graph Laplacian eigenvectors, $u_{1}(n)$ and $u_{2}(n)$, as a colormap. The same vectors are then used in Figure $4.24(\mathrm{~d})$ as the new coordinates in the reduced two-dimensional Laplacian eigenmap vertex space ( $M=2$ ) for the Swiss roll graph.

### 4.5 Pseudo-Inverse of Graph Laplacian-Based Mappings

The graph Laplacian is a singular matrix $\left(\right.$ since $\left.\lambda_{0}=0\right)$ for which an inverse does not exist. To deal with this issue, the pseudo-inverse of the graph Laplacian, $\mathbf{L}^{+}$, is defined as a matrix that satisfies the property

$$
\mathbf{L L}^{+}=\left[\begin{array}{ll}
0 & \mathbf{0}_{1 \times(N-1)}  \tag{4.35}\\
\mathbf{0}_{(N-1) \times 1} & \mathbf{I}_{(N-1) \times(N-1)}
\end{array}\right]
$$

where we assumed that the graph is connected. The eigenvalues of the graph Laplacian pseudo-inverse are therefore the inverses of the original eigenvalues, $\left\{0,1 / \lambda_{1}, \ldots, 1 / \lambda_{N-1}\right\}$, while it shares the same eigenvectors with the original graph Laplacian, $\mathbf{u}_{0}, \mathbf{u}_{1}, \ldots, \mathbf{u}_{N-1}$. The eigenmaps for which the spectral coordinates are scaled based on the eigenvalues of the pseudo-inverse of graph Laplacian can be interpreted within the Principal Component Analysis (PCA) framework in the following way.


Figure 4.21: Brain atlas representation based on normalized spectral vectors. (a) A two-dimensional Laplacian eigenmap based on the generalized Laplacian eigenvectors. The original $L=3$ dimensional graph from Figure 4.14 is reduced to a two-dimensional representation based on the two smoothest eigenvectors, $u_{1}(n)$ and $u_{2}(n)$, which both serve as spectral coordinates and define color templates in the colormap, as in Figure 4.14. (b) Eigenmaps from (a) but in the space of normalized spectral space coordinates, $\mathbf{q}_{n}=\left[u_{2}(n), u_{3}(n)\right] /\left\|\left[u_{2}(n), u_{3}(n)\right]\right\|_{2}$, with the complexity of graph representation now significantly reduced. Observe that most edges only exists between strongly connected vertices located along the circle.


Figure 4.22: Generalized eigenvectors, $\mathbf{u}_{k}, k=1,2,3,4,5,6$, of the graph Laplacian of the Brain Atlas graph, shown using vertex coloring in the original three-dimensional vertex space. Each panel visualizes a different $\mathbf{u}_{k}, k=1,2,3,4,5,6$.


Figure 4.23: Laplacian eigenmaps of the Brain Atlas graph in the reduced twodimensional space defined by the two smoothest generalized eigenvectors of the graph Laplacian, $\mathbf{u}_{1}$ and $\mathbf{u}_{2}$. The panels each visualize a different generalized eigenvector, $\mathbf{u}_{k}, k=1,2,3,4,5,6$.


Figure 4.24: Laplacian eigenmaps based dimensionality reduction for the Swiss roll graph. (a) Vertex locations for the Swiss roll graph in the original $L=3$ dimensional space with $N=500$ points (vertices). (b) The Swiss roll graph with edges whose weights are calculated based on the Euclidean distances between vertices. (c) The Swiss roll graph with vertices colored using the normalized graph Laplacian eigenvectors, $u_{1}(n)$ and $u_{2}(n)$, as a colormap. (d) The same vectors are used as the new coordinates (spectral vectors) in a reduced two-dimensional Laplacian eigenmap vertex space $(M=2)$. The vertices with high similarity (similar values of the smoothest eigenvectors) are located close to one another, thus visually indicating the expected similarity of data observed at these vertices. (e) Clustering of the Swiss roll graph, in the original $L=3$ dimensional space, using the two smoothest eigenvectors, $u_{1}(n)$ and $u_{2}(n)$. (f) Clustering of the Swiss roll graph using the two smoothest eigenvectors, $u_{1}(n)$ and $u_{2}(n)$, presented in the $M=2$ eigenmap space, where for every vertex its spatial position (quadrant of the coordinate system) indicates the cluster where it belongs.

Notice that the $M$-dimensional eigenmaps based on the pseudoinverse of the Laplacian are the same as those for the original graph Laplacian, since they share the same eigenvectors. If the spectral vectors $\mathbf{q}_{n}=\left[u_{1}(n), u_{2}(n), \ldots, u_{M}(n)\right]$ are scaled with the square roots of the eigenvalues of the Laplacian pseudo-inverse, we obtain

$$
\mathbf{q}_{n}=\left[\frac{u_{1}(n)}{\sqrt{\lambda_{1}}}, \frac{u_{2}(n)}{\sqrt{\lambda_{2}}}, \ldots, \frac{u_{M}(n)}{\sqrt{\lambda_{M}}}\right]
$$

The elements of this spectral vector are now equal to the first $M$ elements (omitting $0 \cdot u_{0}(n)$ ) of the full-dimension spectral vector

$$
\begin{equation*}
\mathbf{q}_{n}=\left[u_{1}(n), u_{2}(n), \ldots, u_{N-1}(n)\right] \bar{\Lambda}^{-1 / 2} \tag{4.36}
\end{equation*}
$$

where $\overline{\boldsymbol{\Lambda}}$ is a diagonal matrix with elements $\lambda_{1}, \lambda_{2}, \ldots, \lambda_{N-1}$.

### 4.5.1 Commute Time Mapping

Physical meaning of the new vector positions in the spectral space, defined by (4.36), is related to the notion of commute time, which is a property of a diffusion process on a graph (Horaud, 2009; Qiu and Hancock, 2007). The commute time, $C T(m, n)$ between vertices $m$ and $n$ is defined as the expected time for the random walk to reach vertex $n$ starting from vertex $m$, and then to return. The commute time is therefore proportional to the Euclidean distance between these two vertices, with the vertex positions in the new spectral space defined by $\mathbf{q}_{n}$ in (4.36), that is

$$
\begin{equation*}
C T(m, n)=V_{\mathcal{V}}\left\|\mathbf{q}_{m}-\mathbf{q}_{n}\right\|_{2}^{2}=V_{\mathcal{V}} \sum_{i=1}^{N-1}\left(q_{i}(m)-q_{i}(n)\right)^{2} \tag{4.37}
\end{equation*}
$$

where $V_{\mathcal{V}}$ is the volume of the whole graph, $V_{\mathcal{V}}=\sum_{n=0}^{N-1} D_{n n}$.
To put this into perspective, in a graph representation of a resistive electric circuit/network, for which the edge weights are equal to the conductances (inverse resistances, see Part III), the commute time, $C T(m, n)$, is defined as the equivalent resistance between the electric circuit nodes (vertices) $m$ and $n$ (Chandra et al., 1996).

The covariance matrix of the scaled spectral vectors in (4.36) is given by

$$
\mathbf{S}=\frac{1}{N} \sum_{n=0}^{N-1} \mathbf{q}_{n}^{T} \mathbf{q}_{n}=\frac{1}{N} \overline{\boldsymbol{\Lambda}}^{-1} .
$$

In other words, the principal directions in the reduced dimensionality space of $M$ eigenvectors, $\mathbf{u}_{1}, \mathbf{u}_{2}, \ldots, \mathbf{u}_{M}$, correspond to the maximum variance of the graph embedding, since $1 / \lambda_{1}>1 / \lambda_{2}>\cdots>1 / \lambda_{M}$. This, in turn, directly corresponds to principal component analysis ( $P C A$ ).
Remark 35: Two-dimensional case comparison. The two-dimensional spectral space of the standard graph Laplacian eigenvectors is defined by $\mathbf{u}_{1}$ and $\mathbf{u}_{2}$, while the spectral vector in this space is given by

$$
\begin{equation*}
\mathbf{q}_{n}=\left[u_{1}(n), u_{2}(n)\right] . \tag{4.38}
\end{equation*}
$$

In the case of commute time mapping, the two-dimensional spectral domain of the vertices becomes

$$
\begin{equation*}
\mathbf{q}_{n}=\left[\frac{u_{1}(n)}{\sqrt{\lambda_{1}}}, \frac{u_{2}(n)}{\sqrt{\lambda_{2}}}\right], \tag{4.39}
\end{equation*}
$$

that is, the commute time mapping is related to the graph Laplacian mapping through axis scaling by $1 / \sqrt{\lambda_{k}}$.

We can conclude that when $\lambda_{1} \approx \lambda_{2}$, the two mappings in (4.38) and (4.39) are almost the same, when normalized.

However, when $\lambda_{1} \ll \lambda_{2}$, the relative eigenvalue gap between the one dimensional and two-dimensional spectral space is large, since $\delta_{r}=1-\lambda_{1} / \lambda_{2}$ is close to 1 . This means that the segmentation into two disjoint subgraphs will be "close" to the original graph, while at the same time this also indicates that the eigenvector $\mathbf{u}_{2}$ does not contribute to a new "closer" segmentation (in the sense of Section 4.3.2), since its gap $\delta_{r}=1-\lambda_{2} / \lambda_{3}$ is not small. Therefore, the influence of $\mathbf{u}_{2}$ should be reduced, as compared to the standard spectral vector of graph Laplacian where both $\mathbf{u}_{1}$ and $\mathbf{u}_{2}$ employ unit weights to give $\mathbf{q}_{n}=\left[u_{1}(n), u_{2}(n)\right]$. Such downscaling of the influence of the almost irrelevant eigenvector, $\mathbf{u}_{2}$, when $\lambda_{1} \ll \lambda_{2}$, is equivalent to the commute time mapping, since $\mathbf{q}_{n}=\left[\frac{u_{1}(n)}{\sqrt{\lambda_{1}}}, \frac{u_{2}(n)}{\sqrt{\lambda_{2}}}\right]=\frac{1}{\sqrt{\lambda_{1}}}\left[u_{1}(n), u_{2}(n) \sqrt{\frac{\lambda_{1}}{\lambda_{2}}}\right] \sim\left[u_{1}(n), 0\right]$.

For example, for the graph from Example 29, shown in Figure 4.17(a), the commute time mapping will produce the same vertex presentation as in Figure 4.17 (b), which is obtained with the eigenvectors of the graph Laplacian, when the vertical axis, $\mathbf{u}_{2}$, is scaled by

$$
\sqrt{\frac{\lambda_{1}}{\lambda_{2}}}=\sqrt{\frac{0.0286}{0.0358}}=0.8932
$$

This eigenmap will also be very close to the eigenmap in Figure 4.17(b), produced based on the graph Laplacian eigenvectors and the spectral vector $\mathbf{q}_{n}=\left[u_{1}(n), u_{2}(n)\right]$.

### 4.5.2 Diffusion (Random Walk) Mapping

Finally, we shall now relate the commute time mapping to the diffusion mapping.
Definition: Diffusion on a graph deals with the problem of propagation along the edges of a graph, whereby at the initial step, $t=0$, the random walk starts at a vertex $n$. At the next step $t=1$, the walker moves from its current vertex $n$ to one of its neighbors $l$, chosen at random from the neighbors of $n$. The probability of going from vertex $n$ to vertex $l$ is equal to the ratio of the weight $W_{n l}$ and the sum of all possible edge weights from the vertex $n$, that is

$$
\begin{equation*}
P_{n l}=\frac{W_{n l}}{\sum_{l} W_{n l}}=\frac{1}{D_{n n}} W_{n l} \tag{4.40}
\end{equation*}
$$

When considering all vertices together, such probabilities can be written in a matrix form, within the weight of a random walk matrix, defined as in (2.10), by

$$
\begin{equation*}
\mathbf{P}=\mathbf{D}^{-1} \mathbf{W} \tag{4.41}
\end{equation*}
$$

Diffusion distance. The Diffusion distance between the vertices $m$ and $n$, denoted by $D_{f}(m, n)$, is equal to the distance between the vector ( $N$-dimensional ordered set) of probabilities for a random walk to move from a vertex $m$ to all other vertices (as in (4.40)), given by

$$
\mathbf{p}_{m}=\left[P_{m 0}, P_{m 1}, \ldots, P_{m(N-1)}\right]
$$

and the corresponding vector of probabilities for a random walk to move from a vertex $n$ to all other vertices, given by

$$
\mathbf{p}_{n}=\left[P_{n 0}, P_{n 1}, \ldots, P_{n(N-1)}\right]
$$

that is

$$
\begin{aligned}
D_{f}^{2}(m, n) & =\left\|\left(\mathbf{p}_{m}-\mathbf{p}_{n}\right) \mathbf{D}^{-1 / 2}\right\|_{2}^{2} V_{\mathcal{V}} \\
& =\sum_{i=0}^{N-1}\left(P_{m i}-P_{n i}\right)^{2} \frac{1}{D_{i i}} V_{\mathcal{V}}
\end{aligned}
$$

where $V_{\mathcal{V}}=\sum_{n=0}^{N-1} D_{n n}$ is constant for a given graph, which is equal to the sum of degrees (volume) of all graph vertices in $\mathcal{V}$.
Example 33: For the graph from Figure 2.2, with its weight matrix, W, and the degree matrix, $\mathbf{D}$, given respectively in (2.4) and (2.6), the random walk weight matrix in (4.41) is of the form

$$
\mathbf{P}=\begin{align*}
& \mathbf{p}_{0}  \tag{4.42}\\
& \mathbf{p}_{1} \\
& \mathbf{p}_{2} \\
& \mathbf{p}_{3} \\
& \mathbf{p}_{4} \\
& \mathbf{p}_{5} \\
& \mathbf{p}_{6} \\
& \mathbf{p}_{7}
\end{align*}\left[\begin{array}{rrrrrrrr}
0 & 0.19 & 0.61 & 0.20 & 0 & 0 & 0 & 0 \\
0.28 & 0 & 0.43 & 0 & 0.28 & 0 & 0 & 0 \\
0.47 & 0.22 & 0 & 0.16 & 0.15 & 0 & 0 & 0 \\
0.29 & 0 & 0.32 & 0 & 0 & 0 & 0.39 & 0 \\
0 & 0.21 & 0.21 & 0 & 0 & 0.46 & 0 & 0.12 \\
0 & 0 & 0 & 0 & 0.77 & 0 & 0 & 0.23 \\
0 & 0 & 0 & 0.50 & 0 & 0 & 0 & 0.50 \\
0 & 0 & 0 & 0 & 0.23 & 0.25 & 0.52 & 0 \\
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7
\end{array}\right]
$$

with $V_{\mathcal{V}}=7.46$.
Therefore, the diffusion distance between, for example, the vertices $m=1$ and $n=3$, for the $t=1$ step, is

$$
D_{f}(1,3)=\left\|\left(\mathbf{p}_{1}-\mathbf{p}_{3}\right) \mathbf{D}^{-1 / 2}\right\|_{2} \sqrt{V_{\mathcal{V}}}=1.54
$$

while the diffusion distance between the vertices $m=6$ and $n=3$ is $D_{f}(6,3)=2.85$. From this simple example, we can see that the diffusion distance is larger for vertices $m=6$ and $n=3$ than for the neighboring vertices $m=1$ and $n=3$. This result is in a perfect accordance with the clustering scheme (expected similarity) in Figure 4.7(b), where the vertices $m=1$ and $n=3$ are grouped into the same cluster, while the vertices $m=6$ and $n=3$ belong to different clusters.

The probability vectors, $\mathbf{p}_{n}$, are called the diffusion clouds (in this case for step $t=1$ ), since they resemble a cloud around a vertex $n$. The diffusion distance can then be considered as a distance between the diffusion clouds (sets of data) around a vertex $m$ and a vertex $n$. If the vertices are well connected (approaching a complete graph structure) then this distance is small, while for vertices with long paths between them, this distance is large.

The diffusion analysis can be easily generalized to any value of the diffusion step, $t$, whereby after $t$ steps, the matrix of probabilities in (4.41) becomes

$$
\mathbf{P}^{t}=\left(\mathbf{D}^{-1} \mathbf{W}\right)^{t}
$$

The elements of this matrix, denoted by $P_{m n}^{(t)}$, are equal to the probabilities that a random walker moves from a vertex $m$ to a vertex $n$, in $t$ steps. The $t$-step diffusion distance between the vertices $m$ and $n$, is accordingly defined as

$$
D_{f}^{(t)}(m, n)=\left\|\left(\mathbf{p}_{m}^{(t)}-\mathbf{p}_{n}^{(t)}\right) \mathbf{D}^{-1 / 2}\right\|_{2} \sqrt{V_{V}}
$$

where

$$
\mathbf{p}_{m}^{(t)}=\left[P_{m 0}^{(t)}, P_{m 1}^{(t)}, \ldots, P_{m(N-1)}^{(t)}\right]
$$

and

$$
\mathbf{p}_{n}^{(t)}=\left[P_{n 0}^{(t)}, P_{n 1}^{(t)}, \ldots, P_{n(N-1)}^{(t)}\right] .
$$

It can be shown that the diffusion distance is equal to the Euclidean distance between the considered vertices when they are presented in a new space of their generalized Laplacian eigenvectors, which are then scaled by their corresponding eigenvalues; this new space is referred to as the diffusion map (cf. eigenmaps).

The eigenanalysis relation for the random walk weight matrix for the state $t=1$ now becomes

$$
\left(\mathbf{D}^{-1} \mathbf{W}\right) \mathbf{u}_{k}=\lambda_{k}^{(P)} \mathbf{u}_{k}
$$

Since the weight matrix can be written as $\mathbf{W}=\mathbf{D}-\mathbf{L}$, this yields $\mathbf{D}^{-1}(\mathbf{D}-\mathbf{L}) \mathbf{u}_{k}=\lambda_{k}^{(P)} \mathbf{u}_{k}$, or

$$
\left(\mathbf{I}-\mathbf{D}^{-1} \mathbf{L}\right) \mathbf{u}_{k}=\lambda_{k}^{(P)} \mathbf{u}_{k},
$$

to finally produce the generalized graph Laplacian equation,

$$
\mathbf{L} \mathbf{u}_{k}=\lambda_{k} \mathbf{D} \mathbf{u}_{k}
$$

with $\lambda_{k}=\left(1-\lambda_{k}^{(P)}\right)$. This relation indicates that a one-step diffusion mapping is directly obtained from the corresponding generalized graph Laplacian mapping.

After $t$ steps, the random walk matrix (of probabilities) becomes

$$
\mathbf{P}^{t}=\left(\mathbf{D}^{-1} \mathbf{W}\right)^{t}
$$

for which the eigenvalues are $\lambda_{k}^{(P) t}=\left(1-\lambda_{k}\right)^{t}$, while the (right) eigenvectors remain the same as for the graph Laplacian, see (3.7).

The spectral space for vertices, for a $t$-step diffusion process (diffusion mapping), is then defined based on the spectral vector

$$
\mathbf{q}_{n}=\left[u_{1}(n), u_{2}(n), \ldots, u_{N-1}(n)\right](\mathbf{I}-\overline{\boldsymbol{\Lambda}})^{t}
$$

and is equal to the generalized Laplacian spectral space mapping, whereby the axis vectors $\mathbf{q}_{n}=\left[u_{1}(n), u_{2}(n), \ldots, u_{N-1}(n)\right]$ are multiplied by the corresponding eigenvalues, $\left(1-\lambda_{k}\right)^{t}$.

It can be shown that the diffusion distance between vertices in the new diffusion map space is equal to their Euclidean distance (Coifman and Lafon, 2006), that is

$$
\begin{equation*}
D_{f}^{(t)}(m, n)=\sqrt{V_{\mathcal{V}}}\left\|\mathbf{q}_{m}-\mathbf{q}_{n}\right\|_{2} \tag{4.43}
\end{equation*}
$$

Example 34: For the graph from Figure 2.2, whose weight matrix, W, and the degree matrix, $\mathbf{D}$, are defined in (2.4) and (2.6), the diffusion distance between the vertices $m=1$ and $n=3$ can be calculated using (4.43) as

$$
D_{f}^{(1)}(1,3)=\sqrt{V_{\mathcal{V}}}\left\|\left(\mathbf{q}_{1}-\mathbf{q}_{3}\right)\right\|_{2}=1.54
$$

where the spectral vectors, $\mathbf{q}_{1}=\left[u_{1}(1)\left(1-\lambda_{1}\right)^{1}, \ldots, u_{N}(1)\left(1-\lambda_{N}\right)^{1}\right]$ and $\mathbf{q}_{3}=\left[u_{1}(3)\left(1-\lambda_{1}\right)^{1}, \ldots, u_{N}(3)\left(1-\lambda_{N}\right)^{1}\right]$ are obtained using the generalized graph Laplacian eigenvectors, $\mathbf{u}_{k}$, and the corresponding eigenvalues, $\lambda_{k}$, from $\mathbf{L} \mathbf{u}_{k}=\lambda_{k} \mathbf{D} \mathbf{u}_{k}$. This is the same diffusion distance value, $D_{f}(1,3)$, as in Example 33.
Dimensionality reduced diffusion maps. Dimensionality of the vertex representation space can be reduced in diffusion maps by keeping
only the eigenvectors that correspond to the $M$ most significant eigenvalues, $\left(1-\lambda_{k}\right)^{t}, k=1,2, \ldots, M$, in the same way as for the Laplacian eigenmaps, For example, the two-dimensional spectral domain of the vertices in the diffusion mapping is defined as

$$
\mathbf{q}_{n}=\left[u_{1}(n)\left(1-\lambda_{1}\right)^{t}, u_{2}(n)\left(1-\lambda_{2}\right)^{t}\right] .
$$

While the analysis and intuition for the diffusion mapping is similar to that for the commute time mapping, presented in Remark 35, diffusion maps have an additional degree of freedom, the step $t$.

Example 35: For the graph in Figure 4.10, which corresponds to a set of real-world images, the commute time two-dimensional spectral vectors in (4.39), normalized by the first eigenvector value through a multiplication of its coordinates by $\sqrt{\lambda_{1}}$, assume the form

$$
\mathbf{q}_{n}=\left[u_{1}(n), \frac{\sqrt{\lambda_{1}}}{\sqrt{\lambda_{2}}} u_{2}(n)\right]=\left[u_{1}(n), 0.62 u_{2}(n)\right]
$$

The corresponding vertex colors designate diffusion-based clustering, as shown in Figure $4.25(\mathrm{a})$. Figure $4.25(\mathrm{~b})$ shows the vertices of this graph, colored with the two-dimensional diffusion map spectral vectors, which are normalized by $\left(1-\lambda_{1}\right)$, to yield

$$
\mathbf{q}_{n}=\left[u_{1}(n), \frac{1-\lambda_{2}}{1-\lambda_{1}} u_{2}(n)\right]=\left[u_{1}(n), 0.09 u_{2}(n)\right]
$$

Finally, the sum over all steps, $t=0,1,2, \ldots$, of the diffusion space yields

$$
\mathbf{q}_{n}=\left[u_{1}(n), u_{2}(n), \ldots, u_{N-1}(n)\right] \bar{\Lambda}^{-1}
$$

since the sum of a geometric progression is equal to

$$
\sum_{t=0}^{\infty}(\mathbf{I}-\bar{\Lambda})^{t}=\bar{\Lambda}^{-1}
$$

This mapping also corresponds to the cumulative diffusion distance, given by

$$
D_{c}(n, l)=\sum_{t=0}^{\infty} D_{f}^{(t)}(n, l)
$$



Figure 4.25: Graph structure for the images from Figure 4.10, with vertex color embedding which corresponds to the two-dimensional normalized spectral vectors in (a) the commute time representation, $\mathbf{q}_{n}=\left[u_{1}(n), 0.62 u_{2}(n)\right]$, and (b) the spectral eigenvectors of the diffusion process, $\mathbf{q}_{n}=\left[u_{1}(n), 0.09 u_{2}(n)\right]$, with $t=1$. For the commute time presentation in (a), the graph Laplacian eigenvectors, $\mathbf{u}_{1}$ and $\mathbf{u}_{2}$, are used, while for the diffusion process presentation in (b) the generalized Laplacian eigenvectors, $\mathbf{u}_{1}$ and $\mathbf{u}_{2}$, are used.

The diffusion eigenmaps can be therefore obtained by appropriate axis scaling of the standard eigenmaps, produced by the generalized eigenvectors of the graph Laplacian.
Remark 36: The commute time and the diffusion process mappings are related in the same way as the mappings based on the graph Laplacian eigenvectors and the generalized eigenvectors of the graph Laplacian.

### 4.6 Summary of Embedding Mappings

A summary of the considered embedding mappings is given in Table 4.1. Notice that various normalization schemes may be used to obtain the axis vectors, $\mathbf{y}_{n}$, from the spectral vectors, $\mathbf{q}_{n}$ (see Algorithm 9).

These examples of dimensionality reduction reveal close connections with spectral clustering algorithms developed in standard machine learning and computer vision; in this sense, the notions of dimensionality reduction and clustering can be considered as two sides of the same coin (Belkin and Niyogi, 2003). In addition to the reduction of dimensionality for visualization purposes, the resulting spectral vertex space of lower dimensionality may be used to mitigate the complexity and accuracy issues experienced with classification algorithms, or in other words to bypass the course of dimensionality.

A recent approach to graph dimensionality reduction, called the Uniform Manifold Approximation and Projection (UMAP), can be found in McInnes et al. (2018). This dimension reduction technique may be used for visualization similarly to t-distributed Stochastic Neighbor Embedding (t-SNE), which employs a probabilistic approach whereby, with high probability, similar objects are modeled by nearby points and dissimilar objects by distant points, as in van der Maaten and Hinton (2008).

## 5

## Graph Sampling Strategies

In the case of extremely large graphs, subsampling and down-scaling of graphs is a prerequisite for their analysis (Leskovec and Faloutsos, 2006). For a given large (in general directed) graph, $\mathcal{G}$, with $N$ vertices, its resampling aims to produce a much simpler graph which retains most of the properties of the original graph, but is both less complex and more physically and computationally meaningful. The similarity between the original large graph $\mathcal{G}$, and the down-scaled graph, $\mathcal{S}$, with $M$ vertices, where $M \ll N$, is defined with respect to the set of parameters of interest, like for example, the connectivity or distribution on a graph. Such criteria may also be related to the spectral behavior of graphs.

### 5.1 Graph Down-Sampling Strategies

Several methods exist for graph down-scaling, of which some are listed below.

- The simplest method for graph down-sampling is the random vertex or random node ( $R N$ ) selection method, whereby a random subset of vertices is used for the analysis and representation of large graphs and data observed on such large graphs. Even though
the vertices are here selected with equal probabilities, this method produces good results in practical applications.
- Different from the RN method, where the vertices are selected with a uniform probability, the random degree vertex/node ( $R D N$ ) selection method is based on the probability of vertex selection that is proportional to the vertex degree. In other words, vertices with more connections, thus having larger $D_{n}=\sum_{m} W_{n m}$, are selected with higher probability. This makes the RDN approach biased with respect to highly connected vertices.
- The PageRank method is similar to the RDN, and is based on the vertex rank. The PageRank is defined by the importance of the vertices connected to the considered vertex $n$. Then, the probability that a vertex $n$ will be used in a down-scaled graph is proportional to the PageRank of this vertex. This method is also known as the random PageRank vertex ( $R P N$ ) selection, and is biased with respect to the highly connected vertices (with a high PageRank).
- A method based on a random selection of edges that will remain in the simplified graph is called the random edge ( $R E$ ) method. This method may lead to graphs that are not well connected, and which exhibit large diameters.
- The RE method may be combined with random vertex selection to yield a combined RNE method, whereby the initial random vertex selection is followed by a random selection of one of the edges that is connected to the selected vertex.
- In addition to these methods, more sophisticated methods based on random vertex selection and random walk ( $R W$ ) analysis may be defined. For example, we can randomly select a small subset of vertices and form several random walks starting from each selected vertex. The Random Walk ( $R W$ ), Random Jump ( $R \mathrm{~J}$ ) and Forest Fire graph down-scaling strategies are all defined in this way.


### 5.2 Graph Sparsification

We now provide an in-depth discussion of graph sparsification, one of the main graph sampling strategies that approximates a given graph by a sparse graph (a graph for which the number of the edges is significantly smaller than quadratic in the number of vertices). Appropriately sparsified graphs allow for a simpler analysis of large graphs, while producing similar results as if the original graphs were analyzed.

Definition: A subgraph or sparsifier, $\mathcal{G}^{\prime}$, of a graph, $\mathcal{G}$, is a graph which maintains the same set of vertices, $\mathcal{V}$, but with a fewer edges. The design of a sparsification strategy should ensure that a desired property/operation of the original graph is approximately preserved.

### 5.2.1 Cut-Preserving Sparsification

This approach to the sparsification of graphs aims at preserving (approximately) graph cuts. Consider an unweighted graph $\mathcal{G}$ with $N$ vertices. A new, cut-preserving sparsified graph $\mathcal{G}^{\prime}$ is then obtained by randomly pruning the edges of the original graph $\mathcal{G}$ with the aim of preserving the cut values. The set of vertices is the same for both the original and the resulting graphs. Assume next that the vertices, $\mathcal{V}$, are grouped into disjoint subsets, $\mathcal{E}$ and $\mathcal{H}$, with $\mathcal{E} \cup \mathcal{H}=\mathcal{V}$. The aim is to ensure that every cut of the sparsified graph, $\mathcal{G}^{\prime}$, with the same set of vertices, $\mathcal{V}$, and the new edges with weighs $W_{m n}^{\prime}$, denoted by

$$
C u t_{\mathcal{G}^{\prime}}(\mathcal{E}, \mathcal{H})=\sum_{\substack{m \in \mathcal{E} \\ n \in \mathcal{H}}} W_{m n}^{\prime}
$$

is close to the corresponding cut of the original graph, that is

$$
\begin{equation*}
(1-\epsilon) C u t_{\mathcal{G}}(\mathcal{E}, \mathcal{H}) \leq C u t_{\mathcal{G}^{\prime}}(\mathcal{E}, \mathcal{H}) \leq(1+\epsilon) C u t_{\mathcal{G}}(\mathcal{E}, \mathcal{H}) \tag{5.1}
\end{equation*}
$$

where $\epsilon$ is sufficiently small.
To this end, random edge selection is achieved in the following way:

- every edge is kept in the new graph, $\mathcal{G}^{\prime}$, with an assumed probability $p$;
- the weight of the edge which is kept in the new graph, $\mathcal{G}^{\prime}$, is changed from 1 to $1 / p$.

In this way, the number of edges, $N_{e}$, in the original graph, $\mathcal{G}$, is reduced to the expected number of edges equal to $p N_{e}$.

The inequality in (5.1) is satisfied with a certain probability, for a given $\epsilon$. Consider an undirected and unweighted graph, with $M$ edges in one cut. Every edge in this cut is either removed (with probability $(1-p))$ or kept with probability $p$. If the edge is kept, its weight assumes the value $1 / p$. For the $M$ edges in a considered cut, the probability that $k$ of $M$ edges will be kept is equal to

$$
P_{k}=\binom{M}{k} p^{k}(1-p)^{M-k}
$$

The resulting value of the new cut is a random variable with Bernoulli distribution, given by

$$
P\left(\operatorname{Cut}_{\mathcal{G}^{\prime}}(\mathcal{E}, \mathcal{H})=k \frac{1}{p}\right)=\frac{k}{p}\binom{M}{k} p^{k}(1-p)^{M-k} .
$$

The mean value of this cut is

$$
E\left\{C u t_{\mathcal{G}^{\prime}}(\mathcal{E}, \mathcal{H})\right\}=\frac{1}{p}(p M)=M,
$$

while the variance of this Bernoulli distributed random variable is

$$
\operatorname{Var}\left\{\operatorname{Cut}_{\mathcal{G}^{\prime}}(\mathcal{E}, \mathcal{H})\right\}=\frac{1}{p^{2}} p(1-p) M=\frac{1-p}{p} M
$$

Having in mind that, for a large $M$, the Bernoulli distribution approaches the Gaussian distribution we can conclude that the relation is satisfied with a probability of 0.95 for $\epsilon=2 \sqrt{(1-p) /(p M)}$, according to the two-sigma rule for the Gaussian distribution and after the normalization with $\operatorname{Cut}_{\mathcal{G}}(\mathcal{E}, \mathcal{H})=M$.

For cuts with a very small number of edges, if all edges are sampled with the same probability, $p$, there is a significant probability that the minimum cut would be destroyed by removing all edges in this cut. A way to overcome this problem is to slightly adapt the selection procedure, so that the minimum cut is always kept, and the other edges
are sparsified in the usual way. This simple scheme therefore adopts the probability of removing the edges related to the number of the edges in a cut. For example, in cuts with a small number of edges, the probability of removing the edges should be very small.

### 5.2.2 Spectral Graph Sparsification

Recent research efforts on spectral graph sparsification focus on definition of subgraphs or sparsifiers that can robustly preserve the spectrum (eigenvalues and eigenvectors) of the original graph Laplacian (Imre et al., 2020).

The criterion for spectral similarity of two graphs is based on the quadratic Laplacian form

$$
\begin{equation*}
\mathbf{x}^{T} \mathbf{L} \mathbf{x}=\frac{1}{2} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} W_{m n}(x(m)-x(n))^{2}, \tag{5.2}
\end{equation*}
$$

where $\mathbf{x}$ is an arbitrary vector with $N$ elements.
Definition: The graphs, $\mathcal{G}$ and $\mathcal{G}^{\prime}$, with respective graph Laplacians, $\mathbf{L}$ and $\mathbf{L}^{\prime}$, are $\sigma$-spectrally similar if their quadratic forms satisfy

$$
\begin{equation*}
\frac{1}{\sigma} \mathbf{x}^{T} \mathbf{L}^{\prime} \mathbf{x} \leq \mathbf{x}^{T} \mathbf{L} \mathbf{x} \leq \sigma \mathbf{x}^{T} \mathbf{L}^{\prime} \mathbf{x} \tag{5.3}
\end{equation*}
$$

The quality of the sparsification can be evaluated through the condition number, $\lambda_{\max } / \lambda_{\min }$, of the generalized eigenvalue relation

$$
\begin{equation*}
\mathbf{L} \mathbf{u}=\lambda \mathbf{L}^{\prime} \mathbf{u} \tag{5.4}
\end{equation*}
$$

with the constant, $\sigma$, satisfying the relation, $\sigma^{2} \geq \lambda_{\max } / \lambda_{\min }$, where $\lambda_{\max }$ and $\lambda_{\text {min }}$ are respectively the maximum and minimum generalized eigenvalue of (5.4). A smaller $\sigma\left(\sigma \approx 1\right.$ or $\left.\lambda_{\max } \approx \lambda_{\min }\right)$ indicates higher spectral similarity.

The state-of-art techniques in this area employ an analogy with effective resistances in circuit theory (Spielman and Srivastava, 2011). The underpinning idea is as follows; a graph $\mathcal{G}$ with $N$ vertices can be considered as a resistive network with resistances $R_{m n}=1 / W_{m n}$ between the vertices $m$ and $n$, which are connected by an edge (more detail on the equivalence between a general graph and the resistive
network is given in Part III). For any two vertices, $m$ and $n$, that are connected by an edge, the effective resistance can be calculated in several ways: (1) Using transformations of the corresponding electrical circuit (including the so-called star-mesh transformations); (2) Injecting unit current into the vertex, $m$, and taking the same current out from the vertex, $n$. The effective resistance is then equal to the difference of potentials in the vertices $m$ and $n$; and (3) Through the eigenvalue (spectral) decomposition of the corresponding graph. The effective resistance, $R_{\text {eff }}(m, n)$, is then obtained from (4.37) as

$$
\begin{equation*}
C T(m, n)=V_{\mathcal{V}}\left\|\mathbf{q}_{m}-\mathbf{q}_{n}\right\|_{2}^{2}=V_{\mathcal{V}} \sum_{i=1}^{N-1}\left(q_{i}(m)-q_{i}(n)\right)^{2}=V_{\mathcal{V}} R_{e f f}(m, n) \tag{5.5}
\end{equation*}
$$

where $R_{\text {eff }}(m, n)$ denotes the effective resistance between vertices $m$ and $n$ (this relation will be proven in Part III), and is given by

$$
\begin{equation*}
R_{e f f}(m, n)=\sum_{i=1}^{N-1}\left(q_{i}(m)-q_{i}(n)\right)^{2}=\left\|\mathbf{q}_{m}-\mathbf{q}_{n}\right\|_{2}^{2} \tag{5.6}
\end{equation*}
$$

Spectral graph sparsification can now be thought of as a process of sampling edges from the graph, with probabilities of keeping edges proportional to their effective resistances. This approach rests upon the observation that if the effective resistance is small with respect to the resistance of the edge directly connecting the vertices, $m$ and $n$, then these two vertices are well connected via other edges and the considered direct edge can be removed without significant influence on the whole graph. In turn, upon this edge is removed, from (5.2) we see that the total dissipated energy in the circuit corresponding to graph $\mathcal{G}$ will not change significantly, and will remain close to the energy in the electric network corresponding to the new pruned graph.

Note that if the effective resistance is close to $R_{e f f}(m, n) \approx 1 / W_{m n}$, then the other network connections are weak and the considered edge should be kept.

Such a simplified pruning algorithm can be implemented as follows.

- For the considered graph, find the graph Laplacian, $\mathbf{L}$.
- Calculate the eigenvectors, $\mathbf{u}_{k}$, and the eigenvalues, $\lambda_{k}$, of the graph Laplacian, $k=0,1,2, \ldots, N-1$.
- Form the commute time spectral vectors with elements, $q_{k}(n)=$ $u_{k}(n) / \sqrt{\lambda_{k}}$, for $k=1,2, \ldots, N-1$.
- Find the effective resistances, $R_{\text {eff }}(m, n)=\sum_{k=1}^{N-1}\left(q_{k}(m)-q_{k}(n)\right)^{2}$.
- For every pair of vertices, $m$ and $n$, connected by an edge, use $R_{\text {eff }}(m, n)$ as a measure for the probability that the considered edge should be kept in the graph.

The effective conductance (inverse to the effective resistance) between the vertices, $m$ and $n$, is equal to

$$
\frac{1}{R_{e f f}(m, n)}=W_{m n}+C_{m n}
$$

where $C_{m n}=\frac{1}{R_{\text {eff }}(m, n)}-W_{m n}$ is the effective conductance between $m$ and $n$ due to all other connections, except for the direct one defined by $W_{m n}$. Its relative value, normalized by $W_{m n}$, is given by

$$
\frac{C_{m n}}{W_{m n}}=\frac{1}{W_{m n} R_{e f f}(m, n)}-1
$$

We can now state that the influence of indirect connections between the vertices, $m$ and $n$, is significant with respect to the existing direct connection, if

$$
\frac{1}{W_{m n} R_{e f f}(m, n)}-1 \gg 1 .
$$

The probability of keeping the edge $(m, n)$ becomes

$$
P_{m n}=\frac{W_{m n}}{\frac{1}{R_{e f f}(m, n)}}=W_{m n} R_{e f f}(m, n)
$$

If there are no indirect connections between $m$ and $n$, then $1 / R_{e f f}=W_{m n}$ and the edge $(m, n)$ must be kept with probability $P_{m n}=1$. In general, $1 / R_{e f f} \geq W_{m n}$ holds. By increasing the number of indirect connections, $1 / R_{\text {eff }}$ becomes increasingly larger than $W_{m n}\left(c f . R_{\text {eff }}\right.$ increasingly smaller than $\left.1 / W_{m n}\right)$, thus indicating that the probability of keeping this edge should be decreasing.


Figure 5.1: Principle of spectral graph sparsification. (a) The graph from Figure 2.2 with the edge weights, $W_{m n}$, and the effective resistances, $R_{\text {eff }}(m, n)$, for each pair of connected vertices. (b) The pruned graph from (a) whereby the edge (4,7), characterized by the minimum value of $W_{m n} R_{\text {eff }}(m, n)$, is removed.

Example 36: Consider the graph from Figure 2.2. The effective resistances are calculated using $R_{\text {eff }}(m, n)=\sum_{k=1}^{N-1}\left(q_{k}(m)-q_{k}(n)\right)^{2}$, with the spectral vectors calculated using the graph Laplacian eigenvectors as $q_{k}(n)=u_{k}(n) / \sqrt{\lambda_{k}}$, for $k=1,2, \ldots, N-1$. The values of effective resistances are given in red in Figure 5.1(a). When these resistances are multiplied by the corresponding edge weights, the lowest product is obtained for $W_{47} R_{\text {eff }}(4,7)=2.87 \cdot 0.14=0.40$. Therefore, this is the candidate for an edge with the lowest probability of being kept, and the best candidate for pruning. The worst candidate for pruning would be $W_{45} R_{\text {eff }}(4,5)=0.85$. After pruning the edge $(4,7)$, the pruned graph, $\mathcal{G}_{P}$, is shown in Figure 5.1(b). Another common criterion for pruning suggests that the edge with the smallest effective resistance should be pruned; this criterion would suggest to prune the edge $(0,2)$.

In that case, the spectral distance between the original and pruned graphs becomes (Jovanović and Stanić, 2012)

$$
S D\left(\mathcal{G}, \mathcal{G}_{P}\right)=\sum_{k=0}^{N-1}\left|\lambda_{k}-\lambda_{k}^{P}\right|=0.28
$$

Note that if the "worst" edge $(4,5)$, with the maximum value of $W_{m n} R_{\text {eff }}(m, n)$ is pruned, then the spectral distance becomes $S D\left(\mathcal{G}, \mathcal{G}_{P}\right)=1.02$.

A main obstacle for using spectral sparsification is that for large graphs it is computationally very demanding, as the estimation of edge
effective resistances requires computing the eigenvectors and eigenvalues of the graph Laplacian. This topic is currently under intensive investigation.

### 5.2.3 Uniform Graph Sparsifier

This sparsification strategy randomly selects $M$ edges, with replacement, with probabilities proportional to their weights (Sadhanala et al., 2016). The sparsified graph is then formed using the same vertices, but with the selected edges having equal weights, that is

$$
W_{m n}^{\prime}=\frac{1}{2 M} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} W_{m n}=\frac{W}{2 M} .
$$

The so produced random graph maintains its expected energy equal to the energy in the original graph, and for any $x(n)$, that is

$$
\begin{equation*}
\mathrm{E}\left\{\mathbf{x}^{T} \mathbf{L}^{\prime} \mathbf{x}\right\}=\mathbf{x}^{T} \mathbf{L} \mathbf{x} . \tag{5.7}
\end{equation*}
$$

To prove this, note that the number of times, $N_{W_{m n}}$, that an edge between vertices $m$ and $n$, with the corresponding weight $W_{m n}$, is selected, is equal to $E\left\{N_{W_{m n}}\right\}=W_{m n} 2 M / W$. Then, the expected value of the weight $W_{m n}^{\prime}$ is $E\left\{W_{m n}^{\prime}\right\}=E\left\{N_{W_{m n}} \frac{W}{2 M}\right\}$, which gives $E\left\{W_{m n}^{\prime}\right\}=W_{m n}$, and $E\left\{\mathbf{L}^{\prime}\right\}=\mathbf{L}$.

### 5.3 Graph Coarsening

We have so far addressed graph sparsification based on a reduction in the number of edges, while the number of vertices remained unaltered. Note that the number of vertices defines the size and dimensionality of the graph, with the analysis quickly becoming computationally prohibitive for large graphs. Graph coarsening belongs to graph down-sampling strategies and refers to the reduction in the number of vertices of the original graph. Graph coarsening is typically used in graph partitioning and for the visualization of large graphs in a computationally efficient manner (Tremblay and Loukas, 2020). In general, it can be performed by grouping the vertices into $N_{c}<N$ groups, subsequently forming new vertices, and finally connecting these new vertices (former groups


Figure 5.2: Graph coarsening. (a) The original graph from Figure 2.2 with the edges $(0,2)$ and $(4,5)$ used for vertex merging and forming "super-vertices" designated by circles. (b) The coarsened graph with a reduced number of vertices, obtained by forming two "super-vertices" 02 and 45 . The resulting edge weights are obtained by summing up all corresponding edge weights belonging to the "super-vertices".
of vertices) with the "equivalent weights", which represent a sum of all weights between the groups. Groups of vertices are formed using the matching in graphs (explained below).
Example 37: Consider the graph $\mathcal{G}$ from Figure 2.2. To form a coarsened version, $\mathcal{G}_{c}$, of this graph, which has a reduced number of vertices, we shall first form two "super-vertices". For example, the "super-vertices" 02 and 45 can be formed respectively from the vertices 0 and 2 and vertices 4 and 5, as in Figure 5.2(a). The "super-edges" connecting these super-vertices are obtained as cumulative values for the vertex edges forming the new "super-vertices". The weight matrix of this coarsened graph is of dimension $N_{c}=6$, and is given by

$$
\mathbf{W}_{c}=\begin{gather*}
02  \tag{5.8}\\
1 \\
3 \\
45 \\
6 \\
7
\end{gather*}\left[\begin{array}{rrrrrr}
1.48 & 0.58 & 0.50 & 0.24 & 0 & 0 \\
0.58 & 0 & 0 & 0.23 & 0 & 0 \\
0.50 & 0 & 0 & 0 & 0.32 & 0 \\
0.24 & 0.23 & 0 & 1.02 & 0 & 0.29 \\
0 & 0 & 0.32 & 0 & 0 & 0.32 \\
0 & 0 & 0 & 0.29 & 0.32 & 0
\end{array}\right],
$$

with the "super-vertices" exhibiting self-loops with the weights equal to double the value of the removed edge (edge within the "super-vertex"). In some applications, the self-loops are filtered-out (removed).

The new, reduced-dimension weight matrix, $\mathbf{W}_{c}$, of the coarsened graph could be alternatively obtained using the "super-vertex" indicator matrix $\mathbf{P}$, whose elements are 1 if the vertex in the original graph $\mathcal{G}$ belongs to the considered "super-vertex" and zero elsewhere, that is

$$
\mathbf{P}=\begin{gather*}
02  \tag{5.9}\\
1 \\
3 \\
45 \\
6 \\
7
\end{gather*}\left[\begin{array}{cccccccc}
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{array}\right] .
$$

The relation between the weight matrix of the coarse graph, $\mathbf{W}_{c}$, and that of the original graph, $\mathbf{W}$, is therefore

$$
\mathbf{W}_{c}=\mathbf{P} \mathbf{W} \mathbf{P}^{T},
$$

where $\mathbf{W}$ is defined in (2.4) and the resulting coarsened graphs is shown in Figure 5.2(b).

Graph lifting (uncoarsening). Graph lifting is an inverse operation to graph coarsening, and represents a process of obtaining a larger scale (fine) graph from a coarsened (smaller) graph. The weight matrix, $\mathbf{W}_{L}$, of the lifted graph is obtained from the weight matrix of the coarsened graph, $\mathbf{W}_{c}$, as

$$
\mathbf{W}_{L}=\mathbf{P}^{+} \mathbf{W}_{c}\left(\mathbf{P}^{+}\right)^{T},
$$

where $\mathbf{P}^{+}$is the pseudo-inverse of the indicator matrix, such that $\mathbf{P} \mathbf{P}^{+}=\mathbf{I}$, where $\mathbf{I}$ is the identity matrix.

For the considered example, the lifted weight matrix is

$$
\mathbf{W}_{L}=\begin{array}{r}
0  \tag{5.10}\\
1 \\
2 \\
3 \\
4 \\
5 \\
6 \\
7
\end{array}\left[\begin{array}{rrrrrrrr}
0.37 & 0.29 & 0.37 & 0.25 & 0.06 & 0.06 & 0 & 0 \\
0.29 & 0 & 0.29 & 0 & 0.11 & 0.11 & 0 & 0 \\
0.37 & 0.29 & 0.37 & 0.25 & 0.06 & 0.06 & 0 & 0 \\
0.25 & 0 & 0.25 & 0 & 0 & 0 & 0.32 & 0 \\
0.06 & 0.11 & 0.06 & 0 & 0.25 & 0.25 & 0 & 0.14 \\
0.06 & 0.11 & 0.06 & 0 & 0.25 & 0.25 & 0 & 0.14 \\
0 & 0 & 0 & 0.32 & 0 & 0 & 0 & 0.32 \\
0 & 0 & 0 & 0 & 0.14 & 0.14 & 0.32 & 0
\end{array}\right] .
$$

The same relations as for the weights hold for the corresponding graph Laplacian of the original graph, $\mathbf{L}$, graph Laplacian of the coarsened graph, $\mathbf{L}_{c}$, and the graph Laplacian of the lifted graph, $\mathbf{L}_{L}$, that is

$$
\begin{aligned}
\mathbf{L}_{c} & =\mathbf{P L P}^{T} \\
\mathbf{L}_{L} & =\mathbf{P}^{+} \mathbf{L}_{c}\left(\mathbf{P}^{+}\right)^{T} .
\end{aligned}
$$

Notice that for the normalized graph Laplacian, the definition of the indicator matrix should be slightly modified (Jin et al., 2020).

It is of particular interest to consider spectral similarity of the original large size graph and the corresponding reduced-size graph (or a lifted graph). If spectral similarity is preserved by graph coarsening, then instead of operating on the large graph, $\mathcal{G}$, the eigendecomposition is first performed at a lower computational cost on the smaller dimensional coarsened graph $\left(N_{c} \ll N\right)$. Then, spectral analysis is performed by lifting the graph to the original large dimensionality and refining the results (Loukas and Vandergheynst, 2018; Tremblay and Loukas, 2020).

Various criteria for spectral similarity exist, including an elementwise form of (5.3). We here employ as a spectral similarity metric a simple spectral distance between the original graph, $\mathcal{G}$, and the coarsened and lifted graph of the same dimension, $\mathcal{G}_{L}$, defined by

$$
S D\left(\mathcal{G}, \mathcal{G}_{L}\right)=\sum_{k=0}^{N-1}\left|\lambda_{k}-\lambda_{k}^{c L}\right|,
$$

where $\lambda_{k}$ are the eigenvalues of the original graph, and $\lambda_{k}^{c L}$ the eigenvalues of the coarsened and lifted graph. For the considered example, the spectral distance is $S D\left(\mathcal{G}, \mathcal{G}_{L}\right)=1.48$.

Generalization. The process of graph coarsening may be continued until a desired number of vertices is obtained. For example, the vertex 1 and the "super-vertex" 02 can be grouped into a new "super-vertex" 102. Then, the new edges are calculated using the indicator matrix and the matrix $\mathbf{W}_{c}$.

In general, the coarsening involves a sequence of graphs

$$
\begin{gathered}
\mathcal{G}=\mathcal{G}_{0}=\{\mathcal{V}, \mathcal{B}, \mathcal{W}\}=\left\{\mathcal{V}_{0}, \mathcal{B}_{0}, \mathcal{W}_{0}\right\} \\
\mathcal{G}_{1}=\left\{\mathcal{V}_{1}, \mathcal{B}_{1}, \mathcal{W}_{1}\right\} \\
\vdots \\
\mathcal{G}_{c}=\left\{\mathcal{V}_{c}, \mathcal{B}_{c}, \mathcal{W}_{c}\right\},
\end{gathered}
$$

whereby at every iteration, the coarsened graph, $\mathcal{G}_{l+1}=\left\{\mathcal{V}_{l+1}, \mathcal{B}_{l+1}\right.$, $\left.\mathcal{W}_{l+1}\right\}$, is obtained from the previous one through a weight matrix transformation based on the corresponding indicator matrices,

$$
\mathbf{W}_{l}=\mathbf{P}_{l} \mathbf{W}_{l-1} \mathbf{P}_{l}^{T}
$$

while the lifting is performed as $\mathbf{W}_{l-1}=\mathbf{P}_{l}^{+} \mathbf{W}_{l}\left(\mathbf{P}_{l}^{+}\right)^{T}$.
Matching. In forming the "super-vertices" for graph coarsening, the notion of matching is commonly used.

Definition: A matching in a graph is a set of edges such that no vertex belongs to more than one edge.

For example, the edges $(0,2)$ and $(4,5)$ form a matching $\{02,45\}$ in the graph from Example 37 given in Figure 5.2, while the edges $(0,2)$ and $(2,3)$ are not a matching, since they are both connected to vertex 2 .

Definition: A matching is maximal if no more edges can be added to this matching.

For example, for the graph from Figure 5.2, the maximal matching would be the set of edges $\{02,45,36\}$, as no more edges can be added to this matching. However, this is not the largest possible number of
edges in a matching for this graph, and we can define a matching with a larger number of edges, like for example, the matching $\{02,14,57,36\}$.

Definition: The maximum matching in a graph is a set of edges such that no vertex belongs to more than one edge, and another matching with a larger number of edges does not exist.
Example 38: Consider the graph, $\mathcal{G}$, from Figure 2.2. In forming a coarsened version, $\mathcal{G}_{1}$, of this graph we will create "super-vertices" using the maximal matching $\{02,45,36\}$, shown in Figure 5.3(a). The coarsened version of this graph, using the maximal matching, is given in Figure 5.3(b); this graph is coarsened again, by forming "super-vertices" 102 and 367 , as in Figure 5.3(c); the final form is obtained with only two "super-vertices", as shown in Figure 5.3(d).

Notice that the edge weight in the final two-vertex graph is equal to the original graph cut for $\mathcal{E}=\{9,1,2,3,6,7\}$ and $\mathcal{H}=\{4,5\}$, that is, $\operatorname{Cut}(\mathcal{E}, \mathcal{H})=0.23+0.24+0.14+0.15=0.76$.

This example can be repeated by using the maximum matching $\{02,14,57,36\}$ in the first step.

Maximal matching strategies for graph coarsening include:

- Random matching, when a vertex $n$ and one of its edges $(m, n)$ are selected randomly. Next, another neighboring (or any other) vertex is randomly selected, together with one of its edges. The process is continued until no new edge can be added to this matching. The "super-vertices" are formed for each of the selected edges. After one coarsening level, the process can be repeated, until the desired number of vertices in a coarsened graph is reached, or a given number of levels is used.
- Heavy edge matching (HEM) algorithm is similar to the previous one, with the only difference in that once a vertex is randomly selected, then its edge with the maximum weight is used for the matching and "super-vertex" forming. In this way, the edges with the strongest weights are excluded, since they would probably not participate in the minimum cut, so that both the original and the coarsened graph share the same minimum cut.

(a)


$$
\mathcal{G}_{2}=\left\{\mathcal{V}_{2}, \mathcal{B}_{2}, \mathcal{W}_{2}\right\}
$$

(c)
$\mathcal{G}_{1}=\left\{\mathcal{V}_{1}, \mathcal{B}_{1}, \mathcal{W}_{1}\right\}$

(b)

$\mathcal{G}_{3}=\left\{\mathcal{V}_{3}, \mathcal{B}_{3}, \mathcal{W}_{3}\right\}$
(d)

Figure 5.3: Principle of maximal matching for graph coarsening. (a) The original graph from Figure 2.2 with (b)-(d) its coarsened graphs obtained in three steps using the maximal matching, until a two-vertex graph is obtained.

- Sorted Heavy edge matching uses the vertices with the highest degree first, in defining the matching. Vertices with higher degrees are also preferred in the subsequent steps.
- Edge weighted random matching chooses an edge with a probability, $P_{m n}$, proportional to its weight, that is

$$
P_{m n}=\frac{W_{m n}}{\frac{1}{2} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} W_{m n}}
$$

The edges with a higher weight are thus more likely to be selected in each step of the maximal matching procedure.

### 5.4 Kron Reduction of Graphs

A reduction of an electrical network via a Schur complement of the associated conductance matrix is known as the Kron reduction, due to the seminal work of Gabriel Kron. It is based on separating the vertices into two groups: active vertices and inner vertices. The inner vertices can be eliminated from the graph without changing the electric network conditions; this is achieved via equivalent transformations, such as the "star-mesh" transformations (Dorfler and Bullo, 2012). The Kron reduction of graphs is also relevant in other physical domains, including computing applications and the reduction of Markov chains. Since this approach requires quite specific physical interpretation of the active and inner vertices, it will be discussed in detail in Part III of this monograph.

## 6

## Conclusion

Although within the graph data analytics paradigm, graphs have been present in various forms for centuries, the advantages of the graph framework for data analytics, as opposed to the optimization of the graphs themselves, but for recently has received little attention. In order to provide a comprehensive and Data Science friendly introduction to graph data analytics, an overview of graphs from this specific practitioner-friendly signal processing point of view is a prerequisite.

In this part of our tutorial, we have introduced graphs as irregular signal domains, together with their properties that are relevant for data analytics applications which rest upon the estimation of signals on graphs. This has been achieved in a systematic and example rich way and by highlighting links with classic matrix analysis and linear algebra. Spectral analysis of graphs has been elaborated upon in detail, as this is the main underpinning methodology for efficient data analysis, the ultimate goal in Data Science. Both the adjacency matrix and the Laplacian matrix have been used in this context, along with their spectral decompositions. Finally, we have highlighted important aspects of graph segmentation, Laplacian eigenmaps, graph cuts, graph sparsification and coarsening, and have emphasized their role as the
foundation for advances in Data Analytics and unsupervised learning on graphs.

Part II of this monograph will address theory and methods of processing data on graphs, while Part III is devoted to unsupervised graph topology learning, from the observed data, and Machine learning on graphs.

## Part II

## Signals on Graphs

## 7

## Introduction

Graphs are structures, often irregular, constructed in a way to represent the observed data and to account, in a natural way, the specific interrelationships between the data sources. However, traditional approaches have been established outside Machine Learning and Signal Processing, with which largely focus on analyzing the underlying graphs rather than dealing with signals on graphs. Moreover, given the rapidly increasing availability of multisensor and multinode measurements, likely recorded on irregular or ad-hoc grids, it would be extremely advantageous to analyze such structured data as "signals on graphs" and thus benefit from the ability of graphs to account for spatial sensing awareness, physical intuition and sensor importance, together with the inherent "local versus global" sensor association. The aim of Part II of this monograph is therefore to establish a common language between graph signals which are observed on irregular signal domains, and some of the fundamental paradigms in Learning Systems, Signal Processing and Data Analytics, such as spectral analysis, system transfer function, digital filter design, parameter estimation, and optimal denoising.

In classical Data Analytics and Signal Processing, the signal domain is determined by equidistant time instants or by a set of spatial sensing
points on a uniform grid. However, increasingly the actual data sensing domain may not even be related to the physical dimensions of time and/or space, and it typically does exhibit various forms of irregularity, as, for example, in social or web-related networks, where the sensing points and their connectivity pertain to specific objects/nodes and adhoc topology of their links. It should be noted that even for the data acquired on well defined time and space domains, the introduction of new relations between the signal samples, through graphs, may yield new insights into the analysis and provide enhanced data processing (for example, based on local similarity, through neighborhoods). We therefore set out to demonstrate that the advantage of graphs over classical data domains is that graphs account naturally and comprehensively for irregular data relations in the problem definition, together with the corresponding data connectivity in the analysis (Chen et al., 2014; Ekambaram, 2014; Gavili and Zhang, 2017; Hamon et al., 2016b; Moura, 2018; Sandryhaila and Moura, 2013; Shuman et al., 2013; Vetterli et al., 2014).

To build up the intuition behind the fundamental ideas of signals/data on graphs, a simple yet general example of multisensor temperature estimation is first considered in Section 8. Basic concepts regarding the signals and systems on graphs are presented in Section 9 , including basic definitions, operations and transforms, which generalize the foundations of traditional signal processing. Systems on graphs are interpreted starting from a comprehensive account of the existing and the introduction of a novel, isometric, graph signal shift operator. Further, graph Fourier transform is defined based on both the adjacency matrix and the graph Laplacian and it serves as the basis to introduce graph signal filtering concepts. Various ideas related to the sampling of graph signals, and particularly, the challenging topic of their subsampling, are reviewed in Section 10. Sections 12 and 13 present the concepts of time-varying signals on graphs and introduce basic definitions related to random graph signals. Localized graph signal behavior can be simultaneously characterized in the vertex-frequency domain, which is discussed in Section 14. This section also covers the important topics of local graph Fourier transform, various forms of its inversion, relations with the frames and links with the graph wavelet
transform. Energy versions of the vertex-frequency representations are also considered, along with their relations with classical time-frequency analysis.

## 8

## Problem Statement: An Illustrative Example

Consider a multi-sensor setup for measuring a temperature field in a region of interest. The temperature sensing locations are chosen according to the significance of a particular geographic area to local users, with $N=16$ sensing points in total, as shown in Figure 8.1(a). The temperature field is denoted by $\{x(n)\}$, with $n$ as the sensor index, while a snapshot of its values is given in Figure 8.1(b). Each measured sensor signal can then be mathematically expressed as

$$
\begin{equation*}
x(n)=s(n)+\varepsilon(n), \quad n=0,1, \ldots, 15, \tag{8.1}
\end{equation*}
$$

where $s(n)$ is the true temperature that would have been obtained in ideal measuring conditions and $\varepsilon(n)$ comprises the adverse effects of the local environment on sensor readings or faulty sensor activity, and is referred to as "noise" in the sequel. For illustrative purposes, in our study each $\varepsilon(n)$ was modeled as a realization of white, zero-mean, Gaussian process, with standard deviation $\sigma_{\varepsilon}=2$, that is, $\varepsilon(n) \sim \mathcal{N}(0,4)$. It was added to the signal, $s(n)$, to yield the signal-to-noise ratio in $x(n)$ of $\mathrm{SNR}_{\text {in }}=14.2 \mathrm{~dB}$.
Remark 37: Classical data analytics requires a rearrangement of the quintessentially irregular spatial temperature sensing arrangement in Figure 8.1(a) into a linear structure shown in Figure 8.1(b). Obviously,

(a)

(b)

Figure 8.1: Temperature sensing as a classic data analytics problem. (a) Sensing locations in a geographic region along the Adriatic sea. (b) Temperatures measured at $N=16$ sensing locations. In standard data estimation, the spatial sensor index is used for the horizontal axis and serves as the data domain. This domain can be interpreted as a directed path graph structure, shown in the bottom panel (c). Observe that the consecutive samples (vertices) on this path graph offer no physical intuition or interpretation, as in this "brute force" arrangement, for example, vertex 6 is located on a high mountain, whereas its neighboring vertices 5 and 7 are located along the sea; despite the consecutive index numbers these sensors are physically distant, as indicated by their very different temperature measurements.
such "lexicographic" ordering is not amenable to exploiting the information related to the actual sensor locations, which is inherently dictated by the terrain. This renders classical analyses of this multisensor temperature field inapplicable (or at best suboptimal), as the performance critically depends on the chosen sensor ordering scheme. This exemplifies that even a most routine multisensor measurement setup requires a more complex estimation structure than the standard linear one corresponding to the classical signal processing framework, shown in Figure 8.1(b).

To introduce a "situation-aware" noise reduction scheme for the temperature field in Figure 8.1, we proceed to explore a graph-theoretic framework to this problem, starting from a local signal average operator. In classical analysis, this may be achieved through a moving average operator, e.g., by averaging across the neighboring data samples, or equivalently neighboring sensors in the linear data setup in Figure 8.1(b), and for each sensing point. Physically, such local neighborhood should include close neighboring sensing points but only those which also exhibit similar meteorological properties defined by the sensor distance, altitude difference, and other terrain specific properties. In other words, since the sensor network in Figure 8.1 measures a set of related temperatures from irregularly spaced sensors, an effective estimation strategy should include domain knowledge - not possible to achieve with standard methods (linear path graph).

To illustrate the advantages of approaches based on local information (neighborhood based), consider the neighborhoods for the sensing points $n=3$ (low land), $n=6$ (mountains), and 8 (coast), shown in Figure 8.2(a). The cumulative temperature for each sensing point is then given by

$$
y(n)=\sum_{m \text { at and around } n} x(m)
$$

so that the local average temperature for a sensing point $n$ may be obtained by dividing the cumulative temperature, $y(n)$, with the number of included sensing points (size of local neighborhood). For example, for the sensing points $n=3$ and $n=6$, presented in Figure 8.2(a), the


Figure 8.2: Temperature sensing setup as a graph signal estimation problem. (a) Local neighborhood for the sensing points $n=3,6$, and 8 . These neighborhoods are chosen using "domain knowledge" dictated by the local terrain and by taking into account the sensor distance and altitude. Neighboring sensors for each of these sensing locations (vertices) are chosen in a physically meaningful way and their relation is indicated by the connectivity lines, that is, graph edges. (b) Local neighborhoods for all sensing vertices, presented in a graph form (thick lines indicate the edges from (a)).
"domain knowledge aware" local estimation takes the form

$$
\begin{align*}
& y(3)=x(3)+x(0)+x(14)+x(15)  \tag{8.2}\\
& y(6)=x(6)+x(9)+x(10) \tag{8.3}
\end{align*}
$$

For convenience, the full set of relations among the sensing points can now be arranged into a matrix form, to give

$$
\begin{equation*}
\mathbf{y}=\mathrm{x}+\mathbf{A x}, \tag{8.4}
\end{equation*}
$$

where the adjacency matrix $\mathbf{A}$, given in (8.5), indicates the connectivity structure of the sensing locations; this local connectivity structure should be involved in the calculation of each $y(n)$.

This simple real-world example can be interpreted within the graph signal processing framework as follows:

- Sensing points where the signal is measured are designated as the graph vertices, as in Figure 8.1.

$$
\mathbf{A}=\begin{gather*}
0  \tag{8.5}\\
1 \\
2 \\
3 \\
4 \\
5 \\
6 \\
7 \\
9 \\
10 \\
11 \\
12 \\
13 \\
14 \\
1
\end{gather*}\left[\begin{array}{llllllllllllllll}
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}\right]
$$

$$
\begin{aligned}
& \text { TO0 }
\end{aligned}
$$

$$
\begin{aligned}
& 00 \underset{0}{2} 00000 \underset{0}{2} 0000000 \text { N } \\
& 000000000 \underset{\substack{\mathrm{O} \\
010 \\
0}}{ } 00000
\end{aligned}
$$

$$
\begin{aligned}
& \underset{\substack{0} 000000000000}{\infty} \\
& \bigcirc \underset{O}{\circ}
\end{aligned}
$$

$$
\begin{aligned}
& \text { \| }
\end{aligned}
$$



- Vertex-to-vertex lines which indicate physically meaningful connectivity among the sensing points become the graph edges, as in Figure 8.2(a).
- The vertices and edges form a graph, as in Figure 8.2(b), a new very structurally rich signal domain.
- The graph, rather than a standard vector of sensing points, is then used for analyzing and processing data, as it exhibits both spatial and physical domain awareness.
- The measured temperatures are now interpreted as signal samples on graph, as shown in Figure 8.3.
- Similar to traditional signal processing, this new graph signal may have many realizations on the same graph and may comprise noise.
- Through relation (8.4), we have therefore introduced a simple system on a graph for physically and spatially aware signal averaging (a linear first-order system on a graph).

To emphasize our trust in a particular sensor (i.e., to model sensor relevance), a weighting scheme may be imposed, in the form

$$
\begin{equation*}
y(n)=x(n)+\sum_{m \neq n} W_{n m} x(m), \tag{2.8}
\end{equation*}
$$

where $W_{n m}$ are the elements of the weighting matrix, $\mathbf{W}$.
There are three classes of approaches to the definition of graph edges and their corresponding weights, $W_{n m}$ :

- already physically well defined edges and weights,
- definition of edges and weights based on the geometry of vertex positions,
- data similarity based methods for learning the underlying graph topology.


Figure 8.3: From a multi-sensor temperature measurement to a graph signal. The temperature field is represented on a graph that combines the spatially unaware measurements in Figure 8.1(b) and the physically relevant graph topology in Figure 8.2(b). The graph signal values are represented in two ways: (top) by vertical lines for which the length is proportional to the signal values, and (bottom) by using a "hot" colormap to designate the signal values at the vertices.

All the three approaches to define the edge weights are covered in detail in Part III of this monograph.

Since in our case of geographic temperature measurements, the graph weights do not belong to the class of obvious and physically well defined edges and weights, we will employ the "geometry of the vertices" based approach for the definition of the edges and weights. In this way, the weight elements, $W_{n m}$, for the neighboring vertices are calculated based on the horizontal vertex distance, $r_{m n}$, and the altitude difference, $h_{m n}$, as

$$
\begin{equation*}
W_{m n}=e^{-\alpha r_{m n}-\beta h_{m n}} \tag{2.9}
\end{equation*}
$$

where $\alpha$ and $\beta$ are suitable constants. The so obtained weight matrix, $\mathbf{W}$, is given in (2.6).

Based on (8.4), a weighted graph signal estimator of cumulative temperature now becomes

$$
\begin{equation*}
\mathbf{y}=\mathbf{x}+\mathbf{W} \mathbf{x} . \tag{2.10}
\end{equation*}
$$

In order to produce unbiased estimates, instead of the cumulative sums in (8.4) and (2.8), the weighting coefficients within the estimate for each $y(n)$ should sum up to unity. This can be achieved through a normalized form of (2.10), given by

$$
\begin{equation*}
\mathbf{y}=\frac{1}{2}\left(\mathbf{x}+\mathbf{D}^{-1} \mathbf{W} \mathbf{x}\right) \tag{2.11}
\end{equation*}
$$

where the elements of the diagonal normalization matrix, $\mathbf{D}$, are equal to the degree matrix elements, $D_{n n}=\sum_{m} W_{n m}$, while $\mathbf{D}^{-1} \mathbf{W}$ is a random walk (diffusion) shift operator (Stanković et al., 2018b, 2019).

Now that we have defined the graph vertices and edge weights we may resort to the data-agnostic clustering approaches, given in Part I - Section 4.3, to cluster the vertices of this graph based on the graph topology. Figure 8.4 shows the clustering result based on the three smoothest eigenvectors, $\mathbf{u}_{1}, \mathbf{u}_{2}$, and $\mathbf{u}_{3}$ (excluding the constant eigenvector, $\mathbf{u}_{0}$ ), of the graph Laplacian matrix, $\mathbf{L}=\mathbf{D}-\mathbf{W}$, given in (2.7). Notice that even such a simple graph clustering scheme was capable of identifying different physically meaningful geographic regions.


Figure 8.4: Clustering of the graph from Figure 8.2(b) based on the graph Laplacian eigenvectors, $\mathbf{u}_{1}, \mathbf{u}_{2}$, and $\mathbf{u}_{3}$. Observe the correct clustering of the graph into the clusters that belong to the seaside area (blue), low mountains (red), low land (yellow), and high mountains (green).

This also means that temperature estimation can roughly be performed within each cluster, which may even be treated as an independent graph (see graph segmentation and graph cuts in Part I, Section 4), rather than over the whole sensor network.

The above-introduced graph data estimation framework is quite general and admits application to many different scenarios where, after identifying a suitable graph topology, we desire to perform estimation on data acquired on such graphs, the subject of this part of the monograph.

## 9

## Signals and Systems on Graphs

In classical data analytics, a signal is sampled at successive, equally spaced, time instants. This then dictates the ordering of signal samples, with $x(n)$ being preceded by $x(n-1)$ and succeeded by $x(n+1)$. The "time distance" between data samples is therefore an inherent parameter in standard data processing algorithms. The relation between sampling instants can also be represented in a graph form, whereby the vertices that correspond to the instants when the signal is sampled and the corresponding edges define the linear sampling (vertex) ordering. The equally spaced nature of sampling instants in classical scenarios can then be represented with equal weights for all edges (for example, normalized to 1 ), as shown in Figure 9.1.

Algorithms defined in discrete time (like, for example, those based on the DFT or other similar data transforms), usually assume periodicity of the analyzed signals, which means that sample $x(N-1)$ is succeeded by sample $x(0)$, in a perpetual sequence. Notice that this case corresponds to the circular graph, shown in Figure 9.2, which allows us to use this model in many standard data transforms, such as the DFT, DCT, wavelets, and to define graph-counterparts of other processing algorithms, based on these transforms.


Figure 9.1: Directed path graph representation of a classical time-domain signal defined on an equidistant discrete-time grid.


Figure 9.2: Graph representation of periodic data. (a) A directed circular graph. (b) A periodic signal measured on a circular graph. Signal values, $x(n)$, are designated by vertical lines at the corresponding vertex, $n$.

A signal on a general (including also circular) undirected graph is defined by associating real (or complex) data values, $x(n)$, to each vertex, as shown in Figures 9.3 and 9.4. Such signal values can be arranged in a vector form

$$
\mathbf{x}=[x(0), x(1), \ldots, x(N-1)]^{T},
$$

so that a graph may be considered as a generalized signal domain.
This allows, in general, for any linear processing scheme for a graph signal observed at a vertex, $n$, to be defined as a linear combination of the signal value, $x(n)$, at this vertex and the signal samples, $x(m)$, at the neighboring vertices, that is

$$
\begin{equation*}
y(n)=x(n) h(n, n)+\sum_{m \in \mathcal{V}_{n}} x(m) h(m, n) \tag{9.1}
\end{equation*}
$$


(a)

(b)

Figure 9.3: Undirected circular graph (a) and signal on the graph (b). Signal values, $x(n)$, are presented as vertical lines at the corresponding vertex, $n$.


Figure 9.4: Arbitrary undirected graph (a) and signal on graph (b). Signal values, $x(n)$, are presented as vertical lines at the corresponding vertex, $n$.
where $\mathcal{V}_{n}$ is the set of vertices in the neighborhood of vertex $n$, and $h(m, n)$ are the scaling coefficients.

Remark 38: The estimation form in (9.1) is highly vertex-dependent; it is vertex-invariant only in a very specific case of regular graphs, where $\mathcal{V}_{n}$ is a $K$-neighborhood of the vertex $n$, with $h(n, m)=h(n-m)$.

We now proceed to define various forms of vertex-invariant filtering functions, using shifts on a graph. These will then be used to introduce efficient graph signal processing methods (Agaskar and Lu, 2013; Sandryhaila and Moura, 2014a,b; Segarra and Ribeiro, 2016; Venkitaraman et al., 2016; Wang et al., 2016; Yan et al., 2017).

### 9.1 Adjacency Matrix and Graph Signal Shift

Consider a graph signal, $\mathbf{x}$, for which $x(n)$ is the observed sample at a vertex $n$. A signal shift on a graph can be defined as movement of the signal sample, $x(n)$, from its original vertex, $n$, along all walks of length one, that is $K=1$, that start at vertex $n$. If the signal shifted in this way is denoted by $\mathbf{x}_{1}$, then its values can be defined using the graph adjacency matrix, $\mathbf{A}$, as

$$
\begin{equation*}
\mathbf{x}_{1}=\mathbf{A x} . \tag{9.2}
\end{equation*}
$$

Example 39: As an illustration of a graph signal and its shifted version, consider the signal on a circular graph from Figure 9.2(a). The original signal, $\mathbf{x}$, is shown in Figure $9.5(\mathrm{a})$, and its shifted version, $\mathbf{x}_{1}$, in Figure 9.5(b). Another simple signal on the undirected graph from Figure $9.4(\mathrm{a})$ is presented in Figure $9.6(\mathrm{a})$, with its shifted version, $\mathbf{x}_{1}=\mathbf{A x}$, shown in Figure 9.6(b).

A signal shifted by two graph shifts is obtained by further shifting $\mathbf{x}_{1}=\mathbf{A x}$ by one shift. The resulting, twice shifted, graph signal is then given by

$$
\mathbf{x}_{2}=\mathbf{A} \mathbf{x}_{1}=\mathbf{A}(\mathbf{A} \mathbf{x})=\mathbf{A}^{2} \mathbf{x}
$$

Therefore, in general, an $m$ times shifted signal on graph is given by

$$
\mathbf{x}_{m}=\mathbf{A} \mathbf{x}_{m-1}=\mathbf{A}^{m} \mathbf{x}
$$



Figure 9.5: Graph shift operator on a directed graph (classical circular shift). (a) Elements of a signal, $\mathbf{x}$, shown as red lines on a directed circular graph. (b) The shifted version, $\mathbf{A x}$, of the graph signal from (a). The adjacency matrix for this graph is given in (2.14) in Part I.

Remark 39: Like the standard shift operator, the second order shift of a graph signal is obtained by shifting the already once shifted signal. The role of the shift operator is assumed by the adjacency matrix, A.

Remark 40: While this section considers unweighted graphs with the adjacency matrix, A, used as a shift operator, all presented results can be directly applied to the more general class of weighted graphs, where the shift is implemented by the weight matrix, $\mathbf{W}$. The graph Laplacian as a shift operator will be considered in the next section. We will also summarize the various possible shift operators, including those based on the normalized Laplacian and random walk matrices.


(b)

Figure 9.6: Graph signal shift on an undirected graph. (a) A simple signal, x, on an undirected graph. (b) Shifted version, $\mathbf{A x}$, of the graph signal from (a).

### 9.2 Systems Based on Graph Shifted Signals

Very much like in standard linear shift-based systems, a system on a graph can be implemented as a linear combination of a graph signal, $\mathbf{x}$, and its graph shifted versions, $\mathbf{A}^{m} \mathbf{x}, m=1,2, \ldots, M-1$. The output signal from a system on a graph can then be written as

$$
\begin{equation*}
\mathbf{y}=h_{0} \mathbf{A}^{0} \mathbf{x}+h_{1} \mathbf{A}^{1} \mathbf{x}+\cdots+h_{M-1} \mathbf{A}^{M-1} \mathbf{x}=\sum_{m=0}^{M-1} h_{m} \mathbf{A}^{m} \mathbf{x} \tag{9.3}
\end{equation*}
$$

where $\mathbf{A}^{0}=\mathbf{I}$, by definition, and $h_{0}, h_{1}, \ldots, h_{M-1}$ are the system coefficients. For a circular (classical linear system) graph, this relation reduces to the well known Finite Impulse Response (FIR) filter, given by,

$$
\begin{equation*}
y(n)=h_{0} x(n)+h_{1} x(n-1)+\cdots+h_{M-1} x(n-M+1) \tag{9.4}
\end{equation*}
$$

Keeping in mind that the matrix $\mathbf{A}^{m}$ describes walks of the length $K=m$ in a graph (see Property $M_{2}$ in Part I), the output graph signal, $y(n)$, is calculated as a linear combination of the input graph
signal values and the signal values observed at vertices belonging to the ( $M-1$ )-neighborhood of the considered vertex $n$.

Remark 41: When the minimal and characteristic polynomials are of the same degree, a physically meaningful system order $(M-1)$ should be lower than the number of vertices $N$, that is, $M \leq N$. The corresponding condition in classical signal analysis would be that the number, $M$, of the system impulse response coefficients, $h_{m}$, in (9.4) should be lower or equal to the total number of signal samples, $N$ (for the graph in Figure 9.5 it means that the meaningful graph signal shifts are $m=0,1,2, \ldots, N-1$, since the shift for $m=N$ reduces to the shift for $m=0$, the shift for $m=N+1$ is equivalent to the shift for $m=1$, and so on). Therefore, in general, the system order $(M-1)$ should be lower than the degree $N_{m}$ of the minimal polynomial of the adjacency matrix A. For more detail see Part I, Section 3.1.

Remark 42: Any system of order $M-1 \geq N_{m}$ can be reduced to a system of order $N_{m}-1$.

Remark 43: If the system order is greater than or equal to the degree of the minimal polynomial, $M-1 \geq N_{m}$, then there exist more than one system producing the same output signal for a given input signal. All such systems on a graph are called equivalent.

The statements in the last three remarks will be addressed in more detail in Section 9.5, with their proofs also provided.

Example 40: Consider a signal on the graph from Figure 9.4(a), given in Figure 9.7(a), and a linear system which operates on this graph, defined by the coefficients $h_{0}=1, h_{1}=0.5$. Observe that this system on a graph corresponds to a simple classical first-order weighted moving average system. The output graph signal then represents a weighted average of the signal value at a vertex $n$ and the signal values at its $K=1$ neighborhood. The output graph signal is shown in Figure 9.7(b).
General system on graph. A system on a graph may be defined in the vertex domain as

$$
\begin{equation*}
\mathbf{y}=H(\mathbf{A}) \mathbf{x} \tag{9.5}
\end{equation*}
$$

where $H(\mathbf{A})$ is a vertex domain system (filter) function. A system on a graph is then linear and shift invariant if it satisfies the following properties of:

(a)
output signal $\mathbf{y}$
(b)

Figure 9.7: Example of vertex domain signal filtering. (a) An arbitrary graph signal. (b) The output signal obtained through a first-order (averaging) system on a graph, defined as $\mathbf{y}=\mathbf{x}+0.5 \mathbf{A x}$.

1. Linearity

$$
H(\mathbf{A})\left(a_{1} \mathbf{x}_{1}+a_{2} \mathbf{x}_{2}\right)=a_{1} \mathbf{y}_{1}+a_{2} \mathbf{y}_{2}
$$

2. Shift invariance

$$
H(\mathbf{A})[\mathbf{A x}]=\mathbf{A}[H(\mathbf{A}) \mathbf{x}]=\mathbf{A} \mathbf{y}
$$

Remark 44: A system on a graph defined by

$$
\begin{equation*}
H(\mathbf{A})=h_{0} \mathbf{A}^{0}+h_{1} \mathbf{A}^{1}+\cdots+h_{M-1} \mathbf{A}^{M-1} \tag{9.6}
\end{equation*}
$$

is linear and shift invariant since $\mathbf{A} \mathbf{A}^{m}=\mathbf{A}^{m} \mathbf{A}$.

### 9.3. Graph Fourier Transform (GFT), Adjacency Matrix Based Definition

### 9.3 Graph Fourier Transform (GFT), Adjacency Matrix Based Definition

Classical exploratory data analysis often employs estimation of signals in the spectral (Fourier) domain; this has led to a number of simple and efficient algorithms. While standard spectral analysis employs an equidistant grid in both time and frequency, following the ideas of a system on a graph, we next show that spectral domain representations of graph signals are naturally based on spectral decompositions of the adjacency matrix or graph Laplacian.

The graph Fourier transform of a signal, $\mathbf{x}$, is defined as

$$
\begin{equation*}
\mathbf{X}=\mathbf{U}^{-1} \mathbf{x} \tag{9.7}
\end{equation*}
$$

where $\mathbf{X}$ denotes a vector of the GFT coefficients, and $\mathbf{U}$ is a matrix whose columns represent the eigenvectors of the adjacency matrix, $\mathbf{A}$. Denote the elements of the vector $\mathbf{X}$ by $X(k)$, for $k=0,1, \ldots, N-1$, and recall that for undirected graphs, the adjacency matrix is symmetric, that is, $\mathbf{A}^{T}=\mathbf{A}$, and that the eigenmatrices of a symmetric matrix satisfy the property

$$
\mathbf{U}^{-1}=\mathbf{U}^{T} .
$$

Remark 45: In the analysis of directed graphs, it is usually assumed that the adjacency matrix, A, (for unweighted graphs) or the weight matrix, $\mathbf{W}$, (for weighted graphs) are diagonizable. However, these matrices are not always diagonizable, and we have to resort to using the standard Jordan normal form (Sandryhaila and Moura, 2014b). A recently proposed pragmatic approach to address this issue is to first employ the Jordan-Chevalley decomposition of a nondiagonizable matrix ( $\mathbf{A}$ or $\mathbf{W}$ ) into its diagonalizable and nilpotent parts, and subsequently use the diagonizable part (corresponding to the diagonal of the Jordan normal form) to define shifts on a (modified) graph (Misiakos et al., 2020).

The element, $X(k)$, of the graph Fourier transform vector, $\mathbf{X}$, therefore represents a projection of the considered graph signal, $x(n)$, onto
the $k$-th eigenvector of $\mathbf{A}$ (a basis function), given by

$$
\begin{equation*}
X(k)=\sum_{n=0}^{N-1} x(n) u_{k}(n) \tag{9.8}
\end{equation*}
$$

In this way, the graph Fourier transform can be interpreted as a set of projections (signal decomposition) onto the set of eigenvectors, $\mathbf{u}_{0}, \mathbf{u}_{1}, \ldots, \mathbf{u}_{N-1}$, which serve as orthonormal basis functions.

The inverse graph Fourier transform is then straightforwardly obtained from (9.7) as

$$
\begin{equation*}
\mathbf{x}=\mathbf{U} \mathbf{X} \tag{9.9}
\end{equation*}
$$

or element-wise

$$
\begin{equation*}
x(n)=\sum_{k=0}^{N-1} X(k) u_{k}(n) \tag{9.10}
\end{equation*}
$$

Observe that, for example, for a circular graph from Figure 9.2, the graph Fourier transform pair in (9.8) and (9.10) reduces to the standard discrete Fourier transform (DFT) pair. For this reason, the transform in (9.8) and its inverse in (9.10) are referred to as the graph Fourier transform (GFT) and the inverse graph Fourier transform (IGFT).

### 9.4 System on a Graph in the GFT Domain

Consider a general system on a graph defined in (9.6),

$$
\begin{equation*}
\mathbf{y}=H(\mathbf{A}) \mathbf{x}=\left(h_{0} \mathbf{A}^{0}+h_{1} \mathbf{A}^{1}+\cdots+h_{M-1} \mathbf{A}^{M-1}\right) \mathbf{x} \tag{9.11}
\end{equation*}
$$

Upon employing the spectral representation of the adjacency matrix, $\mathbf{A}=\mathbf{U} \boldsymbol{\Lambda} \mathbf{U}^{-1}$, we have

$$
\begin{align*}
\mathbf{y} & =\left(h_{0} \mathbf{U} \boldsymbol{\Lambda}^{0} \mathbf{U}^{-1}+h_{1} \mathbf{U} \boldsymbol{\Lambda}^{1} \mathbf{U}^{-1}+\cdots+h_{M-1} \mathbf{U} \mathbf{\Lambda}^{M-1} \mathbf{U}^{-1}\right) \mathbf{x} \\
& =\mathbf{U}\left(h_{0} \boldsymbol{\Lambda}^{0}+h_{1} \mathbf{\Lambda}^{1}+\cdots+h_{M-1} \boldsymbol{\Lambda}^{M-1}\right) \mathbf{U}^{-1} \mathbf{x} \\
& =\mathbf{U} H(\mathbf{\Lambda}) \mathbf{U}^{-1} \mathbf{x} \tag{9.12}
\end{align*}
$$

with the system on a graph transfer function

$$
\begin{equation*}
H(\boldsymbol{\Lambda})=h_{0} \boldsymbol{\Lambda}^{0}+h_{1} \boldsymbol{\Lambda}^{1}+\cdots+h_{M-1} \boldsymbol{\Lambda}^{M-1} \tag{9.13}
\end{equation*}
$$

where $\boldsymbol{\Lambda}$ is the matrix of eigenvalues of $\mathbf{A}$.

A pre-multiplication of this relation with $\mathbf{U}^{-1}$, yields

$$
\begin{equation*}
\mathbf{U}^{-1} \mathbf{y}=H(\mathbf{\Lambda}) \mathbf{U}^{-1} \mathbf{x} \tag{9.14}
\end{equation*}
$$

From (9.7), the terms $\mathbf{U}^{-1} \mathbf{y}$ and $\mathbf{U}^{-1} \mathbf{x}$ are respectively the GFTs of the output graph signal, $\mathbf{y}$, and the input graph signal, $\mathbf{x}$, so that the spectral domain system on a graph relation becomes

$$
\begin{equation*}
\mathbf{Y}=H(\mathbf{\Lambda}) \mathbf{X} \tag{9.15}
\end{equation*}
$$

The output graph signal in the vertex domain can then be calculated as

$$
\begin{equation*}
\mathbf{y}=H(\mathbf{A}) \mathbf{x}=\operatorname{IGFT}\{H(\mathbf{\Lambda}) \mathbf{X}\} . \tag{9.16}
\end{equation*}
$$

The element-wise form of the system on a graph in (9.15) is of the form

$$
Y(k)=\left(h_{0}+h_{1} \lambda_{k}+\cdots+h_{M-1} \lambda_{k}^{M-1}\right) X(k),
$$

where $\lambda_{k}$ denotes the $k$ th eigenvalue of the adjacency matrix, A. From (9.13) and the above equation, we can now define the transfer function of a system on a graph in the form

$$
\begin{equation*}
H\left(\lambda_{k}\right)=\frac{Y(k)}{X(k)}=h_{0}+h_{1} \lambda_{k}+\cdots+h_{M-1} \lambda_{k}^{M-1} \tag{9.17}
\end{equation*}
$$

Remark 46: The classical linear system in (9.4) can be obtained directly from its graph counterpart in (9.17) when the graph is directed and circular. This is because the adjacency matrix of a directed circular graph has eigenvalues $\lambda_{k}=e^{-j 2 \pi k / N}$ (see Part I, Section 3.2 for more detail on directed circular graphs), which are identical to the samples on the unit circle in classical DFT.

Similar to the $z$-transform in classical signal processing, for systems on graphs we can also introduce the system transfer function in the $z$-domain.
The $z$-domain transfer function of a system on a graph is defined as

$$
\begin{equation*}
H\left(z^{-1}\right)=\mathcal{Z}\left\{h_{n}\right\}=h_{0}+h_{1} z^{-1}+\cdots+h_{M-1} z^{-(M-1)} \tag{9.18}
\end{equation*}
$$

for $n=0,1, \ldots, M-1$. Obviously, from (9.17), we have

$$
H\left(\lambda_{k}\right)=\left.H\left(z^{-1}\right)\right|_{z^{-1}=\lambda_{k}} .
$$

However, the definition of the $z$-transform for arbitrary graph signals, $x(n)$ and $y(n)$, that would satisfy the relation $Y\left(z^{-1}\right)=H\left(z^{-1}\right) X\left(z^{-1}\right)$ is not straightforward, which limits the application of the $z$-transform on graphs. This will be discussed in more detail in Section 9.10.

### 9.5 Graph Signal Filtering in the Spectral Domain of the Adjacency Matrix

The energy of a graph shifted signal is given by

$$
\left\|\mathbf{x}_{1}\right\|_{2}^{2}=\|\mathbf{A} \mathbf{x}\|_{2}^{2}
$$

However, as shown in Figure 9.6, in general, the energy of a shifted signal is not the same as the energy of the original signal, that is

$$
\|\mathbf{A} \mathbf{x}\|_{2}^{2} \neq\|\mathbf{x}\|_{2}^{2}
$$

On the other hand, in graph signal processing it is often desirable that a graph shift does not increase signal energy. One such graph shift operator is introduced below.

Remark 47: Using the matrix two-norm it is straightforward to show that the ratio of energies of the graph shifted signal, $\mathbf{A x}$, and the original graph signal, $\mathbf{x}$, satisfies the relation

$$
\begin{equation*}
\max \left\{\frac{\|\mathbf{A} \mathbf{x}\|_{2}^{2}}{\|\mathbf{x}\|_{2}^{2}}\right\}=\max \left\{\frac{\mathbf{x}^{T} \mathbf{A}^{T} \mathbf{A} \mathbf{x}}{\|\mathbf{x}\|_{2}^{2}}\right\}=\lambda_{\max }^{2} \tag{9.19}
\end{equation*}
$$

where $\lambda_{\text {max }}=\max _{k}\left|\lambda_{k}\right|, k=0,1, \ldots, N-1$.

## Normalization of the Adjacency Matrix

From (9.19), for the energy of a graph shifted signal, $\|\mathbf{A} \mathbf{x}\|_{2}^{2}$, not to exceed the energy of the original graph signal, $\|\mathbf{x}\|_{2}^{2}$, we may employ the normalized adjacency matrix, defined as

$$
\begin{equation*}
\mathbf{A}_{\text {norm }}=\frac{1}{\lambda_{\max }} \mathbf{A} \tag{9.20}
\end{equation*}
$$

as a graph shift operator within any system on a graph. While this kind of normalization still does not make the shift on a graph isometric, the
energy of the signal shifted in this way is guaranteed not to be bigger than the energy of the original graph signal, since

$$
\left\|\mathbf{A}_{\text {norm }} \mathbf{x}\right\|_{2}^{2} \leq\|\mathbf{x}\|_{2}^{2} .
$$

The equality holds only for a very specific signal which is proportional to the eigenvector that corresponds to $\lambda_{\max }$.

The basic shift on a graph, system on a graph, and graph spectral domain representations can be implemented with the normalized adjacency matrix in (9.20) in the same way as with the original adjacency matrix. An important property which does not apply to standard adjacency matrices is that the normalization of adjacency matrix yields a simpler eigenvector and eigenvalue ordering scheme, as shown next.

## Spectral Ordering of Eigenvectors of the Adjacency Matrix

For physically meaningful low-pass and high-pass filtering on a graph, we need to establish the notion of spectral order. This, in turn, requires a criterion to classify the eigenvectors (corresponding to the GFT basis functions) into the slow-varying and fast-varying ones.
Remark 48: In classical Fourier analysis, the basis functions are ordered according to their frequency, whereby, for example, low-pass (slow varying) basis functions are harmonic functions characterized by low frequencies. On the other hand, the notion of frequency of the eigenvectors of the graph adjacency matrix, which serve as a basis for signal decomposition, is not defined and we have to find another criterion to classify or rank the eigenvectors. Again, we draw the inspiration from classical Fourier analysis which suggests that the energy of the "signal change" can be used instead of frequency to indicate the rate of change of an eigenvector along time.
Energy of signal change. The first graph difference can be defined for graph signals as a difference of the original graph signal and its graph shift, that is,

$$
\Delta \mathrm{x}=\mathbf{x}-\mathbf{x}_{1}=\mathbf{x}-\mathbf{A}_{\text {norm }} \mathbf{x} .
$$

In analogy to classical analysis, the energy of signal change can then be defined as the energy of the first difference of a graph signal $\mathbf{x}$, and
takes the form

$$
E_{\Delta x}=\left\|\mathbf{x}-\mathbf{A}_{\text {norm }} \mathbf{x}\right\|_{2}^{2}=\left\|\mathbf{x}-\frac{1}{\lambda_{\max }} \mathbf{A x}\right\|_{2}^{2}
$$

When the graph signal assumes a specific form of an eigenvector, $\mathbf{x}=\mathbf{u}$, of the adjacency matrix, $\mathbf{A}$, the energy of this eigenvector change is equal to

$$
\begin{equation*}
E_{\Delta u}=\left\|\mathbf{u}-\frac{1}{\lambda_{\max }} \lambda \mathbf{u}\right\|_{2}^{2}=\left|1-\frac{\lambda}{\lambda_{\max }}\right|^{2}, \tag{9.21}
\end{equation*}
$$

whereby the normalized adjacency matrix, $\mathbf{A}_{\text {norm }}$, is used to bound the energy of the shifted graph signal. In the derivation we have also used $\mathbf{A u}=\lambda \mathbf{u}$ and $\|\mathbf{u}\|_{2}^{2}=1$.

Now, the lower values of $E_{\Delta u}$ indicate that $\mathbf{u}$ is slow-varying, $E_{\Delta u}=0$ indicates that the signal is constant, while larger values of $E_{\Delta u}$ are associated with fast changes of $\mathbf{u}$ in time. The form in (9.21) is also referred to as the two-norm total variation of a basis function/eigenvector. Therefore, if the change in a basis function, $\mathbf{u}$, has a large energy, then the eigenvector, $\mathbf{u}$, can be considered to belong to the higher spectral content of the graph signal.
Remark 49: From (9.21), the energy of the rate of change of a graph signal is minimal for $\lambda=\lambda_{\max }$ and it increases as $\lambda$ decreases (see Figure 3.1 in Part I).

Now that we have established a criterion for the ordering of eigenvectors, based on the corresponding eigenvalues, we shall proceed to define an ideal low-pass filter on a graph. The intuition behind low-pass filtering in the graph domain is that such a filter should pass unchanged all signal components (eigenvectors of $\mathbf{A}$ ) for which the rates of change are slower than that defined by the cut-off eigenvalue, $\lambda_{c}$ ( $c f$. cut-off frequency), while all signal components (eigenvectors) which exhibit variations which are faster than that defined by the cut-off eigenvalue, $\lambda_{c}$, should be suppressed. The ideal low-pass filter in the graph domain is therefore defined as

$$
f(\lambda)= \begin{cases}1, & \text { for } \lambda>\lambda_{c} \\ 0, & \text { for other } \lambda\end{cases}
$$


(a) original signal, $\mathbf{x}=3.2 \mathbf{u}_{7}+2 \mathbf{u}_{6}$

(b) noisy signal, $\mathbf{x}_{\varepsilon}=\mathbf{x}+\varepsilon$

(c) filtered signal

Figure 9.8: A low-pass graph signal filtering example. (a) Original signal, $\mathbf{x}=$ $3.2 \mathbf{u}_{7}+2 \mathbf{u}_{6}$. (b) Noisy signal, $\mathbf{x}_{\varepsilon}=\mathbf{x}+\varepsilon$, at an SNR $=2.7 \mathrm{~dB}$. (c) Filtered signal, at an $\mathrm{SNR}=18.8 \mathrm{~dB}$. Ideal low-pass filtering based on the two highest eigenvalues in the pass-band was applied. Note that if $\mathbf{u}_{k}$ is an eigenvector then $-\mathbf{u}_{k}$ is also an eigenvector (eigenvectors sign ambiguity).

Example 41: Consider again the undirected graph from Figure 9.4(a) on which we observe a graph signal shown in Figure 9.8(a), which is constructed as a linear combination of two of the eigenvectors of the adjacency matrix of this graph to give $\mathbf{x}=3.2 \mathbf{u}_{7}+2 \mathbf{u}_{6}$ (eigenvectors of the adjacency matrix of the considered graph are presented in Part I,

Figure 3.1). The signal is corrupted by additive white Gaussian noise, $\varepsilon$, at the signal-to-noise (SNR) ratio of $\mathrm{SNR}_{\text {in }}=2.7 \mathrm{~dB}$ and the noisy graph signal, $\mathbf{x}_{\varepsilon}=\mathbf{x}+\varepsilon$, is shown in Figure 9.8(b). This noisy signal is next filtered using an ideal spectral domain graph filter with a cut-off eigenvalue of $\lambda_{c}=1$. The output signal, $\mathbf{x}_{f}$, is shown in Figure 9.8(c). With $\mathrm{SNR}_{\text {out }}=18.8 \mathrm{~dB}$, an increase in signal quality of 16.1 dB is achieved with this type of filtering.

Remark 50: The energy of the rate of change of an eigenvector is consistent with the classical DFT based filtering when $\lambda_{k}=\exp (-j 2 \pi k / N)$ and $\lambda_{\max }=1$.

## Spectral Domain Filter Design

We shall denote by $G(\boldsymbol{\Lambda})$ the desired graph transfer function of a system defined on a graph. Then, a system with this transfer function can be implemented either in the spectral domain or in the vertex domain.

In the spectral domain, the implementation is straightforward and can be performed in the following three steps:

1. calculate the GFT of the input graph signal, $\mathbf{X}=\mathbf{U}^{-1} \mathbf{x}$,
2. multiply the GFT of the input graph signal by the graph transfer function, $G(\boldsymbol{\Lambda})$, to obtain the output spectral form, $\mathbf{Y}=G(\boldsymbol{\Lambda}) \mathbf{X}$, and
3. calculate the output graph signal as the inverse GFT of $\mathbf{Y}$ in Step 2 , that is, $\mathbf{y}=\mathbf{U Y}$.

This procedure may be computationally very demanding for large graphs, where it may be more convenient to implement the desired filter (or its close approximation) directly in the vertex domain.

For the implementation in the vertex domain, the task is to find the coefficients ( $c f$. standard impulse response) $h_{0}, h_{1}, \ldots, h_{M-1}$ in (9.3), such that their spectral representation, $H(\boldsymbol{\Lambda})$, is equal (or approximately equal) to the desired $G(\boldsymbol{\Lambda})$. This is performed in the following way. The transfer function of the vertex domain system is given by (9.17) as $H\left(\lambda_{k}\right)=h_{0}+h_{1} \lambda_{k}^{1}+\cdots+h_{M-1} \lambda_{k}^{M-1}$ and should be equal to the
desired transfer function, $G\left(\lambda_{k}\right)$, for $k=0,1, \ldots, N-1$. This condition leads to a system of linear equations

$$
\begin{align*}
& h_{0}+h_{1} \lambda_{0}^{1}+\cdots+h_{M-1} \lambda_{0}^{M-1}=G\left(\lambda_{0}\right) \\
& h_{0}+h_{1} \lambda_{1}^{1}+\cdots+h_{M-1} \lambda_{1}^{M-1}=G\left(\lambda_{1}\right) \\
& \vdots  \tag{9.22}\\
& h_{0}+h_{1} \lambda_{N-1}^{1}+\cdots+h_{M-1} \lambda_{N-1}^{M-1}=G\left(\lambda_{N-1}\right) .
\end{align*}
$$

The matrix form of this system is then

$$
\begin{equation*}
\mathbf{V}_{\lambda} \mathbf{h}=\mathbf{g} \tag{9.23}
\end{equation*}
$$

where $\mathbf{V}_{\lambda}$ is the Vandermonde matrix form of the eigenvalues $\lambda_{k}$, given by

$$
\mathbf{V}_{\lambda}=\left[\begin{array}{cccc}
1 & \lambda_{0}^{1} & \cdots & \lambda_{0}^{M-1}  \tag{9.24}\\
1 & \lambda_{1}^{1} & \cdots & \lambda_{1}^{M-1} \\
\vdots & \vdots & \ddots & \vdots \\
1 & \lambda_{N-1}^{1} & \cdots & \lambda_{N-1}^{M-1}
\end{array}\right]
$$

and

$$
\begin{equation*}
\mathbf{h}=\left[h_{0}, h_{1}, \ldots, h_{M-1}\right]^{T} \tag{9.25}
\end{equation*}
$$

is the vector of system coefficients which need to be calculated to obtain the desired

$$
\begin{equation*}
\mathbf{g}=\left[G\left(\lambda_{0}\right), G\left(\lambda_{1}\right), \ldots, G\left(\lambda_{N-1}\right)\right]^{T}=\operatorname{diag}(G(\boldsymbol{\Lambda})) \tag{9.26}
\end{equation*}
$$

Comments on the solution in (9.22):

1. Consider the case with $N$ vertices and with all distinct eigenvalues of the adjacency matrix (in other words, the minimal polynomial is equal to the characteristic polynomial, $\left.P_{\text {min }}(\lambda)=P(\lambda)\right)$.
(a) If the filter order, $M$, is such that $M=N$, then the solution to (9.22) is unique, since the determinant of the Vandermonde matrix is always nonzero.
(b) If the filter order, $M$, is such that $M<N$, then the system in (9.22) is overdetermined. Therefore, the solution to (9.22) can only be obtained in the least squares sense (as described later in this section).
2. If some of the eigenvalues are of a degree higher than one (minimal polynomial order, $N_{m}$, is lower than the number of vertices, $N$ ) the system in (9.22) reduces to a system of $N_{m}$ linear equations (by removing multiple equations which correspond to the repeated eigenvalues $\lambda$ ).
(a) If the filter order, $M$, is such that $N_{m}<M \leq N$, the system in (9.22) is underdetermined. In that case ( $M-N_{m}$ ) filter coefficients are free variables and the system has an infinite number of solutions, while all so obtained filters are equivalent.
(b) If the filter order is such that $M=N_{m}$, the solution to the system in (9.22) is unique.
(c) If the filter order is such that $M<N_{m}$, the system in (9.22) is overdetermined and the solution is obtained in the least squares sense.
3. Any filter of an order $M>N_{m}$ has a unique equivalent filter of order $N_{m}$. This equivalent filter can be obtained by setting the free variables to zero, that is, $h_{i}=0$ for $i=N_{m}, N_{m}+1, \ldots, N-1$.

## Finding the system coefficients

Exact solution. For $M=N=N_{m}$, that is, when the filter order is equal to the number of vertices and the order of minimal polynomial, the solution to the system in (9.22) or (9.23) is unique and is obtained from

$$
\mathbf{h}=\mathbf{V}_{\lambda}^{-1} \mathbf{g}
$$

Least-squares solution. For the overdetermined case, when $M<N_{m}$, the mean-square approximation of $\mathbf{h}=\left[h_{0}, h_{1}, \ldots, h_{M-1}\right]^{T}$ in $\mathbf{V}_{\lambda} \mathbf{h}=\mathbf{g}$ is obtained by minimizing the squared error

$$
e=\left\|\mathbf{V}_{\lambda} \mathbf{h}-\mathbf{g}\right\|_{2}^{2} .
$$

From $\partial e / \partial \mathbf{h}^{T}=\mathbf{0}$ we then have

$$
\hat{\mathbf{h}}=\left(\mathbf{V}_{\lambda}^{T} \mathbf{V}_{\lambda}\right)^{-1} \mathbf{V}_{\lambda}^{T} \mathbf{g}=\operatorname{pinv}\left(\mathbf{V}_{\lambda}\right) \mathbf{g} .
$$

Since $M<N_{m}$, the obtained solution, $\hat{\mathbf{h}}$, is the least-squares approximation for $\mathbf{V}_{\lambda} \mathbf{h}=\mathbf{g}$. Given that this solution may not satisfy $\mathbf{V}_{\lambda} \mathbf{h}=\mathbf{g}$, the designed coefficient vector, $\hat{\mathbf{g}}$ (its spectrum $\hat{G}(\boldsymbol{\Lambda})$ ), obeys

$$
\mathbf{V}_{\lambda} \hat{\mathbf{h}}=\hat{\mathbf{g}}
$$

which, in general, differs from the desired system coefficients, $\mathbf{g}$ (their spectrum $G(\boldsymbol{\Lambda})$ ).

Example 42: Consider the unweighted graph from Figure 9.4(a) and the task of the synthesis of a desired filter for which the frequency response is described by

$$
\mathbf{g}=[0,0,0,0,0,0.5,1,1]^{T} .
$$

This filter was designed for various filter orders $M=1,2,4,6$, using (9.22) and the results are shown in Figure 9.9. For clarity, analytically, the vertex domain realization of the filter with $M=4$ is given by

$$
\mathbf{y}=0.1734 \mathbf{A}^{0} \mathbf{x}+0.3532 \mathbf{A}^{1} \mathbf{x}+0.0800 \mathbf{A}^{2} \mathbf{x}-0.0336 \mathbf{A}^{3} \mathbf{x}
$$

however, the exact frequency response $\hat{\mathbf{g}}=\mathbf{g}$ is only obtained with $M=N=8$.

Polynomial (Chebyshev) Approximation of the System on a Graph Transfer Function

Without loss of generality, it can be considered that the desired transfer function, $\mathbf{g}=\left[G\left(\lambda_{0}\right), G\left(\lambda_{1}\right), \ldots, G\left(\lambda_{N-1}\right)\right]^{T}$, consists of samples taken from a continuous function of $\lambda$ within the interval $\lambda_{\min } \leq \lambda \leq \lambda_{\max }$, where $\lambda_{\text {min }}$ and $\lambda_{\text {max }}$ denote the minimum and maximum values of $\left\{\lambda_{0}, \lambda_{1}, \ldots, \lambda_{N-1}\right\}$, respectively. The variable $\lambda$ of the desired transfer function, $G(\lambda)$, is continuous, and the system on graph uses only the values at discrete points $\lambda \in\left\{\lambda_{0}, \lambda_{1}, \ldots, \lambda_{N-1}\right\}$. Therefore, for a polynomial approximation, $P(\lambda)$, of the desired transfer function, $G(\lambda)$, it is important that the error at the points within the considered interval, $\lambda_{\text {min }} \leq \lambda \leq \lambda_{\text {max }}$, is bounded and sufficiently small.

This problem is known in algebra as the min-max approximation, and its goal is to find an approximating polynomial that has the smallest


Figure 9.9: Design of a graph filter with a specified transfer function in the spectral domain ( $c f$. standard frequency response). The desired spectral response, $G\left(\lambda_{k}\right)$, is denoted by blue circles. Red asterisks designate the spectral response of the filter designed in Example 42, denoted by $\hat{G}\left(\lambda_{k}\right)$, obtained with $M$ filter coefficients, $h_{0}$, $h_{1}, \ldots, h_{M-1}$, in the vertex domain.
maximum absolute error from the desired function value. The minmax polynomials can be approximated by the truncated Chebyshev polynomials, $P(\lambda)$, which yield approximations of the desired function having almost min-max behavior.

For this the reason, the approximation of the desired transfer function, $G(\lambda)$, may be performed using the truncated Chebyshev polynomial

$$
\begin{equation*}
P_{M-1}(z)=\frac{c_{0}}{2}+\sum_{m=1}^{M-1} c_{m} T_{m}(z), \tag{9.27}
\end{equation*}
$$

where $T_{m}(z)$ are the Chebyshev polynomials defined as

$$
\begin{gather*}
T_{0}(z)=1 \\
T_{1}(z)=z \\
T_{2}(z)=2 z^{2}-1 \\
T_{3}(z)=4 z^{3}-3 z \\
\vdots  \tag{9.28}\\
T_{m}(z)=2 z T_{m-1}(z)-T_{m-2}(z)
\end{gather*}
$$

with the variable $\lambda$ being centered and normalized as

$$
\begin{equation*}
z=\frac{2 \lambda-\left(\lambda_{\max }+\lambda_{\min }\right)}{\lambda_{\max }-\lambda_{\min }} \tag{9.29}
\end{equation*}
$$

such that $-1 \leq z \leq 1$ (required by the Chebyshev polynomial definition). The inverse mapping, from $z$ to $\lambda$, is given by

$$
\lambda=\frac{1}{2}\left(z\left(\lambda_{\max }-\lambda_{\min }\right)+\lambda_{\max }+\lambda_{\min }\right) .
$$

Since the Chebyshev polynomials are orthogonal, with measure $d z / \sqrt{1-z^{2}}$, the Chebyshev coefficients, $c_{m}$, for an expansion of the desired function, $G(z)$, into the polynomial series, $P_{M-1}(z)$, are easily obtained as

$$
\begin{aligned}
c_{m} & =\frac{2}{\pi} \int_{-1}^{1} G(z) T_{m}(z) \frac{d z}{\sqrt{1-z^{2}}} \\
& =\frac{2}{\pi} \int_{0}^{\pi} \cos (m \theta) G(\cos (\theta)) d \theta
\end{aligned}
$$

Example 43: Consider the unweighted graph from Figure 9.4(a) with the desired transfer function

$$
G(\lambda)=\frac{1+\operatorname{sign}\left(\lambda-\lambda_{5}\right)}{2}
$$

The samples of $G(\lambda)$ at the discrete points

$$
\lambda_{k} \in\{-2,-1.74,-1.28,-0.68,-0.41,1.11,1.81,3.19\}
$$

correspond to the values of $G\left(\lambda_{k}\right)$ in Example 42, Figure 9.9. Since the minimum and maximum eigenvalues are $\lambda_{\min }=-2$ and $\lambda_{\max }=3.19$,


Figure 9.10: Design of a graph filter with a specified transfer function in the spectral domain using the Chebyshev polynomial approximation of order $(M-1)$ with $M$ terms, $T_{0}(z), T_{1}(z), \ldots, T_{M-1}(z)$. The desired spectral response, $G(\lambda)$, is denoted by blue dashed line and blue dots. Red lines designate the spectral response of the designed Chebyshev approximation.
this yields the desired transfer function with a variable $z$ within a normalized interval, $-1 \leq z \leq 1$,

$$
G(z)=\frac{1+\operatorname{sign}\left(z-z_{5}\right)}{2}
$$

where $z_{5}$ is defined by (9.29) as

$$
z_{5}=\frac{2 \lambda_{5}-\left(\lambda_{7}+\lambda_{0}\right)}{\lambda_{7}-\lambda_{0}}=0.2
$$

The Chebyshev series for $(M-1)=3$ is given by

$$
\begin{aligned}
P_{M-1}(z) & =0.43+0.62 T_{1}(z)+0.12 T_{2}(z)-0.18 T_{3}(z) \\
& =0.31+1.16 z+0.24 z^{2}-0.72 z^{3}
\end{aligned}
$$

Upon the change of variables, $z \rightarrow \lambda$, we obtain the form

$$
\bar{P}_{M-1}(\lambda)=0.07+0.36 \lambda+0.11 \lambda^{2}-0.04 \lambda^{3}
$$

The approximations of the spectral domain transfer function of the graph filter, using the Chebyshev polynomial of order $(M-1)$, with $M$ terms, are shown in Figure 9.10, for $M=2,4,6$, and 11.



Figure 9.11: Vertex-domain filtering result for the noisy signal from Figure 9.8, using the Chebyshev approximation of the desired transfer function from Figure 9.10 with $M=4$.

Graph signal filtering can now be performed in the vertex domain using

$$
\mathbf{y}=\bar{P}_{M-1}(\mathbf{A}) \mathbf{x},
$$

where

$$
\bar{P}_{M-1}(\mathbf{A})=0.07+0.36 \mathbf{A}+0.11 \mathbf{A}^{2}-0.04 \mathbf{A}^{3} .
$$

The result of the vertex domain filtering using $\bar{P}_{M-1}(\mathbf{A})$ is shown in Figure 9.11 for the noisy signal from Figure 9.8, with the SNR improvement of 16.76 dB .
Calculation complexity. If the number of nonzero elements in the adjacency matrix, $\mathbf{A}$, is $N_{\mathbf{A}}$, then the number of arithmetic operations (additions) to calculate $\mathbf{A x}$ is of $N_{\mathbf{A}}$ order. The same number of operations is required to calculate $\mathbf{A}^{2} \mathbf{x}=\mathbf{A}(\mathbf{A x})$ using the available $\mathbf{A x}$. This means that the total number of arithmetic operations (additions) to calculate all $\mathbf{A x}, \mathbf{A}^{2} \mathbf{x}, \ldots, \mathbf{A}^{M-1} \mathbf{x}$ is of order $M N_{\mathbf{A}}$. Adding these terms requires additional $M N_{\mathbf{A}}$ arithmetic operations (additions), while the calculation of all terms of the form $c_{m} \mathbf{A}^{m} \mathbf{x}$ requires an order of $M N_{\mathbf{A}}$ multiplications by constants $c_{m}, m=0,1, \ldots, M-1$. Therefore, to calculate the output graph signal, $\mathbf{y}=\bar{P}_{M-1}(\mathbf{A}) \mathbf{x}$, an order of $2 M N_{\mathbf{A}}$ additions and $M N_{\mathbf{A}}$ multiplications is needed. Notice that the eigenanalysis of the adjacency matrix, $\mathbf{A}$, requires an order of $N^{3}$ arithmetic operations. For large graphs, the advantage in calculation complexity of the vertex domain realization with the polynomial transfer function approximation, $\mathbf{y}=\bar{P}_{M-1}(\mathbf{A}) \mathbf{x}$, is obvious.

As is common place in standard filter design theory, the transition intervals of the approximated transfer function, $G(\lambda)$, can be appropriately smoothed, to improve the approximation.

In general, the mapping in (9.29) from $\lambda$ to $z$ can be written as $z=$ $a \lambda+b$, where $a=2 /\left(\lambda_{\max }-\lambda_{\min }\right)$ and $b=-\left(\lambda_{\max }+\lambda_{\min }\right) /\left(\lambda_{\max }-\lambda_{\min }\right)$. The Chebyshev polynomial series in $\lambda$ is then of the form

$$
\begin{equation*}
\bar{P}_{M-1}(\lambda)=\frac{c_{0}}{2}+\sum_{m=1}^{M-1} c_{m} \bar{T}_{m}(\lambda) \tag{9.30}
\end{equation*}
$$

with $\bar{T}_{0}(\lambda)=1, \bar{T}_{1}(\lambda)=a \lambda+b$, and

$$
\bar{T}_{m}(\lambda)=2(a \lambda+b) \bar{T}_{m-1}(\lambda)-\bar{T}_{m-2}(\lambda)
$$

for $m \geq 2$.
The same relations hold for

$$
\begin{equation*}
\bar{P}_{M-1}(\mathbf{A})=\frac{c_{0}}{2}+\sum_{m=1}^{M-1} c_{m} \bar{T}_{m}(\mathbf{A}) \tag{9.31}
\end{equation*}
$$

This change of variables admits recursive calculation, as in (9.28).

### 9.5.1 Inverse System on a Graph

A system on a graph, $H(\boldsymbol{\Lambda})$, which represents an inverse of the system on a graph, given by $G(\boldsymbol{\Lambda})$, can be obtained from their generic relationship

$$
H(\boldsymbol{\Lambda}) G(\boldsymbol{\Lambda}) \mathbf{X}=\mathbf{X}
$$

According to (9.26), this in turn means that if all $G\left(\lambda_{k}\right) \neq 0$ and $P(\lambda)=P_{\min }(\lambda)$, then $H\left(\lambda_{k}\right)=1 / G\left(\lambda_{k}\right)$ for each $k$.

### 9.6 Graph Fourier Transform Based on the Laplacian

Similar to the graph Fourier transform based on the adjacency matrix, spectral representation of a graph signal can be alternatively based on eigenvalue decomposition of the graph Laplacian, given by

$$
\mathbf{L}=\mathbf{U} \boldsymbol{\Lambda} \mathbf{U}^{-1}
$$

or $\mathbf{L U}=\mathbf{U} \boldsymbol{\Lambda}$.

Although the analysis can be conducted in a unified way for spectral decompositions based on both the adjacency matrix and the graph Laplacian, due to their different behavior and scope of application, these will be considered separately.

The graph Fourier transform of a signal, $\mathbf{x}$, which employs the graph Laplacian eigenvalue decomposition to define its basis functions, is
given by

$$
\begin{equation*}
\mathbf{X}=\mathbf{U}^{-1} \mathbf{x} \tag{9.32}
\end{equation*}
$$

where the matrix $\mathbf{U}$ comprises in its columns the eigenvectors of the graph Laplacian. The inverse graph Fourier transform then follows immediately in the form

$$
\begin{equation*}
\mathbf{x}=\mathbf{U X} . \tag{9.33}
\end{equation*}
$$

In the case of undirected circular unweighted graph, such as the graph in Figure 9.3(a), this Laplacian based spectral analysis reduces to the standard Fourier transform, but with real-valued basis functions (eigenvectors), as shown in Part I, Section 3.3.2.

### 9.7 Ordering and Filtering in the Laplacian Spectral Domain

As shown in Section 9.5, the graph shift and the adjacency matrix are related to the first finite difference of eigenvectors in the vertex domain, while the rate of the eigenvector change is related to its corresponding eigenvalue (cf. standard frequency). A similar approach can be used for the Laplacian based eigendecomposition. We have seen that for standard time domain signals, the Laplacian of a circle graph represents the second order finite difference, $y(n)$, of a signal $u(n)$, that is

$$
y(n)=-u(n-1)+2 u(n)-u(n+1),
$$

as shown in Section 3.3 in Part I. A compact expression for all elements of the Laplacian can then be written in a matrix form as $\mathbf{y}=\mathbf{L u}$. It is now obvious that the eigenvectors, $\mathbf{u}$, which exhibit small variations should also have a small cumulative energy of the second order difference

$$
E_{u}=\sum_{n}\left[(u(n)-u(n-1))^{2}+(u(n)-u(n+1))^{2}\right] / 2 .
$$

Recall that this expression corresponds to the quadratic form of the eigenvector, $\mathbf{u}$, defined by $E_{u}=\mathbf{u}^{T} \mathbf{L} \mathbf{u}$.

The above reasoning for the Laplacian quadratic form can also be used for graph signals. As a default case for the Laplacian analysis we will consider undirected weighted graphs, where by definition

$$
\mathbf{L} \mathbf{u}=\lambda \mathbf{u}, \quad \mathbf{u}^{T} \mathbf{u}=1
$$

or

$$
\mathbf{u}^{T} \mathbf{L} \mathbf{u}=\lambda \mathbf{u}^{T} \mathbf{u}=\lambda=E_{u}
$$

This means that the quadratic form of an eigenvector, $\mathbf{u}_{k}$, is equal to its corresponding eigenvalue. This is elaborated in detail in Section 4.2 in Part I, where we have shown that

$$
\begin{equation*}
\mathbf{u}_{k}^{T} \mathbf{L} \mathbf{u}_{k}=\lambda_{k}=\frac{1}{2} \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} W_{n m}\left(u_{k}(n)-u_{k}(m)\right)^{2} \geq 0 \tag{9.34}
\end{equation*}
$$

Obviously, a small $\mathbf{u}_{k}^{T} \mathbf{L} \mathbf{u}_{k}=\lambda_{k}$ implies a small variation of $W_{n m}\left(u_{k}(n)-\right.$ $\left.u_{k}(m)\right)^{2}$ in the eigenvector $\mathbf{u}_{k}$, and for each vertex $n$. Consequently, the eigenvectors corresponding to small $\lambda_{k}$ correspond to the low-pass part of a graph signal. In other words, the smaller the smoothness index (curvature), $\mathbf{u}_{k}^{T} \mathbf{L} \mathbf{u}_{k}=\lambda_{k}$, the smoother the eigenvector, $\mathbf{u}_{k}$.

An ideal low-pass filter in the Laplacian spectrum domain, with a cut-off eigenvalue $\lambda_{c}$, can be therefore defined as

$$
f(\lambda)= \begin{cases}1, & \text { for } \lambda<\lambda_{c} \\ 0, & \text { for other } \lambda\end{cases}
$$

Example 44: Consider a signal on the undirected graph from Figure 2.2 in Part I, shown in Figure 9.12(a). This graph signal is generated as a linear combination of two Laplacian eigenvectors (which correspond to the slow-varying signal part), to give $\mathbf{x}=2 \mathbf{u}_{0}+1.5 \mathbf{u}_{1}$. The Laplacian eigenvectors of the considered graph are presented in Part I, Figure 3.4, while the considered graph signal is shown in Figure 9.12(a). The original signal, $\mathbf{x}$, was then corrupted by white Gaussian noise at the signal-to-noise ratio of $\mathrm{SNR}_{\text {in }}=-1.76 \mathrm{~dB}$, and shown in Figure 9.12(b). Next, this noisy graph signal was filtered using an ideal spectral domain graph filter, with a cut-off eigenvalue $\lambda_{c}=2$, to obtain the filtered signal, $\mathbf{x}_{f}$, shown in Figure $9.12(\mathrm{c})$. The so achieved output SNR was $\mathrm{SNR}_{\text {out }}=21.29 \mathrm{~dB}$, that is, despite its simplicity, the graph filter achieved a gain in SNR of 23.05 dB , as compared to the noisy signal in Figure 9.12(b).

To further illustrate the principle of graph filtering, the noisy signal from Figure 8.3 was filtered using a filter with the spectral cut-off at $\lambda_{c}=0.25$ and the result is shown in Figure 9.13. The same signal was

(a) original signal

(b) noisy signal

(c) filtered signal

Figure 9.12: Graph signal filtering example. (a) Original signal. (b) Noisy signal. (c) Filtered signal. Low pass filtering was performed based on the two lowest eigenvalues of the graph Laplacian.
also filtered using a polynomial approximation to the low-pass system, as illustrated in Figure 9.14.

## Laplacian versus adjacency-based GFT for regular graphs.

 A direct relation between the adjacency-based and Laplacian-based spectral decomposition can be established for $\mathcal{J}$-regular unweighted

Figure 9.13: Denoising results for the noisy signal from Figure 8.3, which was filtered using a low-pass graph filter with $\lambda_{c}=0.25$.
graphs (see Equation (2.13) in Part I), for which

$$
\mathbf{L}=\mathcal{J} \mathbf{I}-\mathbf{A}
$$

to yield

$$
\lambda_{k}^{(A)}=\mathcal{J}-\lambda_{k}^{(L)},
$$

where the eigenvalues of the adjacency matrix and the graph Laplacian are respectively denoted by $\lambda_{k}^{(A)}$ and $\lambda_{k}^{(L)}$, while they share the same eigenvectors.
Remark 51: Rank-ordering of the eigenvectors, $\mathbf{u}_{k}$, from low-pass to high-pass, which is based on the respective eigenvalues, $\lambda_{k}^{(A)}$ and $\lambda_{k}^{(L)}$, yields exactly opposite ordering for these two graph spectral decompositions. For example, the smoothest eigenvector is obtained for $\min _{k} \lambda_{k}^{(L)}=\lambda_{0}^{(L)}=0$ or for $\max _{k} \lambda_{k}^{(A)}=\lambda_{\max }=\mathcal{J}-\lambda_{0}^{(L)}=\mathcal{J}$.


Figure 9.14: Graph filtering of a noisy signal from Figure 8.3, using a fourth-order system given by $\mathbf{y}=h_{0} \mathbf{L}^{0} \mathbf{x}+h_{1} \mathbf{L}^{1} \mathbf{x}+h_{2} \mathbf{L}^{2}+h_{3} \mathbf{L}^{3}+h_{4} \mathbf{L}^{4}$.

### 9.8 Systems on a Graph Defined Using the Graph Laplacian

Following on the discussion in Section 9.2 and Equation (9.3), a system on a graph, defined using the graph Laplacian, has the form

$$
\begin{align*}
\mathbf{y} & =h_{0} \mathbf{L}^{0} \mathbf{x}+h_{1} \mathbf{L}^{1} \mathbf{x}+\cdots+h_{M-1} \mathbf{L}^{M-1} \mathbf{x} \\
& =\sum_{m=0}^{M-1} h_{m} \mathbf{L}^{m} \mathbf{x} \tag{9.35}
\end{align*}
$$

For an unweighted graph, this definition of a system on a graph can be related to the corresponding adjacency matrix form as $\mathbf{L}=\mathbf{D}-\mathbf{A}$.

The spectral domain description of a system on a graph is then obtained through the Laplacian eigenvalue decomposition, to yield

$$
\begin{align*}
\mathbf{y} & =\mathbf{U Y}=\sum_{m=0}^{M-1} h_{m} \mathbf{L}^{m} \mathbf{x}=H(\mathbf{L}) \mathbf{x} \\
& =\mathbf{U} H(\mathbf{\Lambda}) \mathbf{U}^{T} \mathbf{x}=\mathbf{U} H(\boldsymbol{\Lambda}) \mathbf{X} \tag{9.36}
\end{align*}
$$

where we used the property of the eigendecomposition of matrix polynomial,

$$
\begin{equation*}
H(\mathbf{L})=\sum_{m=0}^{M-1} h_{m} \mathbf{L}^{m}=\sum_{m=0}^{M-1} h_{m} \mathbf{U} \boldsymbol{\Lambda}^{m} \mathbf{U}^{T}=\mathbf{U} H(\boldsymbol{\Lambda}) \mathbf{U}^{T} \tag{9.37}
\end{equation*}
$$

described in Section 3.2.3 in Part I, and the notation

$$
\begin{equation*}
H(\boldsymbol{\Lambda})=\sum_{m=0}^{M-1} h_{m} \boldsymbol{\Lambda}^{m} \tag{9.38}
\end{equation*}
$$

to obtain

$$
\mathbf{Y}=H(\boldsymbol{\Lambda}) \mathbf{X}
$$

or in an element-wise form

$$
Y(k)=H\left(\lambda_{k}\right) X(k), \quad k=0,1, \ldots, N-1
$$

In the vertex domain, the $n$-th element of the output signal, $\mathbf{y}=$ $\mathbf{U} H(\boldsymbol{\Lambda}) \mathbf{U}^{T} \mathbf{x}$, of a system on a graph is given by

$$
\begin{equation*}
y(n)=\sum_{k=0}^{N-1} \sum_{i=0}^{N-1} x(i) u_{k}(i) H\left(\lambda_{k}\right) u_{k}(n)=\sum_{i=0}^{N-1} x(i) h_{n}(i) \tag{9.39}
\end{equation*}
$$

for which the transfer function is defined by

$$
\begin{equation*}
H\left(\lambda_{k}\right)=h_{0}+h_{1} \lambda_{k}+\cdots+h_{M-1} \lambda_{k}^{M-1} \tag{9.40}
\end{equation*}
$$

and the graph impulse response is

$$
\begin{equation*}
h_{n}(i)=\sum_{k=0}^{N-1} H\left(\lambda_{k}\right) u_{k}(n) u_{k}(i)=\mathcal{T}_{n}\{h(i)\} \tag{9.41}
\end{equation*}
$$

Remark 52: The expression for $y(n)$ in (9.39) can be interpreted as a generalized convolution on graphs, which is performed using a generalized graph shift of the impulse response, $h_{n}(i)$, in the vertex domain (see also Part III).

We next proceed to describe the generalized convolution on graphs through responses to the unit delta pulses. For illustration, consider the delta function located at a graph vertex $m$, given by

$$
\delta_{m}(n)= \begin{cases}1, & \text { for } m=n  \tag{9.42}\\ 0, & \text { for } m \neq n\end{cases}
$$

with the corresponding GFT

$$
\begin{equation*}
\Delta(k)=\sum_{n=0}^{N-1} \delta_{m}(n) u_{k}(n)=u_{k}(m) \tag{9.43}
\end{equation*}
$$

which is defined based on graph Laplacian eigenvectors.
Observe that, similar to the standard time domain, any graph signal can be written as a sum of delta functions at the graph vertices, that is

$$
x(n)=\sum_{i=0}^{N-1} x(i) \delta_{n}(i)
$$

or in a vector form

$$
\mathbf{x}=\sum_{i=0}^{N-1} x(i) \boldsymbol{\delta}_{i}
$$

where $\boldsymbol{\delta}_{i}$ is a vector with elements $\delta(n-i)$, as in (9.42). Then, the system output, $\mathbf{y}$, takes the form

$$
\begin{aligned}
\mathbf{y} & =\sum_{m=0}^{M-1} h_{m} \mathbf{L}^{m} \mathbf{x}=\mathbf{U} H(\boldsymbol{\Lambda}) \mathbf{U}^{T} \mathbf{x} \\
& =\sum_{i=0}^{N-1} x(i) \mathbf{U} H(\boldsymbol{\Lambda}) \mathbf{U}^{T} \boldsymbol{\delta}_{i}
\end{aligned}
$$

and its elements are obtained as

$$
y(n)=\sum_{i=0}^{N-1} x(i) \sum_{k=0}^{N-1} u_{k}(n) H\left(\lambda_{k}\right) u_{k}(i)=\sum_{i=0}^{N-1} x(i) h_{n}(i),
$$

where $h_{n}(i)$ are related to $H\left(\lambda_{k}\right)$ as in (9.41).
Remark 53: According to (9.36), the form of the graph convolution operator for a vertex $n$, given in (9.39), is localized within the $(M-1)$ neighborhood of the vertex $n$. This localization property is even more important for large graphs.

A generalized convolution for two arbitrary graph signals will be addressed next.

### 9.9 Convolution of Signals on a Graph

Consider two graph signals, $x(n)$ and $h(n)$. A generalized convolution operator for these two signals on a graph is defined using their graph

Laplacian spectra (Shuman et al., 2016), based on the assumption that the spectrum of a convolution on a graph

$$
y(n)=x(n) * h(n)
$$

is equal to the product of the corresponding spectra of graph signals, $x(n)$ and $h(n)$, that is

$$
\begin{equation*}
Y(k)=X(k) H(k) \tag{9.44}
\end{equation*}
$$

in the element-wise form. The output of the generalized graph convolution operation, $x(n) * h(n)$, is then equal to the inverse GFT of the spectral product $Y(k)$ in (9.44), that is

$$
\begin{aligned}
y(n) & =x(n) * h(n) \\
& =\sum_{k=0}^{N-1} Y(k) u_{k}(n)=\sum_{k=0}^{N-1} X(k) H(k) u_{k}(n),
\end{aligned}
$$

where

$$
\begin{equation*}
H(k)=\sum_{n=0}^{N-1} h(n) u_{k}(n) \tag{9.45}
\end{equation*}
$$

Notice the difference between the definition of $H(k)$ in (9.45) and $H\left(\lambda_{k}\right)$ in (9.40). Both these forms will be discussed in more detail in the next section.

Shift on a graph - an alternative definition. The above framework of generalized graph convolution can also serve as a basis for a convenient definition of a shift on a graph. Consider the graph signal, $h(n)$, and the delta function located at a vertex $m$. Here, we will use $h_{m}(n)$ to denote the shifted version of the graph signal, $h(n)$, "toward" a vertex $m$. This kind of shifted signal will be defined following the reasoning in classical signal processing where the shifted signal is obtained as a convolution of the original signal and an appropriately shifted delta function. Therefore, a graph shifted signal is here defined through a generalized graph convolution, $h(n) * \delta_{m}(n)$, whose GFT is equal to $H(k) u_{k}(m)$, according to (9.43) and (9.44). The graph shifted signal is then the IGFT of $H(k) u_{k}(m)$, that is

$$
\begin{equation*}
h_{m}(n)=h(n) * \delta_{m}(n)=\sum_{k=0}^{N-1} H(k) u_{k}(m) u_{k}(n) \tag{9.46}
\end{equation*}
$$

The same relation follows when calculating the inverse GFT of $X(k) H(k)$, to yield

$$
\begin{align*}
y(n) & =\sum_{k=0}^{N-1} X(k) H(k) u_{k}(n) \\
& =\sum_{k=0}^{N-1} \sum_{m=0}^{N-1} x(m) u_{k}(m) H(k) u_{k}(n) \\
& =\sum_{m=0}^{N-1} x(m) h_{m}(n)=x(n) * h(n) \tag{9.47}
\end{align*}
$$

where

$$
\begin{equation*}
h_{m}(n)=\sum_{k=0}^{N-1} H(k) u_{k}(m) u_{k}(n)=T_{m}\{h(n)\} \tag{9.48}
\end{equation*}
$$

is another version of graph shifted signal. Since the definition of $H(k)$ as a GFT of a signal $h(n)$ differs from that in (9.40), these produce different shift operations, which are respectively denoted by $T_{m}\{h(n)\}$ and $\mathcal{T}_{m}\{h(n)\}$.
Remark 54: Note that neither of the two shift operations, (9.41) or (9.48), satisfy the property that a shift by 0 is equal to the original signal, $h_{0}(n) \neq h(n)$.
Example 45: Consider a signal on the graph from Figure 9.4(a), which is defined by its graph Laplacian GFT, given by

$$
H(k)=\exp \left(-2 \lambda_{k} \tau\right)
$$

with $\tau=0.1573$. All shifted signals, $h_{m}(n)=T_{m}\{h(n)\}$, obtained using the shift operator in (9.48), are shown in Figure 9.15.

### 9.10 The $z$-Transform of a Signal on a Graph

The relation between the graph signal shift operators, $T_{m}\{h(n)\}$ and $\mathcal{T}_{m}\{h(n)\}$, which are respectively used to define the generalized convolutions in (9.40) and (9.47), can be established based on the definitions of $H\left(\lambda_{k}\right)$ and $H(k)$. Consider $H\left(\lambda_{k}\right)$, defined by (9.40), as a graph Fourier transform of signal $h(n)$. The samples of the graph signal $h(n)$ are then


Figure 9.15: An example of graph shift operator. Top: The graph signal defined by its Laplacian GFT, given by $H(k)=\exp \left(-2 \lambda_{k} \tau\right)$. Left and right column: The graph signals $h_{m}(n)$ "shifted" for $m=0$ to $m=7$, calculated using $h_{m}(n)=T_{m}\{h(n)\}$ in (9.48). The shifted signal is shown both on the vertex index line (left) and on the graph itself (right).
equal to the IGFT of $H\left(\lambda_{k}\right)$, that is

$$
h(n)=\sum_{k=0}^{N-1} H\left(\lambda_{k}\right) u_{k}(n)
$$

while the system coefficients $h_{n}, n=0,1, \ldots, M-1$, are related to $H\left(\lambda_{k}\right)$ by (9.40), that is

$$
H\left(\lambda_{k}\right)=h_{0}+h_{1} \lambda_{k}+\cdots+h_{M-1} \lambda_{k}^{M-1} .
$$

For $M=N$, the vector forms of the last two relations are

$$
\begin{gathered}
{[h(0), h(1), \ldots, h(N-1)]^{T}=\mathbf{U} H(\boldsymbol{\Lambda})} \\
H(\boldsymbol{\Lambda})=\mathbf{V}_{\lambda}\left[h_{0}, h_{1}, \ldots, h_{N-1}\right]^{T}
\end{gathered}
$$

so that the signal, $h(n)$, and the coefficients, $h_{n}$, can be related as

$$
\begin{equation*}
\left[h_{0}, h_{1}, \ldots, h_{N-1}\right]^{T}=\mathbf{V}_{\lambda}^{-1} \mathbf{U}^{T}[h(0), h(1), \ldots, h(N-1)]^{T} . \tag{9.49}
\end{equation*}
$$

Remark 55: In classical DFT (the case of a directed circular graph and its adjacency matrix, when $\mathbf{U}^{H}$ should be used instead of $\mathbf{U}^{T}$ ), the signal samples, $h(n)$, which are obtained as the inverse DFT of $H\left(\lambda_{k}\right)$ and the system coefficients, $h_{n}$, are the same, since the eigenvalues are equal to the corresponding shift operators in the spectral domain, $\lambda_{k}=\exp (-j 2 \pi k / N)$ and $u_{k}(n)=\exp (j 2 \pi n k / N) / \sqrt{N}=\lambda_{k}^{-n} / \sqrt{N}$, with $h_{n}=h(n) / \sqrt{N}$ and

$$
H(k)=\frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} h(n) e^{-j 2 \pi n k / N}
$$

Therefore, for classical DFT analysis, the following relation holds

$$
\sqrt{N} \mathbf{V}_{\lambda}=\left(\mathbf{U}^{H}\right)^{-1} .
$$

This relation is obvious from (9.24) and $u_{k}^{*}(n)=\lambda_{k}^{n} / \sqrt{N}$, and will be used to define the $z$-transform of a graph signal.
The $z$-transform of graph signals. For a given graph signal $\mathbf{x}=$ $[x(0), x(1), \ldots, x(N-1)]^{T}$, following the reasoning as in (9.49), the coefficients of a system $\left[x_{0}, x_{1}, \ldots, x_{N-1}\right]^{T}$ which corresponds to a system transfer function that would have the same GFT as the graph signal
itself are

$$
\left[x_{0}, x_{1}, \ldots, x_{N-1}\right]^{T}=\mathbf{V}_{\lambda}^{-1} \mathbf{U}^{T}[x(0), x(1), \ldots, x(N-1)]^{T}
$$

or

$$
\left[x_{0}, x_{1}, \ldots, x_{N-1}\right]^{T}=\mathbf{V}_{\lambda}^{-1}[X(0), X(1), \ldots, X(N-1)]^{T}
$$

The graph $z$-transform of a signal $\mathbf{x}$ is therefore equal to the classic $z$-transform of coefficients $\left[x_{0}, x_{1}, \ldots, x_{N-1}\right]^{T}$,

$$
\begin{equation*}
X\left(z^{-1}\right)=\mathcal{Z}\left\{x_{n}\right\}=x_{0}+x_{1} z^{-1}+\cdots+x_{N-1} z^{-(N-1)} \tag{9.50}
\end{equation*}
$$

so that the following holds

$$
Y\left(z^{-1}\right)=H\left(z^{-1}\right) X\left(z^{-1}\right)
$$

The output signal, $y(n)$, can now be obtained as

$$
[y(0), y(1), \ldots, y(N-1)]^{T}=\mathbf{U} \mathbf{V}_{\lambda}\left[y_{0}, y_{1}, \ldots, y_{N-1}\right]^{T}
$$

where the output graph signal, $y(n)$, results from the inverse $z$-transform of the coefficients, $y_{n}$, that is

$$
Y\left(z^{-1}\right)=\mathcal{Z}\left\{y_{n}\right\}=y_{0}+y_{1} z^{-1}+\cdots+y_{N-1} z^{-(N-1)}
$$

The $z$-transform representation in the complex valued $z$-domain may be of interest when the eigenvalues are complex-valued, which occurs in the decomposition of adjacency matrices of directed graphs. For example, for the graph from Figure 2.1(b) in Part I and its adjacency matrix, the eigenvalues are shown in Figure 9.16.

Definition: The analytic graph signal, $X_{a}(k)$, and the graph Hilbert transform, $X_{h}(k)$, are defined in the spectral domain as

$$
\begin{gathered}
X_{a}(k)=\left(1+\operatorname{sign}\left(\Im\left(\lambda_{k}\right)\right)\right) X(k) \\
X_{h}(k)=j \operatorname{sign}\left(\Im\left(\lambda_{k}\right)\right) X(k) \\
X(k)=X_{a}(k)+j X_{h}(k),
\end{gathered}
$$

where $\Im\left(\lambda_{k}\right)$ denotes imaginary part of $\lambda_{k}$. If these relations are applied to the standard DFT with $\lambda_{k}=e^{-j 2 \pi k / N}$ we would obtain the corresponding classical signal processing definitions.


Figure 9.16: Complex eigenvalues of the adjacency matrix of a directed graph in Figure 2.1(b) in Part I.

### 9.11 Shift Operator in the Spectral Domain

A shift operation in the spectral domain can be defined in the same way as the shift in the vertex domain. Consider a product of two graph signals, $x(n) y(n)$, defined on an undirected graph. The GFT of this product then takes the form

$$
\begin{aligned}
\operatorname{GFT}\{x(n) y(n)\} & =\sum_{n=0}^{N-1} x(n) y(n) u_{k}(n) \\
& =\sum_{n=0}^{N-1} \sum_{i=0}^{N-1} X(i) u_{i}(n) y(n) u_{k}(n)=\sum_{i=0}^{N-1} X(i) Y_{i}(k)
\end{aligned}
$$

where

$$
Y_{i}(k)=\sum_{n=0}^{N-1} y(n) u_{i}(n) u_{k}(n)
$$

can be considered as a shift of $Y(k)$ by $i$ spectral indices.

Remark 56: As desired, a shift by $i=0$ in the spectral domain produces the original value, $Y_{0}(k)=Y(k)$, up to a constant factor $1 / \sqrt{N}$. This relation does not hold for the shift operators in the vertex domain.

### 9.12 Parseval's Theorem on a Graph

Consider two graph signals, $x(n)$ and $y(n)$, which are observed on an undirected graph and their spectra, $X(k)$ and $Y(k)$. Then, Parseval's theorem has the form

$$
\begin{equation*}
\sum_{n=0}^{N-1} x(n) y(n)=\sum_{k=0}^{N-1} X(k) Y(k) \tag{9.51}
\end{equation*}
$$

and it holds for any two graph signals.
To prove Parseval's theorem on graphs, consider

$$
\begin{align*}
\sum_{n=0}^{N-1} x(n) y(n) & =\sum_{n=0}^{N-1}\left[\sum_{k=0}^{N-1} X(k) u_{k}(n)\right] y(n) \\
& =\sum_{k=0}^{N-1} X(k) \sum_{n=0}^{N-1} y(n) u_{k}(n), \tag{9.52}
\end{align*}
$$

to yield Parseval's equivalence between the energies in the original vertex and spectral domains. It has been assumed that the graphs are undirected, so that $\mathbf{U}^{-1}=\mathbf{U}^{T}$ holds. This theorem is quite general and applies to both the graph Laplacian and the adjacency matrix based decompositions on undirected graphs.

### 9.13 Optimal Denoising

Consider a measurement, $\mathbf{x}$, composed of a slow-varying graph signal, $\mathbf{s}$, and a fast changing disturbance, $\varepsilon$, to give

$$
\mathrm{x}=\mathrm{s}+\varepsilon .
$$

The aim is to design a filter for disturbance suppression (denoising), the output of which is denoted by $\mathbf{y}$.

The optimal denoising task may then be defined as a minimization of the objective function

$$
\begin{equation*}
J=\frac{1}{2}\|\mathbf{y}-\mathbf{x}\|_{2}^{2}+\alpha \mathbf{y}^{T} \mathbf{L y} . \tag{9.53}
\end{equation*}
$$

Physically, the minimization of the first term $\frac{1}{2}\|\mathbf{y}-\mathbf{x}\|_{2}^{2}$ forces the output signal $\mathbf{y}$ to be as close as possible to the available observations $\mathbf{x}$, in terms of the energy of their Euclidean distance (minimum error energy), while the second term represent a measure of smoothness of $\mathbf{y}$ (see Section 9.7). This is also physically meaningful, as the original input, $\mathbf{s}$, was low-pass and smoother than the disturbance, $\boldsymbol{\varepsilon}$. The parameter $\alpha$ provides a balance between the closeness of output, $\mathbf{y}$, to $\mathbf{x}$ and the output smoothness criterion.

To solve this minimization problem, we differentiate

$$
\frac{\partial J}{\partial \mathbf{y}^{T}}=\mathbf{y}-\mathbf{x}+2 \alpha \mathbf{L} \mathbf{y}=\mathbf{0}
$$

which results in

$$
\mathbf{y}=(\mathbf{I}+2 \alpha \mathbf{L})^{-1} \mathbf{x} .
$$

The spectral domain form of this relation follows from $\mathbf{L}=\mathbf{U}^{T} \boldsymbol{\Lambda} \mathbf{U}$, $\mathbf{Y}=\mathbf{U}^{T} \mathbf{y}$, and $\mathbf{X}=\mathbf{U}^{T} \mathbf{x}$, to yield

$$
\mathbf{Y}=(\mathbf{I}+2 \alpha \boldsymbol{\Lambda})^{-1} \mathbf{X} .
$$

The element-wise transfer function of the above spectral input/output relation then takes the form

$$
\begin{equation*}
H\left(\lambda_{k}\right)=\frac{1}{1+2 \alpha \lambda_{k}} . \tag{9.54}
\end{equation*}
$$

Remark 57: For a small $\alpha$, we have $H\left(\lambda_{k}\right) \approx 1$, that is, an all-pass behavior of (9.54), with no signal smoothing, which yields $\mathbf{y} \approx \mathbf{x}$. On the other hand, for a large $\alpha, H\left(\lambda_{k}\right) \approx \delta(k)$. The resulting $\mathbf{y} \approx$ const. is maximally smooth (a constant output, without any variation).
Example 46: The noisy signal from Figure 8.3 was filtered using the optimal filter in (9.54) with $\alpha=1$, and the result is shown in Figure 9.17. The achieved SNR was 19.16 dB .
Other cost functions. Among many possible alternatives, we will introduce two more cost functions for graph signal denoising, which exploit different constraints imposed on the solution.

Instead of enforcing the smoothness of the output signal, we may instead desire that its deviation from a linear form (which would satisfy


Figure 9.17: Graph signal denoising for a noisy signal from Figure 8.3, which is filtered using an optimal filter in (9.54), with $\alpha=1$.
$\mathbf{L y}=\mathbf{0}$ ) is as small as possible. This can be achieved with the cost function given by

$$
\begin{equation*}
J=\frac{1}{2}\|\mathbf{y}-\mathbf{x}\|_{2}^{2}+\alpha\|\mathbf{L y}\|_{2}^{2}=\frac{1}{2}\|\mathbf{y}-\mathbf{x}\|_{2}^{2}+\alpha \mathbf{y}^{T} \mathbf{L}^{2} \mathbf{y} \tag{9.55}
\end{equation*}
$$

which yields a closed form denoising solution

$$
\mathbf{y}=\left(\mathbf{I}+2 \alpha \mathbf{L}^{2}\right)^{-1} \mathbf{x}
$$

with the corresponding element-wise spectral domain relation $H\left(\lambda_{k}\right)=$ $1 /\left(1+2 \alpha \lambda_{k}^{2}\right)$.

A combination of the two cost function forms in (9.53) and (9.55), may provide additional flexibility in the design of the filter transfer function, for example

$$
J=\frac{1}{2}\|\mathbf{y}-\mathbf{x}\|_{2}^{2}+\alpha \mathbf{y}^{T} \mathbf{L} \mathbf{y}+\beta \mathbf{y}^{T} \mathbf{L}^{2} \mathbf{y}
$$

would yield the transfer function

$$
H\left(\lambda_{k}\right)=\frac{1}{1+2 \alpha \lambda_{k}+2 \beta \lambda_{k}^{2}} .
$$

This transfer function form can be further fine-tuned through the choice of the parameters $\alpha$ and $\beta$. For example, if we desire the component corresponding to $\lambda_{1} \neq 0$ not to be attenuated, we would use $\alpha+\beta \lambda_{1}=0$. Such a cost function can be straightforwardly extended to produce a transfer function for $M$ unattenuated components.

Sparsity promoting solutions. Some applications require to promote the sparsity of the output graph signal, rather than its smoothness. Such solutions then naturally rest upon compressive sensing theory which requires the two-norm in the previous cost functions to be replaced with the norms that promote sparsity. Two examples of such cost functions are

$$
\begin{equation*}
J=\frac{1}{2}\|\mathbf{y}-\mathbf{x}\|_{2}^{2}+\alpha\|\mathbf{L} \mathbf{y}\|_{p}^{p} \tag{9.56}
\end{equation*}
$$

and

$$
\begin{equation*}
J=\frac{1}{2} \sum_{n=0}^{N-1}(y(n)-x(n))^{2}+\alpha \sum_{n=0}^{N-1}\left(\sum_{m=0}^{N-1} W_{m n}(y(n)-y(m))^{2}\right)^{p / 2} \tag{9.57}
\end{equation*}
$$

with $0 \leq p \leq 1$.
Remark 58: The zero-norm, $\ell_{0}$, with $p=0$, is the best in promoting sparsity, since for $p=0$ the second term in the cost function in (9.56) counts (and minimizes) the number of nonzero elements in $\mathbf{L y}$. Minimization of the sparsity of Ly promotes constant (or linear) solutions for $\mathbf{y}$, with the smallest number of discontinuities (nonzero elements of vector $\mathbf{L y}$ ). In the second cost function in (9.57), the zero-norm promotes the smallest possible number of nonzero elements of the term $\sum_{m=0}^{N-1} W_{m n}(y(n)-y(m))^{2}$; this is also known as the total variation (TV) approach. However, the minimization of such objective functions cannot be achieved in an analytic way, like in the standard MSE case of $p=2$.

On the other hand, the choice of $p=1$ with one-norm, $\ell_{1}$, makes the above cost functions convex, allowing for gradient descend methods be used to arrive at the solution, while producing the same solution as with $p=0$, under some mild conditions. The $\ell_{1}$-norm serves as an analytic proxy to the $\ell_{0}$-norm (Kim et al., 2009).

### 9.14 Summary of Shift Operators for Systems on a Graph

The most common choices for the graph shift operator are: (i) adjacency matrix, $\mathbf{S}=\mathbf{A}$, and (ii) graph Laplacian, $\mathbf{S}=\mathbf{L}$.

Various other operators can and have been used as shift operators in systems on a graph, like: (a) normalized versions of the adjacency matrix, (b) normalized graph Laplacian, $\mathbf{S}=\mathbf{D}^{-1 / 2} \mathbf{L D}^{-1 / 2}$, (c) random walk (diffusion) matrix, $\mathbf{S}=\mathbf{D}^{-1} \mathbf{W}$ (Heimowitz and Eldar, 2017; Stanković et al., 2019b), signed Laplacian, and Laplacian for directed graphs.

Various shift operators produce corresponding eigenvector (signal decomposition) bases, such as those analyzed in Part I and given in Table 9.1.

A generalized form of the output from a system on a graph can then be written as

$$
\begin{equation*}
\mathbf{y}=h_{0} \mathbf{S}^{0} \mathbf{x}+h_{1} \mathbf{S}^{1} \mathbf{x}+\cdots+h_{M-1} \mathbf{S}^{M-1} \mathbf{x}=\sum_{m=0}^{M-1} h_{m} \mathbf{S}^{m} \mathbf{x} \tag{9.58}
\end{equation*}
$$

where, by definition $\mathbf{S}^{0}=\mathbf{I}$, while $h_{0}, h_{1}, \ldots, h_{M-1}$ are the system coefficients.

In the next sections we will consider in detail the adjacency matrix of unweighted (directed and undirected graphs) and graph Laplacian of directed graphs.

Table 9.1: Summary of graph spectral basis vectors

| Operator | Eigenanalysis |
| :--- | :--- |
| Graph Laplacian | $\mathbf{L u} \mathbf{u}_{k}=\lambda_{k} \mathbf{u}_{k}$ |
| Generalized eigenvectors | $\mathbf{L} \mathbf{u}_{k}=\lambda_{k} \mathbf{D} \mathbf{u}_{k}$ |
| $\quad$ of graph Laplacian |  |
| Normalized graph Laplacian | $\mathbf{D}^{-\frac{1}{2}} \mathbf{L D}^{-\frac{1}{2}} \mathbf{u}_{k}=\lambda_{k} \mathbf{u}_{k}$ |
| Adjacency matrix | $\mathbf{A u}_{k}=\lambda_{k} \mathbf{u}_{k}$ |
| Normalized adjacency matrix | $\left(\frac{1}{\lambda_{\text {max }}} \mathbf{A}\right) \mathbf{u}_{k}=\lambda_{k} \mathbf{u}_{k}$ |
| Laplacian for directed graphs | $\mathbf{S u}=\lambda_{k} \mathbf{u}_{k}$ |
| $\mathbf{L}=\mathbf{D}^{\text {in }}-\mathbf{W}$ | $\mathbf{S}=\mathbf{I}-\mathbf{L}$ |
| Sign Laplacian | $\mathbf{L}_{a} \mathbf{u}_{k}=\lambda_{k} \mathbf{u}_{k}$ |
| $\mathbf{L}_{a}=\mathbf{D}_{a}-\mathbf{W}$ | $D_{a}(m, m)=\sum_{n=0}^{N-1}\left\|W_{m n}\right\|$ |

## 10

## Subsampling, Compressed Sensing, and Reconstruction

Graphs may comprise of a very large number of vertices, of the order of millions or even higher. The associated computational and storage issues bring to the fore the consideration of potential advantages of signal subsampling and compressive sensing defined on graphs. We here present several basic approaches to subsampling, along with their relations to classical signal processing (Anis et al., 2016; Behjat et al., 2016; Chen et al., 2015a,b,c, 2016; Leskovec and Faloutsos, 2006; Marques et al., 2016; Narang and Ortega, 2011, 2012; Nguyen and Do, 2015; Puy et al., 2018; Sakiyama and Tanaka, 2014; Segarra et al., 2015; Stanković, 2015; Stanković et al., 2018b; Tanaka and Eldar, 2020; Tanaka and Sakiyama, 2014; Tremblay and Borgnat, 2016; Tsitsvero et al., 2016; Wang et al., 2015).

### 10.1 Subsampling of Bandlimited Graph Signals

For convenience, we shall start from the simplest case where the considered graph signal is of a bandlimited nature. Such a signal can be expressed as a linear combination of $K<N$ eigenvectors of the graph

Laplacian which exhibit the lowest smoothness indices,

$$
\begin{equation*}
x(n)=\sum_{k=0}^{K-1} X(k) u_{k}(n), \quad n=0,1, \ldots, N-1 . \tag{10.1}
\end{equation*}
$$

The GFT domain coefficients of this ( $K$-sparse) signal in the GFT domain are of the following form

$$
\begin{equation*}
\mathbf{X}=[X(0), X(1), \ldots, X(K-1), 0,0, \ldots, 0]^{T} \tag{10.2}
\end{equation*}
$$

Recall that a graph signal is sparse in the GFT domain if $K \ll N$. The smallest number of graph signal samples, $M$, needed to recover the sparse signal is therefore $M=K<N$. For stability of reconstruction, it is common to employ $K \leq M<N$ graph signal samples. The vector of available graph signal samples will be referred to as the measurement vector, and will be denoted by $\mathbf{y}$, while the set of vertices (a random subset of $\mathcal{V}=\{0,1,2, \ldots, N-1\}$ ) over which the samples of graph signal are available is denoted by

$$
\mathbb{M}=\left\{n_{1}, n_{2}, \ldots, n_{M}\right\} .
$$

The measurement matrix can now be defined using the IGFT, $\mathbf{x}=\mathbf{U X}$, of which an element-wise form is given by (10.1). The equations in (10.1) corresponding to the available graph signal samples at vertices $n \in \mathbb{M}=\left\{n_{1}, n_{2}, \ldots, n_{M}\right\}$ then define the system

$$
\left[\begin{array}{c}
x\left(n_{1}\right) \\
x\left(n_{2}\right) \\
\vdots \\
x\left(n_{M}\right)
\end{array}\right]=\left[\begin{array}{cccc}
u_{0}\left(n_{1}\right) & u_{1}\left(n_{1}\right) & \ldots & u_{N-1}\left(n_{1}\right) \\
u_{0}\left(n_{2}\right) & u_{1}\left(n_{2}\right) & \ldots & u_{N-1}\left(n_{2}\right) \\
\vdots & \vdots & \ddots & \vdots \\
u_{0}\left(n_{M}\right) & u_{1}\left(n_{M}\right) & \ldots & u_{N-1}\left(n_{M}\right)
\end{array}\right]\left[\begin{array}{c}
X(0) \\
X(1) \\
\vdots \\
X(N-1)
\end{array}\right],
$$

for which the matrix form is given by

$$
\begin{equation*}
\mathbf{y}=\mathbf{A}_{M N} \mathbf{X} \tag{10.3}
\end{equation*}
$$

where $\mathbf{A}_{M N}$ is the measurement matrix and the measurements vector

$$
\mathbf{y}=\left[\begin{array}{llll}
x\left(n_{1}\right) & x\left(n_{2}\right) & \ldots & x\left(n_{M}\right)
\end{array}\right]^{T}
$$

consists of the available graph signal samples. In general, since $M<N$ this system is underdetermined, and cannot be solved uniquely for $\mathbf{X}$ without additional constraints.

The assumption that the spectral representation of a signal contains a linear combination of only $K \leq M$ slowest varying eigenvectors allows us to exclude the GFT coefficients $X(K), X(K+1), \ldots, X(N-1)$ in (10.2) since these are zero-valued and do not contribute to the formation of graph signal samples. With this in mind, the $M \times N$ system of equations in (10.3) is reduced to the following $M \times K$ system

$$
\left[\begin{array}{c}
x\left(n_{1}\right) \\
x\left(n_{2}\right) \\
\vdots \\
x\left(n_{M}\right)
\end{array}\right]=\left[\begin{array}{cccc}
u_{0}\left(n_{1}\right) & u_{1}\left(n_{1}\right) & \ldots & u_{K-1}\left(n_{1}\right) \\
u_{0}\left(n_{2}\right) & u_{1}\left(n_{2}\right) & \ldots & u_{K-1}\left(n_{2}\right) \\
\vdots & \vdots & \ddots & \vdots \\
u_{0}\left(n_{M}\right) & u_{1}\left(n_{M}\right) & \ldots & u_{K-1}\left(n_{M}\right)
\end{array}\right]\left[\begin{array}{c}
X(0) \\
X(1) \\
\vdots \\
X(K-1)
\end{array}\right]
$$

or, in the matrix form

$$
\begin{equation*}
\mathbf{y}=\mathbf{A}_{M K} \mathbf{X}_{K} \tag{10.4}
\end{equation*}
$$

where the definitions of the reduced measurement matrix $\mathbf{A}_{M K}$ and the reduced GFT vector $\mathbf{X}_{K}$ are obvious. For $M=K$ independent measurements, this system can be solved uniquely, while for $M>K$ the system is typically overdetermined and the solution is found in the least squares (LS) sense, as Stanković et al. (2018b)

$$
\begin{equation*}
\mathbf{X}_{K}=\left(\mathbf{A}_{M K}^{T} \mathbf{A}_{M K}\right)^{-1} \mathbf{A}_{M K}^{T} \mathbf{y}=\operatorname{pinv}\left(\mathbf{A}_{M K}\right) \mathbf{y}, \tag{10.5}
\end{equation*}
$$

where $\operatorname{pinv}\left(\mathbf{A}_{M K}\right)=\left(\mathbf{A}_{M K}^{T} \mathbf{A}_{M K}\right)^{-1} \mathbf{A}_{M K}^{T}$ is the matrix pseudo-inverse of $\mathbf{A}_{M K}$.

After $\mathbf{X}_{K}$ is calculated, all GFT values follow directly as $\mathbf{X}=$ $[X(0), X(1), \ldots, X(K-1), 0,0, \ldots, 0]^{T}$, where the assumed zero values are added for $X(K), X(K+1), \ldots, X(N-1)$. The graph signal is then recovered at all vertices using $\mathbf{x}=\mathbf{U} \mathbf{X}$.
Recovery condition. The signal reconstruction in (10.5) is possible if the inverse $\left(\mathbf{A}_{M K}^{T} \mathbf{A}_{M K}\right)^{-1}$ exists, which means that

$$
\begin{equation*}
\operatorname{rank}\left(\mathbf{A}_{M K}^{T} \mathbf{A}_{M K}\right)=K \tag{10.6}
\end{equation*}
$$

In terms of the matrix condition number, this requirement is equivalent to

$$
\operatorname{cond}\left(\mathbf{A}_{M K}^{T} \mathbf{A}_{M K}\right)<\infty,
$$

that is, a nonsingular $\mathbf{A}_{M K}^{T} \mathbf{A}_{M K}$.

Remark 59: For noisy measurements of graph signals, the noise in the reconstructed GFT coefficients is directly related to the input noise and the matrix condition number. If we are able to choose the available signal sample positions (vertices), then the sampling strategy would be to find the set of measurements so that these produce the condition number which is as close to unity as possible (for stability and reduced influence of noise).
Example 47: To demonstrate the principle of reconstruction from a reduced set of graph signal samples, consider the values of a graph signal at $M=3$ vertices, given by

$$
\mathbf{y}=[x(0), x(2), x(6)]^{T}=[1.140,0.996,0.563]^{T},
$$

as shown in Figure 10.1 (upper panel). Assume that the graph signal is of a bandlimited type, with $K=2$ lowest nonzero GFT coefficients $X(0)$ and $X(1)$. The GFT coefficients of this graph signal can then be reconstructed from

$$
\begin{equation*}
\mathbf{y}=\mathbf{A}_{32} \mathbf{X}_{2}, \tag{10.7}
\end{equation*}
$$

that follows from the definition in (10.1) for the assumed available signal samples, $x(n)$, at the three vertices $n=0, n=2$, and $n=6$, for two nonzero coefficients, $X(0)$ and $X(1)$,

$$
\left[\begin{array}{l}
x(0) \\
x(2) \\
x(6)
\end{array}\right]=\left[\begin{array}{ll}
u_{0}(0) & u_{1}(0) \\
u_{0}(2) & u_{1}(2) \\
u_{0}(6) & u_{1}(6)
\end{array}\right]\left[\begin{array}{l}
X(0) \\
X(1)
\end{array}\right] .
$$

The rank of the matrix $\mathbf{A}_{32}$ is 2 . The corresponding matrix condition number is $\operatorname{cond}\left(\mathbf{A}_{32}^{T} \mathbf{A}_{32}\right)=4.33$, while the reconstructed nonzero values of the GFT are $X(0)=2$ and $X(1)=1$, to yield the reconstructed graph signal $\mathbf{x}=\mathbf{U} \mathbf{X}$, with $\mathbf{X}=[2,1,0,0,0,0,0,0]^{T}$, as shown in Figure 10.1 (lower panel).

Remark 60: For a directed circular graph, with the eigenvectors $u_{k}(n)=\exp (j 2 \pi n k / N) / \sqrt{N}$, the above downsampling and interpolation relations are identical to those in classical signal processing (Stanković, 2015).


Figure 10.1: Illustration of the subsampling of a lowpass graph signal. Top: A graph signal with missing samples at vertices $1,3,4,5$, and 7 . Bottom: The reconstructed graph signal.

### 10.2 Subsampling of Sparse Graph Signals

The subsampling of graph signals which are sparse in the GFT domain will be next considered for the cases of both known and unknown positions of the nonzero GFT coefficients. This is a generalization of the previous case with bandlimited signals when the positions of nonzero GFT coefficients are assumed to be known and located at the spectral indices from 0 to $K-1$.

## Known Coefficient Positions in GFT

The previous analysis in Section 10.1 holds not only for a bandlimited type of the graph signal, $\mathbf{x}$, and its corresponding GFT, $\mathbf{X}$, but also for case of GFT, X, with $K$ nonzero values at arbitrary, but known spectral positions, that is,

$$
X(k)=0 \quad \text { for } k \notin \mathbb{K}=\left\{k_{1}, k_{2}, \ldots, k_{K}\right\} .
$$

Similar to (10.3), the corresponding system of equations

$$
\left[\begin{array}{c}
x\left(n_{1}\right)  \tag{10.8}\\
x\left(n_{2}\right) \\
\vdots \\
x\left(n_{M}\right)
\end{array}\right]=\left[\begin{array}{cccc}
u_{k_{1}}\left(n_{1}\right) & u_{k_{2}}\left(n_{1}\right) & \ldots & u_{k_{K}}\left(n_{1}\right) \\
u_{k_{1}}\left(n_{2}\right) & u_{k_{2}}\left(n_{2}\right) & \ldots & u_{k_{K}}\left(n_{2}\right) \\
\vdots & \vdots & \ddots & \vdots \\
u_{k_{1}}\left(n_{M}\right) & u_{k_{2}}\left(n_{M}\right) & \ldots & u_{k_{K}}\left(n_{M}\right)
\end{array}\right]\left[\begin{array}{c}
X\left(k_{1}\right) \\
X\left(k_{2}\right) \\
\vdots \\
X\left(k_{K}\right)
\end{array}\right],
$$

of which the matrix form is $\mathbf{y}=\mathbf{A}_{M K} \mathbf{X}_{K}$, is solved for the nonzero spectral values $X(k), k \in \mathbb{K}$, in the same way as in the case of a bandlimited signal presented in Section 10.1.

## Support Matrices, Subsampling and Upsampling

In graph signal processing literature, the subsampling problem is often defined using the so called support matrices (Chen et al., 2015c; Lorenzo et al., 2018; Tsitsvero et al., 2016). Assume that a graph signal, x, is subsampled in such way that it is available on a subset of vertices $n \in \mathbb{M}=\left\{n_{1}, n_{2}, \ldots, n_{M}\right\}$, rather than on the full set of vertices. For this subsampled signal, we can define its upsampled version, $\mathbf{x}_{s}$, by adding zeros at the vertices where the signal is not available. Using a mathematical formalism, the subsampled and upsampled version, $\mathbf{x}_{s}$, of the original signal, $\mathbf{x}$, is then

$$
\begin{equation*}
\mathbf{x}_{s}=\mathbf{B x} \tag{10.9}
\end{equation*}
$$

where the support matrix $\mathbf{B}$ is an $N \times N$ diagonal matrix with ones at the diagonal positions which correspond to $\mathbb{M}=\left\{n_{1}, n_{2}, \ldots, n_{M}\right\}$ and zeros elsewhere. The subsampled and upsampled version, $\mathbf{x}_{s}$, of the signal $\mathbf{x}$ is obtained is such a way that the signal $\mathbf{x}$ is subsampled on a reduced set of vertices, and then upsampled by adding zeros at the original signal positions where the subsampled signal is not defined.

Recall that in general a signal, $\mathbf{x}$, with $N$ independent values cannot be reconstructed from its $M<N$ nonzero values in $\mathbf{x}_{s}$, without additional constraints. However, for graph signals which are also sparse in the GFT domain, the additional constraint is that the signal, $\mathbf{x}$, has only $K \leq M$ nonzero coefficients in the GFT domain, $\mathbf{X}=\mathbf{U}^{T} \mathbf{x}$, at $k \in \mathbb{K}=\left\{k_{1}, k_{2}, \ldots, k_{K}\right\}$, so that the relation

$$
\mathbf{X}=\mathbf{C X}
$$

holds, where the support matrix $\mathbf{C}$ is an $N \times N$ diagonal matrix with ones at the diagonal positions which correspond to $\mathbb{K}=\left\{k_{1}, k_{2}, \ldots, k_{K}\right\}$ and zeros elsewhere. Note that the presence of the GFT, $\mathbf{X}$, is on both sides of this equation, contrary to $\mathbf{x}_{s}=\mathbf{B x}$ in (10.9). The reconstruction formula then follows from

$$
\mathbf{x}_{s}=\mathbf{B x}=\mathbf{B U X}=\mathbf{B U C X}
$$

as $\mathbf{X}=\operatorname{pinv}(\mathbf{B U C}) \mathbf{x}_{s}$. The inversion

$$
\mathbf{X}=\mathbf{C X}=\operatorname{pinv}(\mathbf{B U C}) \mathbf{x}_{s}
$$

is possible for $K$ nonzero coefficients of $\mathbf{C X}$ if the rank of BUC is $K$ (if there are $K$ linearly independent equations), that is

$$
\operatorname{rank}(\mathbf{C})=K=\operatorname{rank}(\mathbf{B U C})
$$

This condition is equivalent to (10.6) since the nonzero part of matrix BUC is equal to $\mathbf{A}_{M K}$ in (10.8).

## Unknown Coefficient Positions

The reconstruction problem is more complex if the positions of nonzero spectral coefficients $\mathbb{K}=\left\{k_{1}, k_{2}, \ldots, k_{K}\right\}$ are not known. This case has been addressed within standard compressive sensing theory and can be formulated as

$$
\begin{equation*}
\min \|\mathbf{X}\|_{0} \text { subject to } \mathbf{y}=\mathbf{A}_{M N} \mathbf{X} \tag{10.10}
\end{equation*}
$$

where $\|\mathbf{X}\|_{0}$ denotes the number of nonzero elements in $\mathbf{X}$ ( $\ell_{0}$ pseudonorm).

While the ways to solve this minimization problem are manifold, we here adopt a simple, two-step approach:

1. Estimate the positions $\mathbb{K}=\left\{k_{1}, k_{2}, \ldots, k_{K}\right\}$ of the nonzero coefficients using $M>K$ signal samples.
2. Reconstruct the nonzero coefficients of $\mathbf{X}$ at the estimated positions $\mathbb{K}$, along with the signal $\mathbf{x}$ at all vertices, using the methods for the reconstruction with the known nonzero coefficient positions, described in Sections 10.1 and 10.2. The nonzero coefficients at positions $\mathbb{K}$ are calculated as $\mathbf{X}_{K}=\operatorname{pinv}\left(\mathbf{A}_{M K}\right) \mathbf{y}$.

The nonzero positions of the GFT in Step 1 can be estimated through the projection of measurements (available signal samples), $\mathbf{y}$, on the measurement matrix

$$
\mathbf{A}_{M N}=\left[\begin{array}{cccc}
u_{0}\left(n_{1}\right) & u_{1}\left(n_{1}\right) & \ldots & u_{N-1}\left(n_{1}\right) \\
u_{0}\left(n_{2}\right) & u_{1}\left(n_{2}\right) & \ldots & u_{N-1}\left(n_{2}\right) \\
\vdots & \vdots & \ddots & \vdots \\
u_{0}\left(n_{M}\right) & u_{1}\left(n_{M}\right) & \ldots & u_{N-1}\left(n_{M}\right)
\end{array}\right]
$$

to give

$$
\begin{equation*}
\mathbf{X}_{0}=\mathbf{A}_{M N}^{T} \mathbf{y} \tag{10.11}
\end{equation*}
$$

where the positions of $K$ largest values in absolute values of $\mathbf{X}_{0}$ are used as an estimate of the nonzero positions, $\mathbb{K}$.

This procedure can also be implemented in an iterative way (Stanković et al., 2018b), where:
(i) In the first iteration we assume $K=1$ and proceed to estimate the largest spectral component (absolute value) in the graph signal. Upon determining its position as $k_{1}=\operatorname{argmax}\left|\mathbf{A}_{M N}^{T} \mathbf{y}\right|$, the initially empty set of the nonzero positions becomes $\mathbb{K}=\left\{k_{1}\right\}$. The reconstructed vector $\mathbf{y}_{1}=\mathbf{A}_{1} \mathbf{X}_{1}$, where $\mathbf{X}_{1}=\operatorname{pinv}\left(\mathbf{A}_{M 1}\right) \mathbf{y}$, is then removed from the measurements, $\mathbf{y}$. In this case, the matrix $\mathbf{A}_{M 1}$ is a column of the matrix $\mathbf{A}_{M N}$ defined by the index $k_{1}$. The difference $\mathbf{e}=\mathbf{y}-\mathbf{y}_{1}$ is used as the measurement vector in the next step.
(ii) The position of the second largest spectral component in the graph signal is estimated by solving $k_{2}=\operatorname{argmax}\left|\mathbf{A}_{M N}^{T} \mathbf{e}\right|$. The set of nonzero positions now becomes $\mathbb{K}=\left\{k_{1}, k_{2}\right\}$. The first and the second component of the graph signal are now (re)estimated as $\mathbf{X}_{2}=\operatorname{pinv}\left(\mathbf{A}_{M 2}\right) \mathbf{y}$, where the matrix $\mathbf{A}_{M 2}$ is a submatrix of the measurement matrix, $\mathbf{A}_{M N}$, which consists of the columns defined by the indices $k_{1}$ and $k_{2}$. The reconstructed vector $\mathbf{y}_{2}=\mathbf{A}_{2} \mathbf{X}_{2}$, is removed from the measurements, $\mathbf{y}$, with the error, $\mathbf{e}=\mathbf{y}-\mathbf{y}_{2}$, now acting as the new measurement vector.
(iii) The procedure is iteratively repeated $K$ times or until the remaining measurement error values in $\mathbf{e}$ are negligible. In the cases


Figure 10.2: Compressive sensing on graphs. (a) Available samples (measurements), $\mathbf{y}=[x(2), x(3), x(4), x(5), x(7)]^{T}$, with missing samples at $n=0,1,6$. (b) Reconstructed signal, $\mathbf{x}$, over the whole set of vertices. (c) Initial estimate of the GFT, $X_{0}(k)$, (left), and the reconstructed sparse GFT, $X(k)$, (right).
when the sparsity, $K$, is unknown, the procedure is iterated until $\|\mathbf{e}\|_{2}<\varepsilon$, where $\varepsilon$ is a predefined precision.

Example 48: Consider a sparse graph signal, of the sparsity degree $K=2$, measured at vertices $n=2,3,4,5$, and 7 , which takes the values

$$
\mathbf{y}=[0.707,1.307,0.407,1.307,0.407]^{T},
$$

as shown in Figure 10.2(a). Our task is to reconstruct the full signal, that is, to find the missing samples $x(0), x(1)$, and $x(6)$.

To estimate positions of the nonzero elements in the GFT, $\mathbf{X}$, the initial estimate, $\mathbf{X}_{0}$, is calculated for given measurements, $\mathbf{y}$, according to (10.11). The positions of $K=2$ largest values in $\left|\mathbf{X}_{0}\right|$ are estimated as positions of the nonzero elements in $\mathbf{X}$. In the considered case,
$\mathbb{K}=\left\{k_{1}, k_{2}\right\}=\{0,3\}$, as shown in Figure 10.2(c). The GFT coefficients are then reconstructed for the sparsity degree $K=2$, as $\mathbf{X}_{2}=\operatorname{pinv}\left(\mathbf{A}_{52}\right) \mathbf{y}$, resulting in $X(0)=2, X(3)=1.2$, as illustrated in Figure 10.2(c-right). Finally, the reconstructed graph signal at all vertices, $\mathbf{x}=\mathbf{U X}$, is shown in Figure 10.2(b).

## Unique Reconstruction Conditions

When the positions of the nonzero coefficients are not known (standard compressive sensing setup), the uniqueness of the solution can be compromised. To this end, it is crucial to establish that, for a given reduced set of the graph signal samples at vertices $\mathbb{M}$, the set of nonzero positions, $\mathbb{K}$, of the sparse vector, $\mathbf{X}$, is unique. In order to define other unique reconstruction conditions, we shall consider again the solution to $\mathbf{y}=\mathbf{A}_{M N} \mathbf{X}$ which assumes a minimum number of nonzero coefficients in $\mathbf{X}$. Assume that the sparsity degree $K$ is known, then a set of $K$ measurements would yield a possible solution, $\mathbf{X}_{K}$, for any combination of $K$ nonzero coefficients in $\mathbf{X}$. For another set of $K$ measurements, we would obtain another set of possible solutions, $\mathbf{X}_{K}$. Then, a common solution between these two sets of solutions would be the solution to our problem. For a unique solution, there are no two different $K$-sparse solutions $\mathbf{X}_{K}^{(1)}$ and $\mathbf{X}_{K}^{(2)}$ if all possible matrices, $\mathbf{A}_{M 2 K}^{T} \mathbf{A}_{M 2 K}$, are nonsingular. Namely, both of these two different solutions would satisfy measurement equations,

$$
\mathbf{A}_{M 2 K}\left[\begin{array}{c}
\mathbf{X}_{K}^{(1)} \\
\mathbf{0}_{K}
\end{array}\right]=\mathbf{y} \quad \text { and } \quad \mathbf{A}_{M 2 K}\left[\begin{array}{c}
\mathbf{0}_{K} \\
\mathbf{X}_{K}^{(2)}
\end{array}\right]=\mathbf{y}
$$

where $\mathbf{A}_{M 2 K}=\left[\begin{array}{ll}\mathbf{A}_{M K}^{(1)} & \mathbf{A}_{M K}^{(2)}\end{array}\right]$. Obviously, if we subtract these two matrix equations we get a zero-vector on the right-side and a nonzero solution for the resulting vector,

$$
\mathbf{X}_{2 K}=\left[\begin{array}{c}
\mathbf{X}_{K}^{(1)} \\
-\mathbf{X}_{K}^{(2)}
\end{array}\right]
$$

requires the zero-valued determinant of $\mathbf{A}_{M 2 K}$. The nonzero determinant of $\mathbf{A}_{M 2 K}$ guarantees that two such, nonzero solutions, $\mathbf{X}_{K}^{(1)}$ and $\mathbf{X}_{K}^{(2)}$,
cannot exist. If all possible submatrices $\mathbf{A}_{M 2 K}$ of the measurement matrix $\mathbf{A}_{M K}$ are nonsingular, then two solutions of sparsity $K$ cannot exist, and the solution is unique. The requirement that all reduced measurement matrices corresponding to a $2 K$-sparse $\mathbf{X}$ are nonsingular can be written in several forms, listed below

$$
\begin{gathered}
\operatorname{det}\left\{\mathbf{A}_{M 2 K}^{T} \mathbf{A}_{M 2 K}\right\}=d_{1} d_{2} \ldots d_{2 K} \neq 0 \\
\operatorname{cond}\left\{\mathbf{A}_{M 2 K}^{T} \mathbf{A}_{M 2 K}\right\}=\frac{d_{\max }}{d_{\min }} \leq \frac{1+\delta_{2 K}}{1-\delta_{2 K}}<\infty
\end{gathered}
$$

These conditions are satisfied if $d_{\text {min }}>0$.
Remark 61: In classical compressive sensing, it is commonly assumed that the measurement matrix is normalized in such a way that the energy of each column is equal to one. Therefore, to be able to directly use and/or compare the results from classical compressive sensing to those for graph data, it is convenient to normalize the matrix $\mathbf{A}_{M K}$ so that its columns have unit energy. This is equivalent to the condition that all diagonal elements of $\mathbf{A}_{M K}^{T} \mathbf{A}_{M K}$ are equal to one.

Upon normalization, the measurement relation (10.8), becomes

$$
\mathbf{y}=\mathbf{A}_{M K} \mathbf{X}_{K}=\mathbf{A}_{M K} \mathbf{N}_{K}^{-1} \mathbf{N}_{K} \mathbf{X}_{K},
$$

where $\mathbf{N}_{K}$ is a diagonal $K \times K$ matrix, of which the elements are equal to the square root of the energy of the corresponding columns in $\mathbf{A}_{M K}$, that is, $N_{K}(k)=\sqrt{\sum_{m \in \mathbb{M}}\left|u_{k}(m)\right|^{2}}$. Upon introducing $\overline{\mathbf{A}}_{M K}=\mathbf{A}_{M K} \mathbf{N}_{K}^{-1}$ and $\overline{\mathbf{X}}_{K}=\mathbf{N}_{K} \mathbf{X}_{K}$, with the elements $\bar{u}_{k}(n)=u_{k}(n) / N_{K}(k)$ and $\bar{X}(k)=X(k) N_{K}(k)$, we obtain

$$
\begin{equation*}
\mathbf{y}=\overline{\mathbf{A}}_{M K} \overline{\mathbf{X}}_{K} . \tag{10.12}
\end{equation*}
$$

With that, we may directly use the standard compressive sensing results derived for the normalized measurement matrices. After the normalized vector of sparse elements, $\overline{\mathbf{X}}_{K}=\mathbf{N}_{K} \mathbf{X}_{K}$, is found, the reconstruction of nonzero elements is given by $\mathbf{X}_{K}=\mathbf{N}_{K}^{-1} \overline{\mathbf{X}}_{K}$, and the vertex domain signal now becomes $\mathbf{x}=\mathbf{U X}$.

The reconstruction of a $K$-sparse signal is unique if the restricted isometry property (RIP) is satisfied for a $2 K$-sparse signal, that is

$$
1-\delta_{2 K} \leq d_{\min } \leq \frac{\left\|\overline{\mathbf{A}}_{M 2 K} \overline{\mathbf{X}}_{2 K}\right\|_{2}^{2}}{\left\|\overline{\mathbf{X}}_{2 K}\right\|_{2}^{2}} \leq d_{\max } \leq 1+\delta_{2 K}
$$

where $d_{i}$ are the eigenvalues of $\overline{\mathbf{A}}_{M 2 K}^{T} \overline{\mathbf{A}}_{M 2 K}, d_{\text {min }}$ is the minimum eigenvalue, $d_{\max }$ is the maximum eigenvalue, and $\delta_{2 K}$ is the restricted isometry constant. All these conditions are satisfied if $d_{\min }>0$ or $0 \leq \delta_{2 K}<1$.

Noisy data require robust estimators, and thus more strict bounds on $d_{\text {min }}$ and $\delta_{2 K}$. For example, it has been shown that the condition $0 \leq \delta_{2 K}<0.41$ will guarantee stable inversion of $\mathbf{A}_{M 2 K}^{T} \mathbf{A}_{M 2 K}$ and consequently a robust reconstruction for noisy signals; in addition, this bound will allow for convex relaxation of the reconstruction problem (Candes, 2008). Namely, the previous problem, (10.10), can be solved using the convex relaxation from the norm-zero to a norm-one formulation given by

$$
\begin{equation*}
\min \|\mathbf{X}\|_{1} \text { subject to } \mathbf{y}=\mathbf{A}_{M N} \mathbf{X} \tag{10.13}
\end{equation*}
$$

The solutions to these two problem formulations are the same if the measurement matrix satisfies the previous conditions, with $0 \leq \delta_{2 K}<$ 0.41 .

Once the reconstruction of graph signals is formulated within the compressive sensing framework in (10.10) and (10.13), it can be solved using various well-established optimization techniques in this field, such as gradient-based approaches, Bayesian-based reconstruction, and linear programming methods (Candes, 2008; Stanković et al., 2018b).

As is the case with the standard compressive sensing problem, the initial GFT estimate, $\mathbf{X}_{0}$, will produce correct positions of the nonzero elements, $X(k)$, and the reconstruction will be unique, if

$$
K<\frac{1}{2}\left(1+\frac{1}{\mu}\right)
$$

where $\mu$ is equal to the maximum value of the inner product among any two columns of the measurement matrix, $\overline{\mathbf{A}}_{M N}$ ( $\mu$ is referred to as the coherence index) (Stankovic et al., 2020).

For illustration of the uniqueness of reconstruction, recall that a $K$-sparse signal can be written as

$$
x(n)=\sum_{i=1}^{K} X\left(k_{i}\right) u_{k_{i}}(n)=\sum_{i=1}^{K} \bar{X}\left(k_{i}\right) \bar{u}_{k_{i}}(n)
$$

of which the initial estimate in (10.11) is equal to $\overline{\mathbf{X}}_{0}=\overline{\mathbf{A}}_{M N}^{T} \mathbf{y}=$ $\overline{\mathbf{A}}_{M N}^{T} \overline{\mathbf{A}}_{M N} \overline{\mathbf{X}}$, or element-wise

$$
\bar{X}_{0}(k)=\sum_{i=1}^{K} \bar{X}\left(k_{i}\right) \sum_{n \in \mathbb{M}} \bar{u}_{k}(n) \bar{u}_{k_{i}}(n)=\sum_{i=1}^{K} \bar{X}\left(k_{i}\right) \mu\left(k, k_{i}\right),
$$

where $\mathbb{M}=\left\{n_{1}, n_{2}, \ldots, n_{M}\right\}$ and

$$
\mu\left(k, k_{i}\right)=\sum_{n \in \mathbb{M}} \bar{u}_{k}(n) \bar{u}_{k_{i}}(n) .
$$

If the maximum possible absolute value of $\mu\left(k, k_{i}\right)$ is denoted by $\mu=$ $\max \left|\mu\left(k, k_{i}\right)\right|$ (coherence index of $\left.\mathbf{A}_{M N}\right)$ then, in the worst case scenario, the amplitude of the largest component, $X\left(k_{i}\right)$, (assumed with the normalized amplitude 1), will be reduced for the maximum possible influence of other equally strong (unity) components $1-(K-1) \mu$, and should be greater than the maximum possible disturbance at $k \neq k_{i}$, which is $K \mu$. From $1-(K-1) \mu>K \mu$, the unique reconstruction condition follows; see also Stanković et al. (2018b) and Stankovic et al. (2020).

### 10.3 Measurements as Linear Combinations of Samples

It should be mentioned that if some spectrum coefficients of a graph signal are strongly related to only a few of the signal samples, then these signal samples may not be good candidates for the measurements.
Example 49: Consider a graph with one of its eigenvectors of the form close to $u_{i}(n)=\delta(n-m)$. This case is possible on graphs, in contrast to the classic DFT analysis where the basis functions are spread over all sensing instants (vertices). A similar scenario is also possible in wavelet analysis or short time Fourier transforms, which also allow for some of the transform coefficients to be related to only a few of the signal samples. In the assumed simplified case, if a considered sparse signal contains a nonzero coefficient, $X(i)$, corresponding to $u_{i}(n)=\delta(n-m)$, then all information about $X(i)$ is contained in the graph signal sample $x(m)$ only. This is prohibitive to the principle of reduced number of samples, since an arbitrary set of available samples may not contain $x(m)$.

In classical and graph data analysis this class of problems is solved by defining a more complex form of the measurements, $y(n)$, through linear combinations of all signal samples rather than the original samples themselves. In this way, each measurement, $y(n)$, will contain information about all signal samples, $x(n), n=0,1, \ldots, N-1$.

Such measurements are linear combinations of all signal samples, and are given by

$$
\left[\begin{array}{c}
y(1) \\
y(2) \\
\vdots \\
y(M)
\end{array}\right]=\left[\begin{array}{cccc}
b_{11} & b_{12} & \ldots & b_{1 N} \\
b_{21} & b_{12} & \ldots & b_{2 N} \\
\vdots & \vdots & \ddots & \vdots \\
b_{M 1} & b_{M 2} & \ldots & b_{M N}
\end{array}\right]\left[\begin{array}{c}
x(0) \\
x(1) \\
\vdots \\
x(N-1)
\end{array}\right],
$$

or in a matrix form

$$
\mathbf{y}=\mathbf{B}_{M N} \mathbf{x} .
$$

The weighting coefficients for the measurements, $b_{m n}$, in the matrix, $\mathbf{B}_{M N}$, may be, for example, drawn from a Gaussian random distribution.

For reconstruction, the sparsity of a graph signal, $\mathbf{x}$, should be again assumed in the GFT domain. The relation of the measurement vector, $\mathbf{y}$, with this sparsity domain vector of coefficients, $\mathbf{X}$, is then given by

$$
\mathbf{y}=\mathbf{B}_{M N} \mathbf{x}=\mathbf{B}_{M N} \mathbf{U} \mathbf{X}=\mathbf{A}_{M N} \mathbf{X} .
$$

The reconstruction is now obtained as a solution to

$$
\min \|\mathbf{X}\|_{0} \text { subject to } \mathbf{y}=\left(\mathbf{B}_{M N} \mathbf{U}\right) \mathbf{X}
$$

or as a solution of the corresponding convex minimization problem,

$$
\min \|\mathbf{X}\|_{1} \text { subject to } \mathbf{y}=\left(\mathbf{B}_{M N} \mathbf{U}\right) \mathbf{X},
$$

as described in Section 10.2.

### 10.4 Aggregate Sampling

A specific form of a linear combination of graph signals is referred to as aggregate sampling.

For clarity, we shall first establish an interpretation of sampling in classical signal processing through its graph counterpart - sampling on
a directed circular graph (Figure 9.2). Consider a graph signal, x, at a vertex/instant $n$. If the signal is observed at this vertex/instant only, then its value is $y_{0}(n)=x(n)$. Upon applying the graph shift operator, we have $\mathbf{y}_{1}=\mathbf{A} \mathbf{x}$, then for the same vertex, $n$, we have $y_{1}(n)=x(n-1)$. If we continue this "shift and observe" operation on the directed circular graph $N$ times at the same vertex/instant, $n$, we will eventually have all signal values $x(n), x(n-1), \ldots, x(n-N+1)$ observed at the vertex $n$.

To proceed with signal reconstruction, observe that if the shifts are stopped after $M<N$ steps, the available signal samples will be $x(n), x(n-1), \ldots, x(n-M+1)$. From this reduced set of measurements/samples we can still recover the full graph signal, $\mathbf{x}$, using compressive sensing based reconstruction methods, if the appropriate reconstruction conditions are met.

Principle of aggregate sampling on an arbitrary graph. The same procedure can be applied to a signal observed in the same way on an arbitrary graph. Assume that we observe the graph signal at only one vertex, $n$, and obtain one graph signal sample

$$
y_{0}(n)=x(n),
$$

which will be considered as the measurement $y(0)=y_{0}(n)$.
This graph signal may now be "graph shifted" to produce $\mathbf{y}_{1}=\mathbf{A x}$. Recall that in a one-step signal shift on a graph, all signal samples will move by one step along the graph edges, as described in detail in Section 9.1 and illustrated in Figure 10.3. The sample of a graph signal at vertex $n$ will now be a sum of all signal samples that have shifted to this vertex. Its value is obtained as an inner product of the $m$ th row of the adjacency matrix, $\mathbf{A}$, and the original signal vector, $\mathbf{x}$. The value of graph shifted signal at the vertex $n$, is therefore given by

$$
y_{1}(n)=\sum_{m} A_{n m} x(m),
$$

and represents a linear combination of some of the signal samples, which is now considered as the measurement $y(1)=y_{1}(n)$.

One more signal shift on the graph yields

$$
y_{2}(n)=\sum_{m} A_{n m}^{(2)} x(m),
$$


(a) signal $\mathbf{x}$

(b) shifted signal $\mathbf{A x}$

Figure 10.3: Principle of aggregate sampling. (a) A graph signal x. (b) Its graph shifted version Ax. For example, for a graph signal value observed at the vertex $n=7$ in the graph in (a) the measurement is $y(0)=x(7)$, and the aggregate measurement at the same vertex, $n=7$, after the graph signal is shifted, is equal to $y(1)=x(4)+x(5)+x(6)$ in (b). These two observations, $y(0)$ and $y(1)$, would be sufficient to reconstruct a signal whose sparsity degree is $K=2$ with nonzero values at the known spectral index positions, $k_{1}$ and $k_{2}$, if the reconstruction condition (10.6) is satisfied for the matrix $\mathbf{A}_{M N}=\mathbf{B}_{M N} \mathbf{U}$ at the specified spectral index positions.
where $A_{n m}^{(2)}$ are the elements of matrix $\mathbf{A}^{2}=\mathbf{A} \mathbf{A}$ (see Property $M_{2}$ in Part I, Section 2.3). Such an observed value, after two one-step shifts, $y_{2}(n)$ at a vertex $n$, represents a new linear combination of some signal samples and will be considered as the measurement $y(2)=y_{2}(n)$.

If we proceed with shifts $M=N$ times, a system of $N$ linear equations, $\mathbf{y}=\mathbf{B}_{M N} \mathbf{x}$, is obtained from which all signal values, $x(n)$, can be calculated. If we stop at $M<N$, the signal can still be recovered using compressive sensing based reconstruction methods if the signal is sparse and the reconstruction conditions are met.

Instead of $M$ signal samples (instants) at one vertex, we may use, for example, $P$ samples at vertex $n$ and $(M-P)$ samples from a vertex $m$. Other combinations of vertices and samples may be also used to obtain $M$ measurements and to fully reconstruct a signal.

### 10.5 Random Sampling with Optimal Strategy

Consider a realistic case of bandlimited signals on a graph. For convenience, assume that they admit a representation through linear combinations of $K$ eigenvectors with the smallest eigenvalues, that is

$$
x(n)=\sum_{k=0}^{K-1} X(k) u_{k}(n)
$$

Recall that, in graphs, the basis functions may be highly concentrated at specific vertices; this means that for adequate graph sampling some vertices are more important and are almost "must keep", while some vertices can be omitted (with a higher probability). For example, if one of the eigenvectors, for $k=0,1, \ldots, K-1$, is fully concentrated at a certain vertex, then this vertex must be included in the sampling scheme.

To this end, a sampling scheme with an adaptive strategy, proposed by Puy et al. (2018), introduces the probability, $p_{n}$, of a vertex $n$ being selected in the reduced set of signal samples (measurements) and finds its optimal value using the graph weighted coherence.

To clarify this method, consider a signal which is equal to the delta pulse at a vertex $n=m$, that is $x(n)=\delta(n-m)$. In the time domain, the energy of this signal is completely concentrated at the vertex $n=m$, with the GFT of this signal

$$
\Delta_{m}(k)=\sum_{n=0}^{N-1} x(n) u_{k}(n)=u_{k}(m) .
$$

The local graph weighted coherence then represents the energy of GFT within the first $K$ eigenvectors, and is given by

$$
\left\|\boldsymbol{\Delta}_{m}\right\|_{2}^{2}=\sum_{k=0}^{K-1}\left|u_{k}(m)\right|^{2}=\left\|\mathbf{U}_{K}^{T} \mathbf{x}\right\|_{2}^{2} \leq 1
$$

The value of energy equal to 1 indicates that there exist a bandlimited signal whose energy is completely concentrated at the vertex $n=m$, and this vertex must be used in any successful sampling scheme. The lower this value, the larger the spread of signal energy over vertices, so that we can randomly pick any of these vertices.

With the probability of picking a vertex, $p_{m}$, the graph weighted coherence can now be defined as

$$
\nu_{K}=\max _{m}\left\{p_{m}^{-1 / 2}\left\|\boldsymbol{\Delta}_{m}\right\|_{2}^{2}\right\}
$$

The optimal sampling distribution, $p_{m}^{*}$, that minimizes the graph weighted coherence is then equal to

$$
p_{m}^{*}=\frac{1}{K}\left\|\boldsymbol{\Delta}_{m}\right\|_{2}
$$

This can be seen by recognizing that the energy of $K$ normalized eigenvectors equals to $K$, that is

$$
K=\sum_{m=0}^{N-1} \sum_{k=0}^{K-1}\left|u_{k}(m)\right|^{2}
$$

since $\sum_{m=0}^{N-1}\left|u_{k}(m)\right|^{2}=1$, by definition. The above expression for $p_{m}^{*}$ follows from

$$
\begin{aligned}
K & =\sum_{m=0}^{N-1} \sum_{k=0}^{K-1}\left|u_{k}(m)\right|^{2}=\sum_{m=0}^{N-1} \sum_{k=0}^{K-1} p_{m} \frac{\left|u_{k}(m)\right|^{2}}{p_{m}} \\
& \leq \max _{m}\left\{\sum_{m=0}^{N-1} \sum_{k=0}^{K-1} \frac{\left|u_{k}(m)\right|^{2}}{p_{m}}\right\} \sum_{m=1}^{N-1} p_{m}=\nu_{K}^{2}
\end{aligned}
$$

The sampling distribution, $p_{n}^{*}$, is optimal in the sense that the number of measurements needed to embed the set of $K$-bandlimited signals is effectively reduced to its minimum value (Puy et al., 2018).

Example 50: Consider a very simple case of a graph with three disconnected components (sub-graphs). The number of vertices in these graph components is $N_{1}=1$ vertex, $N_{2}=2$ vertices, and $N_{3}=4$ vertices, denoted by $n=0,1,2,3,4,5,6$, respectively. Assume that the signal is constant over the three graph components, and only the three $(K=3)$ lowest eigenvalues with $\lambda_{0}=\lambda_{1}=\lambda_{2}=0$ are considered.

The corresponding nonzero eigenvector elements are $u_{0}(n)=1$, for $n=0, u_{1}(n)=1 / \sqrt{2}$, for $n=1,2$, and $u_{2}(n)=1 / \sqrt{4}$, for $n=3,4,5,6$.

Intuitively, in order to recover this graph signal we must have at least $M=3$ samples; also each graph component should contain a sample.

We will now compare the two strategies: (1) Fully random selection of samples; and (2) Selection of samples with probabilities defined as

$$
p_{m}^{*}=\frac{1}{K}\left\|\boldsymbol{\Delta}_{m}\right\|_{2}^{2}=\frac{1}{K} \sum_{k=0}^{K-1}\left|u_{k}(m)\right|^{2}=\frac{1}{3} \sum_{k=0}^{2}\left|u_{k}(m)\right|^{2}
$$

which results in $p_{m}^{*}=(1 / 3,1 / 6,1 / 6,1 / 12,1 / 12,1 / 12,1 / 12)$ for $m=$ $(0,1,2,3,4,5,6)$. Obviously, when using the optimal sampling strategy the probability of selecting $M=3$ samples from different sets is much higher than when the samples are chosen randomly with $p_{m}=(1 / 7,1 / 7,1 / 7,1 / 7,1 / 7,1 / 7,1 / 7)$.

This kind of variable sampling density is also used in classical compressive sensing to improve the density of samples where this is needed due to signal variations. Two variants of this approach are used: (1) with vertex replacement, when every selected vertex can be chosen again; and (2) without vertex replacement, when a vertex can be selected only once. Within the compressive sensing framework, taking some samples several times means that these vertices are considered with higher importance. Since the selection process accounts for the importance of the vertices, we will assume the latter approach.

For large graphs, it is also very important to try to avoid the eigenvector calculation and even to estimate the factor $\left\|\boldsymbol{\Delta}_{m}\right\|_{2}^{2}$ without the eigendecomposition. One such method was introduced by Puy et al. (2018) using random vectors and the property that an average of the power of several random vectors, filtered by bandlimited filters to the $K$ lowest eigenvectors, can estimate the value of $\left\|\boldsymbol{\Delta}_{m}\right\|_{2}^{2}$. These filters can be implemented using the described graph shifts.

Having in mind that the set of vertices, $\mathcal{M}$, needs to be selected only once to sample all $K$-bandlimited signals on a graph $\mathcal{G}$, the problem of the uniqueness of solution does not exist in this sampling setup, in contrast to classical compressive sensing (meaning that in this case there
is no need for sophisticated tools like the restricted isometry property, since there is no possibility that two different sets of $K$ nonzero elements, with different indices $k$, satisfy the same measurements). Namely, here we assume that the signal is bandlimited and with known indices of nonzero spectral elements, $X(k)$, that is, $X(k)$ may assume nonzero values only for $k=0,1,2, \ldots, K-1$. Once optimal sampling vertices are selected, the problem reduces to the solution of

$$
\begin{equation*}
\mathbf{y}=\mathbf{A}_{M K} \mathbf{X}_{K} \tag{10.14}
\end{equation*}
$$

where the reduced measurement matrix, $\mathbf{A}_{M K}$, is well defined, without any random combination of indices, while the reduced GFT vector, $\mathbf{X}_{K}$, is already explained, with $\mathbf{y}$ containing the signal values at the selected vertices. Since $M \geq K$, the stability of this system solution is well studied within linear algebra, with the condition number being the best parameter for its description.

The reconstruction is performed using

$$
\mathbf{X}_{K}=\operatorname{pinv}\left(\mathbf{A}_{M K}\right) \mathbf{y}
$$

and

$$
\mathbf{x}=\mathbf{U}_{K} \mathbf{X}_{K}
$$

If the measurements, $\mathbf{y}$, are noisy, then

$$
\mathbf{y}+\varepsilon=\mathbf{A}_{M K}\left(\mathbf{X}_{K}+\mathbf{X}_{N}\right)
$$

where $\varepsilon$ denotes additive noise, and $\mathbf{X}_{K}+\mathbf{X}_{N}$ is the reconstructed vector which contains the true elements, $\mathbf{X}_{K}$, and the noise in the reconstruction, $\mathbf{X}_{N}$. Then, the output signal-to-noise ratio is defined by the condition number (ratio of the maximum and the minimum eigenvalue, $d_{\max }$ and $d_{\min }$, of the matrix $\mathbf{A}_{M K}^{T} \mathbf{A}_{M K}$ ), to yield

$$
-10 \log \left(\frac{d_{\max }}{d_{\mathrm{min}}}\right) \leq \mathrm{SNR}_{i}-\mathrm{SNR} \leq 10 \log \left(\frac{d_{\max }}{d_{\min }}\right)
$$

where the input and output signal-to-noise ratios are defined as

$$
\mathrm{SNR}_{i}=10 \log \left(\|\mathbf{y}\|_{2}^{2} /\|\varepsilon\|_{2}^{2}\right)
$$

and

$$
\mathrm{SNR}=10 \log \left(\left\|\mathbf{X}_{K}\right\|_{2}^{2} /\left\|\mathbf{X}_{N}\right\|_{2}^{2}\right)
$$

Here, we have assumed that the columns of the measurement matrix are energy normalized, for notation simplicity.

Other strategies for random sampling can be found, for example, in Chen et al. (2015b, 2016) and Tanaka and Eldar (2020).

## 11

## Filter Bank on a Graph

Subsampling and upsampling are the two standard operators used to alter the scale at which the signal is processed. Subsampling of a signal by a factor of 2 , followed by the corresponding upsampling, can be described in classical signal processing by

$$
f(n)=\frac{1}{2}\left(x(n)+(-1)^{n} x(n)\right)=\frac{1}{2}\left(\left(1+(-1)^{n}\right) x(n)\right),
$$

as illustrated in Figure 11.1.
This is the basic operation used in multiresolution approaches based on filter banks and can be extended to signals on graphs in the following way. Consider a graph with the set of vertices $\mathcal{V}$. Any set of vertices can be considered as a union of two disjoint subsets $\mathcal{E}$ and $\mathcal{H}$, such that $\mathcal{V}=\mathcal{E} \cup \mathcal{H}$ and $\mathcal{E} \cap \mathcal{H}=\emptyset$. The subsampling-upsampling procedure can then be performed in the following two steps:

1. Subsample the signal on a graph by keeping only signal values on the vertices $n \in \mathcal{E}$, while not altering the original graph topology.
2. Upsample the graph signal by setting the signal values for the vertices $n \notin \mathcal{E}$ to zero.


Figure 11.1: Principle of a signal, $x(n)$, downsampling and upsampling in the classical time domain.

This combined subsampling-upsampling operation produces a graph signal

$$
f(n)=\frac{1}{2}\left(1+(-1)^{\beta_{\mathcal{E}}(n)}\right) x(n)
$$

where

$$
\beta_{\mathcal{E}}(n)= \begin{cases}0, & \text { if } n \in \mathcal{E} \\ 1, & \text { if } n \in \mathcal{H}\end{cases}
$$

The values of the resulting graph signal, $f(n)$, are therefore $f(n)=x(n)$ if $n \in \mathcal{E}$ and $f(n)=0$ elsewhere.

The vector form of the subsampled-upsampled graph signal, $f(n)$, which comprises all $n \in \mathcal{V}$, is given by

$$
\begin{equation*}
\mathbf{f}=\frac{1}{2}\left(\mathbf{x}+\mathbf{J}_{\mathcal{E}} \mathbf{x}\right)=\frac{1}{2}\left(\mathbf{I}+\mathbf{J}_{\mathcal{E}}\right) \mathbf{x} \tag{11.1}
\end{equation*}
$$

where $\mathbf{J}_{\mathcal{E}}=\operatorname{diag}\left((-1)^{\beta_{\mathcal{E}}(n)}\right), n \in \mathcal{V}$.
The focus of our analysis will be on the two-channel wavelet filter bank on a graph, shown in Figure 11.2. As in the classical wavelet analysis framework for the time domain signals, such a filter bank


Figure 11.2: Principle of a filter bank for a graph signal.
provides decomposition of a graph signal into the corresponding lowpass (smooth) and high-pass (fast-varying) constituents. The analysis side (left part of the system in Figure 11.2) consists of two channels with filters characterized by the vertex domain operators $H_{L}(\mathbf{L})$ and $H_{H}(\mathbf{L})$, with the corresponding spectral domain operators $H_{L}(\boldsymbol{\Lambda})$ and $H_{H}(\boldsymbol{\Lambda})$. The operator $H_{L}(\mathbf{L})$ acts as a low-pass filter, transferring the low-pass components of the graph signal, while the operator $H_{H}(\mathbf{L})$ does the opposite, acting as a high-pass filter. The low-pass filter, $H_{L}(\mathbf{H})$, is followed by a downsampling operator which keeps only the graph signal values, $\mathbf{x}$, at the vertices $n \in \mathcal{E}$. Similarly, the high-pass filtering with the operator $H_{H}(\mathbf{L})$, is subsequently followed by a downsampling to the vertices $n \in \mathcal{H}$. These operations are crucial to alter the scale at which the graph signal is processed.

The synthesis side (right part in Figure 11.2), comprises the complementary upsampling and filtering operations, aiming to perform the graph signal reconstruction based on the upsampled versions, $\frac{1}{2}\left(\mathbf{I}+\mathbf{J}_{\mathcal{E}}\right) H_{L}(\mathbf{L}) \mathbf{x}$ and $\frac{1}{2}\left(\mathbf{I}+\mathbf{J}_{\mathcal{H}}\right) H_{H}(\mathbf{L}) \mathbf{x}$, of signals obtained on the filter bank analysis side. Therefore, upon performing the upsampling of these signals onto the original set of vertices, $\mathcal{V}$, by adding zeros to the complementary sets of vertices, filtering is performed by adequate low-pass, $G_{L}(\mathbf{L})$, and high-pass, $G_{H}(\mathbf{L})$, filters, to replace the zeros with meaningful values, as required for a successful reconstruction of the original signal. As in the classical wavelet analysis, to achieve the perfect (distortion-free) reconstruction it is necessary to conveniently design the analysis filters, $H_{L}(\mathbf{L})$ and $H_{H}(\mathbf{L})$, and the synthesis filters,
$G_{L}(\mathbf{L})$ and $G_{H}(\mathbf{L})$, as well as to determine adequate downsampling and upsampling operators.

It will be shown that the spectral folding phenomenon, described by Equations (3.9)-(3.10) in Part I, characterized by the specific spectral symmetry in the case of bipartite graphs, can be used to form the basis for the two-channel filter bank framework discussed in this section.

Consider a graph signal, $\mathbf{x}$, and the filter-bank as in Figure 11.2. If the graph signal, $\mathbf{x}$, passes through a low-pass analysis filter, $H_{L}(\mathbf{L})$, the output signal is $H_{L}(\mathbf{L}) \mathbf{x}$. According to (11.1), the downsampled-upsampled form of the output signal, $H_{L}(\mathbf{L}) \mathbf{x}$, is given by $\frac{1}{2}\left(\mathbf{I}+\mathbf{J}_{\mathcal{E}}\right) H_{L}(\mathbf{L}) \mathbf{x}$. After the syntheses filter, $G_{L}(\mathbf{L})$, the graph signal output becomes

$$
\begin{equation*}
\mathbf{f}_{L}=\frac{1}{2} G_{L}(\mathbf{L})\left(\mathbf{I}+\mathbf{J}_{\mathcal{E}}\right) H_{L}(\mathbf{L}) \mathbf{x} \tag{11.2}
\end{equation*}
$$

The same holds for the high-pass part

$$
\begin{equation*}
\mathbf{f}_{H}=\frac{1}{2} G_{H}(\mathbf{L})\left(\mathbf{I}+\mathbf{J}_{\mathcal{H}}\right) H_{H}(\mathbf{L}) \mathbf{x} \tag{11.3}
\end{equation*}
$$

where $\mathbf{J}_{\mathcal{H}}=-\mathbf{J}_{\mathcal{E}}=\operatorname{diag}\left((-1)^{1-\beta_{\mathcal{E}}(n)}\right)$ and

$$
\begin{equation*}
\mathbf{J}_{\mathcal{H}}+\mathbf{J}_{\mathcal{E}}=\mathbf{0} \tag{11.4}
\end{equation*}
$$

The overall output is a sum of these two signals, as illustrated in Figure 11.2, which after rearranging of terms gives

$$
\begin{align*}
\mathbf{y}= & \mathbf{f}_{L}+\mathbf{f}_{H}=\frac{1}{2}\left(G_{L}(\mathbf{L}) H_{L}(\mathbf{L})+G_{H}(\mathbf{L}) H_{H}(\mathbf{L})\right) \mathbf{x} \\
& +\frac{1}{2}\left(G_{L}(\mathbf{L}) \mathbf{J}_{\mathcal{E}} H_{L}(\mathbf{L})+G_{H}(\mathbf{L}) \mathbf{J}_{\mathcal{H}} H_{H}(\mathbf{L})\right) \mathbf{x} \tag{11.5}
\end{align*}
$$

The perfect reconstruction condition, $\mathbf{y}=\mathbf{x}$, is then achieved if

$$
\begin{gather*}
G_{L}(\mathbf{L}) H_{L}(\mathbf{L})+G_{H}(\mathbf{L}) H_{H}(\mathbf{L})=2 \mathbf{I}  \tag{11.6}\\
G_{L}(\mathbf{L}) \mathbf{J}_{\mathcal{E}} H_{L}(\mathbf{L})-G_{H}(\mathbf{L}) \mathbf{J}_{\mathcal{E}} H_{H}(\mathbf{L})=\mathbf{0} \tag{11.7}
\end{gather*}
$$

Spectral solution. For the spectral representation of the filter-bank signals in the domain of Laplacian basis functions, we will use the decomposition of the graph Laplacian in the form

$$
\begin{equation*}
\mathbf{F}=\mathbf{U}^{T} \mathbf{f}=\frac{1}{2}\left(\mathbf{U}^{T} \mathbf{x}+\mathbf{U}^{T} \mathbf{J}_{\mathcal{E}} \mathbf{x}\right)=\frac{1}{2}\left(\mathbf{X}+\mathbf{X}^{(a l i a s)}\right) \tag{11.8}
\end{equation*}
$$

where $\mathbf{X}^{(\text {alias })}=\mathbf{U}^{T} \mathbf{J}_{\mathcal{E}} \mathbf{x}$ is the aliasing spectral component.

In the case of bipartite graphs, the matrix operator $\mathbf{U}^{T} \mathbf{J}_{\mathcal{E}}$ produces the transformation matrix $\mathbf{U}^{T}$ with reversed (left-right flipped) order of eigenvectors. This is obvious from (3.10) in Part I, since

$$
\begin{aligned}
\mathbf{U}^{T} \mathbf{J}_{\mathcal{E}} & =\left[\begin{array}{llll}
\mathbf{u}_{0} & \mathbf{u}_{1} & \ldots & \mathbf{u}_{N-1}
\end{array}\right]^{T} \mathbf{J}_{\mathcal{E}} \\
& =\left[\begin{array}{cccc}
\mathbf{u}_{0 \mathcal{E}} & \mathbf{u}_{1 \mathcal{E}} & & \mathbf{u}_{N-1 \mathcal{E}} \\
-\mathbf{u}_{0 \mathcal{H}} & -\mathbf{u}_{1 \mathcal{H}} & \cdots & -\mathbf{u}_{N-1 \mathcal{H}}
\end{array}\right]^{T} \\
& =\left[\begin{array}{llll}
\mathbf{u}_{N-1} & \mathbf{u}_{N-2} & \ldots & \mathbf{u}_{0}
\end{array}\right]^{T}=\mathbf{U}_{\mathrm{LR}}^{T}
\end{aligned}
$$

where

$$
\mathbf{u}_{k}=\left[\begin{array}{c}
\mathbf{u}_{k \mathcal{E}} \\
\mathbf{u}_{k \mathcal{H}}
\end{array}\right], \quad \mathbf{u}_{N-1-k}=\left[\begin{array}{c}
\mathbf{u}_{k \mathcal{E}} \\
-\mathbf{u}_{k \mathcal{H}}
\end{array}\right], \quad k=0,1, \ldots N-1
$$

and

$$
\mathbf{U}_{\mathrm{LR}}=\left[\begin{array}{llll}
\mathbf{u}_{N-1} & \mathbf{u}_{N-2} & \ldots & \mathbf{u}_{0}
\end{array}\right]
$$

is a left-right flipped version of the eigenvector matrix

$$
\mathbf{U}=\left[\begin{array}{llll}
\mathbf{u}_{0} & \mathbf{u}_{1} & \ldots & \mathbf{u}_{N-1}
\end{array}\right]
$$

The element-wise form of Equation (11.8) is given by

$$
F(k)=\frac{1}{2}(X(k)+X(N-1-k))
$$

For bipartite graphs and the normalized graph Laplacian, we can write

$$
F\left(\lambda_{k}\right)=\frac{1}{2}\left(X\left(\lambda_{k}\right)+X\left(2-\lambda_{k}\right)\right)
$$

The second term in $F\left(\lambda_{k}\right)$ represents an aliasing component of the GFT of the original signal.

The spectral representation of (11.6) is obtained with a left-multiplication by $\mathbf{U}^{T}$ and a right-multiplication by $\mathbf{U}$,

$$
\mathbf{U}^{T} G_{L}(\mathbf{L}) \mathbf{U} \mathbf{U}^{T} H_{L}(\mathbf{L}) \mathbf{U}+\mathbf{U}^{T} G_{H}(\mathbf{L}) \mathbf{U} \mathbf{U}^{T} H_{H}(\mathbf{L}) \mathbf{U}=2 \mathbf{I}
$$

having in mind that we can add $\mathbf{U}^{T} \mathbf{U}=\mathbf{U} \mathbf{U}^{T}=\mathbf{I}$ between $G_{L}(\mathbf{L})$ and $H_{L}(\mathbf{L})$, and between $G_{H}(\mathbf{L})$ and $H_{H}(\mathbf{L})$. Using the spectral domain definition of the transfer functions, $\mathbf{U}^{T} H_{L}(\mathbf{L}) \mathbf{U}=H_{L}(\boldsymbol{\Lambda})$, we obtain
the spectral domain form of the reconstruction condition (11.6) as

$$
\begin{equation*}
G_{L}(\boldsymbol{\Lambda}) H_{L}(\boldsymbol{\Lambda})+G_{H}(\boldsymbol{\Lambda}) H_{H}(\boldsymbol{\Lambda})=2 \mathbf{I} \tag{11.9}
\end{equation*}
$$

For the aliasing part in Equation (11.7), the left-multiplication is performed by $\mathbf{U}^{T}$, while the right-multiplication is done by $\mathbf{U}_{\mathrm{LR}}^{T}$. The first term in (11.7) is then of the form

$$
\begin{align*}
\mathbf{U}^{T} G_{L}(\mathbf{L}) \mathbf{U U}^{T} \mathbf{J}_{\mathcal{E}} H_{L}(\mathbf{L}) \mathbf{U}_{\mathrm{LR}} & =\mathbf{U}^{T} G_{L}(\mathbf{L}) \mathbf{\mathbf { U U } _ { \mathrm { LR } }} H_{L}(\mathbf{L}) \mathbf{U}_{\mathrm{LR}} \\
& =G_{L}(\boldsymbol{\Lambda}) H_{L}^{(R)}(\boldsymbol{\Lambda}), \tag{11.10}
\end{align*}
$$

since $\mathbf{U}^{T} \mathbf{J}_{\mathcal{E}}=\mathbf{U}_{\mathrm{LR}}^{T}$ and $\mathbf{U}_{\mathrm{LR}}^{T} \mathbf{U}_{\mathrm{LR}}=\mathbf{I}$. The term

$$
H_{L}^{(R)}(\boldsymbol{\Lambda})=\mathbf{U}_{\mathrm{LR}}^{T} H_{L}(\mathbf{L}) \mathbf{U}_{\mathrm{LR}}=H_{L}(2 \mathbf{I}-\boldsymbol{\Lambda})
$$

is just a reversed order version of the diagonal matrix $H_{L}(\boldsymbol{\Lambda})$, with diagonal elements $H_{L}\left(\lambda_{N-1-k}\right)=H_{L}\left(2-\lambda_{k}\right)$ instead of $H_{L}\left(\lambda_{k}\right)$.

The same holds for the second term in (11.7) which is equal to $G_{H}(\mathbf{L}) \mathbf{J}_{\mathcal{H}} H_{H}(\mathbf{L})$, yielding the final spectral form of the aliasing condition in (11.7) as

$$
\begin{equation*}
G_{L}(\boldsymbol{\Lambda}) H_{L}(2 \mathbf{I}-\boldsymbol{\Lambda})-G_{H}(\boldsymbol{\Lambda}) H_{H}(2 \mathbf{I}-\boldsymbol{\Lambda})=\mathbf{0} \tag{11.11}
\end{equation*}
$$

An element-wise solution to the system in (11.6)-(11.7), for bipartite graphs and the normalized graph Laplacian, according to (11.9) and (11.11), reduces to

$$
\begin{array}{r}
G_{L}\left(\lambda_{k}\right) H_{L}\left(\lambda_{k}\right)+G_{H}\left(\lambda_{k}\right) H_{H}\left(\lambda_{k}\right)=2, \\
G_{L}\left(\lambda_{k}\right) H_{L}\left(2-\lambda_{k}\right)-G_{H}\left(\lambda_{k}\right) H_{H}\left(2-\lambda_{k}\right)=0 . \tag{11.13}
\end{array}
$$

Remark 62: A quadratic mirror filter solution would be such that for the designed transfer function of the low-pass analysis filter, $H_{L}(\lambda)$, the other filters are

$$
\begin{gather*}
G_{L}(\lambda)=H_{L}(\lambda), \\
H_{H}(\lambda)=H_{L}(2-\lambda), \\
G_{H}(\lambda)=H_{H}(\lambda)=H_{L}(2-\lambda) . \tag{11.14}
\end{gather*}
$$

For this solution, the design equation is given by

$$
\begin{equation*}
H_{L}^{2}(\lambda)+H_{L}^{2}(2-\lambda)=2, \tag{11.15}
\end{equation*}
$$

while the aliasing cancellation condition, (11.13), is always satisfied.

An example of such a system would be an ideal low-pass filter, defined by $H_{L}(\lambda)=\sqrt{2}$ for $\lambda<1$ and $H_{L}(\lambda)=0$ elsewhere. Since $H_{H}(\lambda)=H_{L}(2-\lambda)$ holds for systems on bipartite graphs, this satisfies the reconstruction condition. For the vertex domain realization, an approximation of the ideal filter with a finite neighborhood filtering relation would be required.
Example 51: Consider a simple form of the low-pass system

$$
H_{L}^{2}(\lambda)=2-\lambda,
$$

which satisfies the design equation, $H_{L}^{2}(\lambda)+H_{L}^{2}(2-\lambda)=2$. It also satisfies the condition that its form is of low-pass type for the normalized Laplacian of bipartite graphs, $H_{L}^{2}\left(\lambda_{0}\right)=2-\lambda_{0}=2$, since $\lambda_{0}=0$, and $H_{L}^{2}\left(\lambda_{\max }\right)=2-\lambda_{\max }=0$, as $\lambda_{\max }=2$. The vertex domain system operators which satisfy all four quadratic mirror analysis and synthesis filters in (11.14), are

$$
\begin{gathered}
H_{L}(\boldsymbol{\Lambda})=\sqrt{\mathbf{2 I}-\boldsymbol{\Lambda}}, \quad G_{L}(\boldsymbol{\Lambda})=H_{L}(\boldsymbol{\Lambda})=\sqrt{\mathbf{2 I}-\boldsymbol{\Lambda}}, \\
H_{H}(\boldsymbol{\Lambda})=H_{L}(\mathbf{2 I}-\boldsymbol{\Lambda})=\sqrt{\boldsymbol{\Lambda}}, \quad G_{H}(\boldsymbol{\Lambda})=H_{H}(\boldsymbol{\Lambda})=\sqrt{\boldsymbol{\Lambda}} .
\end{gathered}
$$

The spectral domain filtering form for the low-pass part of graph signal is then obtained from (11.2), as

$$
\begin{aligned}
\mathbf{F}_{L} & =\mathbf{U}^{T} \mathbf{f}_{L}=\frac{1}{2} \mathbf{U}^{T} G_{L}(\mathbf{L})\left(\mathbf{I}+\mathbf{J}_{\mathcal{E}}\right) H_{L}(\mathbf{L}) \mathbf{x} \\
& =\frac{1}{2} \mathbf{U}^{T} G_{L}(\mathbf{L}) \mathbf{U} \mathbf{U}^{T}\left(\mathbf{I}+\mathbf{J}_{\mathcal{E}}\right) H_{L}(\mathbf{L}) \mathbf{U}_{\mathrm{LR}} \mathbf{U}_{\mathrm{LR}}^{T} \mathbf{U X} \\
& =\frac{1}{2} G_{L}(\boldsymbol{\Lambda}) H_{L}(\boldsymbol{\Lambda}) \mathbf{X}+\frac{1}{2} G_{L}(\boldsymbol{\Lambda}) H_{L}(2 \mathbf{I}-\boldsymbol{\Lambda}) \mathbf{X}_{\mathrm{UD}}
\end{aligned}
$$

since $\mathbf{U}^{T} \mathbf{U}=\mathbf{I}, \mathbf{U}_{\mathrm{LR}}^{T} \mathbf{U}_{\mathrm{LR}}=\mathbf{I}, \mathbf{U}^{T} \mathbf{J}_{\mathcal{E}}=\mathbf{U}_{\mathrm{LR}}^{T}, \mathbf{U}_{\mathrm{LR}}^{T} \mathbf{U}=\mathbf{I}_{\mathrm{LR}}$, and $\mathbf{I}_{\mathrm{LR}} \mathbf{X}=\mathbf{X}_{\mathrm{UD}}$, where $\mathbf{I}_{\mathrm{LR}}$ is an anti-diagonal (backward) identity matrix, and $\mathbf{X}_{\mathrm{UD}}$ is the GFT vector, $\mathbf{X}$, with elements flipped upside-down.

The same holds for the high-pass part in (11.3), to yield

$$
\begin{aligned}
\mathbf{F}_{H} & =\frac{1}{2} \mathbf{U}^{T} G_{H}(\mathbf{L})\left(\mathbf{I}+\mathbf{J}_{\mathcal{H}}\right) H_{H}(\mathbf{L}) \mathbf{x} \\
& =\frac{1}{2} G_{H}(\boldsymbol{\Lambda}) H_{H}(\boldsymbol{\Lambda}) \mathbf{X}-\frac{1}{2} G_{H}(\boldsymbol{\Lambda}) H_{H}(2 \mathbf{I}-\boldsymbol{\Lambda}) \mathbf{X}_{\mathrm{UD}}
\end{aligned}
$$

and

$$
\mathbf{F}_{L}+\mathbf{F}_{H}=\mathbf{X}
$$

$$
\mathbf{L}=\begin{gathered}
0 \\
2 \\
4 \\
6 \\
6 \\
10 \\
12 \\
14 \\
3 \\
5 \\
7 \\
7 \\
9 \\
11 \\
11 \\
13 \\
15
\end{gathered}\left[\begin{array}{rrrrrrrrrrrrrrrr}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
-1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{array}\right]
$$

Therefore, after the one-step filter-bank based decomposition on a bipartite graph, we have a new low-pass signal, $\mathbf{f}_{L}$, for which the nonzero values are at the vertices in $\mathcal{E}$, and a high-pass signal, $\mathbf{f}_{H}$, with nonzero values only on $\mathcal{H}$. Note that the high-pass operator on the graph signal is the graph Laplacian, $\mathbf{L}$, while the low-pass operator is $\mathbf{2 I}-\mathbf{L}$, which easily reduces to $|\mathbf{L}|$, for the normalized graph Laplacian used here.

Another simple transfer function that satisfies the design equation (11.15) is $H_{L}(\lambda)=\sqrt{2} \cos (\pi \lambda / 4)$. A similar analysis can also be done for this transfer function and other functions defined by (11.14).

The considered transfer functions $H_{L}(\lambda)=\sqrt{2-\lambda}$ and $H_{L}(\lambda)=$ $\sqrt{2} \cos (\pi \lambda / 4)$ have several disadvantages, the most important being that they are not sufficiently smooth in the spectral domain at the boundary interval points (Stanković, 2015). In addition, although the graph Laplacian, $\mathbf{L}$, is commonly sparse (with a small number of nonzero elements in large graphs), the transfer function form $H_{L}(\mathbf{L})=\sqrt{\mathbf{2 I}-\mathbf{L}}$
is not sparse. This is the reason to use other forms which are sufficiently smooth toward the boundary points, along with their polynomial approximations, $H_{L}(\boldsymbol{\Lambda})=c_{0} \boldsymbol{\Lambda}+c_{1} \boldsymbol{\Lambda}^{2}+\cdots+c_{M-1} \boldsymbol{\Lambda}^{M-1}$, with the coefficients $c_{0}, c_{1}, \ldots, c_{M-1}$, that approximate $H_{L}(\lambda)$ and $H_{H}(\lambda)=H_{L}(2-\lambda)$ for each $\lambda=\lambda_{k}, k=0,1, \ldots, N-1$. This topic will be addressed in detail on a general form of graphs in Section 14.

The classic time-domain Haar wavelet (and scale) functions are easily obtained for a bipartite graph, such that $\mathcal{E}=0,2,4, \ldots, N-2$ and $\mathcal{H}=1,3,5, \ldots, N-1$, with the adjacency/weighting matrix defined by the elements $A_{m n}=1$, for $(m, n) \in\{(0,1),(2,3), \ldots,(N-2, N-1)\}$, as shown in Figure 11.3(a). This adjacency matrix has the block form as in Equation (2.19), Part I. The corresponding graph Laplacian is given in (11.16). Its eigenvectors are equal to the wavelet transform functions. The bipartite graph for the four-vertex resolution level in the Haar wavelet transform is shown in Figure 11.3(b).

Synthesis operators, comprised of more general interpolation methods, may be found in Li et al. (2019a).

(b)

Figure 11.3: Bipartite graph for the Haar wavelet transform with $N=16$ vertices. (a) Vertices in yellow are used for the low-pass part of the signal and correspond to the set $\mathcal{E}$, while the vertices in gray belong to the set $\mathcal{H}$. This is the highest two-vertex resolution level for the Haar wavelet. (b) Graph for a four-vertex resolution level in the Haar wavelet.

## 12

## Time-Varying Signals on Graphs

We shall denote a time-varying signal by $x_{p}(n)$, where $n$ designates the vertex index and $p$ the discrete-time index. For uniform sampling in time, the index $p$ corresponds to the time instant $p \Delta t$, where $\Delta t$ is the sampling interval. In general, this type of data can be considered within the graph Cartesian product framework (given in Property $M_{15}$, Section 2.3, Part I). The resulting graph $\mathcal{G}=(\mathcal{V}, \mathcal{B})$ follows as a Cartesian product of the given graph $\mathcal{G}_{1}=\left(\mathcal{V}_{1}, \mathcal{B}_{1}\right)$ and a simple path (or circular) graph $\mathcal{G}_{2}=\left(\mathcal{V}_{2}, \mathcal{B}_{2}\right)$ that corresponds to the classical uniformly sampled time-domain axis.

Example 52: A graph topology for a time varying signal on a graph is shown in Part I, Figure 2.9, where the graph vertices are designated by $1,2,3,4,5$ and time instants are denoted as the $a, b, c$ vertices on the path graph. The resulting Cartesian product graph, for the analysis of this kind of signals, is shown in Part I, Figure 2.9.

The adjacency matrix of a Cartesian product of two graphs is then given by

$$
\mathbf{A}=\mathbf{A}_{1} \otimes \mathbf{I}_{N_{2}}+\mathbf{I}_{N_{1}} \otimes \mathbf{A}_{2}=\mathbf{A}_{1} \oplus \mathbf{A}_{2},
$$

where $\mathbf{A}_{1}$ is the adjacency matrix of the graph of interest $\mathcal{G}_{1}$, and $\mathbf{A}_{2}$ is the adjacency matrix for the path or circular graph, $\mathcal{G}_{2}$, which designates
the sampling grid, while $N_{1}$ and $N_{2}$ denote, respectively, the number of vertices in $\mathcal{G}_{1}$ and $\mathcal{G}_{2}$.

We will next consider a simple and important example of a timevarying signal defined on graph in an iterative way, which designates the diffusion process on a graph in time.

### 12.1 Diffusion on Graph and Low Pass Filtering

Consider the diffusion equation

$$
\partial \mathbf{x} / \partial t=-\alpha \mathbf{L} \mathbf{x}
$$

Its discrete-time form, at a time instant $p$, may be obtained by using the backward difference approximation of the partial derivative ( $\partial \mathbf{x} / \partial t \sim$ $\mathbf{x}_{p+1}-\mathbf{x}_{p}$ ), and has the form

$$
\mathbf{x}_{p+1}-\mathbf{x}_{p}=-\alpha \mathbf{L} \mathbf{x}_{p+1}
$$

or $\mathbf{x}_{p+1}(\mathbf{I}+\alpha \mathbf{L})=\mathbf{x}_{p}$ to produce

$$
\mathbf{x}_{p+1}=(\mathbf{I}+\alpha \mathbf{L})^{-1} \mathbf{x}_{p} .
$$

On the other hand, the forward difference approximation ( $\partial \mathbf{x} / \partial t \sim$ $\left.\mathbf{x}_{p}-\mathbf{x}_{p-1}\right)$ to the diffusion equation yields

$$
\mathbf{x}_{p+1}-\mathbf{x}_{p}=-\alpha \mathbf{L} \mathbf{x}_{p}
$$

or

$$
\mathbf{x}_{p+1}=(\mathbf{I}-\alpha \mathbf{L}) \mathbf{x}_{p} .
$$

It is interesting to note that these iterative forms lead to the minimization of the quadratic form of a graph signal, $E_{x}=\mathbf{x} \mathbf{L x}{ }^{T}$, (see Section 4.2, Part I). The minimum of this quadratic form can be found based on the steepest descent method, whereby the signal value at a time instant $p$ is moving in the direction opposite to the gradient, toward the energy minimum position, with a step $\alpha$. The gradient of the quadratic form, $E_{x}=\mathbf{x} \mathbf{L x} \mathbf{x}^{T}$, is $\partial E_{x} / \partial \mathbf{x}^{T}=2 \mathbf{x} \mathbf{L}$, which results in an iterative procedure

$$
\begin{equation*}
\mathbf{x}_{p+1}=\mathbf{x}_{p}-\alpha \mathbf{L} \mathbf{x}_{p}=(\mathbf{I}-\alpha \mathbf{L}) \mathbf{x}_{p} . \tag{12.1}
\end{equation*}
$$

This relation can be used for simple and efficient filtering of graph signals (with the aim to minimize $E_{x}$ as a measure of signal smoothness). The spectral domain relation follows immediately, and has the form

$$
\mathbf{X}_{p+1}=(\mathbf{I}-\alpha \boldsymbol{\Lambda}) \mathbf{X}_{p}
$$

or for every individual component

$$
X_{p+1}(k)=\left(1-\alpha \lambda_{k}\right) X_{p}(k) .
$$

Recall that the eigenvalues, $\lambda_{k}$, represent the index of smoothness for a spectral vector (eigenvector), $\mathbf{u}_{k}$, with a small $\lambda_{k}$ indicating smooth slow-varying elements of the eigenvectors; therefore, for low-pass filtering we should retain the slow-varying eigenvectors in a spectral representation of the graph signal. Obviously, these slow-varying components will pass through this system since $\left(1-\alpha \lambda_{k}\right)$ is close to 1 for small $\lambda_{k}$, while the fast-varying components with a larger $\lambda_{k}$, are attenuated. This iterative procedure will converge if $\left|1-\alpha \lambda_{\max }\right|<1$.

In a stationary state of a diffusion process, the trivial minimal energy solution is obtained when

$$
\lim _{p \rightarrow \infty} X_{p+1}(k)=\left(1-\alpha \lambda_{k}\right)^{p+1} X_{0}(k)
$$

that is, all components $X_{p+1}(k)$ tend to 0 , except for the constant component, $X_{p+1}(0)$, for which $\lambda_{0}=0$. This component therefore defines the stationary state (maximally smooth solution). In order to avoid this effect in the processing of data on graphs, and to retain several low-pass components (eigenvectors) in the signal, the iteration process in (12.1) can be used in alternation with

$$
\begin{equation*}
\mathbf{x}_{p+2}=(\mathbf{I}+\beta \mathbf{L}) \mathbf{x}_{p+1} . \tag{12.2}
\end{equation*}
$$

This is the basis for Taubin's $\alpha-\beta$ algorithm, presented next.

### 12.2 Taubin's $\alpha-\beta$ Algorithm

When the two iterative processes in (12.1) and (12.2) are used in a successive order, the resulting system on a graph is referred to as Taubin's $\alpha-\beta$ algorithm. This algorithm is widely used for low-pass
filtering of data on graphs, since it is very simple, and admits efficient implementation in the vertex domain.

Definition: Taubin's $\alpha-\beta$ algorithm is a two-step iterative algorithm for efficient low-pass data filtering on graphs. Its two steps are defined in a unified way as

$$
\begin{equation*}
\mathbf{x}_{p+2}=(\mathbf{I}+\beta \mathbf{L})(\mathbf{I}-\alpha \mathbf{L}) \mathbf{x}_{p} . \tag{12.3}
\end{equation*}
$$

The corresponding element-wise transfer function in the spectral domain of the two iteration steps in (12.3) is given by

$$
H\left(\lambda_{k}\right)=\left(1+\beta \lambda_{k}\right)\left(1-\alpha \lambda_{k}\right) .
$$

After $K$ iterations of this algorithm, the spectral domain transfer function can be written as

$$
\begin{equation*}
H_{K}\left(\lambda_{k}\right)=\left(\left(1+\beta \lambda_{k}\right)\left(1-\alpha \lambda_{k}\right)\right)^{K} . \tag{12.4}
\end{equation*}
$$

For some values of $\alpha<\beta$, this system can be a good and computationally very simple approximation of a graph low-pass filter.
Example 53: Consider the graph from Figure 9.4(a) and its graph Laplacian, $\mathbf{L}$. For the choice of parameters $\alpha=0.1798$ and $\beta=0.2193$, the spectral transfer function in (12.4) is shown in Figure 12.1 for the considered graph filter, and for the numbers of iterations in Taubin's algorithm $K=1,5,30$, and 150 . Observe how the transfer function, $H\left(\lambda_{k}\right)$, approaches the ideal low-pass form as the number of iterations, $K$, increases.

The task is next to low-pass filter the noisy signal from Figure 9.12(b). The initial noisy signal is denoted by $\mathbf{x}_{0}$. Then $\mathbf{x}_{1}=(\mathbf{I}-0.1545 \mathbf{L}) \mathbf{x}_{0}$ is calculated using the corresponding graph Laplacian, followed by obtaining $\mathbf{x}_{2}=(\mathbf{I}+0.1875 \mathbf{L}) \mathbf{x}_{1}$. In the third and fourth iteration, the signal values $\mathbf{x}_{3}=(\mathbf{I}-0.1545 \mathbf{L}) \mathbf{x}_{2}$ and $\mathbf{x}_{4}=(\mathbf{I}+0.1875 \mathbf{L}) \mathbf{x}_{3}$ are calculated. This two-step iteration cycle is repeated $K=20$ times. The resulting signal is the same as the output of an ideal low-pass filter shown in Figure 9.12(c).

Finally, the noisy signal from Figure 8.3 was filtered using Taubin's $\alpha-\beta$ algorithm, with $\alpha=0.15$ and $\beta=0.15$, over $K=100$ iterations, and the result is shown in Figure 12.2. Observe the reduced level of additive noise in the output.


Figure 12.1: Filter approximation in the spectral domain for a varying number of iterations, $K$, using Taubin's algorithm and the graph Laplacian matrix of the graph in Figure 9.4.


Figure 12.2: The noisy signal from Figure 8.3 was filtered using $K=100$ iterations of the Taubin two-step algorithm with $\alpha=0.15$ and $\beta=0.15$.

Processing of time-varying signals on graphs has been a topic of intensive research; for a deeper insight we refer the reader to Isufi et al. (2017), Grassi et al. (2017), and Gama et al. (2019).

## 13

## Random Graph Signal Processing

This section extends the concepts of data analytics for deterministic signals on graphs addressed so far, to introduce notions of random signals on graphs, their properties, and statistical graph-specific methods for their analysis. The main focus is on wide-sense stationary (WSS) data observed on graphs. In general, the stationarity of a signal is inherently related to the signal shift operator and its properties. We have already presented two approaches to define a shift on a graph (through the adjacency matrix and the graph Laplacian, and their spectral decompositions). These will be used, along with other general properties of WSS signals, to define the conditions for wide sense stationarity of random signals on graphs (Chepuri and Leus, 2016; Loukas and Perraudin, 2016; Marques et al., 2017; Perraudin and Vandergheynst, 2017; Puy et al., 2018; Zhang et al., 2015). However the main obstacle toward extending the classical statistical data analytics to graphs is that the shift on a graph typically does not preserve signal energy (isometry property), that is, $\|\mathbf{A x}\|_{2}^{2} \neq\|\mathbf{x}\|_{2}^{2}$.

For completeness, we first provide a short review of WSS definitions in classical signal processing, together with their properties.

### 13.1 Review of WSS and Related Properties for Random Signals in Standard Time Domain

Definition: A real-valued random signal, $x(n)$, is WSS in the standard time domain if its mean value is time-invariant, $\mu_{x}(n)=\mathrm{E}\{x(n)\}=\mu_{x}$, and its autocorrelation function is shift-invariant, that is, $r_{x}(n, n-m)=$ $\mathrm{E}\{x(n) x(n-m)\}=r_{x}(m)$.
Remark 63: A random WSS time-domain signal, $x(n)$, can be considered as an output of a linear shift invariant system with impulse response, $h(n)$, which is driven by a white noise input, $\varepsilon(n)$, with $r_{\varepsilon}(n, m)=\delta(n-m)$.
Remark 64: In classical time domain, the eigenvectors, $\mathbf{u}_{k}$, of the shift operator $y(n)=x(n-1)$, or in a matrix form $\mathbf{y}=\mathbf{A} \mathbf{x}$, are the DFT basis functions, with $\mathbf{A}=\mathbf{U} \boldsymbol{\Lambda} \mathbf{U}^{H}$. This property is discussed in detail and proven in Part I, Section 3.2, Equations (3.4)-(3.5).
Remark 65: For a random signal, its DFT $\mathbf{X}=\mathbf{U}^{H} \mathbf{x}$ is also a random signal with the power spectrum matrix $\mathbf{P}_{x}=\mathrm{E}\left\{\mathbf{X} \mathbf{X}^{H}\right\}$, where $\mathbf{U}^{H}$ is the DFT transformation matrix. For WSS signals, the matrix $\mathbf{P}_{x}$ is diagonal and has the power spectral density (PSD) as its diagonal values

$$
p_{x}(k)=\operatorname{DFT}\left\{r_{x}(n)\right\}=\mathrm{E}\left\{|X(k)|^{2}\right\} .
$$

Remark 66: For WSS random signals, their correlation matrix, $\mathbf{R}_{x}=$ $\mathrm{E}\left\{\mathbf{x x}^{T}\right\}$, is diagonalizable with the same transform matrix, $\mathbf{U}$, which defines the DFT, $\mathbf{X}^{\text {def }}=\mathbf{U}^{H} \mathbf{x}$, with $\mathbf{x} \stackrel{\text { def }}{=} \mathbf{U X}$. The proof follows from

$$
\begin{align*}
\mathbf{R}_{x} & =\mathrm{E}\left\{\mathbf{x} \mathbf{x}^{T}\right\}=\mathrm{E}\left\{\mathbf{U X}(\mathbf{U X})^{H}\right\} \\
& =\mathbf{U E}\left\{\mathbf{X X}^{H}\right\} \mathbf{U}^{H}=\mathbf{U P}_{x} \mathbf{U}^{H}, \tag{13.1}
\end{align*}
$$

and the fact that $\mathbf{P}_{x}$ is a diagonal matrix for WSS signals.
The properties of the WSS signals in classical analyses, presented in this subsection, will be used next to define the corresponding properties of random signals on undirected graphs.

### 13.2 Adjacency Matrix Based Definition of GWSS

Consider a real-valued white noise signal on a graph, $\varepsilon=\{\varepsilon(n)\}$. Following Remark 63, a signal $\mathbf{x}$ on the graph is graph wide-sense stationary (GWSS) if it can be considered as an output of a linear shift invariant system on a graph, $H(\mathbf{A})=\sum_{m=0}^{M-1} h_{m} \mathbf{A}^{m}$, which is driven by a white noise input, $\varepsilon$, that is

$$
\mathbf{x}=H(\mathbf{A}) \varepsilon .
$$

Remark 67: The autocorrelation matrix, $\mathbf{R}_{x}=\mathrm{E}\left\{\mathbf{x x}^{T}\right\}$, of a GWSS signal is diagonalizable with the eigenmatrix of the adjacency matrix, A, since ( $c f$. Remark 66)

$$
\begin{gather*}
\mathbf{A}=\mathbf{U} \boldsymbol{\Lambda} \mathbf{U}^{-1}=\mathbf{U} \boldsymbol{\Lambda} \mathbf{U}^{T} \\
\mathrm{E}\left\{\mathbf{x x}^{T}\right\}=\mathbf{U P}_{x} \mathbf{U}^{T}, \tag{13.2}
\end{gather*}
$$

where $\mathbf{P}_{x}$ is a diagonal matrix. The values on the diagonal of matrix $\mathbf{P}_{x}$ can be comprised into the vector $\mathbf{p}_{x}$, which represents the PSD of a graph signal, $\mathbf{x}, p_{x}(k)=\mathrm{E}\left\{|X(k)|^{2}\right\}$.

To prove this property for a signal $\mathbf{x}=H(\mathbf{A}) \varepsilon$, consider

$$
\mathbf{R}_{x}=\mathrm{E}\left\{\mathbf{x} \mathbf{x}^{T}\right\}=\mathrm{E}\left\{H(\mathbf{A}) \boldsymbol{\varepsilon}(H(\mathbf{A}) \boldsymbol{\varepsilon})^{T}\right\}=H(\mathbf{A}) H^{T}(\mathbf{A}),
$$

since $\mathrm{E}\left\{\varepsilon \varepsilon^{T}\right\}=\mathbf{I}$. Using $H(\mathbf{A})=\mathbf{U}^{T} H(\boldsymbol{\Lambda}) \mathbf{U}$, we obtain

$$
\mathbf{R}_{x}=\mathbf{U}^{T}|H(\boldsymbol{\Lambda})|^{2} \mathbf{U}
$$

which concludes the proof that the matrix $\mathbf{P}_{x}$ is diagonal

$$
\mathbf{P}_{x}=|H(\boldsymbol{\Lambda})|^{2},
$$

with the diagonal elements equal to the PSD of signal $\mathbf{x}$,

$$
p_{x}(k)=\left|H\left(\lambda_{k}\right)\right|^{2} .
$$

The periodogram of a graph signal can be estimated using $K$ realizations of the random signal, denoted by $\mathbf{x}_{i}$, and is equal to the diagonal elements of the matrix

$$
\hat{\mathbf{P}}_{x}=\frac{1}{K} \sum_{i=1}^{K} \mathbf{X}_{i} \mathbf{X}_{i}^{T}=\mathbf{U}^{T} \frac{1}{K} \sum_{i=1}^{K}\left(\mathbf{x}_{i} \mathbf{x}_{i}^{T}\right) \mathbf{U} .
$$

Consider a system on a graph, with a spectral domain transfer function $H(\boldsymbol{\Lambda})$. Assume that the input signal to this system is GWSS, with PSD $p_{x}(k)$. The PSD of the output graph signal, $y(n)$, is then given by

$$
p_{y}(k)=\left|H\left(\lambda_{k}\right)\right|^{2} p_{x}(k) .
$$

This expression is conformal with the output power of a standard linear system.

### 13.3 Wiener Filter on a Graph

Consider a real-valued graph signal, $\mathbf{s}$, which serves as an input to a linear shift-invariant system on an undirected graph, to yield a noisy output

$$
\mathbf{x}=\sum_{m=0}^{M-1} h_{m} \mathbf{A}^{m} \mathbf{s}+\boldsymbol{\varepsilon}
$$

In the spectral domain, this system is described by

$$
\mathbf{X}=H(\boldsymbol{\Lambda}) \mathbf{S}+\mathbf{E},
$$

where $\mathbf{E}$ is the GFT of the noise, $\boldsymbol{\varepsilon}$.
Assume that the signal and noise are statistically independent, and that the noise is a zero-mean GWSS random signal. The aim is to find the system function of the optimal filter, $G(\boldsymbol{\Lambda})$, such that its output $\mathbf{Y}=G(\boldsymbol{\Lambda}) \mathbf{X}$, estimates the GFT of the input, $\mathbf{S}$, in the least squares sense. This condition can be expressed as

$$
e^{2}=\mathrm{E}\left\{\|\mathbf{S}-\mathbf{Y}\|_{2}^{2}\right\}=\mathrm{E}\left\{\|\mathbf{S}-G(\boldsymbol{\Lambda}) \mathbf{X}\|_{2}^{2}\right\} .
$$

Upon setting the derivative of $e^{2}$ with respect to the elements of $G(\boldsymbol{\Lambda})$ to zero, we arrive at

$$
2 \mathrm{E}\left\{(\mathbf{S}-G(\boldsymbol{\Lambda}) \mathbf{X}) \mathbf{X}^{T}\right\}=\mathbf{0},
$$

which results in the system function of the graph Wiener filter in the form (using matrix division in a symbolic way)

$$
\begin{aligned}
G(\boldsymbol{\Lambda}) & =\frac{\mathrm{E}\left\{\mathbf{S X}^{T}\right\}}{\mathrm{E}\left\{\mathbf{X} \mathbf{X}^{T}\right\}}=\frac{\mathrm{E}\left\{\mathbf{S}(H(\boldsymbol{\Lambda}) \mathbf{S}+\mathbf{E})^{T}\right\}}{\mathrm{E}\left\{(H(\boldsymbol{\Lambda}) \mathbf{S}+\mathbf{E})(H(\boldsymbol{\Lambda}) \mathbf{S}+\mathbf{E})^{T}\right\}} \\
& =\frac{H(\boldsymbol{\Lambda}) \mathbf{P}_{s}}{H^{2}(\boldsymbol{\Lambda}) \mathbf{P}_{s}+\mathbf{P}_{\varepsilon}}
\end{aligned}
$$

or element-wise

$$
G\left(\lambda_{k}\right)=\frac{H\left(\lambda_{k}\right) p_{s}(k)}{H^{2}\left(\lambda_{k}\right) p_{s}(k)+E(k)} .
$$

When the noise is not present, the elements of the vector $\mathbf{E}$ are zerovalued, $E(k)=0$ for all $k$, and the graph inverse filter (introduced in Section 9.5 .1 ) directly follows.
Remark 68: The above expressions for the graph Wiener filter are conformal with the standard frequency domain Wiener filter, given by

$$
G(\omega)=\frac{P_{s}(\omega)}{P_{s}(\omega)+P_{\varepsilon}(\omega)},
$$

which again demonstrates the generic nature of Graph Data Analytics.

### 13.4 Spectral Domain Shift Based Definition of GWSS

Consider an $m$-step shift on a graph defined using the graph filter response

$$
\begin{equation*}
\mathcal{T}_{m}\{h(n)\}=h_{m}(n)=\sum_{k=0}^{N-1} H\left(\lambda_{k}\right) u_{k}(m) u_{k}(n) . \tag{13.3}
\end{equation*}
$$

The matrix form of this relation is given by

$$
\begin{equation*}
\mathcal{T}_{h}=H(\mathbf{L})=\sum_{m=0}^{M-1} h_{m} \mathbf{L}^{m}=\mathbf{U} H(\mathbf{\Lambda}) \mathbf{U}^{T}, \tag{13.4}
\end{equation*}
$$

where $\mathcal{T}_{m}\{h(n)\}$ are the elements of $\mathcal{T}_{h}$.
Note that the graph filter response function is well localized on a graph. Namely, if we use, for example, the ( $M-1$ )-neighborhood of a vertex $n$, within a filtering function of order $M$ defined by $H(\boldsymbol{\Lambda})$, then only the vertices within this neighborhood are used in the calculation of graph filter response. From (13.4), we see that the localization operator acts in the spectral domain and associates the corresponding shift to the vertex domain.
Definition: A random graph signal, $x(n)$, is GWSS if its autocorrelation function is invariant with respect to the shift, $\mathcal{T}_{m}\left\{r_{x}(n)\right\}$.

Similar to (13.2), the autocorrelation matrix, $\mathbf{R}_{x}$, of a GWSS signal is diagonalizable based on the matrix of eigenvectors of the graph Laplacian $\mathbf{L}$, whereby

$$
\begin{equation*}
\mathbf{L}=\mathbf{U} \boldsymbol{\Lambda} \mathbf{U}^{T} \tag{13.5}
\end{equation*}
$$

For the basic autocorrelation we use

$$
\mathbf{R}_{x}=\mathbf{U} P_{x}(\boldsymbol{\Lambda}) \mathbf{U}^{T}
$$

so that

$$
\mathcal{T}_{m}\left\{r_{x}(n)\right\}=\sum_{k=0}^{N-1} p_{x}\left(\lambda_{k}\right) u_{k}(m) u_{k}(n)
$$

where

$$
P_{x}(\boldsymbol{\Lambda})=\mathbf{U R}_{x} \mathbf{U}^{T}
$$

is a diagonal matrix.

### 13.5 Isometric Shift Operator

Another possible approach may be based on the shift operator defined as $\mathcal{T}_{h}=\exp (j \pi \sqrt{\mathbf{L} / \rho})$, where $\rho$ is the upper bound on the eigenvalues, $\rho=\max _{k}\left\{\lambda_{k}\right\}$ (Girault, 2015; Girault et al., 2015). Physically, this operator casts the eigenvalues of the Laplacian, $\mathbf{L}$, onto a unit circle, thus preserving in this way the isometry property, since

$$
\begin{equation*}
\mathcal{T}_{h}=\exp (j \pi \sqrt{\mathbf{L} / \rho})=\mathbf{U} \exp (j \pi \sqrt{\boldsymbol{\Lambda} / \rho}) \mathbf{U}^{T} . \tag{13.6}
\end{equation*}
$$

The property $f(\mathbf{L})=\mathbf{U} f(\boldsymbol{\Lambda}) \mathbf{U}^{H}$ was used above. Observe that for real-valued eigenvalues, $\lambda_{k}$, all eigenvalues of the matrix $\exp (j \pi \sqrt{\Lambda / \rho})$ reside on the unit circle, with the frequency $0 \leq \omega_{k}=\pi \sqrt{\lambda_{k} / \rho} \leq \pi$ being associated to the eigenvector $\mathbf{u}_{k}$.

## 14

## Vertex-Frequency Representations

Oftentimes in practical applications concerned with large graphs, we may not be interested in the analysis of the entire graph signal, but rather in its local behavior. Indeed, the Big Data paradigm has revealed the possibility of using smaller and localized subsets of the available information to enable reliable mathematical analysis and local characterization of subsets of data of interest (Sandryhaila and Moura, 2014a). Our aim in this section is to characterize the localized graph signal behavior simultaneously in the vertex-frequency domain, in a natural analogy with classical time-frequency analysis (Boashash, 2015; Cohen, 1995; Stanković et al., 2014). Indeed, the concept of vertex-frequency analysis was introduced in Shuman et al. (2012), by extending the principle of signal localization by applying localization window functions to signals defined on graphs. This concept was further developed in Shuman et al. (2016), with the extensions of this approach including the multi-window form Zheng et al. (2016), a short-graph Fourier transform combined with page-rank vectors (Tepper and Sapiro, 2016), or vertex domain localization windows (Stanković et al., 2017a). Window forms have also been adapted to define the frequency-varying localized graph Fourier transform (Cioacă et al., 2019) and spectral domain wavelet
transform-based vertex-frequency kernels, including the signal adaptive kernels with polynomial approximations and recursive realizations (Behjat and Van De Ville, 2019; Hammond et al., 2011, 2019).

It is important to note that, while the concept of window functions for signal localization has been extended to signals defined on graphs (Shuman et al., 2012, 2016; Stanković et al., 2017a; Tepper and Sapiro, 2016; Zheng et al., 2016), such extensions are not straightforward, since, owing to inherent properties of graphs as irregular but interconnected domains, even an operation which is very simple in classical time-domain analysis, like the time shift, cannot be straightforwardly generalized to graph signal domain. This has resulted in several approaches to the definition of the graph shift operator, and much ongoing research in this domain (Shuman et al., 2012, 2016; Stanković et al., 2017a; Tepper and Sapiro, 2016; Zheng et al., 2016).

A common approach to signal windowing in the graph domain is to utilize the eigenspectrum of a graph to obtain window function for each graph vertex (Shuman et al., 2013). Another possibility is to define the window support as a local neighborhood for each vertex (Stankovic et al., 2017a). In either case, the localization window is defined based on a set of vertices that contain the current vertex, $n$, and all vertices that are close in some sense to the vertex $n$, that is, a neighborhood of vertex $n$. In this monograph, special attention is devoted to the class of local graph Fourier transform approaches which can be implemented in the vertex domain, since this domain often offers a basis for numerically efficient analysis in the case of very large graphs.

Notice that, as in classical signal analysis, a localization window should be narrow enough so as to provide good localization of signal properties, but at the same time wide enough to produce high resolution in the spectral domain.

With vertex-frequency analysis serving as a key to graph signal estimation, filtering, and efficient representation, two forms of the local graph Fourier transform inversion are considered here, while the inversion condition is defined within the framework of frames, that is, based on the analysis of energy of the graph spectrogram. A relation between the graph wavelet transform and the local graph Fourier transform implementation and its inversion is also established.

Remark 69: The energy versions of the vertex-frequency representations are also considered, as these representations can be implemented without a localization window, and they can serve as estimators of the local smoothness index.

The reduced interference vertex-frequency distributions, which satisfy the marginal property and localize graph signal energy in the vertex-frequency domain are also defined, and are subsequently related to classical time-frequency analysis, as a special case.

Consider a graph with $N$ vertices, $n \in \mathcal{V}=\{0,1, \ldots, N-1\}$, which are connected with edges whose weights are $W_{m n}$. Spectral analysis of graphs is most commonly based on the eigendecomposition of the graph Laplacian, $\mathbf{L}$, or the adjacency matrix, A. By default, we shall assume the decomposition of the graph Laplacian, $\mathbf{L}$, if not stated otherwise.

### 14.1 Localized Graph Fourier Transform (LGFT)

The localized graph Fourier transform (LGFT), denoted by $S(m, k)$, can be considered as an extension of the standard time-localized (shorttime) Fourier transform (STFT), and can be calculated as the GFT of a signal, $x(n)$, multiplied by an appropriate vertex localization window function, $h_{m}(n)$, to yield

$$
\begin{equation*}
S(m, k)=\sum_{n=0}^{N-1} x(n) h_{m}(n) u_{k}(n) \tag{14.1}
\end{equation*}
$$

In general, it is desired that a graph window function, $h_{m}(n)$, should be such that it localizes the signal content around the vertex $m$. To this end, its values should be close to 1 at vertex $m$ and vertices in its close neighborhood, while it should approach to 0 for vertices that are far from vertex $m$. For an illustration of the concept of localization window on a graph see Figure 8.2, panels (a) and (c).

The localized GFT in (14.1) admits a matrix notation, $\mathbf{S}$, and contains all elements, $S(m, k), m=0,1, \ldots, N-1, k=0,1, \ldots, N-1$. The columns of $\mathbf{S}$ which correspond to a vertex $m$ are given by

$$
\mathbf{s}_{m}=\operatorname{GFT}\left\{x(n) h_{m}(n)\right\}=\mathbf{U}^{T} \mathbf{x}_{m},
$$

where $\mathbf{x}_{m}$ is the vector of which the elements, $x(n) h_{m}(n)$, are equal to the graph signal samples, $x(n)$, multiplied by the window function,
$h_{m}(n)$, centered at the vertex $m$, while matrix $\mathbf{U}$ is composed of the eigenvectors $\mathbf{u}_{k}$, with elements $u_{k}(n), k=0,1, \ldots, N-1$, of the graph Laplacian as its columns.

## Special cases:

- For $h_{m}(n)=1$, the localized vertex spectrum is equal to the standard spectrum, $S(m, k)=X(k)$, in (14.1) for each $m$; this means that no vertex localization is performed.
- If $h_{m}(m)=1$ and $h_{m}(n)=0$ for $n \neq m$, the localized vertex spectrum is equal to the graph signal, $S(m, 0)=x(m) / \sqrt{N}$, for $k=0$.

In the following, we outline ways to create vertex domain windows with desirable localization characteristics, and address two methods for defining graph localization window functions, $h_{m}(n)$ :

- Spectral domain definition of windows, $h_{m}(n)$, which are defined using their spectral basic function. The spectral domain definition of the window is shown to be related to the wavelet transform.
- Vertex domain window definitions, with one method bearing a direct relation to the spectral analysis of the graph window, while the other method represents a purely vertex domain formulation.


## Windows Defined in the GFT Domain

The basic function of a window, $h(n)$, can be conveniently defined in the spectral domain, for example, in the form

$$
\begin{equation*}
H(k)=C \exp \left(-\lambda_{k} \tau\right), \tag{14.2}
\end{equation*}
$$

where $C$ denotes the "window amplitude" and $\tau>0$ is a constant which determines the window width in the spectral domain. Notice that the graph shifted and "modulated" versions of this window are straightforwardly obtained using the generalized convolution of graph signals, defined in Section 9.9. The graph-shifted window in the vertex domain is then defined by the IGFT of $H(k) u_{k}(m)$, to give the window
localized at the vertex $m$, denoted by $h_{m}(n)$, as in (9.46), in the form

$$
\begin{equation*}
h_{m}(n)=h(n) * \delta_{m}(n)=\sum_{k=0}^{N-1} H(k) u_{k}(m) u_{k}(n) \tag{14.3}
\end{equation*}
$$

An example of two windows obtained in this way is given in Figures 8.2(a), (b).

Observe that the exponential function in (14.2) corresponds to a Gaussian window in classical analysis (thus offering the best timefrequency concentration (Boashash, 2015; Cohen, 1995; Stanković et al., 2014)), since graph signal processing on a path graph reduces to classical signal analysis. In this case, the eigenvalues of the graph Laplacian, $\lambda$, may be related to the frequency, $\omega$, in classical signal analysis as $\lambda \sim \omega^{2}$. Properties of graph window functions. The graph window which is localized at the vertex $m$, and defined by (14.3), satisfies the following properties:
$W_{1}$ : Symmetry, $h_{m}(n)=h_{n}(m)$, which follows from the definition in (14.3).
$W_{2}$ : A sum of all coefficients of a localized window, $h_{m}(n)$, is equal to $H(0)$, since

$$
\begin{aligned}
\sum_{n=0}^{N-1} h_{m}(n) & =\sum_{k=0}^{N-1} H(k) u_{k}(m) \sum_{n=0}^{N-1} u_{k}(n) \\
& =\sum_{k=0}^{N-1} H(k) u_{k}(m) \delta(k) \sqrt{N}=H(0)
\end{aligned}
$$

with $\sum_{n=0}^{N-1} u_{k}(n)=\delta(k) \sqrt{N}$, following from the definition of the eigenvectors, $u_{k}(n)$.
$W_{3}$ : The Parseval theorem for $h_{m}(n)$ has the form

$$
\begin{equation*}
\sum_{n=0}^{N-1}\left|h_{m}(n)\right|^{2}=\sum_{k=0}^{N-1}\left|H(k) u_{k}(m)\right|^{2} \tag{14.4}
\end{equation*}
$$

These properties will be used in the sequel in the inversion analysis of the LGFT.

Based on the above properties, the LGFT can now be written as

$$
\begin{align*}
S(m, k) & =\sum_{n=0}^{N-1} x(n) h_{m}(n) u_{k}(n)  \tag{14.5}\\
& =\sum_{n=0}^{N-1} \sum_{p=0}^{N-1} x(n) H(p) u_{p}(m) u_{p}(n) u_{k}(n) \tag{14.6}
\end{align*}
$$

The modulated (frequency shifted) version of the window centered at a vertex $m$ and for a spectral index $k$ will be referred to as the vertex-frequency kernel, $\mathcal{H}_{m, k}(n)$, which is defined as

$$
\begin{equation*}
\mathcal{H}_{m, k}(n)=h_{m}(n) u_{k}(n)=\left(\sum_{p=0}^{N-1} H(p) u_{p}(m) u_{p}(n)\right) u_{k}(n) . \tag{14.7}
\end{equation*}
$$

Using the kernel notation, it becomes obvious that the LGFT in (14.6), for a given vertex $m$ and a spectral index $k$, physically represents a projection of a graph signal, $x(n)$, onto the graph kernel, $\mathcal{H}_{m, k}(n)$, that is,

$$
\begin{equation*}
S(m, k)=\left\langle\mathcal{H}_{m, k}(n), x(n)\right\rangle=\sum_{n=0}^{N-1} \mathcal{H}_{m, k}(n) x(n) \tag{14.8}
\end{equation*}
$$

Remark 70: The classical STFT, a basic tool in time-frequency analysis, can be obtained as a special case of the GFT when the graph is directed and circular. For this type of graph, the eigendecomposition of the adjacency matrix produces complex-valued eigenvectors of the form $u_{k}(n) \sqrt{N}=\exp (j 2 \pi n k / N)$. Then, having in mind the complex nature of these eigenvectors,

$$
\begin{equation*}
S(m, k)=\sum_{n=0}^{N-1} \sum_{p=0}^{N-1} x(n) H(p) u_{p}^{*}(m) u_{p}(n) u_{k}^{*}(n), \tag{14.9}
\end{equation*}
$$

the value of $S(m, k)$ in (14.5) becomes the standard STFT, that is

$$
\begin{align*}
S(m, k) & =\frac{1}{N^{3 / 2}} \sum_{n=0}^{N-1} \sum_{p=0}^{N-1} x(n) H(p) e^{-j \frac{2 \pi}{N} m p} e^{j \frac{2 \pi}{N} n p} e^{-j \frac{2 \pi}{N} n k} \\
& =\frac{1}{N} \sum_{n=0}^{N-1} x(n) h(n-m) e^{-j 2 \pi n k / N} \tag{14.10}
\end{align*}
$$

where $h(n)$ is the inverse DFT of $H(k)$.

Example 54: To illustrate the principle of local vertex-frequency representations, consider the graph and the graph signal from Figure 8.1. A graph with $N=100$ vertices, randomly placed on the so called Swiss roll surface, is shown in Figure 8.1(a). The vertices are connected with edges whose weights are defined as $W_{m n}=\exp \left(-r_{m n}^{2} / \alpha\right)$, where $r_{m n}$ is the distance between the vertices $m$ and $n$, measured along the Swiss roll manifold, and $\alpha$ is a constant. Small weight values were hard-thresholded


Figure 14.1: Continued.


Figure 8.1: Concept of a signal on a graph. (a) Vertices on a three-dimensional manifold Swiss roll surface. (b) A graph representation on the Swiss roll manifold. (c) Two-dimensional presentation of the three-dimensional graph from (b), with vertex colors defined by the three smoothest graph Laplacian eigenvectors $u_{1}(n), u_{2}(n)$, and $u_{3}(n)$. (d) A signal observed on the graph in (c), which is composed of three Laplacian eigenvectors (signal components). The supports of these three components are designated by different vertex colors. The vertex-frequency representations are then assessed based on their ability to accurately resolve and localize these three graph signal components.
to zero, in order to reduce the number of edges associated with each vertex to only a few strongest ones. The so produced graph is shown in Figure 8.1(b), and its two-dimensional presentation in Figure 8.1(c). Vertices are ordered so that the values of the Fiedler eigenvector, $u_{1}(n)$, are nondecreasing. More detail about the Swiss role graph are given in Part III.

A signal on this graph was created so as to be composed of parts of three Laplacian eigenvectors. For the subset, $\mathcal{V}_{1}$, of all vertices, $\mathcal{V}$, which comprises the vertices with indices from $m=0$ to $m=29$, the eigenvector with the spectral index $k=8$ was used. For the subset, $\mathcal{V}_{2}$, with the vertex indices from $m=30$ to $m=59$, the signal was equal to the eigenvector $u_{66}(n)$, that is, with $k=66$. The remaining vertices form the vertex subset $\mathcal{V}_{3}$, and the signal on this subset was equal to the eigenvector with the spectral index $k=27$. The amplitudes of these eigenvectors were scaled too.

Consider now the vertex-frequency localization kernels,

$$
\mathcal{H}_{m, k}(n)=h_{m}(n) u_{k}(n)
$$

shown in Figure 8.2. The constant eigenvector, $u_{0}(n)=1 / \sqrt{N}$, was used in the panel shown in Figure 8.2(a) at $m=34$. In this case, the localization window, $h_{34}(n)$, is shown since $\mathcal{H}_{34,0}(n)=h_{34}(n) / \sqrt{N}$. The illustration is repeated in the panel in Figure 8.2(c) for the vertex $m=78$. The frequency shifted version of these two vertex-domain kernels, shown in Figures 8.2 (a) and (c), are given respectively in Figures $8.2(\mathrm{~b})$ and $(\mathrm{d})$, where $\mathcal{H}_{m, 20}(n)=h_{m}(n) u_{20}(n)$ is shown for $m=34$ and $m=78$, respectively.

Next, the vertex-frequency representation, $S(n, k)$, using the LGFT and the localization window defined in the spectral domain is shown in Figure 8.3. From this representation, we can clearly identify the three constituent signal components, within their intervals of support. The marginal properties, such as the projections of $S(n, k)$ onto the vertex index axis and the spectral index axis, are also clearly distinguishable. From the marginal properties, we can conclude that the considered graph signal in hand is spread over all vertex indices, while its spectral localization is dominated by the three spectral indices which correspond to the three components of the original graph signal. In an ideal case of


Figure 8.2: Illustration of localization kernels, $\mathcal{H}_{m, k}(n)=h_{m}(n) u_{k}(n)$, for vertexfrequency analysis based on spectral domain defined windows within the local graph Fourier transform, $S(m, k)=\sum_{n=0}^{N-1} x(n) \mathcal{H}_{m, k}(n)$. (a) Localization kernel $\mathcal{H}_{34,0}(n)=$ $h_{34}(n) u_{0}(n) \sim h_{34}(n)$, for a constant eigenvector, $u_{0}(n)=1 / \sqrt{N}$, centered at the vertex $m=34$. (b) The same localization kernel as in (a) but centered at the vertex $m=78$. (c) Localization kernel, $\mathcal{H}_{34,20}(n)=h_{34}(n) u_{20}(n)$, centered at the vertex $m=34$ and frequency shifted by $u_{20}(n)$. Notice that the variations in kernel amplitude indicate the effects of modulation of the localization window, $h_{m}(n)$. (d) The same localization kernel as in (c), but centered at the vertex $m=78$. (e) Three-dimensional representation of the kernel $\mathcal{H}_{34,0}(n)=h_{34}(n) u_{0}(n)$. (f) Threedimensional representation of the kernel $\mathcal{H}_{78,0}(n)=h_{78}(n) u_{0}(n)$.


Figure 8.3: Local vertex-frequency spectrum calculated using the LGFT and the vertex-frequency localized kernels defined in the spectral domain, as in (14.7). From this representation, observe that the graph signal consists of three distinct components located at spectral indices $k=8, k=66$, and $k=27$, with the corresponding vertex index subsets $\mathcal{V}_{1}, \mathcal{V}_{2}$, and $\mathcal{V}_{3}$, where $\mathcal{V}_{1} \cup \mathcal{V}_{2} \cup \mathcal{V}_{3}=\mathcal{V}$. The marginal (vertex and spectrum-wise) properties are shown in the panels to the right and below the vertexfrequency representation. Observe that, while the graph signal is spread across all vertices, its spectral content is localized at the three spectral indices which correspond to the constituent signal components. In an ideal case of vertex-frequency analysis, these marginals should be respectively equal to $|x(n)|^{2}$ and $|X(k)|^{2}$.
vertex-frequency analysis, these marginals should respectively be equal to $|x(n)|^{2}$ and $|X(k)|^{2}$, which is not the case here.

The calculation of (14.8) is computationally demanding, as in addition to the double summation (where it can be reduced having in mind the low-pass nature of the function $H(p)$ and its possible truncation), the bulk of computational load comes from the eigendecomposition of the graph Laplacian. Although this decomposition is performed only once for a given graph and is signal independent, the full eigendecomposition of a graph with $N$ vertices requires an oder of $N^{3}$ numerical operations.

For a large graph, this can limit the application of this approach. The issue of computational complexity was a motivation to introduce vertexfrequency analysis without the need for eigendecomposition, which will be presented later.

## Spectral Domain Localization of the LGFT

Recall that the classical STFT admits frequency localization in the spectral domain; this is achieved based on the DFT of the original signal and a spectral domain window. For graph signals, we may also adapt this approach to perform signal localization in the spectral domain, whereby the LGFT is obtained as an inverse GFT of $X(p)$ that is localized by a spectral domain window, $H(k-p)$, which is centered around spectral index $k$, that is

$$
\begin{equation*}
S(m, k)=\sum_{p=0}^{N-1} X(p) H(k-p) u_{p}(m) \tag{14.11}
\end{equation*}
$$

Note that this form of the LGFT can be entirely implemented in the graph spectral domain, without a graph shift operator in the vertex domain.

Remark 71: Recall that the classical time-frequency analysis counterpart of (14.11) is Stankovic et al. (2014)

$$
S(m, k)=\frac{1}{\sqrt{N}} \sum_{p=0}^{N-1} X(p) H(k-p) e^{j \frac{2 \pi}{N} m p}
$$

The spectral domain LGFT form in (14.11) can be implemented using band-pass transfer functions, $H_{k}\left(\lambda_{p}\right)=H(k-p)$, as

$$
\begin{equation*}
S(m, k)=\sum_{p=0}^{N-1} X(p) H_{k}\left(\lambda_{p}\right) u_{p}(m) \tag{14.12}
\end{equation*}
$$

The elements $S(m, k), m=0,1, \ldots, N-1$ of the LGFT matrix $\mathbf{S}$ can also be written in a matrix form, where the $k$-th column is defined as

$$
\begin{equation*}
\mathbf{s}_{k}=\operatorname{IGFT}_{p}\left\{X(p) H_{k}\left(\lambda_{p}\right)\right\}=\mathbf{U} H_{k}(\boldsymbol{\Lambda}) \mathbf{X}, \tag{14.13}
\end{equation*}
$$

where $H_{k}(\boldsymbol{\Lambda})$ is a diagonal matrix with elements $H_{k}\left(\lambda_{p}\right), p=0,1, \ldots$, $N-1$.

Remark 72: The kernel in (14.7) is defined based on a low-pass transfer function $H(k)$, which is appropriately shifted in the spectral domain using the modulation term, $u_{k}(n)$. The transfer function in (14.12), $H_{k}\left(\lambda_{p}\right)$, is centered (shifted) at a spectral index, $k$, by definition. Hence, in this case, the modulation term, $u_{k}(n)$, is not needed and the kernel is now of the form

$$
\begin{equation*}
\mathcal{H}_{m, k}(n)=\sum_{p=0}^{N-1} H_{k}\left(\lambda_{p}\right) u_{p}(m) u_{p}(n) \tag{14.14}
\end{equation*}
$$

## LGFT Realization with Band-Pass Functions

Assume that the GFT of the localization window, $h_{m}(n)$, corresponds to the transfer function of a band-pass system on a graph, centered at an eigenvalue, $\lambda_{k}$, and around it, and that it is defined in the form of a polynomial given by

$$
\begin{equation*}
H_{k}\left(\lambda_{p}\right)=h_{0, k}+h_{1, k} \lambda_{p}+\cdots+h_{M-1, k} \lambda_{p}^{M-1} \tag{14.15}
\end{equation*}
$$

with $(M-1)$ as the polynomial order and $k=0,1, \ldots, K$, where $K$ is the number of spectral bands.

The vertex shifted version of the window, $h_{m}(n)$, has the GFT of the form, $\operatorname{GFT}\left\{h(n) * \delta_{m}(n)\right\}=H(p) u_{p}(m)$. Therefore, the inverse GFT of $H_{k}\left(\lambda_{p}\right) u_{p}(m)$ represents a vertex domain kernel, where $H_{k}\left(\lambda_{p}\right)$ is centered at the spectral index $k$ by definition, while $u_{p}(m)$ corresponds to the shift in the vertex domain which centers the window at the vertex $m$. In other words, this kernel, centered around the spectral index $k$ and vertex $m$, is defined as

$$
\begin{equation*}
\mathcal{H}_{m, k}(n)=\sum_{p=0}^{N-1} H_{k}\left(\lambda_{p}\right) u_{p}(m) u_{p}(n) \tag{14.16}
\end{equation*}
$$

Remark 73: It is important to emphasize crucial difference between the vertex-frequency kernels in (14.7) and (14.16). The kernel in (14.7) is defined based on the low-pass transfer function $H(k)$, such as in (14.2), appropriately shifted in the vertex domain and the spectral domain, to
be centered at a vertex $m$ and at a spectral index $k$. This is achieved involving adequate modulation terms $u_{k}(n)$ and $u_{p}(m)$. The transfer function in the kernel given by (14.16), $H_{k}\left(\lambda_{p}\right)$, is centered at $k$ by definition (14.15). Hence, it is needed to perform the spectral modulation only, by $u_{p}(m)$, in order to center the kernel, $\mathcal{H}_{m, k}(n)$, at a vertex $m$. Therefore, the main difference between the kernels in (14.7) and (14.16) is that the spectral shift in (14.7) is achieved by a modulation in the vertex domain using $u_{k}(n)$, while in (14.16) the kernel is directly shifted (defined as a pass-band function) in the spectral domain.

Classical time-frequency domain kernel. To additionally clarify the previous two forms of kernels, we will observe their special cases for a circular directed graph and write the kernels in the classical time-frequency domain.

The kernel defined by (14.7) uses low-pass function $H(k)$ and assumes the following form

$$
\begin{aligned}
\mathcal{H}_{m, k}(n) & =\frac{1}{N^{3 / 2}} \sum_{p=0}^{N-1} H(p) e^{-j \frac{2 \pi}{N} m p} e^{j \frac{2 \pi}{N} n p} e^{-j \frac{2 \pi}{N} k n} \\
& =\frac{1}{N} h(n-m) e^{-j \frac{2 \pi}{N} k n}=\frac{1}{\sqrt{N}} h_{k}(n-m)
\end{aligned}
$$

which is shifted for $m$ in time, and modulated by the $k$ th eigenvector elements $u_{k}^{*}(n)=e^{-j \frac{2 \pi}{N} k n} / \sqrt{N}$, to achieve centering around the spectral index $k$.

The classical time-frequency domain form of the kernel in (14.16) is given by

$$
\begin{aligned}
\mathcal{H}_{m, k}(n) & =\frac{1}{N} \sum_{p=0}^{N-1} H_{k}\left(\lambda_{p}\right) e^{-j \frac{2 \pi}{N} m p} e^{j \frac{2 \pi}{N} n p} \\
& =\frac{1}{N} \sum_{p=0}^{N-1} H(p-k) e^{-j \frac{2 \pi}{N} m p} e^{j \frac{2 \pi}{N} n p}=\frac{1}{\sqrt{N}} h_{k}(n-m)
\end{aligned}
$$

where $h_{k}(n-m)$ is the temporary shifted version of $h_{k}(n)=$ $\operatorname{IGFT}\left\{H_{k}\left(\lambda_{p}\right)\right\}=\operatorname{IDFT}\{H(k-p)\}$, which corresponds to the already frequency shifted (band-pass) transfer function $H_{k}\left(\lambda_{p}\right)=H(p-k)$.

In the case of kernel (14.16), the local vertex-frequency transform for a vertex, $m$, and a spectral index, $k$, becomes

$$
\begin{align*}
S(m, k) & =\sum_{n=0}^{N-1} \mathcal{H}_{m, k}(n) x(n) \\
& =\sum_{n=0}^{N-1} \sum_{p=0}^{N-1} x(n) H_{k}\left(\lambda_{p}\right) u_{p}(m) u_{p}(n)=\sum_{p=0}^{N-1} X(p) H_{k}\left(\lambda_{p}\right) u_{p}(m) \tag{14.17}
\end{align*}
$$

The relation (14.17) can be written in a vector form as

$$
\begin{equation*}
\mathbf{s}_{k}=\mathbf{U} H_{k}(\boldsymbol{\Lambda}) \mathbf{U}^{T} \mathbf{x}=H_{k}(\mathbf{L}) \mathbf{x}=\sum_{p=0}^{M-1} h_{p, k} \mathbf{L}^{p} \mathbf{x} \tag{14.18}
\end{equation*}
$$

where $\mathbf{s}_{k}$ is the column vector with elements $S(m, k), m=0,1, \ldots$, $N-1$, and the property of the eigendecomposition of a matrix polynomial is used in derivation. The number of bands (shifted transfer functions, $\left.H_{k}\left(\lambda_{p}\right), k=0,1, \ldots, K\right)$ is equal to $K+1$ and is not related to the total number of indices, $N$.

Example 55: Consider the simplest decomposition into a low-pass and high-pass part of a graph signal, with $K=1$. In this case, the two values, $k=0$ and $k=1$, represent respectively the low-pass part and high-pass part of the graph signal. Such a decomposition can be achieved using the graph Laplacian with $h_{0,0}=1, h_{0,1}=-1 / \lambda_{\max }$, and $h_{1,0}=0, h_{1,1}=1 / \lambda_{\max }$, where the coefficients are chosen so as to form a simple linearly decreasing function of $\lambda_{p}$ for the low-pass, and a linearly increasing function of $\lambda_{p}$ for the high-pass, in the corresponding transfer functions. These low-pass and high-pass transfer functions are respectively given by

$$
H_{0}\left(\lambda_{p}\right)=\left(1-\frac{\lambda_{p}}{\lambda_{\max }}\right), \quad H_{1}\left(\lambda_{p}\right)=\frac{\lambda_{p}}{\lambda_{\max }}
$$

which leads to the vertex domain implementation of the LGFT in the form

$$
\mathbf{s}_{0}=\left(\mathbf{I}-\frac{1}{\lambda_{\max }} \mathbf{L}\right) \mathbf{x}, \quad \mathbf{s}_{1}=\frac{1}{\lambda_{\max }} \mathbf{L} \mathbf{x}
$$

To improve the spectral resolution, we can employ the same transfer function, but divide the low-pass part into its low-pass and high-pass part. The same can be performed for the high-pass part, to obtain

$$
\mathbf{s}_{00}=\left(\mathbf{I}-\frac{\mathbf{L}}{\lambda_{\max }}\right)^{2} \mathbf{x}, \quad \mathbf{s}_{01}=2\left(\mathbf{I}-\frac{\mathbf{L}}{\lambda_{\max }}\right) \frac{\mathbf{L}}{\lambda_{\max }} \mathbf{x}, \quad \mathbf{s}_{11}=\frac{\mathbf{L}^{2}}{\lambda_{\max }^{2}} \mathbf{x}
$$

The factor 2 appears in the new middle pass-band, $\mathbf{s}_{01}$, since the low-high-pass and the high-low-pass components are the same.

A division into $(K+1)$ bands would correspond to the terms of a binomial form

$$
\left(\left(\mathbf{I}-\mathbf{L} / \lambda_{\max }\right)+\mathbf{L} / \lambda_{\max }\right)^{K} \mathbf{x},
$$

with the corresponding transfer functions in the vertex domain given by

$$
H_{k}(\mathbf{L})=\binom{K}{k}\left(\mathbf{I}-\frac{1}{\lambda_{\max }} \mathbf{L}\right)^{K-k}\left(\frac{1}{\lambda_{\max }} \mathbf{L}\right)^{k}
$$

Example 56: Consider the transfer functions $H_{k}\left(\lambda_{p}\right), p=0,1, \ldots$, $N-1, k=0,1, \ldots, K$ in the spectral domain, corresponding to the binomial form terms for $K=25$, which are shown in Figure 8.4(a). These functions are used for the LGFT calculation at vertex indices $m=$ $0,1, \ldots, N-1$ in the $k=0,1, \ldots, K$ bands for the graph and signal from Figure 8.1. Since the bands are quite spread out, the resulting LGFT is also spread along the frequency axis. The frequency concentration can be improved by reassigning the values of $S(m, k)$ to the position of their maximum value along the frequency band index, $k$, for each vertex index, $m$. The so reassigned LGFT values are given in Figure 8.5.

Of course, any band-pass function, $H_{k}(\boldsymbol{\Lambda})$, can be used in (14.12) or (14.17) to produce the LGFT in the form

$$
\begin{equation*}
\mathbf{s}_{k}=\mathbf{U} H_{k}(\boldsymbol{\Lambda}) \mathbf{U}^{T} \mathbf{x}=H_{k}(\mathbf{L}) \mathbf{x} \tag{14.19}
\end{equation*}
$$

Commonly used examples of such band-pass functions are the spline or raised cosine (Hann window) functions. We will next use the general form of the shifted raised cosine functions as the transfer functions,


Figure 8.4: Exemplar of transfer functions in the spectral domain. (a) The spectral domain transfer functions $H_{k}\left(\lambda_{p}\right), p=0,1, \ldots, N-1, k=0,1, \ldots, K$ which correspond to the binomial form terms for $K=25$. (b) The transfer functions $H_{k}\left(\lambda_{p}\right)$, $p=0,1, \ldots, N-1, k=0,1, \ldots, K$ which correspond to the raised cosine (Hann) window form for $K=25$. (c) The spectral index-varying (wavelet-like) transfer functions $H_{k}\left(\lambda_{p}\right), p=0,1, \ldots, N-1, k=0,1, \ldots, K$ which correspond to the raised cosine (Hann) window form for $K=10$. The transfer function $H_{9}(\lambda)$ is designated by the thick black line for each considered domain, while its discrete values at $\lambda_{p}$, $H_{9}\left(\lambda_{p}\right)$, are shown in gray, in panels (b) and (c).


Figure 8.5: Vertex-frequency representation of a three-component signal in Figure 8.1(d). (a) The LGFT of the signal from Figure 8.1(d), calculated using the transfer functions for frequency selection given in Figure 8.4(a). The LGFT values, $S(m, k)$, were reassigned to the position of its maximum value along the frequency band index, $k$, for each vertex index, $m$. (b) The LGFT of the signal from Figure 8.1(d), calculated using the transfer functions for frequency selection given in Figure 8.4(b). The LGFT values, $S(m, k)$, were reassigned to the positions of their maximum values along the frequency band index, $k$, for each vertex index, $m$. (c) The LGFT of the signal from Figure 8.1(d), calculated using the wavelet-like transfer functions for frequency selection given in Figure 8.4(c).
defined by

$$
H_{k}(\lambda)= \begin{cases}\sin ^{2}\left(\frac{\pi}{2} \frac{a_{k}}{b_{k}-a_{k}}\left(\frac{\lambda}{a_{k}}-1\right)\right), & \text { for } a_{k}<\lambda \leq b_{k}  \tag{14.20}\\ \cos ^{2}\left(\frac{\pi}{2} \frac{b_{k}}{c_{k}-b k}\left(\frac{\lambda}{b_{k}}-1\right)\right), & \text { for } b_{k}<\lambda \leq c_{k} \\ 0, & \text { elsewhere }\end{cases}
$$

where $\left(a_{k}, b_{k}\right]$ and $\left(b_{k}, c_{k}\right], k=1,2, \ldots, K$, define the spectral bands for $H_{k}(\boldsymbol{\Lambda})$. For uniform bands within $0 \leq \lambda \leq \lambda_{\max }$, the intervals can be defined by

$$
\begin{align*}
a_{k} & =a_{k-1}+\frac{\lambda_{\max }}{K} \\
b_{k} & =a_{k}+\frac{\lambda_{\max }}{K}  \tag{14.21}\\
c_{k} & =a_{k}+2 \frac{\lambda_{\max }}{K}
\end{align*}
$$

with $a_{1}=0$. The initial transfer function, $H_{0}(\lambda)$, is defined using only $0=b_{0} \leq \lambda \leq c_{0}=\lambda_{\max } / K$, while the last transfer function, $H_{K}(\lambda)$, is defined using the interval $a_{K}<\lambda \leq b_{K}=\lambda_{\text {max }}$ in (14.20).

The raised cosine transfer function satisfy the following condition

$$
\begin{equation*}
\sum_{k=0}^{K} H_{k}\left(\lambda_{p}\right)=1 \tag{14.22}
\end{equation*}
$$

The conditions for graph signal reconstruction from the LGFT will be discussed in Section 14.2.

Example 57: The shifted raised cosine functions, defined by (14.20) and (14.21), are shown in Figure 8.4(b) for the graph from Figure 8.1, for $K=25$. These functions are used for the LGFT calculation of the graph signal from Figure 8.1 at the vertex indices $m=0,1, \ldots, N-1$, and in $(K+1)$ spectral bands, $k=0,1, \ldots, K$. The absolute LGFT values are given in Figure 8.5(b). Spectral resolution depends on the number of bands $K$, with a larger number of spectral bands resulting in a higher spectral resolution.

Example 58: The experiment from Examples 56 and 57 is repeated with varying bounds of the spectral intervals in the raised cosine transfer
functions $H_{k}\left(\lambda_{p}\right), p=0,1, \ldots, N-1, k=0,1, \ldots, K$. The spectral index-varying (wavelet-transform like) form of the raised cosine transfer functions $H_{k}\left(\lambda_{p}\right), p=0,1, \ldots, N-1, k=0,1, \ldots, K$, is defined by the interval bounds $\lambda_{\max }((1.5+p) / 11.5)^{5}$, for $p=0,1,2, \ldots, 10$,

$$
\begin{gathered}
a_{k} \in\{0,0.004,0.02,0.07,0.19,0.44,0.9,1.7,2.9\}, \\
b_{k} \in\{0.004,0.02,0.07,0.19,0.44,0.9,1.7,2.9,4.8\}, \\
c_{k} \in\{0.02,0.07,0.19,0.44,0.9,1.7,2.9,4.8,7.63\}, \\
k=1,2, \ldots, 9
\end{gathered}
$$

and depicted in Figure 8.4(c). The LGFT values, $S(m, k)$, calculated with the so-obtained transfer functions, $H_{k}\left(\lambda_{p}\right)$, are shown in Figure 8.5(c). In order to illustrate the change of resolution in this case, the LGFT was reassigned to each eigenvalue $\lambda_{p}, p=0,1, \ldots, N-1$, and shown in Figure 8.6. As in classical wavelet transform, the spectral resolution is lower for the higher spectral indices.


Figure 8.6: Vertex-frequency representation from Figure 8.5(c) with the axis of the eigenvalue index, $p$, instead of the frequency band index, $k$. The same value of LGFT, $S(m, k)$, is assigned to each spectral index, $p$, when $\lambda_{p} \in\left(\frac{a_{k}+b_{k}}{2}, \frac{b_{k}+c_{k}}{2}\right]$, and without any scaling.

## Signal Adaptive LGFT

The spectral graph wavelet-like transform is just an example of varying spectral transfer functions in the LGFT, where the spectral resolution is the highest (spectral wavelet functions narrowest) for small values of the smoothness index, $\lambda_{p}$ (Behjat and Van De Ville, 2019). The spectral resolution decreases as the spectral wavelet functions become wider for large smoothness index values, Figure 8.4(c). In general, the change of resolution may be arbitrary and signal adaptive, for example, the resolution may be higher for the spectral intervals of $\lambda$ which are rich in signal components and lower within the intervals where there are no signal components.

Before introducing an example with a signal adaptive LGFT, we will modify the transfer functions, $H_{k}\left(\lambda_{p}\right)$, in (14.20) to satisfy the condition

$$
\begin{equation*}
\sum_{k=0}^{K} H_{k}^{2}\left(\lambda_{p}\right)=1 \tag{14.23}
\end{equation*}
$$

as this will be important for the frame-based LGFT inversion.
Notice that a simple transformation of the transfer functions, $H_{k}\left(\lambda_{p}\right) \rightarrow H_{k}^{2}\left(\lambda_{p}\right)$, would allow for the condition $\sum_{k=0}^{K} H_{k}^{2}\left(\lambda_{p}\right)=1$ to hold instead of $\sum_{k=0}^{K} H_{k}\left(\lambda_{p}\right)=1$. This means that a simple removal of squares in the sine and cosine functions in (14.20) would produce a form to satisfy the condition $\sum_{k=0}^{K} H_{k}^{2}\left(\lambda_{p}\right)=1$. Both of these conditions will be used in Section 14.2 in various approaches to the graph signal reconstruction from the LGFT.

By removing the squares in the sine and cosine functions in (14.20), their first derivative loses continuity in $\lambda$ at the end interval points. In order to preserve continuous derivatives, the arguments in the sine and cosine functions can be mapped by a polynomial,

$$
v_{x}(x)=x^{4}\left(35-84 x+70 x^{2}-20 x^{3}\right), \quad \text { for } 0 \leq x \leq 1
$$

with $v_{x}(0)=0$ and $v_{x}(1)=1$. In this way, we arrive at the Meyer wavelet-like transfer functions (Meyer, 1992) for the LGFT calculation,
given by

$$
H_{k}(\lambda)= \begin{cases}\sin \left(\frac{\pi}{2} v_{x}\left(\frac{a_{k}}{b_{k}-a_{k}}\left(\frac{\lambda}{a_{k}}-1\right)\right)\right), & \text { for } a_{k}<\lambda \leq b_{k}  \tag{14.24}\\ \cos \left(\frac{\pi}{2} v_{x}\left(\frac{b_{k}}{c_{k}-b_{k}}\left(\frac{\lambda}{b_{k}}-1\right)\right)\right), & \text { for } b_{k}<\lambda \leq c_{k} \\ 0, & \text { elsewhere }\end{cases}
$$

The initial transfer function, $k=0$, and the last transfer function, $k=K$, are calculated using only the half of the interval, as explained after the spectral band definition in relation (14.21).
Example 59: The transfer functions of the form defined in (14.24) are used with signal adaptive intervals. These intervals are defined in such a way that they are small (fine) around $\lambda$, where a significant signal spectral content is detected, and are big (rough) around $\lambda$ where the signal spectral content is low, as in Figures 8.7(a) and (b). The intervals are narrow (with a high resolution) around the three signal components at $\lambda=0.38, \lambda=1.87$, and $\lambda=4.62$. Vertex-frequency representation with these transfer functions is shown in Figures 8.7(c) and (d) with the spectral band index, $k$, and the assigned eigenvalue (spectral) index, $p$, as a spectral axis. Fine intervals around the spectral signal components allowed for high spectral resolution representation, as in Figure 8.7(c), with a smaller number of transfer functions $K+1=17$. A wider interval width for the third component resulted in a lower spectral resolution than in the case of the other two components.

## Polynomial LGFT Approximation

Bandpass LGFT functions, $H_{k}(\lambda), k=0,1, \ldots, K$, of the form (14.20) or (14.24) can be implemented using the Chebyshev finite $(M-1)$-order polynomial approximation, $\bar{P}_{k, M-1}(\lambda), k=0,1, \ldots, K$, of the form

$$
\begin{equation*}
\bar{P}_{k, M-1}(\lambda)=\frac{c_{k, 0}}{2}+\sum_{m=1}^{M-1} c_{k, m} \bar{T}_{m}(\lambda) \tag{14.25}
\end{equation*}
$$

This leads to the vertex domain implementation of the spectral LGFT form, given by

$$
\mathbf{s}_{k}=\bar{P}_{k, M-1}(\mathbf{L}) \mathbf{x}
$$



Figure 8.7: A graph signal and transfer functions in the spectral domain for a signal adaptive LGFT. (a) Graph signal in the spectral domain, $X(p)$, as a function of the eigenvalues, $\lambda_{p}$. (b) The spectral domain transfer functions $H_{k}\left(\lambda_{p}\right), p=0,1, \ldots, N-1$, $k=0,1, \ldots, K$ which satisfy the condition $\sum_{k=0}^{K} H_{k}^{2}\left(\lambda_{p}\right)=1$, with $K=16$. (c) The LGFT of the signal from Figure 8.1(d), calculated using the transfer functions for frequency selection given in (b). (d) Vertex-frequency representation from (c) with the eigenvalue (spectral) index, $p$, axis instead of the frequency band index, $k$. The same value of LGFT, $S(m, k)$, is assigned to each spectral index, $p$, when $\lambda_{p} \in\left(\frac{a_{k}+b_{k}}{2}, \frac{b_{k}+c_{k}}{2}\right]$, without any scaling.

Table 14.1: Coefficients, $h_{i, k}, i=0,1, \ldots, M-1, k=0,1, \ldots, K$, for the polynomial calculation of the LGFT, $\mathbf{s}_{k}$, of a signal, $\mathbf{x}$, in various spectral bands, $k$, shown in Figure 8.8(b). The obtained LGFT of the three-component signal from Figure 8.1(d) is given in Figure 8.9(a)

| $\mathbf{s}_{k}=\left(h_{0, k} \mathbf{I}+h_{1, k} \mathbf{L}+h_{2, k} \mathbf{L}^{2}+h_{3, k} \mathbf{L}^{3}+h_{4, k} \mathbf{L}^{4}+h_{5, k} \mathbf{L}^{5}\right) \mathbf{x}$ |  |  |  |  |  |  |
| :--- | ---: | ---: | ---: | ---: | ---: | ---: |
| $k$ | $h_{0, k}$ | $h_{1, k}$ | $h_{2, k}$ | $h_{3, k}$ | $h_{4, k}$ | $h_{5, k}$ |
| 0 | 1.062 | -1.925 | 1.168 | -0.3115 | 0.03776 | -0.001702 |
| 1 | -0.002 | 1.773 | -1.655 | 0.5357 | -0.07250 | 0.003508 |
| 2 | -0.154 | 1.016 | -0.601 | 0.1295 | -0.01155 | 0.000349 |
| 3 | 0.005 | -0.301 | 0.621 | -0.2674 | 0.04200 | -0.002225 |
| 4 | 0.089 | -0.748 | 0.869 | -0.3042 | 0.04217 | -0.002040 |
| 5 | 0.060 | -0.381 | 0.319 | -0.0704 | 0.00461 | 0.000000 |
| 6 | -0.024 | 0.277 | -0.430 | 0.2055 | -0.03570 | 0.002040 |
| 7 | -0.076 | 0.598 | -0.714 | 0.2814 | -0.04292 | 0.002225 |
| 8 | -0.027 | 0.159 | -0.122 | 0.0198 | 0.00177 | -0.000349 |
| 9 | 0.087 | -0.699 | 0.868 | -0.3662 | 0.06140 | -0.003508 |
| 10 | -0.026 | 0.220 | -0.293 | 0.1333 | -0.02435 | 0.001536 |

for $k=0,1,2, \ldots, K$, with

$$
\begin{align*}
\bar{P}_{k, M-1}(\mathbf{L}) & =\frac{c_{k, 0}}{2}+\sum_{m=1}^{M-1} c_{k, m} \bar{T}_{m}(\mathbf{L}) \\
& =h_{0, k} \mathbf{I}+h_{1, k} \mathbf{L}+h_{2, k} \mathbf{L}^{2}+\cdots+h_{(M-1), k} \mathbf{L}^{M-1} \tag{14.26}
\end{align*}
$$

as discussed in Section 9.5 and shown in Table 14.1. The polynomial form in (14.26) uses only the ( $M-1$ )-neighborhood in calculation of the LGFT for each considered vertex, without the need for eigendecomposition analysis, thus significantly reducing the computational cost.
Example 60: Consider the shifted transfer functions, $H_{k}(\lambda), k=$ $0,1, \ldots, K$, defined by (14.20) and (14.21), shown in Figure 8.8(a), for $K=10$. Functions $H_{k}(\lambda)$ satisfy $\sum_{k=0}^{K} H_{k}(\lambda)=1$, which is numerically confirmed and designated by the horizontal dotted line in 8.8(a). Each individual transfer function, $H_{k}(\lambda)$, is approximated using the Chebyshev polynomial, $\bar{P}_{k, M-1}, k=0,1, \ldots, K$, as detailed in Section 9.5, with three polynomial orders defined by $M=6, M=20$ and $M=80$. These polynomial approximations are shown in Figures 8.8(b)-(d). In each considered case, summations $\sum_{k=0}^{K} \bar{P}_{k, M-1}(\lambda)$ are calculated. It can


Figure 8.8: Chebyshev approximation of LGFT transfer functions, which correspond to the raised cosine window in the spectral domain. (a) Original transfer functions $H_{k}(\lambda), k=0,1, \ldots, K$, for $K=10$. The dotted horizontal line designates $\sum_{k=0}^{K} H_{k}(\lambda)$. (b) Polynomial Chebyshev approximations, $\bar{P}_{k, M-1}(\lambda), k=0,1, \ldots, K$, with $M=6$. (c) Polynomial Chebyshev approximations, $\bar{P}_{k, M-1}(\lambda), k=0,1, \ldots, K$, with $M=20$. (d) Polynomial Chebyshev approximations, $\bar{P}_{k, M-1}(\lambda), k=0,1, \ldots, K$, with $M=80$. The dotted horizontal line designates $\sum_{k=0}^{K} \bar{P}_{k, M-1}(\lambda)$, which is close to 1 in all considered approximations, thus guaranteeing stable transform invertibility. Transfer function $H_{6}(\lambda)$ and approximations, $\bar{P}_{6, M-1}(\lambda)$, are designated by the thick black line.
be observed that for different values of $M$, the summations in all considered cases are very close to 1 , thus guaranteeing numerically stable invertibility of the LGFT, as discussed later.

The so obtained approximations of transfer functions, $H_{k}(\lambda)$, are used for the LGFT based vertex-frequency analysis. Absolute LGFT values, calculated for the three-component graph signal from Figure 8.1(d), are shown in Figures 8.9 (a)-(c), for $M=6, M=20$ and $M=80$. Low resolution in Figure 8.9(a) is directly related to the imprecise and very wide (with a low spectral resolution) approximation of the spectral transfer functions for $M=6$, in Figure 8.8(b). Notice that high values of the polynomial order, $(M-1)$, increase calculation complexity and require wide vertex neighborhood in the calculation of the LGFT.

Based on the analysis of calculation complexity in Section 9.5 we may conclude that an order of $K M N_{\mathbf{L}}$ of arithmetic operations is needed to calculate the LGFT in the vertex domain, with $(K+1)$ spectral bands, using a polynomial whose order is $(M-1)$. The number of nonzero elements in the graph Laplacian is denoted by $N_{\mathbf{L}}$.

## The Spectral Graph Wavelet Transform

Several attempts have been made to extend the classical wavelet analysis to general graph signals, some of which were performed on specific tree graphs (Ann B. Lee and Wasserman, 2008; Murtagh, 2007). The most significant attempts to define the wavelet transform on general graphs have been: (i) a lifting-based approach for multi-scale representation of graph signals (Jansen et al., 2009; Narang and Ortega, 2009; Rustamov and Guibas, 2013), (ii) diffusion-based wavelets and diffusion based polynomial frames (Coifman and Lafon, 2006; Maggioni and Mhaskar, 2008), and (iii) separable filter-bank wavelets (Narang and Ortega, 2012). The wavelet definition that can be directly related to the presented spectral domain local graph Fourier transform, and has been commonly used in the graph signal analysis, is based on the extension of the spectral domain form of the classical wavelet transform and its polynomial approximations, and was introduced in Hammond et al. (2011a).


Figure 8.9: Vertex-frequency representation of a three-component signal in Figure 8.1(d). The LGFT is based on raised cosine (Hann window) like bandpass transfer functions for frequency selection, with $K=10$, approximated using the Chebyshev polynomials of various order, as shown in Figures 8.8(b)-(d). (a) The LGFT of the signal from Figure 8.1(d), calculated using the Chebyshev polynomial approximation of transfer functions given in Figure 8.8(b), with $M=6$. (b) The LGFT of the signal from Figure 8.1(d), calculated using the transfer functions for frequency selection given in Figure 8.4(c), with $M=20$. (c) The LGFT of the signal from Figure 8.1(d), calculated using the transfer functions for frequency selection given in Figure 8.4(d), with $M=80$. Low resolution in (a) can be directly related with low $M=6$ used in approximation in Figure 8.8(b). The resolution is considerably improved for $M=20$.

In classical signal processing theory, time-frequency analysis has many common goals with the wavelet transform (and its generalization in the form of time-scale analysis). However, these two areas are usually considered separately. The main goals of the wavelet analysis are to perform multi-resolution signal analysis, compression, and signal processing, including the wavelet domain sparsity-driven signal denoising. The main goals in classical time-frequency analysis are in spectral and signal parameter estimation (like, for example, the instantaneous frequency), joint time-frequency domain processing, detection, and denoising of nonstationary signals.

Since the same relation between these areas can be assumed for graph signal processing, we shall consider only the spectral wavelet transform, which is directly related to the frequency-varying LGFT and can be considered as a special case of the frequency-varying vertex-frequency analysis, rather than a transform aimed at graph signal compression and its wavelet-like multi-resolution analysis.

The classic wavelet analysis is based on defining the "mother wavelet" and using its dilatated and translated versions to create signal decomposition kernels. A direct extension of this concept is not possible on graphs as irregular signal domains, since the operations of dilatation and translation are not possible in the same way as in the case of simple regularly sampled line as the signal domain. As in classical signal processing, wavelet coefficients can be defined as a projection of a graph signal onto the wavelet kernel functions. Assume that the basic form for the wavelet definition in the spectral domain is $H\left(\lambda_{p}\right)$. The wavelet in spectral domain then represents a scaled version of $H\left(\lambda_{p}\right)$ in the scale $s_{i}, i=1,2, \ldots, K-1$, and is denoted by Hammond et al. (2019), Behjat and Van De Ville (2019), Behjat et al. (2015), Rustamov and Guibas (2013), Jestrović et al. (2017), Masoumi et al. (2019), and Cioacă et al. (2019)

$$
H_{i}\left(\lambda_{p}\right)=H\left(s_{i} \lambda_{p}\right) .
$$

Additionally, a low-pass scale (father wavelet) function $G\left(\lambda_{p}\right)$, plays the role of low-pass function, $H_{0}\left(\lambda_{p}\right)$, in the LGFT. Therefore, a set of discrete scales for the wavelet calculation, denoted by $s \in$ $\left\{s_{1}, s_{2}, \ldots, s_{K-1}\right\}$, is assumed with the corresponding spectral transfer
functions, $H_{s_{i}}\left(\lambda_{p}\right)$ and $G\left(\lambda_{p}\right)$. Now, in the same way as in the case of the kernel form of the LGFT in (14.8), the graph wavelet transform is defined using the band-pass scaled wavelet kernel, $\psi_{m, s_{i}}(n)$, instead of the LGFT kernel, $\mathcal{H}_{m, k}(n)$, in (14.14). This yields

$$
\begin{equation*}
\psi_{m, s_{i}}(n)=\sum_{p=0}^{N-1} H\left(s_{i} \lambda_{p}\right) u_{p}(m) u_{p}(n) \tag{14.27}
\end{equation*}
$$

which corresponds to the LGFT kernel, $\mathcal{H}_{m, k}(n)$, defined in (14.16). This yields the wavelet coefficients given by

$$
\begin{aligned}
W\left(m, s_{i}\right) & =\sum_{n=0}^{N-1} \psi_{m, s_{i}}(n) x(n) \\
& =\sum_{n=0}^{N-1} \sum_{p=0}^{N-1} H\left(s_{i} \lambda_{p}\right) x(n) u_{p}(m) u_{p}(n)=\sum_{p=0}^{N-1} H\left(s_{i} \lambda_{p}\right) X(p) u_{p}(m)
\end{aligned}
$$

The wavelet coefficients may be interpreted as the IGFT of $H\left(s_{i} \lambda_{p}\right) X(p)$, that is

$$
\begin{equation*}
W\left(m, s_{i}\right)=\operatorname{IGFT}\left\{H\left(s_{i} \lambda_{p}\right) X(p)\right\} \tag{14.28}
\end{equation*}
$$

Remark 74: We will use the notation $H\left(s_{i} \lambda\right)=H_{i}(\lambda)$ with the corresponding matrix function form $H_{i}(\boldsymbol{\Lambda})$. Notice that this scale-based indexing is opposite to the classical frequency band indexing. The largest scale for $H\left(s_{1} \lambda\right), 1<s_{1} \lambda \leq M$, is obtained for the smallest $s_{1}$, $1 / s_{1}<\lambda \leq M / s_{1}$, where $M>1$ is the coefficient of the scale changes, which will be explained later. The associated spectral wavelet transfer function, $H\left(s_{1} \lambda\right)=H_{1}(\lambda)$, corresponds to the highest frequency band. The wavelet transfer function in scale $s_{K}, H\left(s_{K} \lambda\right)=H_{K}(\lambda)$, is associated with the lowest frequency band. Notation for the spectral scale function (low-pass transfer function complementary to $H\left(s_{K} \lambda\right)$ within the lowest spectral interval) is $G(\lambda)$. The spectral scale function, $G(\lambda)$, plays the role of low-pass transfer function with spectral index 0 in the LGFT. Therefore, $K$ spectral wavelet transfer functions $H\left(s_{i} \lambda\right)$, $i=1,2, \ldots, K$, along with the scale function $G(\lambda)$, cover exactly $K+1$ spectral bands as in the LGFT case.

According to (9.36), we can write

$$
\begin{equation*}
\mathbf{w}_{i}=H_{i}(\mathbf{L}) \mathbf{x} \tag{14.29}
\end{equation*}
$$

where $\mathbf{w}_{i}$ is a column vector with elements $W\left(m, s_{i}\right), m=0,1, \ldots, N-1$.
If $H_{i}(\lambda)=H\left(s_{i} \lambda\right)$ can be approximated by a polynomial in $\lambda$, $H_{i}(\lambda) \approx P_{i}(\lambda)$, then the relation

$$
\begin{equation*}
\mathbf{w}_{i} \approx P_{i}(\mathbf{L}) \mathbf{x}, \tag{14.30}
\end{equation*}
$$

follows, where $P_{i}(\mathbf{L})$ is a polynomial in the graph Laplacian (see Section 9.5 and Example 60).
Example 61: The wavelet transform (vertex-scale) representation of a three-component signal in Figure 8.1(d), obtained using the Meyerlike graph wavelet in the spectral domain, $\lambda$, will be illustrated here. As in classical wavelet transform, the wavelet in the first scale should correspond to the high-pass transfer function with nonzero values in the interval $\lambda_{\max } / M<\lambda \leq \lambda_{\max }$, where $M>1$ is the coefficient of the scale changes. In classical wavelet transforms the dyadic scheme with $M=2$ is commonly used. The scale based indexing is opposite to the classical frequency indexing, where large indices indicate the high frequency content. The Meyer-like graph wavelet in the first scale is defined by Meyer (1992) and Leonardi and Van De Ville (2013)

$$
H\left(s_{1} \lambda\right)= \begin{cases}\sin \left(\frac{\pi}{2} v_{x}\left(q\left(s_{1} \lambda-1\right)\right)\right), & \text { for } 1<s_{1} \lambda \leq M \\ 0, & \text { elsewhere }\end{cases}
$$

For $2 \leq i \leq K$ the Meyer-like graph wavelet is given by

$$
H\left(s_{i} \lambda\right)= \begin{cases}\sin \left(\frac{\pi}{2} v_{x}\left(q\left(s_{i} \lambda-1\right)\right)\right), & \text { for } 1<s_{i} \lambda \leq M \\ \cos \left(\frac{\pi}{2} v_{x}\left(q\left(\frac{s_{i} \lambda}{M}-1\right)\right)\right), & \text { for } M<s_{i} \lambda \leq M^{2} \\ 0, & \text { elsewhere },\end{cases}
$$

where $q=1 /(M-1)$. The initial interval is defined by $s_{1}=M / \lambda_{\max }$, so that $1<s_{i} \lambda \leq M$ corresponds to $\lambda_{\max } / M<\lambda \leq \lambda_{\max }$, while the other interval bounds are defined using a geometric sequence of scale factors,

$$
s_{i}=s_{i-1} M=s_{1} M^{i-1}=\frac{1}{\lambda_{\max }} M^{i} .
$$

Observe that the larger the scale factor $s_{i}$ (and the scale index $i$ ), the narrower the transfer function, $H\left(s_{i} \lambda\right)$, while the progression coefficient is

$$
M=(q+1) / q>1 .
$$

In classical wavelet transforms the dyadic scheme with $M=2$ is commonly used. The last value of the scale factor, $s_{K}=M^{K} / \lambda_{\max } / M$, is defined by $K$ and indicates how close the last wavelet transfer function is to $\lambda=0$.

The polynomial function, $v_{x}(x)$, is defined by

$$
\begin{align*}
& v_{x}(x)=x^{4}\left(35-84 x+70 x^{2}-20 x^{3}\right), \quad \text { for } 0 \leq x \leq 1, \text { with } \\
& v_{x}(q(0))=v_{x}(0)=0, \quad v_{x}(q(M-1))=v_{x}(1)=1 \tag{14.31}
\end{align*}
$$

The wavelet transfer functions,

$$
H_{i}(\lambda)=H\left(s_{i} \lambda\right),
$$

are of a band-pass type. The main property (condition for the reconstruction) is that the wavelet functions in two successive scales satisfy the following property

$$
\begin{aligned}
& H_{i}^{2}(\lambda)+H_{i+1}^{2}(\lambda) \\
& \quad=\cos ^{2}\left(\frac{\pi}{2} v_{x}\left(q\left(\frac{s_{i} \lambda}{M}-1\right)\right)\right)+\sin ^{2}\left(\frac{\pi}{2} v_{x}\left(q\left(\frac{s_{i} \lambda}{M}-1\right)\right)\right)=1,
\end{aligned}
$$

within

$$
M<s_{i} \lambda \leq M^{2} .
$$

This property implies $\sum_{i=1}^{K} H^{2}\left(s_{i} \lambda\right)=1$ for all $\lambda$ except in the last interval, $s_{K} \lambda \in\left[0, M^{2}\right]$. To handle the low-pass spectral components (the interval for $\lambda$ closest to $\lambda=0$ ), the low-pass type scale function, $G(\lambda)$ ), is added in the form
$G(\lambda)= \begin{cases}1, & \text { for } 0 \leq \lambda \leq M / s_{K}=\lambda_{\max } / M^{K-1} \\ \cos \left(\frac{\pi}{2} v_{x}\left(q\left(\frac{s_{K} \lambda}{M}-1\right)\right)\right), & \text { for } M<s_{K} \lambda \leq M^{2} \\ 0, & \text { elsewhere. }\end{cases}$

Remark 75: The number of wavelet transfer functions, $K$, does not depend on the other wavelet parameters. A large value of $K$ will only increase the number of intervals and the resolution (producing smaller width of the first interval defined by $\lambda_{\max } / M^{K-1}$ ) toward $\lambda \rightarrow 0$, as shown in Figures 8.10(a)-(c).
Remark 76: The wavelet transfer functions, $H\left(s_{i} \lambda\right)$, including the low-pass scale function, $G(\lambda)$, defined in Example 61 satisfy the relation

$$
\sum_{i=1}^{K} H^{2}\left(s_{i} \lambda\right)+G^{2}(\lambda)=1
$$

Example 62: For $q=1, M=2$, and $K=9$ the Meyer wavelet functions are given in Figure 8.10(a). The Meyer wavelet functions for $q=3$, $M=4 / 3, K=13$ and $q=9, M=10 / 9, K=45$ are shown in Figures 8.10(b) and (c). The vertex-frequency representation of the signal from Figure 8.1 using these three sets of wavelet transfer functions are shown in Figures 8.11(a)-(c).
Polynomial SGWT approximation. Chebyshev approximation of the wavelet functions, $H\left(s_{i} \lambda\right)=H_{i}(\lambda)$, in the form

$$
\begin{equation*}
\bar{P}_{i, M-1}(\lambda)=\frac{c_{i, 0}}{2}+\sum_{m=1}^{M-1} c_{i, m} \bar{T}_{m}(\lambda) \tag{14.32}
\end{equation*}
$$

can be used for the vertex domain wavelet transform implementation

$$
\bar{P}_{i, M-1}(\mathbf{L})=\frac{c_{i, 0}}{2}+\sum_{m=1}^{M-1} c_{i, m} \bar{T}_{m}(\mathbf{L}), \quad i=0,1,2, \ldots, K
$$

using only the ( $M-1$ )-neighborhood of each considered vertex, and without any graph Laplacian eigendecomposition analysis. The Chebyshev polynomials can be calculated recursively, as in (9.28), with a change of variables and the recursive implementation as described in detail in Examples 43 and 60.

## Windows Defined Using the Vertex Neighborhood

In order to show that the window, $h_{m}(n)$, which is localized at a vertex $m$ can also be defined using the vertex neighborhood, recall that the


Figure 8.10: Exemplars of Meyer wavelet functions (acting as transfer functions in the wavelet transform), shown in the spectral domain. (a) Band-pass Meyer wavelet functions $H\left(s_{i} \lambda\right), i=1,2, \ldots, K$ and the low-pass scale function $G(\lambda)$, for $K=9$ and $M=2$. (b) Band-pass Meyer wavelet functions $H\left(s_{i} \lambda\right), i=1,2, \ldots, K$ and the low-pass scale function $G(\lambda)$, for $K=13$ and $M=3 / 2$. (c) Band-pass Meyer wavelet functions $H\left(s_{i} \lambda\right), i=0,1, \ldots, K$ and the low-pass function $G(\lambda)$, for $K=45$ and $M=10 / 9$. Transfer functions $H\left(s_{2} \lambda\right), H\left(s_{2} \lambda\right), H\left(s_{5} \lambda\right)$ are designated by the thick black line, for each of the considered setups in (a)-(c), respectively; their values at $\lambda_{p}$ are shown in gray.


Figure 8.11: Vertex-frequency representation of a three-component signal in Figure 8.1(d). (a) The Meyer wavelet transform of the signal from Figure 8.1(d), calculated using the transfer functions for frequency selection given in Figure 8.10(a). (b) The Meyer wavelet transform of the signal from Figure 8.1(d), calculated using the transfer functions for frequency selection given in Figure 8.10(b). (c) The Meyer wavelet transform of the signal from Figure 8.1(d), calculated using the Meyer wavelet transform transfer functions for frequency selection given in Figure 8.10(c). Wavelet values were reassigned to spectral indices, $p$, in order to illustrate the change in resolution. The same value of SGWT, $W(m, k)$, is assigned to each spectral index, $p$, when $\lambda_{p} \in\left(\frac{a_{k}+b_{k}}{2}, \frac{b_{k}+c_{k}}{2}\right]$, without any scaling.
distance, $d_{m n}$, between vertices $m$ and $n$ is equal to the length of the shortest walk from vertex $m$ to vertex $n$, and that $d_{m n}$ takes integer values. Then, the window function can be defined as a function of vertex distance, in the form

$$
h_{m}(n)=g\left(d_{m n}\right)
$$

where $g(d)$ corresponds to any basic window function in classical signal processing. For example, we can use the Hann window, given by

$$
h_{m}(n)=\frac{1}{2}\left(1+\cos \left(\pi d_{m n} / D\right)\right), \quad \text { for } 0 \leq d_{m n}<D
$$

where $D$ is the assumed window width.
For convenience, window functions for every vertex can be calculated in a matrix form as follows:

- For the vertices for which the distance is $d_{m n}=1$, window functions are defined through an adjacency (neighborhood one) matrix $\mathbf{A}_{1}=\mathbf{A}$. In other words, the vertices which belong to the oneneighborhood of a vertex, $m$, are indicated by unit-value elements in the $m$ th row of the adjacency matrix $\mathbf{A}$ (in unweighted graphs). In weighed graphs, the corresponding adjacency matrix $\mathbf{A}$ can be obtained from the weighting matrix $\mathbf{W}$ as $\mathbf{A}=\operatorname{sign}(\mathbf{W})$.
- Window functions for vertices $m$ and $n$, for which the distance is $d_{m n}=2$ are defined by the matrix

$$
\mathbf{A}_{2}=\left(\mathbf{A} \odot \mathbf{A}_{1}\right) \circ\left(\mathbf{1}-\mathbf{A}_{1}\right) \circ(\mathbf{1}-\mathbf{I}),
$$

where the symbol $\odot$ denotes the logical (Boolean) matrix product, - is the Hadamard (element-by-element) product, and $\mathbf{1}$ is a matrix with all elements equal to 1 . The nonzero elements of the $m$ th row of the matrix $\mathbf{A} \odot \mathbf{A}_{1}$ then designate the vertices that are connected to the vertex $m$ with walks of length $K=2$ or lower. It should be mentioned that the element-by-element multiplication of $\left(\mathbf{A} \odot \mathbf{A}_{1}\right)$ by matrix $\left(\mathbf{1}-\mathbf{A}_{1}\right)$ removes the vertices connected with walks of length 1 , while the multiplication by $(\mathbf{1}-\mathbf{I})$ removes the diagonal elements from $\left(\mathbf{A} \odot \mathbf{A}_{1}\right)$.

- For $d_{m n}=d \geq 2$, we arrive at a recursive relation for the calculation of a matrix which will give the information about the vertices
separated by the distance $d$. Such a matrix has the form

$$
\begin{equation*}
\mathbf{A}_{d}=\left(\mathbf{A} \odot \mathbf{A}_{d-1}\right) \circ\left(\mathbf{1}-\mathbf{A}_{d-1}\right) \circ(\mathbf{1}-\mathbf{I}) . \tag{14.33}
\end{equation*}
$$

The window matrix for an assumed graph window width, $D$, can now be defined as

$$
\mathbf{P}_{D}=g(0) \mathbf{I}+g(1) \mathbf{A}_{1}+\cdots+g(D-1) \mathbf{A}_{D-1},
$$

so that a graph signal which is localized around a vertex $m$, may be formed based on this matrix, as

$$
x_{m}(n)=h_{m}(n) x(n)=P_{D}(n, m) x(n) .
$$

The LGFT representation of a graph signal, $x(n)$, then becomes

$$
\begin{equation*}
S(m, k)=\sum_{n=0}^{N-1} x(n) h_{m}(n) u_{k}(n)=\sum_{n=0}^{N-1} x(n) P_{D}(n, m) u_{k}(n), \tag{14.34}
\end{equation*}
$$

with the vertex-frequency kernel given by

$$
\begin{equation*}
\mathcal{H}_{m, k}(n)=h_{m}(n) u_{k}(n)=P_{D}(n, m) u_{k}(n) . \tag{14.35}
\end{equation*}
$$

This allows us to arrive at the matrix form of the LGFT, given by

$$
\begin{equation*}
\mathbf{S}=\mathbf{U}^{T}\left(\mathbf{P}_{D} \circ[\mathbf{x}, \mathbf{x}, \ldots, \mathbf{x}]\right), \tag{14.36}
\end{equation*}
$$

where $[\mathbf{x}, \mathbf{x}, \ldots, \mathbf{x}]$ is an $N \times N$ matrix, the columns of which are the signal vector, $\mathbf{x}$.

For a rectangular function $g(d)=1$, for $d<D$, the LGFT can be calculated recursively with respect to the window width, $D$, as

$$
\begin{equation*}
\mathbf{S}_{D}=\mathbf{S}_{D-1}+\mathbf{U}^{T}\left(\mathbf{A}_{D-1} \circ[\mathbf{x}, \mathbf{x}, \ldots, \mathbf{x}]\right) \tag{14.37}
\end{equation*}
$$

Example 63: Consider the local vertex-frequency representation of the signal from Figure 8.1, using vertex domain defined windows. The localization kernels, $\mathcal{H}_{m, k}(n)=h_{m}(n) u_{k}(n)$, are shown in Figure 8.12 for two vertices and two spectral indices. Observe that for the spectral index $k=0$, the localization kernel is proportional to the localization function $h_{m}(n)$, given in Figures 8.12(a) and (c) for the vertices $m=34$ and $m=78$. Frequency modulated forms of these localization functions are shown in Figures 8.12(b) and (d), for the same vertices and $k=20$.


$$
\mathcal{H}_{78,0}(n)=h_{78}(n) u_{0}(n) \sim h_{78}(n)
$$


$\mathcal{H}_{34,20}(n)=h_{34}(n) u_{20}(n)$

(d)
$\mathcal{H}_{78,20}(n)=h_{78}(n) u_{20}(n)$

$\mathcal{H}_{34,0}(n)=h_{34}(n) u_{0}(n) \sim h_{34}(n)$


Figure 8.12: Localization kernels for vertex-frequency analysis, $\mathcal{H}_{m, k}(n)=$ $h_{m}(n) u_{k}(n)$, for the case of vertex domain defined windows in the local graph Fourier transform, $S(m, k)=\sum_{n=0}^{N-1} x(n) \mathcal{H}_{m, k}(n)$. (a) Localization kernel $\mathcal{H}_{34,0}(n)=$ $h_{34}(n) u_{0}(n) \sim h_{34}(n)$, for a constant eigenvector, $u_{0}(n)=1 / \sqrt{N}$, centered at the vertex $m=34$. (b) The same localization kernel as in (a), but centered at the vertex $m=78$. (c) Localization kernel, $\mathcal{H}_{34,20}(n)=h_{34}(n) u_{20}(n)$, centered at the vertex $m=35$ and frequency shifted by $u_{20}(n)$. Observe the variations in kernel amplitude, which indicate a modulation of the localization window, $h_{m}(n)$. (d) The same localization kernel as in (c), but centered at the vertex $m=78$. (e) Three-dimensional representation of the kernel $\mathcal{H}_{34,0}(n)=h_{34}(n) u_{0}(n)$. (f) Three-dimensional representation of the kernel $\mathcal{H}_{78,0}(n)=h_{78}(n) u_{0}(n)$.


Figure 8.13: Local vertex-frequency spectrum calculated using the LGFT and vertex neighborhood windows, as in (14.35). This representation immediately shows that the graph signal consists of three components located at spectral indices $k=8$, $k=66$, and $k=27$, with the corresponding vertex indices in their respective vertex subsets $\mathcal{V}_{1}, \mathcal{V}_{2}$, and $\mathcal{V}_{3}$, where $\mathcal{V}_{1} \cup \mathcal{V}_{2} \cup \mathcal{V}_{3}=\mathcal{V}$. The marginal properties are also given in the panels to the right and below the vertex-frequency representation, and they differ from the ideal ones given respectively by $|x(n)|^{2}$ and $|X(k)|^{2}$.

A vertex domain window is next used to analyze the graph signal from Figure 8.1. The vertex-frequency representation, $S(n, k)$, obtained with the LGFT and the vertex domain localization window is given in Figure 8.13. Again, we can observe three constituent graph signal components in three distinct vertex regions. The marginals of $S(n, k)$ are also shown in the right and bottom panels.

Remark 77: Directed graphs. The vertex neighborhood, as a set of vertices that can be reached from the considered vertex by a walk whose length is at most $D$, may be also defined on directed graphs. In this case, this approach corresponds to one-sided windows in classical signal analysis.

If we want to define two-sided window, then we should also include all vertices from which we can reach the considered vertex by walk whose length is at most $D$. This means that for a directed graph we should assume that vertices with distance $d_{m n}=1$ from the considered vertex $m$ are the vertices from which we can reach vertex $m$ with walk of length 1. In this case $\mathbf{A}_{1}=\mathbf{A}+\mathbf{A}^{T}$ where addition is logical operation (Boolean OR). The matrix $\mathbf{A}_{2}$ is

$$
\mathbf{A}_{2}=\left(\mathbf{A} \odot \mathbf{A}+\mathbf{A}^{T} \odot \mathbf{A}^{T}\right) \circ(\mathbf{1}-\mathbf{I}) \circ\left(\mathbf{1}-\mathbf{A}_{1}\right)
$$

This procedure could be continued for walks up to the desired maximal length $D$.

For a circular directed graph in this way, we will get the classical STFT with symmetric window.

## Window Parameter Optimization

The concentration of local vertex spectrum representation can be measured using the normalized one-norm (Stanković, 2001), as

$$
\begin{equation*}
\mathcal{M}=\frac{1}{F} \sum_{m=0}^{N-1} \sum_{k=0}^{N-1}|S(m, k)|=\frac{1}{F}\|\mathbf{S}\|_{1}, \tag{14.38}
\end{equation*}
$$

where

$$
F=\|\mathbf{S}\|_{F}=\sqrt{\sum_{m=0}^{N-1} \sum_{k=0}^{N-1}|S(m, k)|^{2}}
$$

is the Frobenius norm of matrix $\mathbf{S}$. Alternatively, any other norm $\|\mathbf{S}\|_{p}^{p}$, with $0 \leq p \leq 1$ can be used instead of $\|\mathbf{S}\|_{1}$. Recall that norms with $p$ close to 0 are noise sensitive, while the norm with $p=1$ is the only convex norm, which hence allows for gradient based optimization (Stanković, 2001).

Example 64: The concentration measure, $\mathcal{M}(\tau)=\|\mathbf{S}\|_{1} /\|\mathbf{S}\|_{F}$, for the signal from Figure 8.1, the window given in (14.2), and for various $\tau$ is shown in Figure 8.14, along with the optimal vertex frequency representation. This representation is similar to that shown in Figure 8.3, where an empirical value of $\tau=3$ was used, with the same localization window and kernel form.


Figure 8.14: Principle of the optimization of localization window. (a) Measure of the concentration of graph spectrogram for a varying spectral domain window parameter $\tau$. (b) The corresponding optimal vertex-frequency representation, calculated with $\tau=7$, together with its marginals.

The optimal $\tau$ can be obtained in only a few steps through the iteration

$$
\tau_{k}=\tau_{k-1}-\alpha\left(\mathcal{M}\left(\tau_{k-1}\right)-\mathcal{M}\left(\tau_{k-2}\right)\right)
$$

with $\alpha$ a step-size parameter.

The optimization of parameter $\tau$ can also be achieved through graph uncertainty principle based techniques (Agaskar and Lu, 2013; Tsitsvero et al., 2016).

### 14.2 Inversion of the LGFT

The inversion relation of the LGFT, calculated using any of the presented localization (window) forms, will next be considered in a unified way; the two approaches for the LGFT inversion here are: (i) inversion by summation of LGFT and (ii) kernel based inversion.

## Inversion by the Summation of the LGFT

The reconstruction of a graph signal, $x(n)$, from its local spectrum, $S(m, k)$, can be performed through an inverse GFT of (14.5), based on the graph windowed signal

$$
\begin{equation*}
x(n) h_{m}(n)=\sum_{k=0}^{N-1} S(m, k) u_{k}(n) \tag{14.39}
\end{equation*}
$$

followed by a summation over all vertices, $m$, to yield

$$
\begin{equation*}
x(n)=\frac{1}{\sum_{m=0}^{N-1} h_{m}(n)} \sum_{m=0}^{N-1} \sum_{k=0}^{N-1} S(m, k) u_{k}(n) \tag{14.40}
\end{equation*}
$$

Remark 78: If the windows, $h_{m}(n)$, for every vertex, $n$, satisfy the condition

$$
\sum_{m=0}^{N-1} h_{m}(n)=1
$$

then the reconstruction does not depend on the vertex index, $n$, or in other words such reconstruction is vertex independent. This becomes clear from

$$
\begin{equation*}
x(n)=\sum_{m=0}^{N-1} \sum_{k=0}^{N-1} S(m, k) u_{k}(n)=\sum_{k=0}^{N-1} X(k) u_{k}(n) \tag{14.41}
\end{equation*}
$$

where

$$
X(k)=\sum_{m=0}^{N-1} S(m, k)
$$

is a projection of the LGFT onto the spectral index axis. For windows obtained using the generalized graph shift in (14.33), this conditions is always satisfied since $H(0)=1$.

The condition $\sum_{m=0}^{N-1} h_{m}(n)=1$ can be enforced by normalizing the elements of the matrix $\mathbf{A}_{d}, d=1,2, \ldots, D-1$ in (14.33), prior to the calculation of matrix $\mathbf{P}_{D}$, in such a way that the sum of each of its columns is equal to 1 , which allows us to arrive at

$$
\sum_{m=0}^{N-1} h_{m}(n)=\sum_{m=0}^{N-1} P_{D}(n, m)=\sum_{d=1}^{D-1} g(d)=\text { const }
$$

In general, the local vertex spectrum, $S(m, k)$, can also be calculated over a reduced set of vertices, $m \in \mathcal{M} \subset \mathcal{V}$. In this case, the summation over $m$ in the reconstruction formula should be executed over only the vertices $m \in \mathcal{M}$, while a vertex-independent reconstruction is achieved if $\sum_{m \in \mathcal{M}} h_{m}(n)=1$.

## Inversion of the LGFT with Band-Pass Functions

For the LGFT, defined in (14.18) as $\mathbf{s}_{k}=\sum_{p=0}^{M-1} h_{p, k} \mathbf{L}^{p} \mathbf{x}$, the inversion is obtained by a summation over all spectral index shifts, $k=0,1, \ldots, K$, that is

$$
\begin{equation*}
\sum_{k=0}^{K} \mathbf{s}_{k}=\sum_{k=0}^{K} \sum_{p=0}^{N-1} h_{p, k} \mathbf{L}^{p} \mathbf{x}=\sum_{k=0}^{K} H_{k}(\mathbf{L}) \mathbf{x}=\mathbf{x} \tag{14.42}
\end{equation*}
$$

if $\sum_{k=0}^{K} H_{k}(\mathbf{L})=\mathbf{I}$. This condition is equivalent to the following spectral domain form

$$
\begin{equation*}
\sum_{k=0}^{K} H_{k}(\boldsymbol{\Lambda})=\mathbf{I} \tag{14.43}
\end{equation*}
$$

since $\mathbf{U} \sum_{k=0}^{K} H_{k}(\boldsymbol{\Lambda}) \mathbf{U}^{T}=\mathbf{I}$ and $\mathbf{U}^{T} \mathbf{U}=\mathbf{I}$. The condition in (14.43) is used to define the transfer functions in Figure 8.4.

## Kernel-Based Inversion

Another approach to the inversion of the local vertex spectrum, $S(m, k)$, follows the Gabor expansion framework (Stanković et al., 2014), whereby
the local vertex spectrum, $S(m, k)$, is projected back to the vertexfrequency localized kernels, $\mathcal{H}_{m, k}(n)$. The inversion for two forms of the LGFT, defined as in (14.6) and (14.17), will be analyzed.
(a) For the LGFT defined in (14.6), the sum of all of its projections to the localized kernels, $\mathcal{H}_{m, k}(n)$, is

$$
\begin{align*}
\sum_{m=0}^{N-1} & \sum_{k=0}^{N-1} S(m, k) \mathcal{H}_{m, k}(n) \\
& =\sum_{m=0}^{N-1}\left(\sum_{k=0}^{N-1} S(m, k) h_{m}(n) u_{k}(n)\right) \\
& =\sum_{m=0}^{N-1}\left(\sum_{i=0}^{N-1} \operatorname{IGFT}_{k \rightarrow i}\{S(m, k)\} \underset{k \rightarrow i}{\operatorname{IGFT}}\left\{h_{m}(n) u_{k}(n)\right\}\right) \\
& =\sum_{m=0}^{N-1} \sum_{i=0}^{N-1}\left[x(i) h_{m}(i)\right]\left[h_{m}(n) \delta(n-i)\right] \\
& =\sum_{m=0}^{N-1} x(n) h_{m}^{2}(n)=x(n) \sum_{m=0}^{N-1} h_{m}^{2}(n), \tag{14.44}
\end{align*}
$$

where IGFT denotes the inverse GFT transform. Parseval's theorem for graph signals

$$
\sum_{n=0}^{N-1} x(n) y(n)=\sum_{k=0}^{N-1} X(k) Y(k)
$$

was used in the derivation. In this form of the LGFT all possible spectral shifts, $k=0,1, \ldots, N-1$, are used.

The inversion formula for the local vertex spectrum, $S(m, k)$, which yields the original graph signal, $x(n)$, then becomes

$$
\begin{equation*}
x(n)=\frac{1}{\sum_{m=0}^{N-1} h_{m}^{2}(n)} \sum_{m=0}^{N-1} \sum_{k=0}^{N-1} S(m, k) \mathcal{H}_{m, k}(n) \tag{14.45}
\end{equation*}
$$

Remark 79: This kind of kernel-based inversion is vertex-invariant if the sum over all vertices, $m$, is invariant with respect to $n$ and is equal to 1 , that is

$$
\begin{equation*}
\sum_{m=0}^{N-1} h_{m}^{2}(n)=1 \tag{14.46}
\end{equation*}
$$

If the LGFT, $S(m, k)$, is calculated over a reduced set of vertices, $m \in \mathcal{M} \subset \mathcal{V}$, then the vertex independent reconstruction condition becomes $\sum_{m \in \mathcal{M}} h_{m}^{2}(n)=1$.
(b) For the LGFT with spectral shifted spectral windows, defined in (14.17), the kernel based inversion is of the form

$$
\begin{equation*}
x(n)=\sum_{m=0}^{N-1} \sum_{k=0}^{K} S(m, k) \mathcal{H}_{m, k}(n) \tag{14.47}
\end{equation*}
$$

if the following condition

$$
\begin{equation*}
\sum_{k=0}^{K} H_{k}^{2}\left(\lambda_{p}\right)=1 \tag{14.48}
\end{equation*}
$$

is satisfied for all $\lambda_{p}, p=0,1,2, \ldots, N-1$.
The inversion formula in (14.47), with condition (14.48), follows from

$$
\begin{align*}
& \sum_{m=0}^{N-1} \sum_{k=0}^{K} S(m, k) \mathcal{H}_{m, k}(n) \\
& \quad=\sum_{m=0}^{N-1} \sum_{k=0}^{K} \sum_{p=0}^{N-1} X(p) H_{k}\left(\lambda_{p}\right) u_{p}(m) \sum_{l=0}^{N-1} H_{k}\left(\lambda_{l}\right) u_{l}(m) u_{l}(n) . \tag{14.49}
\end{align*}
$$

Since $\sum_{m=0}^{N-1} u_{p}(m) u_{l}(m)=\delta(p-l)$, the last expression reduces to the graph signal, $x(n)$,

$$
\begin{equation*}
\sum_{k=0}^{K} \sum_{p=0}^{N-1} X(p) H_{k}\left(\lambda_{p}\right) H_{k}\left(\lambda_{p}\right) u_{p}(n)=x(n) \tag{14.50}
\end{equation*}
$$

if the transfer functions, $H_{k}\left(\lambda_{p}\right), k=0,1, \ldots, K$, satisfy the condition in (14.48) for all $\lambda_{p}$.

## Vertex-Varying Filtering

Filtering in the vertex-frequency domain may be implemented using a vertex-frequency support function, $B(m, k)$. The filtered LGFT is then given by

$$
S_{f}(m, k)=S(m, k) B(m, k),
$$



(c)

Figure 8.15: Vertex-varying filtering of a graph signal. (a) The original graph signal, $x(n)$, from Figure 8.1 (d). (b) The graph signal, $x(n)$, corrupted by an additive white Gaussian noise, at $\mathrm{SNR}_{i n}=5.3 \mathrm{~dB}$. (c) The graph signal, $x_{f}(n)$, after vertex-varying filtering based on thresholding of the LGFT of noisy graph signal, $S(m, k)$, with the final signal-to-noise ratio $\mathrm{SNR}_{\text {out }}=10.36 \mathrm{~dB}$.
and the filtered signal, $x_{f}(n)$, is obtained by the inversion of $S_{f}(m, k)$ using the above mentioned inversion methods. The filtering support function, $B(m, k)$, can be obtained, for example, by thresholding noisy values of the local vertex spectrum, $S(m, k)$.
Example 65: Consider the graph signal, $x(n)$, from Figure 8.1(d), also shown in Figure 8.15(a), and its version corrupted by an additive white Gaussian noise, at the signal-to-noise ratio of $\mathrm{SNR}_{i n}=5.3 \mathrm{~dB}$, given in Figure 8.15(b). The LGFT, $S(m, k)$ of the noisy graph signal is calculated according to (14.17), using shifted bandpass spectral transfer functions, $H_{k}\left(\lambda_{p}\right), k=0,1, \ldots, K, p=0,1, \ldots, N-1$, given by (14.20) without squares $\left(H_{k}\left(\lambda_{p}\right) \rightarrow H_{k}^{2}\left(\lambda_{p}\right)\right)$, which allows $\sum_{k=0}^{K} H_{k}^{2}\left(\lambda_{p}\right)=1$ to
hold, instead of $\sum_{k=0}^{K} H_{k}\left(\lambda_{p}\right)=1$. In this way, the condition for the inversion (14.48) is satisfied. The transfer functions, $H_{k}\left(\lambda_{p}\right)$, otherwise correspond to those shown in Figure $8.4(\mathrm{~b})$ with $K=25$.

The vertex-varying filtering is performed using $S_{f}(m, k)=S(m, k)$. $B(m, k)$ for $m=0,1, \ldots, N-1, k=0,1, \ldots, K$, with a simple thresholding-based filtering support function

$$
B(m, k)=\left\{\begin{array}{l}
0, \text { for }|S(m, k)|<T \\
1, \text { otherwise }
\end{array}\right.
$$

$m=0,1, \ldots, N-1, k=0,1, \ldots, K$, with the threshold $T=0.09$ set empirically. The output graph signal, $x_{f}(n)$, is obtained using the inversion relation in (14.47) for the filtered LGFT, $S_{f}(m, k)$, and shown in Figure 8.15(c). The achieved output SNR was $\mathrm{SNR}_{\text {out }}=10.36 \mathrm{~dB}$.

If the signal is filtered using the graph Wiener filter, as in Section 13.3, with the estimated noise level $\sigma_{\varepsilon}=0.12$, and the available noisy signal, $H\left(\lambda_{k}\right)=|X(k)|^{2} /\left(|X(k)|^{2}+\sigma_{\varepsilon}^{2}\right)$, with $x_{f}(n)=\operatorname{IGFT}\left\{X(k) H\left(\lambda_{k}\right)\right\}$, then the output SNR is $\mathrm{SNR}_{\text {out }}=7.80 \mathrm{~dB}$. This value is lower that in the vertex-varying filtering case. If we knew the signal without noise and used it in the definition of the Wiener filter, the output SNR would have been improved to 15.78 dB .

### 14.3 Uncertainty Principle for Graph Signals

In the classical signal analysis, the purpose of a window function is to enhance signal localization in the joint time-frequency domain. However, the uncertainty principle prevents an ideal localization in both time and frequency. Various forms of the uncertainty principle in the signal analysis have been defined, with surveys in Ricaud and Torrésani (2014) and Perraudin et al. (2018). Various forms of the uncertainty principle in graph signal processing are studied in Erb (2019).

These forms are closely related to the concentration measures in timefrequency distributions; for a review see Stanković (2001). While the common uncertainty principle form in time-frequency analysis (whose quantum mechanical form is called the Robertson-Schrödinger inequality) establishes the lower bound for the product of effective signal
widths (variances) in the time and the frequency domain (Cohen, 1995; Stankovic, 1997), here we will use a form of the sparsity support measure (Ricaud and Torrésani, 2014; Stanković, 2001) as the one which clearly and in a simple way shows a significant difference in both classical Fourier based analysis and graph signal transforms with respect to the expected concentration in the joint vertex-frequency domain.

In classical signal analysis, the purpose of a window function is to enhance signal localization in the joint time-frequency domain. However, the uncertainty principle prevents the ideal localization in both time and frequency. Indeed, in the classical DFT analysis the uncertainty principle states that

$$
\begin{equation*}
\|\mathbf{x}\|_{0}\|\mathbf{X}\|_{0} \geq N \tag{14.51}
\end{equation*}
$$

or in other words, that the product of the number of nonzero signal values, $\|\mathbf{x}\|_{0}$, and the number of its nonzero DFT coefficients, $\|\mathbf{X}\|_{0}$, is greater or equal than the total number of signal samples $N$; they cannot simultaneously assume small values.

To arrive at the uncertainty principle for graph signals, consider a graph signal, $\mathbf{x}$, and its spectral transform, $\mathbf{X}$, in a domain of orthonormal basis functions, $u_{k}(n)$. Then, the uncertainty principle states that Tsitsvero et al. (2016), Agaskar and Lu (2013), Elad and Bruckstein (2002), and Perraudin et al. (2018)

$$
\begin{equation*}
\|\mathbf{x}\|_{0}\|\mathbf{X}\|_{0} \geq \frac{1}{\max _{k, m}\left\{\left|u_{k}(m)\right|^{2}\right\}} \tag{14.52}
\end{equation*}
$$

This form of the uncertainty principle is generic, and indeed for the basis functions $u_{k}(n)=\frac{1}{\sqrt{N}} \exp (j 2 \pi n k / N)$, the standard DFT uncertainty principle form in (14.51) follows. A simple derivation of the support uncertainty principle shall be given in Section 14.5 (Stanković, 2020).
Remark 80: Note, however, that in graph signal processing, the eigenvectors/basis functions can assume quite different forms than in the standard DFT case. For example, when one vertex is loosely connected with other vertices, then $\max \left\{\left|u_{k}(m)\right|^{2}\right\} \rightarrow 1$ and even $\|\mathbf{x}\|_{0}\|\mathbf{X}\|_{0} \geq 1$ is possible for the uncertainty condition in (14.52). This means that, unlike the classical Fourier transform-based time and frequency domains, a graph signal can be well localized in both the vertex and the spectral domains.

Example 66: For the graph shown in Figure 8.1, we have

$$
\max _{k, m}\left\{\left|u_{k}(m)\right|^{2}\right\}=0.8713
$$

which indicates that even $\|\mathbf{x}\|_{0}\|\mathbf{X}\|_{0} \geq 1.1478$ is possible. In other words, a graph signal for which the number of nonzero samples, $x(n)$, in the vertex domain is just two, will not violate the uncertainty principle even if it has just one nonzero GFT coefficient, $X(k)$.

### 14.4 Graph Spectrogram and Frames

Based on (14.5), the graph spectrogram can be defined as

$$
\begin{equation*}
|S(m, k)|^{2}=\left|\sum_{n=0}^{N-1} x(n) h_{m}(n) u_{k}(n)\right|^{2} . \tag{14.53}
\end{equation*}
$$

Then, according to Parseval's theorem, the vertex marginal property, which is a projection of $|S(m, k)|^{2}$ onto the vertex index axis, is given by

$$
\begin{aligned}
\sum_{k=0}^{N-1}|S(m, k)|^{2} & =\sum_{k=0}^{N-1} S(m, k) \sum_{n=0}^{N-1} x(n) h_{m}(n) u_{k}(n) \\
& =\sum_{n=0}^{N-1}\left|x(n) h_{m}(n)\right|^{2},
\end{aligned}
$$

which would be equal to the signal power, $|x(m)|^{2}$, at the vertex $m$, if $h_{m}(n)=\delta(m-n)$. Since this is not the case, the vertex marginal property of the graph spectrogram is equal to the power of the graph signal in hand, smoothed by the window, $h_{m}(n)$.
Energy of graph spectrogram. For the total energy of graph spectrogram, we consequently have

$$
\begin{equation*}
\sum_{m=0}^{N-1} \sum_{k=0}^{N-1}|S(m, k)|^{2}=\sum_{n=0}^{N-1}\left(|x(n)|^{2} \sum_{m=0}^{N-1}\left|h_{m}(n)\right|^{2}\right) . \tag{14.54}
\end{equation*}
$$

If $\sum_{m=0}^{N-1}\left|h_{m}(n)\right|^{2}=1$ for all $n$, then the spectrogram on the graph is energy unbiased (statistically consistent with respect to the energy),
that is

$$
\begin{equation*}
\sum_{m=0}^{N-1} \sum_{k=0}^{N-1}|S(m, k)|^{2}=\sum_{n=0}^{N-1}|x(n)|^{2}=\|\mathbf{x}\|^{2}=E_{x} \tag{14.55}
\end{equation*}
$$

The LGFT viewed as a frame. A set of functions, $S(m, k)$, is called a frame for the expansion of a graph signal, $\mathbf{x}$, if

$$
A\|\mathbf{x}\|^{2} \leq \sum_{m=0}^{N-1}|S(m, k)|^{2} \leq B\|\mathbf{x}\|^{2}
$$

where $A$ and $B$ are positive constants. If $A=B$, the frame is termed Parseval's tight frame and the signal can be recovered as

$$
x(n)=\frac{1}{A} \sum_{m=0}^{N-1} \sum_{k=0}^{N-1} S(m, k) h_{m}(n) u_{k}(n)
$$

The constants $A$ and $B$ govern the numerical stability of recovering the original signal x from the coefficients $S(m, k)$.

The conditions for two forms of the LGFT, defined as in (14.6) and (14.17), to represent frames will be analyzed next.
(a) The LGFT, defined as in (14.6), is a frame, since in this case Parseval's theorem holds (Behjat et al., 2016; Girault, 2015; Hammond et al., 2011a; Sakiyama and Tanaka, 2014), that is

$$
\begin{equation*}
\sum_{m=0}^{N-1}\left|h_{m}(n)\right|^{2}=\sum_{k=0}^{N-1}|H(k)|^{2}\left|u_{k}(n)\right|^{2} \tag{14.56}
\end{equation*}
$$

which allows us to write

$$
\begin{equation*}
\frac{1}{N} H^{2}(0) \leq \sum_{m=0}^{N-1}\left|h_{m}(n)\right|^{2} \leq \max _{n, k}\left|u_{k}(n)\right|^{2} \sum_{k=0}^{N-1}|H(k)|^{2}=\gamma^{2} E_{h} \tag{14.57}
\end{equation*}
$$

where $\gamma=\max _{n, k}\left|u_{k}(n)\right|$ and

$$
E_{h}=\sum_{k=0}^{N-1}|H(k)|^{2}
$$

By multiplying both sides of the above inequalities by $\|\mathbf{x}\|^{2}$, we arrive at

$$
\begin{equation*}
\frac{1}{N} H^{2}(0)\|\mathbf{x}\|^{2} \leq \sum_{m=0}^{N-1} \sum_{k=0}^{N-1}|S(m, k)|^{2} \leq\|\mathbf{x}\|^{2} \gamma^{2} E_{h} \tag{14.58}
\end{equation*}
$$

A frame is termed a tight frame if the equality in (14.57) holds, that is, if

$$
\sum_{m=0}^{N-1}\left|h_{m}(n)\right|^{2}=1,
$$

which is the same condition as in (14.46).
(b) The LGFT defined in (14.17) is a tight frame if

$$
\begin{equation*}
\sum_{k=0}^{K} \sum_{m=0}^{N-1}|S(m, k)|^{2}=\sum_{k=0}^{K} \sum_{p=0}^{N-1}\left|X(p) H_{k}\left(\lambda_{p}\right)\right|^{2}=E_{x}, \tag{14.59}
\end{equation*}
$$

where Parseval's theorem for the $S(m, k)$ as the GFT of $X(p)$. $H_{k}\left(\lambda_{p}\right)$ was used to yield

$$
\sum_{m=0}^{N-1}|S(m, k)|^{2}=\sum_{p=0}^{N-1}\left|X(p) H_{k}\left(\lambda_{p}\right)\right|^{2} .
$$

This means that the LGFT in (14.17) is a tight frame if

$$
\sum_{k=0}^{K}\left|H_{k}\left(\lambda_{p}\right)\right|^{2}=1 \quad \text { for } p=0,1, \ldots, N-1 .
$$

This condition is used to define transfer functions in Figures 8.4(b) and (c).
From (14.59), it is straightforward to conclude that the graph spectrogram energy is bounded with

$$
\begin{equation*}
A E_{x} \leq \sum_{k=0}^{K} \sum_{m=0}^{N-1}|S(m, k)|^{2} \leq B E_{x}, \tag{14.60}
\end{equation*}
$$

where $A$ and $B$ are respectively the minimum and the maximum of value of

$$
g\left(\lambda_{p}\right)=\sum_{k=0}^{K}\left|H_{k}\left(\lambda_{p}\right)\right|^{2}
$$

## Graph Wavelet Transform Inversion

The wavelet inversion formula

$$
\begin{equation*}
x(n)=\sum_{n=0}^{N-1} \sum_{i=0}^{K} \psi\left(n, s_{i}\right) W\left(n, s_{i}\right) \tag{14.61}
\end{equation*}
$$

can be derived in the same way and under the same condition as in (14.47)-(14.48), where a set of discrete scales for the wavelet calculation, denoted by $s \in\left\{s_{1}, s_{1}, \ldots, s_{K}\right\}$, is assumed, and $\psi\left(n, s_{0}\right)$ is used as a notation for the scale function, $\phi(n)$, whose spectral transfer function is $G(\lambda)$, as explained in Remark 74. In the same way as in the LGFT case, it can be shown that the wavelet transform represents a frame with

$$
\begin{equation*}
A\|\mathbf{x}\|^{2} \leq \sum_{n=0}^{N-1} \sum_{i=0}^{K}\left|W\left(n, s_{i}\right)\right|^{2} \leq B\|\mathbf{x}\|^{2}, \tag{14.62}
\end{equation*}
$$

where (Leonardi and Van De Ville (2013), Hammond et al. (2019), and Behjat and Van De Ville (2019))

$$
\begin{aligned}
& A=\min _{0 \leq \lambda \leq \lambda_{\max }} g(\lambda), \\
& B=\max _{0 \leq \lambda \leq \lambda_{\max }} g(\lambda),
\end{aligned}
$$

and the function $g(\lambda)$ is defined by

$$
g(\lambda)=\sum_{i=1}^{K} H^{2}\left(s_{i} \lambda\right)+G^{2}(\lambda)
$$

The low-pass scale function, $G(\lambda)$, is added in the reconstruction formula, since all $H\left(s_{i} \lambda\right)=0$ for $\lambda=0$, as explained in Example 61 and Remark 74. It should be mentioned that the spectral functions of the wavelet transform, $H\left(s_{i} \lambda\right)$, form Parseval's frame if

$$
g(\lambda)=1 .
$$

Since the number of wavelet transform coefficients, $W\left(n, s_{i}\right)$, for each $n$ and $i$, is greater than the number of signal samples, $N$, this representation is redundant, and this redundancy allows us to implement the transform through a fast algorithm, rather than using the explicit
computation of all wavelet coefficients (Behjat and Van De Ville, 2019; Hammond et al., 2019). Indeed, for large graphs, it can be computationally too complex to compute the full eigendecomposition of the graph Laplacian. A common way to avoid this computational burden is to use a polynomial approximation schemes for $H\left(s_{i} \lambda\right), i=1,2, \ldots, K$, and $G(\lambda)$. One such approach is the truncated Chebyshev polynomial approximation method which is based on the application of the continuous spectral window functions with Chebyshev polynomials, which admit order-recursive calculation (see Section 9.5 and Example 9.5). If, for a given scale, $s_{i}$, the wavelet function is approximated by a polynomial in the Laplacian, $P_{i}(\mathbf{L})$, then the wavelet transform can be efficiently calculated using

$$
\begin{equation*}
\mathbf{w}_{i}=P_{i}(\mathbf{L}) \mathbf{x} \tag{14.63}
\end{equation*}
$$

where $\mathbf{w}_{i}$ a column vector with elements $W\left(m, s_{i}\right), m=0,1, \ldots$, $N-1$. Note that this form corresponds to the LGFT form in (14.18).

### 14.5 Vertex-Frequency Energy Distributions

Like in time-frequency analysis, the distribution of graph signal energy, as a function of the vertex and spectral indices, is an alternative way to approach vertex-frequency analysis without localization windows. A graph form of the Rihaczek distribution is used as the basic distribution to introduce the concepts of vertex-frequency domain energy parameters, such as the local smoothness and marginal properties. The graph Rihaczek distribution is then used to derive the support uncertainty principle and to define a class of reduced interference vertexfrequency energy distributions which satisfy the graph signal marginal properties.

The energy of a general signal is usually defined as

$$
E=\sum_{n=0}^{N-1} x^{2}(n)=\sum_{n=0}^{N-1} x(n) \sum_{k=0}^{N-1} X(k) u_{k}(n)
$$

This expression can be rearranged into

$$
\begin{equation*}
E=\sum_{n=0}^{N-1} \sum_{k=0}^{N-1} x(n) X(k) u_{k}(n)=\sum_{n=0}^{N-1} \sum_{k=0}^{N-1} E(n, k) \tag{14.64}
\end{equation*}
$$

where for each vertex, the vertex-frequency energy distribution, $E(n, k)$, is defined by Stanković et al. (2018b, 2019b)

$$
\begin{equation*}
E(n, k)=x(n) X(k) u_{k}(n)=\sum_{m=0}^{N-1} x(n) x(m) u_{k}(m) u_{k}(n) . \tag{14.65}
\end{equation*}
$$

Remark 81: The definition in (14.65) corresponds to the Rihaczek distribution in classical time-frequency analysis (Boashash, 2015; Cohen, 1995; Stanković et al., 2014). Observe that based on the Rihaczek distribution and the expression in (14.65), we may obtain a vertex-frequency representation even without a localization window. This very important property is also the main advantage (along with the concentration improvement) of classical time-frequency distributions with respect to the spectrogram and STFT based time-frequency representations.

The marginal properties of the vertex-frequency energy distribution, $E(n, k)$, are defined as its projections onto the spectral index axis, $k$, and the vertex index axis, $n$, to give

$$
\sum_{n=0}^{N-1} E(n, k)=|X(k)|^{2} \quad \text { and } \quad \sum_{k=0}^{N-1} E(n, k)=x^{2}(n),
$$

which correspond respectively to the squared spectra, $|X(k)|^{2}$, and the signal power, $x^{2}(n)$, of the graph signal, $x(n)$.
Example 67: Figure 8.16 shows the vertex-frequency distribution, $E(n, k)$, of the graph signal from Figure 8.1, together with its marginal properties. The marginal properties are satisfied up to the computer precision. Observe also that the localization of energy is better than in the cases obtained with the localization windows in Figures 8.3, 8.13, and 8.14. Importantly, the distribution, $E(n, k)$, does not employ a localization window.

## Smoothness Index and Local Smoothness

The smoothness index, $l$, in graph signal processing plays the role of frequency, $\omega$, in classical spectral analysis. For a graph signal, $\mathbf{x}$, the smoothness index is defined as the Rayleigh quotient of the matrix $\mathbf{L}$


Figure 8.16: Vertex-frequency energy distribution for the graph signal whose vertexfrequency representation is given in Figure 8.3. No localization window was used here.
and vector $\mathbf{x}$, that is (see Section 4.2, Part I)

$$
\begin{equation*}
l=\frac{\mathbf{x}^{T} \mathbf{L} \mathbf{x}}{\mathbf{x}^{T} \mathbf{x}} \geq 0 \tag{14.66}
\end{equation*}
$$

Remark 82: The expression in (28.9) indicates that the smoothness index can be considered as a measure of the rate of change of a graph signal. Faster changing signals (corresponding to high-frequency signals) have larger values of the smoothness index. The maximally smooth graph signal is then a constant signal, $x(n)=c$, for which the smoothness index is $l=0$.

In the mathematics literature, the inverse of the smoothness index is known as the curvature (curvature $\sim 1 / l$ ). While larger values of the smoothness index correspond to graph signals with larger rates of
change (less smooth graph signals), the larger values of curvature would indicate smoother graph signals.

Notice that the smoothness index for an eigenvector, $\mathbf{u}_{k}$, of the graph Laplacian, $\mathbf{L}$, is equal to its corresponding eigenvalue, $\lambda_{k}$, that is

$$
\begin{equation*}
\frac{\mathbf{u}_{k}^{T} \mathbf{L} \mathbf{u}_{k}}{\mathbf{u}_{k}^{T} \mathbf{u}_{k}}=\lambda_{k} \tag{14.67}
\end{equation*}
$$

since by definition $\mathbf{L} \mathbf{u}_{k}=\lambda_{k} \mathbf{u}_{k}$.
Remark 83: If the above eigenvectors are the classical Fourier transform basis functions, then the smoothness index corresponds to the squared frequency of the considered basis function, $\lambda_{k} \sim \omega_{k}^{2}$, while the curvature corresponds to the squared period in harmonic signals.

This makes it possible to define the local smoothness index for a vertex $n, \lambda(n)$, in analogy with the standard instantaneous frequency, $\omega(t)$, at an instant $t$, as Daković et al. (2019)

$$
\begin{equation*}
\lambda(n)=\frac{\mathcal{L}_{x}(n)}{x(n)}, \tag{14.68}
\end{equation*}
$$

where it was assumed that $x(n) \neq 0$ and $\mathcal{L}_{x}(n)$ are the elements of the vector $\mathbf{L x}$.

The properties of the local smoothness include:

1. The local smoothness index, $\lambda(n)$, for a monocomponent signal

$$
x(n)=\alpha u_{k}(n),
$$

is vertex independent, and is equal to the global smoothness index, $\lambda_{k}$, since

$$
\mathcal{L}_{x}(n)=\alpha \mathcal{L}_{u_{k}}(n)=\alpha \lambda_{k} u_{k}(n) .
$$

In the standard time-domain signal analysis, this property means that the instantaneous frequency of a sinusoidal signal is equal to its global frequency.
2. Assume a piece-wise monocomponent signal

$$
x(n)=\alpha_{i} u_{k_{i}}(n) \quad \text { for } n \in \mathcal{V}_{i}, i=1,2, \ldots, M,
$$

where $\mathcal{V}_{i} \subset \mathcal{V}$ are the subsets of the vertices such that $\mathcal{V}_{i} \cap \mathcal{V}_{j}=\emptyset$ for $i \neq j, \mathcal{V}_{1} \cup \mathcal{V}_{2} \cup \cdots \cup \mathcal{V}_{M}=\mathcal{V}$, that is, every vertex belongs
to only one subset, $\mathcal{V}_{i}$. Given the monocomponent nature of this signal, within each subset, $\mathcal{V}_{i}$, the considered signal is proportional to the eigenvector, $u_{k_{i}}(n)$.
Then, for each interior vertex, $n \in \mathcal{V}_{i}$, i.e., a vertex whose neighborhood lies in the same set, $\mathcal{V}_{i}$, the local smoothness index is given by

$$
\begin{equation*}
\lambda(n)=\frac{\alpha_{i} \mathcal{L}_{u_{k_{i}}}(n)}{\alpha_{i} u_{k_{i}}(n)}=\lambda_{k_{i}} . \tag{14.69}
\end{equation*}
$$

3. An ideally concentrated vertex-frequency distribution (ideal distribution) can be defined as

$$
I(n, k) \sim|x(n)|^{2} \delta\left(\lambda_{k}-[\lambda(n)]\right)
$$

whereby it is assumed that the local smoothness index is rounded to the nearest eigenvalue.
This distribution can also be used as a local smoothness estimator, since for each vertex, $n$, the maximum of $I(n, k)$ is positioned at $\lambda_{k}=\lambda(n)$. An estimate of the spectral index at a vertex, $n$, denoted by $\hat{k}(n)$, is then obtained as

$$
\hat{k}(n)=\arg \max _{k}\{I(n, k)\},
$$

so that the estimated local smoothness index becomes $\hat{\lambda}(n)=$ $\lambda_{\hat{k}(n)}$. This type of estimator is widely used in classical timefrequency analysis (Boashash, 2015; Cohen, 1995; Stanković et al., 2014).
4. Local smoothness property. The vertex-frequency distribution, $E(n, k)$, satisfies the local smoothness property if

$$
\begin{equation*}
\frac{\sum_{k=0}^{N-1} \lambda_{k} E(n, k)}{\sum_{k=0}^{N-1} E(n, k)}=\lambda(n) . \tag{14.70}
\end{equation*}
$$

In that case, the centers of masses of the vertex-frequency distribution along the spectral index axis, $k$, should be exactly at $\lambda=\lambda(n)$, and can be used as an unbiased estimator of this graph signal parameter.


Figure 8.17: Local smoothness index, $\lambda(n)$, of the graph signal from Figure 8.1.

Example 68: The vertex-frequency distribution, defined by $E(n, k)=$ $x(n) X(k) u_{k}(n)$, satisfies the local smoothness property in (14.70), since

$$
\frac{\sum_{k=0}^{N-1} \lambda_{k} E(n, k)}{\sum_{k=0}^{N-1} E(n, k)}=\frac{\sum_{k=0}^{N-1} \lambda_{k} x(n) X(k) u_{k}(n)}{\sum_{k=0}^{N-1} x(n) X(k) u_{k}(n)}=\frac{\mathcal{L}_{x}(n)}{x(n)}=\lambda(n)
$$

The above relation follows from the fact that $\sum_{k=0}^{N-1} \lambda_{k} X(k) u_{k}(n)$ are the elements of the IGFT of $\lambda_{k} X(k)$. Upon employing the matrix form of the IGFT of $\boldsymbol{\Lambda} \mathbf{X}$, we have $\mathbf{U} \boldsymbol{\Lambda} \mathbf{X}=\mathbf{U} \boldsymbol{\Lambda}\left(\mathbf{U}^{T} \mathbf{U}\right) \mathbf{X}=\left(\mathbf{U} \boldsymbol{\Lambda} \mathbf{U}^{T}\right)(\mathbf{U X})=\mathbf{L x}$. With the notation, $\mathcal{L}_{x}(n)$, for the elements of $\mathbf{L x}$, we next obtain

$$
\sum_{k=0}^{N-1} \lambda_{k} X(k) u_{k}(n)=\mathcal{L}_{x}(n)
$$

The local smoothness index for the graph signal from Figure 8.1 is shown in Figure 8.17.

## Support Uncertainty Principle Derivation

From the energy condition for the Rihaczek distribution in (14.65) and (14.64), for the case of unit energy, we have

$$
\begin{equation*}
1 \leq \sum_{n=0}^{N-1} \sum_{k=0}^{N-1}|E(n, k)| \tag{14.71}
\end{equation*}
$$

Assume, as in Elad and Bruckstein (2002), that the support, $\mathbb{M}$, of the signal, $x(n)$, is $\mathbb{M}=\left\{n_{1}, n_{2}, \ldots, n_{M}\right\}$, meaning that $x(n) \neq 0$ for $n \in \mathbb{M}$ and $x(n)=0$ for $n \notin \mathbb{M}$, while the support of the graph Fourier transform, $X(k)$, is $\mathbb{K}=\left\{k_{1}, k_{2}, \ldots, k_{K}\right\}$, where $X(k) \neq 0$ for $k \in \mathbb{K}$ and $X(k)=0$ for $k \notin \mathbb{K}$. By definition, we can write

$$
\begin{equation*}
\|\mathbf{x}\|_{0}=\operatorname{card}\{\mathbb{M}\}=M \text { and }\|\mathbf{X}\|_{0}=\operatorname{card}\{\mathbb{K}\}=K \tag{14.72}
\end{equation*}
$$

Upon applying the Schwartz inequality to the square of (14.71), we have

$$
\begin{align*}
1 & =\left(\sum_{n \in \mathbb{M}} \sum_{k \in \mathbb{K}} E(n, k)\right)^{2} \leq\left(\sum_{n \in \mathbb{M}} \sum_{k \in \mathbb{K}}|x(n)||X(k)|\left|u_{k}(n)\right|\right)^{2} \\
& =\left(\sum _ { n \in \mathbb { M } } \sum _ { k \in \mathbb { K } } \left(\sqrt{\left.\left.\left|u_{k}(n)\right||x(n)|\right)\left(\sqrt{\left|u_{k}(n)\right|}|X(k)|\right)\right)^{2}}\right.\right.  \tag{14.73}\\
& \leq \sum_{n \in \mathbb{M}} \sum_{k \in \mathbb{K}}\left|u _ { k } ( n ) \left\|\left.x(n)\right|^{2} \sum_{n \in \mathbb{M}} \sum_{k \in \mathbb{K}}\left|u_{k}(n) \| X(k)\right|^{2}\right.\right.  \tag{14.74}\\
& \leq \max _{n, k}\left\{\left|u_{k}(n)\right|^{2}\right\} K M=\max _{n, k}\left\{\left|u_{k}(n)\right|^{2}\right\}\|\mathbf{x}\|_{0}\|\mathbf{X}\|_{0}, \tag{14.75}
\end{align*}
$$

from the unit energy of the graph signal, $\sum_{n \in \mathbb{M}}|x(n)|^{2}=\sum_{k \in \mathbb{K}}|X(k)|^{2}$ $=1$.

The inequality in (14.75) results in the following support uncertainty principle (Elad and Bruckstein, 2002)

$$
\begin{equation*}
\|\mathbf{x}\|_{0}\|\mathbf{X}\|_{0} \geq \frac{1}{\max _{n, k}\left\{\left|u_{k}(n)\right|^{2}\right\}} \tag{14.76}
\end{equation*}
$$

An improved bound of the support uncertainty principle was recently derived in Stanković (2020), using the same relations.

## Reduced Interference Distributions (RID) on Graphs

In order to emphasize the close relations with classical time-frequency analysis, in this subsection we will use the complex-sensitive notation for eigenvectors and spectral vectors. The frequency domain definition of the energy distribution in (14.65) is given by

$$
E(n, k)=x(n) X^{*}(k) u_{k}^{*}(n)=\sum_{p=0}^{N-1} X(p) X^{*}(k) u_{p}(n) u_{k}^{*}(n) .
$$

Then, the general form of a graph distribution can be defined with the help of a kernel $\phi(p, k, q)$, as Stanković et al. (2018a)

$$
\begin{equation*}
G(n, k)=\sum_{p=0}^{N-1} \sum_{q=0}^{N-1} X(p) X^{*}(q) u_{p}(n) u_{q}^{*}(n) \phi(p, k, q) \tag{14.77}
\end{equation*}
$$

Observe that for $\phi(p, k, q)=\delta(q-k)$, the graph Rihaczek distribution in (14.65) follows, while the unbiased energy condition $\sum_{k=0}^{N-1} \sum_{n=0}^{N-1} G(n, k)=$ $E_{x}$ is satisfied if

$$
\sum_{k=0}^{N-1} \phi(p, k, p)=1
$$

The so obtained distribution, $G(n, k)$, may also satisfy the vertex and frequency marginal properties, as elaborated below.

- The vertex marginal property is satisfied if

$$
\sum_{k=0}^{N-1} \phi(p, k, q)=1
$$

This is obvious from

$$
\sum_{k=0}^{N-1} G(n, k)=\sum_{p=0}^{N-1} \sum_{q=0}^{N-1} X(p) X^{*}(q) u_{p}(n) u_{q}^{*}(n)=|x(n)|^{2}
$$

- The frequency marginal property is satisfied if

$$
\phi(p, k, p)=\delta(p-k)
$$

Then, the sum over all vertex indices produces

$$
\sum_{n=0}^{N-1} G(n, k)=\sum_{p=0}^{N-1}|X(p)|^{2} \phi(p, k, p)=|X(k)|^{2}
$$

since $\sum_{n=0}^{N-1} u_{p}(n) u_{q}^{*}(n)=\delta(p-q)$, that is, the eigenvectors are orthonormal.

## Reduced Interference Distribution Kernels

A straightforward extension of classical time-frequency kernels to graph signal processing would be naturally based upon exploiting the relation $\lambda \sim \omega^{2}$, together with an appropriate exponential kernel normalization.


Figure 8.18: The sinc kernel of the reduced interference vertex-frequency distribution in the frequency domain.

The simplest reduced interference kernel in the frequency-frequency shift domain, which would satisfy the marginal properties, is the sinc kernel, given by

$$
\phi(p, k, q)= \begin{cases}\frac{1}{1+2|p-q|}, & \text { for }|k-p| \leq|p-q| \\ 0, & \text { otherwise }\end{cases}
$$

which is shown in Figure 8.18 at the frequency shift corresponding to $k=50$.

Example 69: The sinc kernel was used for a vertex-frequency representation of the signal from Figure $8.1(\mathrm{~d})$, with the results shown in Figure 8.19. This representation is a smoothed version of the energy vertex-frequency distribution in Figure 8.16, whereby both (vertex and frequency) marginals are preserved.

Remark 84: Marginal properties of graph spectrogram. A general vertex-frequency distribution can be written for the vertex-vertex shift domain as a dual form of (14.77), to yield

$$
\begin{equation*}
G(n, k)=\sum_{m=0}^{N-1} \sum_{l=0}^{N-1} x(m) x^{*}(l) u_{k}(m) u_{k}^{*}(l) \varphi(m, n, l) \tag{14.78}
\end{equation*}
$$



Figure 8.19: Reduced interference vertex-frequency distribution of a signal whose vertex-frequency representation is given in Figure 8.3. The marginal properties are given in the panels to the right and below the vertex-frequency representation, and are equal to their corresponding ideal forms given by $|x(n)|^{2}$ and $|X(k)|^{2}$.
where $\varphi(m, n, l)$ is the kernel in this domain (the same mathematical form as for the frequency-frequency shift domain kernel). The frequency marginal is then satisfied if $\sum_{n=0}^{N-1} \varphi(m, n, l)=1$ holds, while the vertex marginal is met if $\varphi(m, n, m)=\delta(m-n)$. The relation of this distribution with the vertex domain spectrogram (14.1) is simple, and is given by

$$
\varphi(m, n, l)=h_{n}(m) h_{n}^{*}(l)
$$

However, this kernel cannot satisfy both the frequency and vertex marginal properties, while the unbiased energy condition $\sum_{n=0}^{N-1} \varphi(m, n$, $m)=1$ reduces to (14.46).
Remark 85: Classical time-frequency analysis follows as a special case from the general form of graph distributions in (14.77), if the considered graph is a directed circular graph. This becomes obvious
upon recalling that the adjacency matrix eigendecomposition produces complex-valued eigenvectors of the form $u_{k}(n)=\exp (j 2 \pi n k / N) / \sqrt{N}$. With the kernel choice

$$
\phi(p, k, q)=\phi(p-q, k-p)=\sum_{n=0}^{N-1} c(p-q, n) e^{-j \frac{2 \pi n k}{N}} e^{j \frac{2 \pi n p}{N}}
$$

in (14.77), the classical (Rihaczek based) Cohen class of distributions directly follows, where $c(k, n)$ is the distribution kernel in the ambiguity domain (Boashash, 2015; Cohen, 1995; Stanković et al., 2014).

A comparison of various vertex-frequency method may be found in Stanković et al. (2020b).

A interesting combination of the time and vertex signal variations into time-vertex signal processing is done in Grassi et al. (2017) and Bohannon et al. (2019).

## 15

## Conclusion

Fundamental ideas of graph signals and their analysis have been introduced starting from an intuitive multisensor estimation example, frequently considered in traditional data analytics. The concept of systems on graphs has been defined using graph signal shift operators, which generalize the signal shift concepts in traditional signal processing. In Part II of our monograph, the Graph Discrete Fourier Transform (GFT) has been at the core of the spectral domain representation of graph signals and systems on graphs, and has been defined based on both the adjacency matrix and graph Laplacian. These spectral domain representations have been used as the basis to introduce graph signal filtering concepts. Methods for the design of graph filters have been presented next, including those based on the polynomial approximation. Various ideas related to the sampling of graph signals, and particularly, the challenging topic of the subsampling, have also been addressed in this part of the monograph. This is followed by conditions for the recovery of signals on graphs, from a reduced number of samples. The concepts of time-varying signals on graphs and basic definitions and methods related to processing random graph signals have also been introduced.

While traditional approaches for graph signal analysis, clustering and segmentation consider only graph topology and spectral properties of graphs, when dealing with signals on graphs, localized analyzes should be employed in order to consider both data on graphs and the graph topology. Such a unified approach to define and implement graph signal localization methods, which takes into account both the data on graph and the corresponding graph topology, is at the core of the presented vertex-frequency analysis. Like in classical time-frequency analysis, main research efforts have been devoted to linear representations of the graph signals which include a localization window for enhanced signal discrimination. Several methods for the definition of localization widows in the spectral and vertex domain have been addressed in Part II of this monograph. Optimization of the window parameters, uncertainty principle, and inversion methods have also been discussed. Following classical time-frequency analysis, energy forms of vertexfrequency energy and reduced interference distributions, which do not use localization windows, have also been considered, together with the elaboration of their role as an estimator of the local smoothness index.

## Part III

## Machine Learning on Graphs, from Graph Topology to Applications

## 16

## Introduction

Graph data analytics have already shown enormous potential, as their flexibility in the choice of graph topologies (irregular data domains) and connections between the entities (vertices) allows for both a rigorous account of irregularly spaced information such as locations and social connections, and also for the incorporation of semantic and contextual cues, even for otherwise regular structures such as images.

In Part I and Part II of this monograph, it was assumed that the graph itself is already defined prior to analyzing data on graphs. The focus of Part I has been on defining graph properties through the mathematical formalism of linear algebra, while Part II introduces graph counterparts of several important standard data analytics algorithms, again for a given graph. However, in many modern applications, graph topology is not known a priori (Cioacă et al., 2019; Das et al., 2017; Dong et al., 2015, 2016; Epskamp and Fried, 2018; Friedman et al., 2008; Hamon et al., 2019, Meinshausen et al., 2006; Pavez and Ortega, 2016; Pourahmadi, 2011; Rabiei et al., 2019; Stanković et al., 2018, 2020), and the focus of this part is therefore on simultaneous estimation of data on a graph and the underlying graph topology. Without loss of generality, it is convenient to assume that the vertices are given, while
the edges and their associated weights are part of the solution to the problem considered and need to be estimated from the vertex geometry and/or the observed data (Bohannon et al., 2019; Caetano et al., 2009; Camponogara and Nazari, 2015; Dal Col et al., 2019; Gu and Wang, 2019; Mao and Gu, 2019; Pasdeloup et al., 2019; Slawski and Hein, 2015; Segarra et al., 2016; Stanković and Sejdić, 2019; Stanković et al., 2017a; Tanaka and Sakiyama, 2019; Thanou et al., 2014; Ubaru et al., 2017; Yankelevsky and Elad, 2016; Zhao et al., 2012; Zheng et al., 2011).

Three scenarios for the estimation of graph edges from vertex geometry or data are considered in this part of the monograph.

- Based on the geometry of vertex positions. In various sensor network setups (such as temperature, pressure, and transportation), the locations of the sensing positions (vertices) are known beforehand, while the vertex distances convey physical meaning about data dependence and thus may be employed for edge/weight determination.
- Based on data association and data similarity. Various statistical measures are available to serve as data association metrics, with the covariance and precision matrices most commonly used. A strong correlation between data on two vertices would indicate a large weight associated with the corresponding edge. A small degree of correlation would indicate nonexistence of an edge (after weight thresholding).
- Based on physically well defined relations among the sensing positions. Examples include electric circuits, power networks, linear heat transfer, social and computer networks, spring-mass systems, to mention but a few. In these cases, edge weighting can usually be well defined based on the underlying context of the considered problem.

After a detailed elaboration of graph definition and graph topology learning techniques, a summary of graph topology learning from data using probabilistic generative models is given. This followed by an account of graph neural networks (GNN), with a special emphasis on
graph convolutional networks (GCN). The analysis is considered from the perspective of graph signal filtering presented in Part II. Graph data analysis is further generalized to the tensor representation of lattice-structured graphs, whereby the graph vertices reside on a highdimensional tensor structure. Finally, two applications of graph-based data analysis are given: (i) an example where domain knowledge is incorporated into financial data analysis (the investment analysis), by means of portfolio cuts; (ii) London underground transportation system. The latter example demonstrates how graph theory can be used to identify the stations in the London underground network which have the greatest influence on the functionality of the traffic, and also to assess the impact of a station closure on service levels across the city.

## 17

## Geometrically Defined Graph Topologies

For a graph that corresponds to a network with geometrically distributed vertices, it is natural to relate the edge weights with the distance between vertices. Consider vertices $m$ and $n$ whose locations in space are defined by the position vectors (coordinates) $\mathbf{r}_{m}$ and $\mathbf{r}_{n}$. The Euclidean distance, $r_{m n}$, between these two vertices is then

$$
r_{m n}=\operatorname{distance}(m, n)=\left\|\mathbf{r}_{m}-\mathbf{r}_{n}\right\|_{2} .
$$

A common way to define the graph weights in such networks is through an exponentially decaying function of the distance, $r_{m n}$, for example as

$$
W_{m n}= \begin{cases}e^{-r_{m n}^{2} / \tau^{2}}, & \text { for } r_{m n} \leq \kappa  \tag{17.1}\\ 0, & \text { for } r_{m n}>\kappa \text { or } m=n,\end{cases}
$$

where $\tau$ and $\kappa$ are suitably chosen constants. This is also physically well justified, as based on $e^{-r_{m n}^{2} / \tau^{2}}$ the weights tend to 1 for closely spaced vertices and diminish for distant vertices.

The rationale for this definition of edge weights is the assumption that the signal value measured at a vertex $n$ is similar to signal values measured at its neighboring vertices. Then, the estimation of a signal at a vertex $n$ should also involve neighboring vertices which are connected
with larger weights (close to 1 ), while the signal values sensed at farther vertices would be less relevant, and are associated with smaller weighting coefficients or are not included at all. A physical interpretation of the weights in (17.1), within the heat distribution and the heat kernel frameworks, can be found in Belkin and Niyogi (2003), where the constant $\tau^{2}=4 t$ is considered as the heat kernel parameter, $t$. Moreover, the Laplacian induced from such weight definition can converge to the continuous Laplace-Beltrami operator if the data is random and uniformly distributed and the number of data point is infinite (Belkin and Niyogi, 2008).

The Gaussian function, used in (17.1), is appropriate in many applications, however, other forms to penalize data values associated with the vertices which are far from the considered vertex may also be used. Examples of such functions include various kernels, such as the kernel given by Chen et al. $(2015,2016)$

$$
W_{m n}= \begin{cases}e^{-r_{m n} / \tau}, & \text { for } r_{m n} \leq \kappa  \tag{17.2}\\ 0, & \text { for } r_{m n}>\kappa \text { or } m=n\end{cases}
$$

or the inverse Euclidean distance between vertices $m$ and $n$, given by

$$
W_{m n}= \begin{cases}\frac{1}{r_{m n}}, & \text { for } r_{m n} \leq \kappa  \tag{17.3}\\ 0, & \text { for } r_{m n}>\kappa \text { or } m=n\end{cases}
$$

Obviously, the simplest form for the edge weighting coefficients is a binary scheme

$$
W_{m n}=A_{m n}= \begin{cases}1, & \text { for } r_{m n} \leq \kappa  \tag{17.4}\\ 0, & \text { for } r_{m n}>\kappa \text { or } m=n\end{cases}
$$

which corresponds to an unweighted graph, with $\mathbf{W}=\mathbf{A}$. This form can be obtained from (17.1) as $\tau^{2} \rightarrow \infty$ (or the heat kernel parameter approaches infinity).
Example 70: We shall illustrate the geometry-based formation of graph structure on the well-known Swiss roll manifold as a domain for data acquisition. This is a three-dimensional surface with the space coordinates, $(x, y, z)$, defined as functions of two parameters, $\xi$ and $\zeta$,
in the following form

$$
\begin{align*}
x & =\frac{1}{4 \pi} \zeta \cos (\zeta) \\
y & =\xi  \tag{17.5}\\
z & =\frac{1}{4 \pi} \zeta \sin (\zeta) .
\end{align*}
$$

The Swiss roll manifold shown in Figure 8.1(a) was created for the parameters, $\xi$ and $\zeta$, ranging within the intervals $-1 \leq \xi \leq 1$ and $\pi \leq \zeta \leq 4 \pi$.

More specifically, we considered a graph with $N=100$ vertices, which were randomly placed on the Swiss roll surface, with the coordinates $\left(x_{k}, y_{k}, z_{k}\right), k=1,2, \ldots, N$, whereby
$\xi_{k}$ was uniformly random within $-1 \leq \xi_{k} \leq 1$
$\zeta_{k}$ was uniformly random within $\pi \leq \zeta_{k} \leq 4 \pi$.
The vertices were connected with edges, with the corresponding edges defined as in (17.1), that is

$$
W_{m n}=\exp \left(-r_{m n}^{2} / \tau^{2}\right),
$$

for $r_{m n}>0.6$, with $W_{m n}=0$ for $r_{m n} \leq 0.6$, as well as for $m=n$; $\tau=1 / 2$. The symbol $r_{m n}$ denotes the shortest geodesic distance between the vertices $m$ and $n$, measured along the Swiss roll manifold, in the following way

$$
r_{m n}^{2}=l_{m n}^{2}+\left(y_{m}-y_{n}\right)^{2},
$$

where the arc length, $l_{m n}$, of the parametric curve in (17.5) is

$$
\begin{aligned}
l_{m n} & =\int_{\zeta_{m}}^{\zeta_{n}} \sqrt{\left(\frac{d x}{d \zeta}\right)^{2}+\left(\frac{d z}{d \zeta}\right)^{2}} d \zeta \\
& =\frac{1}{4 \pi} \int_{\zeta_{m}}^{\zeta_{n}} \sqrt{\left(\frac{d(\zeta \cos (\zeta))}{d \zeta}\right)^{2}+\left(\frac{d(\zeta \sin (\zeta))}{d \zeta}\right)^{2}} d \zeta \\
& =\frac{1}{4 \pi} \int_{\zeta_{m}}^{\zeta_{n}} \sqrt{1+\zeta^{2}} d \zeta=\left.\frac{1}{4 \pi}\left(\frac{1}{2} \zeta \sqrt{\zeta^{2}+1}+\frac{1}{2} \ln \left(\sqrt{\zeta^{2}+1}+\zeta\right)\right)\right|_{\zeta_{m}} ^{\zeta_{n}}
\end{aligned}
$$

Small weight values were hard-thresholded to zero, in order to reduce the number of edges associated with each vertex.


Figure 8.1: Concept of graph definition based on problem geometry. (a) Vertices (points) on a three-dimensional manifold called the Swiss roll surface. (b) A graph representation on the Swiss roll manifold. (c) Two-dimensional presentation of the three-dimensional graph from (b) obtained by unfolding the original 3D surface. (d) Vertices colored using the spectral vector, $\mathbf{q}_{n}=\left[u_{1}(n), u_{2}(n)\right]$, formed from the two smoothest generalized eigenvectors of the graph Laplacian, $\mathbf{u}_{1}$ and $\mathbf{u}_{2}$. (e) Vertices colored using the spectral vector, $\mathbf{q}_{n}=\left[u_{1}(n), u_{2}(n), u_{3}(n)\right]$, formed from the three smoothest eigenvectors of the graph Laplacian, $\mathbf{u}_{1}, \mathbf{u}_{2}$, and $\mathbf{u}_{3}$. The vertex indexing in (d) and (e) is performed based on the sorted values of the smoothest (Fiedler) eigenvector, $\mathbf{u}_{1}$.


Figure 8.1: Continued.

The so produced three-dimensional graph is shown in Figure 8.1(b), and its two-dimensional presentation in Figure 8.1(c). The vertices were ordered so that the values of the Fiedler eigenvector, $u_{1}(n)$, were nondecreasing; the vertices were colored based on the two-dimensional and three-dimensional spectral vectors, $\mathbf{q}_{n}=\left[u_{1}(n), u_{2}(n)\right]$ and $\mathbf{q}_{n}=$ $\left[u_{1}(n), u_{2}(n), u_{3}(n)\right]$, of the Swiss roll in Figures 8.1(d) and (e). This
kind of vertex indexing can also be used for clustering with, for example, the $k$-means clustering presented in Part I, Remark 30.

## Classical Gaussian filter within graph topology formulation.

To illustrate this classical operation on the discrete-time domain data, assume that we desire to perform classical smoothing of a discrete-time domain signal, $x(n)$, at a vertex/instant $n$, through a moving average operation on data observed at neighboring vertices/instants, $x(m)$, using a truncated Gaussian weighting function given by

$$
g(m, n)=e^{-(m-n)^{2} / \tau^{2}}
$$

for $|m-n| \leq \kappa$ and $g(m, n)=0$ for $|m-n|>\kappa$. The smoothed discretetime domain signal, $y(n)$, can be expressed in classical data analysis as

$$
\begin{equation*}
y(n)=\sum_{m} e^{-\frac{(m-n)^{2}}{\tau^{2}}} x(m) \tag{17.6}
\end{equation*}
$$

where the summation is performed for instants/vertices, $m$, such that $|n-m| \leq \kappa$.

We shall now reformulate this classical data processing problem within the graph topology framework. The distance between the sampling instants/vertices, distance ( $m, n$ ), plays a crucial role in signal smoothing, and is defined as

$$
\operatorname{distance}(m, n)=r_{m n}=\|m-n\|_{2}=|m-n| \text {. }
$$

The corresponding edge weights can be defined based on the Gaussian smoothing function, and are given by $W_{m n}=e^{-r_{m n}^{2} / \tau^{2}}$ for $r_{m n} \leq \kappa$, and $W_{m n}=0$ for $r_{m n}>\kappa$ and $m=n$.

The classical smoothed signal, $y(n)$, defined in (17.6) can now be expressed in the form appropriate for the graph framework as

$$
y(n)=x(n)+\sum_{m} x(m) W_{m n}=x(n)+\sum_{m} e^{-\frac{(m-n)^{2}}{\tau^{2}}} x(m)
$$

where the summation is performed for vertices $m$ such that $|m-n| \leq \kappa$ and $m \neq n$. This operation can be defined within the graph analysis framework as a simple first order system on graph, given by

$$
\mathbf{y}=\mathbf{W}^{0} \mathbf{x}+\mathbf{W}^{1} \mathbf{x}
$$



Figure 8.2: Graph which corresponds to the weighted moving average operator with Gaussian weights given in (17.6).
where the edge weights between the vertices $m$ and $n$ are defined by $W_{m n}$.

For example, for $\tau=2$ and $\kappa=2$, the edge weights $W_{m n}$ are shown in Figure 8.2 and this graph-based formulation is identical to the classical discrete-time domain weighted moving average

$$
\begin{equation*}
y(n)=x(n)+\sum_{m} W_{m n} x(m)=\sum_{m=n-2}^{n+2} e^{-\frac{(m-n)^{2}}{4}} x(m) \tag{17.7}
\end{equation*}
$$

with the output signal samples, $y(n)$, equal to the output of a first-order system on the graph given by

$$
\mathbf{y}=\mathbf{W}^{0} \mathbf{x}+\mathbf{W}^{1} \mathbf{x}=3.29 \mathbf{L}^{0} \mathbf{x}-\mathbf{L}^{1} \mathbf{x}
$$

where $\mathbf{W}^{0}$ and $\mathbf{L}^{0}$ are identity matrices, by definition.
For image input data, where the vertices correspond to the pixel positions and the Euclidean distance between pixels is used to model the image domain as a graph, the previous example would model a moving average filtered image, using a radial Gaussian window.
Example 71: Consider the benchmark Minnesota roadmap graph, for which the connectivity map (adjacency matrix) is designated by the road connections and the vertices are located at the road crossings. The edges are defined by the adjacency matrix and were weighted according to their Eucledian distances using the weighting scheme in (17.2), with
$\tau=25 \mathrm{~km}$, to give

$$
W_{m n}=e^{-r_{m n}^{2} / \tau^{2}},
$$

where the threshold $\kappa$ was not used since the connectivity is already determined by the given adjacency matrix.

We considered a simulated temperature signal in the Minnesota area (normalized temperature filed) which was calculated as

$$
\begin{aligned}
x(n)= & 0.9\left(0.1+0.8 e^{-\left(\frac{x-150}{100}\right)^{2}-\left(\frac{y-400}{200}\right)^{2}},\right. \\
& \left.+0.5 e^{-\left(\frac{x-450}{200}\right)^{2}-\left(\frac{y-400}{100}\right)^{2}}+e^{-\left(\frac{x-500}{250}\right)^{2}-\left(\frac{y-150}{200}\right)^{2}}\right)+\nu(n)
\end{aligned}
$$

where $\nu(n)$ is white Gaussian noise with a standard deviation $\sigma_{\nu}=0.3$. The noise-free and noisy versions of this graph temperature signal are given respectively in Figures 8.3(a) and (b). The noisy signal was filtered in the vertex domain by a low-pass filter implemented using Taubin's $\alpha-\beta$ algorithm (presented in Part II, Section 6.2) with $\alpha=0.15$ and $\beta=0.1$, and the so enhanced temperature signal is shown in Figure 8.3(c). The output SNR of 19.34 dB was achieved for the input SNR of 9.35 dB , a gain of 10 dB .


Figure 8.3: Temperatures simulated on the Minnesota roadmap graph. (a) Original synthetic temperature field signal. (b) Noisy temperature signal. (c) Low-pass filtered temperature signal from (b). The signal values are designated by the corresponding vertex color.

## 18

## Graph Topology Based on Signal Similarity

In the previous sections, graph weights were defined on the assumption that the geometric distance of vertices, where the signal is sensed, is a reliable indicator of data similarity, or some other more general data association. Indeed, this is the case with, for example, the measurements of atmospheric temperature and (barometric) pressure when the terrain configuration has no influence on the similarity of measured data. However, in general, the geometric distance between vertices may not be a good indicator of data similarity.

One such example is in image processing, where the pixel color values themselves can be used as an indicator of signal similarity; this can be achieved in combination with the distances between pixels, which play the role of vertices. If the intensity values at pixels indexed by $m$ and $n$ are denoted by $x(m)$ and $x(n)$, then the difference of intensities is defined by

$$
\text { Intensity }_{\text {distance }}(m, n)=r_{m n}=|x(m)-x(n)|,
$$

and the corresponding weights may be defined as

$$
W_{m n}= \begin{cases}e^{-(x(m)-x(n))^{2} / \tau^{2}}, & \text { for } r_{m n} \leq \kappa \text { and } \rho_{m n} \leq \gamma \\ 0, & \text { for } r_{m n}>\kappa \text { or } \rho_{m n}>\gamma \text { or } m=n,\end{cases}
$$

where $\rho_{m n}$ is a geometric distance between the considered pixels/vertices and $\tau, \kappa$, and $\gamma$ are chosen constants.

More reliable measures of data similarity can be defined when it is possible to collect more than one snapshot of data for a given set of sensing points/vertices. Assume that at every vertex $n=0,1, \ldots, N-1$ we have acquired $P$ signal values, denoted by $x_{p}(n), p=1,2, \ldots, P$. Such a dataset may be equally treated as multivariate data or signal measurements in a sequence. Then, an appropriate similarity measure for a real-valued signal at vertices $m$ and $n$ may be

$$
\begin{equation*}
r_{m n}^{2}=\frac{\sum_{p=1}^{P}\left(x_{p}(m)-x_{p}(n)\right)^{2}}{\sum_{m=1}^{N-1} \sum_{n=1}^{N-1} \sum_{p=1}^{P}\left(x_{p}(m)-x_{p}(n)\right)^{2}} \tag{18.1}
\end{equation*}
$$

so that $\sum_{m=1}^{N-1} \sum_{n=1}^{N-1} r_{m n}^{2}=1$.
The graph weights can again be defined using any of the previous forms, for example, as

$$
W_{m n}= \begin{cases}e^{-r_{m n}^{2} / \tau^{2}}, & \text { for } r_{m n} \leq \kappa \\ 0, & \text { for } r_{m n}>\kappa \text { or } m=n\end{cases}
$$

or

$$
W_{m n}= \begin{cases}e^{-r_{m n} / \tau}, & \text { for } r_{m n} \leq \kappa \\ 0, & \text { for } r_{m n}>\kappa \text { or } m=n\end{cases}
$$

The geometric distance between the considered pixels/vertices, $\rho_{m n}$, can also be included in the weight definition.

Random observations. When the signal values, $x_{p}(n)$, acquired over $P$ observations, $p=1,2, \ldots, P$ at $N$ vertices $n=0,1, \ldots, N-1$, are drawn from zero-mean random noise with equal variances, $\sigma_{x}^{2}=1$, the similarity measure can be defined by

$$
r_{m n}^{2}=\frac{\sum_{p=1}^{P}\left(x_{p}(m)-x_{p}(n)\right)^{2}}{\sqrt{\sum_{p=1}^{P} x_{p}^{2}(m) \sum_{p=1}^{P} x_{p}^{2}(n)}}=2\left(1-R_{x}(m, n)\right)
$$

where

$$
R_{x}(m, n)=\frac{1}{P} \sum_{p=1}^{P} x_{p}(m) x_{p}(n)
$$

represents the normalized sample autocorrelation function and $\sigma_{x}^{2}=$ $\frac{1}{P} \sum_{p=1}^{P} x_{p}^{2}(n)=1$ for sufficiently large $P$.

Similarity metrics for images. The same structure can be used for other applications, such as in image classification or handwritten letter recognition. In these cases, the distance between an image $m$ and an image $n$ is equal to

$$
\begin{equation*}
r_{m n}=\operatorname{Image}_{\text {distance }}(m, n)=\left\|\mathbf{x}_{m}-\mathbf{x}_{n}\right\|_{F}, \tag{18.2}
\end{equation*}
$$

where

$$
\|\mathbf{x}\|_{F}=\sqrt{\sum_{m} \sum_{n}|x(m, n)|^{2}}
$$

is the Frobenius norm of an image matrix $\mathbf{x}$ (that is, the square root of the sum of squared image values over all pixels).
Block collaborative image processing. A class of recent efficient image processing algorithms is based on detecting similar blocks within an image, followed by collaborative processing using those similar blocks. Image enhancement algorithms then assume that the basic images are also similar within these blocks, while the corresponding noise is not related and can be averaged out. The similarity between the image blocks, $\mathbf{x}_{m}$ and $\mathbf{x}_{n}$, may then be defined similar to (18.2), using their distance given by

$$
r_{m n}=\operatorname{Block}_{\text {distance }}(m, n)=\left\|\mathbf{x}_{m}-\mathbf{x}_{n}\right\|_{F} .
$$

The similarity among the blocks in an image can be modeled by a graph, and such graph models may be used as bases for collaborative processing of image blocks. Recall that a block of $B \times B$ pixels is an example of a vertex in a $B^{2}$-dimensional space, since it is defined by $B \times B$ independent pixel values (vertex coordinates/dimensions).

Generalized distance measure. The Euclidean distance is typically used in the calculation of the distance between two blocks of data, $\mathbf{x}_{m}$ and $\mathbf{x}_{n}$. It may be generalized by introducing the inner product matrix, $\mathbf{H}$, into distance calculation to yield

$$
r_{m n}^{2}=\left(\mathbf{x}_{m}-\mathbf{x}_{n}\right)^{T} \mathbf{H}\left(\mathbf{x}_{m}-\mathbf{x}_{n}\right),
$$

where the data sets $\mathbf{x}_{m}$ and $\mathbf{x}_{n}$ are represented in the column vector form. When the inner product matrix, $\mathbf{H}$, is an identity matrix, $\mathbf{H}=\mathbf{I}$, the standard Euclidean distance is obtained. If we use, for example, $\mathbf{H}=\mathbf{U}_{C} \mathbf{U}_{C}^{T}$, where $\mathbf{U}_{C}$ is the matrix with cosine transform basis functions as its columns, we will arrive at

$$
\begin{aligned}
r_{m n}^{2} & =\left(\mathbf{x}_{m}-\mathbf{x}_{n}\right)^{T} \mathbf{U}_{C} \mathbf{U}_{C}^{T}\left(\mathbf{x}_{m}-\mathbf{x}_{n}\right) \\
& =\left(\mathbf{C}_{m}-\mathbf{C}_{n}\right)^{T}\left(\mathbf{C}_{m}-\mathbf{C}_{n}\right)=\left\|\mathbf{C}_{m}-\mathbf{C}_{n}\right\|_{2}^{2}
\end{aligned}
$$

where $\mathbf{C}_{n}$ is the 2D discrete cosine transform (2D DCT) of $\mathbf{x}_{n}$, written in a vector column format. By virtue of this representation, problem dimensionality can straightforwardly be reduced using only the $K$ slowest-varying basis functions, $\mathbf{U}_{C}^{(K)}$, instead of the full 2D DCT transformation matrix (this operation corresponds to low-pass filtering of $\mathrm{x}_{n}$ in the 2D DCT domain, by keeping the $K$ slowest-varying coefficients). In this case, the distance, $r_{m n}^{2}$, is of the form

$$
\begin{aligned}
r_{m n}^{2} & =\left(\mathbf{x}_{m}-\mathbf{x}_{n}\right)^{T} \mathbf{U}_{C}^{(K)} \mathbf{U}_{C}^{(K)^{T}}\left(\mathbf{x}_{m}-\mathbf{x}_{n}\right) \\
& =\left\|\mathbf{C}_{m}^{(K)}-\mathbf{C}_{n}^{(K)}\right\|_{2}^{2}
\end{aligned}
$$

and is calculated based on the reduced original dimensionality of $\mathbf{x}_{n}$ or $\mathbf{C}_{n}$ to the dimensionality $K$ of $\mathbf{C}_{n}^{(K)}$.

Another interesting form of the inner product matrix is the inverse covariance matrix $\mathbf{H}=\boldsymbol{\Sigma}^{-1}$, which will be discussed later in Sections 19.4 and 21.8.
Example 72: A noisy image with a designated set of 29 blocks of pixels is shown in Figure 8.1(a). The similarity between any two of the blocks was defined based on the distance

$$
r_{m n}^{2}=\frac{1}{B^{2}}\left\|\mathbf{C}_{m}-\mathbf{C}_{n}\right\|_{F}^{2},
$$

where $\mathbf{C}_{n}$ represents the matrix form of the 2D DCT of the image block $\mathbf{x}_{n}$.

The 2D DCT was then hard-thresholded, with a threshold equal to $0.1 \mathrm{max}\left|\mathbf{C}_{n}\right|$, to reduce the influence of noise (and dimensionality problems), that is, all 2D DCT coefficients bellow this threshold were


Figure 8.1: Graph learning based on the similarity of blocks of image data. (a) Original image with designated blocks of pixels. (b) The graph produced from the blocks in (a). Notice that the resulting graph consists of seven disconnected subgraphs, which correspond to the seven different groups of blocks.
set to zero, to give

$$
C_{n}(k, l)= \begin{cases}C_{n}(k, l), & \text { if }\left|C_{n}(k, l)\right|>0.1 \max \left|\mathbf{C}_{n}\right| \\ 0, & \text { elsewhere }\end{cases}
$$

The edge weights, $W_{m n}$, for a graph representation of the considered blocks (as vertices) were then calculated as

$$
W_{m n}=\exp \left(-r_{m n}^{2} B\right),
$$

for $r_{m n} \leq 0.26$, and $W_{m n}=0$ for $r_{m n}>0.26$, or $m=n$, with $B=16$.
The so obtained graph, which indicates block similarity, is given in Figure 8.1(b). This graph representation is very convenient for collaborative image processing, since the graph structure will ensure that the processing is performed independently on the sets of blocks which share relevant information (connected subgraphs). Notice that the blocks within each subgraph can be considered as a 3D signal of RGB components. Then, for example, a simple averaging over similar blocks (within one subgraph), will not significantly degrade the image detail, while at the same time it will reduce the corresponding noise, as it is uncorrelated in different blocks.

This is precisely the principle of the Block-Matching and 3D filtering (BM3D) algorithm, where the noise and the image are estimated from the set of similar blocks (in our example, from the blocks within a subgraph). The estimation of the related set of blocks in the image and the estimation of noise power is then used to define the Wiener filter. Such Wiener filter is used to filter all related blocks (within the subgraph). The procedure is repeated for each set of similar blocks (subgraphs). Of course, in the case of the BM3D algorithm, for each considered (reference) block, $\mathbf{x}_{n}$, it is desirable to search over the whole image and to find as many similar blocks as possible in order to obtain the best possible Wiener filter and consequently achieve maximum possible noise reduction.

In this example, the blocks and the threshold for edge weights, $W_{m n}$, were selected so as to produce disconnected graph components and a clear segmentation scheme. If this was not the case, vertex clustering and graph segmentation could be performed using the theory presented in Part I.

Recall that in Part I, Example 24 the structural similarity index (SSIM), was used instead of the simple difference/distance, to relate and cluster images.
Example 73: Eight images with the hand-written letter "b" were considered and the task was to create their graph representation. The SSIM was calculated for each pair of images and the edge-weights were equal to the calculated SSIM values, as shown in Figure 8.2(a). For the graph from Figure 8.2(b), the generalized eigenvectors of the Laplacian were calculated and the vertices were colored using first the smoothest (Fiedler) eigenvector, $\mathbf{u}_{1}$, and then using the two smoothest eigenvectors $\mathbf{u}_{1}$ and $\mathbf{u}_{2}$, as a basis for image clusterings, as respectively shown in Figure 8.2(c) (left) and (right).

(a)

(b)

(c)

Figure 8.2: Graph representation of a set of hand-written images of the letter " $b$ ". The images serve as vertices, while the weight matrix for the edges is defined through the structural similarity index metric (SSIM) between the images, with $W_{m n}=\operatorname{SSIM}(m, n)$. The vertices are colored in (c) using first the smoothest (Fiedler) eigenvector, $\mathbf{u}_{1}$, and then the two smoothest eigenvectors, $\mathbf{u}_{1}$ and $\mathbf{u}_{2}$, of the generalized eigenvectors of the Laplacian (with the corresponding spectral vectors $\mathbf{q}_{n}=\left[u_{1}(n)\right]$ and $\left.\mathbf{q}_{n}=\left[u_{1}(n), u_{2}(n)\right]\right)$ respectively shown in Figure 8.2(c) (left) and (right).

## 19

## Learning of Graph Laplacian from Data

Consider a graph signal for which we have available $P$ independent observations. Denote by $x_{p}(n)$ the observed signal at a vertex, $n$, and for an observation, $p$. The column vector with graph signal samples from the $p$ th observation is denoted by $\mathbf{x}_{p}$. All observations from this graph signal can then be arranged into an $N \times P$ matrix, given by

$$
\mathbf{X}_{P}=\left[\mathbf{x}_{1}, \mathbf{x}_{2}, \ldots, \mathbf{x}_{P}\right] .
$$

Designate the $(n+1)$-th row of this matrix by a row vector, $\mathbf{y}_{n}$, which corresponds to the vertex $n$, that is

$$
\begin{equation*}
\mathbf{y}_{n}=\left[x_{1}(n), x_{2}(n), \ldots, x_{P}(n)\right] . \tag{19.1}
\end{equation*}
$$

Then, the matrix of observations can also be written as

$$
\mathbf{X}_{P}=\left[\begin{array}{c}
\mathbf{y}_{0} \\
\mathbf{y}_{1} \\
\vdots \\
\mathbf{y}_{N-1}
\end{array}\right] .
$$

The correlation coefficient between vertices $m$ and $n$, estimated by averaging over the set of $P$ observations, is then given by

$$
R_{x}(m, n)=\frac{1}{P} \sum_{p=1}^{P} x_{p}(m) x_{p}(n)=\frac{1}{P} \mathbf{y}_{m} \mathbf{y}_{n}^{T}
$$

or in a matrix form

$$
\begin{equation*}
\mathbf{R}_{x}=\frac{1}{P} \mathbf{X}_{P} \mathbf{X}_{P}^{T} \tag{19.2}
\end{equation*}
$$

If the observations are not zero-mean, then we should use the covariance matrix, $\boldsymbol{\Sigma}$, with elements

$$
\begin{equation*}
\Sigma_{x}(m, n)=\frac{1}{P} \sum_{p=1}^{P}\left(x_{p}(m)-\mu(m)\right)\left(x_{p}(n)-\mu(n)\right) \tag{19.3}
\end{equation*}
$$

where $\mu(n)$ is the mean of the observations at the vertex $n$.
Remark 86: Since the correlation matrix in (19.2) includes contribution from signals at all vertices, it accumulates correlations obtained through all possible walks from the current vertex, $n$, to any other vertex, $m$. This also means that the correlation coefficient between two vertices will produce misleading results if there exists one or more other vertices, $q$, where the signal is strongly correlated with both of the considered vertices, $m$ and $n$. This is why the naive use of correlation tends to overestimate the strength of direct vertex connections; this renders it a poor metric for establishing direct links (edges) between vertices. To resolve this issue, either additional conditions should be imposed on the correlation matrix, or other statistical parameters may be used for edge weight estimation.
Example 74: Consider four random graph signals observed at the vertices $n=0,1,2,3$, and given by

$$
\begin{align*}
& x_{p}(0)=\nu_{0}(p) \\
& x_{p}(1)=x_{p}(0)+\nu_{1}(p)  \tag{19.4}\\
& x_{p}(2)=x_{p}(1)+\nu_{2}(p) \\
& x_{p}(3)=x_{p}(2)+\nu_{3}(p)
\end{align*}
$$

where $\nu_{0}(p), \nu_{1}(p), \nu_{2}(p), \nu_{3}(p)$ are mutually uncorrelated, white random variables with zero mean and unit variance. The elements of the
correlation matrix for the above signals can be calculated as, for example

$$
R_{x}(0,1)=\mathrm{E}\left\{x_{p}(0) x_{p}(1)\right\}=\mathrm{E}\left\{x_{p}(0)\left(x_{p}(0)+\nu_{1}(p)\right)\right\}=1
$$

or

$$
\begin{aligned}
R_{x}(0,2) & =\mathrm{E}\left\{x_{p}(0) x_{p}(2)\right\}=\mathrm{E}\left\{x_{p}(0)\left(x_{p}(1)+\nu_{2}(p)\right)\right\} \\
& =\mathrm{E}\left\{x_{p}(0)\left(x_{p}(0)+\nu_{2}(p)+\nu_{2}(p)\right)\right\}=1 .
\end{aligned}
$$

Observe from (19.4) that, although the signal value $x_{p}(2)$ is not directly related to $x_{p}(0)$, the correlation coefficient, $R_{x}(0,2)$, is nonzero and even equal to $R_{x}(0,1)$, since there is an indirect link between these two signal values through $x_{p}(1)$. In practical applications, it is therefore desirable to avoid this indirect cumulative contribution to the correlation coefficient which results in an overestimated edge weight.

All correlation coefficients for the above example can be written in a matrix form as

$$
\mathbf{R}_{x}=\left[\begin{array}{cccc}
1 & 1 & 1 & 1  \tag{19.5}\\
1 & 2 & 2 & 2 \\
1 & 2 & 3 & 3 \\
1 & 2 & 3 & 4
\end{array}\right]
$$

with the inverse correlation matrix, called the precision matrix, as

$$
\mathbf{C}=\mathbf{R}_{x}^{-1}=\left[\begin{array}{rrrr}
2 & -1 & 0 & 0  \tag{19.6}\\
-1 & 2 & -1 & 0 \\
0 & -1 & 2 & -1 \\
0 & 0 & -1 & 1
\end{array}\right]
$$

Remark 87: Observe that while the autocorrelation in (19.5) overestimates the strength of edge links, the precision matrix in (19.6) produces the desired results, since for example, $C(0,2)=0$, which indicates that there is no direct relation between $x_{p}(0)$ and $x_{p}(2)$, although $x_{p}(2)$ is indirectly linked to $x_{p}(0)$ through $x_{p}(1)$.

Similar to the normalized correlation, the normalized precision matrix, $\mathbf{C}^{(N)}$, is defined by $C_{m n}^{(N)}=C_{m n} / \sqrt{C_{m m} C_{n n}}$ to produce

$$
\mathbf{C}^{(N)}=\left[\begin{array}{rrrr}
1 & -0.5 & 0 & 0  \tag{19.7}\\
-0.5 & 1 & -0.5 & 0 \\
0 & -0.5 & 1 & -1 / \sqrt{2} \\
0 & 0 & -1 / \sqrt{2} & 1
\end{array}\right]
$$

### 19.1 Imposing Sparsity on the Connectivity Matrix

The minimization of the sparsity of the weight matrix keeps the number of its nonzero values to the minimum (Stanković, 2001; Stanković et al., 2018b), thus resulting in graphs with the smallest possible number of edges.

Consider the vertex $n=0$ and the graph signal observation vector as in (19.1), at this vertex. We can estimate the edge weights from this vertex to all other vertices, $\beta_{0 m}, m=1,2,3, \ldots, N-1$, by minimizing the cost function (Epskamp and Fried, 2018; Meinshausen et al., 2006; Pourahmadi, 2011),

$$
\begin{equation*}
J_{0}=\left\|\mathbf{y}_{0}-\sum_{m=1}^{N-1} \beta_{0 m} \mathbf{y}_{m}\right\|_{2}^{2}+\rho \sum_{m=1}^{N-1}\left|\beta_{0 m}\right| . \tag{19.8}
\end{equation*}
$$

Physically, the first term promotes the correlation between the observations $\mathbf{y}_{0}$ at the considered vertex $(n=0)$ and the observations $\mathbf{y}_{m}$ at all the other vertices, for $m=1,2,3, \ldots, N-1$; the second term promotes sparsity in the coefficient vector $\boldsymbol{\beta}_{0}$ (number of nonzero coefficients $\beta_{0 m}$ ), while the parameter $\rho$ balances between these two criteria.

The matrix form of the cost function (19.8) is given by

$$
\begin{equation*}
J_{0}=\left\|\mathbf{y}_{0}^{T}-\mathbf{Y}_{0}^{T} \boldsymbol{\beta}_{0}^{T}\right\|_{2}^{2}+\rho\left\|\boldsymbol{\beta}_{0}\right\|_{1}, \tag{19.9}
\end{equation*}
$$

where $\mathbf{Y}_{0}$ is obtained from the matrix $\mathbf{X}_{P}$ after the first row is removed and used as $\mathbf{y}_{0}$, with

$$
\boldsymbol{\beta}_{0}=\left[\beta_{01}, \beta_{02}, \ldots, \beta_{0 N-1}\right] .
$$

Example 75: For the correlation matrix from Example 74 and the observation vector, $\mathbf{y}_{0}$, at the vertex $n=0$, given by

$$
\mathbf{y}_{0}=\left[x_{1}(0), x_{2}(0), \ldots, x_{P}(0)\right]=\left[\nu_{0}(1), \nu_{0}(2), \ldots, \nu_{0}(P)\right],
$$

we can find the solution to (19.9) with $\rho=0$, which corresponds to the two-norm minimization of the error function, given by

$$
\frac{\partial J_{0}}{\partial \boldsymbol{\beta}_{0}^{T}}=2 \mathbf{Y}_{0}\left(\mathbf{y}_{0}^{T}-\mathbf{Y}_{0}^{T} \boldsymbol{\beta}_{0}^{T}\right)=\mathbf{0}
$$

or

$$
\boldsymbol{\beta}_{0}^{T}=\left(\mathbf{Y}_{0} \mathbf{Y}_{0}^{T}\right)^{-1} \mathbf{Y}_{0} \mathbf{y}_{0}^{T}=\left[\begin{array}{lll}
2 & 2 & 2 \\
2 & 3 & 3 \\
2 & 3 & 4
\end{array}\right]^{-1}\left[\begin{array}{l}
1 \\
1 \\
1
\end{array}\right]=\left[\begin{array}{c}
0.5 \\
0 \\
0
\end{array}\right]
$$

since $\mathbf{Y}_{0} \mathbf{Y}_{0}^{T}$ and $\mathbf{Y}_{0} \mathbf{y}_{0}^{T}$ are submatrices of the correlation matrix $\mathbf{R}_{x}$, given in (19.5).

In the same way, the other three coefficient vectors, $\boldsymbol{\beta}_{1}^{T}, \boldsymbol{\beta}_{2}^{T}, \boldsymbol{\beta}_{3}^{T}$, were calculated to produce (with added zero-values at the diagonal) the coefficient matrix

$$
\boldsymbol{\beta}=\left[\begin{array}{cccc}
0 & 0.5 & 0 & 0  \tag{19.10}\\
0.5 & 0 & 0.5 & 0 \\
0 & 0.5 & 0 & 0.5 \\
0 & 0 & 1 & 0
\end{array}\right] .
$$

Since this procedure does not guarantee the symmetry of $\beta_{n m}=\beta_{m n}$, the edge weights could have also been calculated through the geometric mean,

$$
\begin{equation*}
W_{n m}=\sqrt{\beta_{n m} \beta_{m n}} \tag{19.11}
\end{equation*}
$$

to produce

$$
\mathbf{W}=\left[\begin{array}{cccc}
0 & 0.5 & 0 & 0  \tag{19.12}\\
0.5 & 0 & 0.5 & 0 \\
0 & 0.5 & 0 & 1 / \sqrt{2} \\
0 & 0 & 1 / \sqrt{2} & 0
\end{array}\right]
$$

This weight matrix is symmetric and corresponds to an undirected graph.

The graph Laplacian, $\mathbf{L}=\mathbf{D}-\mathbf{W}$, is then obtained by changing the signs of the elements in $\mathbf{W}$ and adding appropriate diagonal elements, $\mathbf{D}$, such that the sum for each row or column is zero, that is

$$
\mathbf{L}=\left[\begin{array}{rrrr}
0.5 & -0.5 & 0 & 0 \\
-0.5 & 1 & -0.5 & 0 \\
0 & -0.5 & 1.207 & -0.707 \\
0 & 0 & -0.707 & 0.707
\end{array}\right]
$$

Notice that the structure of nonzero off-diagonal elements in this matrix is the same as in the normalized precision matrix in (19.7), although the corresponding values were obtained through two quite different approaches to the estimation of the relations among graph data observed at different vertices.
LASSO approach. In general, the problem in (19.9) can be solved using the well established least absolute shrinkage and selection operator (LASSO) type minimization, the regression analysis method that
performs both variable selection and regularization, as

$$
\boldsymbol{\beta}_{0}=\operatorname{lasso}\left(\mathbf{Y}_{0}^{T}, \mathbf{y}_{0}^{T}, \rho\right) .
$$

For more detail on the derivation and implementation of LASSO see Section 20 and Algorithm 2.

```
Algorithm 2. LASSO (ISTA variant), \mathbf{B}=\operatorname{lasso(Y, y, \rho)}
Input:
- Observation column vector \(\mathbf{y}, P \times 1\)
```

- Observation matrix $\mathbf{Y}, P \times N$
- Sparsity promotion parameter $\rho$

1: $\mathbf{B} \leftarrow \mathbf{0}_{N \times 1}$
2: $\alpha \leftarrow 2 \max \left\{\operatorname{eig}\left(\mathbf{Y}^{T} \mathbf{Y}\right)\right\}$
3: repeat
4: $\quad \mathbf{s} \leftarrow \frac{1}{\alpha} \mathbf{Y}^{T}(\mathbf{y}-\mathbf{Y B})+\mathbf{B}$
5: $\quad$ for $k \leftarrow 1$ to $N$ do
6: $\quad B(k) \leftarrow \begin{cases}s(k)+\rho, & \text { for } s(k)<-\rho \\ 0, & \text { for }|s(k)| \leq \rho \\ s(k)-\rho, & \text { for } s(k)>\rho\end{cases}$
7: end for
until stopping criterion is satisfied

## Output:

- Reconstructed coefficients B

For the data from Example 75, the LASSO approach yields

$$
\boldsymbol{\beta}_{0}=\operatorname{lasso}\left(\mathbf{Y}_{0}^{T}, \mathbf{y}_{0}^{T}, 0.01\right)=[0.49,0,0] .
$$

This result is almost the same as the first row (excluding the first element assumed to be zero) in the matrix $\boldsymbol{\beta}$ in (19.10), as was expected since the solution in the first row in (19.10) is already with maximum sparsity. Since in this setting the number of independent observations, $P$, could be significantly larger than the number of coefficients, $\beta_{0 m}$,
for this case the least squares estimation is optimal and there are no additional degrees of freedom available to improve the sparsity of the solution (the solution, in this case is already with one nonzero element, that is, with the maximum possible sparsity). On the other hand, ways to promote sparsity would be necessary if the number of observations is smaller than the number of vertices (compressive sensing theory framework).

The minimization in (19.9) was performed for the vertex $n=0$, and should be repeated for all vertices $n=1,2, \ldots, N-1$, through the cost function

$$
J_{n}=\left\|\mathbf{y}_{n}^{T}-\mathbf{Y}_{n}^{T} \boldsymbol{\beta}_{n}\right\|_{2}^{2}+\rho\left\|\boldsymbol{\beta}_{n}\right\|_{1}
$$

to obtain

$$
\boldsymbol{\beta}_{n}=\operatorname{lasso}\left(\mathbf{Y}_{n}^{T}, \mathbf{y}_{n}^{T}, \rho\right)
$$

In general, if the resulting weight matrix, $\boldsymbol{\beta}$, is not symmetric then the edge weights could be calculated as $W_{n m}=\sqrt{\beta_{n m} \beta_{m n}}$, as mentioned in (19.11).

Example 76: As an example for graph learning from data using the LASSO approach, consider the graph from Figure 2.2, Part I and $P=3,000$ observations, which were simulated by assuming external white Gaussian sources with zero-mean and variance $\sigma^{2}=1$, located at two randomly chosen vertices (see Section 20 and Figure 8.2). An $N \times P$ matrix of observed signal values, $\mathbf{X}_{P}$, was then formed, and from its rows the vector $\mathbf{y}_{n}$ and matrix $\mathbf{Y}_{n}$ were obtained. The matrix of coefficients $\boldsymbol{\beta}=\left[\beta_{m n}\right]_{N \times N}$ follows from lasso $\left(\mathbf{Y}_{n}^{T}, \mathbf{y}_{n}^{T}, \rho\right)$ with $n=0,1,2,3,4,5,6,7$ and $\rho=0.2$, to yield

$$
\boldsymbol{\beta}=\left[\begin{array}{llllllll}
0 & 0.0 & 0.75 & 0.16 & 0 & 0 & 0 & 0 \\
0.03 & 0 & 0.35 & 0 & 0.19 & 0 & 0 & 0.18 \\
0.75 & 0.35 & 0 & 0.10 & 0.11 & 0 & 0 & 0 \\
0.16 & 0 & 0.10 & 0 & 0 & 0 & 0.45 & 0 \\
0 & 0.19 & 0.11 & 0 & 0 & 0.74 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.74 & 0 & 0 & 0.19 \\
0 & 0 & 0 & 0.45 & 0 & 0 & 0 & 0.58 \\
0 & 0 & 0 & 0 & 0 & 0.19 & 0.58 & 0
\end{array}\right]
$$



Figure 8.1: Estimation of the weight matrix for the graph from Figure 2.2 in Part I with color-coded element values. (a) Ground truth weight matrix. (b) Estimated weight matrix with LASSO and $\rho=0.2$. (c) Estimated weight matrix with LASSO and $\rho=0.05$. (d) Estimated weight matrix with LASSO and $\rho=1$.

The ground truth weights and the weights estimated through the LASSO are shown in Figures 8.1(a), (b). The estimation was repeated for the cases of (i) a smaller value of balance parameter $\rho=0.05$ (reducing the sparsity contribution and resulting in an increased number of nonzero weights, as in Figure 8.1(c)), and (ii) a larger balance parameter $\rho=1$ (strengthening the sparsity contribution and resulting in a reduced number of nonzero weights, as Figure 8.1(d)).

The same experiment was next repeated for the unweighted graph from Figure 2.1(a) in Part I, and the result is shown in Figure 8.2. In


Figure 8.2: Adjacency matrix for the unweighted graph from Figure 2.1(a) in Part I. (a) Ground truth adjacency matrix. (b) Estimated adjacency matrix with LASSO and $\rho=0.2$.
this case, the obtained values of $\boldsymbol{\beta}$ were used to decide whether $A_{m n}=1$ or $A_{m n}=0$.

Example 77: The graph topology in the temperature estimation example in Part II, Section 2 was determined based on the geometry and geographic distances of the locations/vertices where the temperature is sensed (Stanković et al., 2019b). Now, we shall revisit this example by simulating the temperature field, $\mathbf{X}$, at the locations shown in Figure 8.3(a) and over a period of time with the aim to learn the graph topology from this data. The simulated temperature field over $P=150$ days is shown in Figure 8.3(b). The weight matrix calculated from the geographic positions of the vertices is denoted as the ground truth weight matrix, W, and shown in Figure 8.3(c). The corresponding weight matrix, which is learned from data in Figure 8.3(b) using the column LASSO with $\rho=0.2$, is given in Figure 8.3(d). Before the calculation of the correlation matrices, the mean value of the sensed temperatures was removed from $x_{p}(n)$, for each observation $p$.

### 19.2 Smoothness Constrained Learning of Graph Laplacian

Consider a set of noisy graph data, $x_{p}(n)$, measured over $P$ observations, $p=1,2, \ldots, P$, at $N$ vertices $n=0,1 \ldots, N-1$, of an undirected graph.


Figure 8.3: Data-based learning of graph topology in the temperature sensing example from Part II, Section 17. (a) Sensing locations in a geographic region along the Adriatic sea. (b) Temperatures measured at $N=16$ sensing locations over $P=150$ days. (c) Ground truth weight matrix, $\mathbf{W}$, obtained through geographic properties of the sensing locations as in Part II, Section 17. (d) The weight matrix, $\mathbf{W}$, estimated solely based on the analysis of data from (b) and using the LASSO approach.

The aim is to learn the graph connectivity (its graph Laplacian) from the observed data. To this end, it is necessary to find a signal, $y_{p}(n)$, that is close to the observations, $x_{p}(n)$, under the condition that $y_{p}(n)$ is as smooth as possible on a graph. This formulation is similar to that addressed in Part I.
Remark 88: The smoothness condition may be imposed based on the physically meaningful assumption that the data at close and strongly related vertices should have similar values, that is, without abrupt changes in signal values from vertex to vertex. This requirement imposes gradual change of data over the graph domain, as is the case in many practical applications (Chepuri et al., 2017; Dong et al., 2016, 2019; Kalofolias, 2016; Sadhanala et al., 2016).

The graph signal, $y_{p}(n)$, can now be found by minimizing the cost function

$$
J_{p}=\frac{1}{2}\left\|\mathbf{y}_{p}-\mathbf{x}_{p}\right\|_{2}^{2}+\alpha \mathbf{y}_{p}^{T} \mathbf{L} \mathbf{y}_{p}, \quad \text { for } p=1,2, \ldots, P
$$

whereby the first term aims at finding $\mathbf{y}_{p}$ which is as close as possible to $\mathbf{x}_{p}$, while the second term, $\mathbf{y}_{p}^{T} \mathbf{L} \mathbf{y}_{p}$, promotes the smoothness of graph signal $\mathbf{y}_{p}$.
Remark 89: The difference in the problem considered here from the smoothing problem addressed in Part I is that here the graph Laplacian (graph edges and their weights) is not known. In other words, the graph Laplacian, $\mathbf{L}$, has to be determined along with the output signal $\mathbf{y}_{p}$, that is, the graph topology has to be learned from data.

Since we have available $P$ graph-wise observations, we can form the $N \times P$ matrices

$$
\mathbf{X}_{P}=\left[\mathbf{x}_{1}, \mathbf{x}_{2}, \ldots, \mathbf{x}_{P}\right]
$$

and

$$
\mathbf{Y}_{P}=\left[\mathbf{y}_{1}, \mathbf{y}_{2}, \ldots, \mathbf{y}_{P}\right] .
$$

Notice that here the vectors $\mathbf{y}_{n}$ above have to be calculated, and they are not related to the rearranged signal vectors, defined with the same notation, in the previous section.

### 19.3 Graph Topology Estimation with the Graph Laplacian Energy Condition

In addition to the signal smoothness, it is very useful to introduce the energy of graph Laplacian as an optimization condition, since none of the above conditions is sensitive to the scaling of the graph Laplacian elements and their possibly large values. Such cost function is then of the following form, Dong et al. $(2016,2019)$

$$
J=\sum_{p=1}^{P}\left[\frac{1}{2}\left\|\mathbf{y}_{p}-\mathbf{x}_{p}\right\|_{2}^{2}+\alpha \mathbf{y}_{p}^{T} \mathbf{L} \mathbf{y}_{p}\right]+\beta\|\mathbf{L}\|_{F}^{2},
$$

where the penalty for the energy (squared Frobenius norm of a matrix) of the graph Laplacian, given by

$$
\|\mathbf{L}\|_{F}^{2}=\sum_{m} \sum_{n} L_{m n}^{2}
$$

is involved in order to keep its values as low as possible.
The cost function for the whole set of $P$ observations can now be written in a compact form as

$$
\begin{equation*}
J=\frac{1}{2}\left\|\mathbf{Y}_{P}-\mathbf{X}_{P}\right\|_{F}^{2}+\alpha \operatorname{Trace}\left\{\mathbf{Y}_{P}^{T} \mathbf{L} \mathbf{Y}_{P}\right\}+\beta\|\mathbf{L}\|_{F}^{2} \tag{19.13}
\end{equation*}
$$

where Trace $\left\{\mathbf{Y}_{P}^{T} \mathbf{L} \mathbf{Y}_{P}\right\}$ is a scalar which can be written as

$$
\sum_{p=1}^{P} \mathbf{y}_{p}^{T} \mathbf{L} \mathbf{y}_{p}=\operatorname{Trace}\left\{\mathbf{Y}_{P}^{T} \mathbf{L} \mathbf{Y}_{P}\right\}
$$

The above analysis assumes that the Laplacian has been first normalized. In order to avoid trivial solutions, the condition

$$
\begin{equation*}
\operatorname{Trace}\{\mathbf{L}\}=N \tag{19.14}
\end{equation*}
$$

is also used (as the diagonal elements of the ground truth normalized graph Laplacian are $L_{n n}=1$ ), along with the condition that the offdiagonal elements are either zero or negative, that is

$$
\begin{equation*}
L_{m n}=L_{n m} \leq 0 \quad \text { for } n \neq m . \tag{19.15}
\end{equation*}
$$

As with any Laplacian matrix, the sum of the graph Laplacian elements over every row or column is zero, that is

$$
\begin{equation*}
\sum_{m=0}^{N-1} L_{n m}=0 \quad \text { and } \quad \sum_{n=0}^{N-1} L_{n m}=0 \tag{19.16}
\end{equation*}
$$

Remark 90: The optimization problem in (19.13) aims to learn the graph topology from the graph data by finding the Laplacian of a graph which is most likely to generate the observed graph data. The formulation in (19.13) is obviously jointly convex with respect to both the observed signal and the Laplacian, and can be solved through an iterative two-step procedure, given in Algorithm 3.

Algorithm 3. Iterative procedure for solving the problem of graph learning from data, given in (19.13)
1: Assume that

$$
\mathbf{Y}_{P}=\mathbf{X}_{P}
$$

2: Estimate the graph Laplacian, $\mathbf{L}$, by minimizing

$$
J_{1}=\alpha \operatorname{Trace}\left\{\mathbf{Y}_{P}^{T} \mathbf{L} \mathbf{Y}_{P}\right\}+\|\mathbf{L}\|_{F}^{2}
$$

with the conditions given in (19.14)-(19.16), for the normalized graph Laplacian form.
3: For the Laplacian obtained in the Step 2, the signal $\mathbf{Y}_{P}$ is calculated by minimizing

$$
J_{2}=\frac{1}{2}\left\|\mathbf{Y}_{P}-\mathbf{X}_{P}\right\|_{F}^{2}+\alpha \operatorname{Trace}\left\{\mathbf{Y}_{P}^{T} \mathbf{L} \mathbf{Y}_{P}\right\}
$$

Iteratively repeat Step 2 and Step 3.
Step 3 has a closed form solution explained in Part I.

### 19.4 Learning of Generalized Laplacian-Graphical LASSO

The generalized Laplacian, $\mathbf{Q}$, is defined as Dong et al. $(2016,2019)$

$$
\mathbf{Q}=\alpha \mathbf{I}-\mathbf{N},
$$

where $\mathbf{N}$ is a nonnegative symmetric matrix and $\mathbf{Q}$ is a symmetric positive semidefinite matrix. Any generalized Laplacian can be written as a sum of a standard Laplacian, $\mathbf{L}$, and a diagonal matrix, $\mathbf{P}$, that is

$$
\mathbf{Q}=\mathbf{L}+\mathbf{P} .
$$

Remark 91: The generalized Laplacian allows for the existence of self-loops on the vertices; these self-loops are defined by matrix $\mathbf{P}$.
Example 78: For the data in Example 74, the precision matrix is of the form

$$
\mathbf{C}=\mathbf{R}_{x}^{-1}=\left[\begin{array}{rrrr}
2 & -1 & 0 & 0 \\
-1 & 2 & -1 & 0 \\
0 & -1 & 2 & -1 \\
0 & 0 & -1 & 1
\end{array}\right]
$$

It may be considered as a generalized graph Laplacian since

$$
\begin{aligned}
\mathbf{R}_{x}^{-1} & =\left[\begin{array}{rrrr}
2 & -1 & 0 & 0 \\
-1 & 2 & -1 & 0 \\
0 & -1 & 2 & -1 \\
0 & 0 & -1 & 1
\end{array}\right] \\
& =\left[\begin{array}{rrrr}
1 & -1 & 0 & 0 \\
-1 & 2 & -1 & 0 \\
0 & -1 & 2 & -1 \\
0 & 0 & -1 & 1
\end{array}\right]+\left[\begin{array}{llll}
1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{array}\right] \\
& =\mathbf{L}+\mathbf{P} .
\end{aligned}
$$

This means that $\mathbf{R}_{x}^{-1}$ in this example may be interpreted as standard graph Laplacian with a self-loop at the vertex $n=0$.

We will next show that owing to its physically relevant properties, the precision matrix, $\mathbf{C}=\mathbf{R}_{x}^{-1}$, can be used as an estimate of the generalized Laplacian, $\mathbf{Q}$.
Estimation of graph Laplacian through precision matrix. Consider a set of noisy signals $x_{p}(n)$ acquired over $P$ observations, $p=$ $1,2, \ldots, P$, at $N$ vertices, $n=0,1 \ldots, N-1$, of an undirected graph. Our aim is to learn the graph connectivity (its Laplacian) based on the condition that the observed graph signal in the $p$ th realization, $\mathbf{x}_{p}$, is
as smooth as possible on the graph defined by a generalized Laplacian, Q, as explained in Remark 88. The cost function to achieve this goal can be conveniently defined by the signal smoothness function

$$
J_{p}=\mathbf{x}_{p}^{T} \mathbf{Q} \mathbf{x}_{p}, \quad \text { for } p=1,2, \ldots, P .
$$

The cumulative smoothness for all data $\mathbf{x}_{p}, p=1,2, \ldots, P$, is then expressed as

$$
\begin{equation*}
J=\frac{1}{P} \sum_{p=1}^{P} \mathbf{x}_{p}^{T} \mathbf{Q} \mathbf{x}_{p} \tag{19.17}
\end{equation*}
$$

while the correlation matrix of the all considered observations can be written as

$$
\begin{aligned}
\mathbf{R}_{x} & =\frac{1}{P} \sum_{p=1}^{P} \mathbf{x}_{p} \mathbf{x}_{p}^{T} \\
& =\frac{1}{P}\left[\mathbf{x}_{1}, \mathbf{x}_{2}, \ldots, \mathbf{x}_{P}\right]\left[\mathbf{x}_{1}, \mathbf{x}_{2}, \ldots, \mathbf{x}_{P}\right]^{T} \\
& =\frac{1}{P} \mathbf{X}_{P} \mathbf{X}_{P}^{T}
\end{aligned}
$$

The smoothness index for all observations is now of the following form

$$
J=\frac{1}{P} \sum_{p=1}^{P} \mathbf{x}_{p}^{T} \mathbf{Q} \mathbf{x}_{p}=\operatorname{Trace}\left\{\mathbf{R}_{x} \mathbf{Q}\right\}
$$

since

$$
\begin{aligned}
J & =\frac{1}{P} \sum_{p=1}^{P} \mathbf{x}_{p}^{T} \mathbf{Q} \mathbf{x}_{p} \\
& =\frac{1}{P} \operatorname{Trace}\left\{\left[\mathbf{x}_{1}, \mathbf{x}_{2}, \ldots, \mathbf{x}_{P}\right]^{T} \mathbf{Q}\left[\mathbf{x}_{1}, \mathbf{x}_{2}, \ldots, \mathbf{x}_{P}\right]\right\} \\
& =\frac{1}{P} \operatorname{Trace}\left\{\mathbf{X}_{P}^{T} \mathbf{Q} \mathbf{X}_{P}\right\}=\frac{1}{P} \operatorname{Trace}\left\{\mathbf{X}_{P} \mathbf{X}_{P}^{T} \mathbf{Q}\right\} \\
& =\operatorname{Trace}\left\{\mathbf{R}_{x} \mathbf{Q}\right\}
\end{aligned}
$$

To avoid a trivial solution, the conditions for the generalized Laplacian should be incorporated. For symmetric positive definite matrices, all eigenvalues are positive, and since for any matrix, $\mathbf{Q}$, the product
of its eigenvalues is equal to $\operatorname{det}(\mathbf{Q})$, this condition can be included by adding the term $\ln (\operatorname{det}(\mathbf{Q}))$ to the cost function, to give

$$
\begin{equation*}
J=-\ln (\operatorname{det}(\mathbf{Q}))+\operatorname{Trace}\left\{\mathbf{R}_{x} \mathbf{Q}\right\} \tag{19.18}
\end{equation*}
$$

Maximum likelihood interpretation. The interpretation of the cost function in (19.18) within the theory of Gaussian random signals and maximum likelihood estimation is given in Section 21.8. If we assume that the graph data at $N$ vertices are $N$-dimensional random variables, with zero-mean and an unknown precision matrix $\mathbf{Q}$, then their N -dimensional probability density function is given by

$$
P\left(\mathbf{x}_{p}\right)=\frac{1}{\sqrt{(2 \pi)^{p}}} \sqrt{\operatorname{det}(\mathbf{Q})} \exp \left(-\frac{1}{2} \mathbf{x}_{p}^{T} \mathbf{Q} \mathbf{x}_{p}\right)
$$

Within the maximum likelihood framework, the goal is to find the unknown parameter (matrix) $\mathbf{Q}$ so that the distribution fits the data in an optimal form. This optimal parameter matrix is obtained by differentiating the probability or its logarithm (log-likelihood) function,

$$
\begin{align*}
-\ln \left\{P\left(\mathbf{x}_{p}\right) \sqrt{(2 \pi)^{p}}\right\} & =-\ln \left\{\sqrt{\operatorname{det}(\mathbf{Q})} \exp \left(-\frac{1}{2} \mathbf{x}_{p}^{T} \mathbf{Q} \mathbf{x}_{p}\right)\right\} \\
& =-\frac{1}{2} \ln \{\operatorname{det}(\mathbf{Q})\}+\frac{1}{2} \mathbf{x}_{p}^{T} \mathbf{Q} \mathbf{x}_{p} \tag{19.19}
\end{align*}
$$

and setting the obtained derivative to zero.
Example 79: The concept of finding the best precision, $Q$, the reciprocal of the variance of Gaussian distribution, $Q=1 / \sigma^{2}$, to fit the data will be now illustrated on a simple setup. Assume that four observations of signal $x_{p}(n), p=1,2,3,4$, at the vertex $n=0$ are available, and are given by $x_{1}(0)=0.2, x_{2}(0)=-0.3, x_{3}(0)=-0.4$, and $x_{4}(0)=-0.5$. It is also known that the data are zero-mean. The goal is to find the precision, $Q=1 / \sigma^{2}$, or variance, $\sigma^{2}$, of the Gaussian distribution of the observed data, given by

$$
P\left(x_{p}(0)\right)=\frac{1}{\sigma \sqrt{2 \pi}} \exp \left(-\frac{x_{p}^{2}(0)}{2 \sigma^{2}}\right)=\sqrt{\frac{Q}{2 \pi}} \exp \left(-\frac{1}{2} x_{p}(0) Q x_{p}(0)\right)
$$

which corresponds to the best fit to the observed data. The log-likelihood function of the joint distribution of these four observed data points is
then

$$
\begin{aligned}
J & =-\ln \left(P\left(x_{1}(0)\right) P\left(x_{2}(0)\right) P\left(x_{3}(0)\right) P\left(x_{4}(0)\right)\right) \\
& =-\ln \left(\frac{1}{4 \pi^{2}} Q^{2} e^{-\frac{1}{2} 0.2^{2} Q} e^{-\frac{1}{2} 0.3^{2} Q} e^{-\frac{1}{2} 0.4^{2} Q} e^{-\frac{1}{2} 0.5^{2} Q}\right) \\
& =2 \ln (2 \pi)-2 \ln (Q)+\frac{1}{2}\left(0.2^{2}+0.3^{2}+0.4^{2}+0.5^{2}\right) Q \\
& =2 \ln (2 \pi)-2 \ln (Q)+\frac{1}{2} 0.54 Q
\end{aligned}
$$

The differentiation of this expression with respect to $Q=1 / \sigma^{2}$ produces $-2 / Q+\frac{1}{2} 0.54=0$ or $Q=4 / 0.54=7.4$ and

$$
\sigma=\sqrt{1 / Q}=0.36
$$

The same value would have been produced by a simple standard deviation estimator, $\sigma=\sqrt{\left(0.2^{2}+0.3^{2}+0.4^{2}+0.5^{2}\right) / 4}$.

Example 80: Similar analysis, as in the previous example, can be performed for $P$ observations at two vertices, $n=0$ and $n=1$, $\left[x_{p}(0), x_{p}(1)\right]^{T}$. The goal is to estimate the parameters of the precision matrix

$$
\mathbf{Q}=\left[\begin{array}{ll}
Q_{11} & Q_{12} \\
Q_{21} & Q_{22}
\end{array}\right]
$$

of the joint Gaussian distribution of $\left[x_{p}(0), x_{p}(1)\right]^{T}$, defined as

$$
\begin{align*}
P\left(\left[x_{p}(0), x_{p}(1)\right]^{T}\right)= & \frac{\sqrt{\operatorname{det}(\mathbf{Q})}}{2 \pi} e^{-\frac{1}{2}\left[x_{p}(0), x_{p}(1)\right] \mathbf{Q}\left[x_{p}(0), x_{p}(1)\right]^{T}}  \tag{19.20}\\
= & \frac{\sqrt{Q_{11} Q_{22}-Q_{12} Q_{21}}}{2 \pi} \\
& \times e^{-\frac{1}{2}\left(Q_{11} x_{p}^{2}(0)+\left(Q_{12}+Q_{21}\right) x_{p}(0) x_{p}(1)+Q_{22} x_{p}^{2}(1)\right)} . \tag{19.21}
\end{align*}
$$

Using $P$ available realizations,

$$
\left[x_{1}(0), x_{1}(1)\right],\left[x_{2}(0), x_{2}(1)\right], \ldots,\left[x_{P}(0), x_{P}(1)\right]
$$

and the corresponding $P$-variate normal distribution of two variables as a product of $P$ distributions as in (19.21), we can find the parameters $Q_{11}, Q_{12}, Q_{21}, Q_{22}$ which produce the best distribution fit, using the partial derivatives of the log-likelihood function.

For example, a partial derivative of the log-likelihood function with respect to $Q_{11}$ would produce

$$
-\frac{P}{2} \frac{Q_{22}}{\sqrt{Q_{11} Q_{22}-Q_{12} Q_{21}}}+\frac{1}{2}\left(x_{1}^{2}(0)+x_{2}^{2}(0)+\cdots+x_{P}^{2}(0)\right)=0 .
$$

Observe that the term

$$
\frac{Q_{22}}{\sqrt{Q_{11} Q_{22}-Q_{12} Q_{21}}}=\frac{Q_{22}}{\sqrt{\operatorname{det}(\mathbf{Q})}}
$$

is just the first element of the inverse of matrix $\mathbf{Q}$, while the term $\left(x_{1}^{2}(0)+x_{2}^{2}(0)+\cdots+x_{P}^{2}(0)\right)$ is the first element of the correlation matrix $\mathbf{R}_{x}$, multiplied by $P$. In a similar way, the derivations over $Q_{12}$, $Q_{21}$, and $Q_{22}$, will produce the remaining elements of the inverse of matrix $\mathbf{Q}$ and the correlation matrix $\mathbf{R}_{x}$. In the matrix notation, the solution to the so obtained system of the four equations is given by

$$
\mathbf{Q}^{-1}=\frac{1}{P}\left[\begin{array}{ll}
\sum_{p=1}^{P} x_{p}^{2}(0) & \sum_{p=1}^{P} x_{p}(0) x_{p}(1) \\
\sum_{p=1}^{P} x_{p}(1) x_{p}(0) & \sum_{p=1}^{P} x_{p}^{2}(1)
\end{array}\right]=\mathbf{R}_{x} .
$$

Notice that at least $P=2$ independent observations, $P \geq N$, are needed, since for $P=1$ observation, $P<N$, and the rank of the correlation matrix, $\mathbf{R}_{x}$, would be 1, which is lower than its dimension. In that case, the correlation matrix would not be invertible.

The cost function in (19.18) minimizes the logarithm of the joint probability density function of a graph signal $\mathbf{x}_{p}$ under the Gaussian assumption. The minimization of the cost function $J$ with respect to $\mathbf{Q}$, with $\partial J / \partial \mathbf{Q}=\mathbf{0}$, produces

$$
\begin{equation*}
\frac{\partial J}{\partial \mathbf{Q}}=\frac{\partial}{\partial \mathbf{Q}}\left(-\ln (\operatorname{det}(\mathbf{Q}))+\operatorname{Trace}\left\{\mathbf{R}_{x} \mathbf{Q}\right\}\right) . \tag{19.22}
\end{equation*}
$$

In order to find this derivative, we will use the relation among the trace of a positive semidefinite matrix, its eigenvalues, $\lambda_{k}$, and the trace of the eigenvalue matrix, $\boldsymbol{\Lambda}$, in the form

$$
\begin{align*}
\ln (\operatorname{det}(\mathbf{Q})) & =\sum_{k=1}^{N} \ln \left(\lambda_{k}\right) \\
& =\operatorname{Trace}(\ln (\mathbf{\Lambda}))=\operatorname{Trace}(\ln (\mathbf{Q})) . \tag{19.23}
\end{align*}
$$

Note also that for a differentiable matrix function, $f(\mathbf{Q})$, the following holds

$$
\begin{equation*}
\frac{\partial}{\partial \mathbf{Q}}(\operatorname{Trace}\{f(\mathbf{Q})\})=\frac{\partial f(\mathbf{Q})}{\partial \mathbf{Q}} \tag{19.24}
\end{equation*}
$$

Having in mind the properties in (19.23) and (19.24), we can write

$$
\begin{equation*}
\frac{\partial J}{\partial \mathbf{Q}}=-\mathbf{Q}^{-1}+\mathbf{R}_{x} \tag{19.25}
\end{equation*}
$$

The best estimate of $\mathbf{Q}$ follows from $\partial J / \partial \mathbf{Q}=\mathbf{0}$, and has the form

$$
\begin{equation*}
\mathbf{Q}=\mathbf{R}_{x}^{-1} \tag{19.26}
\end{equation*}
$$

Remark 92: The solution in (19.26), being equal to the precision matrix, can be used as the generalized Laplacian estimate in order to obtain the underlying graph structure.
Example 81: The weight matrix which corresponds to the inverse of the correlation matrix $\mathbf{R}_{x}$, for which the positive and small off-diagonal values were set to zero, is shown in Figure 8.4(b). Here, we consider the graph from Figure 2.2 in Part I and $P=10,000$ observations. The observations were simulated by assuming white Gaussian external sources with zero-mean and variance $\sigma^{2}=1$, located at a randomly chosen vertex (as described in more detail in Section 19).


Figure 8.4: Weight matrix for the graph from Figure 2.2 in Part I. (a) Ground truth weight matrix. (b) Estimated weight matrix using the inverse correlation (precision) matrix.

Remark 93: Notice that the correlation matrix, $\mathbf{R}_{x}$, may be singular. The correlation matrix, $\mathbf{R}_{x}$, is always singular when the number of observations, $P$, is lower than the number vertices (dimension of the correlation matrix, $N$ ) that is, $N>P$. This follows from the fact that the correlation matrix is formed as a combination of $P$ signals, $\mathbf{R}_{x}=\frac{1}{P} \sum_{p=1}^{P} \mathbf{x}_{p} \mathbf{x}_{p}^{T}$, which means that its dimensionality is spanned over at most $P$ independent vectors (eigenvectors), and therefore its rank is equal to or lower than $P$ (see Example 88 in Section 20).

Also, this form will not produce a matrix satisfying the conditions for a generalized Laplacian. The inverse correlation function may also have positive off-diagonal values. Therefore, for a reliable solution, the cost function in (19.18) should have additional constraints. Here, we will present two of such constraints.

Graphical LASSO. In this approach, the classical reconstruction formulation of a sparse signal is used as the additional constraint on the precision matrix and the cost function from (19.18) (Friedman et al., 2008). The sparsity constraint on the generalized Laplacian is added to achieve the solution with the smallest possible number of nonzero entries in the estimated graph weight matrix - the smallest number of edges. The sparsity condition also allows for the problem solution with a reduced correlation matrix rank (as within the compressive sensing framework described in Part II). The cost function, with the included sparsity penalty function, $\|\mathbf{Q}\|_{1}$, is then defined as

$$
\begin{equation*}
J=-\ln (\operatorname{det}(\mathbf{Q}))+\operatorname{Trace}\left\{\mathbf{R}_{x} \mathbf{Q}\right\}+\rho\|\mathbf{Q}\|_{1} . \tag{19.27}
\end{equation*}
$$

This minimization problem can be solved in many ways, one of which is the graphical LASSO algorithm, an extension of the standard LASSO algorithm to graph problems (see Algorithm 4 for the implementation and Section 20 for the derivation of graphical LASSO).
Example 82: For the same signal as in Example 81, the weight matrix obtained using the graphical LASSO,

$$
\mathbf{W}=\operatorname{glasso}\left(\mathbf{R}_{x}, 0.3\right),
$$

where both positive and small element values are set to zero, is shown in Figure 8.5(b) (see also Example 88).

```
Algorithm 4. Graphical LASSO, \(\mathbf{Q}=\operatorname{glasso}(\mathbf{R}, \rho)\)
Input:
    - Correlation matrix \(\mathbf{R}\)
    - Regularization parameter \(\rho\)
    \(M_{i} \leftarrow 100, E_{p} \leftarrow 0.0001\)
    \([p, n] \leftarrow \operatorname{size}(\mathbf{R})\)
    \(C_{p} \leftarrow \operatorname{mean}(|\mathbf{R}-\operatorname{diag}(\operatorname{diag}(\mathbf{R}))|) E_{p}\)
    \(\mathbf{V}_{0}=\mathbf{R}+\rho \mathbf{I}\)
    \(\mathbf{V}=\mathbf{V}_{0}\)
    for \(r=1\) to \(M_{i}\) do
    for \(j=p\) to 1 step -1 do
        \(\mathbf{V}_{11} \leftarrow \mathbf{V}\)
        \(\mathbf{V}_{11} \leftarrow \mathbf{V}_{11}\) with removed \(j\) th row
        \(\mathbf{V}_{11} \leftarrow \mathbf{V}_{11}\) with removed \(j\) th column
        \(v_{22} \leftarrow V(j, j)\)
        \(\mathbf{r}_{12} \leftarrow j\) th column of \(\mathbf{R}\)
        \(\mathbf{r}_{12} \leftarrow \mathbf{r}_{12}\) with removed \(j\) th element
        \(\mathbf{A} \leftarrow \sqrt{\mathbf{V}_{11}}\)
        \(\mathbf{b} \leftarrow\left(\sqrt{\mathbf{V}_{11}}\right)^{-1} \mathbf{r}_{12}\)
        \(\boldsymbol{\beta}=\operatorname{lasso}(\mathbf{A}, \mathbf{b}, \rho)\), as in Algorithm 2
        \(\mathbf{v}_{12} \leftarrow \mathbf{V}_{11} \boldsymbol{\beta}\)
        \(\mathbf{V} \leftarrow \mathbf{V}\) with \(\mathbf{v}_{12}\) inserted as the \(j\) th column
        \(\mathbf{v}_{12} \leftarrow \mathbf{v}_{12}^{T}\) with \(v_{22}\) inserted as the \(j\) th element
        \(\mathbf{V} \leftarrow \mathbf{V}\) with \(\mathbf{v}_{12}\) inserted as the \(j\) th row
        end for
        if mean \(\left(\left|\mathbf{V}-\mathbf{V}_{0}\right|<C_{p}\right)\) break, end
        \(\mathbf{V}_{0}=\mathbf{V}\)
    end for
    \(\mathbf{Q}=\mathbf{V}^{-1}\)
    - Estimated precision matrix \(\mathbf{Q}\)
```

Generalized Laplacian constrained approach. Another possible approach employs the Lagrange multipliers, $\mathbf{B}$, which are added in such


Figure 8.5: Weight matrix for the graph from Figure 2.2 in Part I. (a) Ground truth weight matrix. (b) Estimated weight matrix using the graphical LASSO and inverse correlation (precision) matrix.
a way that these values do not change the diagonal elements of $\mathbf{Q}$, and ensure that all

$$
Q_{m n}=Q_{n m} \leq 0
$$

for $n \neq m$, with $B_{n m}=B_{m n} \geq 0$. The diagonal elements of matrix $\mathbf{B}$ are therefore $B_{n n}=0$. Finally, the condition $B_{n m} Q_{n m}=0$ for all $n$ and $m$ is used. In this case, the minimization solution for the generalized Laplacian is obtained as

$$
\mathbf{Q}=\left(\mathbf{R}_{x}+\mathbf{B}\right)^{-1}
$$

based on the cost function

$$
J=-\ln (\operatorname{det}(\mathbf{Q}))+\operatorname{Trace}\left\{\mathbf{R}_{x} \mathbf{Q}\right\}+\operatorname{Trace}\{\mathbf{B} \mathbf{Q}\} .
$$

The results obtained in this case are similar to those obtained with the graphical LASSO approach.

### 19.5 Graph Topology Learning Based on the Eigenvectors

Assume that the available observations of a graph signal, $x_{p}(n)$, are graph wide sense stationary (GWSS), that is, they can be considered as the output of a linear system $H(\mathbf{L})$, driven by white noise, $\boldsymbol{\varepsilon}_{p}$, as the input. In other words, the signal on a graph is formed using a linear combination of a white noise realization, $\boldsymbol{\varepsilon}_{p}$, and its graph shifted
versions. The output signal after $M$ such graph shifts, defined by the normalized Laplacian, is given by

$$
\begin{equation*}
\mathbf{x}_{p}=\left(h_{M} \mathbf{L}^{M}+h_{M-1} \mathbf{L}^{M-1}+\cdots+h_{1} \mathbf{L}^{1}+h_{0} \mathbf{L}^{0}\right) \boldsymbol{\varepsilon}_{p} \tag{19.28}
\end{equation*}
$$

This graph signal can be written in a compact form

$$
\mathbf{x}_{p}=H(\mathbf{L}) \varepsilon_{p},
$$

with its correlation matrix given by (for $\sigma_{\varepsilon}^{2}=1$ )

$$
\begin{align*}
\mathbf{R}_{x} & =\frac{1}{P} \sum_{p=1}^{P} \mathbf{x}_{p} \mathbf{x}_{p}^{T}=\frac{1}{P} \sum_{p=1}^{P} H(\mathbf{L}) \varepsilon_{p} \varepsilon_{p}^{T} H^{T}(\mathbf{L}) \\
& =H(\mathbf{L})\left(\frac{1}{P} \sum_{p=1}^{P} \varepsilon_{p} \varepsilon_{p}^{T}\right) H^{T}(\mathbf{L}) \\
& =H(\mathbf{L}) H^{T}(\mathbf{L})=\mathbf{U}^{T}|H(\mathbf{\Lambda})|^{2} \mathbf{U} \tag{19.29}
\end{align*}
$$

where $\varepsilon_{p}$ is a white noise with unit variance, and $\mathbf{U}$ is the matrix of graph Laplacian eigenvectors, $\mathbf{L}=\mathbf{U}^{T} \boldsymbol{\Lambda} \mathbf{U}$.

From (19.29), it is now obvious that we can learn about the graph eigenvectors from the decomposition of the autocorrelation matrix. The same holds for the precision matrix, $\mathbf{Q}=\mathbf{R}_{x}^{-1}$, since the inverse matrix has the same eigenvectors as the original matrix.

For the normalized graph Laplacian, it is straightforward to relate the Laplacian, $\mathbf{L}_{N}$, based shift and the normalized weight matrix, $\mathbf{W}_{N}$, based shift since

$$
\mathbf{L}_{N}^{p}=\left(\mathbf{I}-\mathbf{W}_{N}\right)^{p}=\mathbf{I}-p \mathbf{W}_{N}+\cdots+(-1)^{p} \mathbf{W}_{N}^{p} .
$$

Therefore from (19.29), in order to estimate the graph connectivity (estimate its Laplacian or adjacency matrix) we can use the eigenvectors of the autocorrelation matrix.

Remark 94: Since we do not know $H(\boldsymbol{\Lambda})$, it is natural to assume that the graph is defined by the eigenvalues, $\boldsymbol{\Lambda}$, that produce the smallest number of edges. This can be achieved by minimizing the number of nonzero values in $\mathbf{L}$ for the given eigenvectors (Marques et al., 2017; Segarra et al., 2017).

The minimization problem of determining a graph now becomes

$$
\begin{equation*}
\min _{\lambda_{k}}\|\mathbf{L}\|_{0} \text { subject to } \mathbf{L}=\sum_{k=0}^{N-1} \lambda_{k} \mathbf{u}_{k} \mathbf{u}_{k}^{T} \tag{19.30}
\end{equation*}
$$

while the convex (norm-one) form of this minimization problem is

$$
\begin{equation*}
\min _{\lambda_{k}}\|\mathbf{L}\|_{1} \text { subject to } \mathbf{L}=\sum_{k=0}^{N-1} \lambda_{k} \mathbf{u}_{k} \mathbf{u}_{k}^{T} \tag{19.31}
\end{equation*}
$$

Remark 95: The convex norm-one based form in (19.31) can produce the same solution as the original norm-zero form in (19.30) if the Laplacian sparsity is low and the Laplacian satisfies some other conditions (in the sense discussed in Section 19.2).

Since the eigenvectors are obtained from the decomposition of the correlation matrix, spectral analysis performed in this way is related to principal component analysis (PCA), where the signal is decomposed through the set of eigenvectors of the correlation matrix.

This approach to graph topology learning can be summarized through the following steps.

1. For a given set of graph signal observations, $\mathbf{x}_{p}, p=1,2, \ldots, P$, calculate the correlation matrix

$$
\begin{equation*}
\mathbf{R}_{x}=\frac{1}{P} \sum_{p=1}^{P} \mathbf{x}_{p} \mathbf{x}_{p}^{T} \tag{19.32}
\end{equation*}
$$

2. Perform the eigen decomposition of the correlation matrix, in the form

$$
\begin{align*}
\mathbf{R}_{x} & =\mathbf{U}^{T} \boldsymbol{\Lambda}_{R_{x}} \mathbf{U} \\
\boldsymbol{\Lambda}_{R_{x}} & =\mathbf{U R}_{x} \mathbf{U}^{T} \tag{19.33}
\end{align*}
$$

3. Find the eigenvalues, $\lambda_{k}$, of the graph Laplacian, $\mathbf{L}=\mathbf{U}^{T} \boldsymbol{\Lambda} \mathbf{U}$, such that it assumes the sparsest possible form, using the minimization

$$
\begin{equation*}
\min _{\lambda_{k}}\|\mathbf{L}\|_{1} \text { subject to } \mathbf{L}=\sum_{k=0}^{N-1} \lambda_{k} \mathbf{u}_{k} \mathbf{u}_{k}^{T} \tag{19.34}
\end{equation*}
$$

Dimensionality-reduction methods. It is often reasonable to assume that the observed graph signals are generated by exciting a low-order graph system with white noise as the input. However, the problem of estimating the polynomial coefficients from its samples at unknown (eigenvalue) positions is under-determined and cannot be directly solved. By adding the constraint that true eigenvalue positions should produce a sparse graph Laplacian, the solution becomes tractable within the compressive sensing framework (Stanković et al., 2020a).

In this way, instead of the minimization over $N$ variables, $\lambda_{k}, k=$ $0,1, \ldots, N-1$, we can find the Laplacian eigenvalues starting from the eigendecomposition of the correlation matrix of a signal produced by a system on a graph, that is,

$$
\begin{equation*}
\mathbf{R}_{x}=\mathbf{U}|H(\boldsymbol{\Lambda})|^{2} \mathbf{U}^{T}=\mathbf{U} \boldsymbol{\Lambda}_{R_{x}} \mathbf{U}^{T} . \tag{19.35}
\end{equation*}
$$

Assume that the transfer function of the graph system is of a polynomial form

$$
\begin{equation*}
H\left(\lambda_{k}\right)=h_{0}+h_{1} \lambda_{k}+h_{2} \lambda_{k}^{2}+\cdots+h_{M} \lambda_{k}^{M} \tag{19.36}
\end{equation*}
$$

with $M \ll N$. From the correlation matrix eigendecomposition in (19.35), we have $N$ values of $H\left(\lambda_{k}\right)$ obtained as square roots of the eigenvalues of the correlation matrix, $\lambda_{k}^{\left(\mathbf{R}_{x}\right)}$. Without loss of generality, we will assume a nondecreasing $H\left(\lambda_{k}\right)$, that is $H\left(\lambda_{k-1}\right) \leq H\left(\lambda_{k}\right)$. The problem now boils down to the determination of the Laplacian eigenvalues, $\lambda_{k}, k=0,1, \ldots, N-1$, having in mind that $\lambda_{0}=0$, $\sum_{k=0}^{N-1} \lambda_{k}=N$ and that there exist (unknown) coefficients $h_{i}, i=$ $0,1, \ldots, M$ such that (19.36) is satisfied for each $k$, while the true values $\lambda_{k}$ produce the sparsest graph Laplacian, $\mathbf{L}$.

The estimation of the system coefficients, Laplacian eigenvalues and Laplacian itself is performed using this polynomial fitting method in the following way.

1. Select $(M+1)$ indices $m_{0}=0<m_{1}<\cdots<m_{M}=N$ with the corresponding transfer function values $H\left(\lambda_{m_{i}}\right)$, for $i=0,1, \ldots, M$. Assume that $(M+1)$ eigenvalues are $\bar{\lambda}_{0}=0, \bar{\lambda}_{m_{1}}=\xi_{1}, \bar{\lambda}_{m_{2}}=$ $\xi_{2}, \ldots, \bar{\lambda}_{m_{M-1}}=\xi_{M-1}, \bar{\lambda}_{m_{M}}=1$, where $0<\xi_{1}<\xi_{2}<\cdots<$ $\xi_{M-1}<1$.
2. Then, the coefficients of an $M$ th order polynomial

$$
P(\bar{\lambda})=a_{0}+a_{1} \bar{\lambda}+a_{2} \bar{\lambda}^{2}+\cdots+a_{M} \bar{\lambda}^{M}
$$

can be found such that $P\left(\hat{\lambda}_{i}\right)=H\left(\lambda_{m_{i}}\right)$, for $i=0,1, \ldots, M$, is a Lagrange polynomial of $M$ th order defined by $(M+1)$ points.
3. Now, the eigenvalues $\bar{\lambda}_{k}$, for each $k$, can be calculated as a solution of

$$
P(\bar{\lambda})=H\left(\lambda_{k}\right), \quad 0 \leq \bar{\lambda} \leq 1
$$

for the unknown $\bar{\lambda}$. Note that this solution is unique if the polynomial $P(\hat{\lambda})$ is an increasing function for $0 \leq \hat{\lambda} \leq 1$.
4. Having in mind that $\sum_{k=0}^{N-1} \lambda_{k}=N$, the eigenvalues, $\hat{\lambda}_{k}$, can be found by scaling the obtained values, $\bar{\lambda}_{k}$, for each $k$, as $\hat{\lambda}_{k}=$ $N \bar{\lambda}_{k} / \sum_{k=0}^{N-1} \bar{\lambda}_{k}$.
5. For the so obtained estimates of the eigenvalues, $\hat{\lambda}_{k}$, the normalized graph Laplacian can be calculated as $\mathbf{L}=\mathbf{U} \hat{\boldsymbol{\Lambda}} \mathbf{U}^{T}$, where $\hat{\boldsymbol{\Lambda}}$ is a diagonal matrix with $\hat{\lambda}_{k}$ on the diagonal.
6. The above procedure should be repeated for various $0<\xi_{1}<\xi_{2}<$ $\cdots<\xi_{M-1}<1$ and the final solution is obtained by minimizing the energy normalized sparsity condition, given by

$$
\min _{\xi_{1}, \xi_{2}, \ldots, \xi_{M-1}} \frac{\|\mathbf{L}\|_{1}}{\sqrt{\|\mathbf{L}\|_{2}}}
$$

Notice that for $M=1$, we should consider only two points in Step 1, and there is no need for the minimization of variables $\xi_{i}$. For $M=2$, we have one minimization variable $0<\xi_{1}<1$. For $M=3$, the minimization is performed over only two variables, $0<\xi_{1}<\xi_{2}<1$. Given that dimensionality of the minimization problem is $(M-1)$, and since $M \ll N$, the dimensionality reduction of this method when compared to (19.34) is evident.

The spectral indices $0=m_{0}, m_{1}, \ldots, m_{M}=N$, selected in Step 1, should be equally spaced over $N$ possible values. For $M=2$, the index $m_{1}$ should be close to $(N-1) / 2$, while for $M=3$ the indices $m_{1}$ and $m_{2}$ should be close to $(N-1) / 3$ and $2(N-1) / 3$, respectively.

Example 83: Consider a graph with $N=8$ vertices, for which the weight matrix is given in Figure 8.6(a). An $N \times P$ matrix of the simulated signal, $\mathbf{X}_{P}$, was formed by calculating the graph signal as in (19.28), with a given graph, its weight matrix, $\mathbf{W}=\mathbf{I}-\mathbf{L}$, the normalized Laplacian, $\mathbf{L}$, system order $M$, and system coefficients, $h_{0}, h_{1}, \ldots, h_{M}$. White Gaussian external sources, $\boldsymbol{\varepsilon}_{p}$, with zero-mean and variance $\sigma^{2}=1$ were assumed in all $P=10,000$ realizations.

The presented polynomial fitting method was implemented for the assumed degree $M=2$ of the polynomial $H(\lambda)$, with $h_{0}=0.3, h_{1}=0.2$, and $h_{2}=0.5$ used in the graph signal simulation, according to (19.28). By forming $\mathbf{R}_{x}$ from $\mathbf{X}_{P}$ and after its eigendecomposition, the eigenvectors $\mathbf{U}$ were estimated, while the eigenvalues of the correlation matrix were used to calculate $H\left(\lambda_{k}\right)=\sqrt{\lambda_{k}^{\left(\mathbf{R}_{x}\right)}}$.

Observe that the polynomial fitting method reduces to the onedimensional minimization over variable $0<\xi_{1}<1$, as shown in Figure 8.7. After the minimum value of the sparsity measure is found, the eigenvalues are calculated with the corresponding parameter, $\xi_{1}$. The Laplacian then follows from $\mathbf{L}=\mathbf{U}^{T} \boldsymbol{\Lambda} \mathbf{U}$.

In this case, the obtained error in the weight matrix elements (absolute value of the off-diagonal elements of the Laplacian) is characterized by the $\mathrm{MSE}=-35.1 \mathrm{~dB}$, with the results presented in Figure 8.6. The true weight matrix, $\mathbf{W}=\mathbf{I}-\mathbf{L}$, along with the estimated one, is given in Figures 8.6(a) and (b), and the sparsity measure function is plotted in Figure 8.6(c), while the true and the estimated Laplacian eigenvalues are given in Figure 8.6(d).

Example 84: The experiment from Example 83 was repeated for a low number of observations, $P=8 N_{L}=256$, where $N_{L}=32$ is the sparsity of the Laplacian matrix according to practical hints for the number of measurements and sparsity (Candès et al., 2006). The reconstruction using the polynomial fitting produced the MSE $=-18.0 \mathrm{~dB}$.

In this experiment, we assumed $M=3$ and $h_{0}=0.4, h_{1}=0.5$, $h_{2}=0.4$, and $h_{3}=0.2$ when simulating the graph signal, $\mathbf{X}_{P}$. The correlation matrix was estimated using this simulated signal, along with its eigenvectors and eigenvalues. We now have two minimization variables $\xi_{1}$ and $\xi_{2}, 0<\xi_{1}<\xi_{2}<1$. The results for the polynomial fitting


Figure 8.6: Estimation of the weight matrix, $\mathbf{W}=\mathbf{I}-\mathbf{L}$, for the graph with $N=8$ vertices. (a) Ground truth weight matrix. (b) Estimated weight matrix using sparsity minimization of the normalized Laplacian. (c) Sparsity measure minimization, as a function of parameter $\xi_{1}$. (d) The exact (blue lines) and estimated (red crosses) eigenvalues of the normalized Laplacian.
method are presented in Figures 8.8(a)-(d). The obtained estimation error was $\mathrm{MSE}=-34.9 \mathrm{~dB}$. The sparsity measure function (Figure 8.8(c)) is now two-dimensional and is calculated only when unique solutions are obtained in Step 3 of the polynomial fitting method. These results were compared with those obtained using the rows of the correlation matrix, $\boldsymbol{\beta}_{n}=\operatorname{lasso}\left(\mathbf{Y}_{n}^{T}, \mathbf{y}_{n}^{T}, 0.2\right)$ (Figure 8.8(e)) and graphical LASSO, $\mathbf{Q}=\operatorname{glasso}\left(\mathbf{R}_{x}, 0.3\right)$ (Figure 8.8(f)), with the optimized values of the parameter $\rho$. In these cases, the obtained error in the weight matrix elements was characterized by $\mathrm{MSE}=-10.3 \mathrm{~dB}$ and $\mathrm{MSE}=-14.5 \mathrm{~dB}$, respectively.


Figure 8.7: Illustration of eigenvalue calculation based on their second order polynomial, obtained from $H\left(\lambda_{k}\right)=\sqrt{\lambda_{k}^{\left(\mathbf{R}_{x}\right)}}$.

Example 85: Finally, the polynomial fitting method was tested on a larger scale graph, with $N=50$ and $M=2$. The original and estimated weight matrices are shown in Figure 8.9.

So far, the examples related to classical data analytics have used Fourier analysis and a circular directed graph. The problem formulation presented in this section can also be used to define a graph such that the spectral analysis on this graph leads to some other well known transforms.

Example 86: We shall illustrate the method of defining a graph which corresponds to the classical Hadamard transform with $N=8$, and with


Figure 8.8: Estimation of the weight matrix for the graph with $N=8$ vertices. (a) Ground truth weight matrix. (b) Estimated weight matrix using the polynomial fitting method. (c) Sparsity measure minimization, as a function of parameters $\xi_{1}$ and $\xi_{2}$. (d) The exact (blue lines) and estimated (red crosses) eigenvalues of the normalized Laplacian. (e) Estimated weight matrix using the LASSO minimization. (f) Estimated weight matrix using the graphical LASSO.


Figure 8.9: Estimation of the weight matrix for a graph with $N=50$ randomly positioned vertices. (a) Ground truth weight matrix. (b) Estimated weight matrix using sparsity minimization of the normalized Laplacian and the polynomial fitting method.
the eigenvectors

$$
\mathbf{U}=\frac{1}{\sqrt{8}}\left[\begin{array}{rrrrrrrr}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\
1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\
1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\
1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\
1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\
1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\
1 & -1 & -1 & 1 & -1 & 1 & 1 & -1
\end{array}\right] .
$$

If the eigenvalues are found so as to minimize the number of nonzero elements in the Laplacian, we obtain the graphs for $N=8$ and $N=16$, as shown in Figure 8.10.

A methodology to estimate the underlying graph topology by means of capturing both spatial and time dependencies among multiple time series was introduced in Mei and Moura (2016), whereby time dependencies among data channels are modeled by an auto-regressive (AR) process, while spatial dependencies are estimated by describing the matrix coefficients of the AR process as graph polynomial filters. These authors present three algorithms to estimate the graph adjacency matrix and parameters of the graph polynomial filters.

$\qquad$

Figure 8.10: Graphs for which the Laplacian eigenvectors are the Hadamard transform basis functions for $N=8$ (left) and $N=16$ (right). Different colors and widths of the edges correspond to specific indicated weights.

Another important aspect is that the physical process at hand may dictate that graph topologies need to "jump" between a finite number of discrete states, as manifested by sudden changes in their behavior, the problem considered in Baingana and Giannakis (2016). This type of analysis was inspired by the modeling of contagions, such as the spread of popular news stories or infectious diseases, which propagate in cascades over dynamic graphs/networks. For example, an e-mail network may switch topologies from predominantly work-based connections during the week to friend-based connections over the weekend. In such settings, approaches which assume that network dynamics arise as a result of slow topology variations may yield unpredictable results. To this end, Baingana and Giannakis (2016) employ prior knowledge to introduce novel structural equation models with switched dynamics to effectively capture such causal relationships.

## 20

## From Newton Minimization to Graphical LASSO, via LASSO

Most current approaches to the learning of graph topology from the available data are based on the regression method of the least absolute shrinkage and selection operator (LASSO), with its extension to graphs called the graphical LASSO (GLASSO). This class of methods has already been used in the previous section to learn graph topologies. Because of their importance, they will now be derived and explained in detail, starting from simple one-dimensional Newton minimization.

### 20.1 Newton Method

We shall first briefly review the Newton iterative algorithm for finding the minimum of a convex function. Consider a function, $f(x)$, and assume that it is differentiable. Denote the position of the minimum of $f(x)$ by $x^{*}$. The first derivative of $f(x)$ at the minimum point position

$$
x^{*}=x+\Delta x
$$

can be expanded into a Taylor series around an arbitrary position $x$, using the linear model (which is exact if $f^{\prime \prime \prime}(x)=0$ for all $x$ ), as

$$
\begin{equation*}
f^{\prime}\left(x^{*}\right)=f^{\prime}(x)+f^{\prime \prime}(x) \Delta x . \tag{20.1}
\end{equation*}
$$

Since $f^{\prime}\left(x^{*}\right)=0$, by definition, with $\Delta x=x^{*}-x$, the relation in (20.1) can be rewritten as

$$
x^{*}-x=-\frac{f^{\prime}(x)}{f^{\prime \prime}(x)} .
$$

This formula is used to define an iterative procedure (called Newton's iterative method) for finding the position of the minimum of function $f(x)$, denoted by $x^{*}$, starting from an $x=x_{0}$, as

$$
\begin{equation*}
x_{k+1}=x_{k}-\alpha f^{\prime}\left(x_{k}\right) . \tag{20.2}
\end{equation*}
$$

The parameter $\alpha$ is commonly used instead of $1 / f^{\prime \prime}(x)$ to control the iteration step, and its value should be

$$
0<\alpha \leq \max \left(\left|1 / f^{\prime \prime}(x)\right|\right),
$$

for the considered interval of $x$. This is the form of the well-known steepest descent method for convex function minimization.

Notice that the value $x^{*}=x-\alpha f^{\prime}(x)$ would also be obtained as a result of the minimization of a cost function defined by the quadratic form

$$
\begin{aligned}
x^{*} & =\arg \min _{z} G(z) \\
& =\arg \min _{z}\left(f(x)+f^{\prime}(x)(z-x)+\frac{1}{2 \alpha}(z-x)^{2}\right) .
\end{aligned}
$$

Namely, from the zero-value of the derivative of this cost function

$$
\frac{d}{d z}\left(f(x)+f^{\prime}(x)(z-x)+\frac{1}{2 \alpha}(z-x)^{2}\right)=0
$$

we would arrive at

$$
z=x-\alpha f^{\prime}(x)=x^{*} .
$$

Next, assume that we wish to minimize the cost function

$$
J(x)=\frac{1}{2 \alpha}(x-y)^{2}+\rho|x|,
$$

where $\rho$ is a parameter. This cost function corresponds to the minimization of the squared difference between $x$ and $y$, that is $(x-y)^{2}$, with
an additional sparsity constraint on $x$, given by $|x|$. From

$$
\frac{d J(x)}{d x}=\frac{1}{\alpha}(x-y)+\rho \operatorname{sign}(x)=0
$$

we immediately obtain

$$
x+\rho \alpha \operatorname{sign}(x)=y .
$$

Soft-thresholding, denoted as $\operatorname{soft}(y, \alpha \rho)$, may be used as a solution to this equation, to yield

$$
x=\operatorname{soft}(y, \alpha \rho)= \begin{cases}y+\alpha \rho, & \text { for } y<-\alpha \rho  \tag{20.3}\\ 0, & \text { for }|y| \leq \alpha \rho \\ y-\alpha \rho, & \text { for } y>\alpha \rho .\end{cases}
$$

This form could be considered as the LASSO method for one-dimensional variables. Now, we can proceed with deriving the LASSO method for N -dimensional variables.

### 20.2 Standard LASSO

For the LASSO minimization with $N$-dimensional variables, we will consider the cost function

$$
\begin{aligned}
J(\mathbf{X}) & =\|\mathbf{y}-\mathbf{A} \mathbf{X}\|_{2}^{2}+\rho\|\mathbf{X}\|_{1} \\
& =\|\mathbf{y}\|_{2}^{2}-2 \mathbf{X}^{T} \mathbf{A}^{T} \mathbf{y}+\mathbf{X}^{T} \mathbf{A}^{T} \mathbf{A} \mathbf{X}+\rho\|\mathbf{X}\|_{1}
\end{aligned}
$$

where $\mathbf{y}$ is an $M \times 1$ column vector, $\mathbf{X}$ is an $N \times 1$ column vector, and A is an $M \times N$ matrix (Stanković, 2015).

The minimization of this cost function with respect to the $N$-dimensional variable, $\mathbf{X}$, will produce a value which minimizes $\|\mathbf{y}-\mathbf{A X}\|_{2}^{2}$, meaning that $\mathbf{A X}$ is as close to $\mathbf{y}$ as possible, while at the same time promoting the sparsity of $\mathbf{X}$, through the term $\|\mathbf{X}\|_{1}$ in the minimization. The balance between these two requirements is governed by the parameter $\rho$.

Consider first the differentiable part of the cost function $J(\mathbf{X})$ denoted by

$$
\begin{equation*}
J_{D}(\mathbf{X})=\|\mathbf{y}-\mathbf{A X}\|_{2}^{2}=(\mathbf{y}-\mathbf{A X})^{T}(\mathbf{y}-\mathbf{A X}) . \tag{20.4}
\end{equation*}
$$

Its derivatives are

$$
\frac{\partial J_{D}(\mathbf{X})}{\partial \mathbf{X}^{T}}=-2 \mathbf{A}^{T} \mathbf{y}+2 \mathbf{X}^{T} \mathbf{A}^{T} \mathbf{A}
$$

and

$$
\frac{\partial^{2} J_{D}(\mathbf{X})}{\left(\partial \mathbf{X}^{T}\right)^{2}}=2 \mathbf{A}^{T} \mathbf{A}
$$

A linear model for the first derivative of $J_{D}(\mathbf{X})$ around its minimum, $\mathbf{X}^{*}$, which corresponds to (20.1), is given by

$$
\frac{\partial J_{D}\left(\mathbf{X}^{*}\right)}{\partial \mathbf{X}^{T}}=\mathbf{0}=\frac{\partial J_{D}(\mathbf{X})}{\partial \mathbf{X}^{T}}+(\Delta \mathbf{X}) \frac{\partial^{2} J_{D}(\mathbf{X})}{\left(\partial \mathbf{X}^{T}\right)^{2}} .
$$

By replacing the inverse of the second order derivative, $1 /\left(\frac{\partial^{2} J_{D}(\mathbf{X})}{\left(\partial \mathbf{X}^{T}\right)^{2}}\right)$, by a constant diagonal matrix $\alpha \mathbf{I}$, as in (20.2), we have

$$
\Delta \mathbf{X}=\mathbf{X}^{*}-\mathbf{X}=-\frac{\frac{\partial J_{D}(\mathbf{X})}{\partial \mathbf{X}^{T}}}{\frac{\partial^{2} J_{D}(\mathbf{X})}{\left(\partial \mathbf{X}^{T}\right)^{2}}}=-\alpha \frac{\partial J_{D}(\mathbf{X})}{\partial \mathbf{X}^{T}},
$$

or

$$
\begin{equation*}
\mathbf{X}^{*}=\mathbf{X}-\alpha \frac{\partial J_{D}(\mathbf{X})}{\partial \mathbf{X}^{T}}, \tag{20.5}
\end{equation*}
$$

with

$$
0<\alpha<\frac{1}{\max \left\|2 \mathbf{A}^{T} \mathbf{A}\right\|}=\frac{1}{2 \lambda_{\max }},
$$

where $\lambda_{\text {max }}$ is the maximum eigenvalue of matrix $\mathbf{A}^{T} \mathbf{A}$.
In order to find $\mathbf{Z}=\mathbf{X}^{*}$ that minimizes the complete cost function $J(\mathbf{X})$ we can minimize the squared value of the difference

$$
\mathbf{Z}-\left(\mathbf{X}-\alpha \mathbf{I} \frac{\partial J_{D}(\mathbf{X})}{\partial \mathbf{X}^{T}}\right)
$$

and the norm-one of $\mathbf{Z}$, by forming the cost function, $G(\mathbf{Z})$, as

$$
G(\mathbf{Z})=\frac{1}{2 \alpha}\left\|\mathbf{Z}-\left(\mathbf{X}-\alpha \mathbf{I} \frac{\partial J_{D}(\mathbf{X})}{\partial \mathbf{X}^{T}}\right)\right\|_{2}^{2}+\rho\|\mathbf{Z}\|_{1} .
$$

The minimization of $G(\mathbf{Z})$ will produce $\mathbf{Z}$ which is as close as possible to the desired solution in (20.5), while minimizing its norm-one (maximum sparsity) at the same time, with $\rho$ as the balance parameter.

If we use the notation

$$
\mathbf{Y}=\left(\mathbf{X}-\alpha \mathbf{I} \frac{\partial J_{D}(\mathbf{X})}{\partial \mathbf{X}^{T}}\right)
$$

the solution of

$$
\mathbf{X}^{*}=\arg \min _{\mathbf{Z}} G(\mathbf{Z})=\arg \min _{\mathbf{Z}} \frac{1}{2 \alpha}\|\mathbf{Z}-\mathbf{Y}\|_{2}^{2}+\rho\|\mathbf{Z}\|_{1}
$$

is obtained from

$$
\frac{1}{\alpha}\left(\mathbf{X}^{*}-\mathbf{Y}\right)+\rho \operatorname{sign}\left(\mathbf{X}^{*}\right)=\mathbf{0} .
$$

Using the soft-thresholding function as in (20.3), we can further write

$$
\mathbf{X}^{*}=\operatorname{soft}(\mathbf{Y}, \alpha \rho) .
$$

Next, we can replace the value of $\mathbf{Y}$ by

$$
\begin{aligned}
\mathbf{Y} & =\left(\mathbf{X}-\alpha \mathbf{I} \frac{\partial J_{D}(\mathbf{X})}{\partial \mathbf{X}^{T}}\right)=\mathbf{X}-\alpha \mathbf{I}\left(-2 \mathbf{A}^{T} \mathbf{y}+2 \mathbf{X}^{T} \mathbf{A}^{T} \mathbf{A}\right) \\
& =2 \alpha \mathbf{A}^{T} \mathbf{y}+\left(\mathbf{I}-2 \alpha \mathbf{A}^{T} \mathbf{A}\right) \mathbf{X}
\end{aligned}
$$

The iterative formula for the solution of the so defined minimization problem is obtained by replacing $\mathbf{X}^{*}=\mathbf{X}_{k+1}$ and $\mathbf{X}=\mathbf{X}_{k}$, to yield

$$
\begin{equation*}
\mathbf{X}_{k+1}=\operatorname{soft}\left(2 \alpha \mathbf{A}^{T}\left(\mathbf{y}-\mathbf{A} \mathbf{X}_{k}\right)+\mathbf{X}_{k}, \alpha \rho\right) . \tag{20.6}
\end{equation*}
$$

This formula can be rewritten for each element of $\mathbf{X}_{k}$ and implemented as in Algorithm 2. This is the essence of the LASSO (Least Absolute Shrinkage and Selection Operator) iterative algorithm. Notice that $\mathbf{X}_{0}=\mathbf{A}^{T} \mathbf{y}$ is commonly used as the initial estimate.
Example 87: Consider a sparse signal, $X(k)$, with $N=60$ elements. In general, to calculate these signal elements we need at least $M=60$ measurements (linear combinations of signal elements). A signal can be reconstructed from a reduced set of $M<N$ measurements if it is sparse, with $K \ll N$ nonzero elements at unknown positions.

Assume that the original sparse signal of the total length $N=60$ has the values in the transform domain given by $X(k)=0$ for all $k$ except for $X(5)=1, X(12)=0.5, X(31)=0.9$, and $X(45)=-0.75$, and that
it is measured with a matrix A with only $M=40<N$ measurements stored in vector $\mathbf{y}$.

The measurement matrix $\mathbf{A}$ is formed as a Gaussian random matrix of the size $40 \times 60$, with elements $\mathcal{N}\left(0, \sigma^{2}\right)$, where $\sigma^{2}=1 / 40$ is used. All 60 signal values were reconstructed using these 40 measurements, $\mathbf{y}$, and the matrix $\mathbf{A}$, in 1000 iterations. In the initial iteration, $\mathbf{X}_{0}=\mathbf{A}^{T} \mathbf{y}$, was used; then for each next iteration $k$, the new values of $\mathbf{X}$ were calculated using (20.6) and Algorithm 2, given the data $\mathbf{y}$ and matrix $\mathbf{A}$. The results for $\rho=0.1$ and $\rho=0.001$ are shown in Figure 8.1. For a very small $\rho=0.001$, the result is not sparse, since the constraint is too weak.

### 20.3 Graphical LASSO

In graph model learning, the corresponding cost function of the form

$$
J(\mathbf{Q})=-\ln (\operatorname{det} \mathbf{Q})+\operatorname{Trace}\left(\mathbf{Q} \mathbf{R}_{x}\right)+\rho\|\mathbf{Q}\|_{1}
$$

may be used. Here, $\mathbf{Q}$ is the $N \times N$ generalized Laplacian matrix, while $\mathbf{R}_{x}$ is the available $N \times N$ data correlation matrix. Physical meaning of these terms is explained in Section 19.4.

The derivative of the cost function with respect to the elements of Q can be written as

$$
\begin{equation*}
-\mathbf{Q}^{-1}+\mathbf{R}_{x}+\rho \operatorname{sign}(\mathbf{Q})=\mathbf{0} \tag{20.7}
\end{equation*}
$$

at $\partial J(\mathbf{Q}) / \partial \mathbf{Q}=\mathbf{0}$.
Upon introducing the notation

$$
\mathbf{V}=\mathbf{Q}^{-1}
$$

or

$$
\mathrm{VQ}=\mathbf{I}
$$

we can write

$$
\mathbf{V}=\left[\begin{array}{ll}
\mathbf{V}_{11} & \mathbf{v}_{12}  \tag{20.8}\\
\mathbf{v}_{12}^{T} & v_{22}
\end{array}\right] \quad \mathbf{Q}=\left[\begin{array}{ll}
\mathbf{Q}_{11} & \mathbf{q}_{12} \\
\mathbf{q}_{12}^{T} & q_{22}
\end{array}\right]
$$



Figure 8.1: A sparse signal with $N=60$ samples and $K=4$ nonzero elements, which is reconstructed using a reduced set of $M=40$ observations and the LASSO iterative algorithm. The top panel shows the result for the matched filter (initial estimate), $\mathbf{X}_{0}=\mathbf{A}^{T} \mathbf{y}$, and the middle and bottom panel for the LASSO iterative algorithm with $\rho=0.1$ and $\rho=0.001$. Observe a poor reconstruction for a very small $\rho=0.001$, which according to (20.6) could not yield a sparse signal.
and

$$
\left[\begin{array}{ll}
\mathbf{V}_{11} & \mathbf{v}_{12}  \tag{20.9}\\
\mathbf{v}_{12}^{T} & v_{22}
\end{array}\right]\left[\begin{array}{ll}
\mathbf{Q}_{11} & \mathbf{q}_{12} \\
\mathbf{q}_{12}^{T} & q_{22}
\end{array}\right]=\left[\begin{array}{cc}
\mathbf{I} & \mathbf{0} \\
\mathbf{0}^{T} & 1
\end{array}\right],
$$

where $\mathbf{Q}_{11}$ and $\mathbf{V}_{11}$ are $(N-1) \times(N-1)$ matrices, $\mathbf{v}_{12}$ and $\mathbf{q}_{12}$ are $(N-1) \times 1$ column vectors, and $v_{22}$ and $q_{22}$ are scalars.

After multiplying the first row of blocks in $\mathbf{V}$ with the last column of blocks in $\mathbf{Q}$, we have

$$
\mathbf{V}_{11} \mathbf{q}_{12}+\mathbf{v}_{12} q_{22}=\mathbf{0}
$$

which gives

$$
\begin{equation*}
\mathbf{v}_{12}=-\mathbf{V}_{11} \mathbf{q}_{12} / q_{22}=\mathbf{V}_{11} \boldsymbol{\beta}, \tag{20.10}
\end{equation*}
$$

where

$$
\begin{equation*}
\boldsymbol{\beta}=-\mathbf{q}_{12} / q_{22} \tag{20.11}
\end{equation*}
$$

is normalized with $q_{22}>0$.
Now, from the derivative Equation (20.7) we may write

$$
-\left[\begin{array}{ll}
\mathbf{V}_{11} & \mathbf{v}_{12} \\
\mathbf{v}_{12}^{T} & v_{22}
\end{array}\right]+\left[\begin{array}{cc}
\mathbf{R}_{11} & \mathbf{r}_{12} \\
\mathbf{r}_{12}^{T} & r_{22}
\end{array}\right]+\rho \operatorname{sign}\left(\left[\begin{array}{ll}
\mathbf{Q}_{11} & \mathbf{q}_{12} \\
\mathbf{q}_{12}^{T} & q_{22}
\end{array}\right]\right)=\mathbf{0}
$$

For the upper right block we have

$$
-\mathbf{v}_{12}+\mathbf{r}_{12}+\rho \operatorname{sign}\left(\mathbf{q}_{12}\right)=\mathbf{0},
$$

while after replacing $\mathbf{v}_{12}=\mathbf{V}_{11} \boldsymbol{\beta}$ and $\mathbf{q}_{12}=-\boldsymbol{\beta} / q_{22}$ from (20.10) and (20.11) we arrive at

$$
\begin{equation*}
-\mathbf{V}_{11} \boldsymbol{\beta}+\mathbf{r}_{12}-\rho \operatorname{sign}(\boldsymbol{\beta})=\mathbf{0} \tag{20.12}
\end{equation*}
$$

The solution to this equation for $\boldsymbol{\beta}$ has been already defined within the LASSO framework, and is given by

$$
\begin{equation*}
\beta_{i} V_{11}(i)=\operatorname{soft}\left(r_{12}(i)-\sum_{k \neq i} V_{11}(k, i) \beta_{k}, \rho\right) . \tag{20.13}
\end{equation*}
$$

In order to apply the LASSO as in (20.6), we can interpret the minimization of the difference

$$
\mathbf{A}^{T}(\mathbf{y}-\mathbf{A X})=\mathbf{A}^{T} \mathbf{y}-\mathbf{A}^{T} \mathbf{A} \mathbf{X}
$$

in (20.6) as the task of finding the least-squares estimate of $\mathbf{A}^{T} \mathbf{y}$ by $\mathbf{A}^{T} \mathbf{A X}$. Now, we can adjust (20.12) to assume a similar form

$$
\begin{equation*}
-\mathbf{V}_{11}^{1 / 2} \mathbf{V}_{11}^{1 / 2} \boldsymbol{\beta}+\mathbf{V}_{11}^{1 / 2} \mathbf{V}_{11}^{-1 / 2} \mathbf{r}_{12}-\rho \operatorname{sign}(\boldsymbol{\beta})=\mathbf{0} . \tag{20.14}
\end{equation*}
$$

In this case, the matrix $\mathbf{V}_{11}^{1 / 2}$ plays the role of $\mathbf{A}$ in (20.6) and $\mathbf{V}_{11}^{-1 / 2} \mathbf{r}_{12}$ plays the role of $\mathbf{y}$. Therefore, the standard LASSO should be calculated using

$$
\begin{equation*}
\boldsymbol{\beta}=\operatorname{lasso}\left(\mathbf{V}_{11}^{1 / 2}, \mathbf{V}_{11}^{-1 / 2} \mathbf{r}_{12}, \rho\right) \tag{20.15}
\end{equation*}
$$

as in Algorithm 4.
Now, the graphical LASSO (GLASSO) iterative algorithm can be summarized as follows.

- In the initial step, use

$$
\mathbf{V}=\mathbf{R}_{x}+\rho \mathbf{I} .
$$

- For each coordinate, $j=1,2, \ldots, N$, the matrix equation of the form (20.9) is written. For each $j$, the reduced matrix $\mathbf{V}_{11}$ is formed by omitting the $j$ th row and the $j$ th column. Then, the matrix $\mathbf{R}_{x}$ is rearranged accordingly.
- Equation (20.13) is solved using (20.15).
- The matrix $\mathbf{V}$ is updated for each $j$ by inserting the $j$ th column

$$
\mathbf{v}_{12}=\mathbf{V}_{11} \boldsymbol{\beta},
$$

and inserting at the $j$ th row $\mathbf{v}_{12}^{T}$ with the element $v_{22}$ at the $j$ th position.

- After all $j$ indices are used in the calculation, the final estimate of the generalized Laplacian is obtained as $\mathbf{Q}=\mathbf{V}^{-1}$.

This calculation procedure is also presented in Algorithm 4.
Remark 96: Notice that the value of matrix $\mathbf{Q}=\mathbf{V}^{-1}$ is updated for each $j$ and in the last iteration, using the column vector

$$
\mathbf{q}_{12}=-\boldsymbol{\beta} q_{22},
$$

where $q_{22}$ can be calculated from $\mathbf{v}_{12}^{T} \mathbf{q}_{12}+v_{22} q_{22}=1$ or $-\mathbf{v}_{12}^{T} \boldsymbol{\beta} q_{22}+$ $v_{22} q_{22}=1$, finally producing the values

$$
q_{22}=\frac{1}{v_{22}-\mathbf{v}_{12}^{T} \boldsymbol{\beta}},
$$

and

$$
\mathbf{q}_{12}=\frac{\boldsymbol{\beta}}{\mathbf{v}_{12}^{T} \boldsymbol{\beta}-v_{22}}
$$

which are used to update the $j$ th column and row of the matrix $\mathbf{Q}$ in the same as the update of matrix $\mathbf{V}$.

This algorithm can be used for iterative matrix inversion with $\rho=0$.
Example 88: Consider a graph with $N=50$ vertices and with a small number of edges, so that the weight matrix, $\mathbf{W}$, is sparse. The ground truth weight matrix, W, is shown in Figure 8.2(a). This matrix was then estimated from a large number, $P=1000$, of observations of a signal on this graph. Both the precision matrix, $\mathbf{R}^{-1}$, and the graphical LASSO, given in Figures 8.2(b) and (c), produce good estimation of the weight matrix, W. Next, the number of observations was significantly reduced to $P=40<N=50$, a case when the correlation matrix, $\mathbf{R}$, is singular and of rank lower or equal to $P=40$. In this case, the sparsity of the weight matrix is crucial for the solution. Here, only the graphical LASSO, which includes the sparsity constraint, was able to produce good result, as shown in Figure 8.2(e), while the precision matrix could be calculated only through a pseudo-inverse, and cannot be used as the weight matrix estimate, as can be seen from Figure 8.2(d).


Figure 8.2: Estimation of the weight matrix, $\mathbf{W}$, for a graph with $N=50$ randomly positioned vertices. (a) Ground truth weight matrix, W. (b) Precision matrix, for a large number of observations, $P=1000 \gg N=50$. (c) Estimated weight matrix using the graphical LASSO, for a large number of observations, $P=1000 \gg N=50$. (d) Precision matrix, for a small number of observations, $P=40<N=50$ (the correlation matrix, $\mathbf{R}$, is singular and with a rank lower or equal to $P$, so that pseudo-inversion is used). (e) Estimated weight matrix using the graphical LASSO, for a small number of observations, $P=40<N=50$.

## 21

## Physically Well Defined Graphs

The simplest scenario for graph connectivity consideration is when the graph associated with a problem at hand is physically well defined. Examples of such graphs are manifold, including electric circuits, power networks, linear heat transfer, social and computer networks, and springmass systems, all of which will be addressed in this section.

### 21.1 Resistive Electrical Circuits

Graph theory based methods for the analysis and transformations of electrical circuits have long been part of classical courses and textbooks. It is also interesting that some general information theory problems can be interpreted and solved within the graph approach to basic electric circuits. In such cases, the underlying graph topology is well defined and is a part of the problem statement.

The graph Laplacian can also be considered within the basic electric circuit theory. In this case, since the graph Laplacian can be derived based on the Kirchhoff's laws, it is also known as the Kirchhoff matrix.

## Graph Representation of Electric Circuits

Consider a resistive electric circuit, and the electric potential in the circuit vertices (nodes), denoted by $x(n)$. The vertices in an electrical
circuit are connected with edges, where the weight of an edge connecting the vertices $n$ and $m$ is defined by the edge conductance, $W_{n m}$. The conductances are the reciprocal values to edge resistances

$$
W_{n m}=\frac{1}{R_{n m}} .
$$

The current in the edge from vertex $n$ to vertex $m$ is then equal to

$$
i_{n m}=\frac{x(n)-x(m)}{R_{n m}}=W_{n m}(x(n)-x(m)) .
$$

In addition to the edge currents, an external current generator may be attached to every vertex, and can be considered as a source of signal change in the vertices; the external current at a vertex $n$ is denoted by $i_{n}$.

Since the sum of all currents going in/from a vertex $n, n=0,1, \ldots$, $N-1$, must be 0 , that is

$$
-i_{n}+\sum_{m} i_{n m}=0
$$

the current of the external generator at a vertex $n$ must be equal to the sum of all edge currents going in/from this vertex, to give

$$
\begin{array}{r}
i_{n}=\sum_{m} W_{n m}(x(n)-x(m))=d_{n} x(n)-\sum_{m} W_{n m} x(m), \\
n=0,1, \ldots, N-1,
\end{array}
$$

where

$$
d_{n}=\sum_{m} W_{n m}=\sum_{m=0}^{N-1} W_{n m}
$$

is the degree of vertex $n$. The summation over $m$ can be extended to all vertices, $m=0,1, \ldots, N-1$, since $W_{n m}=0$ if there is no edge between vertices $n$ and $m$.

The above equations can be written in a matrix form as

$$
\mathbf{i}=\mathbf{D} \mathbf{x}-\mathbf{W} \mathbf{x}
$$

or

$$
\begin{equation*}
\mathbf{L x}=\mathbf{i} \tag{21.1}
\end{equation*}
$$

where $\mathbf{L}=\mathbf{D}-\mathbf{W}$ is the Laplacian of a graph representing an electric circuit and $\mathbf{i}$ is the vector of currents at every vertex.

If the Laplacian matrix is decomposed as $\mathbf{L}=\mathbf{U} \boldsymbol{\Lambda} \mathbf{U}^{T}$, from (21.1) we have $\boldsymbol{\Lambda} \mathbf{U}^{T} \mathbf{x}=\mathbf{U}^{T} \mathbf{i}$, and

$$
\begin{equation*}
\boldsymbol{\Lambda} \mathbf{X}=\mathbf{I} \tag{21.2}
\end{equation*}
$$

where $\mathbf{X}=\mathbf{U}^{T} \mathbf{x}$ and $\mathbf{I}=\mathbf{U}^{T} \mathbf{i}$ are the GDFT of graph signals $\mathbf{x}$ and $\mathbf{i}$ (see Part II, Section 3.6).

From (21.2), the components of the spectral transform vector, $\mathbf{X}$, satisfy

$$
\lambda_{k} X(k)=I(k)
$$

for each $k$.
A signal measured on an electrical circuit graph can be related to the above theory in several ways. For example, potentials on all vertices could be measured under some measurement noise, which calls for data filtering on a graph. Another possible case is when external conditions are imposed, for example external sources are applied to some vertices. We are then interested in the values of electric potential at all vertices, a problem which corresponds to graph signal reconstruction.

For nontrivial solutions, there should be an external source on at least two vertices. If we assume that a vertex with an external source is chosen as a reference vertex, then the signal or external source values at these vertices (with external sources) are sufficient to find signal values at all other vertices.
Example 89: Consider the graph and signal sensed on the graph presented in Figure 8.1. The signal values are

$$
\mathbf{x}=[6.71,6.88,7.13,5.25,6.67,8.18,2.62,0]^{T}
$$

and the graph Laplacian (as a matrix operator) applied to the signal yields

$$
\mathbf{L x}=[0,0,1,0,0,2,0,-3]^{T} .
$$

Observe that in this case the vertices indexed by $0,1,3,4,6$ are not active, and their values can be obtained as linear combinations of the


Figure 8.1: Electric potential, $x(n)$, as a signal on an electric circuit graph.
signals at neighboring active vertices, that is

$$
\begin{align*}
1.21 x(0)-0.23 x(1)-0.74 x(2)-0.24 x(3) & =0 \\
-0.23 x(0)+0.81 x(1)-0.35 x(2)-0.23 x(4) & =0 \\
-0.24 x(0)-0.26 x(2)+0.82 x(3)-0.32 x(6) & =0  \tag{21.3}\\
-0.23 x(1)-0.24 x(2)+1.12 x(4)-0.51 x(5)-0.14 x(7) & =0 \\
-0.32 x(3)+0.64 x(6)-0.32 x(7) & =0 .
\end{align*}
$$

After solving this system with known signal values $x(2)=7.13$, $x(5)=8.18$, and $x(7)=0$ at the active vertices, we obtain the remaining signal values

$$
\mathbf{x}_{p}=[x(0), x(1), x(3), x(4), x(6)]^{T}=[6.71,6.88,5.25,6.67,2.62]^{T}
$$

## Graph Transformations

A graph with one or more inactive vertices (where the elements of $\mathbf{L x}$ are equal to zero) can be simplified by removing these vertices using the well-known transformations of edges connected in series, parallel, or star-to-mesh transforms. This process corresponds to the downsampling of the graph signal (see also Part II). The reduction of an electrical network via a Schur complement of the associated conductance matrix is known as the Kron reduction, whereby the vertices are separated into two groups: active vertices and inner vertices. The inner vertices can


Figure 8.2: Electric potential, $x(n)$, as a signal on an electric circuit graph observed at the three vertices with nonzero external sources. For this graph, all other values of $x(n)$ in Figure 8.1 can be calculated based on the signal values at vertices $n=2$, $n=5$, and $n=7$.
be eliminated from the graph without changing the electric network conditions; this is achieved via equivalent transformations, such as the "star-mesh" transformations (Dorfler and Bullo, 2012).

Similar procedure can be used to add inactive vertices, either by inserting a vertex within an edge or by transforming meshes to stars, in what corresponds to the interpolation of the graph signal.
Example 90: For the graph and the graph signal from Example 89, the active vertices are $n=2,5,7$, as shown in Figure 8.2, while the signal values at all vertices are given in Figure 8.1. Notice that the existing signal values will not change, for the given external sources, if the graph is "downsampled", as shown in Figure 8.3, or if the graph signal is "interpolated" by adding new vertices, as shown in Figure 8.4. This is closely related to link prediction, that will be addressed later.

## Graph Data Denoising for Sparse External Sources

The set of external sources are considered sparse if their number is much smaller than the number of vertices, $N$. For this scenario, the norm-zero of the external sources vector, $\mathbf{L x}$, is such that $\|\mathbf{L x}\|_{0} \ll N$. If the noisy observations, $\mathbf{y}$, of data on graph, $\mathbf{x}$, are available and we know that the number of external sources is small, then the cost function for denoising


Figure 8.3: Graph signal, $x(n)$, from Figure 8.1, observed on a graph with a reduced number of vertices ("downsampling"), whereby the vertices $n=6$ and $n=1$ are removed (crosses in green dots). Observe that the signal values at the active vertices, $n=2, n=5$, and $n=7$, are not changed. The edge weights in gray shade are the equivalent values obtained using the standard resistor, $R_{m n}=1 / W_{m n}$, transformations.


Figure 8.4: Graph signal, $x(n)$, from Figure 8.1 observed on a graph with an extended number of vertices ("interpolation"). Observe that the signal values at all vertices, $n=0,1,2,3,4,5,6,7$, from Figure 8.2 are not changed. In the locations where the new vertices $n=8$ and $n=9$ are added, the graph signal is interpolated using $x(2), x(5)$, and $x(7)$, as in (21.3), and the corresponding edge weights are shown in gray.
can be written in the form

$$
\begin{equation*}
J=\|\mathbf{y}-\mathbf{x}\|_{2}^{2}+\rho\|\mathbf{L} \mathbf{x}\|_{0} \tag{21.4}
\end{equation*}
$$

This minimization problem can be solved either using the corresponding norm-one form

$$
\begin{equation*}
J=\|\mathbf{y}-\mathbf{x}\|_{2}^{2}+\rho\|\mathbf{L} \mathbf{x}\|_{1} \tag{21.5}
\end{equation*}
$$

or using a kind of matching pursuit, as presented in the next example within a classical data denoising scenario.
Example 91: Consider the classical time domain and a piece-wise linear signal, of which noisy observations are available, as shown in Figure 8.5(a). In standard analysis, the graph representation of the domain of this signal is an undirected and unweighted path graph, where the elements of $\mathbf{L x}$ play the role of external sources, as shown in Figure $8.5(\mathrm{~b})$. We shall assume that $n=0$ is the reference vertex with $x(0)=0$.

The data denoising problem is then solved in the following way. The initial estimate of the external sources is calculated as $\mathbf{L y}$. Since we assumed that the external sources are sparse, we will consider the positions, $k_{1}, k_{2}, k_{3}, k_{4}$, and $k_{5}$, of $K=5$ largest absolute values of the initial estimate.

The largest $K$ nonzero values of the external source vector, $\mathbf{L y}$, are denoted by $\mathbf{J}_{K}$, with the elements $i\left(k_{1}\right), i\left(k_{2}\right), \ldots, i\left(k_{K}\right)$. The value of $\mathbf{J}_{K}$ is found in such a way that it minimizes the difference between the estimated data, $\mathbf{L}_{K}^{(-1)} \mathbf{J}_{K}$, and the observations, $\mathbf{y}$, that is

$$
\min _{\mathbf{J}_{K}}\left\|\mathbf{y}-\mathbf{L}_{K}^{(-1)} \mathbf{J}_{K}\right\|_{2}^{2}
$$

where $\mathbf{L}_{K}^{(-1)}$ is obtained from the inverse transform of the graph Laplacian (after the reference row and column, at $n=0$ are omitted) by keeping only $K$ columns which correspond to the nonzero positions in the external source vector, $\mathbf{J}_{K}$. The solution therefore becomes

$$
\mathbf{J}_{K}=\operatorname{pinv}\left(\mathbf{L}_{K}^{(-1)}\right) \mathbf{y} .
$$

After the nonzero external sources are found, the full external source vector, $[i(1), i(2), \ldots, i(N-1)]^{T}$, is formed using the calculated nonzero values in $\mathbf{J}_{K}$ and inserting zero values at the remaining positions, as shown in Figure 8.5(c).


Figure 8.5: Original piece-wise linear noisy signal (top) and the reconstructed signal (bottom), with the Laplacian of the noisy observations and its re-estimated sparse version (middle panels).

Finally, the reconstructed signal is obtained from

$$
\left[\begin{array}{cccc}
L_{11} & L_{12} & \ldots & L_{1, N-1} \\
L_{21} & L_{22} & \ldots & L_{2, N-1} \\
\vdots & \vdots & \ddots & \vdots \\
L_{N-1,2} & L_{N-1,2} & \ldots & L_{N-1, N-1}
\end{array}\right]\left[\begin{array}{c}
x(1) \\
x(2) \\
\vdots \\
x(N-1)
\end{array}\right]=\left[\begin{array}{c}
i(1) \\
i(2) \\
\vdots \\
i(N-1)
\end{array}\right]
$$

as, $\mathbf{x}=\mathbf{L}^{-1} \mathbf{J}$, with the result shown in Figure 8.5(d). Note that the matrix $\mathbf{L}$ is obtained from the graph Laplacian matrix by removing the reference vertex row and column.

Remark 97: The crucial advantage over the standard total variation (TV) minimization approach in the compressive sensing based denoising is that the cost function used in Example 91 does not penalize for the linear changes of the signal, while the TV approach promotes piece-wise constant signals.

### 21.2 Heat Transfer

The same model as in resistive electrical circuits can be used for a heat transfer network. In this case, the signal values are the measured temperatures, $x(n)=T(n)$, while the heat flux is defined as

$$
q_{n m}=(T(n)-T(m)) C_{n m}=(x(n)-x(m)) W_{n m},
$$

where $C_{n m}$ are the heat transfer constants, which represent edge weights in the underlying graph, $C_{n m}=W_{n m}$.

Then, the input heat flux in a vertex $n$ can be written as

$$
q_{n}=\sum_{m} W_{n m}(x(n)-x(m))=d_{n} x(n)-\sum_{m=0}^{N-1} W_{n m} x(m)
$$

with

$$
\mathbf{q}=\mathbf{L x}
$$

Active vertices are those with an external heat flux, while the passive vertices are those where all heat flux coming to a vertex is forwarded to other vertices, through the edges. An example of a heat transfer graph is given in Figure 8.6.

### 21.3 Spring-Mass Systems

A spring mass system can also be modeled as a graph. Consider a system of $N=4$ masses which correspond to a path graph, as in Figure 8.7. Assume that all displacements and forces are aligned with springs. According to Hook's law, the displacements, $x(n)$, and the


Figure 8.6: Temperature, $x(n)=T(n)$, as a signal on a heat transfer graph, with $q_{1}, q_{2}, \ldots, q_{7}$, as the external heat flux values at the corresponding vertices.


Figure 8.7: Spring-mass system on a path graph.
forces, $F_{n}$, at the steady state are related as

$$
\begin{aligned}
k_{1}(x(1)-x(2)) & =F_{1} \\
k_{1}(x(2)-x(1))+k_{2}(x(2)-x(3)) & =F_{2} \\
k_{2}(x(3)-x(2))+k_{3}(x(3)-x(4)) & =F_{3} \\
k_{3}(x(4)-x(3)) & =F_{4}
\end{aligned}
$$

or in a matrix form

$$
\begin{aligned}
{\left[\begin{array}{cccc}
k_{1} & -k_{1} & 0 & 0 \\
-k_{1} & k_{1}+k_{2} & -k_{2} & \\
0 & -k_{2} & k_{2}+k_{3} & -k_{3} \\
0 & 0 & -k_{3} & k_{3}
\end{array}\right]\left[\begin{array}{l}
x_{1} \\
x_{2} \\
x_{3} \\
x_{4}
\end{array}\right] } & =\left[\begin{array}{l}
F_{1} \\
F_{2} \\
F_{3} \\
F_{4}
\end{array}\right] \\
\mathbf{L x} & =\mathbf{F} .
\end{aligned}
$$

These equations define a weighted graph and its corresponding graph Laplacian.

Given that the graph Laplacian is a singular matrix, in order to solve this system for unknown displacements (graph signal), we should
introduce a reference vertex with a fixed position (zero displacement). Then, the system $\mathbf{L x}=\mathbf{F}$ can be solved.

### 21.4 Social Networks and Linked Pages

Social networks are also examples of well defined graphs, where the vertices are network members and the edges define their relationships within a social network. If two members in a social network are related, then the corresponding edge weight is 1 , and the weight matrix is equal to the adjacency matrix. An example of a small social network with the corresponding member links is shown in Figure 8.8.

Pages with hyper-links can also be considered as a well defined directed graph; an example of links between $N=8$ pages is given in Figure 8.9. An interesting parameter for this kind of graphs is the PageRank.


Figure 8.8: An example of a small social network represented as an undirected graph.


Figure 8.9: Hyper-linked internet pages in a holiday search scenario, which can be represented as a directed graph. The vertices 0 to 7 reflect the nature of web search, as follows. 0: Venue search results. 1: Places to visit at the venue from 0. 2: Search for road trip. 3: Google search engine. 4: Car hire website. 5: Trustpilot ranking of car rental agent. 6: Personalized roadmap for the trip. 7: Review of sites of interest.

### 21.5 PageRank

The PageRank was defined by Google to rank the web pages. For a directed graph, PageRank of a vertex $n$ is defined as a graph signal satisfying the relation

$$
x(n)=\sum_{m} \frac{1}{d_{m}} W_{m n} x(m),
$$

where $W_{m n}$ are weights of the directed edges connecting vertices $m$ and $n$, and $d_{m}$ is the outgoing degree of a vertex $m$. This means that the PageRank of each vertex is related to the PageRank of the vertices connected to it.

The PageRank is usually calculated using an iterative procedure defined by

$$
\begin{equation*}
x_{k+1}(n)=\sum_{m} \frac{1}{d_{m}} W_{m n} x_{k}(m) \tag{21.6}
\end{equation*}
$$

starting from an arbitrary PageRank, for example $x_{0}(n)=1$. In the original definition by Google, the scaling factors of 0.15 and 0.85 were added, to give

$$
\begin{equation*}
x_{k+1}(n)=0.15+0.85 \sum_{m} \frac{1}{d_{m}} W_{m n} x_{k}(m) \tag{21.7}
\end{equation*}
$$

Example 92: Consider the graph from Figure 8.9 (the same graph as in Part I, Figure 2.1(b)). In this case, the vertices represent pages on the Internet, while the directed edges designate their relations. For example, the page which corresponds to vertex 0 cites (gives a hyper-link to) page marked with 1 , while it is cited (hyper-linked) by pages at vertex 2 and vertex 3 . All other vertices are connected by the edges in the same way. Intuitively, we can expect that the rank in this network is higher for the pages that are highly cited (hyper-linked) with other also highly cited (hyper-linked) pages. To find the rank of the pages in this graph/network, we first need to calculate the PageRank for all pages/vertices. The weight/adjacency matrix of this graph, $\mathbf{W}=\mathbf{A}$, is given by (see also Part I, Equation (2.2))

$$
\mathbf{W}=\begin{gather*}
0  \tag{21.8}\\
1 \\
2 \\
3 \\
4 \\
5 \\
6 \\
7
\end{gather*}\left[\begin{array}{llllllll}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 0
\end{array}\right]
$$

The outgoing vertex degrees are calculated as sums of columns of the matrix $\mathbf{W}^{T}$, that is $d_{m}=\sum_{n=0}^{7} W_{m n}$, with their values

$$
\mathbf{d}=\left[\begin{array}{llllllll}
1 & 1 & 4 & 1 & 3 & 1 & 2 & 2
\end{array}\right] .
$$

Now, the PageRank values for vertices can be obtained through an iterative procedure, as in (21.6), starting with the initial page ranks $\mathbf{x}_{0}=[1,1,1,1,1,1,1,1]$. After a few iterations, the results for PageRank
are as follows

$$
\left[\begin{array}{c}
\mathbf{x}_{0}^{T} \\
\mathbf{x}_{1}^{T} \\
\mathbf{x}_{2}^{T} \\
\vdots \\
\mathbf{x}_{5}^{T} \\
\vdots \\
\mathbf{x}_{11}^{T}
\end{array}\right]=\left[\right] .
$$

The matrix form of the iterations in (21.6) is

$$
\mathbf{x}_{k+1}=\mathbf{W}_{N} \mathbf{x}_{k}
$$

where $\mathbf{W}_{N}$ is obtained from $\mathbf{W}^{T}$ by dividing all elements of the $m$ th column, $m=0,1, \ldots, N-1$, by $d_{m}$. The mean-values of matrix $\mathbf{W}_{N}$ columns are normalized.
Example 93: The normalized adjacency/weighting matrix from Example 92 , is

$$
\mathbf{W}_{N}=\left[\begin{array}{cccccccc}
0 & 0 & \frac{1}{4} & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & \frac{1}{3} & 0 & 0 & \frac{1}{2} \\
0 & 0 & \frac{1}{4} & 0 & 0 & 0 & \frac{1}{2} & 0 \\
0 & 0 & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \\
0 & 0 & \frac{1}{4} & 0 & 0 & 1 & \frac{1}{2} & 0
\end{array}\right]
$$

The final, steady state, PageRank $\mathbf{x}$ can then be obtained from

$$
\mathbf{x}=\mathbf{W}_{N} \mathbf{x}
$$

and represents the eigenvector of the matrix $\mathbf{W}_{N}$ which corresponds to the eigenvalue equal to 1 .

Example 94: The eigenvalue decomposition of the matrix $\mathbf{W}_{N}$ in Example 92 results in the eigenvector which corresponds to eigenvalue $\lambda_{k}=1$, whose elements are

$$
\mathbf{x}^{T}=\left[\begin{array}{llllllll}
1.33 & 1.52 & 2.18 & 0.79 & 0.55 & 0.18 & 0.48 & 0.97
\end{array}\right]
$$

The eigenvector is normalized by its mean value, and is obtained via the iterative solution after 11 iterations.

### 21.6 Random Walk

Assume that the signal, $x(n)$, represents the probability that a random walker is present at a vertex $n$. The random walker will then transit from the vertex $n$ to one of its neighboring vertices, $m$, with probability $p_{n m}$. There are several ways to define this probability and the corresponding forms of random walk; for an extensive review see Masuda et al. (2017). Here, we consider two random-walk definitions:

- vertex-centric random walk, and
- edge-centric random walk.

In the vertex-centric random walk, the probability, $p_{n m}$, that a random walker will transit from the vertex $n$ to one of its neighboring vertices, $m$, is defined by

$$
\begin{equation*}
p_{n m}=\frac{W_{n m}}{\sum_{m} W_{n m}}=\frac{1}{d_{n}} W_{n m}, \tag{21.9}
\end{equation*}
$$

where $W_{n m}$ are the affinities of the walker to transit from a vertex $n$ to a vertex $m$, and $d_{n}=\sum_{m} W_{n m}$ is the degree of a vertex $n$. The probability, $x_{p+1}(m)$, that a walker is at the vertex $m$ at the time step $(p+1)$ is then equal to the sum of all probabilities that a walker was at one the vertices, $n$, with the distance equal to one (neighboring vertices
to the vertex $m$ ) multiplied by the probabilities that the walker transits from the vertex $n$ to the vertex $m$, that is

$$
\begin{equation*}
x_{p+1}(m)=\sum_{n} x_{p}(n) p_{n m}=\sum_{n} x_{p}(n) \frac{1}{d_{n}} W_{n m} . \tag{21.10}
\end{equation*}
$$

The calculation of the signal $x(n)$ can now be naturally considered within the graph framework, where $W_{n m}$ are edge weights.

The probabilities at the stage $(p+1)$ of the random walk transition are calculated starting from the probabilities at the previous stage as in (21.10), which, in the compact matrix form, is given by

$$
\mathbf{x}_{p+1}=\mathbf{W D}^{-1} \mathbf{x}_{p}
$$

or

$$
\mathbf{D}^{-1 / 2} \mathbf{x}_{p+1}=\mathbf{D}^{-1 / 2} \mathbf{W} \mathbf{D}^{-1 / 2} \mathbf{D}^{-1 / 2} \mathbf{x}_{p},
$$

where the matrix $\mathbf{W}$ is a matrix of weighting coefficients and $\mathbf{D}$ is the degree matrix.

In the steady state, when $\mathbf{x}_{p+1}=\mathbf{x}_{p}=\mathbf{x}$, we have

$$
\mathbf{y}=\mathbf{D}^{-1 / 2} \mathbf{W} \mathbf{D}^{-1 / 2} \mathbf{y}
$$

where $\mathbf{y}=\mathbf{D}^{-1 / 2} \mathbf{x}$. The solution is the smoothest eigenvector of the normalized Laplacian, $\mathbf{L}_{N}=\mathbf{I}-\mathbf{D}^{-1 / 2} \mathbf{W D}^{-1 / 2}$, calculated from

$$
\left(\mathbf{I}-\mathbf{D}^{-1 / 2} \mathbf{W} \mathbf{D}^{-1 / 2}\right) \mathbf{y}=\mathbf{0},
$$

and is given by $\mathbf{y}=[1,1, \ldots, 1]^{T} / \sqrt{N}$ or

$$
\mathbf{x}=\mathbf{D}^{1 / 2}[1,1, \ldots, 1]^{T} / \sqrt{N}
$$

Note that the vector $\mathbf{x}$ is not constant, and its elements are given by $x(n)=\sqrt{d_{n} / N}$.

In the edge-centric random walk, the probability, $p_{n m}$, is defined by

$$
\begin{equation*}
x_{p+1}(m)=\sum_{n} x_{p}(n) p_{n m}=\frac{1}{d_{m}} \sum_{n} x_{p}(n) W_{n m} . \tag{21.11}
\end{equation*}
$$

In this case, the in-flow probability $\sum_{n} x_{p}(n) W_{n m}$ for the vertex $m$ is equal (balanced) to the out-flow probability of this vertex, $x_{p+1}(m) d_{m}=$ $\sum_{n} x_{p+1}(m) W_{n m}$. This model of random walk is also called the fluid
model and it has a simple interpretation within the electric circuits framework, since the probabilities (if considered as the electric potentials) satisfy the first Kirchoff low for the vertex $m$ serving as an electric circuit node, that is

$$
\sum_{n}\left(x_{p+1}(m)-x_{p}(n)\right) W_{n m}=0 .
$$

The matrix form of the edge-centric random walk is given by

$$
\mathbf{x}_{p+1}=\mathbf{D}^{-1} \mathbf{W} \mathbf{x}_{p}
$$

or $\mathbf{D} \mathbf{x}_{p+1}=\mathbf{W} \mathbf{x}_{p}$. In the steady state, for $\mathbf{x}_{p+1}=\mathbf{x}_{p}=\mathbf{x}$, we have

$$
\mathbf{D x}=\mathbf{W} \mathbf{x}
$$

or

$$
\begin{equation*}
\mathbf{L x}=\mathbf{0} . \tag{21.12}
\end{equation*}
$$

The solution to this equation is the smoothest (constant) eigenvector of the graph Laplacian, $\mathbf{x}=[1,1, \ldots, 1]^{T} / \sqrt{N}$.

The presented graph theory framework admits for various problem formulations and solutions.

Example 95: Consider the graph from Figure 2.2 in Part I and the case where we desire to find the probabilities, $x(n)$, that the walker reaches vertex 5 before it reaches vertex 7 , starting from any vertex $n$, and assuming that transition probabilities may be defined according the edge-centered random walk model. We therefore have to solve the system $\mathbf{L x}=\mathbf{0}$, with $x(5)=1$ and $x(7)=0$.

In the same way, we can solve another practically interesting problem. A piece of information has reached a member of the social network in Figure 8.10 at vertex 4 (in green circle), but it has not reached the member at vertex 3 (in red circle). The task is to find probabilities that the information is known to a vertex $n$.

Since the information is present at vertex 4 , then $x(4)=1$ is a certain event, and the fact that the information has not reached vertex 3 means that $x(3)=0$. Again, according to the analysis from (21.9) to (21.12), we have to solve the system $\mathbf{L x}=\mathbf{0}$, with $x(4)=1$ and


Figure 8.10: A small social network from Figure 8.8, where we are interested in the probability that a piece of news has reached vertex 4 (policeman, in green circle), but has not reached vertex 3 (chef, in red circle). Example 95 considers this scenario within the framework of random walk on graphs.
$x(3)=0$, that is

$$
\left[\begin{array}{cccccccc}
3 & -1 & -1 & -1 & 0 & 0 & 0 & 0  \tag{21.13}\\
-1 & 3 & -1 & 0 & -1 & 0 & 0 & 0 \\
-1 & -1 & 4 & -1 & -1 & 0 & 0 & 0 \\
-1 & 0 & -1 & 3 & 0 & 0 & -1 & 0 \\
0 & -1 & -1 & 0 & 4 & -1 & 0 & -1 \\
0 & 0 & 0 & 0 & -1 & 2 & 0 & -1 \\
0 & 0 & 0 & -1 & 0 & 0 & 2 & -1 \\
0 & 0 & 0 & 0 & -1 & -1 & -1 & 3
\end{array}\right]\left[\begin{array}{c}
x(0) \\
x(1) \\
x(2) \\
0 \\
1 \\
x(5) \\
x(6) \\
x(7)
\end{array}\right]=\mathbf{0}
$$

where the corresponding columns and rows are removed (rows for the known signal values, $x(3)$ and $x(4)$, and the column for the zero-valued signal, $x(3)$ ), while the green font designates the column to be moved on the right side of the equation for the known signal value, $x(4)=1$.

The solution is obtained from

$$
\left[\begin{array}{cccccc}
3 & -1 & -1 & 0 & 0 & 0  \tag{21.14}\\
-1 & 3 & -1 & 0 & 0 & 0 \\
-1 & -1 & 4 & 0 & 0 & 0 \\
0 & 0 & 0 & 2 & 0 & -1 \\
0 & 0 & 0 & 0 & 2 & -1 \\
0 & 0 & 0 & -1 & -1 & 3
\end{array}\right]\left[\begin{array}{l}
x(0) \\
x(1) \\
x(2) \\
x(5) \\
x(6) \\
x(7)
\end{array}\right]=\left[\begin{array}{c}
0 \\
1 \\
1 \\
1 \\
0 \\
1
\end{array}\right]
$$

with the inserted values $x(4)=1$ and $x(3)=0$, in the following form

$$
\mathbf{x}=\left[\begin{array}{lll}
0.375, & 0.625, & 0.5,
\end{array} 0,1,0.875,0.375,0.75\right]^{T}
$$

This means that the information is most probably available to the vertex 5 , with probability $x(5)=0.875$, while the lowest probability is that the information is available to the vertices 0 or 6 , with probability $x(0)=x(6)=0.375$, as can be expected from an intuitive analysis of this graph with a small number of vertices.

### 21.7 Hitting and Commute Time

The random walk problem is closely related to the hitting and commute time. The hitting time, $h(m, n)$, from a vertex $m$ to any vertex $n$ is defined as the expected number of steps for a random walker to travel from the vertex $m$ to a vertex $n$. Denote by $x_{p}^{(m)}(l)$ the hitting time from the reference vertex $m$ to the vertices $l$ which are the neighboring vertices of the considered vertex $n$. Then, the random walker will arrive from a vertex $l$ to the vertex $n$ in one step with the probability that it chooses to transit from the specific $l$ to the considered $n$. The probability that a random walker is at the neighboring vertex $l$, and then transits to the vertex, $n$, is given by

$$
p_{l n}=\frac{W_{l n}}{\sum_{k} W_{n k}}=\frac{1}{d_{n}} W_{l n} .
$$

The hitting time for a vertex $n$ is equal to the sum of all hitting times of neighboring vertices, with one step added, that is

$$
x_{p+1}^{(m)}(n)=\sum_{l} x_{p}^{(m)}(l) p_{l n}+1=\frac{1}{d_{n}} \sum_{l} x_{p}^{(m)}(l) W_{l n}+1
$$

The matrix form of this equation is

$$
\mathbf{x}_{p+1}^{(m)}=\mathbf{D}^{-1} \mathbf{W} \mathbf{x}_{p}^{(m)}+\left[\begin{array}{c}
1 \\
1 \\
\vdots \\
1
\end{array}\right] .
$$

In the steady state, we have

$$
\mathbf{D} \mathbf{x}^{(m)}=\mathbf{W} \mathbf{x}^{(m)}+\mathbf{d},
$$

where $\mathbf{d}=\mathbf{D}[1,1, \ldots, 1]^{T}$ is a degree vector. Finally the hitting time, $h(m, n)=x^{(m)}(n)$, is a solution to the linear system of equations

$$
\begin{equation*}
\mathbf{L}_{m} \mathbf{x}^{(m)}=\mathbf{d} \tag{21.15}
\end{equation*}
$$

with the reference vertex $m$, where $x(m)=0$ is removed from the vector $\mathbf{x}$ to form $\mathbf{x}^{(m)}$ with elements $h(m, n), n=0,1, \ldots, N-1, n \neq m$. The equation for vertex $m$ is also removed, so that the system is of an ( $N-1$ )-order and the matrix $\mathbf{L}_{m}$ is obtained from the graph Laplacian, $\mathbf{L}$, by removing its $m$ th row and $m$ th column.

Example 96: We shall calculate the hitting time for all vertices, $n$, from the vertex $m=3$ for the graph from Figure 2.2 in Part I. For this graph, we have

$$
\left[\begin{array}{lrrrrrr}
1.21 & -0.23 & -0.74 & 0 & 0 & 0 & 0 \\
-0.23 & 0.81 & -0.35 & -0.23 & 0 & 0 & 0 \\
-0.74 & -0.35 & 1.59 & -0.24 & 0 & 0 & 0 \\
0 & -0.23 & -0.24 & 1.12 & -0.51 & 0 & -0.14 \\
0 & 0 & 0 & -0.51 & 0.66 & 0 & -0.15 \\
0 & 0 & 0 & 0 & 0 & 0.64 & -0.32 \\
0 & 0 & 0 & -0.14 & -0.15 & -0.32 & 0.61
\end{array}\right]\left[\begin{array}{l}
h(3,0) \\
h(3,1) \\
h(3,2) \\
h(3,4) \\
h(3,5) \\
h(3,6) \\
h(3,7)
\end{array}\right]=\left[\begin{array}{l}
1.21 \\
0.81 \\
1.59 \\
1.12 \\
0.66 \\
0.64 \\
0.61
\end{array}\right]
$$

and this matrix is obtained from the graph Laplacian by removing the row and column corresponding to $m=3$. The hitting times from the vertex $m=3$ are then obtained as

$$
\left[\begin{array}{l}
h(3,0) \\
h(3,1) \\
h(3,2) \\
h(3,4) \\
h(3,5) \\
h(3,6) \\
h(3,7)
\end{array}\right]=\left[\begin{array}{c}
9.0155 \\
11.3003 \\
9.5942 \\
12.6594 \\
13.1427 \\
6.1930 \\
10.3860
\end{array}\right] .
$$

The commute time, $C T(m, n)$ between vertices $m$ and $n$ is defined as the expected time for the random walker to reach a vertex $n$ starting from vertex $m$, and then to return (see Part I, Section 4.5), to give

$$
C T(m, n)=h(m, n)+h(n, m) .
$$

Example 97: We consider the task of finding the commute time between the vertices $m=0$ and $n=N-1=7$ for the graph from Figure 2 in Part I, also shown in Figure 8.11. If we desire to use the full Laplacian matrix and the electric circuit framework for the calculation of the hitting time, then we should include the $m$ th equation with $h(m, m)=x(m)=0$. Since the sum of all external sources (on the right side of the Equation (21.15)) must be zero, this means that for the vertex $m=0$, the right side terms should be ( $d_{0}-D$ ), and the full Laplacian form of (21.15) for the vertex $m=0$ becomes

$$
\mathbf{L}\left[\begin{array}{c}
0 \\
h(0,1) \\
\vdots \\
h(0,6) \\
h(0,7)
\end{array}\right]=\left[\begin{array}{c}
d_{0}-D \\
d_{1} \\
\vdots \\
d_{6} \\
d_{7}
\end{array}\right],
$$

where $D=\sum_{i=0}^{N-1} d_{i}$, and $d_{i}=\sum_{n} W_{i n}$ are the degrees of vertices, $i$.
The same relation can be written for $m=7$ (or any other vertex $m$ ), to yield

$$
\mathbf{L}\left[\begin{array}{c}
h(7,0) \\
h(7,1) \\
\vdots \\
h(7,6) \\
0
\end{array}\right]=\left[\begin{array}{c}
d_{0} \\
d_{1} \\
\vdots \\
d_{6} \\
d_{7}-D
\end{array}\right] .
$$

The difference between the two previous systems of equations is

$$
\mathbf{L}\left[\begin{array}{c}
-h(7,0) \\
h(0,1)-h(7,1) \\
\vdots \\
h(0,6)-h(7,6) \\
h(0,7)
\end{array}\right]=\left[\begin{array}{c}
-D \\
0 \\
\vdots \\
0 \\
D
\end{array}\right] .
$$



Figure 8.11: Electric circuit interpretation of the commute time, $C T(m, n)=$ $D R_{\text {eff }}^{(m, n)}$.

This system can be interpreted within the electric circuit framework as the electric circuit with an external source at $m=0$ whose current is $i(0)=-D$. This external source is closed at $m=7$ with the current $i(7)=D$, while there are no sources at any other vertex. The difference of voltages in this electric circuit at $m=0$ and $m=7$ is equal to the difference of the seventh element, $h(0,7)$, and the first element, $-h(0,7)$, to yield

$$
\begin{aligned}
x_{0,7} & =h(0,7)-(-h(7,0))=h(0,7)+h(7,0) \\
& =C T(0,7)=R_{\mathrm{eff}}^{(0,7)} i(7)
\end{aligned}
$$

where $R_{\text {eff }}^{(0,7)}$ is the effective electric resistance between $m=0$ and $m=N-1=7$, as illustrated in Figure 8.11.

Finally, the previous relation holds for any two vertices, $m$ and $n$, that is

$$
C T(m, n)=D R_{\mathrm{eff}}^{(m, n)}
$$

where $D=\sum_{i=0}^{N-1} d_{i}$.
Example 98: The commute time between vertices $m=0$ and $n=7$ for the graph from Figure 2.2 in Part I, also shown in Figure 8.11, can be obtained by calculating the hitting times $h(0,7)$ and $h(7,0)$, as in Example 96. The result is

$$
C T(7,0)=h(0,7)+h(7,0)=10.7436+19.6524=30.3960
$$

The same result can be obtained by finding the effective resistance between vertices $m=0$ and $n=7$ in the electric circuit from Figure 8.11
using the elementary calculations for the effective resistance, $R_{\text {eff }}^{(7,0)}$, to give

$$
R_{\mathrm{eff}}^{(7,0)}=4.0745
$$

With $D=\sum_{i=0}^{7} d_{i}=7.46$, the commute time, $C T(7,0)=D R_{\text {eff }}^{(7,0)}=$ 30.3960, follows.

### 21.8 Relating Gaussian Random Signal to Electric Circuits

Consider a random graph signal, $x(n)$, and assume that each sample is Gaussian distributed with mean, $\mu_{n}$, and standard deviation, $\sigma_{n}$. Assuming that the signal values are correlated, the pdf of the signal $\mathbf{x}$ is given by

$$
\begin{equation*}
P(\mathbf{x})=\frac{1}{\sqrt{(2 \pi)^{N}}} \sqrt{\operatorname{det}\left(\boldsymbol{\Sigma}_{x}^{-1}\right)} \exp \left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{T} \boldsymbol{\Sigma}_{x}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right) \tag{21.16}
\end{equation*}
$$

The inverse of the autocovariance matrix is the precision matrix $\mathbf{Q}=\boldsymbol{\Sigma}_{x}^{-1}$. Note that the term precision comes from the one-dimensional case where the precision is inversely proportional to the variance, that is $Q=1 / \sigma^{2}$.

The maximum likelihood estimate of $\mathbf{x}$ is then obtained from (21.16) by minimizing

$$
E_{x}=\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{T} \boldsymbol{\Sigma}_{x}^{-1}(\mathbf{x}-\boldsymbol{\mu})
$$

and the solution is

$$
\begin{equation*}
\boldsymbol{\Sigma}_{x}^{-1}(\mathbf{x}-\boldsymbol{\mu})=\mathbf{0} \tag{21.17}
\end{equation*}
$$

For a zero-mean random signal, $\boldsymbol{\mu}=\mathbf{0}$ and $\boldsymbol{\Sigma}_{x}^{-1} \mathbf{x}=\mathbf{0}$, and the solution in (21.17) corresponds to minimizing the energy of the change (maximal smoothness) in the graph.

The generalized Laplacian corresponding to the precision matrix is defined by

$$
\boldsymbol{\Sigma}_{x}^{-1}=\mathbf{Q}=\mathbf{L}+\mathbf{P}
$$

where $\mathbf{P}$ is a diagonal matrix such that the sum of columns of the Laplacian is zero.

The edge weights can now be extracted from the Laplacian matrix. Since the Laplacian is defined using the observed graph signal values,
this is a point where the presented analysis meets the discussion from the previous section (see also Examples 79 and 80). The electric circuit form of the minimization condition is obtained from

$$
(\mathbf{L}+\mathbf{P})(\mathbf{x}-\boldsymbol{\mu})=\mathbf{0}
$$

or

$$
\mathbf{L} \mathbf{x}=-\mathbf{P} \mathbf{x}+(\mathbf{L}+\mathbf{P}) \boldsymbol{\mu}
$$

In terms of the external current generators, we can define the problem as

$$
\mathbf{L x}=\mathbf{i}_{x}+\mathbf{i}_{g}
$$

where $\mathbf{i}_{x}=-\mathbf{P x}$ are voltage-driven current generators and $\mathbf{i}_{g}=$ $(\mathbf{L}+\mathbf{P}) \boldsymbol{\mu}=\mathbf{Q} \boldsymbol{\mu}$ are constant external current generators. Therefore, the steady-state solution can be interpreted and solved in the same way as we solve the described electric circuit. For example, if the observed state is $x(7)=1$ and $\mu(n)=0$, we can solve the system for other values of $x(n)$ for a given matrix $\boldsymbol{\Sigma}_{x}^{-1}=\mathbf{Q}=\mathbf{L}+\mathbf{P}$.

## 22

## Graph Learning from Data and External Sources

In the seminal work on graph topologies by Gabriel Kron, all vertices are assumed to be separable into two groups: active vertices and inner vertices. Active vertices are exposed to external sources which may be of different natures, depending on the physical system represented by a graph. Scenarios which admit the grouping into the active and inner vertices include the following.

- In a graph representing an electric circuit, active vertices are characterized by external currents/voltages which supply the network with energy, which is consumed in its resistive edges.
- For graphs representing heat transfer, active vertices must include sources/sinks of heat energy.
- In a transportation network, active vertices are those stations where new influx of passengers can be generated, in contrast with e.g., the transition hubs, where the passengers can only change the lines, and cannot exit or enter the station.
- Active and inner vertices can also be recognized in the postal service network. Here, the external vertices are the points where
the mail is accepted and/or delivered from/to outside world, while the inner vertices are places where the mail is only in transit (sorted and redirected).
- In a computer network, the inner vertices are the servers with no external input/output function, but only have the store and data transfer functionality. Computers that can generate input or output data would be active vertices (vertices with external sources).

Notice that the external sources drive the system (graph), while the inner sources can be organized in various ways to improve the efficiency of the system. In the Kron reduction of graphs, inner vertices are commonly reorganized by appropriate transformations of graphs, such as the "star-mesh" transformations.

In the previous section, learning of graph topology from data on the graph has been considered based on the correlation and precision matrices. The fundamental additional assumption is that the graph signal is smooth over the vertices. Notice that if we were able to measure the graph signal and external sources in the active vertices, then this would make it possible to learn graph topology in an exact way.

Consider the $p$ th observation of a signal on a graph,

$$
\mathbf{x}_{p}=\left[x_{p}(0), x_{p}(1), \ldots, x_{p}(N-1)\right]^{T},
$$

and the corresponding external sources,

$$
\mathbf{i}_{p}=\left[i_{p}(0), i_{p}(1), \ldots, i_{p}(N-1)\right]^{T} .
$$

Without loss of generality, assume that the $(N-1)$ th vertex is a reference, where $x_{p}(N-1)=0$ and $i_{p}(N-1)=-\sum_{n=0}^{N-2} i_{p}(n)$. These elements will be removed from the data and equations and only the data on remaining vertices will be considered, and denoted as $\mathbf{x}_{p}=$ $\left[x_{p}(0), x_{p}(1), \ldots, x_{p}(N-2)\right]^{T}$ and $\mathbf{i}_{p}=\left[i_{p}(0), i_{p}(1), \ldots, i_{p}(N-2)\right]^{T}$. The electric circuit equation in (21.1) for the reduced set of data is then

$$
\left[\begin{array}{c}
l_{0} \\
\boldsymbol{l}_{1} \\
\vdots \\
l_{N-2}
\end{array}\right] \mathbf{x}_{p}=\mathbf{i}_{p}
$$

where

$$
\boldsymbol{l}_{i}=\left[L_{i 0}, L_{i 1}, \ldots, L_{i, N-2}\right]
$$

are the rows of the graph Laplacian matrix, $\mathbf{L}$, with the elements ranging from $n=0$ to $n=N-2$. In other words, the last element and the last row in the graph Laplacian, which correspond to the reference vertex, $n=N-1$, are omitted.

If $P$ sets of observations are available, then we can write this system in the form

$$
\left[\begin{array}{c}
\boldsymbol{l}_{0} \\
\boldsymbol{l}_{1} \\
\vdots \\
\boldsymbol{l}_{N-2}
\end{array}\right]\left[\mathbf{x}_{1}, \mathbf{x}_{2}, \ldots, \mathbf{x}_{P}\right]=\left[\mathbf{i}_{1}, \mathbf{i}_{2}, \ldots, \mathbf{i}_{P}\right] .
$$

or

$$
\left[\begin{array}{c}
\boldsymbol{l}_{0} \\
\boldsymbol{l}_{1} \\
\vdots \\
\boldsymbol{l}_{N-2}
\end{array}\right] \mathbf{X}_{N-1, P}=\mathbf{J}_{N-1, P}
$$

The matrices $\mathbf{X}_{N-1, P}$ and $\mathbf{J}_{N-1, P}$ represent respectively the signal on the graph and external source matrices of dimensionality $(N-1) \times P$.

Now, we can consider the following cases.

- With $P \geq N-1$ independent available observations, the exact form of the graph Laplacian (its first ( $N-1$ ) rows and columns) follows from

$$
\left[\begin{array}{c}
\boldsymbol{l}_{0}  \tag{22.1}\\
\boldsymbol{l}_{1} \\
\vdots \\
\boldsymbol{l}_{N-2}
\end{array}\right]=\mathbf{J}_{N-1, P} \operatorname{pinv}\left\{\mathbf{X}_{N-1, P}\right\} .
$$

The last column and the last row of the graph Laplacian, $\mathbf{L}$, are formed in such a way that the sum over every column or row is zero.

- A more complex case arises when $P<N-1$. Then, the number of observations is not sufficient to recover the graph Laplacian.

However, if we assume that the graph Laplacian is sparse, with a small number of nonzero elements (edges), the solution is achievable within the compressive sensing framework. In order to adapt the system in (22.1) to suit the standard LASSO algorithm, we shall rewrite it in the form

$$
\mathbf{X}_{N-1, P}^{T}\left[\begin{array}{c}
\boldsymbol{l}_{0} \\
\boldsymbol{l}_{1} \\
\vdots \\
\boldsymbol{l}_{N-2}
\end{array}\right]^{T}=\mathbf{J}_{N-1, P}^{T}
$$

Now, LASSO minimization can be performed for each data column, $\boldsymbol{l}_{k}^{T}$, and the corresponding column of the external source matrix $\mathbf{J}_{N-1, P}^{T}$, denoted by $\boldsymbol{i}_{k}$, in the form

$$
\boldsymbol{l}_{k}=\operatorname{lasso}\left(\mathbf{X}_{N-1, P}^{T}, \boldsymbol{i}_{k}, \rho\right) .
$$

Another approach would be to transform the matrices with graph Laplacian rows, $\boldsymbol{i}_{k}$, and external source matrix, $\mathbf{J}_{N-1, P}^{T}$, into column vectors to have

$$
\left[\begin{array}{lclc}
\mathbf{X}_{N-1, P}^{T} & \mathbf{0} & \cdots & \mathbf{0} \\
\mathbf{0} & \mathbf{X}_{N-1, P}^{T} & \cdots & \mathbf{0} \\
\vdots & \vdots & \ddots & \vdots \\
\mathbf{0} & \mathbf{0} & \ldots & \mathbf{X}_{N-1, P}^{T}
\end{array}\right]\left[\begin{array}{c}
\boldsymbol{l}_{0}^{T} \\
\boldsymbol{l}_{1}^{T} \\
\vdots \\
\boldsymbol{l}_{N-2}^{T}
\end{array}\right]=\left[\begin{array}{c}
\boldsymbol{i}_{0} \\
\boldsymbol{i}_{1} \\
\vdots \\
\boldsymbol{i}_{N-2}
\end{array}\right] .
$$

Using the notation

$$
\left(\mathbf{I}_{N-1, N-1} \otimes \mathbf{X}_{N-1, P}^{T}\right) \boldsymbol{l}_{\text {vec }}=\boldsymbol{i}_{\text {vec }}
$$

for the matrix and the vectors in the above equation (where $\mathbf{I}_{N-1, N-1}$ is the identity matrix), this system can be solved using

$$
\boldsymbol{l}_{\mathrm{vec}}=\operatorname{lasso}\left(\mathbf{I}_{N-1, N-1} \otimes \mathbf{X}_{N-1, P}^{T}, \boldsymbol{i}_{\mathrm{vec}}, \rho\right) .
$$

Example 99: Consider a graph with $N=50$ vertices, and with a small number of edges. Such a sparse graph Laplacian, $\mathbf{L}$, is shown in Figure 8.1(a). The graph Laplacian was estimated using a large number, $P=60$, of observations of the graph signal and external sources. Both


Figure 8.1: Estimation of the graph Laplacian, $\mathbf{L}$, for a graph with $N=50$ randomly positioned vertices and a small number of edges. (a) Ground truth graph Laplacian, L. (b) Estimated graph Laplacian using the norm-two for a large number of observations, $P=60>N=50$. (c) Estimated graph Laplacian using the LASSO, for a large number of observations, $P=60>N=50$. (d) Estimated graph Laplacian using the norm-two, for a small number of observations, $P=40<N=50$. (e) Estimated graph Laplacian using the LASSO, for a small number of observations, $P=40<N=50$.
the norm-two and the LASSO estimates of the graph Laplacian were accurate, as shown in Figures 8.1 (b) and (c). Next, the number of observations was reduced to $P=40<N=50$. In this case, the sparsity of the graph Laplacian is crucial for the solution. The LASSO algorithm, which includes the sparsity constraint, was still able to produce a good result, as seen from Figure 8.1(e). The norm-two was calculated using the pseudo-inverse of the data matrix, $\mathbf{X}_{N-1, P}^{T}$, and was not appropriate as the graph Laplacian estimator, as seen in Figure 8.1(d).

Finally, we should mention that in Example 99 we have not used the conditions that the graph Laplacian is a symmetric matrix and that the elements of the weight matrix, $W_{m n}$, from which the graph Laplacian elements are formed, are nonnegative. These conditions can be used within linear programming formulations to improve the estimation.

## 23

## Random Signal Simulation on Graphs

This section addresses ways to simulate graph signals, as for testing of any new method for learning graph topology based on the available data, we have to show that the method is reliable and accurate on simulated graphs and graph signals. To this end, we have to assume a graph (randomly structured) and then to simulate data on such a graph. Such data should exhibit a desired degree of randomness, in order to infer the influence of graph connections to the signal form. Notice that a graph signal should be influenced by the structure of a graph in an implicit way. This influence is then used as the basis for graph topology learning. Graph signal simulation is not straightforward and certainly not a unique process. Some of the presented approaches to graph signal simulation are based on the previously introduced analysis of signals on graphs with physically well-defined topology in Section 21.

The representation of a graph and graph signal within the circuit theory framework can be used to simulate random signals on graphs. While several approaches are possible, we will here present some of the most frequently used ones.
(1) Assume that the graph is initiated by external sources that are random variables. In that case, the $p$ th observation of a random
signal on this graph is simulated as a solution of the system of equations

$$
\mathbf{L x}_{p}=\varepsilon_{p},
$$

with $\mathbf{i}_{p}=\varepsilon_{p}$. Note that one of the external sources (randomly chosen for each observation $p$ ) should compensate for all other sources, to ensure $\sum_{n=0}^{N-1} \varepsilon_{p}(n)=0$.
Since the graph Laplacian matrix is singular, the graph signal value (the electric potential in the electric circuit case) at a vertex, for example, for $n=0$, should be considered as a reference and its value should be assumed, for example, $x(0)=0$. This strategy may be used whenever the inversion of the graph Laplacian is required.
(2) The graph is excited at only one of its vertices (and one additional reference vertex) with a random external zero-mean white source. The position of these vertices is randomly selected for each $p$. Then, the random signal observation on a graph is obtained as a solution to

$$
\mathbf{L} \mathbf{x}_{p}=\mathbf{i}_{p}
$$

where $i_{p}(n)=\varepsilon_{p} \delta\left(n-n_{i}\right)-\varepsilon_{p} \delta\left(n-n_{j}\right)$ and $n_{i}$ and $n_{j}$ are two randomly selected vertices at each observation.
(3) A minimal information needed to calculate a random graph signal is the knowledge of signal values at two randomly positioned vertices. Assuming that $x_{p}(n)=\varepsilon_{p} \delta\left(n-n_{i}\right)+\epsilon_{p} \delta\left(n-n_{j}\right)$ and $n_{i}$ and $n_{j}$ are two randomly selected vertices at each observation; then we may solve the system for all other signal samples, based on

$$
\mathbf{L x} x_{p}=\mathbf{0} .
$$

With the two assumed values, $x_{p}(n)$, at $n=n_{i}$ and $n=n_{j}$, we can solve this system for all other signal values. In the case of external sources, the values should be compensated (their sum should be zero), as mentioned earlier. In this case, there is no need for compensation, which means that $\varepsilon_{p}$ and $\epsilon_{p}$ could be independent random variables.
(4) The signal on a graph is formed using a linear combination of white noise $\varepsilon_{p}$ and its graph shifted versions. The output signal after $M$ such graph shifts, defined by the normalized Laplacian, is given by

$$
\begin{equation*}
\mathbf{x}_{p}=\left(h_{M} \mathbf{L}^{M}+h_{M-1} \mathbf{L}^{M-1}+\cdots+h_{1} \mathbf{L}^{1}+h_{0} \mathbf{L}^{0}\right) \boldsymbol{\varepsilon}_{p} \tag{23.1}
\end{equation*}
$$

The resulting graph signal can be written in the form

$$
\mathbf{x}_{p}=H(\mathbf{L}) \boldsymbol{\varepsilon}_{p}
$$

(5) Analysis based on the adjacency matrix and graph shifts. Assume that an undirected graph with the adjacency matrix $\mathbf{A}$, is excited at $N_{a}$ randomly chosen vertices $n_{1}, n_{2}, \ldots, n_{N_{a}}, \eta=N_{a} / N$, with spikes $\delta\left(n-n_{i}\right), i=1,2, \ldots, N_{a}$. After shifting these spikes $K$ times, we obtain

$$
\mathbf{x}=\mathbf{A}^{K} \sum_{i=1}^{N_{a}} \boldsymbol{\delta}_{n_{i}}
$$

where $\boldsymbol{\delta}_{n_{i}}$ is a graph signal with a nonzero value at $n=n_{i}$ only. The parameters $K$ and $N_{a}$ define the resulting signal smoothness. An example of one realization of such a signal is presented in Part II, Figure 3.6 for $\eta=1 / 8, K=1$ (upper subplots) and $\eta=2 / 8$, $K=1$ (lower subplots) using the spikes $a_{i} \delta\left(n-n_{i}\right)$, where $a_{i}$ are the spike amplitudes.
(6) Signals are commonly simulated as sums of the harmonic basis functions, as in classical Fourier analysis. This kind of simulation may be used in graph signal processing, too. Such a signal on a graph can be written as

$$
\mathbf{x}=\sum_{i=1}^{K} a_{k_{i}} \mathbf{u}_{k_{i}}
$$

where $\mathbf{u}_{k}$ are the eigenvectors of the Laplacian or adjacency matrix, and $a_{k}$ are random constants. This kind of graph signal simulation, with or without an additive noise, has been often used in this part.

## 24

## Summary of Graph Learning from Data Using Probabilistic Generative Models

Analytics of data on graphs with known or given topologies are feasible for applications that involve physically meaningful structures, such as citation networks, transport networks and observable social networks. In those applications, various vertex or spectral domain techniques, as mentioned in Part II of this monograph, have been successfully implemented and developed to filter, analyze or visualize graph signals. However, in many situations where the graph topology cannot be directly observed or even when the data is partially observed, the inference of graph structure is a key first step.
Remark 98: Given the observed graph data, graph topology learning is an ill-posed problem. In other words, totally different graphs can generate the same data, while one set of observed data can result in different graph topologies, depending on the graph learning method used. Thus, to infer graph topology we need to employ some priors, for example, to match statistics by imposing sparsity or smoothness conditions on the graph.

Previous sections in this monograph have introduced various sparsity promotion techniques, such as the graphical LASSO (GLASSO) and smoothness constrained graph learning, mostly from the perspective
of linear algebra (Dong et al., 2019; Giannakis et al., 2018; Mateos et al., 2019). However, it is more natural to connect and summarise such techniques under the umbrella of probabilistic generative models. A straightforward approach would be on the basis of some fundamental statistical models, such as the covariance or precision matrices of Gaussian distribution (due to their positive definiteness property), the Gaussian Markov random fields with local independence prior, or a factor analysis models with smoothness assumption. We also envisage further progress of generative models to be based on the concept of diffusion processes on graphs, whereby the signal generating process can be regarded as the graph signal that has been diffused by some graph kernels (for example, polynomial kernels) from a white Gaussian distributed noise.

Generally speaking, graph learning can be treated as an inverse operation to the graph data generation process, that is, $\mathbf{x}=f_{\mathcal{G}}(\mathbf{z})$, where $\mathbf{x}$ denotes the observed data. Here, the data are considered to be the output of an unknown transform (denoted by $f_{\mathcal{G}}$ ) of some initial state, $\mathbf{z}$, on the graph $\mathcal{G}$. Existing literature on learning a graph can be thought of as an attempt to infer the generative process, $f_{\mathcal{G}}$, by matching the data statistics, $\mathbf{x}$, with different priors that are imposed on $\mathbf{z}$. We should point out that in this section we discuss the problem of learning graphs from fully observed graph data because of its underpinning role in a number of advanced techniques, such as graph learning with partially observed data (Grotas et al., 2019; Wai et al., 2019) and dynamic graph learning (Chen et al., 2011; Ioannidis et al., 2019b; Kaplan, 2008).

### 24.1 Basic Gaussian Models

The simplest way of constructing a graph would be to associate edge weights with the covariance of data observed on a graph; this is reasonable under the Gaussian assumption, since the first two moments fully capture the whole statistics of the distribution. Indeed, the non-zero elements of the covariance matrix of graph data naturally provide consistent estimation of the connectivity within a graph. This method is explained within the introductory part of Section 19.

Given a set of $P$ independent and identically distributed (i.i.d.) observed data vectors, $\mathbf{x}_{1}, \mathbf{x}_{2}, \ldots, \mathbf{x}_{P}$, the empirical sample covariance is calculated as

$$
\begin{equation*}
\Sigma_{x}(m, n)=\frac{1}{P} \sum_{p=1}^{P}\left(x_{p}(m)-\mu(m)\right)\left(x_{p}(n)-\mu(n)\right), \tag{24.1}
\end{equation*}
$$

or

$$
\begin{equation*}
\boldsymbol{\Sigma}=\frac{1}{P} \sum_{p=1}^{P}\left(\mathbf{x}_{p}-\overline{\mathbf{x}}\right)\left(\mathbf{x}_{p}-\overline{\mathbf{x}}\right)^{T} \tag{24.2}
\end{equation*}
$$

where $\overline{\mathbf{x}}$ is the mean value of the observed samples. Alternatively, a normalized version of $\boldsymbol{\Sigma}$ can also be employed in order to produce the edge weights as

$$
\begin{equation*}
\sigma_{x}(m, n)=\frac{\Sigma_{x}(m, n)}{\sqrt{\Sigma_{x}(m, m) \Sigma_{x}(n, n)}} . \tag{24.3}
\end{equation*}
$$

For this empirical statistics, we can employ a threshold, $\tau$, to designate the non-zero connections of the adjacency weight, in a way similar to (17.1), to yield

$$
W_{m, n}= \begin{cases}\sigma_{x}(m, n), & \left|\sigma_{x}(m, n)\right| \geq \tau  \tag{24.4}\\ 0, & \left|\sigma_{x}(m, n)\right|<\tau .\end{cases}
$$

A more sophisticated approach would be to use hypothesis testing via setting a false alarm rate, whereby

$$
\begin{equation*}
\mathcal{H}_{0}: \sigma_{x}(m, n)=0 \text { versus } \mathcal{H}_{1}: \sigma_{x}(m, n) \neq 0 \tag{24.5}
\end{equation*}
$$

In these scenarios, the empirical covariance is a common choice of test statistics. Although the density of $\sigma_{x}(m, n)$ may have closed-form representation, it typically needs numerical integration when calculating the $p$-values; however, transformations of $\sigma_{x}(m, n)$ can relax this issue to obtain closed-form densities. For example, under the Gaussian distribution and $\mathcal{H}_{0}$, the weighting

$$
s(m, n)=\frac{\sigma_{x}(m, n) \sqrt{P-2}}{\sqrt{1-\sigma_{x}^{2}(m, n)}}
$$

would satisfy a student t -distribution of $(P-2)$ degrees of freedom, and

$$
s(m, n)=\tanh ^{-1}\left(\sigma_{x}(m, n)\right)
$$

would then result in a Gaussian distribution with zero mean and $1 /(P-3)$ variance (see Chapter 7.3.1 Kolaczyk, 2009). In those transformed test statistics, the statistical significance can be easily adjusted to meet the false alarm rate. However, the limitation of this model is that by employing individual tests, the number of implementations in inferring the graph grows as $\mathcal{O}\left(N^{2}\right)$. This is computational prohibitive for relatively large graphs; on the other hand, this leads to increasingly false judgments even with a constant false alarm rate.

A further possible misleading due to the correlation models stems from the fact that the correlation does not mean the causation. In other words, the $m$ th and $n$th vertices can show a strong correlation when they are both highly influenced by an intermediate vertex, however, they are not the causation of one another, as illustrated in Example 74.

### 24.2 Gaussian Graphical Model

To address the issues with the correlation and causation, and to be able to construct a graph that reflects only direct relationships among its vertices, one classical method employs the partial correlation, whereby the correlation of two vertices is calculated by eliminating associations of other contributing vertices. Under the assumption that vertices satisfy some mild distributions such as elliptical distributions, the partial correlation coincides with the conditional correlation (Baba et al., 2004), and further equals to the conditional independence under the Gaussian assumption on vertices; this allows the partial correlation to be explicitly related to the precision matrix. The so established relationship is crucial in understanding other techniques such as the GLASSO, graph regression and other generative models.

### 24.2.1 Partial Correlation Model

In order to simplify the notation, and without loss of generality, we shall consider vertices $n=0$ and $m=1$. The set of all other vertices, except for the $m$ th and the $n$th vertex, is denoted by $\mathcal{V} \backslash\{m, n\}=$ $\{2,3, \ldots, N-1\}$. Define the data vectors at each vertex by $\mathbf{y}_{n}$, as in (19.1), and denote by $\hat{\mathbf{y}}_{0}$ and $\hat{\mathbf{y}}_{1}$ the best linear approximations to the
signal samples $\mathbf{y}_{0}$ and $\mathbf{y}_{1}$, obtained based on the data at other vertices, $\mathbf{y}_{2}, \mathbf{y}_{3}, \ldots, \mathbf{y}_{N-1}$. The new data values are then defined as

$$
\begin{aligned}
& \mathbf{z}_{0}=\mathbf{y}_{0}-\hat{\mathbf{y}}_{0} \\
& \mathbf{z}_{1}=\mathbf{y}_{1}-\hat{\mathbf{y}}_{1}
\end{aligned}
$$

Now, the (empirical) partial correlation between the vertices $m=0$ and $n=1$ can be defined as

$$
\begin{equation*}
\sigma_{z}(0,1)=\frac{\Sigma_{z}(0,1)}{\sqrt{\Sigma_{z}(0,0)} \sqrt{\Sigma_{z}(1,1)}} \tag{24.6}
\end{equation*}
$$

In a similar way, all other partial correlations, $\sigma_{z}(m, n)$, between pairs of vertices $m$ and $n$ are calculated. Then, one way of hypothesis testing is as follows

$$
\begin{equation*}
\mathcal{H}_{0}: \sigma_{z}(m, n)=0 \text { versus } \mathcal{H}_{1}: \sigma_{z}(m, n) \neq 0 \tag{24.7}
\end{equation*}
$$

where $\sigma_{z}(m, n)$ is employed as the test statistics. Other choices of test statistics, such as the Fisher's transform $s(m, n)=\tanh ^{-1}\left(\sigma_{z}(m, n)\right)$, also obtain an asymptotically Gaussian distribution (Chapter 7.3.2 Kolaczyk, 2009).

### 24.2.2 Gaussian Markov Random Field

A further assumption which can be imposed on the partial correlation model is that of the Gaussian distribution, which in many cases is a common setting as this facilitates closed-form solutions and ease of analysis. For example, under the Gaussian distribution, the partial correlation coincides with the conditional correlation (Baba et al., 2004), or equivalently, conditional independence; this in turn forms the pairwise Markov property of random fields, which constitutes a Gaussian Markov random field.

We shall denote the $m$ th and the $n$th elements of graph signal samples by $\mathbf{y}_{A}$, and all other elements except for the $m$ th and the $n$th element by $\mathbf{y}_{B}$. The covariance of $\mathbf{y}_{A}$ is then designated by $\boldsymbol{\Sigma}_{A A}$, and is of the size $2 \times 2$. The so obtained block structure of (24.2) becomes

$$
\boldsymbol{\Sigma}=\left[\begin{array}{ll}
\boldsymbol{\Sigma}_{A A} & \boldsymbol{\Sigma}_{A B}  \tag{24.8}\\
\boldsymbol{\Sigma}_{B A} & \boldsymbol{\Sigma}_{B B}
\end{array}\right]
$$

The covariance of the corresponding $\mathbf{y}_{A}$ conditioned on $\mathbf{y}_{B}$ is then easily obtained as

$$
\begin{equation*}
\boldsymbol{\Sigma}_{A \mid B}=\boldsymbol{\Sigma}_{A A}-\boldsymbol{\Sigma}_{A B} \boldsymbol{\Sigma}_{B B}^{-1} \boldsymbol{\Sigma}_{B A}, \tag{24.9}
\end{equation*}
$$

which is also called the Schur complement. On the other hand, to rewrite the expression in (24.8) with regard to the precision matrix, $\mathbf{Q}=\boldsymbol{\Sigma}^{-1}$, we can use the following block-wise matrix property

$$
\begin{align*}
\mathbf{Q} & =\boldsymbol{\Sigma}^{-1}=\left[\begin{array}{cc}
\boldsymbol{\Sigma}_{A \mid B}^{-1} & -\boldsymbol{\Sigma}_{A \mid B}^{-1} \boldsymbol{\Sigma}_{A B} \boldsymbol{\Sigma}_{B B}^{-1} \\
-\boldsymbol{\Sigma}_{B B}^{-1} \boldsymbol{\Sigma}_{A B} \boldsymbol{\Sigma}_{A \mid B}^{-1} & \boldsymbol{\Sigma}_{B B}^{-1} \boldsymbol{\Sigma}_{A B} \boldsymbol{\Sigma}_{A \mid B}^{-1} \boldsymbol{\Sigma}_{A B} \boldsymbol{\Sigma}_{B B}^{-1}
\end{array}\right] \\
& =\left[\begin{array}{ll}
\mathbf{Q}_{A A} & \mathbf{Q}_{A B} \\
\mathbf{Q}_{B A} & \mathbf{Q}_{B B}
\end{array}\right] . \tag{24.10}
\end{align*}
$$

From (24.10), observe that $\boldsymbol{\Sigma}_{A \mid B}=\mathbf{Q}_{A A}^{-1}$ if the inverse of $\mathbf{Q}_{A A}$ exists. In other words, to obtain the partial correlations in (24.9), it is more convenient to use the precision matrix than the covariance matrix. Thus, one feasible way to associate the edge weights is via

$$
\begin{equation*}
W_{m, n}=-\frac{Q(m, n)}{\sqrt{Q(m, m) Q(n, n)}}, \tag{24.11}
\end{equation*}
$$

where $\mathbf{Q}=\boldsymbol{\Sigma}^{-1}$ is the empirical precision matrix. Then, the association of edge weights can be used to infer non-zero elements of $W_{m, n}$, which is also known as the covariance selection problem (Dempster, 1972). One feasible method is to recursively update the graph by testing the hypotheses in the form

$$
\begin{equation*}
\mathcal{H}_{0}: W_{m, n}=0 \text { versus } \mathcal{H}_{1}: W_{m, n} \neq 0, \tag{24.12}
\end{equation*}
$$

where the $W_{m, n}$ is used as the test statistic. For large-scale graphs, however, this model also shows limitations which are similar to those of correlation models in Section 24.1. Although this model can relax the vagueness regarding the correlation and causation, it has one additional limitation, in that it requires the number of samples to be larger than the dimension of covariance to ensure a proper inverse of covariance; this does not necessarily hold, especially for large-scale graphs, as stated in Remark 93. The graphical LASSO and linear regression methods may be used to solve this issue.

### 24.2.3 Graphical LASSO and Regression Models

A common way of overcoming the problem of rank deficiency is to involve a regularization term when estimating the precision matrix.

Given the set of independent and identically distributed samples, $\mathbf{x}_{1}, \mathbf{x}_{2}, \ldots, \mathbf{x}_{P}$, the log-likelihood of a Gaussian distribution with zero mean and precision matrix $\mathbf{Q}$ is represented as in (19.19)

$$
\begin{align*}
J= & \sum_{p=1}^{P}\left(-\frac{1}{2} \mathbf{x}_{p}^{T} \mathbf{Q} \mathbf{x}_{p}-\frac{P}{2} \ln (2 \pi)+\frac{1}{2} \ln |\mathbf{Q}|\right)  \tag{24.13}\\
& \propto P \ln |\mathbf{Q}|-\sum_{p=1}^{P}\left(\mathbf{x}_{p}^{T} \mathbf{Q} \mathbf{x}_{p}\right), \tag{24.14}
\end{align*}
$$

where $|\mathbf{Q}|=\operatorname{det}(\mathbf{Q})$. The maximization of this log-likelihood yields the attained optimum in the form

$$
\mathbf{Q}^{-1}=\frac{1}{P} \sum_{p=1}^{P} \mathbf{x}_{p} \mathbf{x}_{p}^{T},
$$

as in (19.25)-(19.26).
However, when $P$ is smaller than the dimension of $\mathbf{x}_{p}$, the term $\sum_{p=1}^{P}\left(\mathbf{x}_{p} \mathbf{x}_{p}^{T}\right)$ is not full rank, which causes the singularity of $\mathbf{Q}$. One way of avoiding this issue is to use the $l_{1}$ norm to promote sparsity in (24.13), in a similar form to (19.27), to yield

$$
\begin{equation*}
\bar{J}=P \ln |\mathbf{Q}|-\sum_{p=1}^{P}\left(\mathbf{x}_{p}^{T} \mathbf{Q} \mathbf{x}_{p}\right)-\rho\|\mathbf{Q}\|_{1}, \tag{24.15}
\end{equation*}
$$

which is known as a GLASSO problem. As shown in Yuan and Lin (2007), in this way the correct graph can be inferred with the probability approaching one, when the parameter $\rho$ is chosen to satisfy $\rho \cdot P \rightarrow \infty$ and $\rho \cdot \sqrt{P} \rightarrow 0$, for $P \rightarrow \infty$.
Remark 99: Apart from the $l_{1}$ norm, other regularization strategies can also be employed in (24.15). For example, solving (24.15) could result in negative values, which of course have no meaning when associating the edge weights. Thus, constraining edge weights to be non-negative is also a common regularization approach in graph learning (see also

Section 19.3). For more detail, we refer to Friedman et al. (2008), Banerjee et al. (2008), and Yuan and Lin (2006).
Graph regression. Another perspective of learning the Gaussian graphical model (described in Section 19.1 and Example 75) is via a regression of data observed at each vertex, $\mathbf{y}_{m}$, given the data observations at other vertices, $\mathbf{y}_{n}, n \in\{0,1,2, \ldots, m-1, m+1, \ldots, N-1\}=\mathcal{V} \backslash\{m\}$. The aim of the regression here is to learn a graph that yields the minimum mean square error, given the observed samples. More specifically, the values $\beta_{n m}, n=\mathcal{V} \backslash\{m\}$, that minimize

$$
\begin{equation*}
J_{m}=\left\|\mathbf{y}_{m}-\sum_{n=0, n \neq m}^{N-1} \beta_{n m} \mathbf{y}_{n}\right\|_{2}^{2} \tag{24.16}
\end{equation*}
$$

follow from

$$
\left(\mathbf{y}_{m}-\sum_{n=0, n \neq m}^{N-1} \beta_{n m} \mathbf{y}_{n}\right) \mathbf{y}_{k}^{T}=\mathbf{0}
$$

or $\sum_{n=0, n \neq m}^{N-1} \beta_{n m} \Sigma_{x}(n, k)=\Sigma_{x}(m, k)$, for $k, n \in \mathcal{V} \backslash\{m\}$. A matrix solution to this equation is

$$
\boldsymbol{\beta}_{m}=\boldsymbol{\Sigma}_{m m}^{-1} \boldsymbol{\Sigma}_{1 m}
$$

where $\boldsymbol{\Sigma}_{1 m}$ is a vector with $(N-1)$ elements $\Sigma_{x}(m, k), k=\mathcal{V} \backslash\{m\}$, and $\boldsymbol{\Sigma}_{m m}$ is an $(N-1) \times(N-1)$ matrix with elements $\Sigma_{x}(n, k)$, $k, n=\mathcal{V} \backslash\{m\}$. On the other hand, under the Gaussian assumption, the conditional mean of $\mathbf{y}_{m}$ on $\mathbf{y}_{n}$ is given by

$$
\mathrm{E}_{p\left(\mathbf{y}_{m} \mid \mathbf{y}_{n}\right)}\left\{\mathbf{y}_{m}\right\}=\left(\boldsymbol{\Sigma}_{m m}^{-1} \boldsymbol{\Sigma}_{1 m}\right)^{T} \mathbf{X}_{P, m}
$$

where

$$
\mathbf{X}_{P, m}=\left[\begin{array}{c}
\mathbf{y}_{0} \\
\mathbf{y}_{1} \\
\vdots \\
\mathbf{y}_{m-1} \\
\mathbf{y}_{m+1} \\
\vdots \\
\mathbf{y}_{N-1}
\end{array}\right]
$$

with

$$
\begin{equation*}
\mathbf{y}_{n}=\left[x_{1}(n), x_{2}(n), \ldots, x_{P}(n)\right] \tag{24.17}
\end{equation*}
$$

Therefore, to infer $\mathbf{Q}$, given the data observed on a graph, $\mathbf{x}_{1}, \mathbf{x}_{2}$, $\ldots, \mathbf{x}_{P}$, we can regress $x_{m}$ for each vertex, $m$, on the basis of (24.16) as follows,

$$
\begin{equation*}
x_{m}=\boldsymbol{\beta}_{m}^{T} \mathbf{X}_{P, m}+\boldsymbol{\epsilon}_{m}, \tag{24.18}
\end{equation*}
$$

where $\boldsymbol{\epsilon}_{m}$ is independent Gaussian noise.
Therefore, the problem of learning $\mathbf{Q}$ turns into the regression problem on $\boldsymbol{\beta}_{m}$, for each vertex, while non-zero elements in $\boldsymbol{\beta}_{m}$ also indicate the corresponding non-zero elements in $\mathbf{Q}$, namely, the edges in the graph.

The main advantage of regression-style methods is that the regressions for all vertices can be computed in parallel, which provides significantly relaxed computation when learning large graphs. However, additional attention should be paid to the symmetry of the learnt regression coefficients when dealing with an undirected graph, for example as in (19.11), more detail can be found in Meinshausen et al. (2006). The condition of coefficient sparsity could also be included, which leads to the LASSO formulation and solution to this problem, as in Section 19.1.

### 24.2.4 Factor Analysis Model

In Sections 24.1 and 24.2, the Gaussian distribution was assumed and on the basis of this distribution, most methods have been proposed to learn the graph edges in a recursive manner, i.e., by learning an edge per iteration. On the other hand, such methods can be regarded as a generative process via a basic Gaussian distribution, whereby the covariance or the precision matrix is nontrivially associated with the graph edges. It is thus natural to adopt more general and sophisticated models in graph learning.

One important model in probabilistic generative models is the factor analysis model, which forms the basis of many important tools, such as the probabilistic principal component analysis. Therefore, the observed data on a graph, $\mathbf{x}$, is assumed to be generated via a factor model that can be represented as

$$
\begin{equation*}
\mathbf{x}=\mathbf{U} \mathbf{v}+\epsilon, \tag{24.19}
\end{equation*}
$$

where $\mathbf{U}$ is a unitary matrix of the graph Laplacian eigenvectors, and $\mathbf{v}$ is a vector of latent variables (or factor loadings) which is Gaussian distributed with zero mean and a diagonal precision matrix corresponding to the graph Laplacian eigenvalues $\boldsymbol{\Lambda}$, that is,

$$
\mathbf{v} \sim \mathcal{N}\left(\mathbf{0}, \boldsymbol{\Lambda}^{-1}\right)
$$

where $\boldsymbol{\Lambda}^{-1}$ is the Moore-Penrose pseudoinverse of $\boldsymbol{\Lambda}$, while $\epsilon \sim \mathcal{N}\left(\mathbf{0}, \alpha^{2} \mathbf{I}\right)$ is also Gaussian distributed but independent of latent variables $\mathbf{v}$.

On the basis of this factor model, it is easy to obtain the distribution of the observations, $\mathbf{x}$, as

$$
\mathbf{x} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{U} \boldsymbol{\Lambda}^{-1} \mathbf{U}^{T}+\alpha^{2} \mathbf{I}\right)
$$

The term $\left(\mathbf{U} \boldsymbol{\Lambda}^{-1} \mathbf{U}^{T}\right)^{-1}=\mathbf{U}^{T} \boldsymbol{\Lambda} \mathbf{U}$ uniquely defines the Laplacian matrix, $\mathbf{L}$, of a graph. This allows us to infer the graph structure by learning $\mathbf{L}=\mathbf{U}^{T} \boldsymbol{\Lambda} \mathbf{U}$ from the factor model via maximizing the posterior distribution of $\mathbf{v}$ given $\mathbf{x}$, that is

$$
P(\mathbf{v} \mid \mathbf{x}) \propto P(\mathbf{x} \mid \mathbf{v}) P(\mathbf{v}) \propto e^{-\frac{(\mathbf{x}-\mathbf{U} \mathbf{v})^{T}(\mathbf{x}-\mathbf{U} \mathbf{v})}{\alpha^{2}}} e^{-\mathbf{v}^{T} \boldsymbol{\Lambda} \mathbf{v}}
$$

The log-likelihood form then follows as Dong et al. (2016)

$$
\begin{equation*}
\min _{\boldsymbol{\Lambda}, \mathbf{U}, \mathbf{v}}\|\mathbf{x}-\mathbf{U} \mathbf{v}\|^{2}+\rho \cdot \mathbf{v}^{T} \boldsymbol{\Lambda} \mathbf{v} \tag{24.20}
\end{equation*}
$$

where $\rho$ is a hyperparameter that balances between the influence of mean square error $\|\mathbf{x}-\mathbf{U v}\|^{2}$ and the positive definiteness constraint $\mathbf{v}^{T} \boldsymbol{\Lambda} \mathbf{v}$. Expression (24.20) can be further rewritten using the notation $\mathbf{y}=\mathbf{U v}$, as

$$
\begin{equation*}
\min _{\mathbf{L}, \mathbf{y}}\|\mathbf{x}-\mathbf{y}\|^{2}+\rho \cdot \mathbf{y}^{T} \mathbf{L} \mathbf{y} \tag{24.21}
\end{equation*}
$$

By inspection of (24.21) we see that the term $\mathbf{y}^{T} \mathbf{L} \mathbf{y}$ measures the smoothness of signal $\mathbf{y}$ on the graph; in other words, (24.21) minimises the distance between the observed samples and the generated signals, whilst imposing the smoothness on the generated signals, as discussed in Section 19.2. Other regularizations can also be imposed onto this model, such as that trace $(\mathbf{L})$ is equal to the dimension of the graph, in order to avoid a trivial all zero optimum and non-positive values in the off-diagonal elements of $\mathbf{L}$, and to learn a feasible graph
(Dong et al., 2016). Finally, (24.21) can be optimized in an alternative manner, as discussed in Section 19.2 and Algorithm 3, namely, by alternatively optimizing one of the two parameters ( $\mathbf{L}$ or $\mathbf{y}$ ) while fixing the other one.

Further improvements following the factor model of learning a smooth graph include the use of a more flexible smoothness prior when optimizing $\mathbf{L}$ in an alternative optimization, as various constrains on the L can lead to complicated optimization implementations (Kalofolias, 2016). This is achieved by rewriting the smoothness prior, $\mathbf{y}^{T} \mathbf{L y}$ in (24.21), as $\mathbf{y}^{T} \mathbf{L} \mathbf{y}=\frac{1}{2} \sum_{m, n} \mathbf{A}_{m n}(y(m)-y(n))^{2}$ so that the constrains can be explicitly imposed on the adjacency matrix $\mathbf{A}$, instead of on the Laplacian $\mathbf{L}$. It is also possible to learn graph by selecting the edges from atoms in a dictionary (called the incidence matrix) (Chepuri et al., 2017). Although this strategy can explicitly control the sparsity of the graph, it cannot optimise the edge weights (Mateos et al., 2019).
Example 100: Figures 8.1 and 8.2 show that different graph connections can exhibit different smoothness features, given the same observed samples, $\mathbf{x}$, shown in Figure 8.1(a). As also indicated in Figure 8.2, the observed sample retains the lowest frequency components for the graph in Figure 8.1(b) and the highest frequency components for the graph in Figure 8.1(c). This is reflected in a smaller smoothness value, $\mathbf{x}^{T} \mathbf{L} \mathbf{x}$, for the graph in Figure 8.1(b). This exemplifies that, given the observed graph samples, the smoothness prior is convenient for learning a graph.

### 24.3 Diffusion Models

It is important to notice that the smoothness that arises from the factor model is imposed in a global manner, which is effective in learning the main structure of a graph. However, promoting the global smoothness can also yield to the overestimation of the details within a graph. To resolve this issue, we can further assume that the observed graph signals are generated via a more complex and powerful model, such as the diffusion model. As shall be discussed in detail in Section 25.6.2, the polynomial filter is a typical choice for treating the diffusion from a

signal $\mathbf{x}$ at the vertices $\{0,1,2,3,4,5,6,7\}$
signal $\mathbf{x}$ on a path graph with smoothness $\mathbf{x}^{T} \mathbf{L} \mathbf{x}=0.94$
(b)

signal $\mathbf{x}$ on a path graph with smoothness $\mathbf{x}^{T} \mathbf{L x}=13.94 \quad$ (c)

Figure 8.1: Smoothness and graph learning. (a) The observed graph signal $\mathbf{x}=$ $[0.7,0.2,0.6,1.1-0.3,-1.1,1.3,-0.7]^{T}$, with (b)-(c) two types of possible path graph connections resulting in different smoothness values, $\mathbf{x}^{T} \mathbf{L x}$.
graph data analytics perspective. The benefits arising from learning a graph based on the diffusion model are mainly three-fold:

- Analytical and computational ease during learning.
- The (weak) stationarity is ensured in the generation system (Mateos et al., 2019).
- Ability to control the local smoothness in the model.

The diffusion model is given by (19.28)

$$
\begin{equation*}
\mathbf{x}=\sum_{m=0}^{M} h_{m} \mathbf{S}^{m} \mathbf{v}+\boldsymbol{\epsilon} \tag{24.22}
\end{equation*}
$$

where $\mathbf{v}$ is white Gaussian noise $\mathbf{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, while similar to the factor model in (24.19), $\boldsymbol{\epsilon} \sim \mathcal{N}\left(\mathbf{0}, \alpha^{2} \mathbf{I}\right)$. From (24.22), recall that $\mathbf{S}$ is the (symmetric) shift operator which can be chosen as e.g., the adjacency


Figure 8.2: Graph signal spectrum values which correspond to the two types of graph connections in Figure 8.1. The top panel corresponds to Figure 8.1(b) and the bottom panel to Figure 8.1(c). The energy is calculated via $\mathbf{x}^{T} \mathbf{L} \mathbf{x}$, where small values indicate a smooth graph.
matrix, or the Laplacian matrix, to name but a few. Here, we continue to use

$$
\mathbf{S}=\mathbf{L}
$$

as in Section 19.5. Furthermore, expression (24.22) can be compactly written in the form of (19.28), as

$$
\begin{equation*}
\mathbf{x}=\sum_{m=0}^{M} h_{m} \mathbf{L}^{m} \mathbf{v} \tag{24.23}
\end{equation*}
$$

where $\mathbf{L}^{0}=\mathbf{I}$ and $h_{0}=\alpha^{2}$ retain the same statistics as those in (24.22).
On the basis of (24.23), the covariance of $\mathbf{x}$ can be calculated as

$$
\begin{align*}
\boldsymbol{\Sigma} & =\mathrm{E}\left\{\mathbf{x} \mathbf{x}^{T}\right\}=\left(\sum_{m=0}^{M} h_{m} \mathbf{L}^{m}\right) \mathrm{E}\left\{\mathbf{v} \mathbf{v}^{T}\right\}\left(\sum_{m=0}^{M} h_{m} \mathbf{L}^{m}\right)^{T} \\
& =\sum_{m=0}^{M} h_{m} \mathbf{L}^{m}\left(\sum_{m=0}^{M} h_{m} \mathbf{L}^{m}\right)^{T}=\mathbf{U}^{T}\left(\sum_{m=0}^{M} h_{m} \boldsymbol{\Lambda}^{m}\right)^{2} \mathbf{U} \tag{24.24}
\end{align*}
$$

where we have used the eigendecomposition $\mathbf{L}=\mathbf{U}^{T} \boldsymbol{\Lambda} \mathbf{U}$.
Eigenvector estimation. From (24.24), we can see that the eigenvectors of $\mathbf{L}$ are the same as those of the covariance matrix of $\mathbf{x}$. This means, in a straightforward way, that we can infer the eigenvectors of $\mathbf{L}$ from the empirical covariance of the observed data, $\mathbf{x}_{1}, \mathbf{x}_{2}, \ldots, \mathbf{x}_{P}$.

Eigenvalue estimation. After obtaining the eigenvectors, the remaining task is to estimate the eigenvalues of $\mathbf{L}$. Without any additional constraints, it is obvious that arbitrary values can be chosen as the eigenvalues of $\mathbf{L}$, because we can always find a corresponding set of $h_{0}, h_{1}, \ldots, h_{M}$ that satisfies (24.24). Thus, to achieve a unique solution, we need to employ some prior on the function $f(\cdot)$ (Segarra et al., 2017), to arrive at

$$
\begin{equation*}
\min _{\mathbf{L}, \boldsymbol{\Lambda}} f(\mathbf{L}), \quad \text { subject to } \mathbf{L}=\mathbf{U}^{T} \boldsymbol{\Lambda} \mathbf{U} \tag{24.25}
\end{equation*}
$$

For example, when $f(\mathbf{L})=\|\mathbf{L}\|_{0}$, the objective function minimises the number of edges, whereas $f(\mathbf{L})=\|\mathbf{L}\|_{2}$ minimises the energy of graph edges. The number of edges can also be minimized using convex relation of $f(\mathbf{L})=\|\mathbf{L}\|_{0}$ in the form $f(\mathbf{L})=\|\mathbf{L}\|_{1}$, as explained in Part II of this monograph and Section 19.5.

Equation (24.23) assumes that the diffusion process starts from the same initial status, that of white Gaussian noise. An enhanced diffusion model has been proposed in Thanou et al. (2017) by assuming that the signals are generated from multiple heat diffusion processes

$$
\begin{equation*}
\mathbf{x}=\sum_{m=0}^{M} e^{-h_{m} \mathbf{L}} \mathbf{v}_{m} \tag{24.26}
\end{equation*}
$$

Here, $\mathbf{v}_{m}$ represents the initial state that can also be optimized, and $h_{m}$ controls the diffusion time (depth). This means that with a small $h_{m}$, the $k$ th column of $e^{-h_{m} \mathbf{L}}$ is localized at the $k$ th vertex. This model can be solved via a dictionary-learning solver by regarding $\left[e^{-h_{0} \mathbf{L}}, e^{-h_{1} \mathbf{L}}, \ldots, e^{-h_{M} \mathbf{L}}\right]$ as the dictionary $\mathbf{D}$ and $\left[\mathbf{v}_{0}, \mathbf{v}_{1}, \ldots, \mathbf{v}_{M}\right]$ as coefficients $\mathbf{V}$. The objective function can now be formulated as

$$
\begin{aligned}
\min _{\mathbf{L}, \mathbf{X}, h_{m}} & \|\mathbf{X}-\mathbf{D V}\|_{F}^{2}+\operatorname{reg}(\mathbf{V})+\operatorname{reg}(\mathbf{L}), \\
\text { subject to } & \left\{h_{m}\right\}_{m=0}^{M} \leq 0
\end{aligned}
$$

where $\operatorname{reg}(\cdot)$ denotes a certain regularization; for more detail, we refer to Thanou et al. (2017).

## 25

## Graph Neural Networks

An emerging area which considers graphs in conjunction with neural networks is that of graph neural networks (GNNs). The underpinning idea is to combine the universal approximation property of neural networks and the ability of graphs to capture higher-order information in a physically meaningful way, thus equipping GNNs with enhanced expressive and modelling power. Work in this direction has been facilitated by steadily growing computational power and the ever increasing amount of available data. The beginning of graph neural networks (GNNs) can be traced back to basic network structures (Gori et al., 2005; Micheli, 2009; Scarselli et al., 2008) one decade ago, while recent developments have been centered around graph convolutional networks (GCNs). The GCNs benefit from their intrinsic graph structure, which allows to account for complex implicit coupling among data and information aggregation when processing (or filtering) data at each vertex. This is particularly desirable in deep neural network (DNN) techniques, where the involvement of graphs provides a balance between the "black-box" (but powerful) DNNs and the purely mathematical tools such as manifold optimization and manifold learning. Benefiting from prior information embedded into a graph structure, GCNs are capable of not only handling irregular data
but also of alleviating the "black-box" nature of DNNs, thus helping resolve two major open issues with current DNNs.

Recent literature on GCNs (Wu et al., 2019; Zhou et al., 2018) typically considers the learning aspects, while highlighting two key properties of CNNs: (i) stationarity (via shift invariance of convolution operations) and (ii) compositionality (via downsampling or pooling operations). Taking a sightly different viewpoint, we start from the graph itself and proceed to illuminate that certain types of graphs correspond to major trends in GCNs. We also outline the advantages of treating GCNs in this way, such as the possibility to open avenues for the design of novel types of GCNs.

We introduce GNNs from the perspective of a diffusion process, because of the role of diffusion which underpins signal propagation in graphs. With the ability of graphs to provide intrinsic structures when aggregating information, this allows us to describe recurrent GNNs as a kind of diffusion processes of task-oriented models; all in all, an intuitive way to reveal the underlying mechanisms of GNNs. For example, a standard GNN "feed-forwards" (aggregates) input information layer-bylayer towards the output, calculates deviation from the ground-truth, and then back-propagates to improve the aggregation strategy (weight update). Such information flow (or message passing) is also found in the diffusion process, for example, in temperature transfer heat shown in Example 71. Therefore, the diffusion process can be rephrased as a "language" of neural nets, even for the basic gradient descent updating process. A more complex version is addressed in Section 25.3, in the form of the diffusion process with external sources, which serves to establish a link with the well-know label propagation method. We show that label propagation can be basically regarded as a one-layer GNN, which despite not having weights to be optimized is still powerful enough in semi-supervised learning. This is shown to naturally extend to multiple layers of GNNs, whereby the stacked layers perform message passing (also similar to the diffusion process). We also employ the concept of system on a graph to explain spectral GCNs, while spatial GCNs are shown to admit interpretation as a relaxation of spectral GCNs to the localization in graphs.

### 25.1 Basic Graph Elements Related to GCNs

The following properties of graphs are helpful in understanding the GCNs (for more detail we refer to Section 2.1 of Part I).

- Property 1: When $\mathbf{A}$ is binary, i.e., it represents the connection of vertices (adjacency matrix), the number of walks of a length $k$, between two vertices $m$ and $n$, is equivalent to the value of the corresponding element $a_{m n}$ of the $k$ th power of $\mathbf{A}$, that is, of $\mathbf{A}^{k}$. The number of walks between the vertices $m$ and $n$, that are of length not higher than $k$, is given by the corresponding element of $\mathbf{B}_{k}$, where $\mathbf{B}_{k}=\mathbf{A}+\mathbf{A}^{2}+\cdots+\mathbf{A}^{k}$. Matrix $\mathbf{B}_{k}$ gives the $k$-neighborhood of a vertex, which is a set of vertices that are reachable from this vertex through walks within $k$ steps.
- Property 2: For any signal on graph, $\mathbf{x}$, the quadratic form of the Laplacian, $\mathbf{x}^{T} \mathbf{L x}$, is of the form

$$
\begin{equation*}
\mathbf{x}^{T} \mathbf{L} \mathbf{x}=\frac{1}{2} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} A_{m n}(x(m)-x(n))^{2} \tag{25.1}
\end{equation*}
$$

This indicates that: (1) the Laplacian matrix, $\mathbf{L}=\mathbf{D}-\mathbf{A}$, is positive semi-definite because $A_{m n}(x(m)-x(n))^{2} \geq 0$; (2) the smoothness of graph signal, $\mathbf{x}$, can be quantified via $\mathbf{x}^{T} \mathbf{L x}$, which ensures that the quadratic form $\mathbf{x}^{T} \mathbf{L} \mathbf{x}$ is equivalent to the Dirichlet energy of $\mathbf{x}$, which has been widely used in probabilistic graph models.

Remark 100: The smoothness of a graph signal, $\mathbf{x}$, implies that the signal value would not change much from one vertex to another within the neighborhood of vertex $n$ (assessed by $(x(m)-x(n))^{2}$ ). However, signal values are allowed to change significantly when the two vertices are not connected (indicated by zero values of $A_{m n}$ ). Therefore, the minimization on $\mathbf{x}^{T} \mathbf{L x}$ finds the smoothest signal $\mathbf{x}$ on the graph.

Note that the absolute maximum smoothness (minimum of the smoothness index) is achieved for a signal which is constant over all vertices; such signal is equal to the eigenvector corresponding to the
smallest eigenvalue, $\lambda_{0}=0$, of the graph Laplacian, $\mathbf{L}$ (owing to the Rayleigh quotient). More importantly, this yields $\mathbf{1}^{T} \mathbf{L} \mathbf{1}=\mathbf{1}^{T}(\mathbf{D}-\mathbf{A}) \mathbf{1}=$ 0 , that is, the smallest eigenvalue is $\lambda_{0}=0$ with the corresponding normalized eigenvector $\mathbf{x}=\mathbf{u}_{0}=\mathbf{1} / \sqrt{N}$, where $\mathbf{1}$ denotes an $N$ dimensional vector of unities.

## Connection to the Laplacian Operator in Function Analysis

One way of understanding the role of the Laplacian matrix in measuring signal smoothness is via its continuous time counterpart - the Laplacian operator in functional analysis. The Laplacian operator over a function $f(\vec{r})$ in the Euclidean space is defined as

$$
\operatorname{div}(\operatorname{grad}(f(\vec{r})))=\nabla(\nabla f(\vec{r}))=\Delta f(\vec{r})
$$

where $\operatorname{grad}(\cdot)$ denotes the gradient operator and $\operatorname{div}(\cdot)$ is the divergence operator. For example, in Cartesian coordinates of two dimensions, $\vec{r}=(x, y)$, we have

$$
\begin{equation*}
\Delta f(x, y)=\frac{\partial^{2} f(x, y)}{\partial x^{2}}+\frac{\partial^{2} f(x, y)}{\partial y^{2}} \tag{25.2}
\end{equation*}
$$

Similarly, we can also define the Laplacian operator on the graph, whereby the different (and difficult) aspect is the differential operator. Namely, while as in the discrete signal space, the difference operation is defined as $\nabla f(x)=f(x+1)-f(x)$, which calculates the difference between $f(x+1)$ and $f(x)$, the differential on a graph is defined for each edge, that is

$$
\nabla f_{m n}=f(m)-f(n)
$$

This means that, in general, the differential on a graph allows for a different number of directions at each point (vertex), while for the path graph, the differential $\nabla f_{m n}$ naturally simplifies into the standard differential in the Euclidean space.

Example 101: To illustrate the role of the graph Laplacian, consider a graph in Figure 8.1, which is a simplified version of Figure 1(a) of Part I. Its adjacency matrix and the corresponding graph Laplacian


Figure 8.1: A simplified version of the default graph considered throughout this work, as in Figure 1.1(a) in Part I.
matrix are given by

$$
\mathbf{A}=\left[\begin{array}{lllll}
0 & 1 & 1 & 1 & 0  \tag{25.3}\\
1 & 0 & 1 & 0 & 1 \\
1 & 1 & 0 & 1 & 1 \\
1 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & 0
\end{array}\right], \quad \mathbf{L}=\left[\begin{array}{rrrrr}
3 & -1 & -1 & -1 & 0 \\
-1 & 3 & -1 & 0 & -1 \\
-1 & -1 & 4 & -1 & -1 \\
-1 & 0 & -1 & 2 & 0 \\
0 & -1 & -1 & 0 & 2
\end{array}\right]
$$

To calculate the gradient, $\operatorname{grad}(\mathbf{f})$, of a signal, $\mathbf{f}$, on this graph

$$
\mathbf{f}=\left[\begin{array}{l}
f(0)  \tag{25.4}\\
f(1) \\
f(2) \\
f(3) \\
f(4)
\end{array}\right],
$$

which represents the differential at each edge, we introduce the so called incidence matrix, $\mathbf{K}$, given by

$$
\mathbf{K}=\begin{array}{r} 
 \tag{25.5}\\
0 \\
1 \\
2 \\
3 \\
4
\end{array}\left[\begin{array}{rrrrrrr}
e_{0} & e_{1} & e_{2} & e_{3} & e_{4} & e_{5} & e_{6} \\
1 & 0 & 1 & 0 & 1 & 0 & 0 \\
-1 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & -1 & -1 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & -1 & -1 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 & -1
\end{array}\right] .
$$

The gradient on the graph now becomes

$$
\operatorname{grad}(\mathbf{f})=\mathbf{K}^{T} \mathbf{f}=\begin{gather*}
\nabla f  \tag{25.6}\\
e_{0} \\
e_{1} \\
e_{2} \\
e_{3} \\
e_{4}
\end{gather*}\left[\begin{array}{c}
f(0)-f(1) \\
f(1)-f(4) \\
e_{5} \\
e_{6}
\end{array}\right]
$$

Due to the adjoint property of the divergence operator with regard to inner products, the graph Laplacian for this graph becomes

$$
\begin{align*}
\Delta \mathbf{f} & =\operatorname{div}(\operatorname{grad}(\mathbf{f}))=\mathbf{K}\left(\mathbf{K}^{T} \mathbf{f}\right)=\left(\mathbf{K} \mathbf{K}^{T}\right) \mathbf{f} \\
& =\left[\begin{array}{rrrrr}
3 & -1 & -1 & -1 & 0 \\
-1 & 3 & -1 & 0 & -1 \\
-1 & -1 & 4 & -1 & -1 \\
-1 & 0 & -1 & 2 & 0 \\
0 & -1 & -1 & 0 & 2
\end{array}\right]\left[\begin{array}{l}
f\left(v_{0}\right) \\
f\left(v_{1}\right) \\
f\left(v_{2}\right) \\
f\left(v_{3}\right) \\
f\left(v_{4}\right)
\end{array}\right] . \tag{25.7}
\end{align*}
$$

It is now obvious that $\mathbf{K K}^{T}$ is equivalent to the graph Laplacian matrix $\mathbf{L}$ in (25.3).

Remark 101: The analysis in (25.4)-(25.7) exemplifies that a graph effectively defines local coordinates with a prior or learnt linkage information, and thus in some sense it can then be considered as a discrete approximation to a manifold. This insight is particularly useful in the design and interpretation of GNNs.

### 25.2 Gradient Descent as a Diffusion Process

Consider a physical diffusion process, and in particular the Newton's law of cooling, which states that the energy (or heat) loss rate is proportional to the temperature difference between the body (node) and its surrounding environment. The diffusion process can then be understood as an iterative process that converges toward the state of minimum energy, given by $\mathbf{x}^{T} \mathbf{L} \mathbf{x}$, from any initial condition. Since the
gradient of energy is

$$
\operatorname{grad}\left(\mathbf{x}^{T} \mathbf{L} \mathbf{x}\right)=\frac{\partial\left(\mathbf{x}^{T} \mathbf{L} \mathbf{x}\right)}{\partial \mathbf{x}^{T}}=2 \mathbf{L} \mathbf{x}
$$

the iterative discrete-time solution for the diffusion process, at an instant $(t+1)$, is given by

$$
\begin{equation*}
\mathbf{x}_{t+1}-\mathbf{x}_{t}=-\alpha \mathbf{L} \mathbf{x}_{t}, \tag{25.8}
\end{equation*}
$$

or

$$
\mathbf{x}_{t+1}=\mathbf{x}_{t}-\alpha \mathbf{L} \mathbf{x}_{t}
$$

where $\alpha$ is a constant. This solution to the diffusion process can also be formulated as

$$
\begin{equation*}
\nabla x(n) \approx-\alpha \sum_{m \in \mathcal{V}_{n}}(x(n)-x(m)), \tag{25.9}
\end{equation*}
$$

where $\mathcal{V}_{n}$ is the set of vertices within the neighborhood-one of the vertex $n$, while $\sum_{m \in \mathcal{V}_{n}}(x(n)-x(m))$ denotes an aggregate temperature difference between the vertex $n$ and its surrounding vertices.

Remark 102: Equation (25.8) models the change in temperature along time, starting from an initial state $\mathbf{x}_{0}$. In the following, we will show that this provides an ideal means for designing recurrent GNNs. For more detail on Recurrent Neural Networks (RNN), we refer to Mandic and Chambers (2001) and Mandic and Goh (2009).

The quadratic term, $\mathbf{x}^{T} \mathbf{L} \mathbf{x}$, is frequently used in data analytics on graphs, for example for estimating smoothness. The gradient of $\mathbf{x}^{T} \mathbf{L} \mathbf{x}$ is $\partial\left(\mathbf{x}^{T} \mathbf{L x}\right) / \partial \mathbf{x}=2 \mathbf{L x}$, so that the diffusion process in (25.8) will find the exact minimum of this quadratic form. As mentioned in Section 25.1, the minimum of $\mathbf{x}^{T} \mathbf{L} \mathbf{x}$ is reached for a constant eigenvector with all elements equal to 1 , which indicates that such a diffusion process, when left without any external sources, will eventually settle to the same temperatures for all vertices.

### 25.3 Label Propagation as a Diffusion Process with External Sources

The stable state (equilibrium) of a diffusion process without external sources cannot give us any useful information because in this case the
data at all the vertices have the same value (i.e., the lowest entropy on the graph). In Physics, we can alter the stable state by adding some constant external sources, which ensures that the final temperatures are not all the same but exhibit some fluctuations governed by their inherent relationships. This is also the basic idea behind many graph machine learning approaches, especially in semi-supervised learning tasks, such as the label propagation given in Algorithm 5.

```
Algorithm 5. Label Propagation
    procedure Initialization
            Initialise a graph by treating each data sample separately, as a single vertex;
            Connect all vertices in the graph, whereby edge weights are defined by some
    similarity measure;
            Assign the labels from the labeled samples to the corresponding vertices;
            Randomly assign values to the unlabeled vertices.
            while Not converged: do
            Propagate from the labeled to the unlabeled vertices: \(\mathbf{x} \leftarrow \mathbf{L x}\).
                                    \(\triangleright\) Diffusion process
            Re-assign the original labels to the labeled vertices, \(\mathbf{x}_{L}\).
                                    \(\triangleright\) Keep external resources
            end while
            return x
    end procedure
```

The final state of this modified diffusion process can be easily shown to be Zhu (2005),

$$
\begin{equation*}
\mathbf{x}_{U}=\left(\mathbf{I}-\mathbf{L}_{U U}\right)^{-1} \mathbf{L}_{U L} \mathbf{x}_{L}, \tag{25.10}
\end{equation*}
$$

where

$$
\mathbf{L}=\left[\begin{array}{ll}
\mathbf{L}_{L L} & \mathbf{L}_{L U}  \tag{25.11}\\
\mathbf{L}_{U L} & \mathbf{L}_{U U}
\end{array}\right],
$$

and the subscripts $U$ and $L$ designate respectively the unlabelled and labelled sets. Note that for a graph shift, instead of $\mathbf{L}$ we may also use $\mathbf{A}$.

The final stable state will now no longer have the same signal values for all vertices (at least $\mathbf{x}_{U} \neq \mathbf{x}_{L}$ ). This is due to the "external constant" sources of the labelled samples (Line 8 in Algorithm 5), which ensures that the diffusion process results in stable states with signals which are different for each vertex; this also gives the predicted labels for
unlabelled signal samples (or vertices) in the inner structures of the graph.

Example 102: To provide a simple illustration of label propagation in handwritten digit recognition, we used three sets of handwritten digits, 1, 5 and 9, each with ten images from the MNIST database (LeCun et al., 1998). We adopted the structural similarity (SSIM) metric (Wang et al., 2004) to measure the similarity between images and to construct a graph accordingly, as shown in Figure 8.2(a). In this example, we chose only two labels for each digit type to act as the external sources in the diffusion process. It needs to be pointed out that without the external sources, the final state would settle to a constant vector; this does not provide any informative predictions.

The total of six given labels are annotated in Figure 8.2(c) and the predicted labels are shown in Figure 8.2(d). By comparing with the ground truth shown in Figure 8.2 (b), we can see that label propagation achieved adequate prediction accuracy, given a correctly constructed graph. The level of certainty in the prediction is designated by the node color in Figure $8.2(\mathrm{~d})$, with the provided labels (ground truth) in the red color, and the nodes on the intersections of two types of digits in green colors, indicating the large uncertainty of predictions in these vertices. Therefore, when regarding label propagation as a diffusion process, the temperature can be interpreted as the level of certainty, whereby the external sources (the six given labels) have the highest temperature (designated with the red color) and the heat diffusion performs "certainty propagation". Vertices surrounding the external sources, as a consequence, would retain relatively high temperatures (we are much more sure about the predictions on these nodes).

### 25.4 GNNs of a Recurrent Style

Now that we have shown that different diffusion models can be utilized to aggregate information across graph vertices, we may employ diffusion to design neural networks on graphs, as neural networks also rely upon information aggregation. One such frequently used recurrent GNN was


Figure 8.2: Principle of label propagation. We used two labelled images out of ten available images per digit from the MNIST dataset. Three sets of digits (1, 5 and 9) are chosen and each set contains ten images. (a) The resulting graph constructed via the SSIM metric, where two images (nodes) are connected when their SSIM is larger than a threshold (set to 0.35). (b) The ground truth labels for the 30 images considered. (c) Only two labels are provided for each set of images, as indicated by the red color. (d) Predicted labels from the given six (i.e., $2 \times 3$ ) labels via label propagation over the graph Laplacian matrix $\mathbf{L}$. The color bar designates the certainty of predictions, namely, the red color denotes an almost sure prediction with probability approaching 1 and green color poor prediction.
proposed by Scarselli et al. (2008), which aggregates information as

$$
\begin{gather*}
x_{t+1}(n)=\sum_{m \in \mathcal{V}_{n}} f\left(x_{t}(n), q(n), x_{t}(m), q(m)\right)  \tag{25.12}\\
o(n)=\phi(x(n), q(n)) \tag{25.13}
\end{gather*}
$$

where $x_{t}(x)$ is the signal value at the $n$th vertex at a time instant $t$, $\mathcal{V}_{n}$ denotes the neighborhood-one of the vertex $n, q(n)$ is a pre-defined feature of $x(n), q(m)$ represents the pre-defined features at the neighbor vertices, and $o(n)$ is the output at the $n$th vertex. The operators $f(\cdot)$ and $\phi(\cdot)$ can be chosen to form neural networks so that they can be learnt via back-propagation; in other words, the diffusion style model can be learnt from data samples. In a particular case when $q(n)$ and $q(m)$ are omitted, and

$$
f(x(n), q(n), x(m), q(m))=(x(n)-x(m))
$$

Equation (25.12) turns into the original diffusion process in (25.9).
The aggregation function in (25.12) motivates much recent work on GNNs and spatial GCNs, however, this variant of recurrent GNNs needs to undergo the diffusion process until convergence, for every iteration of back-propagation. Moreover, the mapping $f(\cdot)$ in (25.12) needs to be carefully designed to be a contraction mapping to ensure convergence (Mandic, 2007). More recent efforts to improve this model include the gated recurrent GNN (Li et al., 2015) that employs a gated unit as $f(\cdot)$ to ensure convergence within a fixed number of steps, while stochastic steady-state recurrent GNNs (Dai et al., 2018) perform update in (25.12) in a stochastic manner.

Another approach which incorporates both spatial convolutions and temporal diffusions, is the diffusion convolution neural network (DCNN) (Atwood and Towsley, 2016), which can be formulated as

$$
\begin{equation*}
\mathbf{h}^{l}=\phi\left(\mathbf{w}^{l} \odot \mathbf{L}^{l} \mathbf{x}\right), \tag{25.14}
\end{equation*}
$$

where $\mathbf{h}^{l}$ is the hidden state of the $l$ th layer, $\mathbf{w}^{l}$ are convolution kernels that are to be learnt, and $\mathbf{L}^{l}$ is the power series up to $l$ of a certain probability transition matrix (in this case graph Laplacian $\mathbf{L}$ ) which is similar to Line 7 in Algorithm 5; recall that $\odot$ denotes the element-wise product and $\phi$ the activation function. It should be pointed out that the model in (25.14) implies that $\mathbf{h}^{l}$ does not depend on the previous layer (state) $\mathbf{h}^{l-1}$, and that the dimensions of each layer need to be the same; this limits the number of degrees of freedom in the design. The overall output of this GCN is a composition of all layers $\left\{\mathbf{h}^{l}\right\}_{l=1}^{L}$, so
that (25.14) can be interpreted as a set of diffusion processes of different depths (by regarding $l$ as time instant $t$ ).

Another way of understanding the operation in (25.14) is that each diffusion step, $\mathbf{L}^{l} \mathbf{x}$, aggregates (to a certain degree) the heat (or general features and labels). This is a kind of message passing and aggregation that equips the network with the ability to extract statically salient features, which belong to spatial GCNs, introduced below.

Remark 103: Almost all approaches to recurrent GNNs aim to find an efficient and stable diffusion strategy to propagate and aggregate the labels or information at each vertex, so as to facilitate reliable and robust predictions at the final stable stage of the GNNs.

### 25.5 Spatial GCNs via Localization of Graphs

It is important to note that while CNNs have been an enabling technology for modern machine learning applications, they also suffer from the limitations inherited from the underlying assumption of a regular time/space grid sampling, such as in images and videos. The effort to extend CNNs to GCNs that are able to operate on data acquired on irregular domains therefore needs to accommodate both the convolution (to learn local stationary features) and the pooling (to compose multiscale patterns) operators. Our main focus is on ways to accommodate the data on irregular domains, while the generalization of pooling is naturally related to the downsampling on the graph (see Part II and Bacciu and Di Sotto, 2019; Ioannidis et al., 2019a; Sakiyama et al., 2019; Tanaka and Eldar, 2019; Zhang et al., 2019a). The key difficulty in defining the convolution on a graph is the absence of a rigorous translation (shift) operator. To this end, the basic idea behind spatial GCNs is the information aggregation principle, which is very similar (sometimes even intertwined with) to the diffusion GNNs in Section 25.4. Instead of waiting for a stable state (along the time instants) of recurrent GNNs, spatial GCNs directly aggregate information by the stacked layers, which is also called message passing. The initial work in this area was by Micheli (2009), the so called neural network for graphs (NN4G). A more general model is the message passing neural networks
(MPNNs) (Gilmer et al., 2017), which is given by

$$
\begin{equation*}
x^{l+1}(n)=\phi\left(x^{l}(n), \sum_{m \in \mathcal{V}_{n}} f\left(x^{l}(n), x^{l}(m), e_{n m}\right)\right), \tag{25.15}
\end{equation*}
$$

where $x^{l}(n)$ represents the data value at the $n$th vertex of the $l$ th layer, $e_{n m}$ denotes the edge between the $n$th and the $m$ th vertex, while $f(\cdot)$ is the message passing function and $\phi(\cdot)$ denotes the activation (or vertex updating) function. The model in (25.15) caters for many GCNs, such as those in Micheli (2009) and Kipf and Welling (2016a) which all have different forms of functions $f(\cdot)$ and $\phi(\cdot)$. This model also involves the basic steps for processing graph signals in the spatial domain, i.e., by aggregating the previous messages and passing to the next layer. Bacciu et al. further extended this idea to a probabilistic framework Bacciu et al. (2018), which enables a probabilistic explanation on each state of each layer.

Furthermore, instead of looking for all neighbors of the central vertex in (25.14), the GraphSAGE approach proposes to sample several neighbors around every vertex (Hamilton et al., 2017), as follows

$$
\begin{equation*}
x^{l+1}(n)=\phi\left(\mathbf{W}^{l} \cdot \operatorname{concat}\left\{x^{l}(n), f\left\{x^{l}(m), m \in \widetilde{\mathcal{V}}_{n}\right\}\right\}\right), \tag{25.16}
\end{equation*}
$$

where concat $\{\cdot, \cdot\}$ denotes the concatenation and $f\{\cdot\}$ the aggregation function, $\mathbf{W}^{l}$ is the matrix of learnable parameters, and $\widetilde{\mathcal{V}}_{n}$ denotes a randomly chosen neighbor of the $n$th vertex. This strategy allows for a mini-batch operation on graphs, which is extremely useful for large graphs.

A further possible improvement is to learn the weights while choosing the neighboring vertices; this includes the graph attention network (GAT) (Veličković et al., 2017), and the mixture model network (MoNet) (Monti et al., 2017). Within GATs, an attention weight, $\alpha_{n, m}$, is added to the parameters in (25.15), which allows us to assign different importance levels to vertices, even within the same neighborhood. The attention weight can be further learnt from an additional convolution sub-network, as proposed in Zhang et al. (2018a). On the other hand, the MoNet defines the weights of neighboring edges as a consequence of local coordinates, which has an intrinsic link with the manifolds. More specifically, it defines the importance of the edge connecting the $n$th
and $m$ th vertex as a probability, $p$, over some local coordinates, $\mathbf{u}(m, n)$, which reflects the difference (or distance) between the $n$th and $m$ th vertex. Then, the $n$th vertex can be aggregated via a specially defined convolution, given by

$$
\begin{equation*}
(\mathbf{x} * \mathbf{g})(n)=\sum_{j=1}^{J} g_{j} \sum_{m \in \mathcal{V}_{n}} p(\mathbf{u}(m, n)) x(m) \tag{25.17}
\end{equation*}
$$

where $g_{j}$ is the $j$ th index (element) of the convolution kernel, $\mathbf{g}$. In Monti et al. (2017), the probability, $p(\mathbf{u}(m, n))$, was chosen as a Gaussian mixture model, which has $J$ clusters to cater for the size of convolution kernel. It has also been shown that the framework of (25.17) accounts for various geometric deep neural networks, through a choice of different local coordinates and weight functions.

### 25.6 Spectral GCNs via Graph Fourier Transform

As shown in Section 25.5, message passing via the convolution operation plays a crucial role in spatial GCNs. Here, we focus on the methods that operate in a transfer domain and benefit from the mathematically well-defined convolution in the graph spectral domain to yield a class of spectral GCNs.

### 25.6.1 Graph Fourier Transform

Due to the positive semi-definiteness of $\mathbf{L}$, there are $N$ (the number of vertices) real-valued eigenvalues $\left(\lambda_{0}=0 \leq \lambda_{1}<\lambda_{2}<\cdots<\right.$ $\left.\lambda_{N-1}\right)$, which correspond to $N$ distinct orthogonal eigenvectors ( $\left[\mathbf{u}_{0}, \mathbf{u}_{1}\right.$, $\left.\ldots, \mathbf{u}_{N-1}\right]$ ). As mentioned in Section 25.1, the quadratic form, $\mathbf{x}^{T} \mathbf{L x}$, measures the smoothness of the data, $\mathbf{x}$, on a graph. Further, when $\mathbf{x}$ represents one of the eigenvectors, $\mathbf{u}_{j}$, the term $\mathbf{x}^{T} \mathbf{L} \mathbf{x}$ then measures the smoothness of the eigenvectors, $\mathbf{u}_{j}^{T} \mathbf{L} \mathbf{u}_{j}=\lambda_{j}$. The matrix of eigenvectors, $\mathbf{U}=\left[\mathbf{u}_{0}, \mathbf{u}_{1}, \ldots, \mathbf{u}_{N}\right]$, represents an orthogonal transform basis, which is similar to principal component analysis (PCA), while benefiting from a physically more important and beneficial property in practice because the graph Laplacian bases indicate the smoothness of eigenvectors.

Remark 104: Through multiplication of the data, $\mathbf{x}$, by the eigenmatrix, $\mathbf{U x}$, the original data $\mathbf{x}$ are decomposed into different constituent components, which vary from the most smooth to the most non-smooth. This is exactly the principle of the Fourier transform, which transforms a signal to different frequency components (bases). In this case, $\lambda_{j}$ has the physical meaning of (squared) frequency, as shown in Section 3.5.2 of Part II. In particular, when the graph structure is a path graph, the original Fourier transform is obtained.

Based on the graph Fourier transform, covered in detail in Part II of this monograph, we can now define the graph convolution operator which states that the convolution in the spatial (vertex) domain is equal to the multiplication in the spectral domain. This bypasses the requirement for translation (or shift operator) to define convolution in the vertex domain, whilst maintaining the concept of "convolution" over graph signals. In this way, the graph convolution is given by

$$
\begin{equation*}
\mathbf{U}^{T}(\mathbf{x} * \mathbf{g})=\left(\mathbf{U}^{T} \mathbf{x}\right) \odot\left(\mathbf{U}^{T} \mathbf{g}\right) \tag{25.18}
\end{equation*}
$$

where $\mathbf{x}$ and $\mathbf{g}$ are two vectors whose elements are the data values at vertices $n \in \mathcal{V}$. Recall that $\mathbf{U}$ in (25.18) is the Fourier basis composed by the eigenvectors of $\mathbf{L}$ and $\odot$ denotes the Hadamard (element-wise) product. It is worth mentioning that the matrix $\mathbf{U}$ is a graph counterpart of the frequency shift operator in the continuous time Fourier transform.

### 25.6.2 Graph Spectral Filtering as Multiple Diffusion Processes

Upon inspection of the diffusion process of the cooling law in Section 25.3, we can see that it actually aggregates the data values at the connected vertices to process the current vertex. Consider now a polynomial filter of the diffusion process, given by

$$
\begin{equation*}
\mathbf{x} \leftarrow \mathbf{B}_{k} \mathbf{x}=\left(\mathbf{A}+\mathbf{A}^{2}+\cdots+\mathbf{A}^{k}\right) \mathbf{x}, \tag{25.19}
\end{equation*}
$$

where $k$ neighboring vertex data values are aggregated to produce the current vertex data sample, according to the Property 1 of Section 25.1. It can be proved that the $k$-neighboring property also holds when $\mathbf{B}_{k}$ is given by the powers of the Laplacian, $\mathbf{L}^{k}$ (Lemma 5.4, Hammond et al., 2011b), as we are still using the $k$-neighbor information when
aggregating, that is

$$
\begin{equation*}
\mathbf{x} \leftarrow\left(\mathbf{L}+\mathbf{L}^{2}+\cdots+\mathbf{L}^{k}\right) \mathbf{x} \tag{25.20}
\end{equation*}
$$

Upon rewriting (25.20) in the graph spectral domain, we have

$$
\begin{equation*}
\mathbf{x} \leftarrow \mathbf{U}\left(\boldsymbol{\Lambda}+\boldsymbol{\Lambda}^{2}+\cdots+\boldsymbol{\Lambda}^{k}\right) \mathbf{U}^{T} \mathbf{x} \tag{25.21}
\end{equation*}
$$

or equivalently

$$
\begin{equation*}
\mathbf{X} \leftarrow\left(\boldsymbol{\Lambda}+\boldsymbol{\Lambda}^{2}+\cdots+\boldsymbol{\Lambda}^{k}\right) \mathbf{X}, \tag{25.22}
\end{equation*}
$$

where $\mathbf{X}$ is the spectral representation of $\mathbf{x}$, through $\mathbf{X}=\mathbf{U}^{T} \mathbf{x}$, and $\boldsymbol{\Lambda}$ a diagonal matrix of which the elements are the ordered eigenvalues of $\mathbf{L}$. By combining (25.18) and (25.22), the convolution operation on the graph can be chosen as

$$
\begin{equation*}
\mathbf{U}^{T} \mathbf{g}=\operatorname{poly}(\boldsymbol{\Lambda})=\boldsymbol{\Lambda}+\boldsymbol{\Lambda}^{2}+\cdots+\boldsymbol{\Lambda}^{k} \tag{25.23}
\end{equation*}
$$

We should point out that although there are many choices for the convolutional filter, $\mathbf{g}$, we typically choose the polynomial kernel as $\operatorname{poly}(\boldsymbol{\Lambda})=\boldsymbol{\Lambda}+\boldsymbol{\Lambda}^{2}+\cdots+\boldsymbol{\Lambda}^{k}$, which ensures the localization in the vertex domain within $k$-neighbors.

### 25.6.3 Graph Spectral Filtering via Neural Networks

Given the importance of convolution in the modelling of data propagation on graphs, and the computational difficulties in its evaluation, it is natural to employ neural networks to implement the function $\mathbf{g}$ in (25.23), per layer. In this way we also take advantages of the spatial convolution operations and the universal approximation property of neural networks. This forms the basis of various spectral GCN methods.

The spectral GCN was proposed by Bruna et al. (2013), and is based on a simple spectral model given by

$$
\begin{equation*}
\mathbf{x}_{j}^{l+1}=\phi\left(\mathbf{U} \sum_{i=1}^{c_{l}} \mathbf{\Theta}_{i, j}^{l} \mathbf{U}^{T} \mathbf{x}_{i}^{l}\right) \quad j=1,2, \ldots, c_{l+1} \tag{25.24}
\end{equation*}
$$

where $l$ represents the index of each layer, $c_{l}$ is the number of filters (channels) of the $l$ th layer, $\boldsymbol{\Theta}_{i, j}^{l}$ is a diagonal matrix which contains the set of learnt parameters of the $l$ th layer, and $\phi(\cdot)$ is the activation
function of neurons. In (25.24), the summation ensures the aggregation of features filtered by different convolutional kernels, $\boldsymbol{\Theta}_{i, j}^{l}$, which is similar to a linear combination across kernels in CNNs. Although it achieves graph convolution through neural networks, this methodology has two main limitations: (i) the localization in the vertex domain cannot be ensured by $\boldsymbol{\Theta}_{i, j}^{l}$, although it is a key to the success of convolutional neural networks in extracting local stationary features; (ii) computational burden arising from the $\mathcal{O}\left(N^{2}\right)$ multiplications of $\mathbf{U}$ and $\mathbf{U}^{T}$, and the eigendecomposition of $\mathbf{L}$ to obtain $\mathbf{U}$ may be prohibitive for large graphs.

A possible way of mitigating these issues is to employ a polynomial form similar to that of (25.22), as mentioned in Section 25.6.2. This both relieves the first issue of the localization, and helps to control a balance between the localization in the vertex domain and the localization in the spectral domain (see Part II of this monograph). More specifically, to further improve the localization in the spatial domain in order to extract local patterns, we promote smoothness in the spectral domain through filtering by $\operatorname{poly}(\boldsymbol{\Lambda})$, whereby the term $\operatorname{poly}(\boldsymbol{\Lambda})$ is designed with a set of learnable parameters $\boldsymbol{\Theta}=\left\{\theta_{i}\right\}_{i=1}^{k}$, in the form

$$
\begin{equation*}
\operatorname{poly}_{\boldsymbol{\Theta}}(\boldsymbol{\Lambda})=\theta_{1} \boldsymbol{\Lambda}+\theta_{2} \boldsymbol{\Lambda}^{2}+\cdots+\theta_{k} \boldsymbol{\Lambda}^{k} . \tag{25.25}
\end{equation*}
$$

In this way, the update rule of (25.21) can now be rewritten as

$$
\begin{equation*}
\mathbf{x} \leftarrow \operatorname{poly}_{\boldsymbol{\Theta}}(\mathbf{L}) \mathbf{x}=\mathbf{U p o l y}_{\boldsymbol{\Theta}}(\boldsymbol{\Lambda}) \mathbf{U}^{T} \mathbf{x} \tag{25.26}
\end{equation*}
$$

Notice that in (25.26), the multiplication by $\mathbf{U}$ is not necessary at every layer, but the powers of $\mathbf{L}$ are needed and are computational demanding. On the basis of (25.26), Defferrard et al. (2016) further proposed the Chebyshev graph neural network, which employs the Chebyshev polynomial to ease the computation burden of poly $\boldsymbol{\Theta}(\boldsymbol{\Lambda})$, in the form

$$
\begin{equation*}
\operatorname{poly}_{\boldsymbol{\Theta}}(\boldsymbol{\Lambda})=\sum_{i=1}^{k} \theta_{i} T_{i}(\widetilde{\boldsymbol{\Lambda}}) \tag{25.27}
\end{equation*}
$$

where $\tilde{\mathbf{\Lambda}}=2 \boldsymbol{\Lambda} / \lambda_{\max }-\mathbf{I}_{N}$, while $T_{i}(\tilde{\mathbf{\Lambda}})$ is the Chebyshev polynomial that has an easy-to-compute recurrent form $T_{i}(\widetilde{\mathbf{\Lambda}})=2 \widetilde{\mathbf{\Lambda}} T_{i-1}(\widetilde{\mathbf{\Lambda}})-T_{i-2}(\widetilde{\boldsymbol{\Lambda}})$ $\left(T_{0}(\widetilde{\mathbf{\Lambda}})=\mathbf{I}\right.$, and $\left.T_{1}(\widetilde{\mathbf{\Lambda}})=\widetilde{\mathbf{\Lambda}}\right)$. With this Chebyshev polynomial, we are
able to elegantly avoid the computation of the powers of $\mathbf{L}$, through

$$
\begin{equation*}
\mathbf{x} \leftarrow \operatorname{poly}_{\mathbf{\Theta}}(\mathbf{L}) \mathbf{x}=\sum_{i=1}^{k} \theta_{i} T_{i}(\widetilde{\mathbf{L}}) \mathbf{x} \tag{25.28}
\end{equation*}
$$

where $\widetilde{\mathbf{L}}=2 \mathbf{L} / \lambda_{\max }-\mathbf{I}_{N}$. This framework significantly reduces the computational complexity from $\mathcal{O}\left(N^{2}\right)$ to $\mathcal{O}(k N)$, and has been widely used in various graph learning tasks. Recent work Kipf and Welling (2016a) further simplifies (25.28) by only employing the first-order Chebyshev polynomial $(k=1)$, which achieves superior performances in semi-supervised learning. The authors claimed that it is unnecessary to employ a $k$-order format because the first-order Chebyshev polynomial is sufficient to mitigate overfitting, while the localization of $k$-neighbors can be achieved by stacking layers of neural networks.

Despite mathematical elegance and physical intuition, spectral GCNs have been mainly limited to fixed network structures during both training and testing. More specifically, when employing spectral GCNs, the graph connections should be ascertained in advance because even a slight change in a graph connection would lead to a totally different eigenbasis. This, in turn, means that the whole graph needs to be initialized before training, which implies that spectral GCNs cannot be trained in a mini-batch manner, as the trained model is domain dependent.

Example 103: To illustrate an implementation of one typical spectral GCN (Kipf and Welling, 2016a) in semi-supervised learning, we employed the Cora dataset (Motl and Schulte, 2015) that contains 2708 machine learning related publications with seven classes (case based, genetic algorithms, neural networks, probabilistic methods, reinforcement learning, rule learning and theory). Each publication has a feature vector that indicates whether an article includes any unique selected keywords. Furthermore, the graph is constructed via its citation relationships.

For the GCN method, we employed a Pytorch implementation of the work in Kipf and Welling (2016a) which is available at https://github. com/tkipf/pygen. The basic structure of the GCN network is illustrated in Figure 8.3, and its pseudo-code is provided in Algorithm 6. In this example, the number of hidden units was set to 256 . We used different


Figure 8.3: The structure of the GCN proposed in Kipf and Welling (2016a) for semi-supervised learning.

```
Algorithm 6. Training Process of a Typical GCN (Kipf and Welling,
2016a)
    Input: Node features, \(\mathbf{X}^{0} \in \mathbb{R}^{N \times P}\), adjacency matrix, \(\mathbf{A} \in \mathbb{R}^{N \times N}\);
    while Not converged: do
    Layer 1: \(\mathbf{X}^{1}=\operatorname{GCO}^{1}\left(\mathbf{X}^{0}, \mathbf{A}\right)\)
    Output: \(\mathbf{X}^{2}=\operatorname{GCO}^{2}\left(\mathbf{X}^{1}, \mathbf{A}\right)\)
    Loss calculation on \(\mathbf{X}^{2}\) and back-propagation for optimization \(\triangleright\)
    One iteration of GCN training
    end while
    Output: The GCN with optimal \(\mathbf{W}\) for each layer
    procedure \(\mathrm{GCO}^{l}(\mathbf{X}, \mathbf{A}) \triangleright\) Graph convolution operation for the
    \(l\) th layer
    Renormalization trick: \(\tilde{\mathbf{A}}=\mathbf{I}+\mathbf{D}^{-\frac{1}{2}} \mathbf{A D}{ }^{-\frac{1}{2}} \quad \triangleright\) First-order of
    Chebyshev polynomials of (25.28)
            Graph convolution: \(\mathbf{Y}=\tilde{\mathbf{A}} \cdot \mathbf{X} \cdot \mathbf{W}^{l} \quad \triangleright \mathbf{W}^{l}\) are learnable
    parameters in the layer
    Non-linearity: \(\mathbf{Z}=\operatorname{act}(\mathbf{Y}) \quad\) Examples of activation functions
    act are sigmoid and ReLU functions
    Return: Z
    end procedure
```

ratios of data for training and plotted the test accuracy in classifying those publications into the seven classes in Figure 8.4. Observe that


Figure 8.4: Portions of data used for training versus the test accuracy on Cora dataset (Motl and Schulte, 2015). We considered a simple implementation of one typical GCN (Kipf and Welling, 2016a) for semi-supervised learning. In this example, we used one hidden layer with 256 neurons. The dropout rate was set to 0.5 and learning rate to 0.01 .
with only $10 \%$ of the available samples, a simple GCN with one hidden layer can achieve $>80 \%$ classification accuracy. It is possible to further improve the test accuracy by extending the number of hidden units or increasing network depth. This simple example, however, highlights the powerful learning ability of GCNs on structured data.

### 25.7 Link Prediction via Graph Neural Nets

Oftentimes, the dynamics of the underpinning problem at hand dictate that it is necessary to establish additional connections in a graph, in addition to the already existing edges. This is achieved through so called link prediction, which can be used in both graph completion (interpolation) and graph extension (expansion) (Liben-Nowell and Kleinberg, 2007). A direct way to perform link prediction would be to apply some heuristic similarity method to the vertices and sub-graphs, in order to estimate missing links between the vertices, as is the case with the PageRank method introduced in Section 21.5, SimRank (Jeh and Widom, 2002) and SEAL (Zhang and Chen, 2018) approaches. Alternatively, learning strategies may be employed to infer such links automatically in some "well-behaved" embedded spaces of graphs; this
is highly related to the field of graph representation learning, that is, learning a representative latent space given an existing graph.

Remark 105: The spectrum of a graph, elaborated in detail in Part I and Part II of this monograph, is a simple yet effective candidate for an embedding space, since it reflects the smoothness (frequency) of data on a graph. In this way, spectral clusters can be utilized to train a classifier to predict links (Tang and Liu, 2011).

More advanced latent space methodologies for link prediction include Deepwalk (Perozzi et al., 2014), Note2vec (Grover and Leskovec, 2016) and Line (Tang et al., 2015), all of which learn meaningful and continuous low-dimensional latent spaces by preserving (encoding) neighboring information at the vertices. For more detail, we refer to the recent reviews in Wu et al. (2019), Zhang et al. (2018b), and Chami et al. (2020).

More recently, owing to their ability to represent probabilistic generative models, GCNs have also been applied to link prediction tasks, owing to their ability to implicitly process local information in graphs. Within GCNs, two types of generative models are commonly used, the graph variational auto-encoder (VAE) and the graph generative adversarial model (GAN) (Bojchevski et al., 2018; De Cao and Kipf, 2018; Wang et al., 2017). Standard autoregressive models have also been considered to progressively generate graphs (Li et al., 2018; You et al., 2018). We here focus on VAE-based methods because they are designed to straightforwardly learn representations for link prediction, while GAN related methods are motivated by graph generation. We should also point out that the VAE- and GAN-based approaches are not independent, as VAE-based approaches take advantage of additional adversarial modules to enhance learning capacity (Pan et al., 2018; Yu et al., 2018).

The graph VAE (Kipf and Welling, 2016b) employs the VAE framework proposed in Kingma and Welling (2013), with the underpinning idea similar to the probabilistic models covered in Section 24, whereby a signal, $\mathbf{x}$ (or the graph in the case of graph VAEs), is generated by Gaussian random samples, v. However, different from the linear model approaches (Section 24), the VAE employs a neural net (decoder) as a
non-linear way of graph generation. To avoid trivial generation from random noise governed by the distribution $p(\mathbf{x} \mid \mathbf{v})$, we need to find a reasonable set of random samples, $\mathbf{v}$, that is likely to generate meaningful graph signals, $\mathbf{x}$. This resembles an encoder structure, governed by the distribution $q(\mathbf{v} \mid \mathbf{x})$, which can be trained by minimizing the distance between $q(\mathbf{v} \mid \mathbf{x})$ to the true posterior ${ }^{1} p(\mathbf{v} \mid \mathbf{x})$. We refer to Doersch (2016) for a detailed tutorial on the VAE. By using the Kullback-Leibler (KL) divergence as the distance metric in the minimization, the VAE arrives at the following relationship

$$
\begin{align*}
\log p(\mathbf{x})-\operatorname{KL}(q(\mathbf{v} \mid \mathbf{x}) \| p(\mathbf{v} \mid \mathbf{x}))= & \mathbb{E}_{q(\mathbf{v} \mid \mathbf{x})}[\log p(\mathbf{x} \mid \mathbf{v})] \\
& -\operatorname{KL}(q(\mathbf{v} \mid \mathbf{x}) \| p(\mathbf{v})) \tag{25.29}
\end{align*}
$$

Observe that by maximizing the right-hand side of (25.29), we are effectively maximizing the $\log$-likelihood of $p(\mathbf{x})$, whilst at the same time minimizing the distance between the true posterior, $p(\mathbf{v} \mid \mathbf{x})$, and the assumed one, $q(\mathbf{v} \mid \mathbf{x})$.

Remark 106: The right-hand side of (25.29) is called the evidence lower bound (ELBO) in Bayesian variational inference; more importantly, it is tractable and yields clear and physically meaningful structures, whereby $q(\mathbf{v} \mid \mathbf{x})$ is an encoder and $p(\mathbf{x} \mid \mathbf{v})$ the corresponding decoder, with $\mathbb{E}_{q(\mathbf{v} \mid \mathbf{x})}[\log p(\mathbf{x} \mid \mathbf{v})]$ as the reconstruction loss, while $\operatorname{KL}(q(\mathbf{v} \mid \mathbf{x}) \| p(\mathbf{v}))$ regularises the feasible set of $\mathbf{v}$ on some well-behaved manifolds in the latent space.

A direct extension of the framework in (25.29) to link prediction would be to estimate the missing graph connections using the VAE. The initial attempt, called the variational graph auto-encoder (VGAE) (Kipf and Welling, 2016b), models the connectivity (adjacency matrix) through the encoder, by $q(\mathbf{v} \mid \mathbf{A}, \mathbf{x})$, whilst the decoder remains the standard neural network used to reconstruct the connectivity. In this way, the encoder in the VGAE can be implemented as a GCN, while the decoder may be simplified to only a product operation $p(\mathbf{A} \mid$

[^1]

Figure 8.5: Principle of the use of VGAE in link prediction. The training process of the VGAE is based on the provided training edges and node features, whilst the test edges are not connected. Different from the standard auto-encoder, the encoder within the VGAE yields the estimated mean and covariance values of Gaussian distributions ( $\boldsymbol{\mu}_{1}, \boldsymbol{\mu}_{2}, \ldots, \boldsymbol{\mu}_{6}$ and $\boldsymbol{\sigma}_{1}, \boldsymbol{\sigma}_{2}, \ldots, \boldsymbol{\sigma}_{6}$ ), while the input to the decoder is randomly drawn from the corresponding Gaussian distributions. Being a graph variance of (25.29), the loss consists of two parts: The term $\operatorname{KL}(q(\mathbf{v} \mid \mathbf{A}, \mathbf{x}) \| p(\mathbf{v}))$ of Loss 1 which reflects how well the output Gaussian approaches the standard Gaussian distribution, $p(\mathbf{v})$ (zero mean and identity covariance), while the term $\mathbb{E}_{q(\mathbf{v} \mid \mathbf{A}, \mathbf{x})}[\log p(\mathbf{A} \mid \mathbf{v})]$ of Loss 2 corresponds to the quality of reconstruction of the adjacency matrix, $\mathbf{A}$.
$\mathbf{v})=\operatorname{sigmoid}\left(\mathbf{v} \mathbf{v}^{T}\right)$, where the sigmoid function is used to satisfy the probability constraint. Figure 8.5 provides a closer insight into the VGAE, with the corresponding pseudo-code given in Algorithm 7. To enable the use of a GCN also in the decoder and to comply with graph theory, work in Grover et al. (2019) proposes to employ an intermediate and learnable adjacency matrix within the decoder. Other improvements include regularizing VGAE through semantic validity (Ma et al., 2018), use of rich models (such as mixture models) as a prior, $p(\mathbf{v})$ (Hasanzadeh et al., 2019), and an asynchronous message passing scheme for directed acyclic graphs (Zhang et al., 2019b).
Remark 107: The VGAE differs from the standard graph auto-encoder (GAE) in the latent space, which arises from the use of variational inference, in that the representation (embedding) of each graph node is a Gaussian distribution described by the learnt mean and covariance.

A potential problem with the standard GAE is that its reconstruction loss enforces the training of the GAE towards recovering an incomplete training adjacency matrix, whereby the test and validation edges are masked out. Therefore, if a perfect reconstruction was achieved in training, in the test stage, the GAE would only recover the training edges, whilst ignoring the test and validation edges. The way of relieving this issue by the VGAE rests upon its usage of noise and uncertainty inferred through the learning process. However, this also adds disturbance and poses difficulties to the operation of the decoder during reconstruction, as this increases the reconstruction loss in an implicit adversarial way, but forces the VGAE to learn robust and meaningful graph representations.

Most recently, a framework named RCF-GAN has been proposed to seamlessly combine the benefits of the auto-encoder and adversarial learning; this is achieved via a reciprocal requirement in the latent space, while to enhance robustness, characteristic functions are employed in design of the losses (Li et al., 2020).

Remark 108: By virtue of the RCF-GAN, a meaningful representation can be learnt in the embedded space; this means that the links can be directly predicted by the embedded features, instead of being reconstructed by the decoder as with the VGAE.

The application of the RCF-GAN in graph link prediction differs from the VGAE in two main aspects: (1) the decoder no longer reconstructs the adjacency matrix but yields the node features, in order to satisfy the reciprocal requirement of the RCF-GAN; (2) the reconstruction loss in the RCF-GAN is based on the node features, and operates directly in the embedded space, as opposed to that based on the adjacency matrix in the data domain of the VGAE. The so learnt embedded features are therefore immediately graph representations and hence generalise well, thus equipping the link prediction via the RCF-GAN with the ability to effectively avoid overfitting of the training edges.

The following example demonstrates that although the test connections were not provided in the training stage, the VGAE and RCF-GAN were still able to successfully recover the missing links through the inner product (decoder) of the trained graph representations.

Example 104: We employed two well-known datasets, Cora and Citeseer (Sen et al., 2008) on a graph link prediction task. For a fair comparison, the split of datasets into training, validation and test sets was exactly the same as that in the VGAE (Kipf and Welling, 2016b), that is, $85 \%$ for training, $5 \%$ for validation and $10 \%$ for testing. For the encoder of the RCF-GAN, we adopted the same basic GCN as that adopted in the VGAE. The dimensions of the encoder for both the RCF-GAN and the VGAE were set to $\left\{m_{\text {node }}, m_{\text {layer }}, 128\right\}$, where $m_{\text {node }}$ denotes the dimension of node features and $m_{\text {layer }}$ represents the dimension of the middle layer. In the experiments, $m_{\text {layer }}$ assumed the values from $\{256,512,1024,2048\}$. The VGAE was also run with different $m_{\text {layer }}$, for a fair comparison with the available VGAE implementation (Kipf and Welling, 2016b). For the generator of the RCF-GAN, we used a simple 3-layer fully connected neural net with dimensions $\left\{128, m_{\text {layer }}\right.$, $\left.m_{\text {node }}\right\}$. We repeatedly ran the training and testing process 10 times, with both models trained over 300 epochs. Performance was evaluated through the mean values of the area under the ROC curve (AUC) and average precision (AP) metrics, with the results given in Figure 8.6.

Observe from Figure 8.6 that for all $m_{\text {layer }}$ the RCF-GAN in link prediction consistently outperformed the VGAE and the spectral clustering methods, with a significant margin. More specifically, the VGAE achieved its best performance at approximately $m_{\text {layer }}=512$, with the obtained AUC (AP) of 0.910 (0.929) for Cora and 0.904 (0.921) for Citeseer. Further increasing $m_{\text {layer }}$ may lead to overfitting of the VGAE, thus decreasing its performances in link prediction. However, with the increase in network sizes, the performances of the RCF-GAN improved correspondingly, and the AUC (AP) scores reached 0.936 (0.941) for Cora and 0.944 ( 0.946 ) for Citeseer. This may be due to the fact that the RCF-GAN does not directly perform the reconstruction of the adjacency matrix but learns a semantic embedded space for link prediction. This improvement also validates the effectiveness and efficiency of the learnt embedded space within the RCF-GAN.

(b) Citeseer

Figure 8.6: The AUC and AP scores for graph link prediction using the RCF-GAN and VGAE, with $m_{\text {layer }}$ ranging from 256 to 2048. The solid lines show the AUC and AP metrics of the RCF-GAN in testing, whereas the dashed lines designate the results for the standard VGAE. Since spectral clustering based graph link prediction (Tang and Liu, 2011) is a typical baseline for using GCNs, we also plot the spectral clustering results from Kipf and Welling (2016b) in dotted lines.

```
Algorithm 7. Training Process of VGAE Kipf and Welling (2016b)
    Input: Node features, \(\mathbf{X}^{0} \in \mathbb{R}^{N \times P}\), (incomplete) training adjacency
    matrix, \(\mathbf{A} \in \mathbb{R}^{N \times N}\);
    while Not converged: do
        Forward process of the encoder:
        Layer 1: \(\mathbf{X}^{1}=\operatorname{GCO}^{1}\left(\mathbf{X}^{0}, \mathbf{A}\right)\)
        Layer 2: \(\mathbf{U}=\operatorname{GCO}^{2}\left(\mathbf{X}^{1}, \mathbf{A}\right) \triangleright\) Output Gaussian mean vector of
    each node
    Layer 3: \(\mathbf{C}=\operatorname{GCO}^{3}\left(\mathbf{X}^{1}, \mathbf{A}\right) \quad \triangleright\) Output Gaussian diagonal
    covariance vector of each node
```

        Forward process of the decoder:
        For each node, \(n\), draw one sample, \(\mathbf{v}_{n}\), from \(\mathcal{N}\left(\mathbf{u}_{n}, \mathbf{c}_{n}\right)\), where
    \(\mathbf{u}_{n}\) and \(\mathbf{v}_{n}\) are the \(n\)th rows of \(\mathbf{U}\) and \(\mathbf{C}\)
        Inner product operation: \(\hat{\mathbf{A}}=\operatorname{sigmoid}\left(\mathbf{V} \mathbf{V}^{T}\right)\), where \(\mathbf{v}_{n}\) is the
    \(n\)th row of \(\mathbf{V}\)
    Loss calculation: \(\frac{1}{N} \sum_{n=1}^{N} \operatorname{KL}\left(\mathcal{N}\left(\mathbf{u}_{n}, \mathbf{c}_{n}\right) \| \mathcal{N}(\mathbf{0}, \mathbf{I})\right)+\operatorname{BCE}(\hat{\mathbf{A}}, \mathbf{A})\)
    \(\triangleright\) BCE denotes the binary cross entropy loss
    Back-propagation for optimization \(\triangleright\) One iteration of
    training the VGAE
    end while
    Output: Optimal embedding \(\mathbf{U}\) of nodes
    procedure \(\mathrm{GCO}^{l}(\mathbf{X}, \mathbf{A}) \triangleright\) Graph convolution operation for the
    \(l\) th layer
        Renormalization trick: \(\tilde{\mathbf{A}}=\mathbf{I}+\mathbf{D}^{-\frac{1}{2}} \mathbf{A D}^{-\frac{1}{2}} \quad \triangleright\) First-order of
        Chebyshev polynomials of (25.28)
            Graph convolution: \(\mathbf{Y}=\tilde{\mathbf{A}} \cdot \mathbf{X} \cdot \mathbf{W}^{l} \quad \triangleright \mathbf{W}^{l}\) is learnable
        parameters in the layer
            Non-linearity: \(\mathbf{Z}=\operatorname{act}(\mathbf{Y}) \quad \triangleright\) Examples of activation functions
        act can be sigmoid and ReLU functions
            Return: Z
    end procedure
    
## 26

## Tensor Representation of Lattice-Structured Graphs

It is often desirable to generalize graphs in order to account directly for higher-order and higher-dimensional relationships between data sources (Cooper and Dutle, 2012; Saito et al., 2018; Zhou et al., 2007). One such way is via the hypergraph approach, which allows the edges to link more than two vertices (Berge, 1984). Another possibility is through a multi-layer network of graphs, whereby graph vertices reside on a high dimensional regular lattice structure which results from the Cartesian product of several one-dimensional path graphs (for more detail, see Part I). We next show that tensors (multidimensional data arrays) are perfectly suited to model the latter approach. It is further shown that tensors can be considered as a special class of graph signals, which in turn allows the associated adjacency matrices to exhibit a physically meaningful structured form, referred to as Kronecker summable. By virtue of the underlying multilinear tensor algebra, this effectively reduces the number of parameters required to model the entire graph connectivity structure (Bacciu and Mandic, 2020).

### 26.1 Tensorization of Graph Signals in High-Dimensional Spaces

A tensor of order $M$ is an $M$-way data array, denoted by $\mathcal{X} \in \mathbb{R}^{I_{1} \times \cdots \times I_{M}}$. For example, a vector $\mathbf{x} \in \mathbb{R}^{I}$ is an order-1 tensor, a matrix $\mathbf{X} \in \mathbb{R}^{I_{1} \times I_{2}}$ is an order- 2 tensor, while a 3 -way array $\mathcal{X} \in \mathbb{R}^{I_{1} \times I_{2} \times I_{3}}$ is an order-3 tensor. The $m$ th dimension of an order- $M$ tensor, $\mathcal{X} \in \mathbb{R}^{I_{1} \times \cdots \times I_{M}}$, is referred to as the $m$ th mode which is of size $I_{m}$ entries.

To establish a relationship between graph signals and tensors, we begin by considering an $N$-vertex graph, denoted by $\mathcal{G}=\{\mathcal{V}, \mathcal{E}\}$. With each vertex on the graph we can associate a variable (signal), denoted by $x(n) \in \mathbb{R}$, which maps a vertex, $n \in \mathcal{V}$, to a real, that is, $x: \mathcal{V} \mapsto \mathbb{R}$. In other words, each vertex represents a scalar-valued field in a singledimensional coordinate system. When considering all $N$ vertices in $\mathcal{V}$, we can form the vector $\mathbf{x} \in \mathbb{R}^{N}$ which defines the mapping $\mathbf{x}: \mathcal{V} \mapsto \mathbb{R}^{N}$.

On the other hand, if a graph resides in an $M$-dimensional space, then each vertex, $n \in \mathcal{V}$, has a one-to-one correspondence with a unique coordinate vector in this space, denoted by $\left(i_{1}, \ldots, i_{M}\right) \in \mathbb{N}^{M}$, where $i_{m} \in \mathbb{N}$ is the coordinate associated with the $m$ th axis. In other words, there exists a unique mapping $n \mapsto\left(i_{1}, \ldots, i_{M}\right)$. In this way, the graph vertex signal can be viewed as a field in an $M$-dimensional coordinate system, that is, each vertex can be defined equivalently as $x(n) \equiv x\left(i_{1}, \ldots, i_{M}\right) \in \mathbb{R}$, that is, it induces the mapping $x: \mathbb{N}^{M} \mapsto \mathbb{R}$.

When discrete points in the field, $x: \mathbb{N}^{M} \mapsto \mathbb{R}$, are sampled using a regular lattice of dimensions $I_{1} \times \cdots \times I_{M}$, thereby sampling a total of

$$
\prod_{m=1}^{M} I_{m} \equiv N
$$

discrete points, the collection of samples naturally forms the tensor $\mathcal{X} \in \mathbb{R}^{I_{1} \times \cdots \times I_{M}}$, with its $\left(i_{1}, \ldots, i_{M}\right)$ th entry defined as

$$
\begin{equation*}
[\mathcal{X}]_{i_{1} \ldots i_{M}}=x\left(i_{1}, \ldots, i_{M}\right), \quad i_{m} \in \mathbb{N}, m=1,2, \ldots, M \tag{26.1}
\end{equation*}
$$

Figure 8.1 illustrates a collection of discrete points from a field in a 3 -dimensional coordinate system, which together form an order-3 tensor. This procedure is referred to as tensorization.


Figure 8.1: Tensorization of discrete samples from a field $x: \mathbb{N}^{3} \mapsto \mathbb{R}$.

Remark 109: Real-world examples of a field in $M$-dimensional coordinates include:

- Netflix ratings in the user $\times$ movie space $(M=2)$;
- Temperature measurements in the longitude $\times$ latitude $\times$ altitude space $(M=3)$;
- Video pixels in the time $\times$ column $\times$ row $\times R G B$ space $(M=4)$;
- EEG signals in the time $\times$ frequency $\times$ channel $\times$ subject $\times$ trial space $(M=5)$.


### 26.2 Tensor Decomposition

If the underlying field, $x: \mathbb{N}^{M} \mapsto \mathbb{R}$, is defined as a multilinear map of the form

$$
\begin{equation*}
x: \underbrace{\mathbb{N} \times \cdots \times \mathbb{N}}_{M \text { times }} \mapsto \mathbb{R} \tag{26.2}
\end{equation*}
$$

then it is said to be linearly separable, and therefore admits the following decomposition

$$
\begin{equation*}
x\left(i_{1}, \ldots, i_{M}\right)=\prod_{m=1}^{M} x_{m}\left(i_{m}\right) \tag{26.3}
\end{equation*}
$$

In other words, the value of $x\left(i_{1}, \ldots, i_{M}\right)$ is given by the product of $M$ independent single-dimensional functions, $x_{m}: \mathbb{N} \mapsto \mathbb{R}$, each of


Figure 8.2: Rank-1 CPD of an order-3 tensor.
which is associated with the $m$ th coordinate axis of the underlying $M$-dimensional coordinate system. In this way, a tensor, $\mathcal{X} \in \mathbb{R}^{I_{1} \times \cdots \times I_{M}}$, which is sampled from a linearly separable field of the kind in (26.3) admits the following rank-1 canonical polyadic decomposition (CPD)

$$
\begin{equation*}
\mathcal{X}=\mathbf{x}_{1} \circ \cdots \circ \mathbf{x}_{M} \tag{26.4}
\end{equation*}
$$

with the symbol $\circ$ denoting the outer product operator, and $\mathbf{x}_{m} \in \mathbb{R}^{I_{m}}$ being a parameter vector associated with the $m$ th coordinate axis.
Remark 110: The property in (26.4) is referred to as the Kronecker separability condition, which is fundamental to most tensor decompositions and learning algorithms.

With regard to the linear separability property in (26.3), the $i$ th entry of $\mathbf{x}_{m}$ is given by $\left[\mathbf{x}_{m}\right]_{i}=x_{m}(i)$. Figure 8.2 shows the rank- 1 CPD of an order-3 tensor.

Kronecker separable tensors admit a vector representation (vectorization), denoted by $\mathbf{x}=\operatorname{vec}(\mathcal{X}) \in \mathbb{R}^{N}$, which can be expressed as

$$
\begin{equation*}
\mathbf{x}=\mathbf{x}_{M} \otimes \cdots \otimes \mathbf{x}_{1} \tag{26.5}
\end{equation*}
$$

and is a direct consequence of (26.4), where the symbol $\otimes$ denotes the Kronecker product operator (see Part I).
Example 105: Consider the data matrix, $\mathbf{X} \in \mathbb{R}^{I \times J}$, which contains the Netflix ratings assigned by $I$ users to $J$ movies, whereby the $(i, j)$ th entry designates the rating assigned by the $i$ th user to the $j$ th movie, $x(i, j) \in \mathbb{R}$. The graph representation of this dataset consists of $(I J)$ vertices residing in a two-dimensional space (user $\times$ movie). Owing to the lattice-like structure of the graph, we can employ its inherent


Figure 8.3: Rank-1 CPD of the Netflix rating data matrix.
order-2 tensor representation, whereby the data can be approximated using the following rank-1 CPD

$$
\begin{equation*}
\mathbf{X} \approx \mathbf{x}_{1} \circ \mathbf{x}_{2} \equiv \mathbf{x}_{1} \mathbf{x}_{2}^{T} \tag{26.6}
\end{equation*}
$$

with $\mathbf{x}_{1} \in \mathbb{R}^{I}$ being the factor associated with the user axis, and $\mathbf{x}_{2} \in \mathbb{R}^{J}$ the factor associated with the movie axis. Note that for order-2 tensors, the CPD is equivalent to the singular value decomposition (SVD). Figure 8.3 illustrates the tensor decomposition of the Netflix rating data matrix.

The factorization of $\mathbf{X}$ assumes that the rating assigned by the $i$ th user to the $j$ th movie can be approximated as

$$
\begin{equation*}
x(i, j) \approx x_{1}(i) x_{2}(j) \tag{26.7}
\end{equation*}
$$

where $x_{1}(i) \equiv\left[\mathbf{x}_{1}\right]_{i}$ and $x_{2}(j) \equiv\left[\mathbf{x}_{2}\right]_{j}$. In other words, the rating, $x(i, j)$, can be approximated by a rating assigned by the $i$ th user to all movies, $x_{1}(i)$, multiplied by a rating assigned to the $j$ th movie by all users, $x_{2}(j)$.

The so achieved parameter reduction becomes evident, since we have reduced a fully connected $(I J)$ parameter model to an $(I+J)$ parameter model. This parameter reduction is most pronounced for higher-order tensors, e.g., an order- $N$ tensor model with $\prod_{n=1}^{N} I_{n}$ parameters (exponential) reduces to a $\sum_{n=1}^{N} I_{n}$ parameter (linear) model.

### 26.3 Connectivity of a Tensor

We next show that the tensor structure inherent to $\mathcal{X} \in \mathbb{R}^{I_{1} \times \cdots \times I_{M}}$ can be modelled naturally as a graph. This is achieved by exploiting
the well-known property of lattice-structured graphs which can be decomposed into constituent single-dimensional path graphs.

Building upon the Cartesian product of two disjoint path graphs, as considered in Part I of this monograph, the Cartesian product of $M$ disjoint $I_{m}$-vertex path graphs, $\mathcal{G}_{m}=\left(\mathcal{V}_{m}, \mathcal{E}_{m}\right)$ for $m=1, \ldots, M$, yields a graph with an $M$-dimensional regular lattice structure, denoted by $\mathcal{G}=\mathcal{G}_{M} \square \cdots \square \mathcal{G}_{1}=(\mathcal{V}, \mathcal{B})$, with the symbol $\square$ denoting the graph Cartesian product. In this way, the resulting vertex set takes the form $\mathcal{V}=\mathcal{V}_{M} \times \cdots \times \mathcal{V}_{1}$, and the resulting graph contains a total of $\prod_{m=1}^{M} I_{m} \equiv$ $N$ vertices.

If the adjacency matrix of the $m$ th path graph, $\mathcal{G}_{m}$, is denoted by $\mathbf{A}_{m} \in \mathbb{R}^{I_{m} \times I_{m}}$, then the adjacency matrix of the resulting $M$-dimensional regular lattice graph, $\mathcal{G}$, is given by

$$
\begin{equation*}
\mathbf{A}=\left(\mathbf{A}_{M} \oplus \cdots \oplus \mathbf{A}_{1}\right) \in \mathbb{R}^{N \times N} \tag{26.8}
\end{equation*}
$$

where the symbol $\oplus$ denotes the Kronecker sum operator (we refer to Part I). Such an adjacency matrix is said to be Kronecker summable.
Remark 111: The adjacency matrix, A, when interpreted through the underlying tensor, describes the connectivity between the entries of vectorization of a tensor, $\mathbf{x} \in \mathbb{R}^{N}$, while $\mathbf{A}_{m} \in \mathbb{R}^{I_{m} \times I_{m}}$ describes the connectivity between entries along the $m$ th mode. Under this model, the entries of the tensor are only connected to neighboring entries which reside in the same fiber.

For illustration purposes, Figure 8.4 shows the Cartesian product of three disjoint path graphs, which results in a graph with a threedimensional lattice structure. This graph would naturally represent the connectivity between the entries of an order-3 tensor, $\mathcal{X} \in \mathbb{R}^{2 \times 3 \times 2}$. Next, consider the order-2 tensor, $\mathbf{X} \in \mathbb{R}^{I_{1} \times I_{2}}$, with entries sampled from the field, $x: \mathbb{N}^{2} \mapsto \mathbb{R}$, using a 2-dimensional regular lattice as illustrated in Figure 8.5.

Example 106: Consider a field on a two-dimensional coordinate system, denoted by $x: \mathbb{N}^{2} \mapsto \mathbb{R}$, and illustrated in Figure 8.6. If the scalar field is linearly separable, that is, $x\left(t_{1}, t_{2}\right)=x_{1}\left(t_{1}\right) x_{2}\left(t_{2}\right)$, then the sampled tensor, $\mathbf{X}$, is Kronecker separable, and can therefore be expressed as

$$
\begin{equation*}
\mathbf{X}=\mathbf{x}_{1} \circ \mathbf{x}_{2} \Longleftrightarrow \operatorname{vec}(\mathbf{X})=\mathbf{x}_{2} \otimes \mathbf{x}_{1} \tag{26.9}
\end{equation*}
$$



Figure 8.4: Cartesian product of 3 path graphs.


Figure 8.5: Order-2 tensor, $\mathbf{X} \in \mathbb{R}^{I_{1} \times I_{2}}$, sampled from $x$ : $\mathbb{N}^{2} \mapsto \mathbb{R}$.


Figure 8.6: An example of a field, $x: \mathbb{N}^{2} \mapsto \mathbb{R}$.


Figure 8.7: Path graph signal, $\mathbf{x}_{1} \in \mathbb{R}^{I_{1}}$, sampled from $x_{1}: \mathbb{N} \mapsto \mathbb{R}$.
through $x_{1} \in \mathbb{R}^{I_{1}}$ and $x_{2} \in \mathbb{R}^{I_{2}}$ as data on path graphs sampled respectively from the single-dimensional fields, $x_{1}: \mathbb{N} \mapsto \mathbb{R}$ and $x_{2}: \mathbb{N} \mapsto$ $\mathbb{R}$, as illustrated in Figures 8.7-8.8.


Figure 8.8: Path graph signal, $\mathbf{x}_{2} \in \mathbb{R}^{I_{2}}$, sampled from $x_{2}: \mathbb{N} \mapsto \mathbb{R}$.

### 26.4 DFT of a Tensor

We have shown in expression (26.8) above that tensors can be considered as a special class of graphs which exhibit a Kronecker summable adjacency matrix. In that case, the DFT of a tensor can be naturally obtained from the graph Fourier transform (GFT), which was introduced in Part II of this monograph. In this way, the GFT of a graph with a lattice structure can be performed by evaluating the eigenvalue decomposition of the adjacency matrix $\mathbf{A}$, given by

$$
\begin{equation*}
\mathbf{A}=\mathbf{U} \boldsymbol{\Lambda} \mathbf{U}^{-1} \tag{26.10}
\end{equation*}
$$

where $\mathbf{U} \in \mathbb{R}^{N \times N}$ and $\boldsymbol{\Lambda} \in \mathbb{R}^{N \times N}$ denote respectively the matrices of eigenvectors and eigenvalues of $\mathbf{A}$.

Owing to the Kronecker sum structure of $\mathbf{A}$ in (26.8), the eigenvector and eigenvalue matrices of GFT exhibit the following structure

$$
\begin{align*}
\mathbf{U} & =\left(\mathbf{U}_{M} \otimes \cdots \otimes \mathbf{U}_{1}\right)  \tag{26.11}\\
\mathbf{\Lambda} & =\left(\mathbf{\Lambda}_{M} \oplus \cdots \oplus \boldsymbol{\Lambda}_{1}\right) \tag{26.12}
\end{align*}
$$

where $\mathbf{U}_{m} \in \mathbb{R}^{I_{m} \times I_{m}}$ and $\boldsymbol{\Lambda} \in \mathbb{R}^{I_{m} \times I_{m}}$ respectively denote the matrices of eigenvectors and eigenvalues of the $m$ th path graph adjacency matrix, $\mathbf{A}_{m}$, obtained through

$$
\begin{equation*}
\mathbf{A}_{m}=\mathbf{U}_{m} \boldsymbol{\Lambda}_{m} \mathbf{U}_{m}^{-1} \tag{26.13}
\end{equation*}
$$

Therefore, the eigenvectors of $\mathbf{A}$ are said to be Kronecker separable, while the eigenvalues are Kronecker summable.

### 26.5 Unstructured Graphs

Consider an $N$-vertex graph, $\mathcal{G}$, with vertex signals sampled from the field, $x: \mathbb{R}^{M} \mapsto \mathbb{R}$, using a regular lattice, which together form the order- $M$ tensor, $\mathcal{X} \in \mathbb{R}^{I_{1} \times \cdots \times I_{M}}$, with $\prod_{m=1}^{M} I_{m} \equiv N$.

Similarly, consider a $K$-vertex graph, $\tilde{\mathcal{G}}$, with vertex signals also sampled from the same field, $x: \mathbb{R}^{M} \mapsto \mathbb{R}$, but using instead an unstructured sampling scheme. In this way, the unstructured graph can be defined as a subset of a lattice-structured graph, i.e., $\tilde{\mathcal{G}} \subset \mathcal{G}$.

The vertex signals of $\tilde{\mathcal{G}}$, denoted by the vector $\tilde{\mathbf{x}} \in \mathbb{R}^{K}$, can therefore be defined as

$$
\begin{equation*}
\tilde{\mathbf{x}}=\Pi \operatorname{vec}(\boldsymbol{\mathcal { X }}) \tag{26.14}
\end{equation*}
$$

where $\Pi \in \mathbb{R}^{K \times N}$ is a sampling matrix, with entries defined as

$$
[\boldsymbol{\Pi}]_{k n}= \begin{cases}1, & \text { if } \tilde{x}(k) \equiv x(n)  \tag{26.15}\\ 0, & \text { otherwise }\end{cases}
$$

with $\tilde{x}(k) \in \mathbb{R}$ and $x(n) \in \mathbb{R}$ denoting respectively the $k$ th vertex of $\tilde{\mathcal{G}}$ and the $n$th vertex of $\mathcal{G}$.

Although the lattice-structured graph, $\mathcal{G}$, exhibits a Kronecker separable signal vector and a Kronecker summable adjacency matrix, the associated unstructured graph, $\tilde{\mathcal{G}}$, does not have such properties because, in general, $\boldsymbol{\Pi}$ is not separable. This can be seen from the relationship between the adjacency matrices of $\tilde{\mathcal{G}}$ and $\mathcal{G}$, which is given by

$$
\begin{equation*}
\tilde{\mathbf{A}}=\boldsymbol{\Pi} \mathbf{A} \boldsymbol{\Pi}^{T}=\boldsymbol{\Pi}\left(\mathbf{A}_{M} \oplus \cdots \oplus \mathbf{A}_{1}\right) \boldsymbol{\Pi}^{T} . \tag{26.16}
\end{equation*}
$$

Notice that the last term above cannot be decomposed further if $\boldsymbol{\Pi}$ is not separable. A direct consequence of the result in (26.16) is that the GFT bases of $\tilde{\mathcal{G}}$ (eigenvalue decomposition of $\tilde{\mathbf{A}}$ ) do not exhibit the Kronecker summability either.
Example 107: Referring back to Example 106, the graph signal resulting from an irregular sampling of the field $x: \mathbb{R}^{2} \mapsto \mathbb{R}$ is not Kronecker separable as it cannot be represented as a Cartesian product of two path graphs (as in Figures 8.5-8.8), as illustrated in Figure 8.9.


Figure 8.9: Unstructured graph, $\tilde{\mathbf{x}} \in \mathbb{R}^{K}$, sampled from $x: \mathbb{N}^{2} \mapsto \mathbb{R}$.

### 26.6 Tensor Representation of Multi-Relational Graphs

The rapidly growing prominence of multi-relational network data in areas as diverse as social network modeling, the semantic web, bioinformatics and artificial intelligence, has brought to light the increasing importance of Data Analytics on domains where the entities are interconnected by multiple relations. To put this into context of graphs, while traditional graph models only account for a single relation type, designated by the adjacency matrix, $\mathbf{A} \in \mathbb{R}^{N \times N}$, a multi-relational $N$-vertex graph may exhibit a large number, say $M$, of distinct relation types between vertices. In this case, a multi-relational graph would be defined by $M$ adjacency matrices, $\mathbf{A}_{m} \in \mathbb{R}^{N \times N}$ for $m=1, \ldots, M$; one for each relation type.

While it is possible to model this situation through a short and wide $N \times M N$ dimensional matrix, this would both involve numerical difficulties and obscure physical relevance. To this end, to model such a multi-relational graph in a parsimonious and compact manner, we may construct a three-way tensor, $\mathcal{A} \in \mathbb{R}^{N \times N \times M}$, whereby its $m$ th frontal slice is given by $\mathbf{A}_{m}$. In this way, the first two modes define the entity domain, while the third mode represents the relation domain, as illustrated in Figure 8.10. The tensor entry $[\mathcal{A}]_{i j k}=1$ therefore designates the existence of a relation between the $i$ th and $j$ th entities within the $k$ th relation type; otherwise, for non-existing and unknown relations, the entry is set to zero.

The work in Lin et al. (2008, 2009), Tang et al. (2009), Nickel et al. (2011), Papalexakis et al. (2013), Gauvin et al. (2014), Verma and Bharadwaj (2017a), Verma and Bharadwaj (2017b), and Katsimpras and Paliouras (2019) employs such tensor model to learn an inherent


Figure 8.10: Construction of a multi-relational adjacency tensor, $\mathcal{A} \in \mathbb{R}^{N \times N \times M}$, where $E_{n}$ denotes the $n$th entity and $R_{m}$ the $m$ th relation type.
structure from multi-relational data. The following rank- $L$ factorization was employed, known as the RESCAL decomposition (Nickel et al., 2011), whereby each frontal slice of $\mathcal{A}$ is factorized as

$$
\begin{equation*}
\mathbf{A}_{m}=\mathbf{U} \mathbf{R}_{m} \mathbf{U}^{T}, \quad m=1, \ldots, M \tag{26.17}
\end{equation*}
$$

where $\mathbf{U} \in \mathbb{R}^{N \times L}$ is a factor matrix which maps the $N$-dimensional entity space to an $L$-dimensional latent component space, and $\mathbf{R}_{m} \in$ $\mathbb{R}^{L \times L}$ models the interactions of latent components within the $m$ th relation type. Alternatively, this can be expressed in terms of the factorization of the tensor $\mathcal{A}$, in the form

$$
\begin{equation*}
\mathcal{A}=\boldsymbol{R} \times{ }_{1} \mathbf{U} \times_{2} \mathbf{U} \tag{26.18}
\end{equation*}
$$

where the symbol $\times_{n}$ denotes the mode- $n$ product, and $\mathcal{R} \in \mathbb{R}^{L \times L \times M}$ is the latent core tensor with $\mathbf{R}_{m}$ being its $m$ th frontal slice, as illustrated in Figure 8.11. Such a factorization allows for link-based clustering, whereby the entities $E_{1}, \ldots, E_{N}$ are clustered according to the information in $\mathbf{U}$ only. In doing so, the similarity between entities is computed based on their similarity across multiple relations.
Example 108: Social networks play an important role in the functionality of an organization and it is therefore of considerable interest to analyze the properties of such networks. The adoption of social networking services within organizations can largely facilitate the interaction


Figure 8.11: Factorization of a multi-relational adjacency tensor, $\mathcal{A} \in \mathbb{R}^{N \times N \times M}$ as in (26.17).
and collaboration between employees. For example, a social network could reveal information about the characteristics of an employee which could then be used to improve efficiency and influence team structuring.

As shown in Figure 8.8 and Example 95, a social network can be modelled as a graph, whereby each vertex represents an individual (employee) and each edge designates the existence of a social relationship between two individuals. While a conventional graph can model social networks involving one type of relationship, multi-relational graphs allow for the modelling of multiple (and different) types of relationships. Figure 8.12 illustrates a multi-relational social network involving three employees (vertices) who communicate via email (blue edge), LinkedIn (green edge) and Skype (orange edge). Observe that social relationships may be directed, e.g., employee A sends emails (blue edge) to employee B but not vice versa. If the adjacency matrix associated with the $m$ th relationship type is given by $\mathbf{A}_{m} \in \mathbb{R}^{3 \times 3}$ for $m=1,2,3,4$, then the adjacency tensor, $\mathcal{A} \in \mathbb{R}^{3 \times 3 \times 4}$, can be constructed to model the entire social network. Once the latent components (factor) matrix, $\mathbf{U} \in \mathbb{R}^{3 \times L}$, is inferred from $\mathcal{A}$ using the factorization in (26.17), it is possible to apply feature-based clustering to obtain the inherent community structure in such multi-relational network. The output of this step would be a set of $K$ disjoint communities (sub-graphs), $\left\{\mathcal{V}_{1}, \ldots, \mathcal{V}_{K}\right\}$.

### 26.7 Multi-Graph Tensor Networks

After exploring the inherent links between graphs and tensors, in Section 26.6, and between graphs and neural networks, in Section 25, it is natural to further explore the possibilities enabled by a joint consideration of these three domains. One such general approach is referred to


Email
(a)

(b)

Figure 8.12: Social network modelled as a multi-relational graph. (a) Graph representation of the social network. (b) Adjacency tensor, $\mathcal{A} \in \mathbb{R}^{3 \times 3 \times 4}$, associated with the social network in (a).
as the Multi-Graph Tensor Network (MGTN) model (Xu et al., 2020), which aims to fully exploit the virtues of both graphs and tensors in a deep learning setting. For a joint account between tensors and neural networks, we refer to Calvi et al. (2019) and Bacciu and Mandic (2020). In this way, the MGTN framework is capable of:

- Handling irregular data that reside on multiple graph domains;
- Leveraging on the compression and structure-preserving properties of tensor networks, to enhance the expressive power of NNs, at a reduced parameter complexity.

The MGTN generalises the Recurrent Graph Tensor Network (RGTN) model, introduced in Xu and Mandic (2020), which enables deep modelling on irregular domains and was developed with the aim
to model time-series problems related to sequential data, and was only defined for a single graph domain. To make this concept suitable for applications beyond time-series and in a Big Data setting, the MGTN operates in a multi-modal data setting defined on multiple graph domains and thus not limited to time-series.

More precisely, the time-based multi-linear graph filter, $\boldsymbol{\mathcal { R }}$, which underpins the RGTN in Xu and Mandic (2020), employs a time-graph adjacency matrix that reflects the temporal flow of information. On the other hand, for a given weighted graph adjacency matrix, $\mathbf{A} \in \mathbb{R}^{I_{1} \times I_{1}}$, the MGTN approach constructs a multi-linear graph filter in the tensor domain, $\mathcal{F} \in \mathbb{R}^{J_{1} \times I_{1} \times J_{1} \times I_{1}}$, given by

$$
\begin{equation*}
\mathcal{F}=\operatorname{ten}(\mathbf{I}+(\mathbf{A} \otimes \mathbf{P})) \tag{26.19}
\end{equation*}
$$

where the propagation matrix, $\mathbf{P} \in \mathbb{R}^{J_{1} \times J_{1}}$, models the flow of information between neighboring vertices (as opposed to successive time-steps in the RGTN case) and the operator ten(.) represents a suitable tensorization, as, for example, that in Figure 8.1. This allows us to adapt the multi-linear graph filter, $\mathcal{F}$, to any given graph domain of any data modality.

Consider a general multi-graph learning problem where the input is an order- $(M+1)$ tensor, $\mathcal{X} \in \mathbb{R}^{J_{0} \times I_{1} \times I_{2} \times \cdots \times I_{M}}$, with $J_{0}$ features indexed along $M$ physical modes $\left\{I_{1}, I_{2}, \ldots, I_{M}\right\}$, such that a graph, $\mathcal{G}^{(m)}$, is associated with each of the $I_{m}$ modes, $m=1, \ldots, M$. For this problem, we can define:

1. $\mathcal{A}=\left\{\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(M)}\right\}$, a set of adjacency matrices, $\mathbf{A}^{(m)} \in$ $\mathbb{R}^{I_{m} \times I_{m}}$, constructed from the corresponding graphs $\mathcal{G}^{(m)}$.
2. $\mathcal{W}=\left\{\mathbf{W}^{(1)}, \mathbf{W}^{(2)}, \ldots, \mathbf{W}^{(M)}\right\}$, a set of weight matrices, $\mathbf{W}^{(m)} \in$ $\mathbb{R}^{J_{m} \times J_{m-1}}$, used for feature transforms, where $J_{m}$, for $m=1, \ldots, M$, controls the number of feature maps at every mode $m$.
3. $\mathcal{P}=\left\{\mathbf{P}^{(1)}, \mathbf{P}^{(2)}, \ldots, \mathbf{P}^{(M)}\right\}$, a set of propagation matrices, $\mathbf{P}^{(m)} \in$ $\mathbb{R}^{J_{m} \times J_{m}}$, modelling the propagation of information over the neighbors of the graph $\mathcal{G}^{(m)}$.

The general Multi-Graph Tensor Network (gMGTN) layer is characterized by the forward pass

$$
\begin{align*}
\boldsymbol{\mathcal { Y }}=\phi & \left(\mathcal{F}^{(M)} \times{ }_{3,4}^{1, M+1} \mathbf{W}^{(M)} \times{ }_{2}^{1} \cdots \times{ }_{2}^{1} \mathcal{F}^{(2)} \times{ }_{3,4}^{1,3} \mathbf{W}^{(2)}\right. \\
& \left.\times{ }_{2}^{1} \mathcal{F}^{(1)} \times{ }_{3,4}^{1,2} \mathbf{W}^{(1)} \times{ }_{2}^{1} \boldsymbol{\mathcal { X }}\right) \tag{26.20}
\end{align*}
$$

where $\phi(\cdot)$ is an optional non-linear activation function and $\boldsymbol{\mathcal { F }}^{(m)}=$ $\operatorname{ten}\left(\mathbf{I}+\left(\mathbf{A}^{(m)} \otimes \mathbf{P}^{(m)}\right)\right)$. The so defined forward pass generates a feature map, $\mathcal{Y} \in \mathbb{R}^{J_{M} \times I_{1} \times \cdots \times I_{M}}$, from the input tensor, $\mathcal{X}$, through a series of multi-linear graph filter and weight matrix contractions, which essentially iterates the graph filtering operation across all $M$ graph domains.

Since the gMGTN learns a propagation matrix, $\mathbf{P}^{(m)}$, and a weight matrix, $\mathbf{W}^{(m)}$, for each of the $M$ graphs, when $J_{1}=J_{2}=\cdots=J_{M}=J$, this results in a parameter complexity of $\mathcal{O}\left(M J^{2}\right)$, which is linear in the number of graphs, $M$, but quadratic in the size of feature maps, $J$, so that the computation quickly becomes intractable for high dimensional multi-graph problems.

The computational bottleneck can be resolved by approximating $\mathbf{P}^{(m)} \approx \mathbf{I}$ for $m=1, \ldots, M$, and by using a single weight matrix, $\mathbf{W}^{(x)} \in \mathbb{R}^{J_{1} \times J_{0}}$, for all of the graph domains, where $J_{1}$ controls the number of hidden units (feature maps).

The resulting fast MGTN (fMGTN) is shown in Figure 8.13, and exhibits the following reduced forward pass

$$
\begin{equation*}
\boldsymbol{\mathcal { Y }}=\phi\left(\mathbf{F}^{(M)} \times{ }_{2}^{M+1} \cdots \times_{2}^{4} \mathbf{F}^{(2)} \times{ }_{2}^{3} \mathbf{F}^{(1)} \times{ }_{2}^{2} \mathbf{W}^{(1)} \times \frac{1}{2} \boldsymbol{\mathcal { X }}\right) \tag{26.21}
\end{equation*}
$$

where $\mathbf{F}^{(m)}=\left(\mathbf{I}+\mathbf{A}^{(m)}\right)$ is a standard graph shift filter. As the fMGTN does not have to learn $\mathbf{P}^{(m)}$ or $\mathbf{W}^{(m)}$, the parameter complexity of the forward pass is reduced from $\mathcal{O}\left(M J^{2}\right)$ to $\mathcal{O}\left(J^{2}\right)$ but at the cost of lower expressive power. After extracting the feature map, $\mathcal{Y} \in \mathbb{R}^{J_{1} \times I_{1} \times \cdots \times I_{M}}$, it is customary to flatten the extracted features into a vector, in order to pass them through dense neural network layers to generate the fMGTN output. To further reduce parameter complexity, the weight matrices of the dense layers can be tensorized and represented in some compressed tensor format, as discussed in Novikov et al. (2015), Cichocki et al. (2016, 2017), Calvi et al. (2019). This not only further reduces

Forward-pass

Figure 8.13: Illustration on the structure of a fast multi-graph tensor network.


Figure 8.14: Tensor network representation of the fast Multi-Graph Tensor Network (fMGTN) used in Example 109 and shown in Figure 8.13. The section encircled in dotted line denotes the multi-graph filtering operation for $M=2$ as in (26.21). The yellow region designates a tensorized dense layer weight matrix, represented in the Tensor-Train format (TTD). The input data used for the experiment is an order-3 tensor of FOREX data, with $J_{0}$ pricing features, $I_{1}$ past time-steps, and $I_{2}$ currencies. Note that as input data modes, this MGTN employs a time-domain graph filter of dimension $I_{1}$ and a currency-domain graph filter of dimension $I_{2}$.
the number of parameters, but also maintains compatibility with the inherent multi-modal nature of the problem.
Example 109: We considered the task of Foreign Exchange (FOREX) algorithmic trading in order to illustrate the possibilities enabled by the MGTN framework. A conceptual application of the combination of graphs, tensors and neural networks in this scenario is shown in Figure 8.14.

The MGTN setting is general enough to be applicable in a range of other domains, including social networks, communication networks, and cognitive neuroscience.

## 27

## Metro Traffic Modeling Through Graphs

With the rapid development of many economies, an increasing proportion of the world's population moving to cities, urban traffic congestion is becoming a serious issue. For example, underground traffic networks routinely undergo general maintenance, frequently exhibit signal failures and train derailments, and may even occasionally experience emergency measures because of various accidents. Such events ultimately require the closure of at least one station which may severely impact the service across the entire network. The economic costs of these transport delays to central London business are estimated to be $£ 1.2$ billion per year. Hence, appropriate and physically meaningful tools to understand, quantify, and plan for the resilience of these traffic networks to disruptions are much needed.

In this section, we demonstrate how the concept of vertex centrality of an adjacency matrix (for more detail, see Part I of this monograph) may be employed to identify those stations in the London underground network which have the greatest influence on the functionality of the traffic, and proceed, in an innovative way, to assess the impact of a station closure on service levels across the city. Such underground network vulnerability analysis offers the opportunity to analyze, optimize
and enhance the connectivity of the London underground network in a mathematically tractable and physically meaningful manner.

### 27.1 Traffic Centrality as a Graph-Theoretic Measure

The underground network can be modelled as an undirected $N$-vertex graph, denoted by $\mathcal{G}=\{\mathcal{V}, \mathcal{E}\}$, with $\mathcal{V}$ as the set of $N$ vertices (stations) and $\mathcal{E}$ the set of edges (underground lines) connecting the vertices (stations) (Dees et al., 2019). The connectivity of the network is encoded within the (undirected) adjacency matrix, $\mathbf{A} \in \mathbb{R}^{N \times N}$. Figure 8.1 illustrates the proposed graph model of the London underground network, with each vertex representing a station, and each edge designating the underground line connecting two adjacent stations. Notice that standard data analytics domains are ill-equipped to deal with this class of problems.


Figure 8.1: Graph model of the London underground network in Zones 1-3.


Figure 8.2: Betweenness centrality, designated by magenta-colored bars, of the London underground network in Zones 1-3. The largest betweenness centrality is observed for the following stations: Green Park, Earl's Court, Baker Street, Waterloo and Westminster.

We employ the following metrics to characterize the topology of the network and model its vulnerability.

- Betweenness centrality, which reflects the extent to which a given vertex lies in between pairs or groups of other vertices of the graph, and is given by

$$
\begin{equation*}
B_{n}=\sum_{k, m \in \mathcal{V}} \frac{\sigma(k, m \mid n)}{\sigma(k, m)} \tag{27.1}
\end{equation*}
$$

where $\sigma(k, m)$ denotes the number of shortest paths between vertices $k$ and $m$, and $\sigma(k, m \mid n)$ the number of those paths passing through vertex $n$ (Freeman, 1977). In terms of the actual metro traffic, this can also be interpreted as the extent to which a vertex is an intermediate in the communication over the network. Figure 8.2 shows that, as expected, the stations at the center of the city exhibit the largest betweenness centrality, and their disconnection would therefore severely impact the communication over the underground network.


Figure 8.3: Closeness vitality, designated in magenta bars, of the London underground network in Zones 1-3.

- Closeness vitality, which represents the change in the sum of distances between all vertex pairs after excluding the $n$th vertex (Brandes, 2005). Figure 8.3 shows that the stations located in the more remote areas of Zones 2-3 exhibit the largest closeness vitality measure. This is because their removal from the network would disconnect the stations located at the boundaries from the rest of the network.


### 27.2 Modeling Commuter Population from Net Passenger Flow

In this section, we employ graph theory to analyze the net passenger flow at all stations of the London underground network. In particular, we demonstrate that it is possible to infer the resident population surrounding each station based on the net passenger flow during the morning rush hour alone (Dees et al., 2019).

To derive the corresponding graph model, we employed the Fick law of diffusion (closely related to Newton's law of cooling discussed in Section 25.2 and Laplacian diffusion maps described in Part I) which
relates the diffusive flux to the concentration of a given vector field, under a steady state assumption. This model asserts that the flux flows from regions of high concentration (population) to regions of low concentration (population), with a magnitude that is proportional to the concentration gradient. Mathematically, the Fick law is given by

$$
\begin{equation*}
\mathbf{q}=-k \nabla \phi \tag{27.2}
\end{equation*}
$$

where

- $\mathbf{q}$ is the flux which measures the amount of substance per unit area per unit time $\left(\mathrm{mol} \mathrm{m}^{-2} \mathrm{~s}^{-1}\right)$;
- $k$ is the coefficient of diffusivity, with its value equal to area per unit time $\left(\mathrm{m}^{2} \mathrm{~s}^{-1}\right)$;
- $\phi$ represents the concentration $\left(\mathrm{mol} \mathrm{m}^{-3}\right)$.

In this way, we can model the passenger flows in the London underground network as a diffusion process, whereby during the morning rush hour the population mainly flows from concentrated residential areas to sparsely populated business districts. Therefore, the variables in our model are:

- $\mathbf{q} \in \mathbb{R}^{N}$, the net passenger flow vector, where the $i$ th entry represents the net passenger flow at the $i$ th station during the morning rush hour, that is

$$
\begin{align*}
q(i)= & (\text { passengers exiting station } i) \\
& -(\text { passengers entering station } i) \tag{27.3}
\end{align*}
$$

with its value equal to "passengers per station per unit time";

- $k=1$, the coefficient of diffusivity, with its dimension equal to "stations per unit time";
- $\phi \in \mathbb{R}^{N}$, the resident population in the area surrounding the station.

This model therefore suggests that, in the morning, the net passenger flow at the $i$ th station, $q(i)$, is proportional to the population difference between the areas surrounding a station $i$ and the adjacent stations $j$, that is

$$
\begin{align*}
q(i) & =-k \sum_{j} A_{i j}(\phi(i)-\phi(j)) \\
& =-k\left(\phi(i) \sum_{j} A_{i j}-\sum_{j} A_{i j} \phi(j)\right) \\
& =-k\left(\phi(i) D_{i i}-\sum_{j} A_{i j} \phi(j)\right) \\
& =-k \sum_{j}\left(\delta_{i j} D_{i i}-A_{i j}\right) \phi(j)=-k \sum_{j} L_{i j} \phi(j) . \tag{27.4}
\end{align*}
$$

When considering $N$ stations together, the above model assumes the matrix form

$$
\begin{equation*}
\mathbf{q}=-k \mathbf{L} \phi \tag{27.5}
\end{equation*}
$$

where $\mathbf{L}=(\mathbf{D}-\mathbf{A}) \in \mathbb{R}^{N \times N}$ is the Laplacian matrix of the graph model (see Part I of this monograph). For clarity, Figure 8.4 illustrates a signal within this diffusion model on a 2 -vertex path graph obeying the Fick law.

The data for the average daily net flow of passengers during the morning rush hour at each station in 2016 was obtained from Transport


Figure 8.4: Towards a graph representation of the London underground network. A simplified path graph with two stations surrounded by the respective populations, $\phi(1)$ and $\phi(2)$, exhibits the corresponding net fluxes, $q(1)$ and $q(2)$. Intuitively, stations surrounded by large populations experience net in-flows of passengers, whereas stations surrounded by low populations experience net out-flows of passengers.

Table 27.1: Daily average passenger flows during the morning rush hour per transportation Zone in London

| Zone | Entries | Exits | Net Outflow |
| :--- | ---: | ---: | ---: |
| 1 | 455,704 | 844,123 | 388,419 |
| 2 | 343,145 | 264,732 | $-78,413$ |
| 3 | 275,965 | 104,414 | $-171,551$ |
| $4-10$ | 206,408 | 72,152 | $-134,256$ |
| Total | $1,281,222$ | $1,285,421$ | 4,199 |

for London (TFL) (Transport for London n.d.), and is plotted as a signal on the graph model of the London underground in Figure 8.5. For illustration purposes, Table 27.1 shows the daily average net flow of passengers per transportation zone. As expected, Zone 1 is the only zone to exhibit a net outflow of passengers, while Zones $2-10$ show a net inflow of passengers. In particular, Zone 3 exhibits the largest inflow. In an ideal scenario, the total net outflow across Zones $1-10$ should sum up to 0 , however, the residual net outflow is attributed to passengers entering the underground network through other transport services not considered in our model, for example, rail services.

Moreover, Table 27.2 shows the average net flow of passengers for the top 5 stations with the greater net inflow and outflow. The stations with the greatest net outflow of passengers are located within the financial (Bank, Canary Wharf, Green Park) and commercial (Oxford Circus, Holborn) districts. In contrast, the greatest net inflow of passengers is attributed to the contribution from the railway stations located in residential areas.

To obtain an estimate of the resident population surrounding each station, we can simply rearrange the passenger flow in (27.5) to obtain

$$
\begin{equation*}
\hat{\phi}=-\frac{1}{k} \mathbf{L}^{+} \mathbf{q} \tag{27.6}
\end{equation*}
$$

where the symbol $(\cdot)^{+}$denotes the matrix pseudo-inverse operator. However, notice that the population vector can only be estimated up to a constant, hence the vector $\hat{\phi}$ actually quantifies the relative population

Table 27.2: Stations in the London underground system with greatest net passenger outflow and inflow during the morning rush hour

| Zone | Entries | Exits | Net Outflow |
| :--- | ---: | ---: | :---: |
| Bank | 17,577 | 69,972 | 52,395 |
| Canary Wharf | 8,850 | 56,256 | 47,406 |
| Oxford Circus | 3,005 | 44,891 | 41,886 |
| Green Park | 2,370 | 30,620 | 28,250 |
| Holborn | 1,599 | 25,294 | 23,695 |
| Finsbury Park | 20,773 | 8,070 | $-12,703$ |
| Canada Water | 31,815 | 14,862 | $-16,953$ |
| Brixton | 24,750 | 4,369 | $-20,381$ |
| Stratford | 43,473 | 22,360 | $-21,113$ |
| Waterloo | 61,129 | 22,861 | $-38,268$ |



Figure 8.5: Net passenger outflow during the morning rush hour within Zones 1-3 of the London underground network. The magenta bars designate a net outflow of passengers while the cyan bars designate a net inflow of passengers. Stations located within business districts exhibit the greatest net outflow of passengers, while stations located in residential areas, toward the boundaries of Zones $2-3$, exhibit the largest net inflow of passengers.


Figure 8.6: Population distribution implied by our graph model in (27.6), calculated from the net passenger outflow during the morning rush hour within Zones 1-3 of London underground system. As expected, business districts exhibit the lowest population density, while residential areas (Zones 2-3) exhibit the highest commuter population density.
between stations, whereby the station with the lowest estimated surrounding population takes the value of 0 . The so estimated resident population surrounding each station, based on the morning net passenger flow, is displayed in Figure 8.6 as a signal on a graph. Observe that these estimates are reasonable and physically meaningful since most of the resident population in London is concentrated toward the more remote areas of Zones 2-3, while business districts at the center of Zone 1 are sparsely populated in the evening.

## 28

## Portfolio Cuts

Investment returns naturally reside on irregular domains, however, standard multivariate portfolio optimization methods are agnostic to data structure. To this end, we investigate ways for the domain knowledge to be meaningfully incorporated into the analysis, by means of portfolio cuts. Such a graph-theoretic portfolio partitioning technique would allow the investor to devise robust and tractable asset allocation schemes, by virtue of a rigorous graph framework for considering smaller, computationally feasible, and economically meaningful clusters of assets, based on graph cuts. In turn, this makes it possible to fully utilize the covariance matrix of asset returns for constructing the portfolio, even without the requirement for its inversion.

Modern portfolio theory suggests an optimal strategy for minimizing the investment risk, which is based on the second-order moments of asset returns (Markowitz, 1952). The solution to this optimization task is referred to as the minimum-variance (MV) portfolio. Consider the vector, $\mathbf{r}(t) \in \mathbb{R}^{N}$, which contains the returns of $N$ assets at a time $t$, the $i$ th entry of which is given by

$$
\begin{equation*}
r_{t}(i)=\frac{p_{t}(i)-p_{t-1}(i)}{p_{t-1}(i)} \tag{28.1}
\end{equation*}
$$

where $p_{t}(i)$ denotes the value of the $i$ th asset at a time $t$. The MV portfolio asserts that the optimal vector of asset holdings, $\mathbf{w} \in \mathbb{R}^{N}$, is obtained through the following optimization problem

$$
\begin{equation*}
\min _{\mathbf{w}} \mathbf{w}^{T} \boldsymbol{\Sigma} \mathbf{w}, \quad \text { subject to } \mathbf{w}^{T} \mathbf{1}=1 \tag{28.2}
\end{equation*}
$$

where $\boldsymbol{\Sigma}=\operatorname{cov}\{\mathbf{r}\} \in \mathbb{R}^{N \times N}$ is the covariance matrix of returns, $\mathbf{1}=$ $[1, \ldots, 1]^{T}$, and the constraint, $\mathbf{w}^{T} \mathbf{1}=1$, enforces full investment of the capital. The optimal portfolio holdings (using the method of Lagrange multipliers) then become

$$
\begin{equation*}
\mathbf{w}=\frac{\boldsymbol{\Sigma}^{-1} \mathbf{1}}{\mathbf{1}^{T} \boldsymbol{\Sigma}^{-1} \mathbf{1}} \tag{28.3}
\end{equation*}
$$

It is important to highlight that the matrix inversion of $\boldsymbol{\Sigma}$ required in (28.3) may lead to significant errors for ill-conditioned matrices. These instability concerns have received substantial attention in recent years (Kolm et al., 2014), and alternative procedures have been proposed to promote robustness by either incorporating additional portfolio constraints (Clarke et al., 2002), introducing Bayesian priors (Black and Litterman, 1992) or improving the numerical stability of covariance matrix inversion (Ledoit and Wolf, 2003). A more recent approach has been to model assets using market graphs (Boginski et al., 2003), that is, based on graph-theoretic techniques. Intuitively, a universe of assets can be naturally modelled as a network of vertices on a graph, whereby an edge between two vertices (assets) designates both the existence of a link and the degree of similarity between assets (Simon, 1962).
Remark 112: A graph-theoretic perspective offers an interpretable explanation for the underperformance of minimum-variance optimization (MVO) techniques in practice. Namely, since the covariance matrix $\boldsymbol{\Sigma}$ is dense, standard multivariate models implicitly assume full connectivity of the graph, and are therefore not adequate to account for the structure inherent to real-world markets (Calkin and Lopez de Prado, 2014a,b, 2016). Moreover, it can be shown that the optimal holdings under the MVO framework are inversely proportional to the vertex centrality, thereby suggesting over-investing in assets with low centrality (Li et al., 2019b; Peralta and Zareei, 2016).

Intuitively, it would be highly desirable to remove unnecessary graph edges in order to more appropriately model the underlying structure between assets (graph vertices); this can be achieved through vertex clustering of the market graph (Boginski et al., 2003). Various portfolio diversification frameworks employ this technique to allocate capital within and across clusters of assets at multiple hierarchical levels. For instance, the hierarchical risk parity scheme (Calkin and Lopez de Prado, 2016) employs an inverse-variance weighting allocation which is based on the number of assets within each asset cluster. Similarly, the hierarchical clustering based asset allocation in Raffinot (2017) finds a diversified weighting by distributing capital equally among each of the cluster hierarchies.

Despite mathematical elegance and physical intuition, direct vertex clustering is an NP hard problem. Consequently, existing graph-theoretic portfolio constructions employ combinatorial optimization formulations (Boginski et al., 2003, 2005, 2006, 2014; Gunawardena et al., 2012, Kalyagin et al., 2014), which become computationally intractable for large graph systems. To alleviate this issue, we employ the minimum cut vertex clustering method to the graph of portfolio assets, to introduce the concept of portfolio cut (Scalzo et al., 2020). In this way, smaller graph partitions (cuts) can be evaluated quasi-optimally, using algebraic methods, and in an efficient and rigorous manner.

### 28.1 Structure of Market Graph

A universe of $N$ assets can be represented as a set of vertices on a market graph (Boginski et al., 2003), whereby the edge weight, $W_{m n}$, between vertices $m$ and $n$ is defined as the absolute correlation coefficient, $\left|\rho_{m n}\right|$, of their respective returns of assets $m$ and $n$, that is

$$
\begin{equation*}
W_{m n}=\frac{\left|\sigma_{m n}\right|}{\sqrt{\sigma_{m m} \sigma_{n n}}}=\left|\rho_{m n}\right| \tag{28.4}
\end{equation*}
$$

where $\sigma_{m n}=\operatorname{cov}\left\{r_{t}(m), r_{t}(n)\right\}$ is the covariance of returns between the assets $m$ and $n$. In this way, we have $W_{m n}=0$ if the assets $m$ and $n$ are statistically independent (not connected), and $W_{m n}>0$ if they are statistically dependent (connected on a graph). Note that the resulting weight matrix is symmetric, $\mathbf{W}^{T}=\mathbf{W}$.

### 28.2 Minimum Cut Based Vertex Clustering

Vertex clustering aims to group together vertices from the asset universe, $\mathcal{V}$, into multiple disjoint clusters, $\mathcal{V}_{i}$. For a market graph, assets which are grouped into a cluster, $\mathcal{V}_{i}$, are expected to exhibit a larger degree of mutual within-cluster statistical dependency than with the assets in other clusters, $\mathcal{V}_{j}, j \neq i$. The most popular classical graph cut methods are based on finding the minimum set of edges whose removal would disconnect a graph in some "optimal" sense; this is referred to as minimum cut based clustering (Schaeffer, 2007a) (see Part I of this monograph for a comprehensive review of the minimum graph cut problem and other graph spectral clustering methods).

Consider an $N$-vertex market graph, $\mathcal{G}=\{\mathcal{V}, \mathcal{E}\}$, which is grouped into $K=2$ disjoint subsets of vertices, $\mathcal{V}_{1} \subset \mathcal{V}$ and $\mathcal{V}_{2} \subset \mathcal{V}$, with $\mathcal{V}_{1} \cup \mathcal{V}_{2}=\mathcal{V}$ and $\mathcal{V}_{1} \cap \mathcal{V}_{2}=\emptyset$. A cut of this graph, for the given clusters, $\mathcal{V}_{1}$ and $\mathcal{V}_{2}$, is equal to a sum of all weights that correspond to the edges which connect the vertices between the subsets, $\mathcal{V}_{1}$ and $\mathcal{V}_{2}$, that is

$$
\begin{equation*}
\operatorname{Cut}\left(\mathcal{V}_{1}, \mathcal{V}_{2}\right)=\sum_{m \in \mathcal{V}_{1}} \sum_{n \in \mathcal{V}_{2}} W_{m n} . \tag{28.5}
\end{equation*}
$$

A cut which exhibits the minimum value of the sum of weights between the disjoint subsets, $\mathcal{V}_{1}$ and $\mathcal{V}_{2}$, considering all possible divisions of the set of vertices, $\mathcal{V}$, is referred to as the minimum cut.

Finding the minimum cut in (28.5) is a relatively easy problem and can be solved efficiently (Stoer and Wagner, 1997). However, in practice, this minimum cut formation in (28.5) often leads to unsatisfactory performance (Von Luxburg, 2007). For example, assume that all the weights are positive and that we allow an empty set as a cluster; upon taking one cluster as an empty set and another cluster as a whole graph, that would yield the minimum cut, which is 0 . This result is not a reasonable partition we desire. To overcome this problem, we may "balance" the sizes of cluster and cut, i.e., each cluster should be reasonably large, while at the same time the cut itself should be minimized. Instead of using (28.5), two balanced cuts are often used, Ratio Cut (Hagen and Kahng, 1992a) and Normalized Cut (Shi and Malik, 2000), where a balancing term is incorporated into the cut in (28.5).

Within graph cuts, a number of optimization approaches may be employed to enforce some desired properties on graph clusters:
(i) Ratio cut. The value of $\operatorname{Cut}\left(\mathcal{V}_{1}, \mathcal{V}_{2}\right)$ is normalized by an additional term to enforce the subsets, $\mathcal{V}_{1}$ and $\mathcal{V}_{2}$, to be simultaneously as large as possible. The ratio cut formulation is given by Hagen and Kahng (1992a)

$$
\begin{equation*}
\operatorname{Cut} R\left(\mathcal{V}_{1}, \mathcal{V}_{2}\right)=\left(\frac{1}{N_{1}}+\frac{1}{N_{2}}\right) \sum_{m \in \mathcal{V}_{1}} \sum_{n \in \mathcal{V}_{2}} W_{m n} \tag{28.6}
\end{equation*}
$$

where $N_{1}$ and $N_{2}$ are the respective numbers of vertices in the sets $\mathcal{V}_{1}$ and $\mathcal{V}_{2}$. Since $N_{1}+N_{2}=N$, the term $\frac{1}{N_{1}}+\frac{1}{N_{2}}$ reaches its minimum for $N_{1}=N_{2}=\frac{N}{2}$.
(ii) Volume normalized cut. Since the vertex weights are involved when designing the size of subsets $\mathcal{V}_{1}$ and $\mathcal{V}_{2}$, then by defining the volumes of these sets as $V_{1}=\sum_{n \in \mathcal{V}_{1}} D_{n n}$ and $V_{2}=\sum_{n \in \mathcal{V}_{2}} D_{n n}$, we arrive at the volume normalized cut (Shi and Malik, 2000) (see also Part I)

$$
\begin{equation*}
\operatorname{Cut} N\left(\mathcal{V}_{1}, \mathcal{V}_{2}\right)=\left(\frac{1}{V_{1}}+\frac{1}{V_{2}}\right) \sum_{m \in \mathcal{V}_{1}} \sum_{n \in \mathcal{V}_{2}} W_{m n} \tag{28.7}
\end{equation*}
$$

Since $V_{1}+V_{2}=V$, the term $\frac{1}{V_{1}}+\frac{1}{V_{2}}$ reaches its minimum for $V_{1}=V_{2}=\frac{V}{2}$. Notice that vertices with a higher degree, $D_{n n}$, are considered as structurally more important than those with lower degrees. In turn, for market graphs, assets with a higher average statistical dependence to other assets are considered as more central.

Remark 113: It is important to note that clustering results based on the two above graph cut forms are different. While the ratio cut in (i) favors the clustering into subsets with (almost) equal number of vertices, the volume normalized cut in (ii) favors subsets with (almost) equal volumes, that is, subgraphs with vertices exhibiting (almost) equal average statistical dependence to the other vertices.
Remark 114: Although the optimization algorithm for the cut in (28.5) is simple, by introducing the balancing terms into this cut, the task
of finding the minimum of the objective functions in (28.6) and (28.7) becomes NP hard (Von Luxburg, 2007; Wagner and Wagner, 1993). However, if we relax the problem from a discrete valued to a real valued one, then this boils down to the eigenproblem of graph Laplacian, which is considered next.

### 28.3 Spectral Bisection Based Minimum Cut

To overcome the computational burden of finding the ratio cut, we may opt for an approximative spectral solution which clusters vertices using the eigenvectors of the graph Laplacian, $\mathbf{L}$. The algorithm employs the second (Fiedler, 1973a) eigenvector of the graph Laplacian, $\mathbf{u}_{2} \in \mathbb{R}^{N}$, to yield quasi-optimal vertex clustering on a graph. Despite its simplicity, the algorithm is typically accurate and gives a good approximation to the minimum cut (Ng et al., 2002; Spielman and Teng, 2007a).

To relate the problem of the minimum cut in (28.6) and (28.7) to that of eigenanalysis of graph Laplacian, we employ an indicator vector, denoted by $\mathbf{x} \in \mathbb{R}^{N}$ (Stanković et al., 2019a), for which the elements take sub-graph-wise constant values within each disjoint subset (cluster) of vertices, with these constants taking different values for different clusters of vertices. In other words, the elements of $\mathbf{x}$ uniquely reflect the assumed cut of the graph into disjoint subsets $\mathcal{V}_{1}, \mathcal{V}_{2} \subset \mathcal{V}$.

For a general graph, we consider two possible solutions for the indicator vector, $\mathbf{x}$, that satisfy the subset-wise constant form:
(i) Ratio cut. It can be shown that if the indicator vector is defined as (see Part I of this monograph)

$$
x(n)= \begin{cases}\frac{1}{N_{1}}, & \text { for } n \in \mathcal{V}_{1}  \tag{28.8}\\ -\frac{1}{N_{2}}, & \text { for } n \in \mathcal{V}_{2}\end{cases}
$$

then the ratio cut, $\operatorname{Cut} R\left(\mathcal{V}_{1}, \mathcal{V}_{2}\right)$ in $(28.6)$, is equal to the Rayleigh quotient of $\mathbf{L}$ and $\mathbf{x}$, that is

$$
\begin{equation*}
\operatorname{Cut} R\left(\mathcal{V}_{1}, \mathcal{V}_{2}\right)=\frac{\mathbf{x}^{T} \mathbf{L} \mathbf{x}}{\mathbf{x}^{T} \mathbf{x}} \tag{28.9}
\end{equation*}
$$

Therefore, the indicator vector, $\mathbf{x}$, which minimizes the ratio cut also minimizes (28.9). From the indicator vector, we see

$$
\begin{equation*}
\sum_{n \in V} x(n)=\sum_{n \in V_{1}} x(n)+\sum_{n \in V 2} x(n)=N_{1} \times \frac{1}{N_{1}}-N_{2} \times \frac{1}{N_{2}}=0 . \tag{28.10}
\end{equation*}
$$

In other words, we can say that the vector $\mathbf{x}$ is orthogonal to $\mathbf{1}$. Moreover, we can see that the objective function in (28.9) is invariant of the scale of $\mathbf{x}$. From this discussion, we can relax the problem of the objective function in (28.6) through the constraints, as

$$
\begin{equation*}
\min _{\mathbf{x}} \mathbf{x}^{T} \mathbf{L} \mathbf{x}, \quad \text { subject to } \mathbf{x}^{T} \mathbf{x}=1, \quad \text { and } \quad \mathbf{x}^{T} \mathbf{1}=0 \tag{28.11}
\end{equation*}
$$

Given that the considered graph is undirected and therefore $\mathbf{L}$ is symmetric, the first eigenvector of the graph Laplacian is constant (proportional to vector $\mathbf{1}, \mathbf{u}_{0}=\mathbf{1} / \sqrt{N}$ ) and the associated first eigenvalue is $\lambda_{0}=0$. Therefore, by the Rayleigh-Ritz theorem, the solution to the objective function in (28.11) is given by the second eigenvector of the graph Laplacian, $\mathbf{L}$, obtained as

$$
\begin{equation*}
\mathbf{L x}=\lambda_{1} \mathbf{x} \tag{28.12}
\end{equation*}
$$

with the second eigenvalue, $\lambda_{k}=\lambda_{1}$.
(ii) Volume normalized cut. Similarly, by defining $\mathbf{x}$ as

$$
x(n)= \begin{cases}\frac{1}{V_{1}}, & \text { for } n \in \mathcal{V}_{1},  \tag{28.13}\\ -\frac{1}{V_{2}}, & \text { for } n \in \mathcal{V}_{2},\end{cases}
$$

the volume normalized cut, $\operatorname{Cut} N\left(\mathcal{V}_{1}, \mathcal{V}_{2}\right)$ in (28.7), takes the form of a generalized Rayleigh quotient of $\mathbf{L}$, given by (see again Part I)

$$
\begin{equation*}
\operatorname{CutN}\left(\mathcal{V}_{1}, \mathcal{V}_{2}\right)=\frac{\mathbf{x}^{T} \mathbf{L} \mathbf{x}}{\mathbf{x}^{T} \mathbf{D} \mathbf{x}} \tag{28.14}
\end{equation*}
$$

Similarly to the ratio cut, we see from the indicator vector that

$$
\begin{align*}
\sum_{n \in V} d(n) x(n) & =\sum_{n \in V_{1}} d(n) x(n)+\sum_{n \in V 2} d(n) x(n) \\
& =\sum_{n \in V_{1}} d(n) \times \frac{1}{V_{1}}-\sum_{n \in V_{2}} d(n) \times \frac{1}{V_{2}} \\
& =V_{1} \times \frac{1}{V_{1}}-V_{2} \times \frac{1}{V_{2}}=0 \tag{28.15}
\end{align*}
$$

which yields $(\mathbf{D} \mathbf{x})^{T} \mathbf{1}=0$. Also, the objective function is invariant to the scale of $\mathbf{x}$. Therefore, we can formulate the optimization problem from the objective function (28.14) as

$$
\begin{equation*}
\min _{\mathbf{x}} \mathbf{x}^{T} \mathbf{L} \mathbf{x}, \quad \text { subject to } \mathbf{x}^{T} \mathbf{D} \mathbf{x}=1, \quad \text { and } \quad(\mathbf{D} \mathbf{x})^{T} \mathbf{1}=0 \tag{28.16}
\end{equation*}
$$

The solution is given by the second generalized eigenvector of the generalized eigenproblem of the graph Laplacian as

$$
\begin{equation*}
\mathbf{L x}=\lambda_{1} \mathbf{D} \mathbf{x} \tag{28.17}
\end{equation*}
$$

since $\mathbf{D}^{-1 / 2} \mathbf{1}$ is the first generalized eigenvector of graph Laplacian.

For the spectral solutions above, the membership of a vertex, $n$, to either the subset $\mathcal{V}_{1}$ or $\mathcal{V}_{2}$ is uniquely defined by the sign of the indicator vector, $\mathbf{x}=\mathbf{u}_{1}$, that is

$$
\operatorname{sign}(x(n))= \begin{cases}1, & \text { for } n \in \mathcal{V}_{1}  \tag{28.18}\\ -1, & \text { for } n \in \mathcal{V}_{2}\end{cases}
$$

Notice that a scaling of $\mathbf{x}$ by any constant would not influence the solution for clustering into the subsets $\mathcal{V}_{1}$ or $\mathcal{V}_{2}$.

### 28.4 Repeated Portfolio Cuts

Although the above analysis has focused on the case with $K=2$ disjoint sub-graphs, it can be straightforwardly generalized to $K \geq 2$ disjoint sub-graphs through the method of repeated bisection.


Figure 8.1: Graph cut based asset allocation strategies. (a) Hierarchical graph structure resulting from $K=4$ portfolio cuts. (b) A graph tree based on the $\frac{1}{2^{K_{i}}}$ scheme. (c) A graph tree based on the $\frac{1}{K+1}$ scheme.

A single application of the portfolio cut on the market graph, $\mathcal{G}$, produces two disjoint sub-graphs, $\mathcal{G}_{1}$ and $\mathcal{G}_{2}$, as illustrated in Figure 8.1(a). Notice that in this way we construct a hierarchical binary tree structure, whereby the union of the leaves of the network is equal to the original market graph, $\mathcal{G}$. We can then perform a subsequent portfolio cut operation on one or both of the leaves based on some suitable criterion (e.g., the leaf with the greatest number of vertices or volume). Therefore, ( $K+1$ ) disjoint sub-graphs (leaves) can be obtained by performing the portfolio cut procedure $K$ times (Scalzo et al., 2020).
Example 110: Figure 8.1(a) illustrates the hierarchical structure resulting from $K=4$ portfolio cuts of a market graph, $\mathcal{G}$. The leaves
of the resulting binary tree are denoted by $\left\{\mathcal{G}_{3}, \mathcal{G}_{4}, \mathcal{G}_{5}, \mathcal{G}_{7}, \mathcal{G}_{8}\right\}$ (in red), whereby the number of disjoint sub-graphs is equal to $(K+1)=5$. Notice that the union of the leaves amounts to the original graph, i.e., $\mathcal{G}_{3} \cup \mathcal{G}_{4} \cup \mathcal{G}_{5} \cup \mathcal{G}_{7} \cup \mathcal{G}_{8}=\mathcal{G}$.

### 28.5 Graph Asset Allocation Schemes

We next elaborate upon some intuitive asset allocation strategies, inspired by the work in Calkin and Lopez de Prado (2016) and Raffinot (2017), which naturally builds upon the portfolio cut. The aim is to determine a diversified weighting scheme by distributing capital among the disjoint clusters (leaves) so that highly correlated assets within a given cluster receive the same total allocation, thereby being treated as a single investment entity.

Upon denoting the portion of the total capital allocated to a cluster $\mathcal{G}_{i}$ by $w_{i}$, we consider two simple asset allocation schemes:
(AS1) $w_{i}=\frac{1}{2^{K_{i}}}$, where $K_{i}$ is the number of portfolio cuts required to obtain a sub-graph $\mathcal{G}_{i}$;
(AS2) $w_{i}=\frac{1}{K+1}$, where $(K+1)$ is the number of disjoint sub-graphs.
Remark 115: An equally-weighted asset allocation strategy may now be employed within each cluster, i.e., every asset within the $i$ th cluster, $\mathcal{G}_{i}$, will receive a weighting equal to $\frac{w_{i}}{N_{i}}$.
Remark 116: The weighting scheme in AS1 above is closely related to the strategy proposed in Raffinot (2017), while the scheme in AS2 is inspired by the generic equal-weighted (EW) allocation scheme (De Miguel and R. Uppal, 2009). These schemes are convenient in that they require no assumptions regarding the across-cluster statistical dependence. In addition, unlike the EW scheme, they implicitly consider the inherent market risks (asset correlation) by virtue of the portfolio cut formulation, which is based on the eigenanalysis of the market graph Laplacian, $\mathbf{L}$.
Example 111: Figures 8.1(b) and (c) illustrate respectively the asset allocation schemes in AS1 and AS2 for $K=4$ portfolio cuts, based on
the market graph partitioning in Figure 8.1(a). Notice that the weights associated to the disjoint sub-graphs (leaves in red) sum up to unity.

### 28.6 Numerical Example

The performance of the portfolio cuts and the associated graph-theoretic asset allocation schemes was investigated using historical price data comprising of the 100 most liquid stocks in the $\mathrm{S} \& \mathrm{P} 500$ index, based on the average trading volume, in the period 2014-01-01 to 2018-01-01. The data was split into: (i) the in-sample dataset (2014-01-01 to 2015-12-31) which was used to estimate the asset correlation matrix and to compute the portfolio cuts; and (ii) the out-sample dataset (2016-01-01 to 2018-01-01), used to objectively quantify the profitability of the asset allocation strategies (Scalzo et al., 2020).

Figure 8.2 displays the $K$ th iterations, for $K=1,2,10$, of the normalized portfolio cut in (28.9), applied to the original 100-vertex


Figure 8.2: Visualization of the 100 -vertex market graph connectivity for the 100 most liquid stocks in S\&P 500 index, and its partitions into disjoint sub-graphs (separated by dashed grey lines). The edges (blue lines) were calculated based on the correlation between assets. (a) Fully connected market graph with 5050 edges. (b) Partitioned graph after $K=1$ portfolio cuts (CutN), with 2746 edges. (c) Partitioned graph after $K=2$ portfolio cuts (CutN), with 1731 edges. (d) Partitioned graph after $K=10$ portfolio cuts (CutN), with 575 edges. Notice that the number of edges required to model the market graph is significantly reduced with each subsequent portfolio cut, since $\sum_{i=1}^{K+1} \frac{1}{2}\left(N_{i}^{2}+N_{i}\right)<\frac{1}{2}\left(N^{2}+N\right), \forall K>0$.

(a) Evolution of wealth for both the traditional (EW and MV) and graph-theoretic asset allocation strategies, based on ( $K=10$ ) portfolio cuts.

| Cut Method | Allocation | $K=1$ | $K=2$ | $K=3$ | $K=4$ | $K=5$ | $K=10$ |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| CutN | AS1 | 1.82 | 1.80 | 1.80 | 1.93 | 1.96 | $\mathbf{1 . 9 8}$ |
| CutN | AS2 | 1.82 | 1.81 | 1.94 | 2.03 | 1.95 | $\mathbf{2 . 0 5}$ |
| CutR | AS1 | 1.93 | 2.01 | 2.08 | 2.23 | 2.22 | $\mathbf{2 . 2 5}$ |
| CutR | AS2 | 1.93 | 2.04 | 2.17 | $\mathbf{2 . 6 5}$ | 2.51 | 2.48 |

(b) Sharpe ratios attained for a varying number of portfolio cuts $K$.

Figure 8.3: Out-sample performance of the graph cut based asset allocation strategies. Notice that the Sharpe ratio typically improves with each subsequent portfolio cut. The traditional portfolio strategies, EW and MV, attained the respective Sharpe ratios of $\mathrm{SR}_{\mathrm{EW}}=1.85$ and $\mathrm{SR}_{\mathrm{MV}}=1.6$.
market graph obtained from the in-sample data set. Next, for the outsample dataset, graph representations of the portfolio, for the number of cuts $K$ varying in the range $[1,10]$, were employed to assess the performance of the asset allocation schemes described in Section 28.5. The standard equally-weighted (EW) and minimum-variance (MV) portfolios were also simulated for comparison purposes, with the results displayed in Figure 8.3.

Conforming with the findings in Calkin and Lopez de Prado (2016) and Raffinot (2017), the proposed graph asset allocation schemes consistently delivered lower out-sample variance than the standard EW and MV portfolios, thereby attaining a higher Sharpe ratio, i.e., the ratio of the mean to the standard deviation of portfolio returns. This verifies that the removal of possibly spurious statistical dependencies in the "raw" format, through portfolio cuts, allows for robust and flexible portfolio constructions.

Such an approach enables the creation of graph-theoretic capital allocation schemes, based on measures of connectivity which are inherent to the portfolio cut formulation. In addition, the proposed portfolio construction employs full information contained in the asset covariance matrix, and without requiring its inversion, even in the critical cases of limited data length or singular covariance matrices.

## 29

## Conclusion

In many modern applications, graph topology is not known a priori and hence its determination becomes part of the problem definition, rather than serving as prior knowledge to aid solution. To perform simultaneous estimation of both data on a graph and the underlying graph topology, without loss of generality we assume that the vertices (their number, location, etc.) are given, while the edges and their associated weights form part of the solution to the problem under consideration. Three possible scenarios for the estimation of graph edges from the data observed on a graph have been considered. Namely, in various sensor network sensing setups (temperature, pressure, transportation) the locations of the sensor positions (vertices) may be known while the vertex distances convey physical meaning about data and inter-sensor dependence and thus may be employed for weight determination. Another possibility is to employ the covariance and precision matrices, which are commonly used as data similarity metrics and are thus a natural choice of a metric for learning graph topology from data. The third scenario are graphs for which the relations among the sensing positions are physically well defined, such as in electric circuits, power
networks, linear heat transfer, social and computer networks, and springmass systems. Next, the problem of simulation of graph signals has been addressed and a detailed derivation and elaboration of sparsity structure promoting optimization approaches, such as the LASSO and graph-based version of LASSO (GLASSO), has been given. The inherent connection between graphs and deep neural networks (DNNs) has been further addressed, and the concepts of graph neural networks (GNN) and graph convolutional neural networks (GCN) have been introduced. It has been shown that the diffusion process on graphs underpins the operation of GNNs. The enormous potential of the combination of the universal function approximation property of neural networks with the elegance and generality of graph models has been demonstrated through the concepts of recurrent GNNs, spatial GNNs, spectral GNNs, together with the interpretation of graph signal filtering as a diffusion process in a "neural network" language. The advantages of these concepts have been illustrated over the paradigms of semi-supervised learning and label propagation, while the use of GNNs in graph link prediction has been addressed based on an innovative but natural combination of characteristic functions and generative adversarial nets, referred to as reciprocal adversarial learning via characteristic functions (RCF-GAN). Furthermore, the application of graphs in Big Data scenarios has been demonstrated through their link with tensors, and tensor factorizations. This is particularly significant, as multidimensional graphs are common in practice, but are inadequately modelled through their imbalanced and "flat view" adjacency matrices. To this end, we show that multilinear algebra, whereby multidimensional graphs are modelled via the corresponding adjacency tensor, is a natural choice to discover intrinsic relations in such multidimensional data. This has led to the concept of multi-graph tensor network (MGNT), which serves as a general framework for neural network learning in big data settings and on multiple irregular domains. Finally, innovative and comprehensively elaborated case studies have been given in support of the concepts, ranging from portfolio cuts in finance to the modelling of vulnerability of stations in underground metro traffic.

Appendices

## A

## Power Method for Eigenanalysis

Computational complexity of the eigenvalue and eigenvector calculation for a symmetric matrix is of the order of $\mathcal{O}\left(N^{3}\right)$, which is computationally prohibitive for very large graphs, especially when only a few the smoothest eigenvectors are needed, like in spectral graph clustering. To mitigate this computational bottleneck, an efficient iterative approach, called the Power Method, may be employed.

Consider the normalized weight matrix,

$$
\mathbf{W}_{N}=\mathbf{D}^{-1 / 2} \mathbf{W D}^{-1 / 2}
$$

and assume that the eigenvalues of $\mathbf{W}_{N}$ are $\left|\lambda_{0}\right|>\left|\lambda_{1}\right|>\cdots>\left|\lambda_{M-1}\right|$, with the corresponding eigenvectors, $\mathbf{u}_{1}, \mathbf{u}_{2}, \ldots, \mathbf{u}_{M-1}$. Consider also an arbitrary linear combination of the eigenvectors, $\mathbf{u}_{n}$, through the coefficients $\alpha_{n}$,

$$
\mathbf{x}=\alpha_{1} \mathbf{u}_{1}+\alpha_{2} \mathbf{u}_{2}+\cdots+\alpha_{M-1} \mathbf{u}_{M-1} .
$$

A further multiplication of the vector $\mathbf{x}$ by the normalized weight matrix, $\mathbf{W}_{N}$, results in

$$
\begin{aligned}
\mathbf{W}_{N} \mathbf{x} & =\alpha_{1} \mathbf{W}_{N} \mathbf{u}_{1}+\alpha_{2} \mathbf{W}_{N} \mathbf{u}_{2}+\cdots+\alpha_{M-1} \mathbf{W}_{N} \mathbf{u}_{M-1} \\
& =\alpha_{1} \lambda_{1} \mathbf{u}_{1}+\alpha_{2} \lambda_{2} \mathbf{u}_{2}+\cdots+\alpha_{M-1} \lambda_{M-1} \mathbf{u}_{M-1} .
\end{aligned}
$$

A repetition of this multiplication $k$ times yields

$$
\begin{aligned}
\mathbf{W}_{N}^{k} \mathbf{x} & =\alpha_{1} \lambda_{1}^{k} \mathbf{u}_{1}+\alpha_{2} \lambda_{2}^{k} \mathbf{u}_{2}+\cdots+\alpha_{M-1} \lambda_{M-1}^{k} \mathbf{u}_{M-1} \\
& =\alpha_{1} \lambda_{1}^{k}\left(\mathbf{u}_{1}+\alpha_{2} \frac{\lambda_{2}^{k}}{\lambda_{1}^{k}} \mathbf{u}_{2}+\cdots+\alpha_{M-1} \frac{\lambda_{M-1}^{k}}{\lambda_{1}^{k}} \mathbf{u}_{M-1}\right) \\
& \approx \alpha_{1} \lambda_{1}^{k} \mathbf{u}_{1} .
\end{aligned}
$$

In other words, we have just calculated the first eigenvector of $\mathbf{W}_{N}$, given by

$$
\mathbf{u}_{1}=\mathbf{W}_{N}^{k} \mathbf{x} /\left\|\mathbf{W}_{N}^{k} \mathbf{x}\right\|_{2}
$$

which are achieved through only matrix products of $\mathbf{W}_{N}$ and $\mathbf{x}$ (Tammen et al., 2018; Trevisan, 2013). The convergence of this procedure depends on the eigenvalue ratio $\lambda_{2} / \lambda_{1}$, and requires that $\alpha_{1}$ is not close to 0 . Note that $\mathbf{W}_{N}$ is a highly sparse matrix, which significantly reduces the calculation complexity.

After the eigenvector $\mathbf{u}_{1}$ is obtained, the corresponding eigenvalue can be calculated as its smoothing index, $\lambda_{1}=\mathbf{u}_{1}^{T} \mathbf{W}_{N} \mathbf{u}_{1}$.

After calculating $\mathbf{u}_{1}$ and $\lambda_{1}$, we can remove their contribution from the normalized weight matrix, $\mathbf{W}_{N}$, through deflation, as $\mathbf{W}_{N} \leftarrow$ $\mathbf{W}_{N}-\lambda_{1} \mathbf{u}_{1} \mathbf{u}_{1}^{T}$, and then continue to calculate the next largest eigenvalue and its eigenvector, $\lambda_{2}$ and $\mathbf{u}_{2}$. This procedure can be repeated iteratively until the desired number of eigenvectors is found.

The relation of the normalized weight matrix, $\mathbf{W}_{N}$, with the normalized graph Laplacian, $\mathbf{L}_{N}$, is given by

$$
\mathbf{L}_{N}=\mathbf{I}-\mathbf{W}_{N},
$$

while the relation between the eigenvalues and eigenvectors of $\mathbf{L}$ and $\mathbf{W}_{N}$ follows from $\mathbf{W}_{N}=\mathbf{U}^{T} \boldsymbol{\Lambda} \mathbf{U}$, to yield

$$
\mathbf{L}_{N}=\mathbf{I}-\mathbf{U}^{T} \boldsymbol{\Lambda} \mathbf{U}=\mathbf{U}^{T}(\mathbf{I}-\boldsymbol{\Lambda}) \mathbf{U} .
$$

The eigenvalues of $\mathbf{L}_{N}$ and $\mathbf{W}_{N}$ are therefore related as $\lambda_{n}^{(L)}=1-\lambda_{n}$, and share the same corresponding eigenvectors, $\mathbf{u}_{n}$, of the normalized graph Laplacian and the normalized weight matrix. This means that $\lambda_{1}=1$ corresponds to $\lambda_{0}^{(L)}=0$ and that the second largest eigenvalue of $\mathbf{W}_{N}$ produces the Fiedler vector of the normalized Laplacian.

Note that the second largest eigenvalue of $\mathbf{W}_{N}$ is not necessarily $\lambda_{2}$ since the eigenvalues of $\mathbf{W}_{N}$ can be negative.
Example 112: The weight matrix $\mathbf{W}$ from (2.4) is normalized by the degree matrix from (2.6) to arrive at $\mathbf{W}_{N}=\mathbf{D}^{-1 / 2} \mathbf{W} \mathbf{D}^{-1 / 2}$. The power algorithm is then used to calculate the four largest eigenvalues and the corresponding eigenvectors of $\mathbf{W}_{N}$ in 200 iterations, to give $\lambda_{n} \in$ $\{1.0000,-0.7241,-0.6795,0.6679\}$. These are very close to the four exact largest eigenvalues of $\mathbf{W}_{N}, \lambda_{n} \in\{1.0000,-0.7241,-0.6796,0.6677\}$. Note that the Fiedler vector of the normalized graph Laplacian is associated with $\lambda_{4}=0.6679$ as it corresponds to the second largest eigenvalue of $\mathbf{W}_{N}$, when the eigenvalue signs are accounted for. Even when calculated using the approximative power method, the Fiedler vector is close to its exact value, as shown in Figure 4.8(d), with the maximum relative error of its elements being 0.016 .

Notice that it is possible to calculate the Fiedler vector of a graph Laplacian even without using the weight matrix. Consider a graph whose eigenvalues of the Laplacian are $\lambda_{0}=0>\lambda_{1}>\lambda_{2}>\cdots>$ $\lambda_{N-1}$, where $\lambda_{1}$ corresponds to the largest value of the sequence $\lambda_{0}=$ $0,1 / \lambda_{1}, 1 / \lambda_{2}, \ldots, 1 / \lambda_{N-1}$. These are also the eigenvalues of the pseudoinverse of the graph Laplacian, $\mathbf{L}^{+}=\operatorname{pinv}(\mathbf{L})$. Now, since the pseudoinverse of the graph Laplacian, $\mathbf{L}^{+}$, and the graph Laplacian, $\mathbf{L}$, have the same eigenvectors, we may apply the power method to the pseudoinverse of the graph Laplacian, $\mathbf{L}^{+}$, and the eigenvector corresponding to the largest eigenvalue is the Fiedler vector.

Algorithm 8. Power Method for eigenanalysis.

## Input:

- Normalized weight matrix $\mathbf{W}_{N}$
- Number of iterations, It
- Number of the desired largest eigenvectors, $M$

1: for $m=1$ to $M$ do
2: $\mathbf{u}_{m} \in\{-1,1\}^{M}$, drawn randomly (uniformly)
3: for $i=1$ to $I t$ do
4: $\mathbf{u}_{m} \leftarrow \mathbf{W}_{N} \mathbf{u}_{m} /\left\|\mathbf{W}_{N} \mathbf{u}_{m}\right\|_{2}$
5: $\lambda_{m} \leftarrow \mathbf{u}_{m}^{H} \mathbf{W}_{N} \mathbf{u}_{m}$
6: end do
7: $\mathbf{W}_{N} \leftarrow \mathbf{W}_{N}-\lambda_{m} \mathbf{u}_{m} \mathbf{u}_{m}^{H}$
8: end do

## Output:

- Largest $M$ eigenvalues $\left|\lambda_{0}\right|>\left|\lambda_{1}\right|>\cdots>\left|\lambda_{M-1}\right|$ and the corresponding eigenvectors $\mathbf{u}_{1}, \ldots, \mathbf{u}_{M-1}$
- Fiedler vector of the normalized graph Laplacian is the eigenvector $\mathbf{u}_{n_{1}}$ of the second largest eigenvalue, $\lambda_{n_{1}}, \lambda_{0}=1>$ $\lambda_{n_{1}}>\cdots>\lambda_{n_{M-1}}$.


## B

## Algorithm for Graph Laplacian Eigenmaps

The algorithm for the Laplacian eigenmap and spectral clustering based on the eigenvectors of the graph Laplacian, the generalized eigenvectors of the graph Laplacian, and the eigenvectors of the normalized Laplacian, is given in the pseudo-code form in Algorithm 9.

Comments on the Algorithm: For the normalized Laplacian, Line 2 should be replaced by $\mathbf{L} \leftarrow \mathbf{I}-\mathbf{D}^{-1 / 2} \mathbf{W} \mathbf{D}^{-1 / 2}$, while for the generalized eigenvectors Line 3 should be replaced by $[\mathbf{U}, \boldsymbol{\Lambda}] \leftarrow \operatorname{eig}(\mathbf{L}, \mathbf{D})$, see also Table 4.1. The indicator values of vertex positions in the output graph are: $P=0$, for the original vertex space, and $P=1$, for the spectral vertex space. The indicator of mapping is: $M a p=1$, for the commute time mapping (matrix $\bar{\Lambda}$ is obtained from $\boldsymbol{\Lambda}$, by omitting the trivial element $\lambda_{0}=0$ ), and $M a p=2$, for the diffusion mapping (in this case the generalized eigenvectors must be used in Line $3,[\mathbf{U}, \boldsymbol{\Lambda}] \leftarrow \operatorname{eig}(\mathbf{L}, \mathbf{D})$ and the diffusion step $t$ should be given as an additional input parameter), otherwise $M a p=0$. The indicator of the eigenvectors normalization is: $S=0$, for the case without normalization, $S=1$, for two-norm normalization, $S=2$, for the case of binary normalization, $S=3$, for binary normalization with the mean as a reference, and $S=4$, for marginal normalization. The indicator of

Algorithm 9. Graph Laplacian Based Eigenmaps.

## Input:

- Vertex $\mathcal{V}=\{0,1, \ldots, N-1\}$ positions, rows of $\mathbf{X}$
- Weight matrix $\mathbf{W}$, with elements $W_{m n}$
- Laplacian eigenmap dimensionality, $M$
- Position, mapping, normalization, and coloring indicators P, Map, S,C

1: $\mathbf{D} \leftarrow \operatorname{diag}\left(D_{n n}=\sum_{m=0}^{N-1} W_{m n}, n=0,1, \ldots, N-1\right)$
2: $\mathbf{L} \leftarrow \mathbf{D}-\mathbf{W}$
3: $[\mathbf{U}, \boldsymbol{\Lambda}] \leftarrow \operatorname{eig}(\mathbf{L})$
4: $u_{k}(n) \leftarrow U(n, k)$, for $k=1, \ldots, M, n=0,1, \ldots, N-1$.
5: $\mathbf{M} \leftarrow \max _{n}(U(n, 1: L)), \mathbf{m} \leftarrow \min _{n}(U(n, 1: L))$
6: $\mathbf{q}_{n} \leftarrow\left[u_{1}(n), u_{2}(n), \ldots, u_{L}(n)\right]$, for all $n$
7: If $\mathrm{Map}=1, \mathbf{q}_{n} \leftarrow \mathbf{q}_{n} \bar{\Lambda}^{-1 / 2}$, end
8: If Map $=2, \mathbf{q}_{n} \leftarrow \mathbf{q}_{n}(\mathbf{I}-\bar{\Lambda})^{t}$, end
$9: \mathbf{y}_{n} \leftarrow \begin{cases}\mathbf{q}_{n}, & \text { for } S=0, \\ \mathbf{q}_{n} /\left\|\mathbf{q}_{n}\right\|_{2}, & \text { for } S=1, \\ \operatorname{sign}\left(\mathbf{q}_{n}\right), & \text { for } S=2, \\ \operatorname{sign}\left(\mathbf{q}_{n}-(\mathbf{M}+\mathbf{m}) / 2\right), & \text { for } S=3, \\ \left(\mathbf{q}_{n}-\mathbf{m}\right) \cdot /(\mathbf{M}-\mathbf{m}), & \text { for } S=4\end{cases}$
10: $\mathbf{Y} \leftarrow \mathbf{y}_{n}$, as the rows of $\mathbf{Y}$
11: $\mathbf{Z} \leftarrow\left\{\begin{array}{l}\mathbf{X}, \text { for } P=0, \\ \mathbf{Y}, \text { for } P=1\end{array}\right.$
12: ColorMap $\leftarrow\left\{\begin{array}{l}\text { Constant, for } C=0, \\ (\mathbf{Y}+1) / 2, \text { for } C=1\end{array}\right.$
13: GraphPlot( $\mathbf{W}, \mathbf{Z}, \mathbf{C o l o r M a p})$
14: Cluster the vertices according to $\mathbf{Y}$ and refine using the $k$-means algorithm (Remark 30) or the ratio cut recalculation algorithm (Remark 33).

## Output:

- New graph
- Subsets of vertex clusters
vertex coloring is: $C=0$, for the same color for all vertices is used, and $C=1$, when the spectral vector defines the vertex colors.


## C

## Other Graph Laplacian Forms

We here review some other forms of the graph Laplacian, including the Laplacian for directed graphs, graph Laplacian for the graphs with negative weights, and graph $p$-Laplacian.

## C. 1 Graph Laplacian for Directed Graphs

Directed graphs are typically analyzed based on the adjacency matrix and its spectrum. It is important to notice that the difference between a constant vector $\mathbf{x}$ and a vector $\mathbf{A x}$ does not result in a zero-valued vector; this is because, in general, the solution to the eigenvalue relation, $\mathbf{A u}=\lambda \mathbf{u}$, is not a vector with constant elements. The value $\mathbf{A x}$ and the difference $\mathbf{x}-\mathbf{A x} / \lambda_{\text {max }}$ will be used in Part II to define the shift on a graph. In order to introduce an operator on a directed graph which will restore the property of zero-valued total-variation for constant vectors, the Laplacian for directed graphs was introduced in Singh et al. (2016) and further analyzed in Sardellitti et al. (2017).

Since each edge in a directed graph connects one outgoing and one incoming vertex, in order to avoid ambiguity, the edges will be assigned
to the incoming vertex. The in-degree is then calculated as

$$
D^{i n}(m, m)=\sum_{n=0}^{N-1} W_{m n},
$$

with the Laplacian of a directed graph defined by

$$
\mathbf{L}=\mathbf{D}^{i n}-\mathbf{W} .
$$

In this way, the sum of each row in the Laplacian of a directed graph is zero-valued, meaning that any constant vector is also an eigenvector, with the corresponding $\lambda=0$. Spectral analysis is then performed using the eigendecomposition of $\mathbf{L}$, and since in this case $\mathbf{L}$ is not symmetric, the eigenvalues may be complex-valued.

The total variation of a vector, $\mathbf{x}$, for a shift operator, $\mathbf{S}$, is defined by

$$
\|\mathbf{x}-\mathbf{S} \mathbf{x}\|_{1}
$$

using the $L_{1}$ norm or

$$
\|\mathbf{x}-\mathbf{S} \mathbf{x}\|_{2},
$$

using the $L_{2}$ norm. If the operator $\mathbf{S}$ is defined as $\mathbf{S}=\mathbf{I}-\mathbf{L}$, where $\mathbf{I}$ is the identity matrix, then

$$
\|\mathbf{x}-\mathbf{S} \mathbf{x}\|_{2}^{2}=\|\mathbf{L} \mathbf{x}\|_{2}^{2}=(\mathbf{L} \mathbf{x})^{T} \mathbf{L} \mathbf{x}=\mathbf{x}^{T} \mathbf{L}^{T} \mathbf{L} \mathbf{x} .
$$

If the vector $\mathbf{x}$ is an eigenvector, $\mathbf{x}=\mathbf{u}_{k}$, then

$$
\left\|\mathbf{u}_{k}-\mathbf{S} \mathbf{u}_{k}\right\|_{2}^{2}=\mathbf{u}_{k}^{T} \mathbf{L}^{T} \mathbf{L} \mathbf{u}_{k}=\left|\lambda_{k}\right|^{2},
$$

which indicates that the smoothness of an eigenvector is proportional to $\left|\lambda_{k}\right|^{2}$. Therefore, in analogy to frequency in classical signal analysis it can be used as an indicator of the variation of an eigenvector.
Example 113: For the directed graph from Figure 2.1, the adjacency matrix is given by (2.2) and the in-degree matrix by

$$
\mathbf{D}^{i n}=\left[\begin{array}{llllllll}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0  \tag{C.1}\\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 2
\end{array}\right],
$$

with the corresponding graph Laplacian

$$
\mathbf{L}=\left[\begin{array}{rrrrrrrr}
1 & -1 & 0 & 0 & 0 & 0 & 0 & 0  \tag{C.2}\\
0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 4 & -1 & -1 & 0 & 0 & -1 \\
-1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & -1 & -1 & 0 & 3 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \\
0 & 0 & 0 & -1 & 0 & 0 & 2 & -1 \\
0 & 0 & -1 & 0 & 0 & 0 & -1 & 2
\end{array}\right] .
$$

Remark 117: Graph Laplacian of an undirected graph is a special case of the graph Laplacian of a directed graph, with each undirected edge being a combination of an incoming and an outgoing edge of the same weight.

## C. 2 Signed Graphs and Signed Graph Laplacian

Graphs for which edge weights may assume both positive and negative values are called signed graphs, and were introduced in Harary et al. (1953), where the authors motivated graphs with weights $\{1,0,-1\}$ through the modeling of social relations such as like, indifference, and dislike.

The vertex degree in a signed graph is defined as a sum of the absolute values of its weights (Hou, 2005), that is

$$
D_{a}(m, m)=\sum_{n=0}^{N-1}\left|W_{m n}\right|=\sum_{n=0}^{N-1} W_{m n} \operatorname{sign}\left(W_{m n}\right) .
$$

The corresponding signed graph Laplacian then becomes

$$
\mathbf{L}_{a}=\mathbf{D}_{a}-\mathbf{W},
$$

with the quadratic form of the Laplacian of a signed graph given by

$$
\mathbf{x}^{T} \mathbf{L}_{a} \mathbf{x}=\frac{1}{2} \sum_{m=0}^{N-1} \sum_{m=0}^{N-1}\left|W_{m n}\right|\left(x(m)-\operatorname{sign}\left(W_{m n}\right) x(n)\right) .
$$

Notice that the signed graph Laplacian is positive-semidefinite.

Cut of a signed graph. The cut of a signed graph represents a sum of all absolute weights that correspond to the edges which connect the vertices between the subsets, $\mathcal{E}$ and $\mathcal{H}$, that is

$$
\operatorname{Cut}(\mathcal{E}, \mathcal{H})=\sum_{\substack{m \in \mathcal{E} \\ n \in \mathcal{H}}}\left|W_{m n}\right| .
$$

All tools for the analysis of standard graphs can also be applied to signed graphs.

Notice that since the signed graph Laplacian may be positive definite, it then follows that a constant vector (with a zero eigenvalue) may not represent an eigenvector of the signed Laplacian. The concept of balanced graphs is introduced to deal with this issue, whereby a graph is said to be balanced if there exists a partition of its vertices into two disjoint subsets, $\mathcal{E}$ and $\mathcal{H}$, such that all positive edges reside within either $\mathcal{E}$ or $\mathcal{H}$, while all negative edges connect the vertices between $\mathcal{E}$ or $\mathcal{H}$. Then, the signed Laplacian, $\mathbf{L}_{a}$, of a connected signed graph is positive definite iff the graph is not balanced (Harary et al., 1953).

## C. 3 Graph $p$-Laplacian

A generalization of the graph Laplacian, called the $p$-Laplacian, and denoted by $\mathbf{L}_{p}$, is obtained from the generalization of the quadratic Laplacian form as (Bühler and Hein, 2009)

$$
\begin{equation*}
\mathbf{x}^{T} \mathbf{L}_{p} \mathbf{x}=\frac{1}{2} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} W_{m n}|x(n)-x(m)|^{p} \tag{C.3}
\end{equation*}
$$

Obviously, for $p=2$, the quadratic form of standard graph Laplacian, $\mathbf{L}$, is obtained. The elements of $\mathbf{L}_{p} \mathbf{x}$, denoted by $\mathcal{L}_{x}^{p}(n)$, that satisfy (C.3) are defined as

$$
\mathcal{L}_{x}^{p}(n)=\sum_{m=0}^{N-1} W_{m n}|x(n)-x(m)|^{p-1} \operatorname{sign}(x(n)-x(m))
$$

and it can be straightforwardly verified that the inner product of $x(n)$ and $\mathcal{L}_{x}^{p}(n)$ produces (C.3).

According to (28.9), the ratio graph cut, $\operatorname{Cut} N(\mathcal{E}, \mathcal{H})$, can be obtained by solving the minimization problem

$$
\begin{equation*}
\operatorname{Cut} N(\mathcal{E}, \mathcal{H})=\min _{\mathcal{E} \subset \mathcal{V}}\left\{\frac{\mathbf{x}^{T} \mathbf{L} \mathbf{x}}{\mathbf{x}^{T} \mathbf{x}}\right\} \tag{C.4}
\end{equation*}
$$

with $\mathbf{x}=\mathbf{u}_{1}$. Therefore, the eigenvector $\mathbf{u}_{1}$ can be considered as a (non-constant) solution to the minimization problem in (C.4).

Similarly, the minimization problem for the $p$-Laplacian becomes

$$
\begin{equation*}
\min _{\mathcal{E} \subset \mathcal{V}}\left\{\frac{\mathbf{x}^{T} \mathbf{L}_{p} \mathbf{x}}{\min _{c}\|\mathbf{x}-c\|_{p}^{p}}\right\} . \tag{C.5}
\end{equation*}
$$

with the solution in the form of the first eigenvector, $\mathbf{u}_{1}^{(p)}$, comprising the elements, $v_{1}^{(p)}(n)$, of the $p$-Laplacian (Bühler and Hein, 2009)

$$
\mathcal{L}_{v_{1}^{(p)}}^{p}(n)=\lambda_{1}^{(p)}\left|v_{1}^{(p)}(n)\right|^{p-1} \operatorname{sign}\left(v_{1}^{(p)}(n)\right) .
$$

Notice that for $\mathbf{x}=\mathbf{v}_{1}^{(p)}$ the minimum value of (C.5) is equal to the eigenvalue, $\lambda_{1}^{(p)}$.

The Cheeger ratio cut, $\phi(\mathcal{V})$, with the $p$-Laplacian exhibits the following general bounds

$$
\begin{align*}
\left(\frac{2}{\max _{i} d_{i}}\right)^{p-1}\left(\frac{\phi(\mathcal{V})}{p}\right)^{p} & \leq \lambda_{1}^{(p)} \leq 2^{p-1} \phi(\mathcal{V})  \tag{C.6}\\
\frac{\phi(\mathcal{V})}{\max _{i} d_{i}} & \leq \frac{\phi^{*}(\mathcal{V})}{\max _{i} d_{i}} \leq p\left(\frac{\phi(\mathcal{V})}{\max _{i} d_{i}}\right)^{1 / p} \tag{C.7}
\end{align*}
$$

where $d_{i}$ is the degree of vertex $i$ and $\phi^{*}(\mathcal{V})$ is the minimum Cheeger's ratio cut obtained by an optimal thresholding of the eigenvector $\mathbf{u}_{1}$ with a threshold $t$, that is, a vertex $n$ belongs to the subset of vertices $\mathcal{E}$ if $u_{1}(n)>t$.

The above inequality implies that the bounds are tight for $p \rightarrow 1$, which indicates that the 1-Laplacian based cut is equivalent to the Cheeger ratio cut; this may be used to improve the cut performance in practical applications. Still, the main problem remains in the computational issues related to calculation of the $p$-Laplacian eigenvectors, especially for $p \rightarrow 1$ (Bühler and Hein, 2009; Chang, 2016; Chang et al., 2016).

## Acknowledgments

We wish to express our sincere gratitude to Yao Lei Xu, Kriton Konstantinidis, Ghena Hammour, Shota Saito, and Giacomo Kahn whose thorough proofreading and deep insight have been of great help at various stages of manuscript preparation.

## References

Afrati, F. and A. G. Constantinides. (1978). "The use of graph theory in binary block code construction". In: Proceedings of the International Conference on Digital Signal Processing. 228-233.
Agaskar, A. and Y. M. Lu. (2013). "A spectral graph uncertainty principle". IEEE Transactions on Information Theory. 59(7): 43384356.

Anis, A., A. Gadde, and A. Ortega. (2016). "Efficient sampling set selection for bandlimited graph signals using graph spectral proxies". IEEE Transactions on Signal Processing. 64(14): 3775-3789.
Ann B. Lee, B. N. and L. Wasserman. (2008). "Treelets: An adaptive multiscale basis for sparse unordered data". The Annals of Applied Statistics. 2(2): 435-471.
Atwood, J. and D. Towsley. (2016). "Diffusion-convolutional neural networks". In: Advances in Neural Information Processing Systems. 1993-2001.
Baba, K., R. Shibata, and M. Sibuya. (2004). "Partial correlation and conditional correlation as measures of conditional independence". Australian $\mathcal{\xi}$ New Zealand Journal of Statistics. 46(4): 657-664.
Bacciu, D. and L. Di Sotto. (2019). "A non-negative factorization approach to node pooling in graph convolutional neural networks". In: Proceedings of the International Conference of the Italian Association for Artificial Intelligence. Springer. 294-306.

Bacciu, D., F. Errica, and A. Micheli. (2018). "Contextual graph Markov model: A deep and generative approach to graph processing". arXiv preprint arXiv:1805.10636.
Bacciu, D. and D. P. Mandic. (2020). "Tensor decompositions in deep learning". In: Proc. of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN'20). 441-450.
Baingana, B. and G. B. Giannakis. (2016). "Tracking switched dynamic network topologies from information cascades". IEEE Transactions on Signal Processing. 65(4): 985-997.
Banerjee, O., L. E. Ghaoui, and A. d'Aspremont. (2008). "Model selection through sparse maximum likelihood estimation for multivariate Gaussian or binary data". Journal of Machine Learning Research. 9(Mar): 485-516.
Bapat, R. (1996). "The Laplacian matrix of a graph". Mathematics Student-India. 65(1): 214-223.
Barik, S., R. B. Bapat, and S. Pati. (2015). "On the Laplacian spectra of product graphs". Applicable Analysis and Discrete Mathematics: 39-58.
Behjat, H., N. Leonardi, L. Sörnmo, and D. Van De Ville. (2015). "Anatomically-adapted graph wavelets for improved group-level fMRI activation mapping". NeuroImage. 123: 185-199.
Behjat, H., U. Richter, D. Van De Ville, and L. Sörnmo. (2016). "Signaladapted tight frames on graphs". IEEE Transactions on Signal Processing. 64(22): 6017-6029.
Behjat, H. and D. Van De Ville. (2019). "Spectral design of signaladapted tight frames on graphs". In: Vertex-Frequency Analysis of Graph Signals. Springer. 177-206.
Belkin, M. and P. Niyogi. (2003). "Laplacian eigenmaps for dimensionality reduction and data representation". Neural Computation. 15(6): 1373-1396.
Belkin, M. and P. Niyogi. (2008). "Towards a theoretical foundation for Laplacian-based manifold methods". Journal of Computer and System Sciences. 74(8): 1289-1308.
Berge, C. (1984). Hypergraphs: Combinatorics of Finite Sets. Vol. 45. Elsevier.

Black, F. and R. Litterman. (1992). "Global Portfolio Optimization". Financial Analysts Journal. 48(5): 280-291.
Boashash, B. (2015). Time-frequency signal analysis and processing: A comprehensive reference. Academic Press.
Boginski, V., S. Butenko, S. O., S. Trunkhanov, and J. Gil Lafuente. (2014). "A Network-Based Data Mining Approach to Portfolio Selection via Weighted Clique Relaxations". Annals of Operations Research. 216: 23-34.
Boginski, V., S. Butenko, and P. M. Pardalos. (2003). "On Structural Properties of the Market Graph". In: Innovations in Financial and Economic Networks. Ed. by A. Nagurney. Edward Elgar Publishers. 29-45.
Boginski, V., S. Butenko, and P. M. Pardalos. (2005). "Statistical Analysis of Financial Networks". Computational Statistics \& Data Analysis. 48(2): 431-443.
Boginski, V., S. Butenko, and P. M. Pardalos. (2006). "Mining Market Data: A Network Approach". Computers \& Operations Research. 33(11): 3171-3184.
Bohannon, A. W., B. M. Sadler, and R. V. Balan. (2019). "A filtering framework for time-varying graph signals". In: Vertex-Frequency Analysis of Graph Signals. Springer. 341-376.
Bojchevski, A., O. Shchur, D. Zügner, and S. Günnemann. (2018). "NetGAN: Generating graphs via random walks". arXiv preprint arXiv:1803.00816.
Brandes, U. (2005). Network Analysis: Methodological Foundations. Springer.
Brouwer, A. E. and W. H. Haemers. (2012). Spectra of Graphs. New York: Springer-Verlag.
Bruna, J., W. Zaremba, A. Szlam, and Y. LeCun. (2013). "Spectral networks and locally connected networks on graphs". arXiv preprint arXiv:1312.6203.
Bühler, T. and M. Hein. (2009). "Spectral clustering based on the graph p-Laplacian". In: Proceedings of the 26th ACM Annual International Conference on Machine Learning. 81-88.

Bunse-Gerstner, A. and W. B. Gragg. (1988). "Singular value decompositions of complex symmetric matrices". Journal of Computational and Applied Mathematics. 21(1): 41-54.
Caetano, T. S., J. J. McAuley, L. Cheng, Q. V. Le, and A. J. Smola. (2009). "Learning graph matching". IEEE Transactions on Pattern Analysis and Machine Intelligence. 31(6): 1048-1058.
Calkin, N. J. and M. Lopez de Prado. (2014a). "Stochastic Flow Diagrams". Algorithmic Finance. 3(1-2): 21-42.
Calkin, N. J. and M. Lopez de Prado. (2014b). "The Topology of Macro Financial Flows: An Application of Stochastic Flow Diagrams". Algorithmic Finance. 3(1): 43-85.
Calkin, N. J. and M. Lopez de Prado. (2016). "Building Diversified Portfolios that Outperform Out of Sample". The Journal of Portfolio Management. 42(4): 59-69.
Calvi, G. G., A. Moniri, M. Mahfouz, Q. Zhao, and D. P. Mandic. (2019). "Compression and interpretability of deep neural networks via Tucker tensor layer: From first principles to tensor valued backpropagation". arXiv preprint arXiv:1903.06133.
Camponogara, E. and L. F. Nazari. (2015). "Models and algorithms for optimal piecewise-linear function approximation". Mathematical Problems in Engineering. 2015.
Candes, E. J. (2008). "The restricted isometry property and its implications for compressed sensing". Comptes Rendus Mathematique. 346(9-10): 589-592.
Candès, E. J., J. Romberg, and T. Tao. (2006). "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information". IEEE Transactions on Information Theory. 52(2): 489-509.
Chami, I., S. Abu-El-Haija, B. Perozzi, C. Re, and K. Murphy. (2020). "Machine learning on graphs: A model and comprehensive taxonomy". arXiv preprint arXiv:2005.03675.
Chandra, A. K., P. Raghavan, W. L. Ruzzo, R. Smolensky, and P. Tiwari. (1996). "The electrical resistance of a graph captures its commute and cover times". Computational Complexity. 6(4): 312-340.
Chang, K., S. Shao, and D. Zhang. (2016). "The 1-Laplacian Cheeger cut: Theory and algorithms". arXiv preprint arXiv:1603.0168\%.

Chang, K. C. (2016). "Spectrum of the 1-Laplacian and Cheeger's constant on graphs". J. Graph Theory. 81(2): 167-207.
Chen, G., D. R. Glen, Z. S. Saad, J. P. Hamilton, M. E. Thomason, I. H. Gotlib, and R. W. Cox. (2011). "Vector autoregression, structural equation modeling, and their synthesis in neuroimaging data analysis". Computers in Biology and Medicine. 41(12): 1142-1155.
Chen, S., A. Sandryhaila, and J. Kovačević. (2015a). "Sampling theory for graph signals". In: Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 3392-3396.
Chen, S., A. Sandryhaila, J. M. Moura, and J. Kovačević. (2014). "Signal denoising on graphs via graph filtering". In: Proc. 2014 IEEE Global Conference on Signal and Information Processing (GlobalSIP). 872876.

Chen, S., A. Sandryhaila, J. M. Moura, and J. Kovačević. (2015b). "Signal recovery on graphs: Variation minimization". IEEE Transactions on Signal Processing. 63(17): 4609-4624.
Chen, S., R. Varma, A. Sandryhaila, and J. Kovačević. (2015c). "Discrete signal processing on graphs: Sampling theory". IEEE Transactions on Signal Processing. 63(24): 6510-6523.
Chen, S., R. Varma, A. Singh, and J. Kovačević. (2016). "Signal recovery on graphs: Fundamental limits of sampling strategies". IEEE Transactions on Signal and Information Processing over Networks. 2(4): 539-554.
Chepuri, S. P. and G. Leus. (2016). "Subsampling for graph power spectrum estimation". In: Proc. IEEE Sensor Array and Multichannel Signal Processing Workshop (SAM). 1-5.
Chepuri, S. P., S. Liu, G. Leus, and A. O. Hero. (2017). "Learning sparse graphs under smoothness prior". In: Proceedings of the 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE. 6508-6512.
Christofides, N. (1975). Graph Theory: An Algorithmic Approach. Academic Press.
Chung, F. (1997). Spectral Graph Theory. Providence, RI: AMS.
Chung, F. (2005). "Laplacians and the Cheeger inequality for directed graphs". Annals of Combinatorics. 9(1): 1-19.

Chung, F. R. and R. P. Langlands. (1996). "A combinatorial Laplacian with vertex weights". Journal of Combinatorial Theory, Series A. $75(2): 316-327$.
Cichocki, A., N. Lee, I. Oseledets, A.-H. Phan, Q. Zhao, and D. P. Mandic. (2016). "Tensor networks for dimensionality reduction and large-scale optimization. Part 1: Low-rank tensor decompositions". Foundations and Trends ${ }^{(8)}$ in Machine Learning. 9(4-5): 249-429.
Cichocki, A., A.-H. Phan, Q. Zhao, N. Lee, I. V. Oseledets, M. Sugiyama, and D. Mandic. (2017). "Tensor networks for dimensionality reduction and large-scale optimization. Part 2: Applications and future perspectives". Foundations and Trends ${ }^{\circledR}$ in Machine Learning. 9(6): 431-673.
Cioacă, T., B. Dumitrescu, and M.-S. Stupariu. (2019). "Graph-based wavelet multiresolution modeling of multivariate terrain data". In: Vertex-Frequency Analysis of Graph Signals. Springer. 479-507.
Clarke, R., H. De Silva, and S. Thorley. (2002). "Portfolio Constraints and the Fundamental Law of Active Management". Financial Analysts Journal. 58: 48-66.
Cohen, L. (1995). Time-frequency Analysis. Electrical Engineering Signal Processing. Prentice Hall PTR. ISBN: 9780135945322.
Coifman, R. R. and S. Lafon. (2006). "Diffusion maps". Applied and Computational Harmonic Analysis. 21(1): 5-30.
Cooper, J. and A. Dutle. (2012). "Spectra of uniform hypergraphs". Linear Algebra and Its Applications. 436(9): 3268-3292.
Cvetković, D. M. and M. Doob. (1985). "Developments in the theory of graph spectra". Linear and Multilinear Algebra. 18(2): 153-181.
Cvetković, D. M., M. Doob, and H. Sachs. (1980). Spectra of Graphs: Theory and Application. Vol. 87. Academic Press.
Cvetković, D. M. and I. Gutman. (2011). Selected Topics on Applications of Graph Spectra. Matematički Institut SANU (Serbian Academy of Scences and Arts).
Dai, H., Z. Kozareva, B. Dai, A. Smola, and L. Song. (2018). "Learning steady-states of iterative algorithms over graphs". In: Proceedings of the International Conference on Machine Learning. 1114-1122.
Daković, M., L. Stanković, and E. Sejdić. (2019). "Local smoothness of graph signals". Mathematical Problems in Engineering. 2019.

Dal Col, A., P. Valdivia, F. Petronetto, F. Dias, C. T. Silva, and L. G. Nonato. (2019). "Wavelet-based visual data exploration". In: Vertex-Frequency Analysis of Graph Signals. Springer. 459-478.
Das, A., A. L. Sampson, C. Lainscsek, L. Muller, W. Lin, J. C. Doyle, S. S. Cash, E. Halgren, and T. J. Sejnowski. (2017). "Interpretation of the precision matrix and its application in estimating sparse brain connectivity during sleep spindles from human electrocorticography recordings". Neural Computation. 29(3): 603-642.
De Cao, N. and T. Kipf. (2018). "MolGAN: An implicit generative model for small molecular graphs". arXiv preprint arXiv:1805.11973.
De Miguel V., L. G. and R. R. Uppal. (2009). "Optimal Versus Naive Diversification: How Inefficient is the $1 / N$ Portfolio Strategy?" Review of Financial Studies. 22: 1915-1953.
Dees, B. S., A. G. Constantinides, and D. P. Mandic. (2019). "Graph theory and metro traffic modelling". arXiv preprint arXiv:1912.05964.
Defferrard, M., X. Bresson, and P. Vandergheynst. (2016). "Convolutional neural networks on graphs with fast localized spectral filtering". In: Advances in Neural Information Processing Systems. 3844-3852.
Dempster, A. P. (1972). "Covariance selection". Biometrics: 157-175.
Dhillon, I. S., Y. Guan, and B. Kulis. (2004). "Kernel k-means: Spectral clustering and normalized cuts". In: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 551-556.
Doersch, C. (2016). "Tutorial on variational autoencoders". arXiv preprint arXiv:1606.05908.
Dong, X., D. Thanou, P. Frossard, and P. Vandergheynst. "Learning Graphs from Signal Observations under Smoothness Prior".
Dong, X., P. Frossard, P. Vandergheynst, and N. Nefedov. (2012). "Clustering with multi-layer graphs: A spectral perspective". IEEE Transactions on Signal Processing. 60(11): 5820-5831.
Dong, X., D. Thanou, P. Frossard, and P. Vandergheynst. (2016). "Learning Laplacian matrix in smooth graph signal representations". IEEE Transactions on Signal Processing. 64(23): 6160-6173.

Dong, X., D. Thanou, M. Rabbat, and P. Frossard. (2019). "Learning graphs from data: A signal representation perspective". IEEE Signal Processing Magazine. 36(3): 44-63.
Dorfler, F. and F. Bullo. (2012). "Kron reduction of graphs with applications to electrical networks". IEEE Transactions on Circuits and Systems I: Regular Papers. 60(1): 150-163.
Duncan, A. (2004). "Powers of the adjacency matrix and the walk matrix". The Collection: 1-11.
Ekambaram, V. N. (2014). Graph-Structured Data Viewed Through a Fourier Lens. Berkeley: University of California.
Elad, M. and A. M. Bruckstein. (2002). "Generalized uncertainty principle and sparse representation in pairs of bases". IEEE Transactions on Information Theory. 48(9): 2558-2567.
Epskamp, S. and E. I. Fried. (2018). "A tutorial on regularized partial correlation networks." Psychological Methods.
Erb, W. (2019). "Shapes of uncertainty in spectral graph theory". arXiv preprint arXiv:1909.10865.
Fiedler, M. (1973a). "Algebraic Connectivity of Graphs". Czechoslovak Mathematical Journal. 23(2): 298-305.
Fiedler, M. (1973b). "Algebraic connectivity of graphs". Czechoslovak Mathematical Journal. 23(2): 298-305.
Freeman, L. C. (1977). "A Set of Measures of centrality based on Betweenness". Sociometry. 40: 35-41.
Friedman, J., T. Hastie, and R. Tibshirani. (2008). "Sparse inverse covariance estimation with the graphical LASSO". Biostatistics. 9(3): 432-441.
Fujiwara, K. (1995). "Eigenvalues of Laplacians on a closed Riemannian manifold and its nets". Proceedings of the American Mathematical Society. 123(8): 2585-2594.
Gama, F., E. Isufi, A. Ribeiro, and G. Leus. (2019). "Controllability of bandlimited graph processes over random time varying graphs". IEEE Transactions on Signal Processing. 67(24): 6440-6454.
Gauvin, L., A. Panisson, and C. Cattuto. (2014). "Detecting the community structure and activity patterns of temporal networks: a nonnegative tensor factorization approach". PLOS ONE. 9(1): e86028.

Gavili, A. and X.-P. Zhang. (2017). "On the shift operator, graph frequency, and optimal filtering in graph signal processing". IEEE Transactions on Signal Processing. 65(23): 6303-6318.
Giannakis, G. B., Y. Shen, and G. V. Karanikolas. (2018). "Topology identification and learning over graphs: Accounting for nonlinearities and dynamics". Proceedings of the IEEE. 106(5): 787-807.
Gilmer, J., S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. (2017). "Neural message passing for quantum chemistry". In: Proceedings of the 34th International Conference on Machine LearningVolume 70. JMLR. org. 1263-1272.
Girault, B. (2015). "Stationary graph signals using an isometric graph translation". In: Proc. 23rd European Signal Processing Conference (EUSIPCO). 1516-1520.
Girault, B., P. Gonçalves, and É. Fleury. (2015). "Translation on graphs: An isometric shift operator". IEEE Signal Processing Letters. 22(12): 2416-2420.
Gori, M., G. Monfardini, and F. Scarselli. (2005). "A new model for learning in graph domains". In: Proceedings of the IEEE International Joint Conference on Neural Networks, 2005. Vol. 2. IEEE. 729-734.
Grady, L. J. and J. R. Polimeni. (2010). Discrete Calculus: Applied Analysis on Graphs for Computational Science. Springer Science \& Business Media.
Grassi, F., A. Loukas, N. Perraudin, and B. Ricaud. (2017). "A timevertex signal processing framework: Scalable processing and meaningful representations for time-series on graphs". IEEE Transactions on Signal Processing. 66(3): 817-829.
Grebenkov, D. S. and B.-T. Nguyen. (2013). "Geometrical structure of Laplacian eigenfunctions". SIAM Review. 55(4): 601-667.
Grotas, S., Y. Yakoby, I. Gera, and T. Routtenberg. (2019). "Power Systems Topology and State Estimation by Graph Blind Source Separation". IEEE Transactions on Signal Processing. 67(8): 20362051.

Grover, A. and J. Leskovec. (2016). "Node2Vec: Scalable feature learning for networks". In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 855864.

Grover, A., A. Zweig, and S. Ermon. (2019). "Graphite: Iterative generative modeling of graphs". In: Proc. International Conference on Machine Learning. 2434-2444.
Gu, Y. and X. Wang. (2019). "Local-Set-Based Graph Signal Sampling and Reconstruction". In: Vertex-Frequency Analysis of Graph Signals. Springer. 255-292.
Gunawardena, A. A., R. R. Meyer, and W. L. Dougan. (2012). "Optimal Selection of an Independent Set of Cliques in a Market Graph". In Proceedings of the International Conference on Economics, Business and Marketing Management: 281-285.
Hagen, L. and A. B. Kahng. (1992a). "New Spectral Methods for Ratio Cut Partitioning and Clustering". IEEE Transactions on ComputerAided Design of Integrated Circuits and Systems. 11(9): 1074-1085.
Hagen, L. and A. B. Kahng. (1992b). "New spectral methods for ratio cut partitioning and clustering". IEEE Trans. Computer-Aided Design of Int. Circuits and Systems. 11(9): 1074-1085.
Hamilton, W., Z. Ying, and J. Leskovec. (2017). "Inductive representation learning on large graphs". In: Advances in Neural Information Processing Systems. 1024-1034.
Hammond, D., P. Vandergheynst, and R. Gribonval. (2011a). "Wavelets on graphs via spectral graph theory". Appl. Comput. Harmon. Anal. 30(2): 129-150.
Hammond, D. K., P. Vandergheynst, and R. Gribonval. (2011b). "Wavelets on graphs via spectral graph theory". Applied and Computational Harmonic Analysis. 30(2): 129-150.
Hammond, D. K., P. Vandergheynst, and R. Gribonval. (2019). "The spectral graph wavelet transform: Fundamental theory and fast computation". In: Vertex-Frequency Analysis of Graph Signals. Springer. 141-175.

Hamon, R., P. Borgnat, P. Flandrin, and C. Robardet. (2016a). "Relabelling vertices according to the network structure by minimizing the cyclic bandwidth sum". Journal of Complex Networks. 4(4): 534-560.
Hamon, R., P. Borgnat, P. Flandrin, and C. Robardet. (2019). "Transformation from Graphs to Signals and Back". In: Vertex-Frequency Analysis of Graph Signals. Springer. 111-139.
Hamon, R., P. Borgnat, P. Flandrin, and C. Robardet. (2016b). "Extraction of temporal network structures from graph-based signals". IEEE Transactions on Signal and Information Processing Over Networks. 2(2): 215-226.
Harary, F. et al. (1953). "On the notion of balance of a signed graph." The Michigan Mathematical Journal. 2(2): 143-146.
Hasanzadeh, A., E. Hajiramezanali, K. Narayanan, N. Duffield, M. Zhou, and X. Qian. (2019). "Semi-implicit graph variational autoencoders". In: Advances in Neural Information Processing Systems. 10712-10723.
Heimowitz, A. and Y. C. Eldar. (2017). "A unified view of diffusion maps and signal processing on graphs". In Proceedings of the International Conference on Sampling Theory and Applications (SampTA): 308312.

Horaud, R. (2009). "A short tutorial on graph Laplacians, Laplacian embedding, and spectral clustering".
Hou, Y. P. (2005). "Bounds for the least Laplacian eigenvalue of a signed graph". Acta Mathematica Sinica. 21(4): 955-960.
Imre, M., J. Tao, Y. Wang, Z. Zhao, Z. Feng, and C. Wang. (2020). "Spectrum-preserving sparsification for visualization of big graphs". Computers \& Graphics. 87: 89-102.
Ioannidis, V. N., D. Berberidis, and G. B. Giannakis. (2019a). "GraphSAC: Detecting anomalies in large-scale graphs". arXiv preprint arXiv:1910.09589.
Ioannidis, V. N., Y. Shen, and G. B. Giannakis. (2019b). "Semi-Blind Inference of Topologies and Dynamical Processes Over Dynamic Graphs". IEEE Transactions on Signal Processing. 67(9): 2263-2274.

Isufi, E., A. Loukas, A. Simonetto, and G. Leus. (2017). "Filtering random graph processes over random time-varying graphs". IEEE Transactions on Signal Processing. 65(16): 4406-4421.
Jain, A. K. (2010). "Data clustering: 50 years beyond K-means". Pattern Recognition Letters. 31(8): 651-666.
Jansen, M., G. P. Nason, and B. W. Silverman. (2009). "Multiscale methods for data on graphs and irregular multidimensional situations". Journal of the Royal Statistical Society: Series B (Statistical Methodology). 71(1): 97-125.
Jeh, G. and J. Widom. (2002). "SIMRANK: A measure of structuralcontext similarity". In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 538-543.
Jestrović, I., J. L. Coyle, and E. Sejdić. (2017). "A fast algorithm for vertex-frequency representations of signals on graphs". Signal processing. 131: 483-491.
Jin, Y., A. Loukas, and J. JaJa. (2020). "Graph coarsening with preserved spectral properties". In: Proc. International Conference on Artificial Intelligence and Statistics. 4452-4462.
Jones, O. (2013). Spectra of Simple Graphs.
Jordan, M. I. et al. (2004). "Graphical models". Statistical Science. 19(1): 140-155.
Jordan, M. I. (1998). Learning in Graphical Models. Vol. 89. Springer Science \& Business Media.
Jovanović, I. and Z. Stanić. (2012). "Spectral distances of graphs". Linear Algebra and Its Applications. 436(5): 1425-1435.
Kalofolias, V. (2016). "How to learn a graph from smooth signals". In: Proceedings of the Artificial Intelligence and Statistics. 920-929.
Kalyagin, V., A. Koldanov, P. Koldanov, and V. Zamaraev. (2014). "Market graph and Markowitz model". In: Optimization in Science and Engineering. Ed. by T. M. Rassias, C. A. Floudas, and S. Butenko. Springer. 293-306.
Kaplan, D. (2008). Structural equation modeling: Foundations and extensions. Vol. 10. Sage Publications.

Katsimpras, G. and G. Paliouras. (2019). "Class-aware tensor factorization for multi-relational classification". Information Processing $\mathcal{E}$ Management. In Press.
Khuller, S. (1998). "Approximation algorithms for finding highly connected subgraphs". Tech. rep.
Kim, S.-J., K. Koh, S. Boyd, and D. Gorinevsky. (2009). "L1 trend filtering". SIAM review. 51(2): 339-360.
Kingma, D. P. and M. Welling. (2013). "Auto-encoding variational bayes", arXiv preprint arXiv:1312.6114.
Kipf, T. N. and M. Welling. (2016a). "Semi-supervised classification with graph convolutional networks". arXiv preprint arXiv:1609.02907.
Kipf, T. N. and M. Welling. (2016b). "Variational graph auto-encoders". arXiv preprint arXiv:1611.07308.
Kleinberg, J. and E. Tardos. (2006). Algorithm Design. Pearson Education India.
Kolaczyk, E. D. (2009). Statistical Analysis of Network Data - Methods and Models. Springer-Verlag New York.
Kolm, P. N., R. Tutuncu, and F. J. Fabozzi. (2014). "60 Years of Portfolio Optimization: Practical Challenges and Current Trends". European Journal of Operational Research. 234(2): 356-371.
Krim, H. and A. B. Hamza. (2015). Geometric Methods in Signal and Image Analysis. Cambridge University Press.
Kron, G. (1963). Diakoptics: The Piecewise Solution of Large-Scale Systems. Vol. 2. MacDonald.
LeCun, Y., L. Bottou, Y. Bengio, P. Haffner, et al. (1998). "Gradientbased learning applied to document recognition". Proceedings of the IEEE. 86(11): 2278-2324.
Ledoit, O. and M. Wolf. (2003). "Improved Estimation of the Covariance Matrix of Stock Returns With an Application to Portfolio Selection". Journal of Empirical Finance. 10(5): 603-621.
Leonardi, N. and D. Van De Ville. (2013). "Tight wavelet frames on multislice graphs". IEEE Transactions on Signal Processing. 61(13): 3357-3367.
Leskovec, J. and C. Faloutsos. (2006). "Sampling from large graphs". In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 631-636.

Li, S., Z. Yu, M. Xiang, and D. Mandic. (2020). "Reciprocal adversarial learning via characteristic functions". arXiv preprint arXiv:2006. 08413.

Li, S., Y. Jin, and D. I. Shuman. (2019a). "Scalable $M$-channel critically sampled filter banks for graph signals". IEEE Transactions on Signal Processing. 67(15): 3954-3969.
Li, Y., X. F. Jiang, Y. Tian, S. P. Li, and B. Zheng. (2019b). "Portfolio Optimization Based on Network Topology". Physica A. 515: 671681.

Li, Y., D. Tarlow, M. Brockschmidt, and R. Zemel. (2015). "Gated graph sequence neural networks". arXiv preprint arXiv:1511.05493.
Li, Y., O. Vinyals, C. Dyer, R. Pascanu, and P. Battaglia. (2018). "Learning deep generative models of graphs". arXiv preprint arXiv:1803. 03324.

Liben-Nowell, D. and J. Kleinberg. (2007). "The link-prediction problem for social networks". Journal of the American Society for Information Science and Technology. 58(7): 1019-1031.
Lin, Y., J. Sun, P. Castro, R. Konuru, H. Sundaram, and A. Kelliher. (2009). "MetaFac: community discovery via relational hypergraph factorization". In Proceedings of the ACM KDD International Conference on Knowledge Discovery and Data Mining: 527-536.
Lin, Y. R., Y. Chi, S. Zhu, H. Sundaram, and B. L. Tseng. (2008). "Facetnet: a framework for analyzing communities and their evolutions in dynamic networks". In Proceedings of the International Conference on World Wide Web (WWW): 685-694.
Lorenzo, P., S. Barbarossa, and P. Banelli. (2018). "Sampling and recovery of graph signals". In: Cooperative and Graph Signal Processing. Elsevier. 261-282.
Loukas, A. and N. Perraudin. (2016). "Stationary time-vertex signal processing", arXiv preprint arXiv:1611.00255.
Loukas, A. and P. Vandergheynst. (2018). "Spectrally approximating large graphs with smaller graphs". arXiv preprint arXiv:1802.07510.
Lu, H., Z. Fu, and X. Shu. (2014). "Non-negative and sparse spectral clustering". Pattern Recognition. 47(1): 418-426.

Ma, T., J. Chen, and C. Xiao. (2018). "Constrained generation of semantically valid graphs via regularizing variational autoencoders". In: Advances in Neural Information Processing Systems. 7113-7124.
Maggioni, M. and H. Mhaskar. (2008). "Diffusion polynomial frames on metric measure spaces". Applied and Computational Harmonic Analysis. 24(3): 329-353.
Maheswari, S. U. and B. Maheswari. (2016). "Some properties of Cartesian product graphs of Cayley graphs with arithmetic graphs". International Journal of Computer Applications. 138(3): 26-29.
Malik, J., S. Belongie, T. Leung, and J. Shi. (2001). "Contour and texture analysis for image segmentation". International Journal of Computer Vision. 43(1): 7-27.
Mandic, D. and J. Chambers. (2001). Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability. Wiley.
Mandic, D. P. and V. S. L. Goh. (2009). Complex valued nonlinear adaptive filters: noncircularity, widely linear and neural models. Vol. 59. John Wiley \& Sons.
Mandic, D. (2007). "Machine learning and signal processing applications of fixed point theory". Tutorial in IEEE ICASSP, 2007.
Mao, X. and Y. Gu. (2019). "Time-Varying Graph Signals Reconstruction". In: Vertex-Frequency Analysis of Graph Signals. Springer. 293-316.
Markowitz, H. (1952). "Portfolio Selection". Journal of Finance. 7(1): 77-91.
Marques, A., A. Ribeiro, and S. Segarra. (2017). "Graph signal processing: Fundamentals and applications to diffusion processes". In: IEEE Proc. Int. Conf. Accoustic, Speech and Signal Processing (ICASSP), Tutorial.
Marques, A. G., S. Segarra, G. Leus, and A. Ribeiro. (2016). "Sampling of graph signals with successive local aggregations." IEEE Transactions Signal Processing. 64(7): 1832-1843.
Marques, A. G., S. Segarra, G. Leus, and A. Ribeiro. (2017). "Stationary graph processes and spectral estimation". IEEE Transactions on Signal Processing. 65(22): 5911-5926.

Masoumi, M. and A. B. Hamza. (2017). "Spectral shape classification: A deep learning approach". Journal of Visual Communication and Image Representation. 43: 198-211.
Masoumi, M., C. Li, and A. B. Hamza. (2016). "A spectral graph wavelet approach for nonrigid 3D shape retrieval". Pattern Recognition Letters. 83: 339-348.
Masoumi, M., M. Rezaei, and A. B. Hamza. (2019). "Shape analysis of carpal bones using spectral graph wavelets". In: Vertex-Frequency Analysis of Graph Signals. Springer. 419-436.
Masuda, N., M. A. Porter, and R. Lambiotte. (2017). "Random walks and diffusion on networks". Physics reports. 716: 1-58.
Mateos, G., S. Segarra, A. G. Marques, and A. Ribeiro. (2019). "Connecting the dots: Identifying network structure via graph signal processing". IEEE Signal Processing Magazine. 36(3): 16-43.
McInnes, L., J. Healy, N. Saul, and L. Großberger. (2018). "UMAP: Uniform manifold approximation and projection". Journal of Open Source Software. 3(29): 861. DOI: 10.21105/joss. 00861.
Mei, J. and J. M. Moura. (2016). "Signal processing on graphs: Causal modeling of unstructured data". IEEE Transactions on Signal Processing. 65(8): 2077-2092.
Meinshausen, N., P. Bühlmann, et al. (2006). "High-dimensional graphs and variable selection with the LASSO". The Annals of Statistics. 34(3): 1436-1462.
Mejia, D., O. Ruiz-Salguero, and C. A. Cadavid. (2017). "Spectral-based mesh segmentation". International Journal on Interactive Design and Manufacturing (IJIDeM). 11(3): 503-514.
Meyer, Y. (1992). Wavelets and Operators. Cambridge University Press.
Micheli, A. (2009). "Neural network for graphs: A contextual constructive approach". IEEE Transactions on Neural Networks. 20(3): 498511.

Mijalkov, M., E. Kakaei, J. B. Pereira, E. Westman, and G. Volpe. (2017). "BRAPH: A graph theory software for the analysis of brain connectivity". PLOS ONE. 12(8): e0178798.

Misiakos, P., C. Wendler, and M. Püschel. (2020). "Diagonalizable shift and filters for directed graphs based on the Jordan-Chevalley decomposition". In: Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 5635-5639.
Mohar, B. (1989). "Isoperimetric numbers of graphs". Journal of Combinatorial Theory, Series B. 47(3): 274-291.

Monti, F., D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein. (2017). "Geometric deep learning on graphs and manifolds using mixture model CNNs". In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 5115-5124.

Mordeson, J. N. and P. S. Nair. (2012). Fuzzy Graphs and Fuzzy Hypergraphs. Vol. 46. Physica.
Morris, O., M. J. de Lee, and A. Constantinides. (1986). "Graph theory for image analysis: An approach based on the shortest spanning tree". IEE Proceedings F (Communications, Radar and Signal Processing). 133(2): 146-152.
Motl, J. and O. Schulte. (2015). "The CTU prague relational learning repository". arXiv preprint arXiv:1511.03086.
Moura, J. M. (2018). "Graph signal processing". In: Cooperative and Graph Signal Processing. Ed. by P. Djuric and C. Richard. Elsevier. 239-259.
Murtagh, F. (2007). "The Haar wavelet transform of a dendrogram". Journal of Classification. 24(1): 3-32.
Narang, S. K. and A. Ortega. (2009). "Lifting based wavelet transforms on graphs". In: Proc. Asia-Pacific Signal and Information Processing Association, 2009 Annual Summit and Conference. 441-444.
Narang, S. K. and A. Ortega. (2011). "Downsampling graphs using spectral theory". In: Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 4208-4211.
Narang, S. K. and A. Ortega. (2012). "Perfect reconstruction twochannel wavelet filter banks for graph structured data". IEEE Transactions on Signal Processing. 60(6): 2786-2799.
Ng, A. Y., M. I. Jordan, and Y. Weiss. (2002). "On spectral clustering: Analysis and an algorithm". In: Proc. Advances in Neural Information Processing Systems. 849-856.

Nguyen, H. Q. and M. N. Do. (2015). "Downsampling of signals on graphs via maximum spanning trees." IEEE Transactions on Signal Processing. 63(1): 182-191.
Nickel, M., V. Tresp, and H.-P. Kriegel. (2011). "A Three-Way Model for Collective Learning on Multi-Relational Data". In Proceedings of the 28th International Conference on Machine Learning: 809-816.
Novikov, A., D. Podoprikhin, A. Osokin, and D. P. Vetrov. (2015). "Tensorizing neural networks". In: Advances in Neural Information Processing Systems (NIPS). 442-450.
O'Rourke, S., V. Vu, and K. Wang. (2016). "Eigenvectors of random matrices: A survey". Journal of Combinatorial Theory, Series A. 144: 361-442.
Pan, S., R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang. (2018). "Adversarially regularized graph autoencoder for graph embedding". arXiv preprint arXiv:1802.0440\%.
Papalexakis, E. E., L. Akoglu, and D. Lence. (2013). "Do more views of a graph help? Community detection and clustering in multi-graphs". In Proceedings of the 16th International Conference on Information Fusion: 899-905.
Pasdeloup, B., V. Gripon, R. Alami, and M. G. Rabbat. (2019). "Uncertainty Principle on Graphs". In: Vertex-Frequency Analysis of Graph Signals. Springer. 317-340.
Pavez, E. and A. Ortega. (2016). "Generalized Laplacian precision matrix estimation for graph signal processing". In: Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2016. IEEE. 6350-6354.
Peralta, G. and A. Zareei. (2016). "A Network Approach to Portfolio Selection". Journal of Empirical Finance. 38(A): 157-180.
Perona, P. and W. Freeman. (1998). "A factorization approach to grouping". In: Prof. European Conference on Computer Vision. Springer. 655-670.
Perozzi, B., R. Al-Rfou, and S. Skiena. (2014). "Deepwalk: Online learning of social representations". In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 701-710.

Perraudin, N., B. Ricaud, D. I. Shuman, and P. Vandergheynst. (2018). "Global and local uncertainty principles for signals on graphs". APSIPA Transactions on Signal and Information Processing. 7(e3): 1-26.
Perraudin, N. and P. Vandergheynst. (2017). "Stationary signal processing on graphs". IEEE Transactions on Signal Processing. 65(13): 3462-3477.
Pourahmadi, M. (2011). "Covariance estimation: The GLM and regularization perspectives". Statistical Science: 369-387.
Puy, G., N. Tremblay, R. Gribonval, and P. Vandergheynst. (2018). "Random sampling of bandlimited signals on graphs". Applied and Computational Harmonic Analysis. 44(2): 446-475.
Qiu, H. and E. R. Hancock. (2007). "Clustering and embedding using commute times". IEEE Transactions on Pattern Analysis and Machine Intelligence. 29(11): 1873-1890.
Rabiei, H., F. Richard, O. Coulon, and J. Lefèvre. (2019). "Estimating the Complexity of the Cerebral Cortex Folding with a Local Shape Spectral Analysis". In: Vertex-Frequency Analysis of Graph Signals. Springer. 437-458.
Raffinot, T. (2017). "Hierarchical Clustering-Based Asset Allocation". The Journal of Portfolio Management. 44(2): 89-99.
Ray, S. S. (2012). Graph Theory with Algorithms and Its Applications: In Applied Science and Technology. Springer Science \& Business Media.
Ricaud, B. and B. Torrésani. (2014). "A survey of uncertainty principles and some signal processing applications". Advances in Computational Mathematics. 40(3): 629-650.
Rubinov, M. and O. Sporns. (2010). "Complex network measures of brain connectivity: Uses and interpretations". NeuroImage. 52(3): 1059-1069.
Rustamov, R. and L. J. Guibas. (2013). "Wavelets on graphs via deep learning". In: Advances in Neural Information Processing systems. 998-1006.
Sadhanala, V., Y.-X. Wang, and R. Tibshirani. (2016). "Graph sparsification approaches for Laplacian smoothing". In: Artificial Intelligence and Statistics. 1250-1259.

Saito, S., D. P. Mandic, and H. Suzuki. (2018). "Hypergraph p-Laplacian:
A differential geometry view". In: Proc. of the Thirty-Second AAAI Conference on Artificial Intelligence. 3984-3991.
Sakiyama, A. and Y. Tanaka. (2014). "Oversampled graph Laplacian matrix for graph filter banks". IEEE Transactions on Signal Processing. 62(24): 6425-6437.
Sakiyama, A., Y. Tanaka, T. Tanaka, and A. Ortega. (2019). "Eigendecompositionfree sampling set selection for graph signals". IEEE Transactions on Signal Processing. 67(10): 2679-2692.
Sandryhaila, A. and J. M. Moura. (2013). "Discrete signal processing on graphs". IEEE Transactions on Signal Processing. 61(7): 1644-1656.
Sandryhaila, A. and J. M. Moura. (2014a). "Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure". IEEE Signal Processing Magazine. 31(5): 80-90.
Sandryhaila, A. and J. M. Moura. (2014b). "Discrete signal processing on graphs: Frequency analysis". IEEE Transactions on Signal Processing. 62(12): 3042-3054.
Sardellitti, S., S. Barbarossa, and P. Di Lorenzo. (2017). "On the graph Fourier transform for directed graphs". IEEE Journal of Selected Topics in Signal Processing. 11(6): 796-811.
Scalzo, B., L. Stanković, A. G. Constantinides, and D. P. Mandic. (2020). "Portfolio cuts: A graph-theoretic framework to diversification". In: Proc. of the 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 8454-8458.
Scarselli, F., M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. (2008). "The graph neural network model". IEEE Transactions on Neural Networks. 20(1): 61-80.
Schaeffer, S. E. (2007a). "Graph Clustering". Computer Science Review. 1(1): 27-64.
Schaeffer, S. E. (2007b). "Graph clustering". Computer Science Review. 1(1): 27-64.
Scott, G. L. and H. C. Longuet-Higgins. (1990). "Feature grouping by relocalisation of eigenvectors of the proximity matrix." In: Proc. of the British Machine Vision Conference (BMVC). 1-6.

Segarra, S., A. G. Marques, G. Leus, and A. Ribeiro. (2015). "Interpolation of graph signals using shift-invariant graph filters". In: Proc. 23rd European Signal Processing Conference (EUSIPCO). 210-214.
Segarra, S., A. G. Marques, G. Mateos, and A. Ribeiro. (2016). "Blind identification of graph filters with multiple sparse inputs." In: Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 4099-4103.
Segarra, S., A. G. Marques, G. Mateos, and A. Ribeiro. (2017). "Network topology inference from spectral templates". IEEE Transactions on Signal and Information Processing over Networks. 3(3): 467-483.
Segarra, S. and A. Ribeiro. (2016). "Stability and continuity of centrality measures in weighted graphs". IEEE Transactions on Signal Processing. 64(3): 543-555.
Sen, P., G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. EliassiRad. (2008). "Collective classification in network data". AI Magazine. 29(3): 93-93.
Shi, J. and J. Malik. (2000). "Normalized cuts and image segmentation". Departmental Papers (CIS): 107.
Shuman, D. I., S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. (2013). "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains". IEEE Signal Processing Magazine. 30(3): 83-98.
Shuman, D. I., B. Ricaud, and P. Vandergheynst. (2012). "A windowed graph Fourier transform". In: Proc. IEEE Statistical Signal Processing Workshop (SSP). 133-136.
Shuman, D. I., B. Ricaud, and P. Vandergheynst. (2016). "Vertexfrequency analysis on graphs". Applied and Computational Harmonic Analysis. 40(2): 260-291.
Simon, H. A. (1962). "The Architecture of Complexity". In Proceedings of the American Philosophical Society.
Singh, R., A. Chakraborty, and B. Manoj. (2016). "Graph Fourier transform based on directed Laplacian". In: Proc. of the IEEE 2016 International Conference on Signal Processing and Communications (SPCOM). 1-5.

Slawski, M. and M. Hein. (2015). "Estimation of positive definite Mmatrices and structure learning for attractive Gaussian Markov random fields". Linear Algebra and Its Applications. 473: 145-179.
Spielman, D. A. and S. H. Teng. (2007a). "Spectral Partitioning Works: Planar Graphs and Finite Element Meshes". Linear Algebra and Its Applications. 421(2-3): 284-305.
Spielman, D. A. and N. Srivastava. (2011). "Graph sparsification by effective resistances". SIAM Journal on Computing. 40(6): 19131926.

Spielman, D. A. and S.-H. Teng. (2007b). "Spectral partitioning works: Planar graphs and finite element meshes". Linear Algebra and Its Applications. 421(2-3): 284-305.
Stankovic, L., D. Mandic, M. Dakovic, I. Kisil, E. Sejdic, and A. G. Constantinides. (2019a). "Understanding the basis of graph signal processing via an intuitive example-driven approach". IEEE Signal Processing Magazine, arXiv preprint arXiv:1903.11179, November.
Stankovic, L. (1997). "Highly concentrated time-frequency distributions: Pseudo quantum signal representation". IEEE Transactions on Signal Processing. 45(3): 543-551.
Stankovic, L., D. P. Mandic, M. Dakovic, I. Kisil, E. Sejdic, and A. G. Constantinides. (2019b). "Understanding the Basis of Graph Signal Processing via an Intuitive Example-Driven Approach [Lecture Notes]". IEEE Signal Processing Magazine. 36(6): 133-145.
Stankovic, L., D. P. Mandic, M. Dakovic, and I. Kisil. (2020). "Demystifying the coherence index in compressive sensing [Lecture Notes]". IEEE Signal Processing Magazine. 37(1): 152-162.
Stanković, L., M. Daković, D. Mandic, M. Brajović, B. Scalzo, and A. Constantinides. (2020a). "A low-dimensionality method for data-driven graph learning". In: Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 5340-5344.
Stanković, L. (2020). "The support uncertainty principle and the graph Rihaczek distribution: Revisited and improved". IEEE Signal Processing Letters. 27: 1030-1034.
Stanković, L., D. P. Mandic, M. Daković, M. Brajović, B. Scalzo Dees, and T. Constantinides. (2019a). "Graph Signal Processing - Part I: Graphs, Graph Spectra, and Spectral Clustering". arXiv:1907.03467.

Stanković, L. (2001). "A measure of some time-frequency distributions concentration". Signal Processing. 81(3): 621-631.
Stanković, L. (2015). Digital Signal Processing with Selected Topics. CreateSpace Independent Publishing Platform, An Amazon.com Company.
Stanković, L., M. Daković, and E. Sejdić. (2017a). "Vertex-frequency analysis: A way to localize graph spectral components [Lecture Notes]". IEEE Signal Processing Magazine. 34(4): 176-182.
Stanković, L., M. Daković, and T. Thayaparan. (2014). Time-Frequency Signal Analysis with Applications. Artech House.
Stanković, L., M. Daković, and E. Sejdić. (2019b). "Introduction to graph signal processing". In: Vertex-Frequency Analysis of Graph Signals. Springer. 3-108.
Stanković, L., M. Daković, and E. Sejdić. (2019c). "Vertex-frequency energy distributions". In: Vertex-Frequency Analysis of Graph Signals. Ed. by L. Stanković and E. Sejdić. Springer. 377-415.
Stanković, L., D. Mandic, M. Daković, B. Scalzo, M. Brajović, E. Sejdić, and A. G. Constantinides. (2020b). "Vertex-frequency graph signal processing: A comprehensive review". Digital Signal Processing: 102802.

Stanković, L. and E. Sejdić. (2019). Vertex-Frequency Analysis of Graph Signals. Springer.
Stanković, L., E. Sejdić, and M. Daković. (2017b). "Vertex-frequency energy distributions". IEEE Signal Processing Letters. 25(3): 358362.

Stanković, L., E. Sejdić, and M. Daković. (2018a). "Reduced interference vertex-frequency distributions". IEEE Signal Processing Letters. 25(9): 1393-1397.
Stanković, L., E. Sejdić, S. Stanković, M. Daković, and I. Orović. (2018b). "A tutorial on sparse signal reconstruction and its applications in signal processing". Circuits, Systems, and Signal Processing: 1-58.
Stoer, M. and F. Wagner. (1997). "A simple min-cut algorithm". Journal of the $A C M$ (JACM). 44(4): 585-591.

Tammen, M., I. Kodrasi, and S. Doclo. (2018). "Complexity reduction of eigenvalue decomposition-based diffuse power spectral density estimators using the power method". In: Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 451-455.
Tanaka, Y. and A. Sakiyama. (2014). "M-channel oversampled graph filter banks". IEEE Transactions Signal Processessing. 62(14): 35783590.

Tanaka, Y. and Y. C. Eldar. (2019). "Generalized sampling on graphs with subspace and smoothness priors". arXiv preprint arXiv:1905.04441.
Tanaka, Y. and Y. C. Eldar. (2020). "Generalized sampling on graphs with subspace and smoothness priors". IEEE Transactions on Signal Processing. 68: 2272-2286.
Tanaka, Y. and A. Sakiyama. (2019). "Oversampled transforms for graph signals". In: Vertex-Frequency Analysis of Graph Signals. Springer. 223-254.
Tang, J., M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. (2015). "Line: Large-scale information network embedding". In: Proceedings of the 24th International Conference on World Wide Web. 1067-1077.
Tang, L. and H. Liu. (2011). "Leveraging social media networks for classification". Data Mining and Knowledge Discovery. 23(3): 447478.

Tang, W., Z. Lu, and I. S. Dhillon. (2009). "Clustering with multiple graphs". In Proceedings of Ninth IEEE International Conference on Data Mining: 1016-1021.
Tepper, M. and G. Sapiro. (2016). "A short-graph Fourier transform via personalized pagerank vectors". In: Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 4806-4810.
Thanou, D., D. I. Shuman, and P. Frossard. (2014). "Learning Parametric Dictionaries for Signals on Graphs". IEEE Transactions Signal Processessing. 62(15): 3849-3862.
Thanou, D., X. Dong, D. Kressner, and P. Frossard. (2017). "Learning heat diffusion graphs". IEEE Transactions on Signal and Information Processing over Networks. 3(3): 484-499.
Transport for London. URL: https://tfl.gov.uk/.

Tremblay, N. and P. Borgnat. (2016). "Subgraph-based filterbanks for graph signals". IEEE Transactions Signal Processing. 64(15): 38273840.

Tremblay, N. and A. Loukas. (2020). "Approximating spectral clustering via sampling: A review". In: Sampling Techniques for Supervised or Unsupervised Tasks. Springer. 129-183.
Trevisan, L. (2013). "Lecture notes on expansion, sparsest cut, and spectral graph theory".
Tsitsvero, M., S. Barbarossa, and P. Di Lorenzo. (2016). "Signals on graphs: Uncertainty principle and sampling". IEEE Transactions Signal Processing. 64(18): 539-554.
Ubaru, S., J. Chen, and Y. Saad. (2017). "Fast Estimation of $\operatorname{tr}(\mathrm{f}(\mathrm{A}))$ via Stochastic Lanczos Quadrature". SIAM Journal on Matrix Analysis and Applications. 38(4): 1075-1099.
Van Dam, E. R. and W. H. Haemers. (2003). "Which graphs are determined by their spectrum?" Linear Algebra and Its Applications. 373: 241-272.
van der Maaten, L. and G. Hinton. (2008). "Visualizing data using t-SNE". Journal of Machine Learning Research. 9(Nov): 2579-2605.

Veličković, P., G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. (2017). "Graph attention networks". arXiv preprint arXiv:1710.10903.

Venkitaraman, A., S. Chatterjee, and P. Händel. (2016). "Hilbert transform, analytic signal, and modulation analysis for graph signal processing". arXiv preprint arXiv:1611.05269.
Verma, A. and K. K. Bharadwaj. (2017a). "A comparative study based on tensor factorization and clustering techniques for community mining in heterogeneous social network". In Proceedings of the International Conference on Computing, Communication and Networking Technologies (ICCCNT). In Press: 1-6.
Verma, A. and K. K. Bharadwaj. (2017b). "Identifying community structure in a multi-relational network employing non-negative tensor factorization and GA k-means clustering". Wires: Data Mining and Knowledge Discovery. 7(1): 1-32.
Vetterli, M., J. Kovačević, and V. Goyal. (2014). Foundations of Signal Processing. Cambridge University Press.

Von Luxburg, U. (2007). "A tutorial on spectral clustering". Statistics and Computing. 17(4): 395-416.
Wagner, D. and F. Wagner. (1993). "Between min cut and graph bisection". In: Proc. of the International Symposium on Mathematical Foundations of Computer Science. Springer. 744-750.
Wai, H.-T., Y. C. Eldar, A. E. Ozdaglar, and A. Scaglione. (2019). "Community Inference from Graph Signals with Hidden Nodes". In: ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE. 4948-4952.
Wainwright, M. J., M. I. Jordan, et al. (2008). "Graphical models, exponential families, and variational inference". Foundations and Trends ${ }^{(®)}$ in Machine Learning. 1(1-2): 1-305.
Wang, H., J. Wang, J. Wang, M. Zhao, W. Zhang, F. Zhang, X. Xie, and M. Guo. (2017). "GraphGAN: Graph representation learning with generative adversarial nets". arXiv preprint arXiv:1711.08267.
Wang, X., J. Chen, and Y. Gu. (2016). "Local measurement and reconstruction for noisy bandlimited graph signals". Signal Processing. 129: 119-129.
Wang, X., P. Liu, and Y. Gu. (2015). "Local-set-based graph signal reconstruction". IEEE Transactions on Signal Processing. 63(9): 2432-2444.
Wang, Z., A. C. Bovik, H. R. Sheikh, E. P. Simoncelli, et al. (2004). "Image quality assessment: from error visibility to structural similarity". IEEE Transactions on Image Processing. 13(4): 600-612.
Wang, Z., E. P. Simoncelli, and A. C. Bovik. (2003). "Multiscale structural similarity for image quality assessment". In: Proc. of the ThritySeventh Asilomar Conference on Signals, Systems $\mathcal{G}$ Computers. Vol. 2. 1398-1402.
Weiss, Y. (1999). "Segmentation using eigenvectors: A unifying view". In: Proceedings of the Seventh IEEE International Conference on Computer Vision. Vol. 2. IEEE. 975-982.
Wu, Z., S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu. (2019). "A comprehensive survey on graph neural networks". arXiv preprint arXiv:1901.00596.
Xu, Y. L. and D. P. Mandic. (2020). "Recurrent graph tensor networks". arXiv preprint arXiv:2009.08727. Sept. arXiv: 2009.08727 [cs.LG].

Xu, Y. L., K. Konstantinidis, and D. P. Mandic. (2020). "Multi-graph tensor networks". In: Advances in Neural Information Processing Systems.
Yan, X., B. M. Sadler, R. J. Drost, P. L. Yu, and K. Lerman. (2017). "Graph filters and the z-Laplacian". IEEE Journal of Selected Topics in Signal Processing. 11(6): 774-784.
Yankelevsky, Y. and M. Elad. (2016). "Dual graph regularized dictionary learning". IEEE Transactions on Signal and Information Processing over Networks. 2(4): 611-624.
You, J., R. Ying, X. Ren, W. L. Hamilton, and J. Leskovec. (2018). "GraphRNN: Generating realistic graphs with deep auto-regressive models". arXiv preprint arXiv:1802.08773.
Yu, W., C. Zheng, W. Cheng, C. C. Aggarwal, D. Song, B. Zong, H. Chen, and W. Wang. (2018). "Learning deep network representations with adversarially regularized autoencoders". In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery E Data Mining. 2663-2671.
Yuan, M. and Y. Lin. (2006). "Model selection and estimation in regression with grouped variables". Journal of the Royal Statistical Society: Series B (Statistical Methodology). 68(1): 49-67.
Yuan, M. and Y. Lin. (2007). "Model selection and estimation in the Gaussian graphical model". Biometrika. 94(1): 19-35.
Zhang, C., D. Florêncio, and P. A. Chou. (2015). "Graph signal processing - A probabilistic framework". Microsoft Research, Redmond, WA, USA, Tech. Rep. MSR-TR-2015-31.
Zhang, J., X. Shi, J. Xie, H. Ma, I. King, and D.-Y. Yeung. (2018a). "Gaan: Gated attention networks for learning on large and spatiotemporal graphs". arXiv preprint arXiv:1803.07294.
Zhang, J., X. Shi, S. Zhao, and I. King. (2019a). "STAR-GCN: Stacked and Reconstructed Graph Convolutional Networks for Recommender Systems". arXiv preprint arXiv:1905.13129.
Zhang, M. and Y. Chen. (2018). "Link prediction based on graph neural networks". In: Advances in Neural Information Processing Systems. 5165-5175.

Zhang, M., S. Jiang, Z. Cui, R. Garnett, and Y. Chen. (2019b). "D-VAE: A variational autoencoder for directed acyclic graphs". In: Advances in Neural Information Processing Systems. 1588-1600.
Zhang, Z., P. Cui, and W. Zhu. (2018b). "Deep learning on graphs: A survey". arXiv preprint arXiv:1812.04202.
Zhao, T., H. Liu, K. Roeder, J. Lafferty, and L. Wasserman. (2012). "The huge package for high-dimensional undirected graph estimation in R". Journal of Machine Learning Research. 13(Apr): 1059-1062. Zheng, M., J. Bu, C. Chen, C. Wang, L. Zhang, G. Qiu, and D. Cai. (2011). "Graph regularized sparse coding for image representation". IEEE Transactions on Image Processing. 20(5): 1327-1336.
Zheng, X.-W., Y. Y. Tang, J.-T. Zhou, H.-L. Yuan, Y.-L. Wang, L.-N. Yang, and J.-J. Pan. (2016). "Multi-windowed graph Fourier frames". In: Proc. IEEE International Conference on Machine Learning and Cybernetics (ICMLC). Vol. 2. 1042-1048.
Zhou, D., J. Huang, and B. Schölkopf. (2007). "Learning with hypergraphs: Clustering, classification, and embedding". In: Advances in Neural Information Processing Systems. 1601-1608.
Zhou, J., G. Cui, Z. Zhang, C. Yang, Z. Liu, and M. Sun. (2018). "Graph neural networks: A review of methods and applications". arXiv preprint arXiv:1812.08434.
Zhu, X. J. (2005). "Semi-supervised learning literature survey". Tech. rep. University of Wisconsin-Madison Department of Computer Sciences.

## Index

active vertices, 399
adjacency matrix, 9
characteristic polynomial, 30
eigendecomposition, 32
properties, 19
spectrum, 33
aggregate sampling, 212
analytic signal on a graph, 191
balanced graph, 508
Bayesian variational inference, 446
betweeness vertex/edge, 23
bipartite graph, 14
block collaborative processing, 325
block matching (BM3D) algorithm, 328
brain connectivity graph, 89
canonical polyadic decomposition (CPD), 456
Cartesian product, 26
characteristic polynomial, 30
properties, 31

Chebyshev polynomial, 173
Chebyshev series, 176
Cheeger ratio cut, 58
circular (ring) graph, 17
closeness vitality, 474
clossenes of graphs, 79
clustering
link-based, 463
minimum cut, 484
normalized cut, 485
ratio cut, 485
spectral minimum cut, 486
coactivation matrix, 89
commute time, 109, 393
complete graph, 14
compressed sensing on graphs, 199, 205
connected graph, 23
convolution (generalized), 185, 186
correlation matrix, 332
cost of spanning tree, 23
covariance matrix, 332
curvature, 295
cut of a signed graph, 508
degree matrix, 12
diffusion (random walk)
shift operator, 152
diffusion (random walk) mapping, 111
diffusion clouds, 113
diffusion distance, 111
diffusion model, 433
dimensionality reduction, 65, 91
compressive sensing, 205
graph coarsening, 127
graph down-scaling, 119
graph sparsification, 121
Laplacian eigenmaps, 114
directed circular graph, 17
directed graph, 9
directed path graph, 17
edge expansion, 58
edges, 7
effective resitance, 124, 397
eigendecomposition, 36
algebraic multiplicity, 30
generalized eigenvectors, 73, 75
geometric multiplicity, 30
matrix powers, 39
of adjacency matrix, 29, 36
of graph Laplacian, 39
of graph products, 37
smoothness of eigenvectors, 59
eigenmaps, 91, 92
commute time, 73,111
diffusion (random walk), 73,
generalized Laplacian eigenvectors, 73
graph Laplacian, 73
normalized Laplacian, 73
Swiss roll graph, 104
equivalent filters, 172
unique, 172
Erdös-Renyi graph, 18
Euler path, 21
external sources, 379, 399
factor analysis model, 417
fast multi-graph tensor network (fMGTN), 467
Fiedler vector, 61, 76
filter bank on a graph, 221
filter design, 170
foreign exchange (FOREX) model, 469
Fourier analysis, 36, 47
Gaussian filter, 318
Gaussian graphical model, 416
Gaussian Markov random field, 410
Gaussian random signal, 346, 397
relation with electric circuits, 397
general multi-graph tensor network (gMGTN), 467
generalized eigenvectors, 75
GLASSO, see graphical LASSO
gradient descent as a diffusion process, 426
graph
weighted, 11
balanced, 508
bipartite, 14
circular (ring), 17
complete, 14
components, 23
connected, 23
directed, 9
directed circular, 17
directed path, 17
Erdös-Renyi, 18
Kuratowski, 16
lattice-structured, 453
multipartite, 16
path, 17
planar, 16
preferential attachment model, 18
regular, 16
signed, 507
sparsification, 121
star, 16
stochastic block, 18
transformations, 378
undirected, 9
unstructured, 461
unweighted, 11
graph $p$-Laplacian, 508
graph adjacency spectrum, 33
graph auto-encoder, 447
graph coarsening, 127
graph convolutional networks (GCN), 427
graph cut
Cheeger ratio, 58
minimum, 51
minimum flow, 52
normalized, 57
portfolio, 481
ratio, 56
sparisty, 58
graph diameter, 22
graph down-sampling, 119
graph financial asset allocation, 490
graph Fourier transform
adjacency matrix, 164
graph Laplacian, 178
graph Laplacian, 12
p-Laplacian, 508
directed graphs, 505
generalized, 343
normalized, 13
properties, 19
random-walk, 13
signed graph, 507
spectrum, 40
vertex-weighted, 14
graph lifting, 129
graph neural networks (GNN), 425
Bayesian variational inference, 446
diffusion process, 439
generative adversarial model (GAN), 445
gradient descent, 430
graph convolutional networks, 425
graph Fourier transform, 438
label propagation, 431
link prediction, 444
localization of graphs, 436
multi-graph tensor networks (MGTN), 464
recurrent style, 433
signal smoothness and Laplacian operator, 428
spatial, 436
spectral, 438
spectral filtering, 440
standard graph auto-encoder, 447
variational graph auto-encoder, 446
graph sampling
aggregate sampling, 212
bandlimited graph signals, 199
filter bank, 221
sparse signals, 203, 204, 215
graph separability, 56
graph signal, 150
analytic signal on a graph, 191
denoising, 194
nonstationary signals, 270
optimal denoising, 193
random signals, 237
sparsity driven denoising, 270
wide-sense stationary, 237
graph signal filtering, 166
block collaborative processing, 325
convolution, 187, 188, 246
diffusion, 233
graph wide-sense stationarity (GWSS), 239
low-pass, 232
low-pass filtering, 168
spectral domain, 227
Taubin's algorithm, 233
vertex domain, 241
vertex-varying, 286

Wiener filter, 240
graph signal frames, 290
graph signal shift, 158, 197
graph signals
tensorization, 454
graph sparsification, 121
cut-preserving, 121
of the connectivity matrix, 122
spectral, 123
uniform, 127
graph spectrogram, 290, 302
graph spectrum, $33,40,80,123$
graph topology, 7
graph topology learning
based on eigenvectors, 352
based on signal similarity, 323
difussion models, 419
external sources, 399
factor analysis, 417
Gaussian Markov random field, 413
Gaussian models, 410
generalized distance, 325
generalized Laplacian, 343
geometric distance, 314
GLASSO, 343, 350, 415
Laplacian energy condition, 342
LASSO, 335
link prediction, 496
maximum likelihood interpretation, 346
partial correlation model, 412
polynomial fitting, 355
precision matrix, 344,346
probabilistic generative models, 409
reduced dimensionality, 355
regression models, 415
similarity measure, 324
smoothess contraint, 339
sparsity, 354
structural similarity index, 329
graph wide-sense stationarity (GWSS), 239
graphical LASSO (GLASSO), 343, 350, 368
maximum likelihood interpretation, 346
Newton method, 363
Haar wavelet, 229
Hamiltonian path, 21
heat kernel parameter, 314
heat transfer, 383
hitting time, 393
hyper-linked pages, 385
hypergraph approach, 453
image clustering, 85
in-degree matrix, 506
incidence matrix, 429
indicator vector, 67,486
normalized Laplacian, 72
inner vertices, 399
inverse system, 178
$k$-means algorithm, 84
Kron reduction, 134
Kronecker (tensor) product, 26
Kronecker separability condition, 456
Kuratowski graph, 16

Lagrange multipliers, 70
Laplace-Beltrami operator, 314
Laplacian, see graph Laplacian
Laplacian eigenmaps, 91
least absolute shrinkage and selection operator (LASSO), 335, 365
least absolute shrinkage and selection operator (LASSO)
graphical LASSO, 343
length of walk, 19
linked-pages, 385
local graph Fourier transform, 245
band-pass functions, 255
binomial, 258
frame, 291
Hann spectral window, 261
inversion, 283
kernel-based inversion, 284
Meyer wavelet, 264
Parseval's tight frame, 291
polynomial approximation, 264
raised cosine transfer function, 261
reconstruction condition, 261, 263
signal adaptive, 263
spectral, 254
low-pass filtering, 168, 181
diffusion, 232
marginal property
frequency, 301
vertex, 301
market graph, 483
matching, 131
edge weighted, 133
heavy edge, 132
maximal, 131
maximum, 132
random, 132
sorted heavy edge, 133
maximum-flow minimum-cut, 52
measurement matrix, 200
measurement vector, 200
metro traffic modelling, 471
minimal polynomial, 30
minimum graph cut, 51, 484
bounds, 71
minimum spanning tree, 23
Minnesota roadmap graph, 319
multi-graph tensor networks (MGTN), 464
multi-relational graphs
tensor representation, 462
multipartite graph, 16
multirelational graphs, 454
multivariate data, 324, 331
neighborhood, 20
Netflix recommender system, 455
Newton method, 363
normalized cut, 57, 485
normalized Laplacian, 13
number of walks, 20
optimal denoising, 193
out-degree matrix, 506
pageRank, 120, 386
Parseval's theorem, 193
path, 21
path graph, 17
perfect reconstruction condition, 224
periodogram of a graph signal, 239
permutation matrix, 10
Perona-Freeman algorithm, 81
planar graph, 16
polynomail approximation
Chebyshev, 173
portfolio optimization
cut, 481
graph cut, 481
graph financial asset allocation, 490
hierarchical clustering based asset allocation, 483
hierarchical risk parity scheme, 483
market graph, 484
minimum cut, 484
minimum variance, 481
probabilistic generative models, 310
repeated cut, 488
power method, 499
power spectral density, 238
precision matrix, 333
maximum likelihood, 346
preferential attachment model graph, 18
principal component analysis (PCA), 104
Pytorch implementation, 442
quadratic mirror filter, 226
random degree vertex selection, 120
random edge selection, 120
random graph signals, 237
random vertex selection, 119
random walk, 389
edge-centric, 390
vertex-centric, 389
random-walk analysis, 120
random-walk Laplacian, 13
ratio cut, 56, 485
Rayleigh quotient, 69
reconstruction conditions, 208
coherence index, 210
restricted isometry property, 209
recovery condition, 201
recurrent graph tensor network, 465
regression analysis, 335
regular graph, 16
relative eigenvalue gap, 80
RESCAL decomposition, 463
resitive electrical circuit, 375
segmentation, 50
Shi-Malik algorithm, 80
shift operator
isometric, 242
on a graph, 187
spectral domain, 192
signal similarity, 323
signal simulation, 405
signed graph, 507
similarity measure, 324
simulation of graph signals
diffusion, 407
external sources, 405
graph shifts, 407
using eigenvectors, 407
singular value decomposition (SVD), 457
smoothness, 295
condition, 339
index, 59, 295
local, 295
of eigenvectors, 59
Rayleigh quotient, 69
social networks, 385, 464
spanning tree, 23
cost, 23
minimum, 23
sparsification, see graph sparsification
random degree vertex selection, 120
sparsity promotion, 196
spectral decomposition, 32
DFT, 36
matrix polynomials, 39
adjacency matrix, 32
bisection based minimum cut, 488
commute time, 111
diffusion, 111
Fourier analysis, 48
generalized Laplacian eigenvectors, 73
graph Laplacian, 39
graph products, 37
normalized Laplacian, 73
spectral distance, 126
spectral domain based shift, 241
spectral embedding, 65
spectral graph theory, 29
spectral graph wavelet transform, Taubin's algorithm, 233 268
inversion, 293
polynomial approximation, 273
reconstruction condition, 274
spectral manifold, 64
spectral ordering
adjacency matrix, 167
graph Laplacian, 180
spectral similarity, 62
spectral space, 62
spectral vector, 63
spectrogram on a graph, 302
spectrum, 33,40
spectrum folding, 46
spring-mass system, 383
star graph, 16
steepest descent method, 364
stochastic block graph, 18
subgraph, 24
subsampling
bandlimited graph signals, 199
filter bank, 221
optimal sampling strategy, 215
sparse signals, 203
support matrix, 204
super-vertex, 131
Swiss roll graph, 104, 314
system on a graph, 150, 160
convolution, 185-187
equivalent systems, 161
graph Fourier transform, 164
graph Laplacian, 184
impulse response, 161
properties, 161
system coefficients, 160
tensor decomposition, 455
tensor network
fast multi-graph, 467
general multi-graph, 467
multi-graph, 464
recurrent graph, 465
tensor representation of multirelational graphs, 453
tensorization of graph signals
tensor representation of multirelational graphs, 454
tensors and graphs
canonical polyadic decomposition (CPD), 456
connectivity, 457
DFT, 460
fiber, 458
multi-graph tensor network (MGNT), 496
time-varying signals, 231
traffic networks
betweeness centrality, 473
centrality, 472
closeness vitality, 474
coefficient of diffusivity, 475
flux, 475
metro network, 471
net passenger flow, 474
passenger flow, 475
transfer function
polynomial approximation, 173
system on a graph, 165
transport for London, 477
uncertainty principle, 288
support, 294, 299
undirected graph, 9
unweighted graph, 11
variational graph auto-encoder (VGAF 446
vertex, 7
vertex closeness centrality, 22
vertex clustering, 49
$k$-means algorithm, 84
based on graph topology, 50
based on more than one eigenvector, 81
brain connecticity graph, 89
closeness of segmented graphs, 79
collaborative data processing, 85
Fiedler vector, 76
generalized eigenvectors, 79
image clustering, 85
indicator vector, 67
minimum cut, 76
minimum-flow, 52
Minnesota roadmap graph, 88
normalized Laplacian, 79
quasi-optimal, 59
segmentation, 79
spectral, 76, 92
spectral methods, 58
vertex dimensionality, 64
reduction, 92
spectral embedding, 65
spectral manifold, 64
spectral space, 62
vertex distance, 21

Euclidean, 313
generalized measure, 325
vertex ordering, 61
vertex-frequency analysis, 243
Eertex-frequency distributions energy, 294
kernel, 248, 301
local graph Fourier transform, 245
local smoothness, 245
marginal properties, 295
reduced interference (RID), 300
spectrogram, 290
vertex-varying filtering, 286
vertex-weighted Laplacian, 14
volume normalized cut, see normalized cut
walk on a graph, 19
length of, 19
number of, 20
weight matrix, 11
sparsity condition, 334
weighted graph, 11
wide-sense stationarity (WSS), 238
Wiener filter on a graph, 240, 328, see graph signal filetring
windows
graph signal analysis, 246
parameter optimization, 281
vertex domain, 274
$z$-transform, 165, 188


[^0]:    All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, mechanical, photocopying, recording or otherwise, without prior written permission of the publishers.

    Photocopying. In the USA: This journal is registered at the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923. Authorization to photocopy items for internal or personal use, or the internal or personal use of specific clients, is granted by now Publishers Inc for users registered with the Copyright Clearance Center (CCC). The 'services' for users can be found on the internet at: www.copyright.com

    For those organizations that have been granted a photocopy license, a separate system of payment has been arranged. Authorization does not extend to other kinds of copying, such as that for general distribution, for advertising or promotional purposes, for creating new collective works, or for resale. In the rest of the world: Permission to photocopy must be obtained from the copyright owner. Please apply to now Publishers Inc., PO Box 1024, Hanover, MA 02339, USA; Tel. +1 781 871 0245; www.nowpublishers.com; sales@nowpublishers.com
    now Publishers Inc. has an exclusive license to publish this material worldwide. Permission to use this content must be obtained from the copyright license holder. Please apply to now Publishers, PO Box 179, 2600 AD Delft, The Netherlands, www.nowpublishers.com; e-mail: sales@nowpublishers.com

[^1]:    ${ }^{1}$ The reason we do not directly use $p(\mathbf{v} \mid \mathbf{x})$ is due to the fact that the posterior is intractable when modelling the likelihood $p(\mathbf{x} \mid \mathbf{v})$ through neural nets. Thus, the variational method employs another distribution, $q(\mathbf{v} \mid \mathbf{x})$, and minimises its distance to $p(\mathbf{v} \mid \mathbf{x})$ by means of some tractable formats (i.e., the evidence lower bound, or ELBO for short).

