# Introduction to Graph Signal Processing

**Ljubiša Stanković, Miloš Daković, and Ervin Sejdić**

Published in: L. Stanković and E. Sejdić (eds), *Vertex-Frequency Analysis of Graph Signals*, Springer-Nature, 2019. ISBN 978-030-03573-0

## Contents

1 Introduction . . . 3
2 Graphs . . . 6
2.1 Basic Definitions . . . 6
2.2 Properties of Graphs and Associated Matrices . . . 10
2.3 Eigenvalue Decomposition of the Adjacency Matrix . . . 15
2.4 Eigenvalue Decomposition of the Laplacian . . . 21
2.5 Vertex Ordering, Coloring, and Segmentation . . . 25
3 Signals on Graphs . . . 28
3.1 Adjacency Matrix and Graph Signal Shift . . . 30
3.2 Systems Based on a Graph Shifted Signals . . . 32
3.3 Graph Fourier transform based on the Adjacency matrix . . . 33
3.4 Filtering in the Adjacency Matrix Spectral Domain . . . 35
3.4.1 Normalization . . . 35
3.4.2 Spectral Domain Filtering . . . 35
3.4.3 Spectral Ordering of the Adjacency Matrix Eigenvectors . . . 36
3.4.4 Spectral Domain Filter Design . . . 37
3.5 Graph Fourier Transform Based on the Laplacian . . . 41
3.6 Ordering and Filtering in the Laplacian Spectral Domain . . . 41
3.7 Systems on a Graph Defined Using the Laplacian . . . 44
3.8 Convolution of Signals on a Graph . . . 45
3.9 Graph $z$-transform of a Signal . . . 46
3.10 Shift in the Spectral Domain . . . 49
3.11 Parseval's Theorem on a Graph . . . 49
3.12 Optimal Denoising . . . 50
4 Subsampling, Compressed Sensing, and Reconstruction . . . 52
4.1 Subsampling of the Low-Pass Graph Signals . . . 52
4.2 Subsampling of the Sparse Graph Signals . . . 54

L. Stanković and M. Daković
University of Montenegro, Podgorica, Montenegro
E-mail: ljubisa@ac.me, milos@ac.me

E. Sejdić
University of Pittsburg, Pittsburg, PA, USA
E-mail: esejdic@pitt.edu

4.2.1 Known Coefficient Positions . . . 54
4.2.2 Support Matrices, Subsampling-Upsampling . . . 55
4.2.3 Unknown Coefficient Positions . . . 56
4.2.4 On the Unique Reconstruction Conditions . . . 57
4.3 Linear Combinations of Samples and Aggregated Sampling . . . 58
4.4 On the Sampling Strategies . . . 59
4.5 Filter Bank on a Graph . . . 61
5 Time-Varying Signals on Graphs . . . 68
5.1 Diffusion on Graph and Low Pass Filtering . . . 68
6 Random Graph Signals . . . 70
6.1 Adjacency Matrix Based Definition . . . 71
6.2 Spectral Domain Shift Based Definition of GWSS . . . 73
7 Examples of Graphs Topologies . . . 74
7.1 Geometric Graph Topologies . . . 74
7.2 Topology Based on the Signal Similarity . . . 76
7.3 Correlation Based Graph Laplacian learning . . . 78
7.4 Graph Laplacian Learning with the Smoothness Constraint . . . 80
7.5 Generalized Laplacian Learning . . . 81
7.6 Graph Topology with some a Priori Knowledge . . . 83
7.7 Graph Topology Based on the Eigenvectors . . . 85
7.8 Physically Well Defined Graphs . . . 86
7.8.1 Resistive Electrical Circuits . . . 86
7.8.2 Heat Transfer . . . 88
7.8.3 Spring-Mass Systems . . . 89
7.8.4 Social Networks and Linked Pages . . . 90
7.8.5 PageRank . . . 91
7.8.6 Random Walk . . . 92
7.9 Gaussian Random Signal . . . 93
8 Appendix . . . 93
8.1 Graph Signal Calculation Using Laplacian . . . 93
8.2 Random Signal Simulation on a Graph . . . 96
8.3 From the Newton Minimization to the Graphical LASSO . . . 98
8.3.1 Newton Method . . . 98
8.3.2 LASSO . . . 100
8.3.3 Graphical LASSO . . . 101
9 Conclusion . . . 103

**Abstract** Graph signal processing deals with signals whose domain, defined by a graph, is irregular. An overview of basic graph forms and definitions is presented first. Spectral analysis of graphs is discussed next. Some simple forms of processing signal on graphs, like filtering in the vertex and spectral domain, subsampling and interpolation, are given. Graph topologies are reviewed and analyzed as well. Theory is illustrated through examples, including few applications at the end of the chapter.

## 1 Introduction

Graph signal processing is an active research area in recent years resulting in many advanced solutions in various applications. In numerous practical cases the signal domain is not a set of equidistant instants in time or a set of points in space on a regular grid. The data sensing domain could be irregular and, in some cases, not related to the time or space. The data sensing domain is then related to other properties of the considered system/network. For example, in many social or web related networks, the sensing points and their connectivity are related to specific objects and their links. In some physical processes other properties than the space or time coordinates define the relation between points where the signal is sensed. Even for the data sensed in the well defined time and space domain, the introduction of new relations between the sensing points may produce new insights in the analysis and result in more advanced data processing techniques.

The data domain, in these cases, is defined by a graph. The graph consists of vertices, where the data values are defined/sensed, and the edges connecting these vertices. Graph exploit the fundamental relations among the data based on their relevant properties. Processing of signals whose sensing domains are defined by graphs resulted in graph data processing as an emerging field in big data signal processing today. This is a big step forward from the classical time (or space) series data analysis.

Here we will present one simplified example for graph signal analysis. Assume that we measure temperature in a geographical region. The temperature sensing points are chosen according to a plan and significance of specific areas (for example, according to the population density). They are illustrated in Fig. 1.

The distance between sensing points is proportional to their distances in Fig. 1(a). The measured temperature at each location is denoted by $x(n)$, Fig. 1(b). It is equal to $x(n) = s(n) + \varepsilon(n)$, where $s(n)$ is a temperature that would be obtained in the ideal measuring conditions and $\varepsilon(n)$ represents influence of the measurement micro-location, for example the proximity of trees, concrete structures, or ventilation. This part of the signal is called noise. We want to average the measured temperatures in order to reduce the influence of random noise to the temperature at a specific measurement point. Of course, if we average over a too large area around the considered sensing point we will lose the local temperature and produce a bias caused by very distant and probably

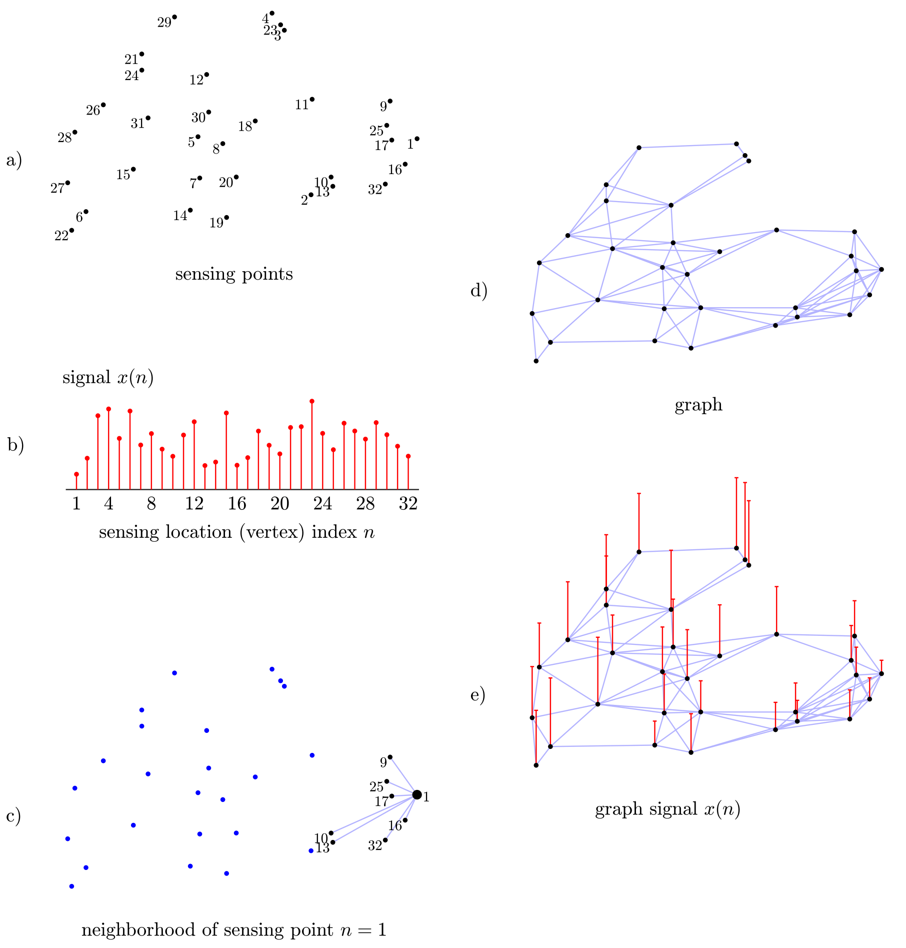


**Fig. 1** Graph and a signal on the graph illustration.

different temperatures. Therefore, we have to average temperatures from sensing points within a small area around each of the measurement points. From the sensing locations, we can see that some points have more dense structures around them, while around some of the sensing locations the structure of measurement points is sparse. Intuitively we can conclude that it would be the best to perform averaging over a local region of each point, taking measured values from its neighborhood, as

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

Graphical illustration of this calculation formula can be obtained by using lines that connect the considered point with its neighboring points. The sensing locations included in the calculation of $y(1)$ for the sensing point $n = 1$ are shown in Fig. 1(c). The matrix form of this calculation, for all sensing points, can be written as

$$\mathbf{y} = \mathbf{x} + \mathbf{A}\mathbf{x},$$

where matrix $\mathbf{A}$ indicates what neighboring measurement locations should be included for each $y(n)$ calculation. This matrix will be referred to as the connectivity or adjacency matrix. Sensing locations with corresponding connections are shown in Fig. 1(d). The sensing locations and their connectivity, along with the measured signal at each location, are presented in Fig. 1(e).

In the calculation, we may add weights for different measurement points,

$$y(n) = x(n) + \sum_{m \neq n} W_{nm} x(m).$$

The weights are higher for close points and smaller for distant points. They are zero outside a specified local area. This case can be represented by lines connecting the measurement points that are included in averaging each measurement point. Each line has its weight $W_{nm}$. A matrix form of the weighted averaging operation is

$$\mathbf{y} = \mathbf{x} + \mathbf{W}\mathbf{x}.$$

If we want the averaging to be appropriately scaled, in order to produce unbiased estimates, then the sum of all summation coefficients, for each $y(n)$, should be 1. It means that we can calculate

$$\mathbf{y} = \frac{1}{2}(\mathbf{x} + \mathbf{D}^{-1}\mathbf{W}\mathbf{x}),$$

where the elements of diagonal normalization matrix $\mathbf{D}$ are $D_{nn} = \sum_m W_{nm}$. This matrix is called the degree matrix.

The simple example presented here can be interpreted within the graph signal processing framework as follows:

– The measurement points are the graph vertices, Fig. 1(a).
– The lines indicating mutual relation and connectivity of the measurement points are the graph edges.
– The vertices and edges form a graph, Fig. 1(d). The graph is now the signal domain. It indicates the points where the signal is measured and how these sensing points are related to each other. The graph is then used for the analysis and processing of the measured data.
– The measured temperatures are the signal samples on this graph. They represent a graph signal Fig. 1(e). This signal may have many realizations on the same graph. It can include noise.
– The presented local averaging, taking into account the location of measurement points, that is, taking into account the graph structure, is one simple graph signal processing algorithm (linear first-order graph system).

Now we can use this framework for many different scenarios. For example, we can perform an opinion poll among the members of a social network. The members of a social network are the vertices (measurement points). Their friendship relations are the edges, representing graph connectivity. The answers they give are the graph signal values. This is then the basis for various graph signal processing algorithms.

The graph-based data processing approach can be applied not only to technological, biological, and social networks but its application has also lead to improvements and new methods in classical signal processing. In these cases the classical signal domain (that may be represented as a linear or circular graph) is structured in a more advanced way, considering the sensing points connectivity from their properties or signal similarity points of view.

In graph signal processing, the first step is to define the graph as a signal domain. While the data sensing points (graph vertices) are commonly well defined, that is not the case with their connectivity (graph edges). In some applications, the vertex connectivity is naturally defined, resulting in an exact underlying graph topology, such as the various computer, social, road, transportation, and electrical networks. For some other cases, the data domain definition in a graph form is a part of the problem itself. The vertex connectivity is not defined in advance. It can be determined based on the properties of the sensing positions or the acquired data. The definition of appropriate graph structure is of crucial importance for a meaningful and efficient application of the graph signal processing approach.

In this chapter, we will review the basic definitions and properties of graphs. Next, the signal on a graph and basic signal processing techniques in vertex and spectral domains are described. In the third part, some graph topologies are reviewed and discussed.

## 2 Graphs

Graph theory as a branch in mathematics has existed for almost three centuries. It has been used for a long time in chemistry, operational research, engineering, social networks, and computer sciences. The beginning of graph theory applications in electrical engineering dates back to the mid-XIX century when Kirchoff's laws were defined. Recently, graph theory has been a rapidly developing application and research area in signal processing. A short review of the basic graph definitions and properties will be presented in this section [1–23].

### 2.1 Basic Definitions

A graph is defined as a set of vertices $\mathcal{V}$ and set of edges $\mathcal{B} \subset \mathcal{V} \times \mathcal{V}$ connecting the vertices, where $\times$ is a direct product operation.

Examples of graphs with $N = 8$ vertices $\mathcal{V} = \{0, 1, 2, 3, 4, 5, 6, 7\}$ are presented in Fig. 2, along with the corresponding edges. The vertices are presented as points (circles) and the edges are presented as lines. A line between vertices

$n$ and $m$ means that $(m, n) \in \mathcal{B}$. The graph from Fig. 2(b) is described by

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

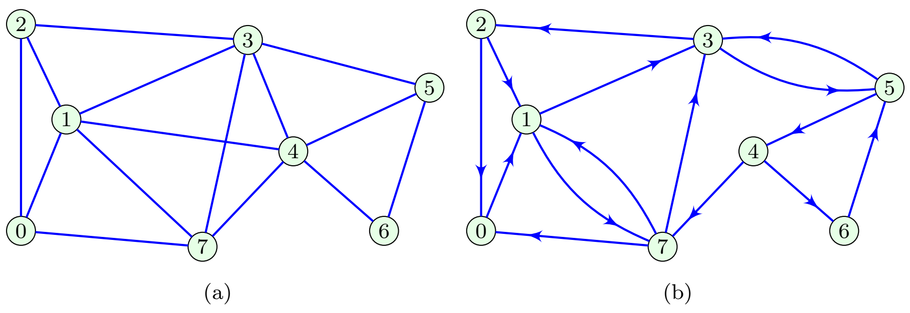


**Fig. 2** Examples of: (a) Undirected graph and (b) Directed graph.

A graph can be undirected and directed. In the case of undirected graphs, as in Fig. 2(a), it is assumed that the edge connecting the vertex $n$ to the vertex $m$ also connects the vertex $m$ to the vertex $n$. This means that if $(n, m) \in \mathcal{B}$ then $(m, n) \in \mathcal{B}$.

In general, this property does not hold for directed graphs. An example of a directed graph is shown in Fig. 2(b). The undirected graphs can be considered as a special case of directed graphs.

For a given set of vertices and edges, the graph can be represented by an adjacency matrix $\mathbf{A}$. This matrix describes the vertices connectivity. If there are $N$ vertices then $\mathbf{A}$ is an $N \times N$ matrix. The elements $A_{mn}$ of the adjacency matrix $\mathbf{A}$ assume values $A_{mn} \in \{0, 1\}$. The value $A_{mn} = 0$ is assigned if the vertices $m$ and $n$ are not connected with an edge, and $A_{mn} = 1$ if these vertices are connected,

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

The adjacency matrices for the graphs from Fig. 2(a) and (b) are

$$\mathbf{A} = \begin{array}{c|cccccccc|} 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 2 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 3 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 4 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 5 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 6 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 7 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array}, \quad \mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{1}$$

respectively.

In the case of an undirected graph the adjacency matrix is symmetric

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

A graph is fully determined by its adjacency matrix, defined for a given set of vertices. If the vertex numbering is changed it will cause corresponding changes in the adjacency matrix. However vertices renumbering does not change the graph itself (these graphs are isomorphic). Relation between adjacency matrix of original and renumerated graphs is defined using a permutation matrix $\mathbf{P}$ as

$$\mathbf{A}_2 = \mathbf{P}\,\mathbf{A}_1\mathbf{P}^T. \tag{2}$$

If we change the vertex numbering as $[0,1,2,3,4,5,6,7] \rightarrow [3,2,4,5,1,0,6,7]$, the corresponding permutation and adjacency matrices of a graph isomorph to the graph presented in Fig. 2(a) are

$$\mathbf{P} = \begin{bmatrix} 0&0&0&1&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&1&0&0&0&0&0&0\\ 1&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&1 \end{bmatrix}, \; \mathbf{A}_2 = \begin{bmatrix} 0&1&1&1&1&0&0&1\\ 1&0&0&0&1&1&0&0\\ 1&0&0&1&1&0&1&1\\ 1&0&1&0&0&0&1&0\\ 1&1&1&0&0&1&0&1\\ 0&1&0&0&1&0&0&1\\ 0&0&1&1&0&0&0&0\\ 1&0&1&0&1&1&0&0 \end{bmatrix}. \tag{3}$$

Relation (2) can easily be checked for this example. A permutation matrix has exactly one value equal to 1 in each row and in each column.

In general, the edges can be weighted. If the weights of edges are defined, a weighted graph is obtained. The set of weights $\mathcal{W}$ corresponds to the set of edges $\mathcal{B}$. A weighted graph is a more general case than the unweighted graph. Commonly, it is assumed that the edge weights are nonnegative real numbers. If we associate weight 0 to the nonexisting edges then the graph can be described with a weight matrix $\mathbf{W}$ similar to the adjacency matrix $\mathbf{A}$. A nonzero element $W_{mn}$ describes an edge between the vertices $m$ and $n$ and the corresponding weight. The value $W_{mn} = 0$ means that there is not an edge between the vertices $m$ and $n$.

An example of a weighted undirected graph is presented in Fig. 3.

The weight matrix for this graph is

$$\mathbf{W} = \begin{bmatrix} 0&0.54&0.14&0&0&0&0&0.47\\ 0.54&0&0.63&0.35&0.30&0&0&0.31\\ 0.14&0.63&0&0.31&0&0&0&0\\ 0&0.35&0.31&0&0.54&0.43&0&0.13\\ 0&0.30&0&0.54&0&0.54&0.62&0.54\\ 0&0&0&0.43&0.54&0&0.37&0\\ 0&0&0&0&0.62&0.37&0&0\\ 0.47&0.31&0&0.13&0.54&0&0&0 \end{bmatrix}. \tag{4}$$

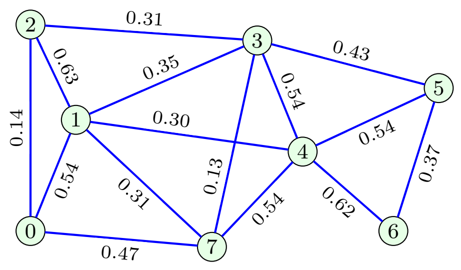


**Fig. 3** An example of a weighted graph.

In this manner the adjacency matrix $\mathbf{A}$ can be considered as a special case of the weight matrix $\mathbf{W}$ where all nonzero weights are equal to 1.

For undirected graphs the weighting matrix is symmetric,

$$\mathbf{W} = \mathbf{W}^T.$$

For directed graphs this property does not hold, in general.

A degree matrix for an undirected graph, denoted by $\mathbf{D}$, is a diagonal matrix where the diagonal elements $D_{mm}$ are equal to the sum of weights of all edges connected with the vertex $m$

$$D_{mm} = \sum_n W_{mn}.$$

In the case of an unweighted and undirected graph, the value of $D_{mm}$ is equal to the number of edges connected to the $m$-th vertex.

The degree matrix for the graph from Fig. 3 is

$$\mathbf{D} = \begin{bmatrix} 1.15 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2.13 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.08 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.76 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2.54 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.34 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.99 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.45 \end{bmatrix}. \tag{5}$$

The Laplacian matrix $\mathbf{L}$ of a graph is defined as

$$\mathbf{L} = \mathbf{D} - \mathbf{W}.$$

For an undirected graph the Laplacian matrix is symmetric $\mathbf{L} = \mathbf{L}^T$.

The Laplacian for the graph from Fig. 3 is

$$\mathbf{L} = \begin{bmatrix} 1.15 & -0.54 & -0.14 & 0 & 0 & 0 & 0 & -0.47 \\ -0.54 & 2.13 & -0.63 & -0.35 & -0.30 & 0 & 0 & -0.31 \\ -0.14 & -0.63 & 1.08 & -0.31 & 0 & 0 & 0 & 0 \\ 0 & -0.35 & -0.31 & 1.76 & -0.54 & -0.43 & 0 & -0.13 \\ 0 & -0.30 & 0 & -0.54 & 2.54 & -0.54 & -0.62 & -0.54 \\ 0 & 0 & 0 & -0.43 & -0.54 & 1.34 & -0.37 & 0 \\ 0 & 0 & 0 & 0 & -0.62 & -0.37 & 0.99 & 0 \\ -0.47 & -0.31 & 0 & -0.13 & -0.54 & 0 & 0 & 1.45 \end{bmatrix}. \quad (6)$$

The normalized Laplacian is defined as

$$\mathbf{L}_N = \mathbf{D}^{-1/2}(\mathbf{D} - \mathbf{W})\mathbf{D}^{-1/2}.$$

The normalized Laplacian for the graph from Fig. 3 is

$$\mathbf{L}_N = \begin{bmatrix} 1 & -0.35 & -0.13 & 0 & 0 & 0 & 0 & -0.36 \\ -0.35 & 1 & -0.42 & -0.18 & -0.13 & 0 & 0 & -0.18 \\ -0.13 & -0.42 & 1 & -0.22 & 0 & 0 & 0 & 0 \\ 0 & -0.18 & -0.22 & 1 & -0.26 & -0.28 & 0 & -0.08 \\ 0 & -0.13 & 0 & -0.26 & 1 & -0.29 & -0.39 & -0.28 \\ 0 & 0 & 0 & -0.28 & -0.29 & 1 & -0.32 & 0 \\ 0 & 0 & 0 & 0 & -0.39 & -0.32 & 1 & 0 \\ -0.36 & -0.18 & 0 & -0.08 & -0.28 & 0 & 0 & 1 \end{bmatrix}. \quad (7)$$

## 2.2 Properties of Graphs and Associated Matrices

1. For an undirected graph, the matrices **A**, **W**, and **L** are symmetric.

2. A graph is complete if there is an edge between each pair of vertices. The adjacency matrix of a complete graph has elements $A_{mn} = 1$ for all $n \neq m$ and $A_{nn} = 0$. An example of a complete graph is presented in Fig. 4(a).

3. Bipartite graph is a graph where the graph vertices $\mathcal{V}$ could be partitioned into two disjunct sets $\mathcal{E}$ and $\mathcal{H}$, $\mathcal{V} = \mathcal{E} \cup \mathcal{H}$ and $\mathcal{E} \cap \mathcal{H} = \emptyset$, such that there are no edges between vertices in the same set. If the vertex ordering is done in a such way that all vertices belonging to $\mathcal{E}$ are before vertices belonging to $\mathcal{H}$ then the adjacency matrix can be written as

$$\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{A}_{\mathcal{E}\mathcal{H}} \\ \mathbf{A}_{\mathcal{H}\mathcal{E}} & \mathbf{0} \end{bmatrix},$$

where the matrices $\mathbf{A}_{\mathcal{E}\mathcal{H}}$ and $\mathbf{A}_{\mathcal{H}\mathcal{E}}$ define the connections between the vertices in set $\mathcal{E}$ and set $\mathcal{H}$. For an undirected bipartite graph $\mathbf{A}_{\mathcal{E}\mathcal{H}} = \mathbf{A}_{\mathcal{H}\mathcal{E}}^T$. An example of a bipartite undirected graph is given in Fig. 4(b), with $\mathcal{E} = \{1, 2, 3\}$ and $\mathcal{H} = \{4, 5, 6, 7\}$. The graph in Fig. 4(b) is also a complete bipartite graph because all possible edges are present.

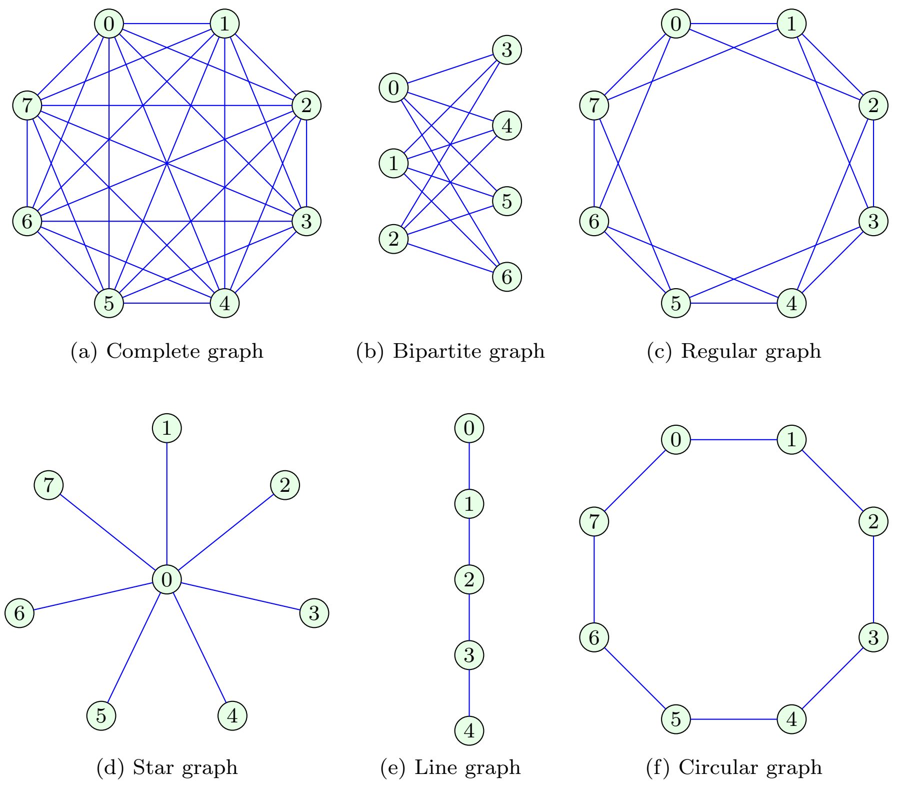


**Fig. 4** Special cases: (a) complete graph with 8 vertices, (b) complete bipartite graph, (c) regular graph where each vertex is connected to 4 vertices, (d) star graph, (e) line graph, (f) circular graph

4. An unweighted graph is regular (or $K$-regular) if all vertices are with the same degree $K$. An example of the regular graph with $K = 4$ is given in Fig. 4(c). The Laplacian and the normalized Laplacian of a $K$-regular graph are
$$\mathbf{L} = K\,\mathbf{I} - \mathbf{A} \text{ and } \mathbf{L}_N = \mathbf{I} - \frac{1}{K}\mathbf{A}.$$

5. A star graph has one central vertex that is connected to all other vertices. There are no other edges. An example of a star graph is given in Fig. 4(d). A star graph is also a complete bipartite graph where there is only one vertex in the first set.

6. A line graph is defined by a series of connected vertices. The first and the last vertex have a degree equal to 1, while all other vertices are with the degree 2. An example of a line graph with 5 vertices is presented in Fig. 4(e).

7. A graph is circular if each vertex has the degree 2. This graph is also a regular graph with $K = 2$. An example of a circular graph with 8 vertices is given in Fig. 4(f).

8. A walk between a vertex $n$ and a vertex $m$ is a connected sequence of edges and vertices that begins at the vertex $n$ and ends at the vertex $m$. Edges and vertices can be included into a walk more than once.
   The length of a walk is equal to the number of included edges.
   The number of walks between the vertex $n$ and the vertex $m$ of the length $K$ is equal to the element of matrix $\mathbf{A}^K$ in the $n$-th row and the $m$-th column.
   As an example consider vertex 1 and vertex 5 in the graph from Fig. 5. Consider the walks of length $K = 2$. There are two such walks ($1 \to 3 \to 5$ and $1 \to 4 \to 5$). The element in the second row and the sixth column of matrix $\mathbf{A}^2$ is 2 indicating that there are two walks between these vertices.

$$\mathbf{A}^2 = \begin{bmatrix} 3 & 2 & 1 & 3 & \mathbf{2} & 0 & 0 & 1 \\ 2 & 5 & 2 & 3 & 2 & 2 & 1 & 3 \\ 1 & 2 & 3 & 1 & 2 & 1 & 0 & 3 \\ 3 & 3 & 1 & 5 & 3 & 1 & 2 & 2 \\ 2 & 2 & 2 & 3 & 5 & 2 & 1 & 2 \\ 0 & 2 & 1 & 1 & 2 & 3 & 1 & 2 \\ 0 & 1 & 0 & 2 & 1 & 1 & 2 & 1 \\ 1 & 3 & 3 & 2 & 2 & 2 & 1 & 4 \end{bmatrix}. \quad (8)$$

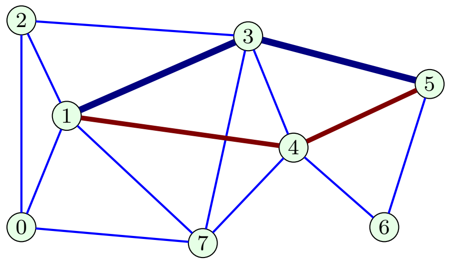


**Fig. 5** Walks of length $K = 2$ from vertex 1 to vertex 5.

9. The number of walks between the vertices $n$ and $m$ of the length not higher than $K$ is equal to the element of matrix

$$\mathbf{B}_K = \mathbf{A} + \mathbf{A}^2 + \cdots + \mathbf{A}^K$$

   in the $n$-th row and the $m$-th column.

10. The $K$-neighborhood of a vertex $n$ is defined as a set of vertices that are reachable from the vertex $n$ in up to $K$ length walks. It can be obtained as a set of positions of non-zero elements in the $n$-th row of matrix $\mathbf{B}_K$. The $K$-neighborhoods of vertex 0 for $K = 1$ and $K = 2$ are illustrated in Fig. 6.

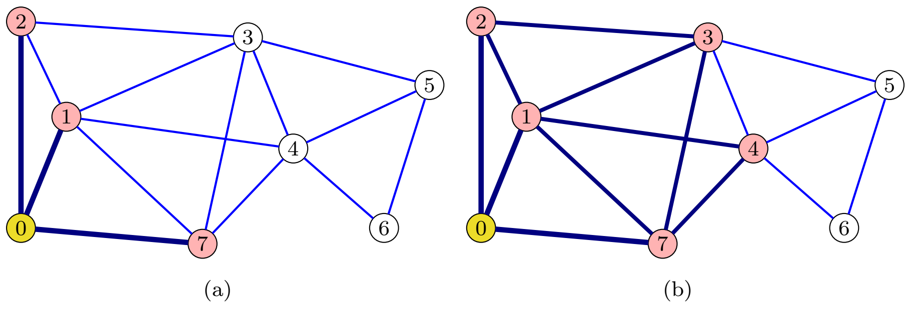


**Fig. 6** The $K$-neighborhoods of vertex 0 for: (a) $K = 1$ and (b) $K = 2$. The neighboring vertices are shaded.

11. Path is a walk where each vertex can be included only once. The path length is equal to the number of edges included in a path.

12. Distance between two vertices is equal to the minimal path length between them. The distance between vertex 1 and vertex 5 for the graph presented in Fig. 5 is 2.

13. The diameter $d$ of a graph is equal to the largest distance between all pairs of the vertices in the graph. The diameter of a complete graph is $d = 1$. For the graph presented in Fig. 5 its diameter is 3.

14. An undirected graph is connected if there exists a walk between each pair of its vertices.

15. If the graph is not connected, it consists of two or more connected graphs (components). The components represent disjoint graphs. Components produce a block diagonal form of the adjacency matrix and Laplacian. For $M$ components of a graph this form would be

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{A}_M \end{bmatrix} \quad \text{and} \quad \mathbf{L} = \begin{bmatrix} \mathbf{L}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{L}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{L}_M \end{bmatrix}.$$

Note that the block diagonal form is obtained only if the vertex numbering follows graph components.

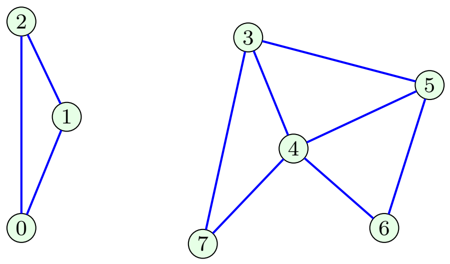


**Fig. 7** A disconnected graph.

As an example, let us consider a graph derived form Fig. 2(a) by removing some edges. This graph is presented in Fig. 7.
The adjacency matrix for this graph is

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{9}$$

with the corresponding Laplacian

$$\mathbf{L} = \begin{bmatrix} 2 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & -1 & -1 & 0 & -1 \\ 0 & 0 & 0 & -1 & 4 & -1 & -1 & -1 \\ 0 & 0 & 0 & -1 & -1 & 3 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & -1 & 2 & 0 \\ 0 & 0 & 0 & -1 & -1 & 0 & 0 & 2 \end{bmatrix}. \tag{10}$$

These matrices are in a block-diagonal form with two blocks.
If there is an isolated vertex in a graph, then the corresponding row and column of the matrices **A** and **L** will be zero-valued.

16. If we have two graphs defined on the same vertices, with adjacency matrices $\mathbf{A}_1$ and $\mathbf{A}_2$, we can define a sum of the graphs as a new graph with the adjacency matrix
$$\mathbf{A} = \mathbf{A}_1 + \mathbf{A}_2.$$
If we want to keep binary values $0, 1$ in the adjacency matrix then the logical (Boolean) summation rule $1 + 1 = 1$ should be used in the matrix addition. In this chapter we will use the arithmetic summation rule only.

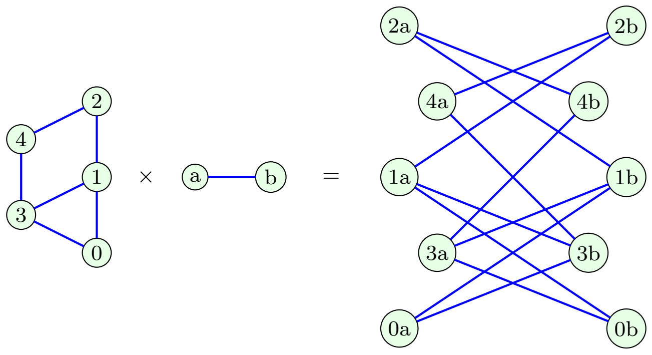


**Fig. 8** Kronecker (tensor) product of two graphs.

17. Kronecker (tensor) product of two disjoint graphs $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{B}_1)$ and $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{B}_2)$ is a graph $\mathcal{G} = (\mathcal{V}, \mathcal{B})$ where $\mathcal{V} = \mathcal{V}_1 \times \mathcal{V}_2$ and $\big((n_1, m_1), (n_2, m_2)\big) \in \mathcal{B}$ only if $(n_1, n_2) \in \mathcal{B}_1$ and $(m_1, m_2) \in \mathcal{B}_2$. The adjacency matrix of $\mathcal{G}$ is equal to the Kronecker product of adjacency matrices $\mathbf{A}_1$ and $\mathbf{A}_2$,

$$\mathbf{A} = \mathbf{A}_1 \otimes \mathbf{A}_2.$$

The Kronecker product of two simple graphs is illustrated in Fig. 8.

18. Cartesian product (graph product) of two disjoint graphs $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{B}_1)$ and $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{B}_2)$ is a graph $\mathcal{G} = (\mathcal{V}, \mathcal{B})$, where $\mathcal{V} = \mathcal{V}_1 \times \mathcal{V}_2$ and $\big((n_1, m_1), (n_2, m_2)\big) \in \mathcal{B}$ only if

$$m_1 = m_2 \text{ and } (n_1, n_2) \in \mathcal{B}_1 \text{ or}$$
$$n_1 = n_2 \text{ and } (m_1, m_2) \in \mathcal{B}_2.$$

The adjacency matrix of a Cartesian product of two graphs is

$$\mathbf{A} = \mathbf{A}_1 \otimes \mathbf{I}_{N_2} + \mathbf{I}_{N_1} \otimes \mathbf{A}_2,$$

where $\mathbf{A}_1$ and $\mathbf{A}_2$ are the adjacency matrices of graphs $\mathcal{G}_1$ and $\mathcal{G}_2$, respectively. The numbers of vertices in $\mathcal{G}_1$ and $\mathcal{G}_2$ are denoted by $N_1$ and $N_2$, respectively.
The Cartesian product of two simple graphs is illustrated in Fig. 9.

## 2.3 Eigenvalue Decomposition of the Adjacency Matrix

Graph matrices can be decomposed using the eigenvalue decomposition. A column vector $\mathbf{u}$ is an eigenvector of the adjacency matrix $\mathbf{A}$ if

$$\mathbf{Au} = \lambda \mathbf{u}$$

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

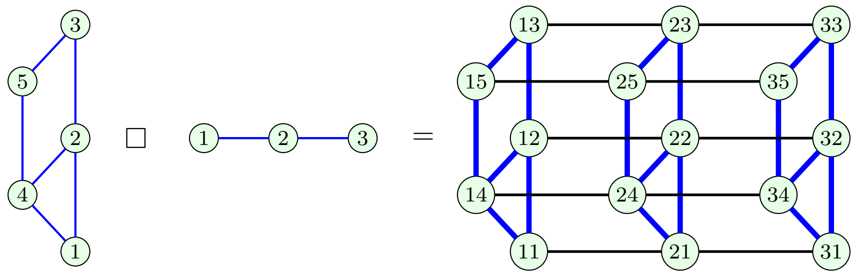


**Fig. 9** Cartesian product of two graphs

The previous relation can be written as $(\mathbf{A}-\lambda\mathbf{I})\mathbf{u}=\mathbf{0}$. A nontrivial solution for $\mathbf{u}$ exists if

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

The determinant $\det\|\mathbf{A}-\lambda\mathbf{I}\|$ is a polynomial of $\lambda$. It is called the characteristic polynomial of matrix $\mathbf{A}$,

$$P(\lambda)=\det\|\mathbf{A}-\lambda\mathbf{I}\|=\lambda^N+c_1\lambda^{N-1}+\cdots+c_N.$$

Order of the characteristic polynomial is equal to the number of vertices $N$. Eigenvalues are the roots of the characteristic polynomial, $P(\lambda)=0$. In general, there are $N$ eigenvalues $\lambda_0$, $\lambda_1$, $\dots$, $\lambda_{N-1}$. In some cases the eigenvalues can be repeated meaning that the zeros of algebraic multiplicity higher than one exist in the characteristic polynomial. Denote distinct eigenvalues as $\mu_1$, $\mu_2$, $\dots$, $\mu_{N_m}$, and their corresponding algebraic multiplicities as $p_1$, $p_2$, $\dots$, $p_{N_m}$, 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 be written in a form

$$P(\lambda)=(\lambda-\mu_1)^{p_1}(\lambda-\mu_2)^{p_2}\cdots(\lambda-\mu_{N_m})^{p_{N_m}}.$$

The minimal polynomial of the considered matrix is obtained from the characteristic polynomial by reducing algebraic multiplicities of each eigenvalue to 1. Its form is

$$P_{min}(\lambda)=(\lambda-\mu_1)(\lambda-\mu_2)\cdots(\lambda-\mu_{N_m}).$$

*Properties of the characteristic and minimal polynomial:*

– The characteristic polynomial order is equal to the number of vertices.
– For $\lambda=0$, $P(0)=\det(\mathbf{A})=(-\lambda_0)(-\lambda_1)\cdots(-\lambda_{N-1})$.
– A sum of the eigenvalues is equal to the sum of diagonal elements of matrix $\mathbf{A}$. It means that $c_1=0$ for the characteristic polynomial of the adjacency matrix.
– Coefficient $c_2$ in $P(\lambda)$ is equal to the number of edges multiplied by $-1$.

– Degree of the minimal polynomial $N_m$ is larger than the graph diameter. As an example consider a connected graph with only two distinct eigenvalues $\lambda_0$ and $\lambda_1$. The minimal polynomial order is 2. It means that the diameter of this graph is $d = 1$. This is a complete graph.
– For a connected graph the multiplicity of the largest eigenvalue is 1.

The characteristic polynomial of the adjacency matrix for the graph from Fig. 2(a) is

$$P(\lambda) = \lambda^8 - 15\lambda^6 - 18\lambda^5 + 33\lambda^4 + 60\lambda^3 + 16\lambda^2 - 6\lambda$$

with eigenvalues $(-2.1929, -1.7498, -1.3214, -0.7958, 0, 0.2039, 1.7963, 4.0597)$. The minimal polynomial is equal to characteristic polynomial $P_{min}(\lambda) = P(\lambda)$.

The characteristic polynomial of the adjacency matrix for the graph from Fig. 7 is

$$P(\lambda) = \lambda^8 - 10\lambda^6 - 8\lambda^5 + 24\lambda^4 + 34\lambda^3 + 3\lambda^2 - 12\lambda - 4$$

with eigenvalues $(-\frac{1+\sqrt{5}}{2}, -1.4728, -1, -1, -0.4626, \frac{\sqrt{5}-1}{2}, 2, 2.9354)$. There is an eigenvalue of multiplicity higher than 1. The minimal polynomial is

$$P_{min}(\lambda) = \lambda^7 - \lambda^6 - 9\lambda^5 + \lambda^4 + 23\lambda^3 + 11\lambda^2 - 8\lambda - 4.$$

If all eigenvalues are distinct (of algebraic multiplicity 1), instead of $N$ equations $\mathbf{A}\mathbf{u}_k = \lambda_k \mathbf{u}_k$, $k = 0, 1, \dots, N-1$, we can write one matrix equation for the adjacency matrix

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

or

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

where $\mathbf{\Lambda}$ is a diagonal matrix with eigenvalues on the diagonal and $\mathbf{U}$ is a matrix composed of eigenvectors $\mathbf{u}_k$ as columns. Since one coefficient of the eigenvector can be arbitrarily chosen, common choice is such that $\|\mathbf{u}_k\|_2^2 = 1$ for each $k = 0, 1, \dots, N-1$.

For an undirected graph, the matrix $\mathbf{A}$ is symmetric. The eigenvalues of a symmetric matrix are real-valued. For undirected graphs, if matrix $\mathbf{A}$ is diagonalizable then

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

All real symmetric matrices are diagonalizable. The adjacency matrix of an undirected graph is always diagonalizable. The square matrix is diagonalizable if all its eigenvalues are distinct (this condition is sufficient, but not necessary). For some directed graphs, when the eigenvalues of algebraic multiplicity higher than one exist, the matrix $\mathbf{A}$ may not be diagonalizable. In such cases the algebraic multiplicity is higher than the geometrical multiplicity of the considered eigenvalue and the Jordan normal form could be used.

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

For the graph presented in Fig. 2(a), the graph adjacency spectrum is given in Fig. 10. The spectrum of the graph adjacency matrix transformed with perturbation matrix (3) is given in Fig. 11. We can see that vertex renumbering with the perturbation matrix does not change the eigenvalues. The eigenvectors are also the same with coefficient reordering induced by the vertex renumbering.

### *The DFT basis functions as a special case of adjacency matrix eigenvectors*

The eigenvalue decomposition for the directed circular graph presented in Fig. 19 will follow from the definition $\mathbf{A}\mathbf{u}_k = \lambda_k \mathbf{u}_k$. For this particular graph it reduces to

$$u_k(n-1) = \lambda_k u_k(n),$$

where $u_k(n)$ are the elements of vector $\mathbf{u}_k$. A solution of this linear difference equation is

$$u_k(n) = \frac{1}{\sqrt{N}} e^{j2\pi nk/N} \text{ and } \lambda_k = e^{-j2\pi k/N},\ k = 0, 1, \dots, N-1. \qquad (11)$$

These eigenvectors correspond to the DFT basis function in this case.

### *Decomposition of matrix powers and polynomials*

The eigenvalue decomposition of the adjacency matrix $\mathbf{A}\mathbf{A} = \mathbf{A}^2$ is

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

assuming that $\mathbf{U}^{-1}$ exists, that is, that matrix $\mathbf{A}$ is diagonalizable.

This form can easily be generalized for an arbitrary integer power

$$\mathbf{A}^n = \mathbf{U}\mathbf{\Lambda}^n\mathbf{U}^{-1}.$$

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

$$f(\mathbf{A}) = h_0\mathbf{A}^0 + h_1\mathbf{A}^1 + h_2\mathbf{A}^2 + \dots + h_{N-1}\mathbf{A}^{N-1}$$

the eigenvalue decomposition is

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

The proof is evident using the matrix power and linearity properties.

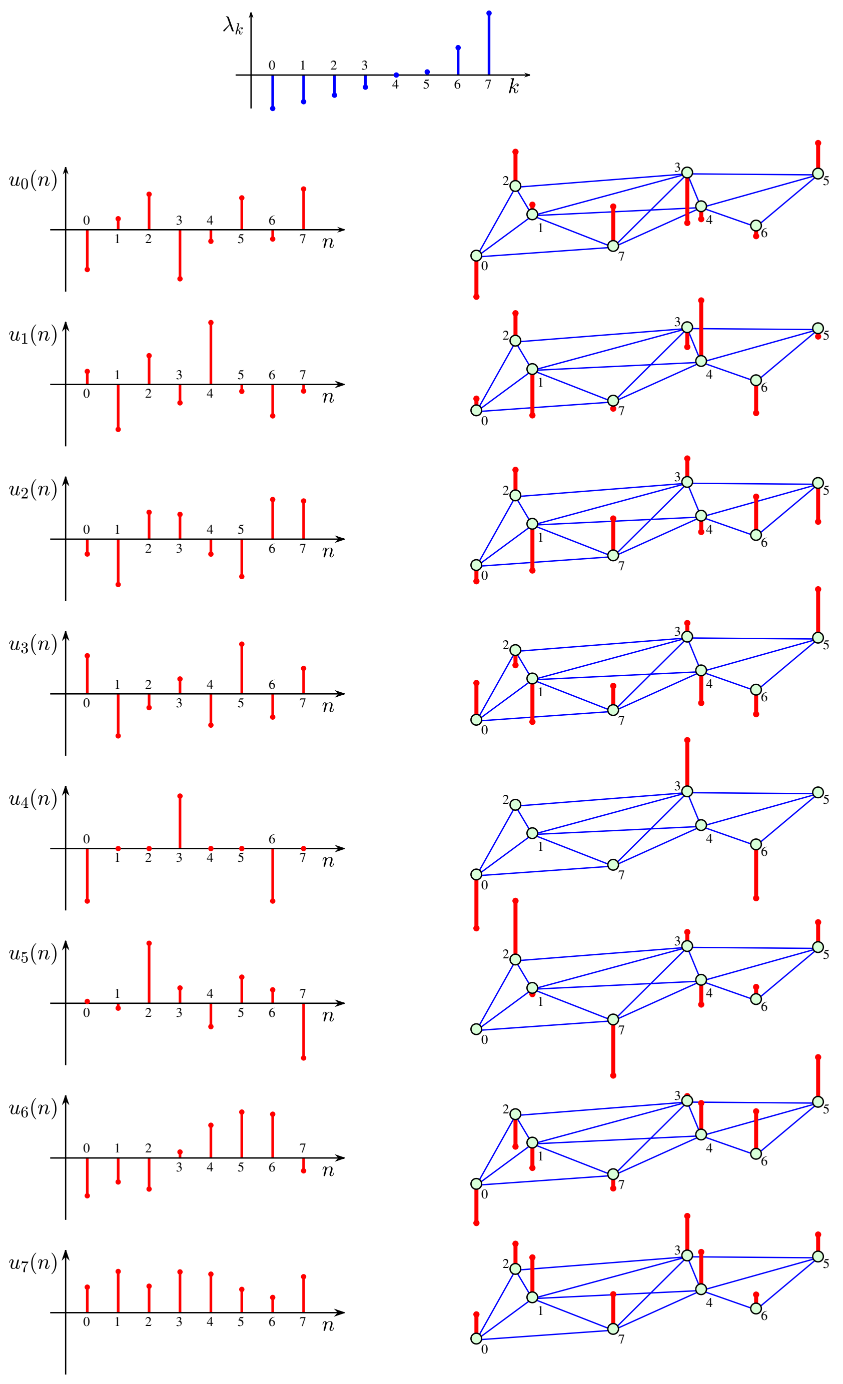


**Fig. 10** Eigenvalues $\lambda_k$ and corresponding eigenvectors $u_k(n)$ for the adjacency matrix of the graph presented in Fig. 2(a). The eigenvectors are shown on the vertex index line (left) and on the graph (right).

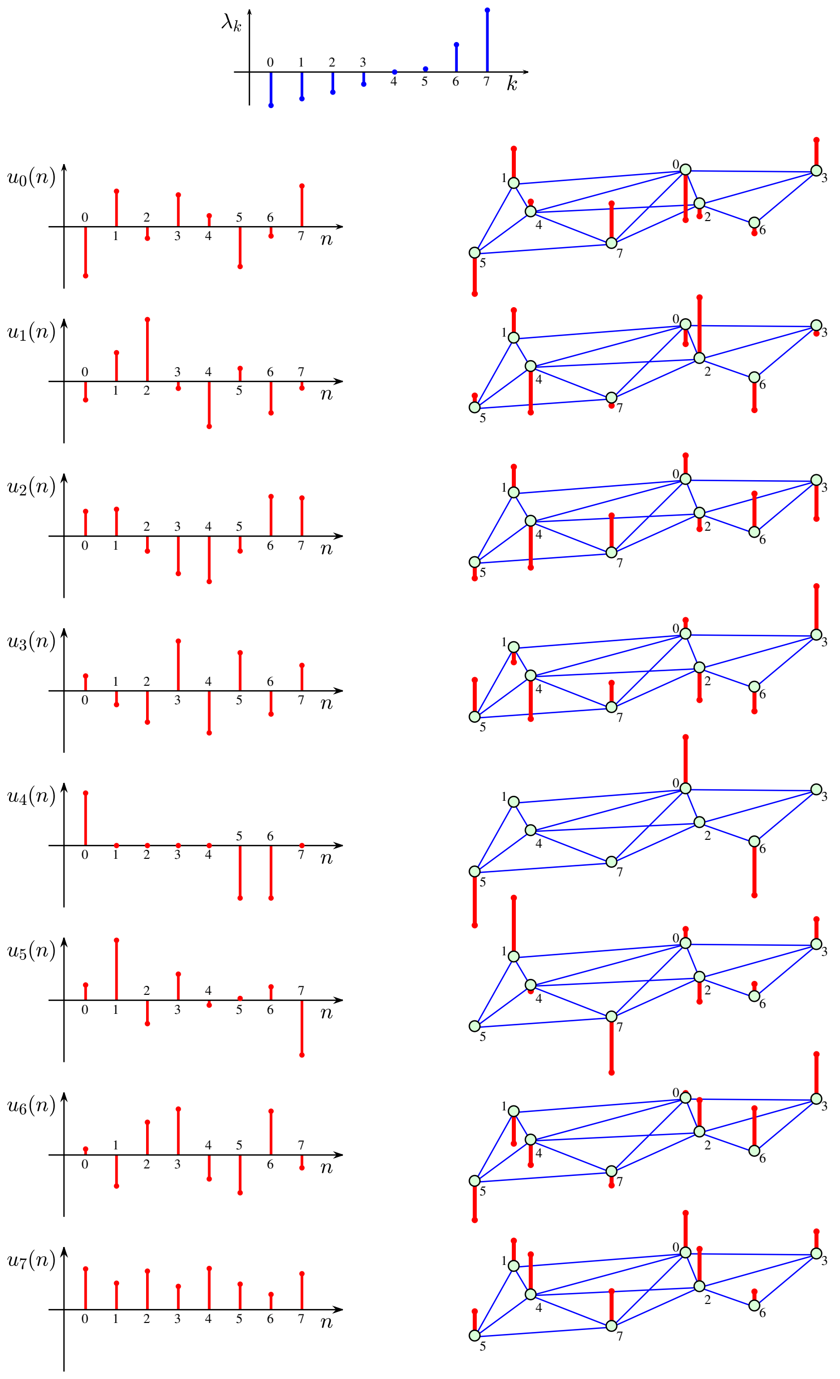


**Fig. 11** Eigenvalues $\lambda_k$ and corresponding eigenvectors $u_k(n)$ for the adjacency matrix of the graph presented in Fig. 2(a) with vertex renumbering $[0, 1, 2, 3, 4, 5, 6, 7] \rightarrow [3, 2, 4, 5, 1, 0, 6, 7]$.

## 2.4 Eigenvalue Decomposition of the Laplacian

Eigenvalue decomposition can be done for the Laplacian as well. Here we will use the same notation for the eigenvalues and eigenvectors, as general mathematical forms, although they are not related to the eigenvalues and eigenvectors of the adjacency matrix. For an undirected graph the Laplacian can be written as

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

where $\mathbf{\Lambda}$ is a diagonal matrix with the Laplacian eigenvalues and $\mathbf{U}$ is the matrix of its eigenvectors (as columns), with $\mathbf{U}^{-1} = \mathbf{U}^T$. Note that the Laplacian of an undirected graph is always diagonalizable since it is a real symmetric matrix.

Each eigenvector $\mathbf{u}_k$ of a Laplacian satisfies

$$\mathbf{L}\mathbf{u}_k = \lambda_k \mathbf{u}_k.$$

The set of the Laplacian eigenvalues is called the graph spectrum (or graph Laplacian spectrum).

The Laplacian spectrum of the graph from Fig. 2(a) is presented in Fig. 12, along with the corresponding eigenvectors. The Laplacian spectrum of the disconnected graph from Fig. 7 is plotted in Fig. 13.

*Properties of the Laplacian eigenvalue decomposition:*

– Since the Laplacian is defined in such a way that the sum of each row (column) elements is zero, then at least one eigenvalue of the Laplacian is zero with the corresponding eigenvector $\mathbf{u}_0 = [1, 1, \dots 1]^T/\sqrt{N} = \mathbf{1}/\sqrt{N}$. The relation $\mathbf{L}\mathbf{u}_0 = 0\,\mathbf{u}_0$ is always satisfied.

– Multiplicity of zero as an eigenvalue of the Laplacian is equal to the number of connected components in a graph. For example, if $\lambda_0 = \lambda_1 = 0$ then the graph is not connected. If $\lambda_2 > 0$ then there are two connected components in this graph.

– Sum of the eigenvalues is equal to the trace of the Laplacian matrix. For the normalized Laplacian the sum of its eigenvalues is equal to $N$ if there are no isolated vertices.

– Coefficient $c_N$ in the Laplacian characteristic polynomial

$$P(\lambda) = \det ||\mathbf{L} - \lambda\mathbf{I}|| = \lambda^N + c_1\lambda^{N-1} + \cdots + c_N$$

is equal to 0. Coefficient $c_1$ is equal to the number of edges multiplied by $-2$.

The characteristic polynomial of the Laplacian for graph from Fig. 2(a) is

$$P(\lambda) = \lambda^8 - 30\lambda^7 + 374\lambda^6 - 2500\lambda^5 + 9618\lambda^4 - 21106\lambda^3 + 24094\lambda^2 - 10712\lambda$$

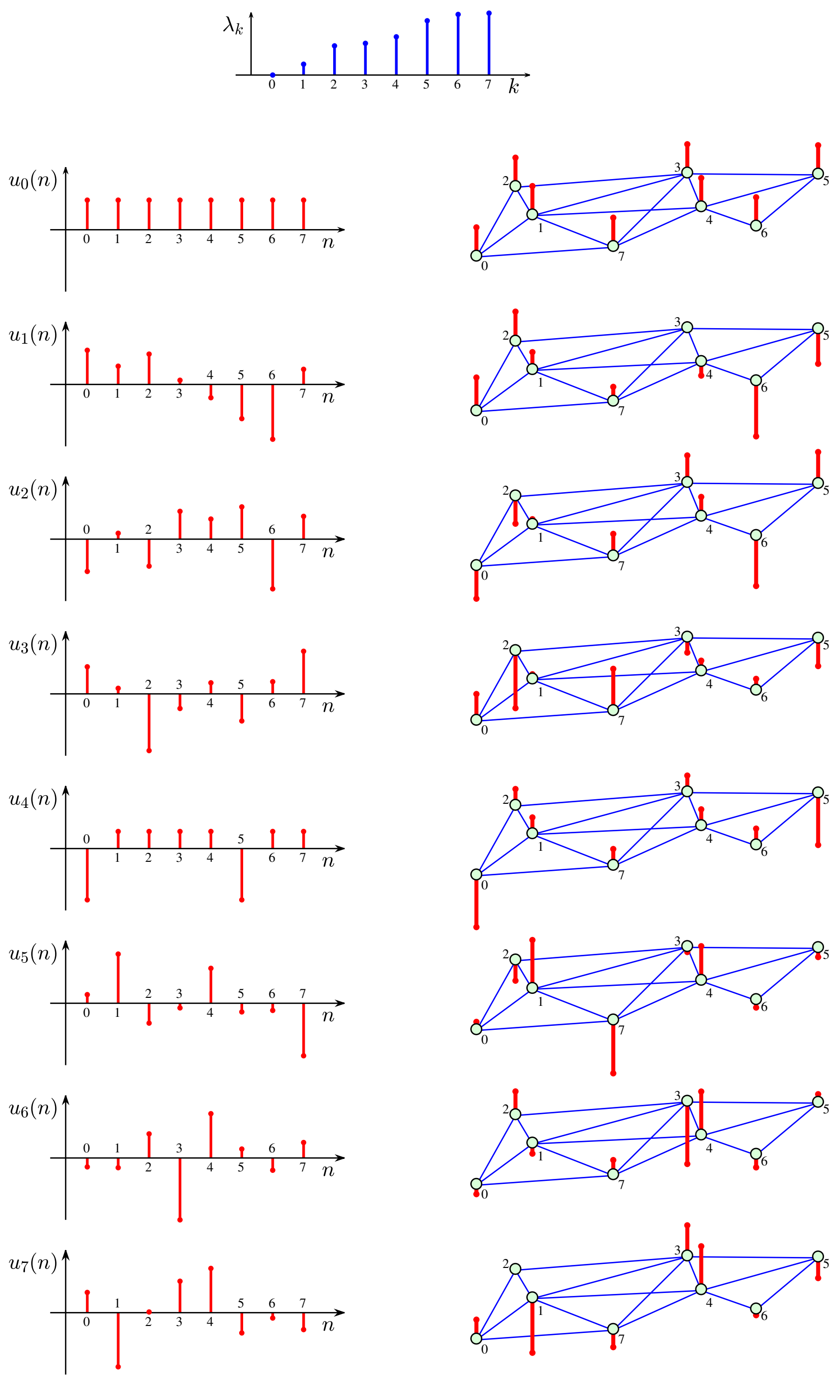


**Fig. 12** Eigenvalues $\lambda_k$ and corresponding eigenvectors $u_k(n)$ for the Laplacian of the graph presented in Fig. 2(a). The eigenvectors are shown on the vertex index line (left) and on the graph (right).

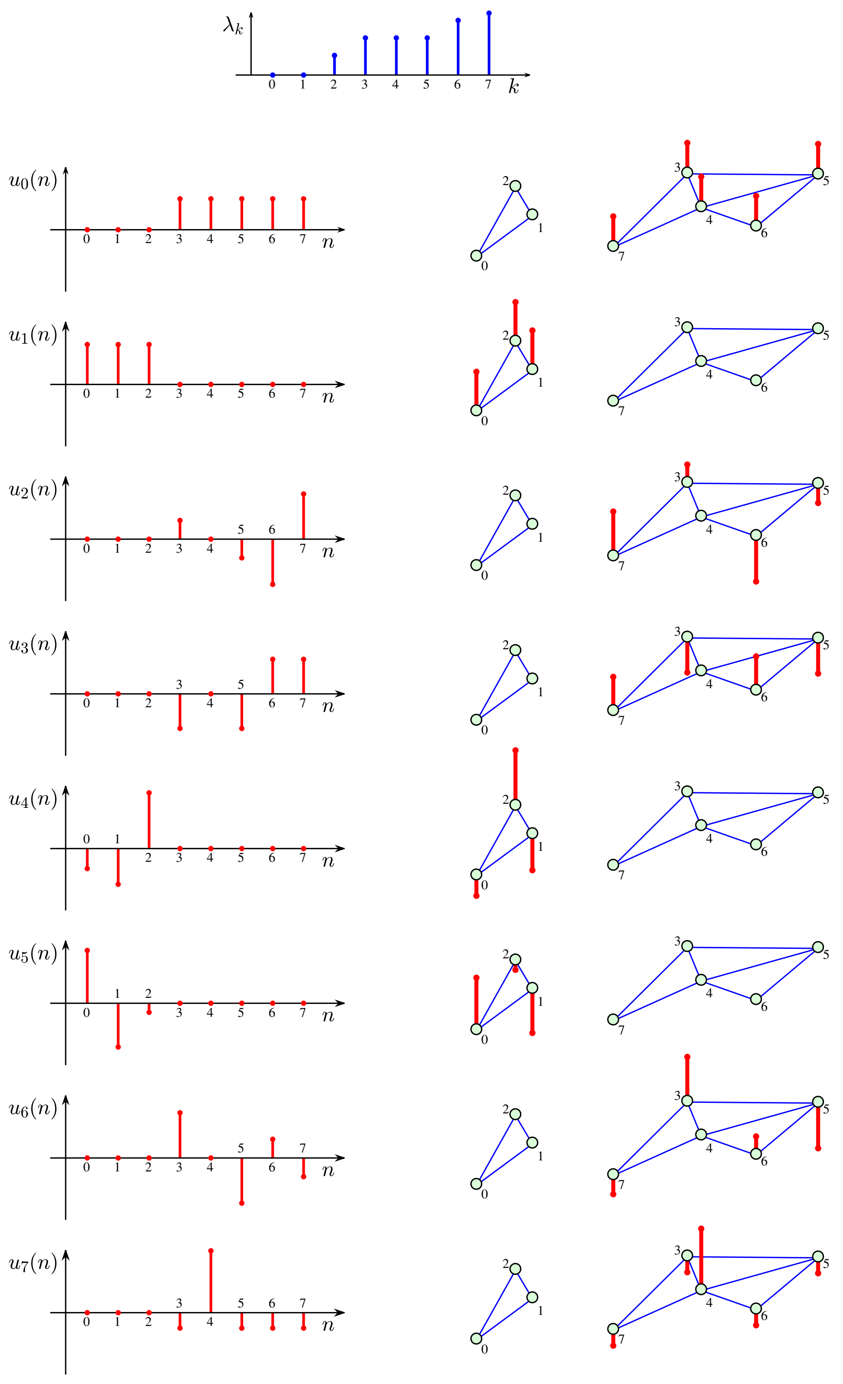


**Fig. 13** Eigenvalues $\lambda_k$ and corresponding eigenvectors $u_k(n)$ for Laplacian of the graph presented in Fig. 7.

with eigenvalues $(0, 1.134, 3.054, 3.317, 4, 5.679, 6.342, 6.473)$. In this case all eigenvalues are of multiplicity one so the minimal polynomial is equal to characteristic polynomial $P_{min}(\lambda) = P(\lambda)$.
The characteristic polynomial of the Laplacian for the graph from Fig. 7 is

$$P(\lambda) = \lambda^8 - 20\lambda^7 + 163\lambda^6 - 692\lambda^5 + 1611\lambda^4 - 1944\lambda^3 + 945\lambda^2.$$

with eigenvalues $(0, 0, 3-\sqrt{2}, 3, 3, 3, 3+\sqrt{2}, 5)$. Eigenvalue 0 is of multiplicity 2 and eigenvalue 3 is of multiplicity 3, so the minimal polynomial is

$$P_{min}(\lambda) = \lambda^5 - 14\lambda^4 + 70\lambda^3 - 146\lambda^2 + 105\lambda$$

– Graphs with the same spectrum are called isospectral or cospectral graphs. Construction of isospectral graphs that are not isomorph is an interesting topic in graph theory. A complete graph is uniquely defined by its spectrum.

– For a $K$-regular graph the eigenvectors of the Laplacian and adjacency matrix are the same, while for the eigenvalues the relation

$$\lambda_k^{(L)} = K - \lambda_k^{(A)}.$$

holds. It follows from $\mathbf{U}^T\mathbf{L}\mathbf{U} = \mathbf{U}^T(K\mathbf{I} - \mathbf{A})\mathbf{U}$.

– Eigenvalues of the normalized Laplacian $\mathbf{L}_N = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ satisfy the relation

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

The upper bound equality holds if and only if the graph is a bipartite graph.

– The eigenvalues and eigenvectors of the normalized Laplacian of a bipartite graph with vertices $\mathcal{E}$ and $\mathcal{H}$ satisfy the relation (graph spectrum folding)

$$\lambda_k = 2 - \lambda_{N-k}$$
$$\mathbf{u}_k = \begin{bmatrix} \mathbf{u}_{\mathcal{E}} \\ \mathbf{u}_{\mathcal{H}} \end{bmatrix} \text{ and } \mathbf{u}_{N-k} = \begin{bmatrix} \mathbf{u}_{\mathcal{E}} \\ -\mathbf{u}_{\mathcal{H}} \end{bmatrix}, \tag{12}$$

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

$$\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{A}_{\mathcal{E}\mathcal{H}} \\ \mathbf{A}_{\mathcal{E}\mathcal{H}}^T & \mathbf{0} \end{bmatrix} \quad \text{and} \quad \mathbf{L}_N = \begin{bmatrix} \mathbf{I} & \mathbf{L}_{\mathcal{E}\mathcal{H}} \\ \mathbf{L}_{\mathcal{E}\mathcal{H}}^T & \mathbf{I} \end{bmatrix}.$$

The eigenvalue relation is

$$\mathbf{L}_N\mathbf{u}_k = \begin{bmatrix} \mathbf{u}_{\mathcal{E}} + \mathbf{L}_{\mathcal{E}\mathcal{H}}\mathbf{u}_{\mathcal{H}} \\ \mathbf{L}_{\mathcal{E}\mathcal{H}}^T\mathbf{u}_{\mathcal{E}} + \mathbf{u}_{\mathcal{H}} \end{bmatrix} = \lambda_k \begin{bmatrix} \mathbf{u}_{\mathcal{E}} \\ \mathbf{u}_{\mathcal{H}} \end{bmatrix}.$$

From this relation we get $\mathbf{L}_{\mathcal{E}\mathcal{H}}\mathbf{u}_{\mathcal{H}} = (\lambda_k - 1)\mathbf{u}_{\mathcal{E}}$ and $\mathbf{L}_{\mathcal{E}\mathcal{H}}^T\mathbf{u}_{\mathcal{E}} = (\lambda_k - 1)\mathbf{u}_{\mathcal{H}}$, resulting in

$$\mathbf{L}_N \begin{bmatrix} \mathbf{u}_{\mathcal{E}} \\ -\mathbf{u}_{\mathcal{H}} \end{bmatrix} = (2 - \lambda_k) \begin{bmatrix} \mathbf{u}_{\mathcal{E}} \\ -\mathbf{u}_{\mathcal{H}} \end{bmatrix}.$$

It completes the proof of the property.

*Fourier analysis as a special case of the Laplacian spectrum*

For the undirected circular graph from Fig. 4(f) the eigenvalues of the Laplacian are

$$\lambda_k = \begin{cases} 2 - 2\cos(\pi(k+1)/N) & \text{for odd } k \\ 2 - 2\cos(\pi k/N) & \text{for even } k. \end{cases}$$

Note that for $k = 0$ we have $\lambda_0 = 0$. Most of the eigenvalues are of algebraic multiplicity 2, i.e., $\lambda_1 = \lambda_2$, $\lambda_3 = \lambda_4$, and so on. If $N$ is odd then $\lambda_{N-2} = \lambda_{N-1}$. For an even $N$ we have $\lambda_{N-1} = 2$ of algebraic multiplicity 1.

The corresponding eigenvectors $\mathbf{u}_0$, $\mathbf{u}_1$, ..., $\mathbf{u}_{N-1}$, are

$$u_k(n) = \begin{cases} \sin(\pi(k+1)n/N) & \text{for odd } k,\ k < N-1 \\ \cos(\pi k n/N) & \text{for even } k \\ \cos(\pi n) & \text{for odd } k,\ k = N-1, \end{cases} \tag{13}$$

where $k = 0, 1, \dots, N-1$ and $n = 0, 1, \dots, N-1$.

Note that an arbitrary linear combination of eigenvectors $\mathbf{u}_{2k-1}$ and $\mathbf{u}_{2k}$, $1 \leq k < N/2$ is also an eigenvector since the corresponding eigenvalues are equal. Having this fact in mind we can write an alternative set of the eigenvectors as

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

where $j^2 = -1$. It can be easily checked that this set of eigenvectors is orthonormal and the eigenvectors correspond to the DFT basis functions.

## 2.5 Vertex Ordering, Coloring, and Segmentation

The ordering of vertices of a graph can be arbitrary. This is an important difference from classical signal processing where the ordering is assumed and inherent. Any change or data ordering would produce significant changes in the classical signal processing results, in general. In the previous section we have seen (Fig. 10 and Fig. 11) that a reordering of the vertices will imply corresponding indices reordering within each eigenvector.

However, the presentation of a graph signal, in any other form than the presentation which uses the graph as the domain, would benefit from an appropriate vertex ordering. This is of particular importance in vertex-frequency graph signal representations.

Here we will describe a possible method of vertex ordering. It is based on the Laplacian eigendecomposition. The aim of this vertex ordering is to get the smoothest possible representation of the eigenvectors, corresponding to low eigenvalues, if the vertices are represented sequentially. The smoothness of a graph signal can be defined using the Laplacian quadratic form. The Laplacian quadratic form of an eigenvector is equal to the corresponding eigenvalue

$$\mathbf{u}_k^T(\mathbf{L}\mathbf{u}_k) = \mathbf{u}_k^T(\lambda_k \mathbf{u}_k) = \lambda_k.$$

This will be discussed in details in Section 3.6. The eigenvector corresponding to $\lambda_0 = 0$ is constant (maximally smooth for any vertex ordering) and it is not appropriate for the vertex ordering. The next smoothest eigenvector is $\mathbf{u}_1$ with eigenvalue $\lambda_1$. The aim here is to order vertices in a such way that the presentation of this vector, as a function of the vertex index, is also maximally smooth. This can be achieved by sorting the values of $\mathbf{u}_1$ into a nondecreasing order (the second eigenvalue $\lambda_1$ is called the Fiedler value or algebraic connectivity, while the corresponding eigenvector $\mathbf{u}_1$ is known as the Fiedler vector). The order of vertices in the sorted $\mathbf{u}_1$ corresponds to its smoothest presentation.

As an example, consider the vector $\mathbf{u}_1$ in Fig. 12. We see that there are big changes of the subsequent values of $u_1(n)$ with the presented vertex ordering, for example, the change from $u_1(6)$ to $u_1(7)$ is significant. The representation of $u_1(n)$ would be smoother if we sort the values of $u_1(n)$ and appropriately reorder the vertices. This kind of reordering for the graph presented in Fig. 12, would produce vertex order

$$[6, 5, 4, 3, 7, 1, 2, 0]$$

instead of the presented order $[0, 1, 2, 3, 4, 5, 6, 7]$. With this vertex ordering, the presentation of $u_1(n)$ would be smoother.

In general, the spectral similarity of vertices can be defined using more than one eigenvector. Coefficients $u_k(n)$, $k = 0, 1, \dots, N-1$ are assigned to the vertex $n$. We can assign an $N$ dimensional spectral vector

$$\mathbf{q}_n = [u_0(n), u_1(n), \dots, u_{N-1}(n)]^T$$

to each vertex $n$. If $\mathbf{u}_0$ is omitted then $\mathbf{q}_n = [u_1(n), \dots, u_{N-1}(n)]^T$.

The spectral similarity between vertices $n$ and $m$ is defined using norm-two $\|\mathbf{q}_n - \mathbf{q}_m\|_2$. We can restrict spectral similarity to a few lower-order (smooth) spectral coefficients. If we restrict the spectral similarity to the two (or three) smoothest coefficients then the spectral vector is $\mathbf{q}_n = [u_1(n), u_2(n)]^T$ in the two-dimensional case (or $\mathbf{q}_n = [u_1(n), u_2(n), u_3(n)]^T$ for the three dimensional case).

From this point we can proceed in two ways:

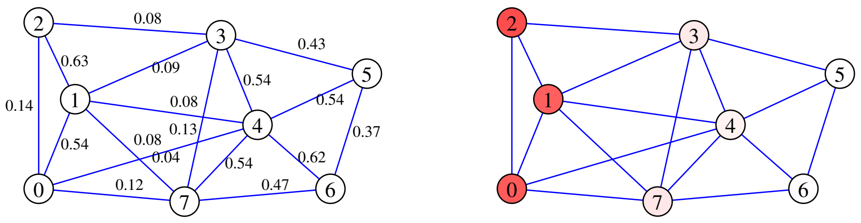


**Fig. 14** Vertex coloring in a graph with some weak connections using the Laplacian eigenvector $\mathbf{u}_1$ and corresponding intensities of red color

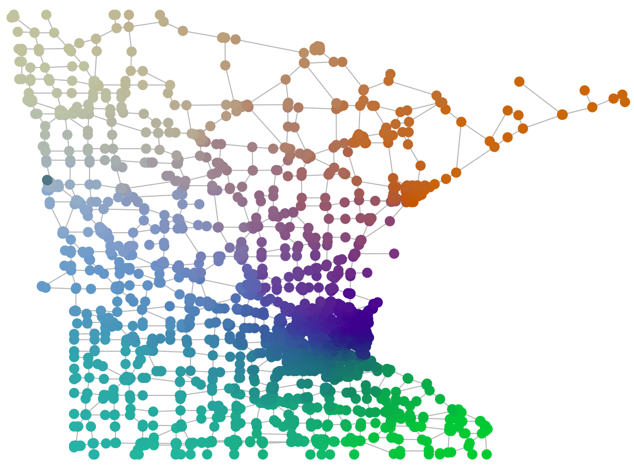


**Fig. 15** Vertex three-dimensional coloring in the Minnesota road-map graph using the Laplacian eigenvectors $\{\mathbf{u}_2,\mathbf{u}_3,\mathbf{u}_4\}$ as the coordinates in the RGB coloring system

- The first one is to keep the original vertex positions and to color them according to the spectral vector $\mathbf{q}_n$. Single color vertex coloring using values of $\mathbf{u}_1$ for the vertex color intensity is presented in Fig. 14. Based on this coloring we can clearly see three graph segments, $\{0, 1, 2\}$, $\{3, 4, 7\}$, and $\{5, 6\}$. Three color vertex coloring, using $\mathbf{u}_2$, $\mathbf{u}_3$, and $\mathbf{u}_4$ for the Minnesota graph is given in Fig. 15. Eigenvectors $\mathbf{u}_0$ and $\mathbf{u}_1$ are omitted in the Minnesota graph case, since corresponding eigenvalues are $\lambda_0 = \lambda_1 = 0$. The graph segmentation can be done by grouping vertices with similar colors, using appropriate thresholds, and assigning the vertices from each group to segments (with constant colors).
- Another approach is to use the spectral vector $\mathbf{q}_n$ as a position of a vertex in a new space (Laplacian eigenmaps or LE). In the case of two or three dimensional spectral vectors, this approach can be used to graphically present vertex spectral similarity. For the Minnesota graph, the Lalplacian eigenmap for the two-dimensional case is given in Fig. 16, and for the three-dimensional case in Fig. 17.

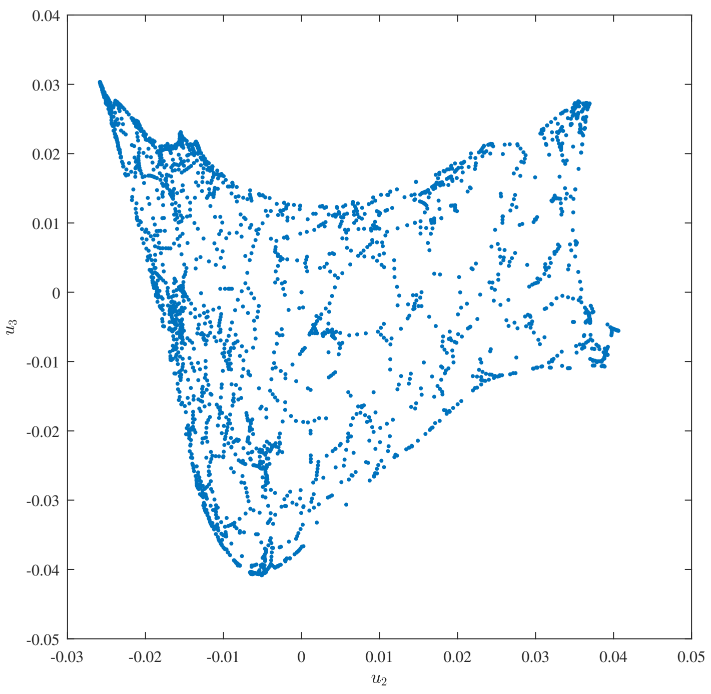


**Fig. 16** The Minnesota road-map graph with new two-dimensional vertex positions defined by the Laplacian eigenvectors $\{\mathbf{u}_2, \mathbf{u}_3\}$ as the vertex coordinates (the 2D Laplacian eigenmap).

## 3 Signals on Graphs

In classical signal processing, signal is sampled at successive, equally spaced, time instants. The ordering of signal samples is then obvious with $x(n)$ being preceded by $x(n-1)$ and succeeded by $x(n+1)$. The time distance between samples is considered as the basic parameter in various processing algorithms. This relation between sampling instants can be represented in a graph form. The vertices correspond to the instants when the signal is sampled and the edges define sampling (vertex) ordering. The fact that sampling instants are equally spaced can be represented with same weights for all edges (for example normalized to 1). Graphical illustration of this signal is given in Fig. 18.

In digital signal processing algorithms, periodicity of the analyzed signals is usually assumed, meaning that sample $x(N-1)$ is succeeded by $x(0)$. This case correspond to the circular graph, Fig. 19. This model is used in many common transforms, like DFT, DCT, wavelets, and corresponding processing algorithms based on these transforms.

Signal on a graph is defined by associating real (or complex) values $x(n)$ to each vertex, Fig. 20. Signal values can be arranged in a vector form

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

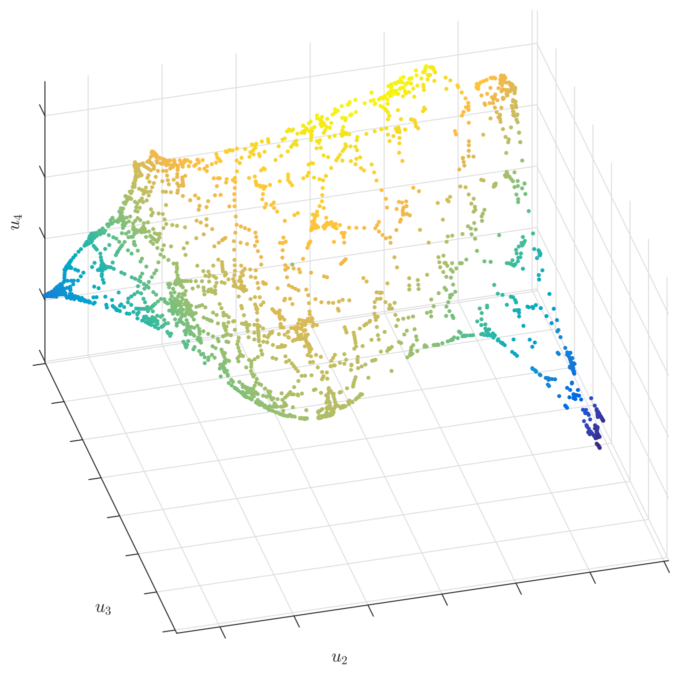


**Fig. 17** The Minnesota road-map graph with new thee-dimensional vertex positions defined by the Laplacian eigenvectors $\{\mathbf{u}_2,\mathbf{u}_3,\mathbf{u}_4\}$ as the vertex coordinates (the 3D Laplacian eigenmap)

;

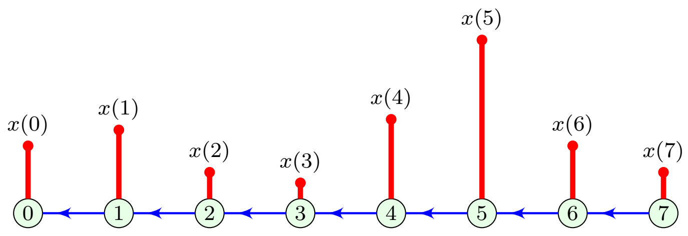


**Fig. 18** Graph representation of a classical time-domain signal.

The graph is considered as a generalized signal domain.

In general, any linear processing of a graph signal at a vertex $n$ can be defined as a linear combination of the signal value $x(n)$ at this vertex and the signal samples $x(m)$ at vertices around this vertex

$$y(n) = x(n)h(n,n) + \sum_{m\in\mathcal{V}_n} x(m)h(n,m),$$

where $\mathcal{V}_n$ is the set of vertices in the neighborhood of vertex $n$. This form is highly vertex-varying. Only in a specific case of regular graphs can it be vertex invariant. Then $\mathcal{V}_n$ is a $K$-neighborhood of the vertex $n$ with $h(n,m) = h(n-m)$.

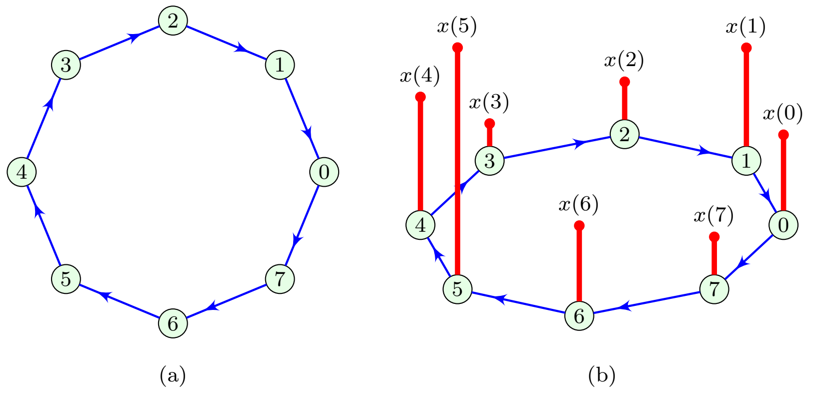


**Fig. 19** (a) A circular graph. (b) A periodic signal on a graph. Signal values are presented as vertical lines starting from the corresponding vertex.

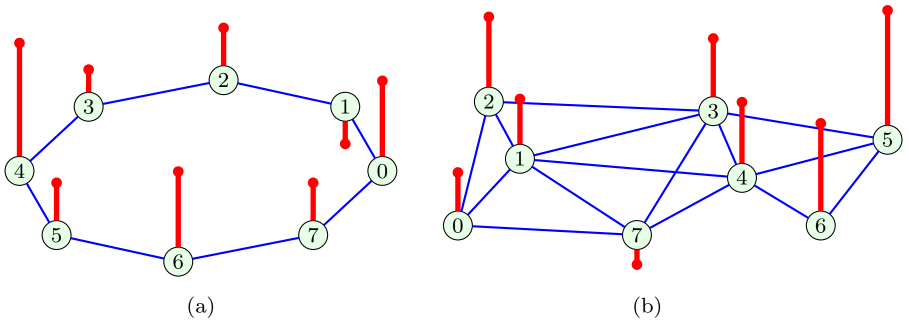


**Fig. 20** (a) A signal on an undirected circular graph. (b) Undirected arbitrary graph. Signal values are presented as vertical lines starting from the corresponding vertex.

For a general graph we can define a vertex-invariant filtering function, using shifts on a graph. Various forms of signal shifts on a graph will be introduced in the next sections. They are used to introduce efficient graph signal processing methods [24–38].

## 3.1 Adjacency Matrix and Graph Signal Shift

Consider a graph signal $\mathbf{x}$. Its sample at a vertex $n$ is $x(n)$. The signal shift on a graph can be defined as the movement of the signal sample from the vertex $n$ along all walks, with the length equal to one. The movement is done for all vertices. The signal shifted in this way is denoted by $\mathbf{x}_1$. Its values can be defined using the graph adjacency matrix as

$$\mathbf{x}_1 = \mathbf{A}\mathbf{x} \tag{14}$$

As an illustration of a signal and its shifted version, consider classical signal processing, where the adjacency matrix is defined by graph Fig. 19(a). The

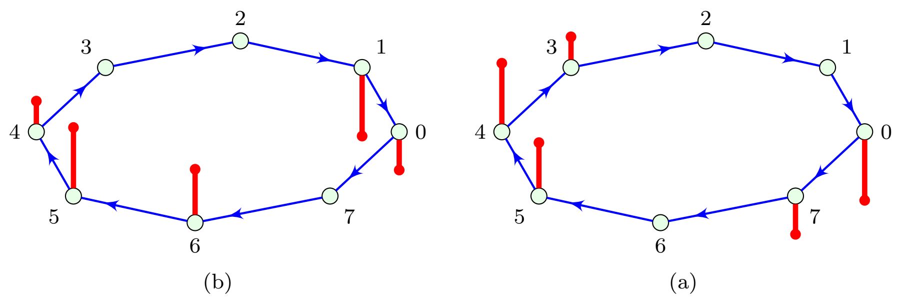


**Fig. 21** (a) A signal on the directed circular graph. (b) A shifted version of the graph signal from (a).

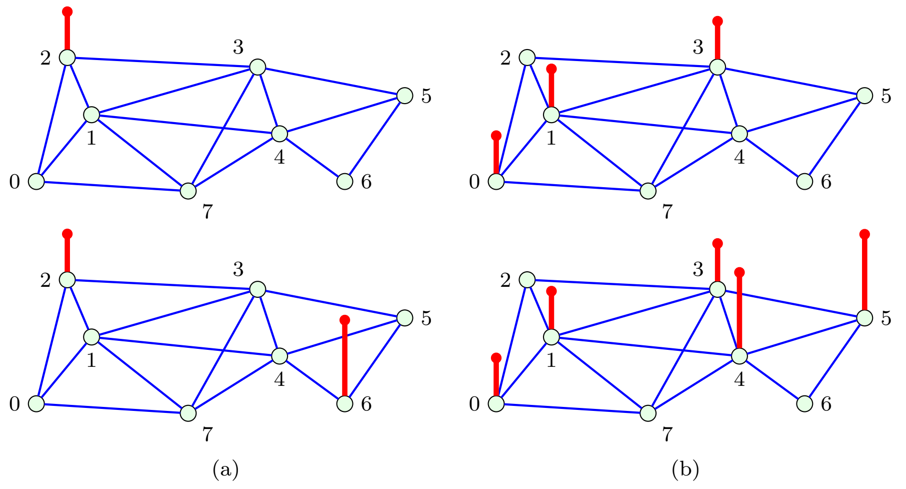


**Fig. 22** (a) Two simple signals on an undirected graph. (b) Shifted versions of the graph signals from (a).

original signal $\mathbf{x}$ is presented in Fig. 21(a). The shifted version of this signal $\mathbf{x}_1$ is shown in Fig. 21(b). Two simple signals on an undirected graph are presented on the left of Fig. 22(a). The corresponding shifted signals with $\mathbf{x}_1 = \mathbf{A}\mathbf{x}$ are presented on the right of Fig. 22(b).

A signal shifted by two is obtained by a shift for one of the shifted signals. The resulting, twice shifted, signal is

$$\mathbf{x}_2 = \mathbf{A}(\mathbf{A}\,\mathbf{x}) = \mathbf{A}^2\,\mathbf{x}.$$

In general, a graph signal shifted for $m$ is obtained as a shift by one of the graph signal already shifted for $m-1$

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

## 3.2 Systems Based on a Graph Shifted Signals

A system on a graph can be implemented as a linear combination of a graph signal and its graph shifted versions. The output signal from a system on a graph can be written as

$$\mathbf{y} = h_0\mathbf{A}^0\,\mathbf{x} + h_1\mathbf{A}^1\,\mathbf{x} + \cdots + h_{M-1}\mathbf{A}^{M-1}\,\mathbf{x} = \sum_{m=0}^{M-1} h_m\mathbf{A}^m\,\mathbf{x} \tag{15}$$

where $\mathbf{A}^0 = \mathbf{I}$, by definition.

The system coefficients are $h_0, h_1, \ldots, h_{M-1}$. For a circular (classical signal processing) graph this relation reduces to the well known FIR filter,

$$y(n) = h_0x(n) + h_1x(n-1) + \cdots + h_{M-1}x(n-M+1). \tag{16}$$

Having in mind that the matrix $\mathbf{A}^m$ describes walks of the length $m$ in a graph, the output graph signal $y(n)$ is calculated as a linear combination of the input graph signal values within $M-1$ neighborhood of the considered vertex $n$.

It is obvious that the system order $M-1$ should be lower than the number of vertices $N$ in the case when the minimal and characteristic polynomial are of the same degree. In general, the system order $M-1$ should be lower than the degree $N_m$ of the minimal polynomial of the adjacency matrix $\mathbf{A}$.

Any system of order $M-1 \geq N_m$ can be reduced to a system of order $N_m - 1$.

If the system order is higher or equal to the degree of 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 of these systems are called equivalent. This topic will be addressed in Section 3.4.4 dealing with the filter design in the spectral domain.

As an example consider a graph signal Fig. 23(left) and a linear system on this graph with coefficients $h_0 = 1$, $h_1 = 0.5$. The output graph signal is presented in Fig. 23(right).

In general, a system on a graph is defined in the vertex domain by

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

This system is linear since

$$H(\mathbf{A})(a_1\mathbf{x}_1 + a_2\mathbf{x}_2) = a_1\mathbf{y}_1 + a_2\mathbf{y}_2.$$

A system on a graph is shift invariant if

$$H(\mathbf{A})(\mathbf{A}\mathbf{x}) = \mathbf{A}(H(\mathbf{A})\mathbf{x}).$$

A system on a graph defined by

$$H(\mathbf{A}) = h_0\mathbf{A}^0 + h_1\mathbf{A}^1 + \cdots + h_{M-1}\mathbf{A}^{M-1} \tag{17}$$

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

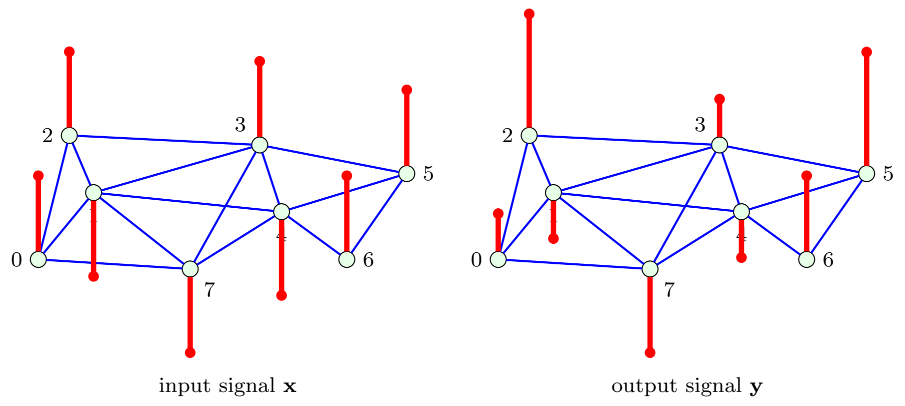


**Fig. 23** A vertex domain signal filtering example. An input graph signal (left) and the output signal obtained as $\mathbf{y} = 1.0\,\mathbf{x} + 0.5\,\mathbf{A}\mathbf{x}$ (right).

## 3.3 Graph Fourier transform based on the Adjacency matrix

In the classical signal analysis the signals are often analyzed and processed in the spectral (Fourier) domain. The spectral domain approach to signal processing has lead to many simple and efficient algorithms in classical signal processing.

The spectral analysis and processing approach can be extended to the graph signals as well. Spectral domain representations of the graph signals can be based on the adjacency matrix or Laplacian spectral decomposition. Both of these approaches will be described in this and the next section, respectively.

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

$$\mathbf{X} = \mathbf{U}^{-1}\mathbf{x} \tag{18}$$

where $\mathbf{U}$ is a matrix with eigenvectors of the adjacency matrix in its columns. Denote elements of vector $\mathbf{X}$ as $X(k)$, for $k = 0, 1, \dots, N-1$. If $\mathbf{U}^{-1} = \mathbf{U}^T$ the element $X(k)$ is a projection of the considered signal to the $k$-th eigenvector, as a graph signal decomposition basis function,

$$X(k) = \sum_{n=0}^{N-1} x(n)u_k(n). \tag{19}$$

Therefore the graph Fourier transform can be understood as a signal decomposition onto the set of eigenvectors as orthonormal basis functions.

The inverse graph Fourier transform is obtained as

$$\mathbf{x} = \mathbf{U}\,\mathbf{X} \tag{20}$$

or

$$x(n) = \sum_{k=0}^{N-1} X(k)u_k(n). \tag{21}$$

In the case of a circular graph from Fig. 19, this transform reduces to the standard discrete Fourier transform (DFT), equation (11). It is the reason why the transform (19) and its inverse (21) are referred to as the graph discrete Fourier transform (GDFT) and inverse graph discrete Fourier transform (IGDFT).

Consider a system on a graph (15). If we use the spectral representation of the adjacency matrix $\mathbf{A} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^{-1}$ we will get

$$\mathbf{y} = h_0\mathbf{U}\boldsymbol{\Lambda}^0\mathbf{U}^{-1}\,\mathbf{x} + h_1\mathbf{U}\boldsymbol{\Lambda}^1\mathbf{U}^{-1}\,\mathbf{x} + \cdots + h_{M-1}\mathbf{U}\boldsymbol{\Lambda}^{M-1}\mathbf{U}^{-1}\,\mathbf{x} \tag{22}$$

or

$$\mathbf{y} = \mathbf{U}\big(h_0\boldsymbol{\Lambda}^0 + h_1\boldsymbol{\Lambda}^1 + \cdots + h_{M-1}\boldsymbol{\Lambda}^{M-1}\big)\mathbf{U}^{-1}\,\mathbf{x} = \mathbf{U}\,H(\boldsymbol{\Lambda})\mathbf{U}^{-1}\,\mathbf{x}. \tag{23}$$

By left multiplying this relation with $\mathbf{U}^{-1}$ we get

$$\mathbf{U}^{-1}\mathbf{y} = H(\boldsymbol{\Lambda})\mathbf{U}^{-1}\,\mathbf{x} \tag{24}$$

If the GDFTs of the input and output graph signal

$$\mathbf{X} = \mathbf{U}^{-1}\mathbf{x}, \qquad \mathbf{Y} = \mathbf{U}^{-1}\mathbf{y}$$

are used, we will obtain the spectral domain system relation as

$$\mathbf{Y} = H(\boldsymbol{\Lambda})\,\mathbf{X} \tag{25}$$

or

$$Y(k) = (h_0 + h_1\lambda_k + \cdots + h_{M-1}\lambda_k^{M-1})X(k).$$

The transfer function of a system on a graph is defined by

$$H(\lambda_k) = \frac{Y(k)}{X(k)} = (h_0 + h_1\lambda_k + \cdots + h_{M-1}\lambda_k^{M-1}). \tag{26}$$

The classical signal analysis system (16) is obtained with the adjacency matrix whose eigenvalues are $\lambda_k = e^{-j2\pi k/N}$, defined by (11). Note that any classical system whose transfer function can be described using the DFT with periodicity $N$ can be written in this form with $M = N$.

Similar to the $z$ transform in the classical signal processing, we can introduce system transfer function in the $z$-domain for systems on graphs.

The $z$-domain transfer function of a system on a graph is defined as

$$H(z^{-1}) = \mathcal{Z}\{h_n\} = h_0 + h_1z^{-1} + \cdots + h_{M-1}z^{-(M-1)}. \tag{27}$$

Obviously

$$H(\lambda_k) = H(z^{-1})\big|_{z^{-1}=\lambda_k}$$

and we can use results defined for the classical $z$-domain transfer function.

Definition of the $z$-transform for arbitrary graph signals $x(n)$ and $y(n)$ that would satisfy the relation $Y(z^{-1}) = H(z^{-1})X(z^{-1})$ is not straightforward. It will be discussed later in Section 3.9.

## 3.4 Filtering in the Adjacency Matrix Spectral Domain

### *3.4.1 Normalization*

The energy of a graph shifted signal is $\|\mathbf{x}_1\|_2^2 = \|\mathbf{A}\mathbf{x}\|_2^2$. The graph shift does not satisfy isometry property. In general, the energy of shifted signal is not the same as the energy of the original signal, $\|\mathbf{A}\mathbf{x}\|_2^2 \neq \|\mathbf{x}\|_2^2$. In processing graph signals it is commonly desirable that a graph shift does not increase the signal energy.

Using the matrix norm-two it can be easily shown that the ratio of energies of the graph shifted and the original signal satisfies the relation

$$\max\{\frac{\|\mathbf{A}\mathbf{x}\|_2^2}{\|\mathbf{x}\|_2^2}\} = \max\{\frac{\mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x}}{\|\mathbf{x}\|_2^2}\} = \lambda_{\max}^2. \tag{28}$$

where $\lambda_{\max} = \max_k |\lambda_k|$.

If we do not want that the energy of a graph shifted signal $\|\mathbf{A}\mathbf{x}\|_2^2$ exceeds the energy of the original graph signal $\|\mathbf{x}\|_2^2$ then we should use the normalized adjacency matrix

$$\mathbf{A}_{norm} = \frac{1}{\lambda_{\max}}\mathbf{A}$$

in the graph shift operation and in any system on a graph. This normalization does not make the shift on graph operation isometric. The energy of the shifted signal is less than or equal to the energy of the original graph signal. The equality is achieved only for a very specific signal proportional to the eigenvector that corresponds to $\lambda_{\max}$.

The basic shift on a graph is then defined by using normalized adjacency matrix as

$$\mathbf{x}_1 = \mathbf{A}_{norm}\mathbf{x}. \tag{29}$$

A system on a graph with the normalized adjacency matrix is of the form

$$\mathbf{y} = \sum_{m=0}^{M-1} h_m \mathbf{A}_{norm}^m \mathbf{x}. \tag{30}$$

### *3.4.2 Spectral Domain Filtering*

The filtering relation, as a special kind of a system on a graph, can be written as

$$\mathbf{y} = \sum_{m=0}^{M-1} h_m \mathbf{A}_{norm}^m \mathbf{x} = H(\mathbf{A}_{norm})\mathbf{x}.$$

Its spectral domain form follows from the decomposition of $H(\mathbf{A}_{norm})$ as

$$\mathbf{y} = H(\mathbf{A}_{norm})\mathbf{x} = \mathbf{U}H(\mathbf{\Lambda})\mathbf{U}^{-1}\mathbf{x}$$

with

$$\mathbf{U}^{-1}\mathbf{y} = H(\mathbf{\Lambda})\mathbf{U}^{-1}\mathbf{x},$$

as

$$\mathbf{Y} = H(\mathbf{\Lambda})\mathbf{X},$$

where $\mathbf{X} = \mathbf{U}^{-1}\mathbf{x}$ and $\mathbf{Y} = \mathbf{U}^{-1}\mathbf{y}$ are the graph Fourier transforms of the input and output signal, respectively. The transfer function of a filter on a graph is again

$$H(\lambda_k) = \frac{Y(k)}{X(k)} = h_0 + h_1\lambda_k^1 + \cdots + h_{M-1}\lambda_k^{M-1},$$

where $\lambda_k$ are the eigenvalues of the normalized adjacency matrix $\mathbf{A}_{norm}$.

### *3.4.3 Spectral Ordering of the Adjacency Matrix Eigenvectors*

For proper low-pass and high-pass filtering we have to establish the spectral order. This means that we have to establish a criterion to classify the eigenvectors, corresponding to the basis functions, as slow varying or fast varying. In classical Fourier analysis, the basis functions are ordered according to the frequency. Low-pass (slow varying) basis functions are the functions with small frequencies. The frequencies of the graph eigenvectors, as functions for signal decomposition, are not defined. We have to find another criterion to classify the eigenvectors. Again, an inspiration will be found in the classical Fourier analysis. In that case, instead of the frequency, energy of the signal change can be used as an indicator of the speed of a signal change in time.

The energy of a signal (a basis function) $u(n)$ change in classical analysis can be defined as the energy of the first difference

$$E_{\Delta u} = \sum_{n=0}^{N-1} |u(n) - u(n-1)|^2.$$

Lower values of $E_{\Delta u}$ means that $u(n)$ is slow-varying. Value $E_{\Delta u} = 0$ indicates, in classical signal analysis, that the signal is constant. Large values of $E_{\Delta u}$ are associated with fast signal changes in time. This form is also called the norm-two total variation of a signal. If the energy of a basis function $u(n)$ change is large it means that this eigenvector can be considered as the one belonging to the higher spectral content of the signal.

In graph signals the first graph difference can be defined as a difference of the graph signal and its graph shift. For an eigenvector $\mathbf{u}$, its form is

$$\mathbf{\Delta u} = \mathbf{u} - \mathbf{u}_1 = \mathbf{u} - \mathbf{A}_{norm}\mathbf{u}.$$

The energy of signal change is the energy of the first graph difference of signal $\mathbf{u}$

$$E_{\Delta u} = \|\mathbf{u} - \mathbf{A}_{norm}\mathbf{u}\|_2^2 \tag{31}$$

$$= \left\|\mathbf{u} - \frac{1}{\lambda_{\max}}\mathbf{A}\mathbf{u}\right\|_2^2 = \left\|\mathbf{u} - \frac{1}{\lambda_{\max}}\lambda\mathbf{u}\right\|_2^2 = |1 - \frac{\lambda}{\lambda_{\max}}|^2 \tag{32}$$

For eigenvectors $\mathbf{A}\mathbf{u} = \lambda\mathbf{u}$ and $\|\mathbf{u}\|_2^2 = 1$ hold.

The energy of signal change is minimal for $\lambda = \lambda_{\max}$ and increases as $\lambda$ decreases, Fig. 10.

After we have established a criterion for the eigenvector ordering, based on the corresponding eigenvalues, we will define an ideal low-pass filter. This filter should pass unchanged all signal components (eigenvectors) whose changes are slower than the one defined by the cut-off eigenvalue $\lambda_c$. It should stop all signal components (eigenvectors) whose variations are faster than the one defined by the cut-off eigenvalue. The ideal low-pass filter is defined as

$$f(\lambda) = \begin{cases} 1 & \text{for } \lambda > \lambda_c \\ 0 & \text{for other } \lambda. \end{cases}$$

As an example, consider a signal on a graph presented in Fig. 2(a). The graph signal is obtained as a linear combination of two adjacency matrix eigenvectors $\mathbf{x} = 3.2\mathbf{u}_7 + 2\mathbf{u}_6$ (adjacency matrix eigenvectors of the considered graph are presented in Fig. 10). The signal is presented in Fig. 24(a). The signal is corrupted by a white Gaussian noise with signal-to-noise (SNR) ratio $SNR_{in} = 2.7$dB. The noisy graph signal is presented in Fig. 24(b). The noisy signal is filtered by using an ideal spectral domain graph filter with a cut-off eigenvalue $\lambda_c = 1$. The output signal, presented in Fig. 24(c), is obtained. The output SNR is $SNR_{out} = 18.8$dB.

The energy of signal change criterion is consistent with the classical DFT based filtering when $\lambda_k = \exp(-j2\pi k/N)$ and $\lambda_{\max} = 1$. In that case the decision on the low-pass and high-pass basis functions is made based on $E_{\Delta u}$ value and a given threshold.

### *3.4.4 Spectral Domain Filter Design*

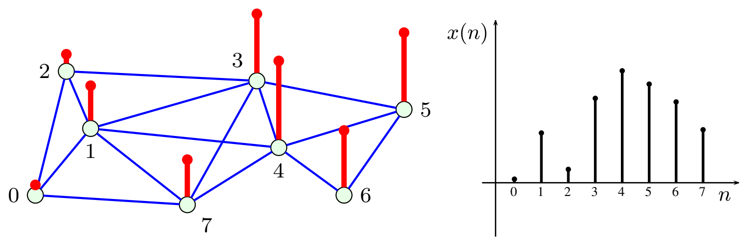
Let the desired graph transfer function be $G(\mathbf{\Lambda})$. A system with this transfer function can be implemented either in the spectral domain or in the vertex domain.

In the spectral domain the implementation is straightforward. It can be performed in the following three steps:

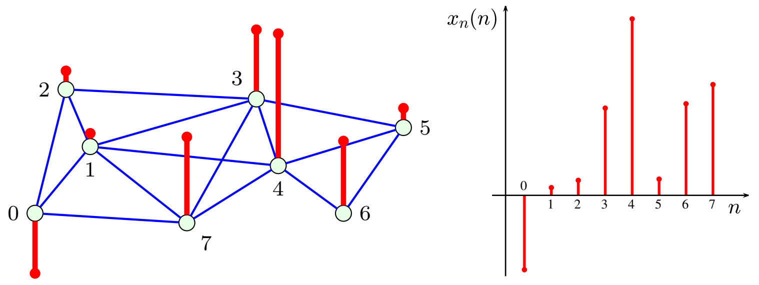
1. Calculate the GDFT of the input graph signal $\mathbf{X} = \mathbf{U}^{-1}\mathbf{x}$,
2. Multiply the GDFT of the input graph signal by $G(\mathbf{\Lambda})$ to get $\mathbf{Y} = G(\mathbf{\Lambda})\mathbf{X}$, and
3. Calculate the output graph signal as the inverse GDFT, $\mathbf{y} = \mathbf{U}\mathbf{Y}$.

This procedure may computationally be very demanding for large graphs. In the case of a large graph it would be easier to implement the desired filter (or its close approximation) in the vertex domain.

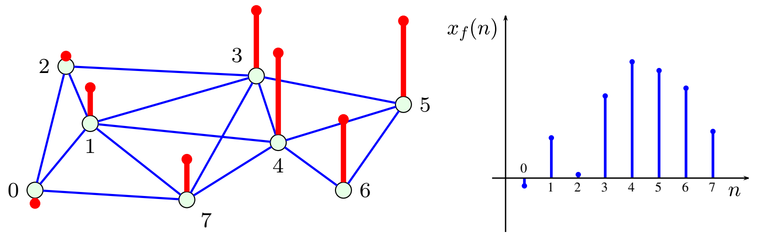
For the implementation in the vertex domain, we have to find the coefficients $h_0, h_1, \ldots, h_{M-1}$ in (15) such that its spectral representation $H(\mathbf{\Lambda})$ is equal (or approximately equal) to $G(\mathbf{\Lambda})$. This is done in the following way. The transfer function of the vertex domain system is given by (26) as



(a) original signal



(b) noisy signal



(c) filtered signal

**Fig. 24** Signal filtering example. Original signal (a), noisy signal (b) and filtered signal (c). An ideal low-pass filtering with two highest eigenvalues in the pass-band is applied.

$H(\lambda_k) = h_0 + h_1\lambda_k^1 + \dots h_{M-1}\lambda_k^{M-1}$. It should be equal to the desired transfer function $G(\lambda_k)$, for $k = 0, 1, \dots, N-1$. This condition leads to a system of linear equations

$$\begin{gathered} h_0 + h_1\lambda_0^1 + \dots h_{M-1}\lambda_0^{M-1} = G(\lambda_0) \\ h_0 + h_1\lambda_1^1 + \dots h_{M-1}\lambda_1^{M-1} = G(\lambda_1) \\ \vdots \\ h_0 + h_1\lambda_{N-1}^1 + \dots h_{M-1}\lambda_{N-1}^{M-1} = G(\lambda_{N-1}). \end{gathered} \tag{33}$$

The matrix form of this system is

$$\mathbf{V}_\lambda \, \mathbf{h} = \mathbf{g}, \tag{34}$$

where $\mathbf{V}_\lambda$ is the Vandermonde matrix form of eigenvalues $\lambda_k$,

$$\mathbf{h} = [h_0, h_1, \dots, h_{M-1}]^T$$

is the vector of the system coefficients that we want to calculate, and

$$\mathbf{g} = [G(\lambda_0), G(\lambda_1), \dots, G(\lambda_{N-1})]^T = \mathrm{diag}(G(\mathbf{\Lambda})).$$

Now we will comment on the solution of system (33), (34).

*Comments on the System of Equations Solution:*

1. Consider the case when all eigenvalues are distinct (minimal polynomial is equal to characteristic polynomial, $P_{min}(\lambda) = P(\lambda)$).
   (a) If the filter order is such that $M = N$, then the solution of (33) is unique, since the Vandermonde determinant is always nonzero.
   (b) If the filter order is such that $M < N$, then system (33) is overdetermined. The solution of (33) is obatined in the mean squared sense only (as it will be 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 $N$) system (33) reduces to a system of $N_m$ linear equations (by removing multiple equations for the repeated eigenvalues $\lambda$).
   (a) If the filter order is such that $N_m < M \leq N$ the system is underdetermined. In that case $M - N_m$ filter coefficients are free variables. The system has an infinite number of soulutions. All obtained filters are equivalent.
   (b) If the filter order is such that $M = N_m$ the solution of system (33) is unique.
   (c) If the filter order is such that $M < N_m$ the system (33) is overdetermined and the solution is obtained in the mean squared sense.
3. Any filter of an order $M > N_m$ has a unique equivalent filter whose order is $N_m$. It can be obtained by setting free variables to zero, $h_i = 0$ for $i = N_m, N_m + 1, \dots, N - 1$.

*Solution of the System*

For $M = N = N_m$ the solution of system (33) or (34) is

$$\mathbf{h} = \mathbf{V}_\lambda^{-1}\mathbf{g}.$$

For the overdetermined case (when $M < N_m$) the mean-square approximation of $\mathbf{h} = [h_0, h_1, \dots, h_M]^T$ is obtained by minimizing the squared error

$$e = \|\mathbf{V}_\lambda \mathbf{h} - \mathbf{g}\|_2^2 .$$

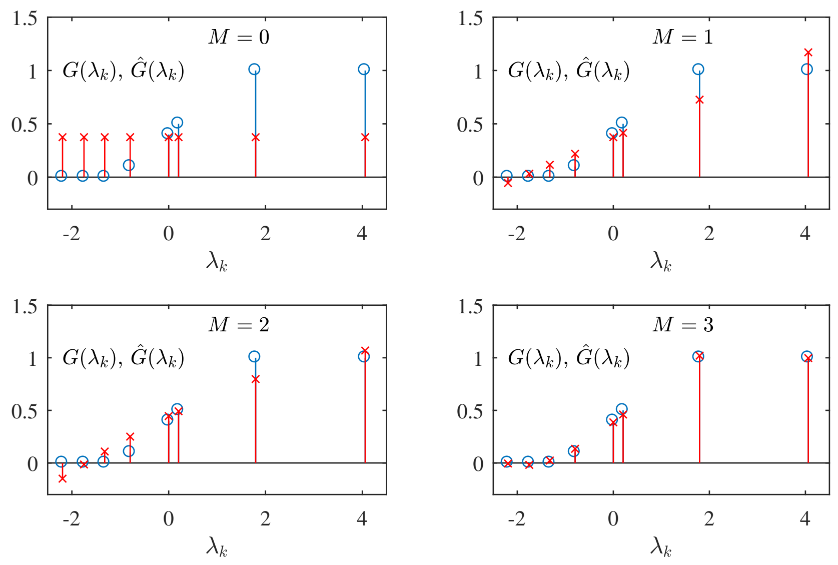


**Fig. 25** Designing a filter with the specified transfer function in the spectral domain. The desired spectral response $G(\lambda_k)$ is presented with blue circles. The designed graph system response $\hat{G}(\lambda_k)$, obtained with $M+1$ filter coefficients $h_0, h_1, \dots, h_M$ in the vertex domain, is presented with red asterisks.

From $\partial e/\partial \mathbf{h}^T = \mathbf{0}$ we get

$$\hat{\mathbf{h}} = (\mathbf{V}_\lambda^T \mathbf{V}_\lambda)^{-1}\mathbf{V}_\lambda^T \mathbf{g} = \text{pinv}(\mathbf{V}_\lambda)\mathbf{g}.$$

In the cases when $M < N_m$ the obtained solution $\hat{\mathbf{h}}$ is the mean square solution for $\mathbf{V}_\lambda \mathbf{h} = \mathbf{g}$. Since this solution may not satisfy $\mathbf{V}_\lambda \mathbf{h} = \mathbf{g}$ then the designed coefficients $\hat{\mathbf{g}}$ (its spectrum $\hat{G}(\boldsymbol{\Lambda})$)

$$\mathbf{V}_\lambda \hat{\mathbf{h}} = \hat{\mathbf{g}}$$

in general differs from the desired system coefficients $\mathbf{g}$ (its spectrum $G(\boldsymbol{\Lambda})$).

As an example consider the graph from Fig. 2(a) and the synthesis of a desired filter whose frequency response would be

$$\mathbf{g} = [1, 1, 0.5, 0.4, 0.1, 0, 0, 0]^T.$$

Consider the following cases: $M = 0, 1, 2, 3$. The filter is designed for various $M$ using (33). The solution is obtained according the presented procedure. The results are shown in Fig. 25. The vertex domain realization of the filter with $M = 3$ is

$$\mathbf{y} = 0.4456\mathbf{A}^0\mathbf{x} + 0.2298\mathbf{A}^1\mathbf{x} - 0.0188\mathbf{A}^2\mathbf{x}. \tag{35}$$

For $M = N = 8$ the exact frequency response $\hat{\mathbf{g}} = \mathbf{g}$ is obtained.

*Inverse System*

An inverse filter $H(\mathbf{\Lambda})$ to $G(\mathbf{\Lambda})$ is obtained from

$$H(\mathbf{\Lambda})G(\mathbf{\Lambda})\mathbf{X} = \mathbf{X}.$$

It means that $H(\lambda_k) = 1/G(\lambda_k)$ for each $k$ if all $G(\lambda_k) \neq 0$ and $P(\lambda) = P_{min}(\lambda)$.

## 3.5 Graph Fourier Transform Based on the Laplacian

Like in the case of an adjacency matrix, the spectral decomposition of a graph signal can be done using the eigenvalue decomposition of the Laplacian $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1}$ or $\mathbf{L}\mathbf{U} = \mathbf{U}\mathbf{\Lambda}$. Although the analysis can be done in a unified way for both the adjacency matrix and the Laplacian based spectral decomposition, due to their different behavior and importance they will be considered separately.

The graph Fourier transform of a signal $\mathbf{x}$, using the Laplacian eigenvalue decomposition, is defined as

$$\mathbf{X} = \mathbf{U}^{-1}\mathbf{x}, \tag{36}$$

where $\mathbf{U}$ is a matrix with the Laplacian eigenvectors. The inverse graph Fourier transform is

$$\mathbf{x} = \mathbf{U}\,\mathbf{X}. \tag{37}$$

In the case of circular unweighted graph this spectral analysis also reduces to the standard Fourier transform, but with real-valued basis functions (13).

## 3.6 Ordering and Filtering in the Laplacian Spectral Domain

The graph shift and adjacency matrix are related to the first finite difference in the vertex domain. The eigenvectors (basis functions) variations are related to the energy of the graph signal change, Section 3.4.3. A similar approach can be used for the Laplacian based decomposition.

In the case of classical time domain signals, the Laplacian on a circle graph represents the second order finite difference $y(n) = -u(n-1)+2u(n)-u(n+1)$. This difference can be written in a matrix form as $\mathbf{y} = \mathbf{L}\mathbf{u}$. It is obvious that the eigenvectors $u(n)$ with small changes should have small cumulative energy of the second order difference $E_u = \sum_n((u(n)-u(n-1))^2+(u(n)-u(n+1))^2)/2$. This value corresponds to the quadratic form of eigenvector $\mathbf{u}$ defined by $E_u = \mathbf{u}^T\mathbf{L}\mathbf{u}$. This reasoning can be used in the graph signals as well. As a default case for the Laplacian analysis we will use weighted undirected graphs.

By definition

$$\mathbf{L}\mathbf{u} = \lambda\mathbf{u}$$

or

$$\mathbf{u}^T\mathbf{L}\mathbf{u} = \lambda\mathbf{u}^T\mathbf{u} = \lambda = E_u,$$

since $\mathbf{u}^T\mathbf{u} = 1$ by definition. It means that the quadratic form of an eigenvector is equal to the corresponding eigenvalue. Next we will show that it can be used as a measure of the signal smoothness. By definition

$$\mathbf{u}^T\mathbf{Lu} = \sum_{n=0}^{N-1} u(n) \sum_{m=0}^{N-1} W_{nm}(u(n) - u(m)) =$$
$$\sum_{n=0}^{N-1} \sum_{m=0}^{N-1} W_{nm}(u^2(n) - u(n)u(m)).$$

In full summations over $n$ and $m$ we can replace the summation of $u^2(n)$ by a half of the summations of both $u^2(n)$ and $u^2(m)$ over $n$ and $m$, since $W_{nm} = W_{mn}$. The same can be done for $u(n)u(m)$. Then we can write

$$\mathbf{u}^T\mathbf{Lu} = \frac{1}{2}\sum_{n=0}^{N-1} \sum_{m=0}^{N-1} W_{nm}(u^2(n) - u(n)u(m) + u^2(m) - u(m)u(n)) =$$
$$\frac{1}{2}\sum_{n=0}^{N-1} \sum_{m=0}^{N-1} W_{nm}(u(n) - u(m))^2 \geq 0. \tag{38}$$

Obviously small $\mathbf{u}^T\mathbf{Lu} = \lambda$ means small variations $W_{nm}(u(n) - u(m))^2$ in the eigenvector for each vertex $n$. The eigenvectors corresponding to small $\lambda$ belong to the low-pass part of a graph signal.

From the previous analysis we may also conclude that the eigenvalues of a Laplacian are nonnegative (positive semi-definite matrix). At least one Laplacian eigenvalue is equal to 0. According to the Laplacian definition we have that the sum of each row (column) is equal to 0. It means that for $\mathbf{x} = \mathbf{1}$ we have $\mathbf{Lx} = \mathbf{0} = 0 \cdot \mathbf{x}$. We can conclude that there is eigenvalue $\lambda_0 = 0$ with corresponding eigenvector $\mathbf{u}_0 = \mathbf{1}/\sqrt{N}$. A vector with all values equal to 1 is denoted by $\mathbf{1}$.

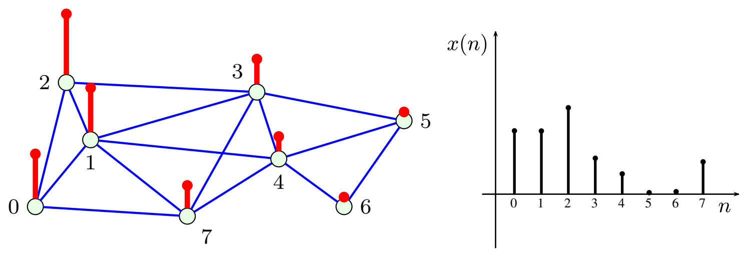
In general, the smoothness of a graph signal $\mathbf{x}$ is defined by the quadratic form

$$E_x = \mathbf{x}^\mathbf{T}\mathbf{Lx}.$$

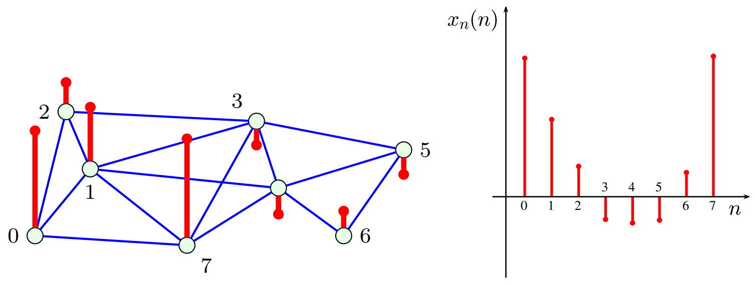
An ideal low-pass filter in the Laplacian spectrum domain, with a cut-off eigenvalue $\lambda_c$, will be defined as

$$f(\lambda) = \begin{cases} 1 & \text{for } \lambda < \lambda_c \\ 0 & \text{for other } \lambda. \end{cases}$$

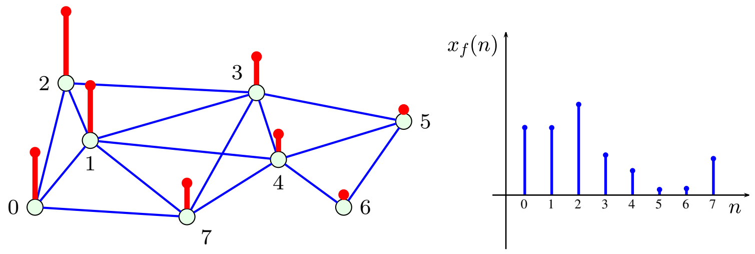
As an example consider a signal on the graph presented in Fig. 3. The graph signal is obtained as a linear combination of two Laplacian eigenvectors $\mathbf{x} = 2\mathbf{u}_0 + 1.5\mathbf{u}_1$ (Laplacian eigenvectors of the considered graph are presented in Fig. 12). This signal is presented in Fig. 26(a). The signal is corrupted by a white Gaussian noise. The noisy graph signal is described by a signal-to-noise (SNR) ratio $SNR_{in} = -1.76$dB. Noisy graph signal is presented in Fig. 26(b). Using an ideal spectral domain graph filter, with a cut-off eigenvalue $\lambda_c = 2$,



(a) original signal



(b) noisy signal



(c) filtered signal

**Fig. 26** Signal filtering example. Original signal (a), noisy signal (b) and filtered signal (c). Low pass filtering with two largest eigenvalues is applied.

the noisy graph signal is filtered. The output signal, presented in Fig. 26(c), is obtained. The output SNR, for this signal, is $SNR_{out} = 21.29\text{dB}$.

A direct relation between the adjacency and Laplacian spectral decomposition can be established for $K$-regular unweighted graphs. For these graphs holds

$$\mathbf{L} = K\mathbf{I} - \mathbf{A}$$

resulting in

$$\lambda_A = K - \lambda_L,$$

where the adjacency matrix and the Laplacian eigenvalues are denoted by $\lambda_A$ and $\lambda_L$, respectively. The eigenvectors are the same. Ordering with respect to $\lambda$ from the low-pass to the high-pass part is just opposite for these two graph spectral decompositions.

## 3.7 Systems on a Graph Defined Using the Laplacian

A system on a graph can be defined using the Laplacian as well

$$\mathbf{y} = h_0\mathbf{L}^0\,\mathbf{x} + h_1\mathbf{L}^1\,\mathbf{x} + \cdots + h_{M-1}\mathbf{L}^{M-1}\,\mathbf{x} = \sum_{m=0}^{M-1} h_m\mathbf{L}^m\,\mathbf{x}. \tag{39}$$

For an unweighted graph this system form can be related to the adjacency matrix form using $\mathbf{L} = \mathbf{D} - \mathbf{A}$.

The spectral domain description of a graph system is obtained using the Laplacian eigenvalue decomposition,

$$\mathbf{y} = \sum_{m=0}^{M-1} h_m\mathbf{L}^m\,\mathbf{x} = H(\mathbf{L})\mathbf{x} = \mathbf{U}H(\mathbf{\Lambda})\mathbf{U}^T\mathbf{x} = \mathbf{U}H(\mathbf{\Lambda})\mathbf{X} = \mathbf{U}\mathbf{Y}, \tag{40}$$

where

$$\mathbf{Y} = H(\mathbf{\Lambda})\mathbf{X}$$

or

$$Y(k) = H(\lambda_k)X(k),\ \ k = 0, 1, \ldots, N-1.$$

In the vertex domain the $n$-th element of $\mathbf{y} = \mathbf{U}H(\mathbf{\Lambda})\mathbf{U}^T\mathbf{x}$ is

$$y(n) = \sum_{k=0}^{N-1}\sum_{i=0}^{N-1} x(i)u_k(i)H(\lambda_k)u_k(n) = \sum_{i=0}^{N-1} x(i)h_n(i), \tag{41}$$

where

$$H(\lambda_k) = h_0 + h_1\lambda_k + \cdots + h_{M-1}\lambda_k^{M-1} \tag{42}$$

and

$$h_n(i) = \sum_{k=0}^{N-1} H(\lambda_k)u_k(n)u_k(i) = \mathcal{T}_n\{h(i)\}.$$

The value of $y(n)$ can be interpreted as a generalized convolution, using a generalized shift of impulse response in the vertex domain. It can be described by using responses to the unite delta pulses. Let us consider the delta function located at a graph vertex $m$ and its spectrum. The delta function at the vertex $m$ is defined as

$$\delta_m(n) = \begin{cases} 1 & \text{for } n = m \\ 0 & \text{for } n \neq m, \end{cases} \tag{43}$$

and the corresponding spectrum is given by

$$\mathit{\Delta}(\lambda_k) = \sum_{n=0}^{N-1} \delta_m(n)u_k(n) = u_k(m). \tag{44}$$

Any graph signal can be written as

$$x(n)=\sum_{i=0}^{N-1}x(i)\delta_n(i)$$

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 $\delta(n-i)$. Then

$$\mathbf{y}=\sum_{m=0}^{M-1}h_m\mathbf{L}^m\,\mathbf{x}=\mathbf{U}H(\boldsymbol{\Lambda})\mathbf{U}^T\mathbf{x}=\sum_{i=0}^{N-1}x(i)\mathbf{U}H(\boldsymbol{\Lambda})\mathbf{U}^T\boldsymbol{\delta}_i$$

with elements

$$y(n)=\sum_{i=0}^{N-1}x(i)\sum_{k=0}^{N-1}u_k(n)H(\lambda_k)u_k(i)=\sum_{i=0}^{N-1}x(i)h_n(i).$$

Calculation of this form of convolution for a vertex $n$, given by (41), is localized to the $(M-1)$ neighborhood of vertex $n$, according to (40). This is an important property for large graphs. A generalized convolution for two arbitrary graph signals will be explained next.

## 3.8 Convolution of Signals on a Graph

Consider two graph signals $x(n)$ and $h(n)$. A generalized convolution operator of these two signals on a graph is defined using their spectra [39]. The assumption is that the spectrum of a convolution

$$y(n)=x(n)*h(n)$$

on a graph is equal to the product of the graph signal spectra

$$Y(k)=X(k)H(k). \tag{45}$$

The result of the generalized graph convolution operation $x(n)*h(n)$ is equal to the inverse GDFT of $Y(k)$,

$$y(n)=x(n)*h(n)=\sum_{k=0}^{N-1}Y(k)u_k(n)=\sum_{k=0}^{N-1}X(k)H(k)u_k(n).$$

In this case

$$H(k)=\sum_{n=0}^{N-1}h(n)u_k(n).$$

A shift on the graph can be defined within the framework of the generalized convolution. Consider the graph signal $h(n)$ and the delta function located at the vertex $m$. Here, we will use $h_m(n)$ to denote the shifted version of the graph signal $h(n)$. The signal corresponding to a shift to a vertex $m$ is equal to

$$h_m(n) = h(n) * \delta_m(n) = \sum_{k=0}^{N-1} H(k)u_k(m)u_k(n). \tag{46}$$

The same relation follows from the inverse GDFT of $X(k)H(\lambda_k)$,

$$y(n) = \sum_{k=0}^{N-1} X(k)H(k)u_k(n) = \sum_{k=0}^{N-1}\sum_{m=0}^{N-1} x(m)u_k(m)H(k)u_k(n) = \\ \sum_{m=0}^{N-1} x(m)h_m(n) = x(n) * h(n), \tag{47}$$

where

$$h_m(n) = \sum_{k=0}^{N-1} H(k)u_k(m)u_k(n) = T_m\{h(n)\}$$

plays the role of a shifted signal. Since the definition of $H(k)$ as a GDFT of a signal $h(n)$ differs from (42) it produces different shift operation. These two shift operations are denoted by $T_m\{h(n)\}$ and $\mathcal{T}_m\{h(n)\}$, respectively.

Consider, for example, the signal with Laplacian GDFT

$$H(k) = \exp(-k/4).$$

Shifted signals $h(n)$ obtained using $h_m(n) = T_m\{h(n)\}$ are presented in Fig. 27.

## 3.9 Graph $z$-transform of a Signal

The relation between $T_m\{h(n)\}$ and $\mathcal{T}_m\{h(n)\}$ can be established based on the definitions of $H(\lambda_k)$ and $H(k)$. For $H(\lambda_k)$ defined by (42) the corresponding IGDFT coefficients $h(n)$

$$h(n) = \sum_{k=0}^{N-1} H(\lambda_k)u_k(n)$$

and the coefficients $h_n$ in

$$H(\lambda_k) = h_0 + h_1\lambda_k + \cdots + h_{M-1}\lambda_k^{M-1}$$

are not the same, $h(n) \neq h_n$.

Vector form of the last two relations is

$$[h(0)\ h(1) \ldots h(N-1)]^T = \mathbf{U}H(\mathbf{\Lambda})$$
$$H(\mathbf{\Lambda}) = \mathbf{V}_\lambda[h_0\ h_1 \ldots h_{N-1}]^T.$$

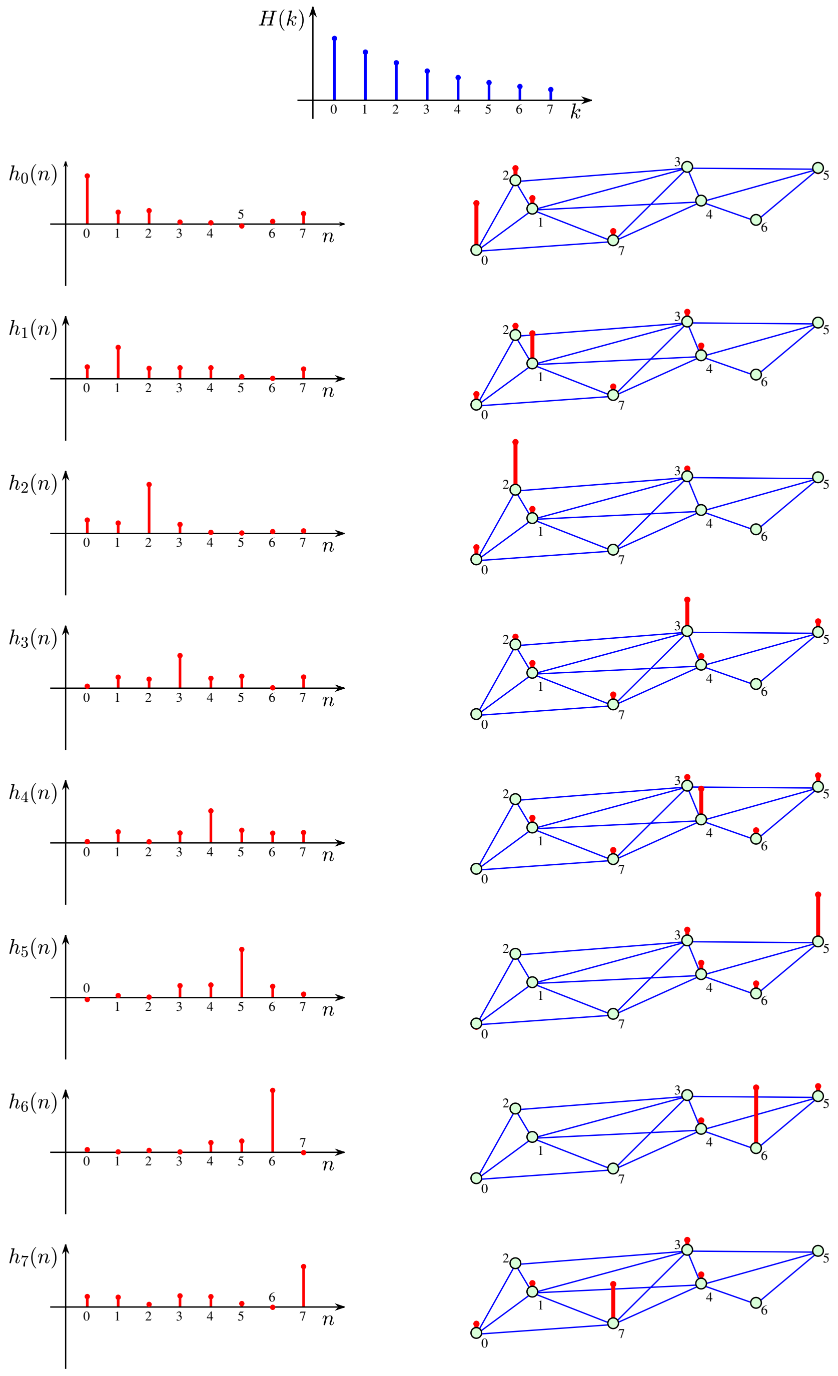


**Fig. 27** Graph signal shifts based on the Laplacian eigendecomposition.

The signal $h(n)$ and the coefficients $h_n$ can easily be related via

$$[h_0\ h_1 \dots h_{N-1}]^T = \mathbf{V}_\lambda^{-1}\mathbf{U}^T[h(0)\ h(1)\dots h(N-1)]^T.$$

These coefficients would be the same in the classical DFT (with directed adjacency matrix) when $\lambda_k = \exp(-j2\pi k/N)$ and $u_k(n) = \exp(j2\pi nk/N)/\sqrt{N} = \lambda_k^{-n}/\sqrt{N}$, with $h_n = h(n)$ and $H(\lambda_k) = \sum_{n=0}^{N-1} h(n)u_k^*(n)$.

The previous relation will be used to define the $z$-transform of a graph signal. For given signal $\mathbf{x} = [x(0)\ x(1)\dots x(N-1)]^T$ the signal corresponding to a system transfer function that would have the same GDFT is

$$[x_0\ x_1 \dots x_{N-1}]^T = \mathbf{V}_\lambda^{-1}\mathbf{U}^T[x(0)\ x(1)\dots x(N-1)]^T.$$

The $z$-transform of these coefficients is

$$X(z^{-1}) = \mathcal{Z}\{x_n\} = x_0 + x_1 z^{-1} + \dots + x_{N-1}z^{-(N-1)}. \tag{48}$$

For this $z$-transform holds

$$Y(z^{-1}) = H(z^{-1})X(z^{-1}).$$

The output signal $y(n)$ can be obtained as

$$[y(0)\ y(1)\dots y(N-1)]^T = \mathbf{U}\mathbf{V}_\lambda[y_0\ y_1\dots y_{N-1}]^T,$$

where the output graph signal $y(n)$ is obtained from the inverse $z$-transform of the coefficients $y_n$ of $Y(z^{-1}) = H(z^{-1})X(z^{-1})$

$$Y(z^{-1}) = \mathcal{Z}\{y_n\} = y_0 + y_1 z^{-1} + \dots + y_{N-1}z^{-(N-1)}.$$

The $z$-transform representation may be of interest when the eigenvalues are complex-valued. They may appear in decomposition of adjacency matrices of undirected graphs. For example, for the graph from Fig. 2(b) and its adjacency matrix the eigenvalues are presented in Fig. 28.

The analytic signal and Hilbert transform are defined as

$$X_a(k) = (1 + \operatorname{sign}(\operatorname{Imag}(\lambda_k)))X(k)$$

$$X_h(k) = j\operatorname{sign}(\operatorname{Imag}(\lambda_k))X(k)$$

$$X(k) = X_a(k) + jX_h(k)$$

If these relations are applied to the standard DFT with $\lambda_k = \exp(-j2\pi k/N)$ we would get the classical signal processing definitions.

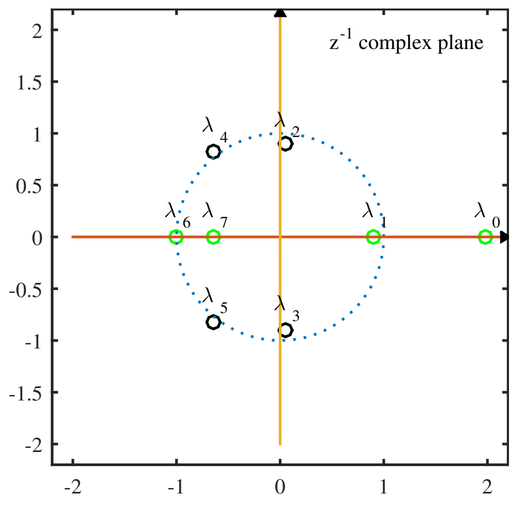


**Fig. 28** Eigenvalues of the directed graph adjacency matrix.

## 3.10 Shift in the Spectral Domain

We can define a shift in the spectral domain in the same way as the shift in the vertex domain has been defined. Consider a product of two signals $x(n)y(n)$ on an undirected graph. Its GDFT is

$$\text{GDFT}\{x(n)y(n)\} = \sum_{n=0}^{N-1} x(n)y(n)u_k(n) =$$

$$\sum_{n=0}^{N-1}\sum_{i=0}^{N-1} X(i)u_i(n)y(n)u_k(n) = \sum_{i=0}^{N-1} X(i)Y_i(k),$$

where

$$Y_i(k) = \sum_{n=0}^{N-1} y(n)u_i(n)u_k(n)$$

can be considered as a shift of $Y(k)$ for $i$. Obviously $Y_0(k) = Y(k)$ up to a constant factor. This relation does not hold for the shift in the vertex domain.

## 3.11 Parseval's Theorem on a Graph

For two signals $x(n)$ and $y(n)$ on an undirected graph and their spectra $X(k)$ and $Y(k)$, Parseval's theorem holds

$$\sum_{n=0}^{N-1} x(n)y(n) = \sum_{k=0}^{N-1} X(k)Y(k). \tag{49}$$

To prove Parseval's theorem on graphs we can write

$$\sum_{n=0}^{N-1} x(n)y(n) = \sum_{n=0}^{N-1} \left[ \sum_{k=0}^{N-1} X(k)u_k(n) \right] y(n) = \sum_{k=0}^{N-1} X(k) \sum_{n=0}^{N-1} y(n)u_k(n), \quad (50)$$

producing Parseval's theorem. It has been assumed that $\mathbf{U}^{-1} = \mathbf{U}^T$ for undirected graphs. This theorem holds for both the Laplacian and the adjacency matrix based decompositions on undirected graphs.

## 3.12 Optimal Denoising

Consider a signal composed of a slow-varying signal $\mathbf{s}$ and a fast changing disturbance $\boldsymbol{\varepsilon}$

$$\mathbf{x} = \mathbf{s} + \boldsymbol{\varepsilon}.$$

The aim is to design a filter for disturbance suppression (denoising). The output of this filter is denoted by $\mathbf{y}$.

The optimal denoising task could be defined as a minimization of

$$J = \frac{1}{2}\|\mathbf{y} - \mathbf{x}\|_2^2 + \alpha \mathbf{y}^T \mathbf{L} \mathbf{y}.$$

Before we solve this problem we will explain the meaning of terms in the cost function $J$. Minimization of the first term $\frac{1}{2}\|\mathbf{y} - \mathbf{x}\|_2^2$ forces that the output signal $\mathbf{y}$ is as close to the available observations $\mathbf{x}$ as possible. The second term is a measure of signal $\mathbf{y}$ smoothness. It promotes the solution smoothness (see Section 3.6). Parameter $\alpha$ is a balance between output closeness to $\mathbf{x}$ and smoothness of $\mathbf{y}$ criterion.

The solution of the minimization problem is

$$\frac{\partial J}{\partial \mathbf{y}^T} = \mathbf{y} - \mathbf{x} + 2\alpha \mathbf{L}\mathbf{y} = 0$$

resulting in

$$\mathbf{y} = (\mathbf{I} + 2\alpha \mathbf{L})^{-1}\mathbf{x}.$$

The Laplacian spectral domain form of this relation follows with $\mathbf{L} = \mathbf{U}^T \boldsymbol{\Lambda} \mathbf{U}$, $\mathbf{Y} = \mathbf{U}^T \mathbf{y}$, and $\mathbf{X} = \mathbf{U}^T \mathbf{x}$ as

$$\mathbf{Y} = (\mathbf{I} + 2\alpha \boldsymbol{\Lambda})^{-1}\mathbf{X}.$$

The transfer function of this filter is

$$H(\lambda_k) = \frac{1}{1 + 2\alpha\lambda_k}.$$

For a small $\alpha$, $H(\lambda_k) \approx 1$ and $\mathbf{y} \approx \mathbf{x}$. For a large $\alpha$, $H(\lambda_k) \approx \delta(k)$ and $\mathbf{y} \approx$ *const.* is maximally smooth (a constant, without any variation).

Here we will state two more cost function forms used for graph signal denoising.

Instead of the resulting signal smoothness, we may add the condition that its deviation from a linear form is as small as possible. Then the cost function is

$$J = \frac{1}{2}\|\mathbf{y}-\mathbf{x}\|_2^2 + \alpha\|\mathbf{L}\mathbf{y}\|_2^2 = \frac{1}{2}\|\mathbf{y}-\mathbf{x}\|_2^2 + \alpha\mathbf{y}^T\mathbf{L}^2\mathbf{y}$$

resulting in a closed form solution

$$\mathbf{y} = (\mathbf{I} + 2\alpha\mathbf{L}^2)^{-1}\mathbf{x}$$

with the corresponding spectral domain relation $H(\lambda_k) = 1/(1+2\alpha\lambda_k^2)$.

A combination of the previous two cost function forms may provide additional flexibility in the transfer function design. If we use

$$J = \frac{1}{2}\|\mathbf{y}-\mathbf{x}\|_2^2 + \alpha\mathbf{y}^T\mathbf{L}\mathbf{y} + \beta\mathbf{y}^T\mathbf{L}^2\mathbf{y}$$

we would get the transfer function

$$H(\lambda_k) = \frac{1}{1+2\alpha\lambda_k+2\beta\lambda_k^2}.$$

We can change the transfer function form by choosing appropriate values of the parameters $\alpha$ and $\beta$. For example, if we want the component corresponding to $\lambda_1 \neq 0$ to be unattenuated we would use $\alpha+\beta\lambda_1 = 0$. This cost function can be extended to produce a transfer function for $M$ unattenuated components.

In some applications we would like to promote the sparsity of the resulting signal change, instead of its smoothness. Then the compressive sensing theory requires that the quadratic form or the norm-two in the previous equations is replaced with the forms that promote sparsity. Two possible such cost functions are:

$$J = \frac{1}{2}\|\mathbf{y}-\mathbf{x}\|_2^2 + \alpha\|\mathbf{L}\mathbf{y}\|_p^p$$

and

$$J = \frac{1}{2}\sum_{n=0}^{N-1}(y(n)-x(n))^2 + \alpha\sum_{n=0}^{N-1}\left(\sum_{m=0}^{N-1}W_{nm}(y(n)-y(m))^2\right)^{p/2}$$

with $0 \leq p \leq 1$. Minimization of these functions can not be done in an analytic way, like in the case of $p=2$. The norm-zero, with $p=0$, is the best in promoting sparsity. For $p=0$, the second term in minimization counts and minimizes the number of nonzero elements in $\mathbf{L}\mathbf{y}$. In the second minimization form the norm-zero promotes the smallest possible number of nonzero elements of the form $\sum_{m=0}^{N-1}W_{nm}(y(n)-y(m))^2$. This is the total variations (TV) approach.

The norm-one with $p=1$ in previous relations is convex, allowing the gradient descend methods in the solution, while producing the same solution as with $p=0$, under some conditions.

## 4 Subsampling, Compressed Sensing, and Reconstruction

Graphs may have a large number of vertices. This number can be of the order of millions or even higher. The fact that the number of vertices and corresponding graph signal values can be extremely large makes the problem of subsampling and compressive sensing crucially important in graph signal processing. The problems of subsampling and compressive sensing are closely related to the reconstruction possibility from a reduced set of measurements (signal samples or their linear combinations). Here we will present several basic approaches to the subsampling, along with their relations to classical signal processing [40–58].

### 4.1 Subsampling of the Low-Pass Graph Signals

We will start with the simplest case where we can assume that the considered graph signal is of low-pass type. This signal can be written as a linear combination of $K < N$ eigenvectors with the slowest changes. For example, for the Laplacian spectrum of a signal with $K$ nonzero values

$$\mathbf{X} = [X(0), X(1), \dots, X(K-1), 0, 0, \dots, 0]^T$$

the signal is of form

$$x(n) = \sum_{k=0}^{K-1} X(k) u_k(n).$$

The smallest number of graph signal samples needed to recover this signal is $M = K < N$. Assume that $M$ signal samples are available, $K \leq M < N$. The vector of available graph signal samples will be referred to as the measurement vector. It will be denoted by $\mathbf{y}$. The set of vertices where the graph signal samples are available is

$$\mathcal{M} = \{n_1, n_2, \dots, n_M\}.$$

The measurement matrix can be defined using the IGDFT $\mathbf{x} = \mathbf{U}\,\mathbf{X}$ or

$$x(n) = \sum_{k=0}^{N-1} u_k(n) X(k), \;\; n = 0, 1, \dots, N.$$

Keeping the equations corresponding to the available graph signal samples at $n \in \mathcal{M} = \{n_1, n_2, \dots, n_M\}$ we get

$$\begin{bmatrix} x(n_1) \\ x(n_2) \\ \vdots \\ x(n_M) \end{bmatrix} = \begin{bmatrix} u_0(n_1) & u_1(n_1) & \dots & u_{N-1}(n_1) \\ u_0(n_2) & u_1(n_2) & \dots & u_{N-1}(n_2) \\ \vdots & & & \\ u_0(n_M) & u_1(n_M) & \dots & u_{N-1}(n_M) \end{bmatrix} \begin{bmatrix} X(0) \\ X(1) \\ \vdots \\ X(N-1) \end{bmatrix}.$$

The matrix form of this system of equations is

$$\mathbf{y} = \mathbf{A}_{MN}\mathbf{X},$$

where $\mathbf{A}_{MN}$ is the measurement matrix and $\mathbf{y} = [x(n_1), x(n_2), \dots, x(n_M)]^T$ are the available samples. This system is underdetermined for $M < N$. It cannot be solved uniquely for $\mathbf{X}$ without additional constraints. Since we have assumed that the signal contains a linear combination of only $K \leq M$ the slowest varying eigenvectors, we can exclude the GDFT coefficients $X(K), X(K+1), \dots, X(N-1)$, since they are zero-valued and do not contribute to the graph signal samples. Then we can write

$$\begin{bmatrix} x(n_1) \\ x(n_2) \\ \vdots \\ x(n_K) \end{bmatrix} = \begin{bmatrix} u_0(n_1) & u_1(n_1) & \dots & u_{K-1}(n_1) \\ u_0(n_2) & u_1(n_2) & \dots & u_{K-1}(n_2) \\ \vdots & & & \\ u_0(n_M) & u_1(n_M) & \dots & u_{K-1}(n_M) \end{bmatrix} \begin{bmatrix} X(0) \\ X(1) \\ \vdots \\ X(K-1) \end{bmatrix}.$$

This system in a matrix form reads

$$\mathbf{y} = \mathbf{A}_{MK}\mathbf{X}_K,$$

where the definition of the reduced measurement matrix $\mathbf{A}_{MK}$ and the reduced GDFT vector $\mathbf{X}_K$ is obvious. For $K = M$ this system can be solved. If $M > K$ the system is overdetermined and the solution is found in the mean squared error (MSE) sense. This solution is

$$\mathbf{X}_K = (\mathbf{A}_{MK}^T\mathbf{A}_{MK})^{-1}\mathbf{A}_{MK}^T\mathbf{y} = \mathrm{pinv}(\mathbf{A}_{MK})\mathbf{y},$$

where $\mathrm{pinv}(\mathbf{A}_{MK}) = (\mathbf{A}_{MK}^T\mathbf{A}_{MK})^{-1}\mathbf{A}_{MK}^T$ is a matrix pseudo-inverse.

After $\mathbf{X}_K$ is calculated all GDFT values directly follow by adding the assumed zero values as $\mathbf{X} = [X(0), X(1), \dots, X(K-1), 0, 0, \dots, 0]^T$ . The graph signal is then recovered at all vertices using $\mathbf{x} = \mathbf{U}\,\mathbf{X}$.

The recovery condition is that the inverse $(\mathbf{A}_{MK}^T\mathbf{A}_{MK})^{-1}$ exists. It means that

$$\mathrm{rank}(\mathbf{A}_{MK}^T\mathbf{A}_{MK}) = K. \tag{51}$$

In terms of the matrix condition number, the requirement is

$$\mathrm{cond}(\mathbf{A}_{MK}^T\mathbf{A}_{MK}) < \infty.$$

In the case of noisy measurements, the noise in the reconstructed GDFT coefficients is directly related to the input noise and the matrix condition number. If we are able to choose the signal sample positions (vertices), then the sampling strategy should be to find the set of measurements producing the condition number as close to one as possible.

As an example of the reconstruction from a reduced set of signal samples, consider a graph's signal values at $M = 3$ vertices

$$\mathbf{y} = [x(0) \;\; x(2) \;\; x(6)]^T = [0.299 \;\; 0.345 \;\; 1.361]^T.$$

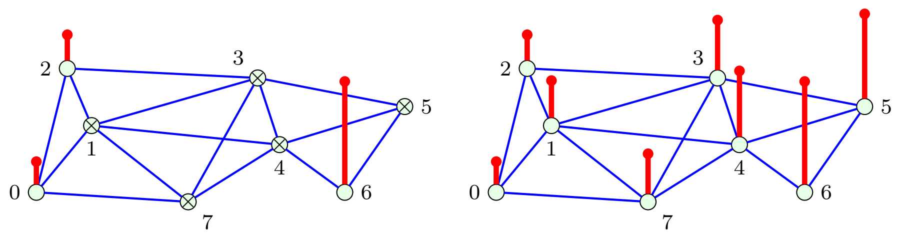


**Fig. 29** Subsampling of a lowpass graph signal example.

Assume that the graph signal is of low-pass type with $K = 2$ lowest nonzero GDFT coefficients $X(0)$ and $X(1)$. The signal GDFT coefficients can be reconstructed from

$$\begin{bmatrix} x(0) \\ x(2) \\ x(6) \end{bmatrix} = \begin{bmatrix} u_0(0) & u_1(0) \\ u_0(2) & u_1(2) \\ u_0(6) & u_1(6) \end{bmatrix} \begin{bmatrix} X(0) \\ X(1) \end{bmatrix}.$$

since the rank of matrix $\mathbf{A}_{MK}$ is 2. The matrix condition number value is $\text{cond}(\mathbf{A}_{MK}^T \mathbf{A}_{MK}) = 1.97$. The reconstructed nonzero values of the GDFT are $X(0) = 2$ and $X(1) = 1$. The reconstructed graph signal $\mathbf{x} = \mathbf{U}\,\mathbf{X}$ is presented in Fig. 29.

The classical signal processing downsampling and interpolation relations are obtained with $u_k(n) = \exp(j2\pi nk/N)/\sqrt{N}$.

## 4.2 Subsampling of the Sparse Graph Signals

### *4.2.1 Known Coefficient Positions*

The previous analysis holds not only for a low-pass type of $\mathbf{X}$, but for a general $\mathbf{X}$ with $K$ nonzero values at arbitrary, known, spectral positions,

$$X(k) = 0 \text{ for } k \notin \mathcal{K} = \{k_1, k_2, \dots, k_K\}$$

as well. Then the system of equations

$$\begin{bmatrix} x(n_1) \\ x(n_2) \\ \vdots \\ x(n_M) \end{bmatrix} = \begin{bmatrix} u_{k_1}(n_1) & u_{k_2}(n_1) & \dots & u_{k_K}(n_1) \\ u_{k_1}(n_2) & u_{k_2}(n_2) & \dots & u_{k_K}(n_2) \\ \vdots & & & \\ u_{k_1}(n_M) & u_{k_2}(n_M) & \dots & u_{k_K}(n_M) \end{bmatrix} \begin{bmatrix} X(k_1) \\ X(k_2) \\ \vdots \\ X(k_K) \end{bmatrix}, \tag{52}$$

whose matrix form reads $\mathbf{y} = \mathbf{A}_{MK}\mathbf{X}_K$, is solved for the nonzero spectral values $X(k)$, $k \in \mathcal{K}$, in the same way as in the case of low-pass signal presented in Section 4.1.

### *4.2.2 Support Matrices, Subsampling-Upsampling*

In graph signal processing literature, *the subsampling problem is often defined using the so called support matrices.* Assume that a graph signal, $\mathbf{x}$, is subsampled in such way that it is available on a subset of vertices $n \in \mathbb{M} = \{n_1, n_2, \dots, n_M\}$, 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

$$\mathbf{x}_s = \mathbf{B}\mathbf{x}, \tag{53}$$

where *the support matrix* $\mathbf{B}$ is an $N \times N$ diagonal matrix with ones at the diagonal positions which correspond to $\mathbb{M} = \{n_1, n_2, \dots, n_M\}$ 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 GDFT domain, the additional constraint is that the signal, $\mathbf{x}$, has only $K \leq M$ nonzero coefficients in the GDFT domain, $\mathbf{X} = \mathbf{U}^T\mathbf{x}$, at $k \in \mathbb{K} = \{k_1, k_2, \dots, k_K\}$, so that the relation

$$\mathbf{X} = \mathbf{C}\mathbf{X}$$

holds, where the support matrix $\mathbf{C}$ is an $N \times N$ diagonal matrix with ones at the diagonal positions which correspond to $\mathbb{K} = \{k_1, k_2, \dots, k_K\}$ and zeros elsewhere. Note the presence of the GDFT, $\mathbf{X}$, is on both sides of this equation, contrary to $\mathbf{x}_s = \mathbf{B}\mathbf{x}$ in (53). The reconstruction formula then follows from

$$\mathbf{x}_s = \mathbf{B}\mathbf{x} = \mathbf{B}\mathbf{U}\mathbf{X} = \mathbf{B}\mathbf{U}\mathbf{C}\mathbf{X}.$$

as $\mathbf{X} = \operatorname{pinv}\big(\mathbf{B}\mathbf{U}\mathbf{C}\big)\mathbf{x}_s$. The inversion

$$\mathbf{X} = \mathbf{C}\mathbf{X} = \operatorname{pinv}\big(\mathbf{B}\mathbf{U}\mathbf{C}\big)\mathbf{x}_s$$

is possible for $K$ nonzero coefficients of $\mathbf{C}\mathbf{X}$ if the rank of $\mathbf{B}\mathbf{U}\mathbf{C}$ is $K$ (if there are $K$ linearly independent equations), that is

$$\operatorname{rank}(\mathbf{C}) = K = \operatorname{rank}\big(\mathbf{B}\mathbf{U}\mathbf{C}\big).$$

This condition is equivalent to (51) since the nonzero part of matrix $\mathbf{B}\mathbf{U}\mathbf{C}$ is equal to $\mathbf{A}_{MK}$ in (52).

### *4.2.3 Unknown Coefficient Positions*

The problem is more complex if the positions of nonzero spectral coefficients $\mathcal{K} = \{k_1, k_2, \dots, k_K\}$ are not known. This problem has been formulated and solved within compressive sensing theory. The reconstruction problem formulation is

$$\min \|\mathbf{X}\|_0 \text{ subject to } \mathbf{y} = \mathbf{A}_{MN}\mathbf{X},$$

where $\|\mathbf{X}\|_0$ denotes the number of nonzero elements in $\mathbf{X}$ ($\ell_0$ pseudo-norm).

The minimization problem can be solved in many ways. Here we will present a simple, two step solution:

1. Using $M \gg K$ signal samples, the positions $\mathcal{K}$ of nonzero coefficients are estimated.
2. The nonzero coefficients of $\mathbf{X}$ at the estimated positions $\mathcal{K}$ are reconstructed, along with the signal $\mathbf{x}$ at all vertices.

For the estimation of nonzero positions in Step 1 we can use the projection of measurements to the measurement matrix

$$\mathbf{A}_{MN} = \begin{bmatrix} u_0(n_1) & u_1(n_1) & \dots & u_{N-1}(n_1) \\ u_0(n_2) & u_1(n_2) & \dots & u_{N-1}(n_2) \\ \vdots & & & \\ u_0(n_M) & u_1(n_M) & \dots & u_{N-1}(n_M) \end{bmatrix}$$

defined as

$$\mathbf{X}_0 = \mathbf{A}_{MN}^T\mathbf{y}. \tag{54}$$

Positions of $K$ largest values in $\mathbf{X}_0$ are used as an estimate of the nonzero positions $\mathcal{K}$. This procedure can also be implemented in an iterative way. In the first iteration we assume $K = 1$ and the largest component is estimated and removed. The position of the second component is then estimated. The first and the second component are then reconstructed and removed. The procedure is iteratively repeated $K$ times.

As an example consider a sparse graph signal with sparsity $K = 2$ measured at vertices 2, 3, 4, 5, and 7

$$\mathbf{y} = [0.224 \quad 1.206 \quad 1.067 \quad 1.285 \quad 1.116]^T.$$

Measurements (available signal samples) are illustrated in Fig. 30 (upper left subplot). The estimate $\mathbf{X}_0$ is calculated according to (54). The positions of two non-zero coefficients are estimated as positions of two largest values in $\mathbf{X}_0$. In the considered case $\mathcal{K} = \{0, 2\}$, Fig. 30 (lower left subplot). The GDFT coefficients are then reconstructed for sparsity $K = 2$, resulting in $X(0) = 2$, $X(2) = 1.5$, Fig. 30 (lower right). The reconstructed graph signal at all vertices is presented in Fig. 30 (upper right subplot).

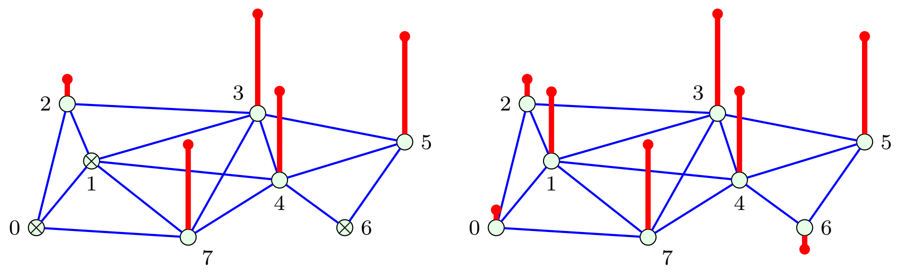


**Fig. 30** Compressive sensing on graphs

### *4.2.4 On the Unique Reconstruction Conditions*

It is easy to show that the initial 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 maximal value of the inner product of two columns of the measurement matrix $\mathbf{A}_{MN}$ (the coherence index).

A $K$-sparse signal can be written as $x(n) = \sum_{i=1}^{K} X(k_i)u_{k_i}(n)$. Its initial estimate values are

$$X_0(k) = \sum_{i=1}^{K} X(k_i) \sum_{n\in\mathcal{M}} u_k(n)u_{k_i}(n) = \sum_{i=1}^{K} X(k_i)\mu(k,k_i)$$

where $\mathcal{M} = \{n_1, n_2, \dots, n_M\}$ and $\mu(k,k_i) = \sum_{n\in\mathcal{M}} u_k(n)u_{k_i}(n)$. If the maximal possible absolute value of $\mu(k,k_i)$ is denoted by $\mu = \max|\mu(k,k_i)|$ then, in the worst case, the amplitude of the strongest component $X(k_i)$ (assumed with the normalized amplitude 1), reduced for the maximal possible influence of other equally strong (unity) components $1-(K-1)\mu$, should be greater than the maximal possible disturbance at $k \neq k_i$, being $K\mu$. From $1-(K-1)\mu > K\mu$, the unique reconstruction condition follows.

In order to define other unique reconstruction conditions we will consider again the solution of $\mathbf{y} = \mathbf{A}_{MN}\mathbf{X}$ with a minimal number of nonzero coefficients in $\mathbf{X}$. Assume that the sparsity $K$ is known. Then a set of $K$ measurements would produce a possible solution for any combination of $K$ nonzero coefficients in $\mathbf{X}$. If we assume that we have another set of $K$ measurements, we would get another set of possible solutions. A common solution in these two sets of solutions would be the solution of our problem. There are no two different $K$ sparse solutions $\mathbf{X}_K^{(1)}$ and $\mathbf{X}_K^{(2)}$ (the solution is unique) if all possible $\mathbf{A}_{M2K}^T\mathbf{A}_{M2K}$ matrices are nonsingular. The requirement that all reduced measurement matrices corresponding to a $2K$ sparse $\mathbf{X}$ are nonsingular can be written in several forms,

$$\det\{\mathbf{A}_{M2K}^T\mathbf{A}_{M2K}\} = d_1 d_2 \dots d_{2K} \neq 0$$

$$\operatorname{cond}\{\mathbf{A}_{M2K}^T\mathbf{A}_{M2K}\} = \frac{d_{\max}}{d_{\min}} \leq \frac{1+\delta_{2K}}{1-\delta_{2K}} < \infty$$

$$1-\delta_{2K} \leq d_{\min} \leq \{\frac{\|\mathbf{A}_{M2K}\mathbf{X}_{2K}\|_2^2}{\|\mathbf{X}_{2K}\|_2^2}\} \leq d_{\max} \leq 1+\delta_{2K}$$

where $d_i$ are the eigenvalues of $\mathbf{A}_{M2K}^T\mathbf{A}_{M2K}$, $d_{\min}$ is the minimal eigenvalue, $d_{\max}$ is the maximal eigenvalue, and $\delta_{2K}$ is the restricted isometry constant.

All these conditions are satisfied if $d_{\min} > 0$ or $0 \leq \delta_{2K} < 1$.

In noisy cases robustness is required, and therefore more strict bounds for $d_{\min}$ and $\delta_{2K}$ are required. For example, it has been shown, that $0 \leq \delta_{2K} < 0.41$ will guarantee stable inversion and robust reconstruction of the noisy signals. In addition, this bound will allow convex relaxation of the reconstruction problem.

Namely, the previous problem can be solved using the convex relation of the norm-zero to a norm-one formulation

$$\min \|\mathbf{X}\|_1 \text{ subject to } \mathbf{y} = \mathbf{A}_M\mathbf{X}.$$

The solution of these two problem formulation are the same if the measurement matrix satisfies the previous conditions with $\delta_{2K} < 0.41$. The signal reconstruction problem can now be solved using gradient-based approaches or linear programming methods.

## 4.3 Linear Combinations of Samples and Aggregated Sampling

If some spectrum coefficients are strongly related to only a few of the signal samples, then the signal samples may not be good candidates for the measurements. In that case linear combinations of all signal samples

$$\mathbf{y} = \mathbf{B}_{MN}\mathbf{x} = \mathbf{B}_{MN}\mathbf{U}\mathbf{X} = \mathbf{A}_{MN}\mathbf{X}$$

should be used. The weighting coefficients $\mathbf{B}_{MN}$ for the measurements could be, for example, the Gaussian random numbers. The reconstruction is obtained as the solution of

$$\min \|\mathbf{X}\|_0 \text{ subject to } \mathbf{y} = (\mathbf{B}_{MN}\mathbf{U})\mathbf{X}$$

or the solution of the corresponding convex minimization problem.

A specific form of linear combinations of the graph signals are described as aggregate sampling. Sampling in classical signal processing can be interpreted on a directed circular graph (Fig. 19) in the following way. Consider the signal at an instant $n$. If we sense this signal at this vertex/instant only, we get its value $y_0(n) = x(n)$. If we apply the shift operator we get $\mathbf{y}_1 = \mathbf{Ax}$. If this signal is sampled at the same vertex $n$ we get $y_1(n) = x(n-1)$. If we continue this shift and sample operation $N$ times we will get all signal values $x(n), x(n-1), \dots, x(n-N+1)$. If we stop the shifts after $M < N$ steps the signal can still be recovered using the compressive sensing based reconstruction methods, if the reconstruction conditions are met.

The same procedure can be used on a signal on an arbitrary graph. Assume that we sample the graph signal at a vertex $n$ and get

$$y_0(n) = x(n).$$

If the signal is now graph shifted $\mathbf{y}_1 = \mathbf{Ax}$ we will get a new measurement as a shifted signal value at the considered vertex,

$$y_1(n) = \sum_m A_{nm} x(m).$$

If we continue with one more shift we will get

$$y_2(n) = \sum_m A_{nm}^{(2)} x(m),$$

where $A_{nm}^{(2)}$ are the elements of matrix $\mathbf{A}^2 = \mathbf{AA}$. If we continue $M = N$ times we would get a system of $N$ linear equations $\mathbf{y} = \mathbf{B}_{MN}\mathbf{x}$. From these equations we can calculate all signal values $x(n)$. If we stop at $M < N$ the signal can still be recovered using the 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 other $M - P$ samples from a vertex $m$. Other combinations of vertices and samples can be used to obtain $M$ measurements and to fully reconstruct a signal.

## 4.4 On the Sampling Strategies

Signal sampling strategy is a process that will result in an optimal set of signal samples which will guarantee a unique reconstruction of a sparse signal. In classical signal processing, the sampling strategies are based on the minimization of parameters defining the solution uniqueness. In the case when the positions of nonzero GDFT coefficients are known (including the low-pass filtering as a special case) the requirement is that the rank of the measurement matrix is equal to the signal sparsity. In more general cases when the nonzero coefficient positions are not known, the restricted isometry property is the most commonly used uniqueness criterion. However, its application in practice is almost impossible, since it is an NP hard combinatorial problem (requiring $2K$ class combinations of $N$ elements). A sampling strategy that will minimize the coherence index will guarantee the maximal number of nonzero coefficients reconstruction with a given set of signal samples. This process is less computationally intensive. However, the coherence index based criterion for signal sampling strategy is quite pessimistic.

In graph signals, the application of these sampling strategy criteria is even more difficult. The number of vertices may be extremely large. Large dimensionality of the graphs makes these approaches almost unsuitable for graph signal processing.

Subsampling of graphs is of crucial importance in the cases of extremely large graphs. Several approaches are possible in the graph analysis and graph signal processing.

We have already described a possible graph signal orineted subsampling strategy under the assumption that the GDFT is sparse, with a few nonzero coefficients. Then the graph signal can be reconstructed with a very reduced number of signal samples or their linear combinations.

Another class of approaches is graph oriented. The problem is defined as follows. Given a large, in general directed, graph $\mathcal{G}$ with $N$ vertices, the goal is to find a much simpler graph with similar properties. In this so called down-scale method, the goal is to define a smaller size graph $\mathcal{S}$ with a very reduced number of vertices $M \ll N$ that will behave in a similar way as the original large graph. Similarity is defined with respect to the parameters of interest, like for example, the connectivity or distribution on graph. The criteria may be related to the spectral behavior of graphs as well.

Several methods are used for graph down-scaling. Some of them will be listed next:

- The simplest method for graph down-sampling is a random vertex or random node (RN) selection method. In this method, a random subset of vertices is used for analysis and representation of large graphs and signals on them. Vertices are selected with equal probabilities. This method will produce good results in many applications. Random selection has been preferred in classical compressive signal analysis as well.
- Different from the RN method, where the vertices are selected with a uniform probability, is the random degree vertex/node (RDN) selection, in which the probability that a vertex is selected is proportional to the vertex degree. Vertices with more connections, having larger $d_n = \sum_m W_{nm}$, are selected with a higher probability. This approach is biased with respect to highly connected vertices.
- A similar method to the RDN is based on the vertex rank (PageRank). The PageRank is defined by the importance of the vertices connected to the considered vertex $n$ (see Section 7.8.5). 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 called random PageRank vertex (RPN) selection. It is also biased with respect to the highly connected vertices with a high PageRank.

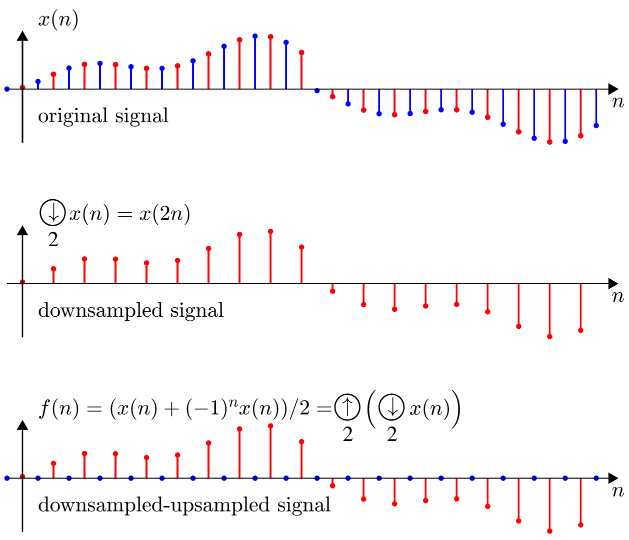


**Fig. 31** Principle of a signal, $x(n)$, downsampling and upsampling in the classical time domain.

– A method based on the random selection of edges, that will be left in the simpler graph, is called the random edge (RE) method. This method may lead to graphs that are not well connected, with large diameters.
– The RE method may be combined with the random vertex selection to get a combined RNE method. It consists in a random vertex selection followed by a random selection of one of the edges corresponding to the selected vertex.
– In addition to these methods, more sophisticated methods based on the random vertex selection and the random walks (RW) analysis may be defined. For example, we can randomly select small subset of vertices and form several random walks starting from each selected vertex. In this way Random Walk (RW), Random Jump (RJ) and Forest Fire graph downscaling strategies are defined.

## 4.5 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

$$f(n) = \frac{1}{2}\Big(x(n) + (-1)^n x(n)\Big) = \frac{1}{2}\Big((1 + (-1)^n)x(n)\Big),$$

as illustrated in Fig. 31.

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.

This combined subsampling-upsampling operation produces a graph signal

$$f(n) = \frac{1}{2}\Big(1 + (-1)^{\beta_{\mathcal{E}}(n)}\Big)x(n),$$

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 subsamped-upsampled graph signal, $f(n)$, which comprises all $n \in \mathcal{V}$, is given by

$$\mathbf{f} = \frac{1}{2}(\mathbf{x} + \mathbf{J}_{\mathcal{E}}\mathbf{x}) = \frac{1}{2}(\mathbf{I} + \mathbf{J}_{\mathcal{E}})\mathbf{x}, \tag{55}$$

where $\mathbf{J}_{\mathcal{E}} = \mathrm{diag}((-1)^{\beta_{\mathcal{E}}(n)}),\ n \in \mathcal{V}$.

The focus of our analysis will be on the two-channel wavelet filter bank on a graph, shown in Fig. 32. As in the classical wavelet analysis framework for temporary signals, such a filter bank provides decomposition of a graph signal into the corresponding low-pass (smooth) and high-pass (fast-varying) constituents. The analysis side (left part of the system in Fig. 32) 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 $H_L(\mathbf{\Lambda})$ and $H_H(\mathbf{\Lambda})$. 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 Fig. 32), comprises the complementary upsampling and filtering operations, aiming to perform the graph signal reconstruction based on the upsampled versions, $\frac{1}{2}(\mathbf{I} + \mathbf{J}_{\mathcal{E}})H_L(\mathbf{L})\mathbf{x}$ and $\frac{1}{2}(\mathbf{I} + \mathbf{J}_{\mathcal{H}})H_H(\mathbf{L})\mathbf{x}$, 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, 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.

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

$$\lambda_k = 2 - \lambda_{N-k} \tag{56}$$

$$\mathbf{u}_k = \begin{bmatrix} \mathbf{u}_{\mathcal{E}} \\ \mathbf{u}_{\mathcal{H}} \end{bmatrix} \quad \text{and} \quad \mathbf{u}_{N-k} = \begin{bmatrix} \mathbf{u}_{\mathcal{E}} \\ -\mathbf{u}_{\mathcal{H}} \end{bmatrix}, \tag{57}$$

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

$$\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{A}_{\mathcal{E}\mathcal{H}} \\ \mathbf{A}_{\mathcal{E}\mathcal{H}}^T & \mathbf{0} \end{bmatrix} \quad \text{and} \quad \mathbf{L}_N = \begin{bmatrix} \mathbf{I} & \mathbf{L}_{\mathcal{E}\mathcal{H}} \\ \mathbf{L}_{\mathcal{E}\mathcal{H}}^T & \mathbf{I} \end{bmatrix}.$$

The eigenvalue relation, $\mathbf{L}_N \mathbf{u}_k = \lambda_k \mathbf{u}_k$, can now be evaluated as

$$\mathbf{L}_N \mathbf{u}_k = \begin{bmatrix} \mathbf{u}_{\mathcal{E}} + \mathbf{L}_{\mathcal{E}\mathcal{H}} \mathbf{u}_{\mathcal{H}} \\ \mathbf{L}_{\mathcal{E}\mathcal{H}}^T \mathbf{u}_{\mathcal{E}} + \mathbf{u}_{\mathcal{H}} \end{bmatrix} = \lambda_k \begin{bmatrix} \mathbf{u}_{\mathcal{E}} \\ \mathbf{u}_{\mathcal{H}} \end{bmatrix}.$$

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 $\mathbf{L}_{\mathcal{E}\mathcal{H}} \mathbf{u}_{\mathcal{H}} = (\lambda_k - 1)\mathbf{u}_{\mathcal{E}}$ and $\mathbf{L}_{\mathcal{E}\mathcal{H}}^T \mathbf{u}_{\mathcal{E}} = (\lambda_k - 1)\mathbf{u}_{\mathcal{H}}$, to finally yield

$$\mathbf{L}_N \begin{bmatrix} \mathbf{u}_{\mathcal{E}} \\ -\mathbf{u}_{\mathcal{H}} \end{bmatrix} = (2 - \lambda_k) \begin{bmatrix} \mathbf{u}_{\mathcal{E}} \\ -\mathbf{u}_{\mathcal{H}} \end{bmatrix}.$$

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 (56), it then follows that the largest eigenvalue is $\lambda_N = 2$, which also proves the property $L_7$ for a bipartite graph.

Consider a graph signal, $\mathbf{x}$, and the filter-bank as in Fig. 32. 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 (55), the downsampled-upsampled form of the output

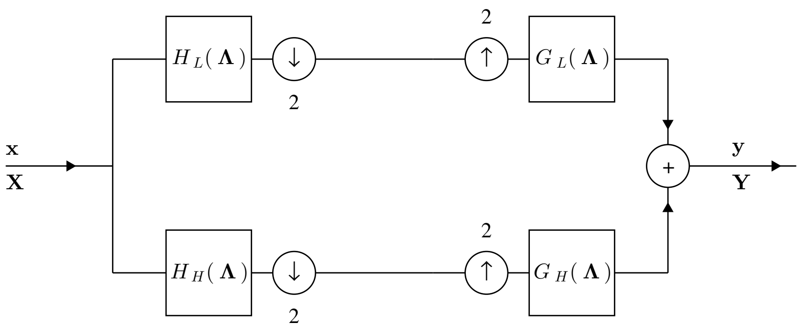


**Fig. 32** Principle of a filter bank for a graph signal.

signal, $H_L(\mathbf{L})\mathbf{x}$, is given by $\frac{1}{2}(\mathbf{I}+\mathbf{J}_\mathcal{E})H_L(\mathbf{L})\mathbf{x}$. After the syntheses filter, $G_L(\mathbf{L})$, the graph signal output becomes

$$\mathbf{f}_L = \frac{1}{2}G_L(\mathbf{L})(\mathbf{I}+\mathbf{J}_\mathcal{E})H_L(\mathbf{L})\mathbf{x}. \tag{58}$$

The same holds for the high-pass part

$$\mathbf{f}_H = \frac{1}{2}G_H(\mathbf{L})(\mathbf{I}+\mathbf{J}_\mathcal{H})H_H(\mathbf{L})\mathbf{x}, \tag{59}$$

where $\mathbf{J}_\mathcal{H} = -\mathbf{J}_\mathcal{E} = \mathrm{diag}((-1)^{1-\beta_\mathcal{E}(n)})$ and

$$\mathbf{J}_\mathcal{H} + \mathbf{J}_\mathcal{E} = \mathbf{0}. \tag{60}$$

The overall output is a sum of these two signals, as illustrated in Fig. 32, which after rearranging of terms gives

$$\begin{aligned}\mathbf{y} = \mathbf{f}_L + \mathbf{f}_H = &\frac{1}{2}(G_L(\mathbf{L})H_L(\mathbf{L}) + G_H(\mathbf{L})H_H(\mathbf{L}))\mathbf{x}+\\ &\frac{1}{2}(G_L(\mathbf{L})\mathbf{J}_\mathcal{E}H_L(\mathbf{L}) + G_H(\mathbf{L})\mathbf{J}_\mathcal{H}H_H(\mathbf{L}))\mathbf{x}.\end{aligned} \tag{61}$$

The perfect reconstruction condition, $\mathbf{y} = \mathbf{x}$, is then achieved if

$$G_L(\mathbf{L})H_L(\mathbf{L}) + G_H(\mathbf{L})H_H(\mathbf{L}) = 2\mathbf{I}, \tag{62}$$

$$G_L(\mathbf{L})\mathbf{J}_\mathcal{E}H_L(\mathbf{L}) - G_H(\mathbf{L})\mathbf{J}_\mathcal{E}H_H(\mathbf{L}) = \mathbf{0}. \tag{63}$$

**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

$$\mathbf{F} = \mathbf{U}^T\mathbf{f} = \frac{1}{2}(\mathbf{U}^T\mathbf{x} + \mathbf{U}^T\mathbf{J}_\mathcal{E}\mathbf{x}) = \frac{1}{2}(\mathbf{X} + \mathbf{X}^{(alias)}), \tag{64}$$

where $\mathbf{X}^{(alias)} = \mathbf{U}^T\mathbf{J}_\mathcal{E}\mathbf{x}$ 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 (57), since

$$\begin{aligned}\mathbf{U}^T\mathbf{J}_\mathcal{E} &= \begin{bmatrix}\mathbf{u}_0\ \mathbf{u}_1\ \dots\ \mathbf{u}_{N-1}\end{bmatrix}^T\mathbf{J}_\mathcal{E}\\ &= \begin{bmatrix}\mathbf{u}_{0\mathcal{E}} & \mathbf{u}_{1\mathcal{E}} & & \mathbf{u}_{N-1\mathcal{E}}\\ -\mathbf{u}_{0\mathcal{H}} & -\mathbf{u}_{1\mathcal{H}} & \cdots & -\mathbf{u}_{N-1\mathcal{H}}\end{bmatrix}^T\\ &= \begin{bmatrix}\mathbf{u}_{N-1}\ \mathbf{u}_{N-2}\ \dots\ \mathbf{u}_0\end{bmatrix}^T = \mathbf{U}_{\mathrm{LR}}^T\end{aligned}$$

where

$$\mathbf{u}_k = \begin{bmatrix}\mathbf{u}_{k\mathcal{E}}\\ \mathbf{u}_{k\mathcal{H}}\end{bmatrix},\ \mathbf{u}_{N-1-k} = \begin{bmatrix}\mathbf{u}_{k\mathcal{E}}\\ -\mathbf{u}_{k\mathcal{H}}\end{bmatrix},\ k = 0, 1, \dots N-1,$$

and

$$\mathbf{U}_{\mathrm{LR}} = \begin{bmatrix}\mathbf{u}_{N-1}\ \mathbf{u}_{N-2}\ \dots\ \mathbf{u}_0\end{bmatrix}$$

is a left-right flipped version of the eigenvector matrix

$$\mathbf{U} = \begin{bmatrix}\mathbf{u}_0\ \mathbf{u}_1\ \dots\ \mathbf{u}_{N-1}\end{bmatrix}.$$

The element-wise form of equation (64) 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

$$F(\lambda_k) = \frac{1}{2}(X(\lambda_k) + X(2-\lambda_k)).$$

The second term in $F(\lambda_k)$ represents an aliasing component of the GDFT of the original signal.

The spectral representation of (62) 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, $\mathbf{U}^T H_L(\mathbf{L})\mathbf{U} = H_L(\mathbf{\Lambda})$, we get the spectral domain form of the reconstruction condition (62) as

$$G_L(\mathbf{\Lambda})H_L(\mathbf{\Lambda}) + G_H(\mathbf{\Lambda})H_H(\mathbf{\Lambda}) = 2\mathbf{I}. \tag{65}$$

For the aliasing part in equation (63), 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 (63) is then of the form

$$\begin{aligned}\mathbf{U}^T G_L(\mathbf{L})\mathbf{U}\mathbf{U}^T\mathbf{J}_\mathcal{E} H_L(\mathbf{L})\mathbf{U}_{\mathrm{LR}} &= \mathbf{U}^T G_L(\mathbf{L})\mathbf{U}\mathbf{U}_{\mathrm{LR}}^T H_L(\mathbf{L})\mathbf{U}_{\mathrm{LR}}\\ &= G_L(\mathbf{\Lambda})H_L^{(R)}(\mathbf{\Lambda}),\end{aligned} \tag{66}$$

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 $H_L(\lambda_{N-1-k}) = H_L(2 - \lambda_k)$ instead of $H_L(\lambda_k)$.

The same holds for the second term in (63) 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 (63) as

$$G_L(\boldsymbol{\Lambda})H_L(2\mathbf{I} - \boldsymbol{\Lambda}) - G_H(\boldsymbol{\Lambda})H_H(2\mathbf{I} - \boldsymbol{\Lambda}) = \mathbf{0}. \tag{67}$$

An element-wise solution to the system in (62)-(63), for bipartite graphs and the normalized graph Laplacian, according to (65) and (67), reduces to

$$G_L(\lambda_k)H_L(\lambda_k) + G_H(\lambda_k)H_H(\lambda_k) = 2, \tag{68}$$
$$G_L(\lambda_k)H_L(2 - \lambda_k) - G_H(\lambda_k)H_H(2 - \lambda_k) = 0. \tag{69}$$

*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

$$\begin{aligned} G_L(\lambda) &= H_L(\lambda), \\ H_H(\lambda) &= H_L(2 - \lambda), \\ G_H(\lambda) = H_H(\lambda) &= H_L(2 - \lambda). \end{aligned} \tag{70}$$

For this solution, the design equation is given by

$$H_L^2(\lambda) + H_L^2(2 - \lambda) = 2, \tag{71}$$

while the aliasing cancellation condition, (69), 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.

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, $H_L^2(\lambda_0) = 2 - \lambda_0 = 2$, since $\lambda_0 = 0$, and $H_L^2(\lambda_{\max}) = 2 - \lambda_{\max} = 0$, as $\lambda_{\max} = 2$. The vertex domain system operators which satisfy all four quadratic mirror analysis and synthesis filters in (70), are

$$H_L(\boldsymbol{\Lambda}) = \sqrt{2\mathbf{I} - \boldsymbol{\Lambda}}, \quad G_L(\boldsymbol{\Lambda}) = H_L(\boldsymbol{\Lambda}) = \sqrt{2\mathbf{I} - \boldsymbol{\Lambda}},$$
$$H_H(\boldsymbol{\Lambda}) = H_L(\mathbf{2I} - \boldsymbol{\Lambda}) = \sqrt{\boldsymbol{\Lambda}}, \quad G_H(\boldsymbol{\Lambda}) = H_H(\boldsymbol{\Lambda}) = \sqrt{\boldsymbol{\Lambda}}.$$

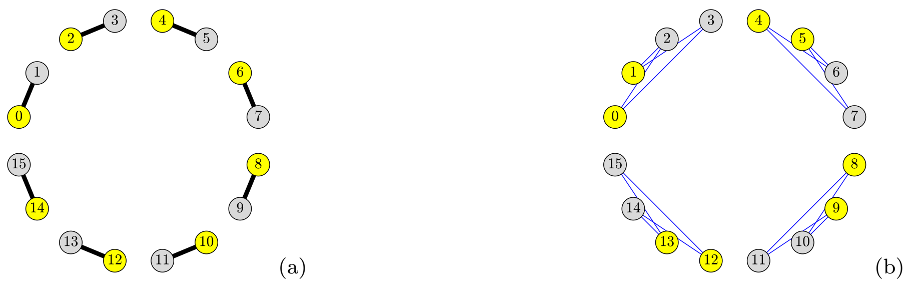


**Fig. 33** Bipartite graph for the Haar wavelet transform with $N = 16$ vertices. (a) Vertices in yellow are used for the low-pass part of the signal and correspond to the set $\mathcal{E}$, while the vertices in gray belong to the set $\mathcal{H}$. This is the highest two-vertex resolution level for the Haar wavelet. (b) Graph for a four-vertex resolution level in the Haar wavelet.

The spectral domain filtering form for the low-pass part of graph signal is then obtained from (58), as

$$\begin{aligned}\mathbf{F}_L = \mathbf{U}^T\mathbf{f}_L &= \frac{1}{2}\mathbf{U}^T G_L(\mathbf{L})(\mathbf{I}+\mathbf{J}_{\mathcal{E}})H_L(\mathbf{L})\mathbf{x}\\ &= \frac{1}{2}\mathbf{U}^T G_L(\mathbf{L})\mathbf{U}\mathbf{U}^T(\mathbf{I}+\mathbf{J}_{\mathcal{E}})H_L(\mathbf{L})\mathbf{U}_{\mathrm{LR}}\mathbf{U}_{\mathrm{LR}}^T\mathbf{U}\mathbf{X}\\ &= \frac{1}{2}G_L(\boldsymbol{\Lambda})H_L(\boldsymbol{\Lambda})\mathbf{X} + \frac{1}{2}G_L(\boldsymbol{\Lambda})H_L(2\mathbf{I}-\boldsymbol{\Lambda})\mathbf{X}_{\mathrm{UD}}\end{aligned}$$

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 GDFT vector, $\mathbf{X}$, with elements flipped upside-down.

The same holds for the high-pass part in (59), to yield

$$\begin{aligned}\mathbf{F}_H &= \frac{1}{2}\mathbf{U}^T G_H(\mathbf{L})(\mathbf{I}+\mathbf{J}_{\mathcal{H}})H_H(\mathbf{L})\mathbf{x}\\ &= \frac{1}{2}G_H(\boldsymbol{\Lambda})H_H(\boldsymbol{\Lambda})\mathbf{X} - \frac{1}{2}G_H(\boldsymbol{\Lambda})H_H(2\mathbf{I}-\boldsymbol{\Lambda})\mathbf{X}_{\mathrm{UD}}\end{aligned}$$

and

$$\mathbf{F}_L + \mathbf{F}_H = \mathbf{X}.$$

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 $\mathbf{2I} - \mathbf{L}$, which easily reduces to $|\mathbf{L}|$, for the normalized graph Laplacian used here.

Another simple transfer function that satisfies the design equation (71) 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 (70).

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 [47]. 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 $H_L(\mathbf{L}) = \sqrt{\mathbf{2I}-\mathbf{L}}$ is not sparse. This is the reason to use other forms which are sufficiently smooth toward the boundary points, along with their polynomial approximations, $H_L(\mathbf{\Lambda}) = c_0\mathbf{\Lambda} + c_1\mathbf{\Lambda}^2 + \cdots + c_{M-1}\mathbf{\Lambda}^{M-1}$, 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 $\lambda = \lambda_k$, $k = 0, 1, \ldots, N-1$.

## 5 Time-Varying Signals on Graphs

If we assume that the signal on graph changes in time at each vertex, then we have a signal $x_p(n)$ where $n$ indicates the vertex index and $p$ corresponds to the discrete-time index. If the signal sampling in time is uniform then the index $p$ corresponds to $p\Delta t$ time instant.

In general, this kind of data can be considered within the Cartesian product framework, as shown in Fig. 9. The resulting graph $\mathcal{G} = (\mathcal{V}, \mathcal{B})$ follows as a Cartesian product of the given graph $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{B}_1)$ and a simple line (or circular) graph $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{B}_2)$ that corresponds to the classical time-domain signal processing.

The adjacency matrix of a Cartesian product of two graphs is

$$\mathbf{A} = \mathbf{A}_1 \otimes \mathbf{I}_{N_2} + \mathbf{I}_{N_1} \otimes \mathbf{A}_2,$$

where $\mathbf{A}_1$ is the adjacency matrix of the given graph $\mathcal{G}_1$ and $\mathbf{A}_2$ is the adjacency matrix for the line or circle graph. The numbers of vertices in $\mathcal{G}_1$ and $\mathcal{G}_2$ are denoted by $N_1$ and $N_2$.

Next we will consider a simple and important example of a time-varying signal defined in an iterative way.

### 5.1 Diffusion on Graph and Low Pass Filtering

Consider the diffusion equation $\partial\mathbf{x}/\partial t = -\mathbf{L}\mathbf{x}$. Its discrete-time form obtained by using the backward difference approximation of the partial derivative is

$$\mathbf{x}_{p+1} - \mathbf{x}_p = -\alpha\mathbf{L}\mathbf{x}_{p+1}$$

or $\mathbf{x}_{p+1}(\mathbf{I} + \alpha\mathbf{L}) = \mathbf{x}_p$ producing

$$\mathbf{x}_{p+1} = (\mathbf{I} + \alpha\mathbf{L})^{-1}\mathbf{x}_p.$$

The forward difference approximation of the diffusion equation results in

$$\mathbf{x}_{p+1} - \mathbf{x}_p = -\alpha\mathbf{L}\mathbf{x}_p$$

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 quadratic form of graph signal minimization. The signal quadratic form on a graph is $E_x = \mathbf{x}\mathbf{L}\mathbf{x}^T$, (see Section 3.6). If we want to find its minimum we can use the steepest descent method. Then the signal value at an instant $p$ is changed in the opposite direction of the gradient, toward the energy minimum position. The gradient of quadratic form is $\partial E_x/\partial \mathbf{x}^T = 2\mathbf{x}\mathbf{L}$, resulting in the iterative procedure

$$\mathbf{x}_{p+1} = \mathbf{x}_p - \alpha\mathbf{L}\mathbf{x}_p = (\mathbf{I} - \alpha\mathbf{L})\mathbf{x}_p.$$

This relation can be used for simple and efficient filtering of graph signals (with the aim to minimize $E_x$ as the measure of the signal smoothness). If we assume that in one iteration the input graph signal is $\mathbf{x}_p$ and the output graph signal is $\mathbf{x}_{p+1}$, then the spectral domain relation of this system is

$$\mathbf{X}_{p+1} = (\mathbf{I} - \alpha\boldsymbol{\Lambda})\mathbf{X}_p$$

or

$$X_{p+1}(k) = (1 - \alpha\lambda_k)X_p(k).$$

Obviously, the low varying components pass through this system since $(1 - \alpha\lambda_k)$ is close to 1 for small $\lambda_k$, while the high varying components with larger $\lambda_k$ are attenuated. Iterative procedure will converge if $|1 - \alpha\lambda_{\max}| < 1$. In the stationary case, when

$$\lim_{p\to\infty} X_{p+1}(k) = \lim_{p\to\infty} (1 - \alpha\lambda_k)^{p+1} X_0(k)$$

all components $X_{p+1}(k)$ tend to 0 except the constant component $X_{p+1}(0)$, since $\lambda_0 = 0$. This component defines the stationary state (maximally smooth) solution. In order to avoid this effect, the iteration process can be used in alternation with

$$\mathbf{x}_{p+2} = (\mathbf{I} + \beta\mathbf{L})\mathbf{x}_{p+1}.$$

When these two iterative processes are used in a successive order the resulting system (Taubin's $\alpha - \beta$ algorithm) is

$$\mathbf{x}_{p+2} = (\mathbf{I} + \beta\mathbf{L})(\mathbf{I} - \alpha\mathbf{L})\mathbf{x}_p. \tag{72}$$

The resulting transfer function in the spectral domain in these two iteration steps is

$$H(\lambda_k) = (1 + \beta\lambda_k)(1 - \alpha\lambda_k).$$

After $K$ iterations the transfer function is

$$H_K(\lambda_k) = ((1 + \beta\lambda_k)(1 - \alpha\lambda_k))^K. \tag{73}$$

For some values of $\alpha < \beta$, this system can be a good and very simple approximation of a graph low-pass filter.

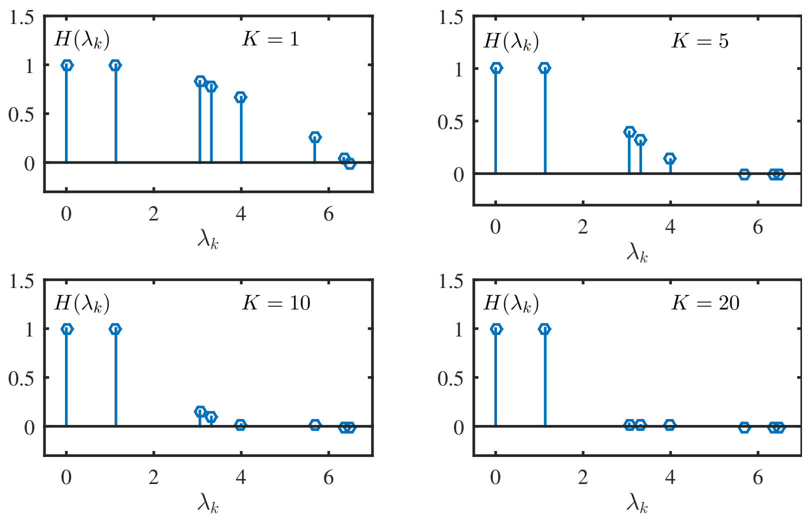


**Fig. 34** Filter approximation in the spectral domain for various number of iterations $K$. The Graph from Fig. 3 and its Laplacian are considered.

The Graph from Fig. 3 and its Laplacian are considered in this example. For the parameters $\alpha = 0.1545$, $\beta = 0.1875$, the spectral transfer function (73) is presented in Fig. 34 for the considered graph filter. The results for the following numbers of iterations $K = 1, 5, 10, 20$ are shown. We have filtered the noisy signal from Fig. 26(b). The initial noisy signal is denoted as $\mathbf{x}_0$. Then $\mathbf{x}_1 = (\mathbf{I} - 0.1545\mathbf{L})\mathbf{x}_0$ is calculated with the Laplacian defined by (6). Next $\mathbf{x}_2 = (\mathbf{I} + 0.1875\mathbf{L})\mathbf{x}_1$ is obtained. 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 almost the same as an output of the ideal low-pass filter presented in Fig. 26(c).

## 6 Random Graph Signals

In this section we will present basic definitions of the random signals on graphs [59–64]. The conditions for wide-sense stationarity (WSS) will be considered. The stationarity is related to the signal shift on a graph. We have presented two approaches to define the shift on a graph (using the adjacency matrix or Laplacian and their spectral decompositions). The main problem is that the shift on a graph does not preserve energy (isometry property) $\|\mathbf{A}\mathbf{x}\|_2^2 \neq \|\mathbf{x}\|_2^2$. In order to define an equivalent of the WSS on graphs (GWSS) other properties of the WSS signals in the classical time domain processing are used. This is the reason for providing a short review of the classic signal processing WSS definitions and properties first.

A real-valued random signal $x(n)$ is WSS if its mean value is time invariant, $\mu_x(n) = \mathrm{E}\{x(n)\} = \mu_x$ and its autocorrelation function is shift invariant $r_x(n, n-m) = \mathrm{E}\{x(n)x(n-m)\} = r_x(m)$.

A WSS random time-domain signal $x(n)$ can be considered as an output of a liner shift invariant system with impulse response $h(n)$ to a white noise input $\varepsilon(n)$ with $r_\varepsilon(n,m) = \delta(n-m)$.

In the time domain, the eigenvectors $\mathbf{u}_k$ of the shift operator $y(n) = x(n-1)$ are the DFT basis functions, $\mathbf{A} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^H$. For a random signal, its DFT $\mathbf{X} = \mathbf{U}^H\mathbf{x}$ is a random signal with the autocorrelation matrix $\mathbf{P}_x = \mathrm{E}\{\mathbf{X}\mathbf{X}^H\}$, where $\mathbf{U}^H$ is the DFT transformation matrix. For WSS signals, the matrix $\mathbf{P}_x$ is diagonal with the power spectral density (PSD) values

$$p_x(k) = \mathrm{DFT}\{r_x(n)\} = \mathrm{E}\{|X(k)|^2\}$$

on diagonal.

For WSS random signals $\mathbf{R}_x = \mathrm{E}\{\mathbf{x}\mathbf{x}^H\}$ is diagonalizable with the same transform matrix $\mathbf{U}$ as in $\mathbf{X} = \mathbf{U}^H\mathbf{x}$,

$$\mathbf{R}_x = \mathrm{E}\{\mathbf{x}\mathbf{x}^H\} = \mathrm{E}\{\mathbf{U}\mathbf{X}(\mathbf{U}\mathbf{X})^H\} = \mathbf{U}\mathrm{E}\{\mathbf{X}\mathbf{X}^H\}\mathbf{U}^H = \mathbf{U}\mathbf{P}_x\mathbf{U}^H \tag{74}$$

since $\mathbf{P}_x$ is a diagonal matrix for WSS signals.

These properties of the WSS signals in classical analysis will be used for the graph signals next.

## 6.1 Adjacency Matrix Based Definition

Consider a white noise signal $\boldsymbol{\varepsilon}$ on a graph with samples $\varepsilon(n)$. A signal $\mathbf{x}$ on the graph is graph wide sense stationary (GWSS) if it can be considered an output of a linear and shift invariant graph system $H(\mathbf{A}) = \sum_{m=0}^{M-1} h_m\mathbf{A}^m$ to the white noise input $\boldsymbol{\varepsilon}$,

$$\mathbf{x} = H(\mathbf{A})\boldsymbol{\varepsilon}.$$

The autocorrelation matrix $\mathbf{R}_x = \mathrm{E}\{\mathbf{x}\mathbf{x}^H\}$ of a GWSS signal is diagonalizable with the matrix of the adjacency matrix $\mathbf{A}$ eigenvectors

$$\mathbf{A} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^H \tag{75}$$

$$\mathrm{E}\{\mathbf{x}\mathbf{x}^H\} = \mathbf{U}\mathbf{P}_x\mathbf{U}^H, \tag{76}$$

where $\mathbf{P}_x$ is a diagonal matrix. The values on the diagonal of matrix $\mathbf{P}_x$ denoted by $\mathbf{p}_x$ represent the PSD of a graph signal $p_x(k) = \mathrm{E}\{|X(k)|^2\}$.

In order to prove this property for a signal $\mathbf{x} = H(\mathbf{A})\boldsymbol{\varepsilon}$, consider

$$\mathbf{R}_x = \mathrm{E}\{\mathbf{x}\mathbf{x}^H\} = \mathrm{E}\{H(\mathbf{A})\boldsymbol{\varepsilon}(H(\mathbf{A})\boldsymbol{\varepsilon})^H\} = H(\mathbf{A})H^H(\mathbf{A}).$$

Using $H(\mathbf{A}) = \mathbf{U}^T H(\boldsymbol{\Lambda})\mathbf{U}$ we get

$$\mathbf{R}_x = \mathbf{U}^T|H(\boldsymbol{\Lambda})|^2\mathbf{U},$$

what concludes the proof. The diagonal matrix is

$$\mathbf{P}_x = |H(\boldsymbol{\Lambda})|^2,$$

with the PSD of this signal

$$p_x(k) = |H(\lambda_k)|^2.$$

Periodogram on the graph can be estimated using $K$ realizations of the random signal denoted by $\mathbf{x}_i$. It is equal to the diagonal elements of matrix

$$\hat{\mathbf{P}}_x = \frac{1}{K}\sum_{i=1}^{K}\mathbf{X}_i\mathbf{X}_i^T = \mathbf{U}^T\frac{1}{K}\sum_{i=1}^{K}(\mathbf{x}_i\mathbf{x}_i^T)\mathbf{U}.$$

Consider a system on a graph with 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

$$p_y(k) = |H(\lambda_k)|^2 p_x(k).$$

*Wienner filter on a graph*

Consider a real-valued signal $\mathbf{s}$ as an input to a linear shift-invariant system on a graph, whose noisy output is

$$\mathbf{x} = \sum_{m=0}^{M-1} h_m \mathbf{A}^m \mathbf{s} + \boldsymbol{\varepsilon}.$$

In the spectral domain the system is described by

$$\mathbf{X} = H(\boldsymbol{\Lambda})\mathbf{S} + \mathbf{E}.$$

Assume that the signal and the noise are statistically independent. The noise is a zero-mean random signal. The aim is to find a filter $G(\boldsymbol{\Lambda})$ such that $\mathbf{Y} = G(\boldsymbol{\Lambda})\mathbf{X}$ estimates $\mathbf{S}$ in the mean squared sense. We will minimize

$$e^2 = \mathrm{E}\{\|\mathbf{S}-\mathbf{Y}\|_2^2\} = \mathrm{E}\{\|\mathbf{S}-G(\boldsymbol{\Lambda})\mathbf{X}\|_2^2\}.$$

The zero value of the derivative with respect to the elements of $G(\boldsymbol{\Lambda})$ produces

$$2\mathrm{E}\{(\mathbf{S}-G(\boldsymbol{\Lambda})\mathbf{X})\mathbf{X}^T\} = 0.$$

Since all matrices are diagonal we may use symbolic matrix division resulting in

$$G(\boldsymbol{\Lambda}) = \frac{\mathrm{E}\{\mathbf{S}\mathbf{X}^T\}}{\mathrm{E}\{\mathbf{X}\mathbf{X}^T\}} = \frac{\mathrm{E}\{\mathbf{S}(H(\boldsymbol{\Lambda})\mathbf{S}+\mathbf{E})^T\}}{\mathrm{E}\{(H(\boldsymbol{\Lambda})\mathbf{S}+\mathbf{E})(H(\boldsymbol{\Lambda})\mathbf{S}+\mathbf{E})^T\}} = \frac{H(\boldsymbol{\Lambda})\mathbf{P}_s}{H^2(\boldsymbol{\Lambda})\mathbf{P}_s+\mathbf{P}_\varepsilon}$$

or

$$G(\lambda_k) = \frac{H(\lambda_k)p_s(k)}{H^2(\lambda_k)p_s(k)+E(k)}.$$

In a non-noisy case, $E(k)=0$ for all $k$, the inverse filter follows, as expected.

## 6.2 Spectral Domain Shift Based Definition of GWSS

This approach is based on the shift on a graph defined using the graph filter response

$$\mathcal{T}_m\{h(n)\} = h_m(n) = \sum_{k=0}^{N-1} H(\lambda_k)u_k(m)u_k(n). \tag{77}$$

The matrix form of this relation is

$$\mathcal{T}_h = H(\mathbf{L}) = \sum_{m=0}^{M-1} h_m\mathbf{L}^m = \mathbf{U}H(\mathbf{\Lambda})\mathbf{U}^H, \tag{78}$$

where $\mathcal{T}_m\{h(n)\}$ are the elements of $\mathcal{T}_h$.

Note that the filter response function is well localized on a graph. If we use, for example, the $M-1$ neighborhood of the vertex $n$ in filtering defined by $H(\mathbf{\Lambda})$, then only the region within this neighborhood is used in the calculation. The localization operator acts in the spectral domain and associates the corresponding shift to the vertex domain.

The definition of the GWSS within this framework reads: The signal is GWSS if its autocorrelation function is invariant with respect to the shift $\mathcal{T}_m\{r_x(n)\}$

$$r_x(m) = \mathcal{T}_m\{r_x(n)\}.$$

For a GWSS signal the autocorrelation matrix $\mathbf{R}_x$ is diagonalizable with the matrix of eigenvectors of the Laplacian $\mathbf{L}$

$$\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T. \tag{79}$$

For the basic autocorrelation we can assume that

$$\mathbf{R}_x = \mathbf{U}P_x(\mathbf{\Lambda})\mathbf{U}^H$$

$$\mathcal{T}_m\{r_x(n)\} = \sum_{k=0}^{N-1} p_x(\lambda_k)u_k(m)u_k(n)$$

where

$$P_x(\mathbf{\Lambda}) = \mathbf{U}\mathbf{R}_x\mathbf{U}^H$$

is a diagonal matrix.

Finally, we will mention one more approach based on the shift operator defined as $\mathcal{T}_m = \exp(j\pi\sqrt{\mathbf{L}/\rho})$. It maps the eigenvalues of the Laplacian $\mathbf{L}$ on a unit circle, preserving the isometry.

## 7 Examples of Graphs Topologies

In the previous section, we have assumed that the graph is defined and the signal processing on the given graph topology is considered. However, the graph topology is very often a part of the processing problem rather than it being given within the problem definition. In this case, we may assume that the vertices are given, while the edges and their weights are part of the problem solution [65–82].

In general, we will consider three possible scenarios for the graph edges:

– The first group of problems is related to the geometry of the vertex positions. In the cases of various sensing setups (temperature, pressure, transportation,...) the locations of the sensing positions (vertices) are known and the vertex distances play a crucial role in data relations.
– The second group consists of problems where the relation among the sensing positions are physically well defined. Examples of such problems are electric circuit networks, linear heat transfer, social and computer networks, spring-mass systems, .... In these cases the edge weights are well defined based on the physics of the considered problem.
– In the third group we will include the problems where the data similarity plays a crucial role for the underlying graph topology. Various approaches are used to define the data similarity.

### 7.1 Geometric Graph Topologies

For a graph corresponding to a geometric network, the edge weights are related to the vertices distance. Consider vertices $n$ and $m$ whose positions in space are defined by position vectors $\mathbf{r}_n$ and $\mathbf{r}_m$. The Euclidean distance between these two vertices is

$$r_{nm} = \text{distance}(m,n) = \|\mathbf{r}_n - \mathbf{r}_m\|_2 .$$

A common way to define the graph weights in such networks is

$$W_{nm} = \begin{cases} e^{-r_{nm}^2/\tau^2} & \text{for } r_{nm} \le \kappa \\ 0 & \text{for } r_{nm} > \kappa, \end{cases} \tag{80}$$

where $r_{nm}$ is the Euclidian distance between the vertices $n$ and $m$, and $\tau$ and $\kappa$ are constants. The weights tend to 1 for close vertices. The weights are 0 or close to 0 for distant vertices.

The basic idea behind this definition of the edge weights is the assumption that the signal value measured at vertex $n$ is similar to signal values measured at the neighboring vertices. According to this definition, the processing of a signal at vertex $n$ should include close vertices with higher weights (close to 1) while the signal values sensed at farther vertices would be less relevant. They are included with smaller weighting coefficients or not included at all.

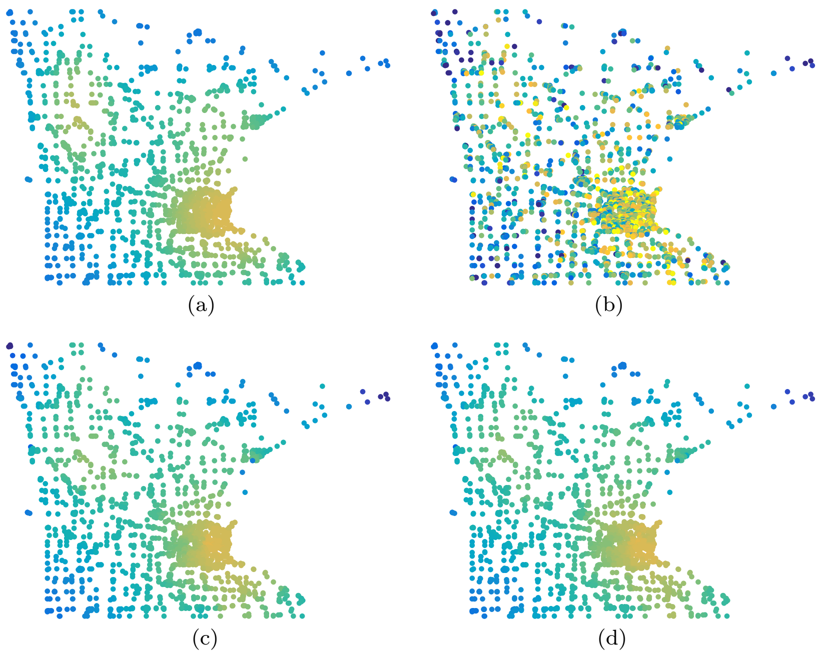


**Fig. 35** Minnesota roadmap graph: (a) simulated signal, (b) noisy signal, (c) low-pass filtering, and (d) iterative filtering example.

While the Gaussian function, used in (80), is the most appropriate in many applications, other forms to penalize signal values at the vertices far from the considered vertex can be used. Examples of such functions are

$$W_{nm} = \begin{cases} e^{-r_{nm}/\tau} & \text{for } r_{nm} \le \kappa \\ 0 & \text{for } r_{nm} > \kappa \end{cases} \tag{81}$$

or

$$W_{nm} = \begin{cases} 1/r_{nm} & \text{for } r_{nm} \le \kappa \\ 0 & \text{for } r_{nm} > \kappa. \end{cases} \tag{82}$$

The simplest form of the weighting coefficients would be

$$W_{nm} = \begin{cases} 1 & \text{for } r_{nm} \le \kappa \\ 0 & \text{for } r_{nm} > \kappa. \end{cases} \tag{83}$$

This form would correspond to an unweighted graph with $\mathbf{W} = \mathbf{A}$.

As an example, consider the Minnesota roadmap graph. The edges of the given adjacency matrix are weighted according to distances using (81) with $\tau = 25$km and $\kappa$ is not used since the connectivity is already determined by the given adjacency matrix.

A simulated signal and its noisy version are given in Fig. 35 (a) and (b). The noisy signal is filtered by low-pass filter in the Laplacian spectral domain using the 50 lowest changing spectral components. Filtering is also done using the two-step iterative procedure (72) with 200 iterations using $\alpha = 0.1$ and $\beta = 0.15$. Filtered signals are presented in Fig. 35 (c) and (d). The input SNR is 8.7dB and the output SNR of 23.0dB and 23.3dB is achieved.

A more general form of smoothing would be obtained using weights with appropriate low-pass filter coefficients in the adjacency or Laplacian spectrum, like for example the one presented by (15).

In classical signal analysis, using

$$\text{distance}(m,n) = r_{nm} = \|n-m\|_2 = |n-m|,$$

and $W_{nm} = e^{-r_{nm}^2/\tau^2}$ for $r_{nm} \le \kappa$ and $W_{nm} = 0$ for $r_{nm} > \kappa$, the value

$$\mathbf{y} = \mathbf{A}^0\mathbf{x} + \mathbf{A}^1\mathbf{x}$$

produces a Gaussian weighted mean, localized around the vertex (time instant) $n$ (moving average). For example, for $\tau = 4$ and $\kappa = 8$ it would produce the time domain wighted moving average

$$y(n) = x(n) + \sum_m W_{nm}x(n-m) = \sum_{m=-8}^{8} e^{-\frac{(n-(n-m))^2}{16}} x(n-m).$$

Unweighted moving average within $-\kappa \le m \le \kappa$ would be obtained with a large $\tau$. If the Euclidean distance between pixels is used in an image, we would get a moving average filtered image with a radial Gaussian window.

## 7.2 Topology Based on the Signal Similarity

In the previous section the graph weights are defined assuming that the geometric distance of vertices, where the signal is sensed, is a good and reliable indicator of the data similarity. That may be the case in some applications like, for example, the measurements of atmospheric temperature and pressure when the terrain configuration has no influence to the similarity of measured data. However, in general the geometric distance of vertices may not be a good indicator of data similarity.

In some applications like, for example, image processing, the signal value by themselves can be used as an indicator of the signal similarity, often in combination with the pixel/vertex distances. If the image intensity values at pixels indexed by $n$ and $m$ are denoted by $x(n)$ and $x(m)$ then the difference of intensities is defined by

$$\text{Intensity distance}(m,n) = s_{nm} = |x(n) - x(m)|.$$

Then the weights can be defined as

$$W_{nm} = \begin{cases} e^{-(x(n)-x(m))^2/\tau^2} & \text{for } r_{nm} \le \kappa \\ 0 & \text{for } r_{nm} > \kappa \end{cases}$$

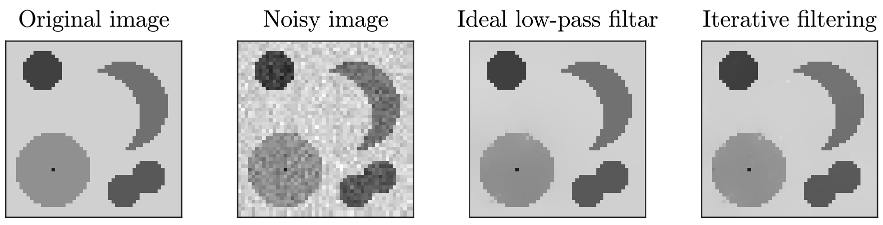


**Fig. 36** Original and noisy images (left) and filtered images (right) using frequency domain low-pass filter and iterative vertex domain filtering. The image is 8bit grayscale. Edge weights are calculated with $\kappa = \sqrt{2}$ and $\tau = 20$. Low-pass filter in frequency domain is done using an ideal low-pass filter with 10 lowest spectral components. Iterative filtering (72) is performed with 200 iterations using $\alpha = 0.1$ and $\beta = 0.15$.

where $r_{nm}$ is a geometric distance of the considered pixels/vertices.

An example with these kinds of weights applied to simple image graph filtering is presented in Fig. 36.

In some applications we are able to collect more than one data set for a given set of sensing points/vertices. In that case a more reliable measure of data similarity can be defined. Assume that at each vertex $n = 0, 1, \dots, N-1$ we have acquired $P$ signal values denoted by $x_p(n)$. This data set may be a set of multivariate data or signal measurements in a sequence. Then a good similarity measure function for real-valued signal at vertices $n$ and $m$ is

$$s^2_{nm} = \frac{\sum_{p=1}^{P} \big(x_p(n) - x_p(m)\big)^2}{\sqrt{\sum_{p=1}^{P} x_p^2(n) \sum_{p=1}^{P} x_p^2(m)}}.$$

The graph weights can again be defined using any of the previous penalty functions, for example,

$$W_{nm} = \begin{cases} e^{-s^2_{nm}/\tau^2} & \text{for } r_{nm} \le \kappa \\ 0 & \text{for } r_{nm} > \kappa. \end{cases}$$

The function of the form

$$W_{nm} = \begin{cases} e^{-s_{nm}/\tau} & \text{for } r_{nm} \le \kappa \\ 0 & \text{for } r_{nm} > \kappa. \end{cases}$$

is also used quite often.

As an example assume that $x_p(n)$ in $P$ observations, $p = 1, 2, \dots, P$ at $N$ vertices $n = 0, 1, \dots, N-1$ are zero-mean random noises with equal variances $\sigma_x^2 = 1$. Then

$$s^2_{n,m} = \frac{\sum_{p=1}^{P} \big(x_p(n) - x_p(m)\big)^2}{\sqrt{\sum_{p=1}^{P} x_p^2(n) \sum_{p=1}^{P} x_p^2(m)}} = 2(1 - R_x(n,m))$$

where

$$R_x(n,m) = \frac{1}{P}\sum_{p=1}^{P} x_p(n)x_p(m)$$

is the normalized autocorrelation function.

The same structure can be used for image classification, handwriting recognition or in establishing block similarity in large images. In these cases the distance between image/block $n$ and image/block $m$ is equal to

$$s_{nm} = \text{Image/Block distance}(m,n) = \|\mathbf{x}_n - \mathbf{x}_m\|_F,$$

where $\|\mathbf{x}\|_F$ is the Frobenius norm of an image or block matrix $\mathbf{x}$ (that is, the square root of the sum of all its squared elements).

## 7.3 Correlation Based Graph Laplacian learning

Consider a graph signal with $P$ independent observations. Denote the observed signal at vertex $n$ and observation $p$ as $x_p(n)$. The column vector with graph signal samples from the $p$-th observation is denoted by $\mathbf{x}_p$. All observations of this graph signal can be arranged into an $N \times P$ matrix

$$\mathbf{X}_P = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_P \end{bmatrix}.$$

Denote the $n$-th row of this matrix as a row vector $\mathbf{y}_n$

$$\mathbf{y}_n = \begin{bmatrix} x_1(n-1) & x_2(n-1) & \dots & x_P(n-1) \end{bmatrix}$$

Then the matrix of observations can also be written as

$$\mathbf{X}_P = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_N \end{bmatrix}.$$

The correlation coefficients estimated by averaging over the observations are

$$R_x(n,m) = \frac{1}{P}\sum_{p=1}^{P} x_p(n)x_p(m) = \frac{1}{P}\mathbf{y}_n\mathbf{y}_m^T$$

or in a matrix form

$$\mathbf{R}_x = \frac{1}{P}\mathbf{X}_P\mathbf{X}_P^T.$$

Since the correlation includes signal from all vertices, this matrix accumulates all correlations obtained through all possible walks from the current vertex to any other vertex. It means that the correlation coefficient for two vertices will produce misleading results if there exit one or more other vertices where the signal is strongly related to both of the considered vertices. That is the reason why the correlation overestimates the direct connections. It is not a good parameter for establishing direct links (edges) between vertices. Additional conditions should be imposed on the correlation matrix or other statistical parameters should be be used for edge weights estimation. Next we will present a method for the connectivity (edge weights) estimation based on the correlations, imposing the sparsity constraint on the weight coefficients.

Consider the vertex 0 with $n = 1$, eq. (7.3). We can estimate the edge weights from this vertex to all other vertices $\beta_{1m}$, $m = 2, 3, \dots, N$ by minimizing

$$J_1 = \|\mathbf{y}_1 - \sum_{m=2}^{N} \beta_{1m}\mathbf{y}_m\|_2^2 + \lambda \sum_{m=2}^{N} |\beta_{1m}|$$

The first term promotes the correlation between the observations $\mathbf{y}_1$ at the considered vertex with $n = 1$ and the observations $\mathbf{y}_m$ at all other vertices, $m = 2, 3, \dots, N$. The second term promotes sparsity in coefficients $\beta_{1m}$ (number of nonzero coefficients $\beta_{1m}$). Parameter $\lambda$ balances these two conditions. Full matrix form of the previous cost function is

$$J_1 = \|\mathbf{y}_1^T - \mathbf{Y}_1^T\boldsymbol{\beta}_1\|_2^2 + \lambda\|\boldsymbol{\beta}_1\|_1$$

where $\mathbf{Y}_1$ is obtained from the matrix $\mathbf{X}_P$ with the first row being removed and

$$\boldsymbol{\beta}_1 = \begin{bmatrix} \beta_{12} & \beta_{13} & \dots & \beta_{1N} \end{bmatrix}^T.$$

This problem can be solved using commonly defined LASSO minimization as $\boldsymbol{\beta}_1 = \text{lasso}(\mathbf{Y}_1^T, \mathbf{y}_1^T, \lambda)$, see the Appendix.

The minimization is repeated for all vertices with $n = 1, 2, \dots, N$

$$J_n = \|\mathbf{y}_n^T - \mathbf{Y}_n^T\boldsymbol{\beta}_n\|_2^2 + \lambda\|\boldsymbol{\beta}_n\|_1.$$

Since this procedure does not guarantee symmetry $\beta_{nm} = \beta_{mn}$ the edge weights could be calculated as $W_{nm} = \sqrt{\beta_{nm}\beta_{mn}}$.

As an example, consider the graph from Fig. 3 and $P = 10000$ observations. Observations are simulated by assuming white Gaussian external sources with zero-mean and variance 1 located at two randomly chosen vertices (see Appendix 8.1 and Fig. 47). An $N \times P$ matrix of signal $\mathbf{X}_P$ is formed. Using its rows the vector $\mathbf{y}_n$ and matrix $\mathbf{Y}_n$ are obtained. The matrix of coefficients $\beta_{nm}$ follows from $\text{lasso}(\mathbf{Y}_n^T, \mathbf{y}_n^T, \lambda)$ with $n = 1, 2, ..., 8$ and $\lambda = 1.7$ as

$$\boldsymbol{\beta} = \begin{bmatrix}
0 & 0.24 & 0 & 0 & 0 & 0 & 0 & 0.36 \\
0.55 & 0 & 0.86 & 0.22 & 0 & 0 & 0 & 0.18 \\
0 & 0.31 & 0 & 0.13 & 0 & 0 & 0 & 0 \\
0 & 0.19 & 0 & 0 & 0.28 & 0.18 & 0 & 0 \\
0 & 0.01 & 0 & 0.39 & 0 & 0.42 & 0.43 & 0.36 \\
0 & 0 & 0 & 0.18 & 0.19 & 0 & 0.44 & 0 \\
0 & 0 & 0 & 0 & 0.23 & 0.30 & 0 & 0 \\
0.32 & 0.16 & 0 & 0 & 0.21 & 0 & 0 & 0
\end{bmatrix}.$$

The groundtruth and estimated weights are presented in Fig. 37.

The same experiment is repeated for the unweighted graph from Fig. 2(a). The result is presented in Fig. 38. In this case the obtained values of $\boldsymbol{\beta}$ are used to decide weather $A_{mn} = 1$ or $A_{mn} = 0$.

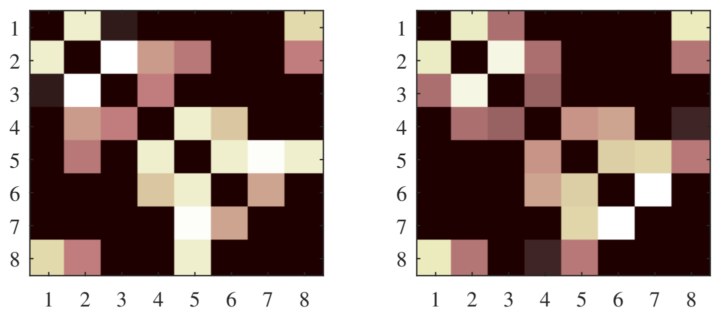


**Fig. 37** Groundtruth (left) and estimated weighting matrix (right). The axis index $n$ corresponds to the vertex $n-1$.

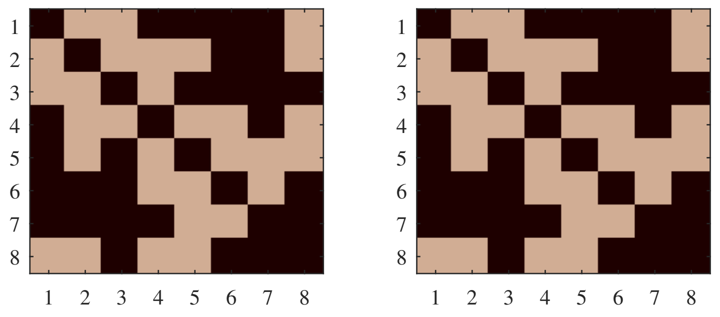


**Fig. 38** Groundtruth (left) and estimated adjacency matrix (right) for unweighted graph.

## 7.4 Graph Laplacian Learning with the Smoothness Constraint

Consider a set of noisy signal values $x_p(n)$ in $P$ observations, $p = 1, 2, \dots, P$ on $N$ vertices $n = 0, 1 \dots, N-1$ of an undirected graph. The goal is to get the graph connectivity (its Laplacian). To this aim we will calculate a signal $y_p(n)$ that is close to the observations $x_p(n)$ under the condition that $y_p(n)$ is also as smooth as possible on a graph. This formulation is similar to the one described and explained in Section 3.12. The signal $y_p(n)$ can be found by minimizing the cost function

$$J_p = \frac{1}{2}\|\mathbf{y}_p - \mathbf{x}_p\|_2^2 + \alpha \mathbf{y}_p^T \mathbf{L} \mathbf{y}_p, \text{ for } p = 1, 2, \dots, P.$$

The difference here is that the Laplacian (graph edges and their weights) is unknown as well. It has to be determined along with the output signal $\mathbf{y}_p$.

Since we have $P$ observations we can form $N \times P$ matrices

$$\mathbf{X}_P = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_P \end{bmatrix} \text{ and } \mathbf{Y}_P = \begin{bmatrix} \mathbf{y}_1 & \mathbf{y}_2 & \dots & \mathbf{y}_P \end{bmatrix}.$$

The cost function for the whole set of observations is

$$J = \sum_{p=1}^{P} J_p = \frac{1}{2}\|\mathbf{Y}_P - \mathbf{X}_P\|_F^2 + \alpha \mathrm{Trace}\{\mathbf{Y}_P^T \mathbf{L} \mathbf{Y}_P\} + \beta\|\mathbf{L}\|_F^2,$$

where $\text{Trace}\{\mathbf{Y}_P^T\mathbf{L}\mathbf{Y}_P\}$ is the matrix form of the term $\mathbf{y}_p^T\mathbf{L}\mathbf{y}_p$ and the constraint about the energy of Laplacian $\|\mathbf{L}\|_F^2 = \sum_n\sum_m L_{mn}^2$ is added in order to keep its values as low as possible.

It has been assumed that the the Laplacian is normalized. In order to avoid trivial solutions, the condition

$$\text{Trace}\{\mathbf{L}\} = N$$

is used as well, along with

$$L_{mn} = L_{nm} \le 0 \text{ for } n \ne m, \text{ and } \sum_{m=0}^{N-1} L_{nm} = 0.$$

The problem is jointly convex with respect to the signal and Laplacian. It is solved in an iterative two-step procedure:

(1) Assume that

$$\mathbf{Y}_P = \mathbf{X}_P.$$

(2) Estimate Laplacian $\mathbf{L}$ by minimizing

$$J_1 = \alpha\text{Trace}\{\mathbf{Y}_P^T\mathbf{L}\mathbf{Y}_P\} + \|\mathbf{L}\|_F^2$$

with given conditions for the Laplacian form.

(3) For the obtained Laplacian in Step (2), the signal $\mathbf{Y}_P$ is calculated minimizing

$$J_2 = \frac{1}{2}\|\mathbf{Y}_P - \mathbf{X}_P\|_F^2 + \alpha\text{Trace}\{\mathbf{Y}_P^T\mathbf{L}\mathbf{Y}_P\}.$$

Steps (2) and (3) are iteratively repeated. Step (3) has a closed form solution as presented in Section 3.12.

## 7.5 Generalized Laplacian Learning

The generalized Laplacian $\mathbf{Q}$ is defined as

$$\mathbf{Q} = \alpha\mathbf{I} - \mathbf{N},$$

where $\mathbf{N}$ is a nonnegative symmetric matrix and $\mathbf{Q}$ is a symetric positive semidefinite matrix. Any generalized Laplacian can be written as a sum of a standard Laplacian $\mathbf{L}$ and a diagonal matrix $\mathbf{P}$

$$\mathbf{Q} = \mathbf{L} + \mathbf{P}.$$

The generalized Laplacian allows self-loops on the vertices. They are defined by $\mathbf{P}$.

Consider a set of noisy signals $x_p(n)$ and their $P$ observations, $p = 1, 2, \dots, P$ on $N$ vertices $n = 0, 1 \dots, N-1$ of an undirected graph. The main goal is again to get the graph connectivity (its Laplacian) from the condition that the observed signal is as smooth as possible on the graph defined by a generalized

Laplacian $\mathbf{Q}$. The cost function to achieve this goal is defined by the signal smoothness function

$$J_p = \mathbf{x}_p^T \mathbf{Q} \mathbf{x}_p, \text{ for } p = 1, 2, \dots, P.$$

The correlation matrix of the considered observations is $\mathbf{R}_x = \mathrm{E}\{\mathbf{x}\mathbf{x}^T\} \approx \frac{1}{P}\sum_{p=1}^{P} \mathbf{x}_p \mathbf{x}_p^T$. The smoothness promoting cost function for all data is

$$J = \sum_{p=1}^{P} \mathbf{x}_p^T \mathbf{Q} \mathbf{x}_p = \mathrm{Trace}\{\mathbf{X}_P \mathbf{Q} \mathbf{X}_P^T\} = \mathrm{Trace}\{\mathbf{R}_x \mathbf{Q}\}.$$

Next the conditions for the generalized Laplacian should be added (otherwise a trivial solution would be obtained). For symmetric positive definite matrices, all eigenvalues must be positive. Since the product of the eigenvalues is equal to $\det(\mathbf{Q})$, this condition is included by adding the term $\log(\det(\mathbf{Q}))$ to the cost function. Then the cost function is of the form

$$J = \log(\det(\mathbf{Q})) + \mathrm{Trace}\{\mathbf{R}_x \mathbf{Q}\}.$$

The interpretation of this cost function within the Gaussian random signal and maximum likelihood estimate is given in Section 7.9. This cost function minimizes the logarithm of the joint probability density function of a graph signal $\mathbf{x}_p$ under the Gaussianity assumption,

$$P(\mathbf{x}_p) \sim \det(\mathbf{Q}) \exp\left(-\frac{1}{2}\mathbf{x}_p \mathbf{Q} \mathbf{x}_p\right).$$

Minimization of the cost function $J$ with respect to $\mathbf{Q}$, with $\partial J/\partial \mathbf{Q} = 0$, produces

$$\mathbf{Q} = \mathbf{R}_x^{-1}.$$

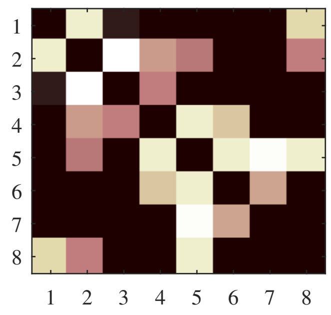
Here we have used the relation between the trace of a positive semidefinite matrix and the trace of its eigenvalue matrix

$$\log(\det(\mathbf{Q})) = \sum_{i=k}^{N} \log(\lambda_k) = \mathrm{Trace}(\log(\mathbf{\Lambda})) = \mathrm{Trace}(\log(\mathbf{Q})).$$

The solution of the previous equation $\mathbf{Q} = \mathbf{R}_x^{-1}$, can be used as the generalized Laplacian estimate and the graph is obtained.

The weighting matrix corresponding to the inverse correlation matrix $\mathbf{R}_x$, with positive and small off diagonal values set to zero is shown in Fig. 39 (right). Here we consider the graph from Fig. 3 and $P = 10000$ observations. The observations are simulated by assuming white Gaussian external sources with zero-mean and variance 1 located at two randomly chosen vertices (as described in more details in Section 7.3).

The correlation function matrix $\mathbf{R}_x$ may be singular. It is always singular when $N > P$. Also, this form will not produce a matrix satisfying the conditions to be a generalized Laplacian. The inverse correlation function may have



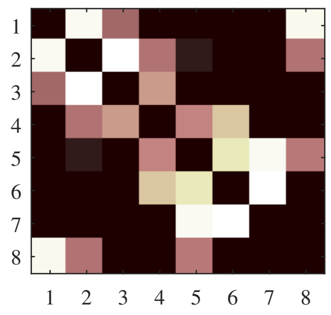


**Fig. 39** Groundtruth (left) and weighting matrix estimated using inverse correlation (right). The axis index $n$ corresponds to the vertex $n-1$.

positive off-diagonal values. Therefore the previous cost function should have additional constraints. Here we will present two of them.

In the first approach, a term corresponding to the Lagrange multipliers $\mathbf{B}$ is added so that these values do not change the diagonal elements of $\mathbf{Q}$ and provide that all

$$Q_{mn} = Q_{nm} \leq 0$$

for $n \neq m$, with $B_{nm} = B_{mn} \geq 0$. The diagonal elements of matrix $\mathbf{B}$ are $B_{nn} = 0$. Finally, $B_{nm}Q_{nm} = 0$ for all $n$ and $m$. In this case the minimization solution for the generalized Laplacian is obtained as

$$\mathbf{Q} = (\mathbf{R}_x + \mathbf{B})^{-1}$$

using the cost function

$$J = \log(\det(\mathbf{Q})) + \text{Trace}\{\mathbf{R}_x\mathbf{Q}\} + \text{Trace}\{\mathbf{B}\mathbf{Q}\}.$$

Another possible approach corresponds to the classical reconstruction formulation of a sparse signal. In this case the sparsity constraint on the generalized Laplacian is added. The cost function is defined as

$$J = \log(\det(\mathbf{Q})) + \text{Trace}\{\mathbf{R}_x\mathbf{Q}\} + \beta\|\mathbf{Q}\|_1.$$

This minimization problem can be solved using various methods. One of them is the graphical LASSO algorithm, an extension of the standard LASSO algorithm to the graph problems (see Appendix). For the same signal as in the previous examples, the weighting matrix obtained using the graphical LASSO, with positive and small values set to zero, is shown in Fig. 40 (right).

## 7.6 Graph Topology with some a Priori Knowledge

Consider a random signal $x(n)$ and its $P$ realizations on a graph. Assume that a random graph signal is a result of white noise signals $\boldsymbol{\varepsilon}$ as external sources

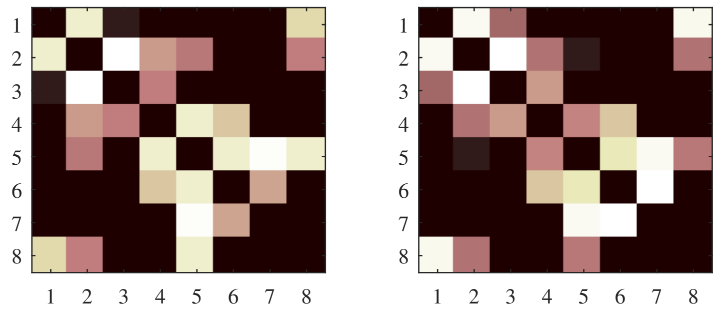


**Fig. 40** Groundtruth (left) and weighting matrix estimated using graphical LASSO (right). The axis index $n$ corresponds to the vertex $n-1$.

in each vertex (in the sense as presented in Fig. 42). Then the external source to the vertex signal relation is

$$\mathbf{L}\mathbf{x} = \boldsymbol{\varepsilon}$$

or $\mathrm{E}\{\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^T\} = \mathbf{L}\,\mathrm{E}\{\mathbf{x}\mathbf{x}^T\}\mathbf{L}^T$ resulting in $\mathbf{I} = \mathbf{L}\mathbf{R}_x\mathbf{L}^T$. The solution is

$$\mathbf{L}^2 = \mathbf{R}_x^{-1} \text{ or } \mathbf{L} = \mathbf{R}_x^{-1/2}.$$

Assume now that the graph signal is considered with a reference vertex. For the notation simplicity assume, without loss of generality, that the reference vertex is denoted by $N-1$ (the last vertex). Assume also that the value of the graph signal at this reference vertex is $x_p(N-1) = 0$ for any $p$. If this is not the case then we can achieve this by subtracting $x_p(N-1)$ from each $x_p(n)$. This operation will not change the properties of the graph signal. In this case the correlation matrix is singular and assumes the form

$$\mathbf{R}_x = \begin{bmatrix} \mathbf{R}_{N-1} & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix},$$

where $\mathbf{R}_{N-1}$ is the correlation of the vertices $n = 0, 1, \dots, N-2$. Assuming nonsingular $\mathbf{R}_{N-1}$ the Laplacian can be calculated as

$$\mathbf{L} = \mathbf{R}_x^{-1/2} = \begin{bmatrix} \sqrt{\mathbf{R}_{N-1}^{-1}} & \mathbf{a} \\ \mathbf{a}^T & -\sum_{m=0}^{N-2} a_m \end{bmatrix},$$

where the column vector $\mathbf{a} = [a_0\ a_1\ \dots\ a_{N-2}]^T$ elements are added so that

$$\sum_{m=0}^{N-2} L_{nm} + a_n = 0$$

holds for each row and column. The matrix square root operation is used.

For the graph from Fig. 7 and random graph signal generated using random white external sources the Laplacian is obtained from

$$\mathbf{R}_x = \begin{bmatrix} 2.92 & 2.91 & 3.53 & 2.98 & 2.50 & 3.08 & 3.05 & 0 \\ 2.91 & 3.78 & 4.61 & 4.14 & 3.55 & 4.43 & 4.42 & 0 \\ 3.53 & 4.61 & 6.27 & 5.13 & 4.27 & 5.33 & 5.26 & 0 \\ 2.98 & 4.14 & 5.13 & 5.43 & 4.71 & 6.05 & 6.05 & 0 \\ 2.50 & 3.55 & 4.27 & 4.71 & 4.52 & 5.73 & 6.01 & 0 \\ 3.08 & 4.43 & 5.33 & 6.05 & 5.73 & 7.76 & 7.77 & 0 \\ 3.05 & 4.42 & 5.26 & 6.05 & 6.01 & 7.77 & 8.81 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

This approach can be generalized to any known

$$H(\mathbf{L})\mathbf{x} = \boldsymbol{\varepsilon}$$

or $H(\mathbf{L})\boldsymbol{\varepsilon} = \mathbf{x}$. In this case a matrix polynomial equation has to be solved for the Laplacian.

## 7.7 Graph Topology Based on the Eigenvectors

Assume that the available observations of a graph signal $x_p(n)$ are graph wide sense stationary (GWSS). A graph signal's observations $\mathbf{x}_p$ are GWSS if they can be considered as the outputs of a linear and shift invariant system $H(\mathbf{A})$ to the white noise inputs $\boldsymbol{\varepsilon}_p$, with $\mathbf{x} = H(\mathbf{A})\boldsymbol{\varepsilon}$, see Section 6.1. The correlation matrix of the observed signal can be written as

$$\mathbf{R}_x = \mathbf{U}^T|H(\boldsymbol{\Lambda})|^2\mathbf{U},$$

where $\mathbf{U}$ is the matrix of graph eigenvectors $\mathbf{L} = \mathbf{U}^T\boldsymbol{\Lambda}\mathbf{U}$. The autocorrelation matrix estimation is done using all available observations $\mathbf{R}_x = \mathrm{E}\{\mathbf{x}\mathbf{x}^T\} \approx \frac{1}{P}\sum_{p=1}^{P}\mathbf{x}_p\mathbf{x}_p^T$. From the autocorrelation matrix decomposition we can learn about the graph eigenvectors. The same holds for the precision matrix $\boldsymbol{\Theta} = \mathbf{R}_x^{-1}$ since the inverse matrix has the same eigenvectors as the original matrix.

In order to estimate the graph connectivity (estimate its Laplacian or adjcency matrix) we can use the autocorrelation matrix eigenvectors. Since we do not know $H(\boldsymbol{\Lambda})$, it will be assumed that the graph is defined by the eigenvalues that produces the smallest number of edges. This can be done by minimizing the number of nonzero values in $\mathbf{L}$ with the given eigenvectors.

The minimization problem is

$$\min_{\lambda_k} \|\mathbf{L}\|_0 \text{ subject to } \mathbf{L} = \sum_{k=0}^{N-1} \lambda_k \mathbf{u}_k \mathbf{u}_k^T.$$

The convex form of this minimization problem is

$$\min_{\lambda_k} \|\mathbf{L}\|_1 \text{ subject to } \mathbf{L} = \sum_{k=0}^{N-1} \lambda_k \mathbf{u}_k \mathbf{u}_k^T.$$

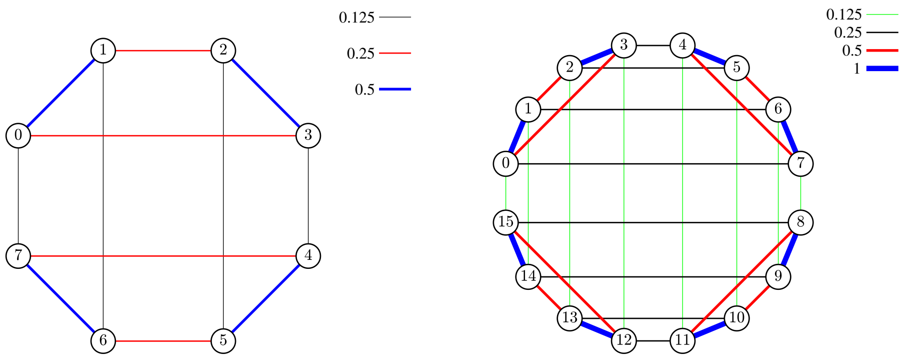


**Fig. 41** Graph whose eigenvectors are the Hadamard transform baisis functions for $N = 8$ and $N = 16$.

The convex norm-one based form can produce the same solution as the original norm-zero form if the Laplacian sparsity is low and satisfies some conditions (in the sense discussed within Section 4.2).

Since the eigenvectors are obtained from the correlation matrix decomposition, the spectral analysis obtained in this way is related to the principal value decomposition (PVD), where the signal is decomposed onto the set of correlation matrix eigenvectors.

For the examples with classical signal processing, we have used the Fourier analysis. The problem formulation presented in this section can be used to define a graph such that the spectral analysis on this graph leads to some other well known transforms. We will illustrate this method on the Hadamard transform with $N = 8$ with eigenvectors

$$\mathbf{U} = \frac{1}{\sqrt{8}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}.$$

If the eigenvalues are found to minimize the number of nonzero elements in the Laplacian, we get the graphs for $N = 8$ and $N = 16$ as shown in Fig. 41.

## 7.8 Physically Well Defined Graphs

### *7.8.1 Resistive Electrical Circuits*

One of the oldest applications of graph theory in engineering was in electrical circuits theory. Graph theory based methods for analysis and transformations

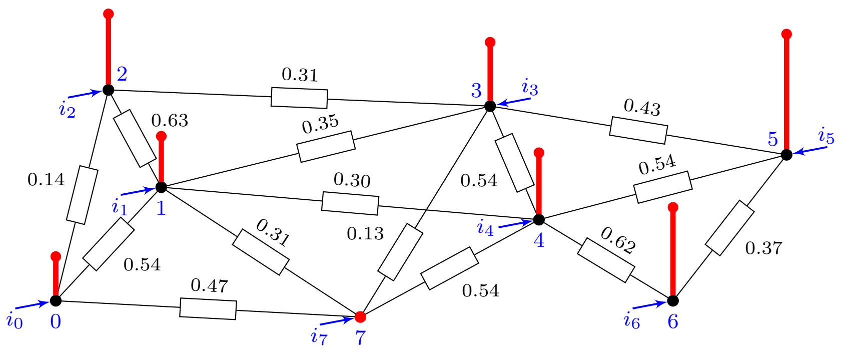


**Fig. 42** Electric potential $x(n)$ as a signal on an electric circuit graph

of electrical circuits are already part of the classical electrical circuits courses and textbooks. This approach can directly be applied to other engineering fields, like heat transfer or mechanical mass strings. It is interesting that some general information theory problems can be interpreted and solved within the graph approach to the basic electrical circuits framework. In these cases the underlying graph topology is well defined and is a part of the problem statement.

The Laplacian can also be considered within the basic electric circuit theory. Since it can be derived based on the Kirchhoff's laws, the Laplacian is also known as the Kirchhoff matrix in electric circuit theory.

Consider a resistive electric circuit. Denote the electric potential in the circuit vertices (nodes) by $x(n)$. The vertices in an electrical circuit are connected with edges. The weight of an edge connecting the vertices $n$ and $m$ is defined by the edge conductance $W_{nm}$. The conductances are the reciprocal values to the edge resistances $W_{nm} = 1/R_{nm}$. The current in the edge from vertex $n$ to vertex $m$ is equal to

$$i_{nm} = \frac{x(n) - x(m)}{R_{nm}} = W_{nm}(x(n) - x(m)).$$

In addition to the edge currents, an external current generator is attached to each vertex. It can be considered as a source of the signal change in the vertices. The external current at a vertex $n$ is denoted by $i_n$.

The sum of all currents going from a a vertex $n$, $n = 0, 1, \dots, N-1$, must be 0,

$$-i_n + \sum_m i_{nm} = 0.$$

The current of the external generator in vertex $n$ must be equal to the sum of all edge currents going from this vertex,

$$i_n = \sum_m W_{nm}(x(n) - x(m)) = d_n x(n) - \sum_m W_{nm} x(m),$$
$$n = 0, 1, \dots, N-1,$$

where

$$d_n = \sum_m W_{nm} = \sum_{m=0}^{N-1} W_{nm}$$

is the degree of vertex $n$. The summation over $m$ can be extended to all vertices $m = 0, 1, \dots, N-1$ since $W_{nm} = 0$ if there is not an edge between vertices $n$ and $m$.

The previous equations can be written in a matrix form as

$$\mathbf{i} = \mathbf{D}\mathbf{x} - \mathbf{W}\mathbf{x}$$

or

$$\mathbf{L}\mathbf{x} = \mathbf{i}$$

where $\mathbf{L} = \mathbf{D} - \mathbf{W}$ is the Laplacian of graph.

If the Laplacian matrix is decomposed as $\mathbf{L} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^T$ we get $\boldsymbol{\Lambda}\mathbf{U}^T\mathbf{x} = \mathbf{U}^T\mathbf{i}$ or

$$\boldsymbol{\Lambda}\mathbf{X} = \mathbf{I}$$

where $\mathbf{X} = \mathbf{U}^T\mathbf{x}$ and $\mathbf{I} = \mathbf{U}^T\mathbf{i}$ are GDFT of graph signals $\mathbf{x}$ and $\mathbf{i}$.

Components of the spectral transform vector $\mathbf{X}$ are such that

$$\lambda_k X(k) = I(k)$$

for each $k$.

Signal on an electrical circuit graph can be related to the presented theory in several ways. For example, potentials on all vertices could be measured with some measurement noise. In that case, filtering on a graph should be applied. Another possible case is when the external conditions are imposed, for example sources are applied to some vertices. We are then interested in potential values at all vertices. This problem corresponds to the graph signal reconstruction.

### *7.8.2 Heat Transfer*

The same model as in the resistive electrical circuit case can be used for a heat transfer network. In this case the signal values are the measured temperatures $x(n) = T(n)$. The heat flux is defined as

$$q_{nm} = (T(n) - T(m))C_{nm} = W_{nm}(x(n) - x(m)),$$

where $C_{nm}$ are the heat transfer constants, representing edge weights in the underlying graph. Then the input heat flux in the vertex $n$ is

$$q_n = \sum_m W_{nm}(x(n) - x(m)) = d_n x(n) - \sum_{m=0}^{N-1} W_{nm} x(m),$$

with

$$\mathbf{q} = \mathbf{L}\mathbf{x}$$

Active vertices are those where there is 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 Fig. 43.

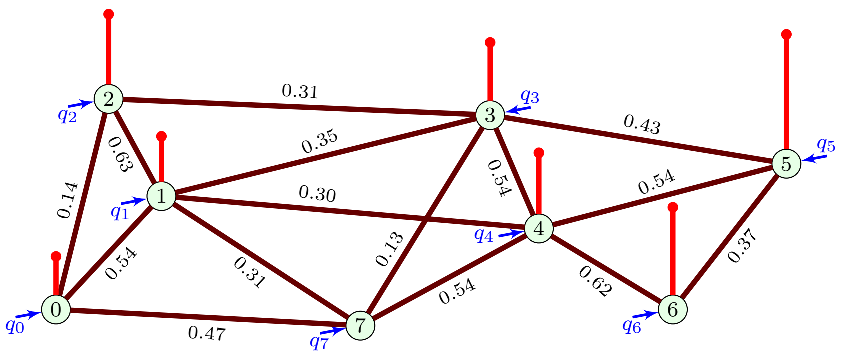


**Fig. 43** Temperature $x(n) = T(n)$ as a signal on a heat transfer graph

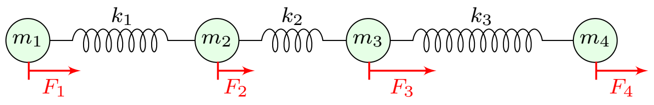


**Fig. 44** Spring-mass system

### *7.8.3 Spring-Mass Systems*

A spring mass system is used as a model of a graph and graph signal simulations. Consider a system of $N = 4$ masses corresponding to the line graph, Fig. 44. Assume that all displacements and forces are in the direction of the system line. The displacements $x(n)$ and the forces $F_n$, according to Hook's law in a steady state, are related as

$$\begin{aligned}
k_1(x(1) - x(2)) &= F_1 \\
k_1(x(2) - x(1)) + k_2(x(2) - x(3)) &= F_2 \\
k_2(x(3) - x(2)) + k_3(x(3) - x(4)) &= F_3 \\
k_3(x(4) - x(3)) &= F_4
\end{aligned}$$

In matrix form

$$\begin{bmatrix} k_1 & -k_1 & 0 & 0 \\ -k_1 & k_1 + k_2 & -k_2 & \\ 0 & -k_2 & k_2 + k_3 & -k_3 \\ 0 & 0 & -k_3 & k_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix}$$

$$\mathbf{Lx} = \mathbf{F}$$

These equations define a weighted graph and its corresponding Laplacian.

The Laplacian is 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 $\mathbf{Lx} = \mathbf{F}$ can be solved.

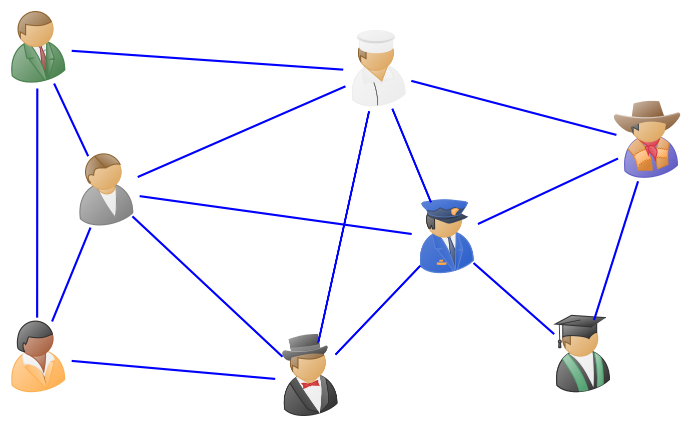


**Fig. 45** Social network graph example

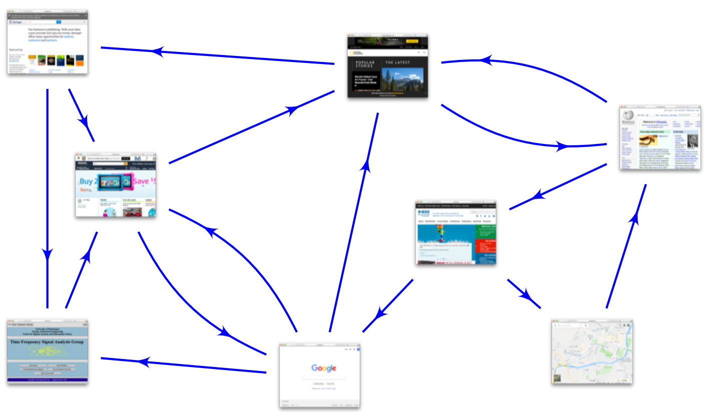


**Fig. 46** Linked pages graph example

## *7.8.4 Social Networks and Linked Pages*

Social networks are also examples of well defined graphs. The vertices are network members and the edges define their relationships in a network. If two members are related, corresponding edge weight is 1. In this case weight matrix is equal to the adjacency matrix. An example of a simple social network with $N = 8$ members is shown in Fig. 45.

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 Fig. 46. An interesting parameter for this kind of graphs is the PageRank.

### *7.8.5 PageRank*

For a directed graph, PageRank of vertex $n$ is defined as a graph signal satisfying the relation

$$x(n) = \sum_m \frac{1}{d_m} W_{mn} x(m),$$

where $W_{mn}$ are weights of the directed edges connecting the vertex $m$ to vertex $n$ and $d_m$ is the outgoing degree of the vertex $m$. This means that the PageRank of each vertex is related to the PageRank of connected vertices.

The PageRank is commonly calculated using an iterative procedure defined by

$$x_{k+1}(n) = \sum_m \frac{1}{d_m} W_{mn} x_k(m),$$

starting from arbitrary page ranks, for example $x_0(n) = 1$.

The PageRank was defined by Google to rank the web pages. In the original definition a scaling factor was added,

$$x_{k+1}(n) = 0.15 + 0.85 \sum_m \frac{1}{d_m} W_{mn} x_k(m),$$

As an example, consider the graph from Fig. 46 (it is the same graph as in Fig. 2(b)). We will calculate the PageRank for all vertices in this graph. The weight/adjacency matrix of this graph $\mathbf{W} = \mathbf{A}$ is given by (1), right. The outgoing vertex degrees are calculated as the sum of the matrix columns, $d_m = \sum_{n=0}^{7} A_{nm}$. Their values are $\mathbf{d} = [2 \;\; 3 \;\; 1 \;\; 3 \;\; 1 \;\; 2 \;\; 1 \;\; 2]$. Now the PageRank values for vertices can be obtained through the iterative procedure starting with initial page ranks $\mathbf{x}_0 = [1 \;\; 1 \;\; 1 \;\; 1 \;\; 1 \;\; 1 \;\; 1 \;\; 1]$. The results for PageRank in a few iterations are

$$\begin{bmatrix} \mathbf{x}_0^T \\ \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_5^T \\ \vdots \\ \mathbf{x}_{11}^T \end{bmatrix} = \begin{bmatrix} 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 \\ 0.83 & 1.83 & 0.50 & 1.33 & 0.50 & 1.50 & 0.50 & 1.00 \\ 0.58 & 1.42 & 0.67 & 2.00 & 0.75 & 1.67 & 0.25 & 1.67 \\ & & & & \vdots & & & \\ 0.73 & 1.60 & 0.82 & 1.71 & 0.61 & 1.11 & 0.31 & 1.10 \\ & & & & \vdots & & & \\ 0.79 & 1.57 & 0.86 & 1.71 & 0.57 & 1.14 & 0.29 & 1.07 \end{bmatrix}.$$

The matrix form of the iterations is

$$\mathbf{x}_{k+1} = \mathbf{W}_N \mathbf{x}_k,$$

where $\mathbf{W}_N$ is obtained from $\mathbf{W}$ 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.

In the considered example, the normalized adjacency/weighing matrix is

$$\mathbf{W}_N = \begin{bmatrix} 0 & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{3} & 0 & 0 & 0 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & \frac{1}{2} \\ 0 & 0 & 0 & \frac{1}{3} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 \end{bmatrix}.$$

The final, stationary state, page rank can be obtained from

$$\mathbf{x} = \mathbf{W}_N\mathbf{x}.$$

The final PageRank $\mathbf{x}$ is the eigenvector of matrix $\mathbf{W}_N$ corresponding to the eigenvalue equal to 1.

For the considered example, the eigenvalue decomposition of the matrix $\mathbf{W}_N$ results in the eigenvector, corresponding to eigenvalue 1,

$$\mathbf{x}^T = [0.79 \;\; 1.57 \;\; 0.86 \;\; 1.71 \;\; 0.57 \;\; 1.14 \;\; 0.29 \;\; 1.07].$$

The eigenvector is normalized with its mean value. It corresponds to the iterative solution obtained after 11 iterations.

### *7.8.6 Random Walk*

Assume that the signal $x(n)$ represents probalities that a random walker is present in vertex $n$. The random walker will transit from vertex $n$ to one of its neigbouring vertices $m$ with probabilities

$$p_{nm} = \frac{W_{nm}}{\sum_m W_{nm}},$$

where $W_{nm}$ are affinities of the walker to transit from vertex $n$ to vertex $m$.

The signal $x(n)$ calculation can be considered within the graph framework where $W_{nm}$ are edge weights.

The probabilities in the stage $(p+1)$ are calculated starting from probabilities in the previous stage as

$$\mathbf{x}_{p+1} = \mathbf{D}^{-1}\mathbf{W}\mathbf{x}_p$$

or $\mathbf{D}\mathbf{x}_{p+1} = \mathbf{W}\mathbf{x}_p$. Matrix $\mathbf{W}$ is a matrix of weighting coefficients and $\mathbf{D}$ is the degree matrix.

In stationary state, when $\mathbf{x}_{p+1} = \mathbf{x}_p = \mathbf{x}$ we have $\mathbf{D}\mathbf{x} = \mathbf{W}\mathbf{x}$ or

$$\mathbf{L}\mathbf{x} = \mathbf{0}.$$

Various problem formulations and solutions are now possible within the presented graph theory framework. For example, if we want to find probabilities that the walker reaches vertex 5 before he reaches vertex 7 from any vertex $n$ we have to solve the system $\mathbf{L}\mathbf{x} = \mathbf{0}$ with $x(5) = 1$ and $x(7) = 0$.

### 7.9 Gaussian Random Signal

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 function of the signal $\mathbf{x}$ is

$$P(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^N}} \det(\mathbf{\Sigma}_x^{-1}) \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \mathbf{\Sigma}_x^{-1}(\mathbf{x}-\boldsymbol{\mu})\right).$$

The inverse value of the autocovariance matrix is the precision matrix $\mathbf{\Theta} = \mathbf{\Sigma}_x^{-1}$. The name precision comes from the one-dimensional case in which the precision is inversely proportional to the variance $\Theta = 1/\sigma^2$.

The maximum likelihood estimate of $\mathbf{x}$ is obtained by minimizing

$$E_x = \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \mathbf{\Sigma}_x^{-1}(\mathbf{x}-\boldsymbol{\mu}).$$

Its solution is

$$\mathbf{\Sigma}_x^{-1}(\mathbf{x}-\boldsymbol{\mu}) = 0.$$

For zero mean random signal, $\boldsymbol{\mu} = \mathbf{0}$ and $\mathbf{\Sigma}_x^{-1}\mathbf{x} = 0$. This corresponds to the energy of change minimization (maximal smoothness) in the graph.

The Laplacian corresponding to the information matrix is defined by

$$\mathbf{\Sigma}_x^{-1} = \mathbf{L} + \mathbf{P}$$

where $\mathbf{P}$ is a diagonal matrix such that sum of the Laplacian columns is zero. Note that some of non-diagonal elements of $\mathbf{\Sigma}_x^{-1}$ can be positive. In that case, additional conditions should be added to find the best solution avoiding positive coefficients on the Laplacian diagonal.

The edge weights can be extracted from the Laplacian matrix. Since the Laplacian is defined using signal values, this is a point when the presented analysis meets the discussion from the previous section.

## 8 Appendix

### 8.1 Graph Signal Calculation Using Laplacian

The graph signal $\mathbf{x}$ can be calculated using this system of linear equations with the vector of external sources. Since the Laplacian is a singular matrix, one graph signal value (potential) should be considered as a free/referent variable.

In the case when there is no external generator, the graph signal $\mathbf{x}$ satisfies the following equation

$$\mathbf{L}\mathbf{x} = \mathbf{0}.$$

This equation corresponds to the minimum of the energy of change in $\mathbf{x}$ defined by the quadratic form

$$E_x = \mathbf{x}^T\mathbf{L}\mathbf{x} = \frac{1}{2}\sum_{n=0}^{N-1}\sum_{m=0}^{N-1} W_{nm}(x(n)-x(m))^2 = \frac{1}{2}\sum_{n=0}^{N-1}\sum_{m=0}^{N-1} P_{nm}.$$

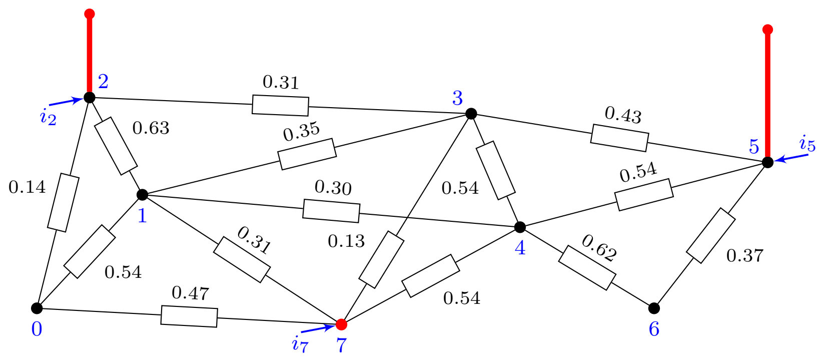


**Fig. 47** Electric potential $x(n)$ as a signal on an electric circuit graph at three vertices with nonzero external sources

where $P_{nm}$ can be considered as a power dissipated in the edge between vertices $n$ and $m$. It follows from $\partial E_x/\partial \mathbf{x}^T = 2\mathbf{L}\mathbf{x} = \mathbf{0}$ and (38). The solution of this equation $x(n)$ is a constant defined by the signal value at the reference vertex.

For nontrivial solutions, there should be an external source in at least two vertices. Assume that one of them is chosen as the referent vertex. Signal or external source values at these vertices are sufficient to find signal values at all other vertices.

As an example, consider the graph and signal sensed on the graph presented in Fig. 42. The signal values are

$$\mathbf{x} = [0.57, 0.67, 1.03, 0.86, 0.90, 1.68, 1.29, 0]^T$$

and the Laplacian of the signal is

$$\mathbf{L}\mathbf{x} = [0, 0, 1, 0, 0, 2, 0, -3]^T.$$

This means that the vertices denoted by $0, 1, 3, 4, 6$ are not active in this case. Their values can be obtained as linear combinations of the signal at neighboring active vertices:

$$\begin{aligned}
-0.47x(7) - 0.14x(2) - 0.54x(1) + 1.15x(0) &= 0\\
-0.31x(7) - 0.30x(4) - 0.35x(3) - 0.63x(2) + 2.13x(1) - 0.54x(0) &= 0\\
-0.43x(5) - 0.54x(4) + 1.82x(3) - 0.31x(2) - 0.54x(1) &= 0\\
-0.62x(6) - 0.54x(5) + 2.00x(4) - 0.54x(3) - 0.30x(1) &= 0\\
0.99x(6) - 0.37x(5) - 0.62x(4) &= 0
\end{aligned}$$

Solving this system with known signal values $x(2) = 1.03$, $x(5) = 1.68$, and $x(7) = 0$ at the active vertices, Fig. 47 we get the remaining signal values

$$\mathbf{x}_p = [x(0), x(1), x(3), x(4), x(6)]^T = [0.57, 0.67, 0.86, 0.90, 1.29]^T.$$

Assume that only some of the vertices are active, with external sources. Denote the set of these $M$ active vertices by $\mathcal{E}$. Then the vertices without external sources are $\mathcal{H}$, such that $\mathcal{V} = \mathcal{E} \cup \mathcal{H}$ and $\mathcal{E} \cap \mathcal{H} = \emptyset$. For a full reconstruction of this signal we have to know only $M$ signal values $x(n)$ at vertices $n \in \mathcal{E}$ or any other linearly independent $M$ graph signal samples. The remaining $N - M$ graph signal samples are obtained from the equations following from the fact that the Laplacian at $n \in \mathcal{H}$ is zero-valued.

From the circuit theory, it is well known that this kind of graph can be downscaled without any influence to the signal at vertices where Laplacian is nonzero. The vertices with zero Laplacian can be omitted by using so called Y-$\Delta$ (star-mesh) transformation. The resulting graph has a reduced number of vertices, while the number of edges may be increased.

This kind of interpolation can be applied to classic, time-domain signals as well. The Laplacian of a time-domain signal at instant (vertex) $n$ is calculated as $2x(n) - x(n-1) - x(n+1)$. Its zero value will indicate that this vertex (instant) is not active (there is no external source). Therefore, this value can be omitted since it can be obtained from the condition that Laplacian is zero from other signal values at instants $n-1$ and $n+1$. An illustration of such a signal and reconstruction based on the signal values at the Laplacian nonzero positions is presented in Fig. 48.

In this way, vertices where the signal behavior is changed are detected using the Laplacian. Note that the two-dimensional Laplacian is a classical tool in the image edge detection.

This kind of subsampling can be considered as a signal processing and graph signal processing equivalence of the Laplace differential equation:

$$\mathbf{L}\{V(x,y)\} = \nabla^2 V(x,y) = \frac{\partial^2 V(x,y)}{\partial x^2} + \frac{\partial^2 V(x,y)}{\partial y^2} = 0$$

with given boundary conditions (Dirichlet problem):

$$V(x,y) = f(x,y) \text{ on boundary } D(x,y) = 0.$$

In graph signal processing, vertices with a nonzero Laplacian define the boundary set and the signal values on these vertices are the boundary conditions.

In general, when the Laplacian of a graph signal is nonzero at all vertices we can apply hard thresholding, keeping large Laplacian values and neglecting the small ones in a linear approximation of the graph signal. In this way we can keep just the signal values on the vertices

$$n \in \mathcal{E} \text{ if } |\mathbf{Lx}| \geq \tau \text{ at vertex } n.$$

The correction (high pass part) of the signal is equal to the Laplacian of signal on $n \notin \mathcal{E}$ . It can be used to adjust the values of linear approximation to the full signal reconstruction. If a graph signal is noisy then the Laplacian can be used for the detection of the active (bounday condition) vertices. The signal values can be obtained by a mean squared linear approximation of the noisy data with the detected positions of the discontinuity of linear behavior.

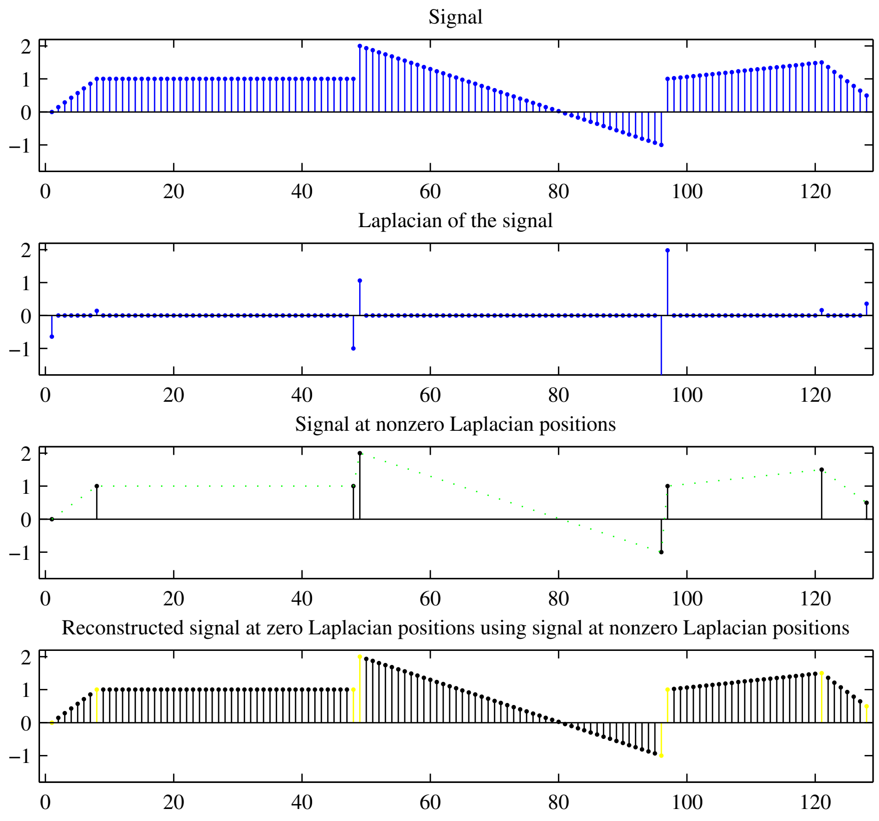


**Fig. 48** Reconstructed signal at zero Laplacian positions using signal at nonzero Laplacian positions. Signals are presented as functions of the vertex index $n$.

Some vertices may be active for some signal realizations and not active for other signal realizations. Some vertices may not be active for the whole set of realizations, meaning that the graph can be downscaled not only for one considered signal $x(n)$ but for a class of signals by omitting some of the vertices using star-mesh conversions.

## 8.2 Random Signal Simulation on a Graph

The presentation of a graph and graph signal within the circuit theory can be used to simulate random signals on graphs. Several approaches are possible. Here we will present few of them.

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 the graph is simulated as a solution of the system of equations

$$\mathbf{L}\mathbf{x}_p = \boldsymbol{\varepsilon}_p$$

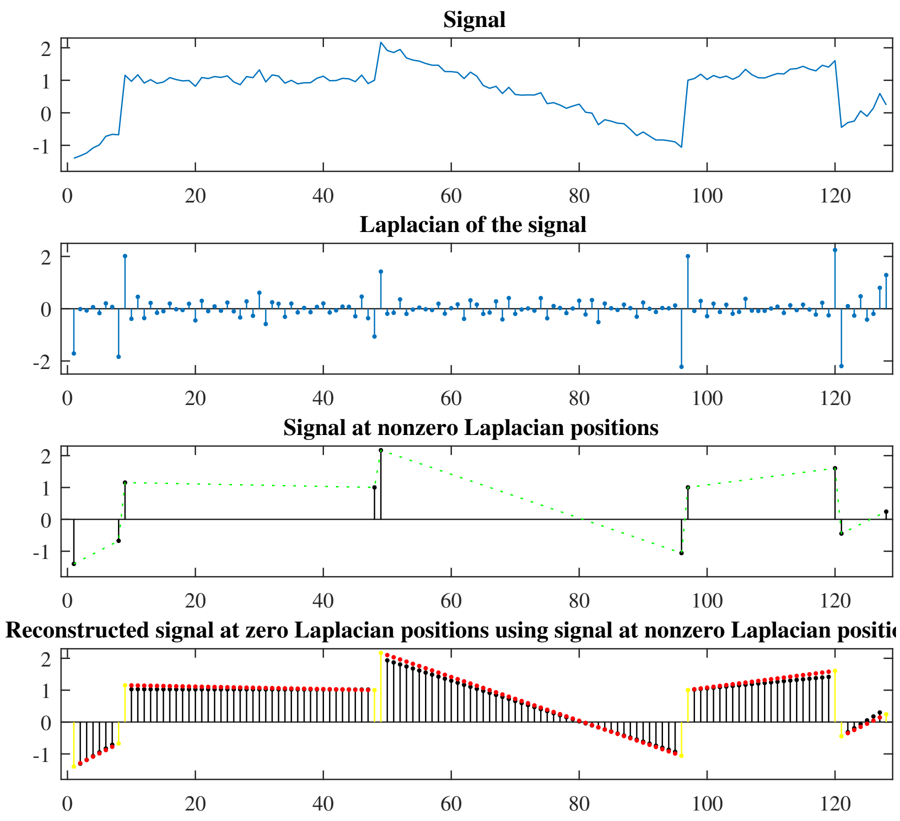


**Fig. 49** Reconstructed signal at small Laplacian positions using signal at nonzero Laplacian positions and linear approximation

using $\mathbf{i}_p = \boldsymbol{\varepsilon}_p$. One of the external sources (randomly chosen for each observation $p$) should compensate all other sources, to ensure $\sum_{n=0}^{N-1} \varepsilon_p(n) = 0$.

2) Assume that the graph is initiated at only one of its vertices (and the reference vertex) with random white 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 obtaied as a solution of

$$\mathbf{L}\mathbf{x}_p = \mathbf{i}_p$$

where $i_p(n) = \varepsilon_p \delta(n - n_i) - \varepsilon_p \delta(n - n_j)$ and $n_i$ and $n_j$ are two randomly selected vertices in each observation.

3) A simple random graph signal can be simulated using its values at two randomly positioned vertices and $\mathbf{L}\mathbf{x}_p = \mathbf{0}$. Assuming that $x_p(n) = \varepsilon_p \delta(n - n_i) + \epsilon_p \delta(n - n_j)$ and $n_i$ and $n_j$ are two randomly selected vertices in each observation, we may solve the system for all other signal samples using

$$\mathbf{L}\mathbf{x}_p = \mathbf{0}.$$

With 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. In this case there is no need for compensation, meaning that $\varepsilon_p$ and $\epsilon_p$ could be independent random variables.

4) Assume that the signal on a graph is formed using a linear combination of a white noise $\mathbf{x}_p^{(0)} = \boldsymbol{\varepsilon}_p$ and its graph shifted versions. The first iteration is

$$\mathbf{x}_p^{(1)} = \alpha_1 \mathbf{L}\mathbf{x}_p^{(0)} + \mathbf{x}_p^{(0)}.$$

After $M$ iterations we get

$$\mathbf{x}_p^{(M)} = \alpha_M \mathbf{L}\mathbf{x}_p^{(M-1)} + \mathbf{x}_p^{(0)} = (h_M \mathbf{L}^M + h_{M-1}\mathbf{L}^{M-1} + \dots + h_1 \mathbf{L}^1 + 1)\boldsymbol{\varepsilon}_p,$$

where $h_m = \alpha_m \alpha_{m-1} \dots \alpha_1$. The final signal

$$\mathbf{x}_p = \mathbf{x}_p^{(M)} = H(\mathbf{L})\boldsymbol{\varepsilon}_p,$$

is a GWSS signal.

5) In graph signal simulation we may also use the adjacency matrix and graph shifts. Assume that an undirected graph with adjacency matrix $\mathbf{A}$ is initiated at $N_a$ randomly chosen vertices $n_1, n_2, \dots, n_{N_a}$, $\eta = N_a/N$, with spikes $\delta(n - n_i)$, $i = 1, 2, \dots, N_a$. If we shift these spikes $K$ times we get

$$\mathbf{x} = \mathbf{A}^K \sum_{i=1}^{N_a} \delta(n - n_i).$$

Parameters $K$ and $N_a$ define the resulting signal smoothness. An example of one realization of such a signal is presented in Fig. 22 for $\eta = 1/8$, $K = 1$ (upper subplots) and $\eta = 2/8$, $K = 1$ (lower subplots) using spikes $a_i\delta(n-n_i)$, where $a_i$ are the spike amplitudes.

6) In classical Fourier analysis, the signals are commonly simulated as sums of the harmonic basis functions. This kind of simulation may be used in graph signal processing as well. 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 Laplacian or adjacency matrix eigenvectors, and $a_k$ are random constants. This kind of graph signal simulation, with or without an additive noise, is often used in this chapter.

## 8.3 From the Newton Minimization to the Graphical LASSO

### *8.3.1 Newton Method*

Fist we will shortly 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 minimum position of $f(x)$ by $x^*$. The first derivative of $f(x)$ at $x^* = x + \Delta x$ can be expanded into a Taylor series around a point $x$, using the linear approximation, as

$$f'(x^*) = f'(x) + f''(x)\Delta x. \tag{84}$$

Since $f'(x^*) = 0$ by definition, with $\Delta x = x^* - x$, relation (84) can be rewritten as

$$x^* - x = -\frac{f'(x)}{f''(x)}.$$

This form is used to define an iterative procedure (Newton's iterative method) for finding $x^*$ starting from an $x = x_0$ as

$$x_{k+1} = x_k - \alpha f'(x_k).$$

Parameter $\alpha$ is commonly used instead of $1/f''(x)$ to control the iteration step. Its value should be $0 < \alpha \leq \max(|1/f''(x)|)$, for the considered interval of $x$. This is the form of the well-known steepest descend method for convex function minimization.

The value $x^* = x - \alpha f'(x)$, with $\alpha = 1/f''(x)$ would be obtained as result of the minimization of a cost function defined by the quadratic form

$$x^* = \arg\min_z G(z) = \arg\min_z (f(x) + f'(x)(z - x) + \frac{1}{2\alpha}(z - x)^2).$$

From $d(f(x) + f'(x)(z - x) + \frac{1}{2\alpha}(z - x)^2)/dz = 0$ we would get $x^* = z$.

Next assume that we want to minimize the cost function

$$J(x) = \frac{1}{2\alpha}(x - y)^2 + \lambda|x|,$$

where $\lambda$ is a parameter. From $dJ(x)/dx = (x - y)/\alpha + \lambda \text{sign}(x) = 0$ we get

$$x + \lambda\alpha \text{sign}(x) = y.$$

The soft-thresholding is used as a solution of this equation. It is denoted as $\text{soft}(y, \alpha\lambda)$ and defined by

$$x = \text{soft}(y, \alpha\lambda) = \begin{cases} y + \alpha\lambda & \text{for } y < -\alpha\lambda \\ 0 & \text{for } |y| \leq \alpha\lambda \\ y - \alpha\lambda & \text{for } y > \alpha\lambda. \end{cases}$$

### *8.3.2 LASSO*

For the lasso minimization we will consider the cost function

$$J(\mathbf{X}) = \|\mathbf{y} - \mathbf{A}\mathbf{X}\|_2^2 + \lambda\|\mathbf{X}\|_1 = \|\mathbf{y}\|_2^2 - 2\mathbf{X}^T\mathbf{A}^T\mathbf{y} + \mathbf{X}^T\mathbf{A}^T\mathbf{A}\mathbf{X} + \lambda\|\mathbf{X}\|_1,$$

where $\mathbf{y}$ is a $M \times 1$ column vector, $\mathbf{X}$ is a $N \times 1$ column vector, and $\mathbf{A}$ is an $M \times N$ matrix.

Minimization of this cost function with respect to $\mathbf{X}$ will produce its value such that $\mathbf{AX}$ is as close to $\mathbf{y}$ as possible, promoting the sparsity of $\mathbf{X}$ at the same time. Balance between these two requirements is defined by the parameter $\lambda$.

Consider first the differentiable part of the cost function $J(\mathbf{X})$ denoted by $J_D(\mathbf{X}) = \|\mathbf{y} - \mathbf{A}\mathbf{X}\|_2^2 = (\mathbf{y} - \mathbf{A}\mathbf{X})^T(\mathbf{y} - \mathbf{A}\mathbf{X})$. 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

$$\frac{\partial^2 J_D(\mathbf{X})}{(\partial \mathbf{X}^T)^2} = 2\mathbf{A}^T\mathbf{A}.$$

The linear model for the first derivative of $J_D(\mathbf{X})$ around its minimum is

$$\frac{\partial J_D(\mathbf{X}^*)}{\partial \mathbf{X}^T} = \frac{\partial J_D(\mathbf{X})}{\partial \mathbf{X}^T} + (\varDelta\mathbf{X})\frac{\partial^2 J_D(\mathbf{X})}{(\partial \mathbf{X}^T)^2}.$$

By replacing the inverse of the second order derivative by a constant diagonal matrix $\alpha\mathbf{I}$ we get

$$\varDelta\mathbf{X} = \mathbf{X}^* - \mathbf{X} = -\alpha\frac{\partial J_D(\mathbf{X})}{\partial \mathbf{X}^T}$$

or

$$\mathbf{X}^* = \mathbf{X} - \alpha\frac{\partial J_D(\mathbf{X})}{\partial \mathbf{X}^T} \tag{85}$$

with $0 < \alpha < 1/\max\|2\mathbf{A}^T\mathbf{A}\| = 1/(2\lambda_{\max})$, where $\lambda_{\max}$ is the maximal 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 difference $\mathbf{Z} - (\mathbf{X} - \alpha\mathbf{I}\frac{\partial J_D(\mathbf{X})}{\partial \mathbf{X}^T})$ and the norm-one of $\mathbf{Z}$, by forming the cost function $G(\mathbf{Z})$ as

$$\mathbf{X}^* = \arg\min_{\mathbf{Z}} G(\mathbf{Z}) = \arg\min_{\mathbf{Z}}(\frac{1}{2\alpha}\|\mathbf{Z} - (\mathbf{X} - \alpha\mathbf{I}\frac{\partial J_D(\mathbf{X})}{\partial \mathbf{X}^T})\|^2 + \lambda\|\mathbf{Z}\|_1).$$

This minimization will produce $\mathbf{Z}$ as close as possible to the desired solution (85), minimizing its norm-one at the same time. The balance parameter is $\lambda$.

The solution of

$$\mathbf{X}^* = \arg\min_{\mathbf{Z}} G(\mathbf{Z}) = \frac{1}{2\alpha}\|\mathbf{Z} - \mathbf{Y}\|^2 + \lambda\|\mathbf{Z}\|_1$$

is obtained from

$$\frac{1}{\alpha}(\mathbf{X}^* - \mathbf{Y}) + \lambda \text{sign}(\mathbf{X}^*) = 0.$$

Using the soft function we can write

$$\mathbf{X}^* = \text{soft}(\mathbf{Y}, \alpha\lambda).$$

Next we will replace the value of $\mathbf{Y}$ by

$$\mathbf{Y} = (\mathbf{X} - \alpha\mathbf{I}\frac{\partial J_D(\mathbf{X})}{\partial \mathbf{X}^T}) = \mathbf{X} - \alpha\mathbf{I}(-2\mathbf{A}^T\mathbf{y} + 2\mathbf{X}^T\mathbf{A}^T\mathbf{A})$$
$$= 2\alpha\mathbf{A}^T\mathbf{y} + (\mathbf{I} - 2\alpha\mathbf{A}^T\mathbf{A})\mathbf{X}.$$

The iterative formula for the solution of the defined minimization problem is obtained by replacing $\mathbf{X}^* = \mathbf{X}_{k+1}$ and $\mathbf{X} = \mathbf{X}_k$ as

$$\mathbf{X}_{k+1} = \text{soft}(2\alpha\mathbf{A}^T(\mathbf{y} - \mathbf{A}\mathbf{X}_k) + \mathbf{X}_k, \alpha\lambda).$$

This formula can easily be written for each element of $\mathbf{X}_k$. This is the LASSO (Least Absolute Shrinkage and Selection Operator) iterative algorithm. As the initial estimate $\mathbf{X}_0 = \mathbf{A}^T\mathbf{y}$ is commonly used.

### *8.3.3 Graphical LASSO*

In graph model learning, the cost function in the form

$$J(\mathbf{Q}) = -\log\det\mathbf{Q} + \text{Trace}(\mathbf{Q}\mathbf{R}_x) + \|\mathbf{Q}\|_1$$

may be obtained. Here $\mathbf{Q}$ is the generalized Laplacian $N \times N$ matrix, while $\mathbf{R}_x$ is the available $N \times N$ correlation matrix. The meaning of these terms is explained within the main part of this chapter.

The derivative of the cost function with respect to the elements of $\mathbf{Q}$ can be written as

$$-\mathbf{Q}^{-1} + \mathbf{R}_x + \text{sign}(\mathbf{Q}) = 0 \qquad (86)$$

at $\partial J(\mathbf{Q})/\partial\mathbf{Q} = 0$.

Note that $\log\det\mathbf{Q} = \sum_{i=0}^{N-1}\log\lambda_i = \text{Trace}(\log\mathbf{\Lambda}) = \text{Trace}(\log\mathbf{Q})$, where $\lambda_i$ are the eigenvalues of $\mathbf{Q}$.

Introducing the notation $\mathbf{W} = \mathbf{Q}^{-1}$ or

$$\mathbf{W}\mathbf{Q} = \mathbf{I}$$

we can write

$$\begin{bmatrix}\mathbf{W}_{11} & \mathbf{w}_{12}\\ \mathbf{w}_{12}^T & w_{22}\end{bmatrix}\begin{bmatrix}\mathbf{Q}_{11} & \mathbf{q}_{12}\\ \mathbf{q}_{12}^T & q_{22}\end{bmatrix} = \begin{bmatrix}\mathbf{I} & \mathbf{0}\\ \mathbf{0}^T & 1\end{bmatrix}. \qquad (87)$$

Multiplying the first row block of $\mathbf{W}$ with the last column block of $\mathbf{Q}$ we get

$$\mathbf{W}_{11}\mathbf{q}_{12} + \mathbf{w}_{12}q_{22} = \mathbf{0}$$

This means that

$$\mathbf{w}_{12} = -\mathbf{W}_{11}\mathbf{q}_{12}/q_{22} = \mathbf{W}_{11}\boldsymbol{\beta}$$

where

$$\boldsymbol{\beta} = -\mathbf{q}_{12}/q_{22}$$

is normalized with $q_{22} > 0$.

Now, from the derivative equation (86) we may write

$$-\begin{bmatrix}\mathbf{W}_{11} & \mathbf{w}_{12}\\ \mathbf{w}_{12}^T & w_{22}\end{bmatrix} + \begin{bmatrix}\mathbf{R}_{11} & \mathbf{r}_{12}\\ \mathbf{r}_{12}^T & r_{22}\end{bmatrix} + \mathrm{sign}(\begin{bmatrix}\mathbf{Q}_{11} & \mathbf{q}_{12}\\ \mathbf{q}_{12}^T & q_{22}\end{bmatrix}) = \mathbf{0}.$$

For the upper right block we can write

$$-\mathbf{w}_{12} + \mathbf{r}_{12} + \mathrm{sign}(\mathbf{q}_{12}) = \mathbf{0}$$

or after replacing $\mathbf{w}_{12} = \mathbf{W}_{11}\boldsymbol{\beta}$ and $\mathbf{q}_{12} = -\boldsymbol{\beta}/q_{22}$ we get

$$-\mathbf{W}_{11}\boldsymbol{\beta} + \mathbf{r}_{12} - \mathrm{sign}(\boldsymbol{\beta}) = \mathbf{0}.$$

The solution of this equation for $\boldsymbol{\beta}$ is already defined within LASSO consideration. It is

$$\beta_i = \mathrm{soft}(r_{12}(i) - \sum_{k\neq i} W_{11}(k,i)\beta_k, \lambda)/W_{11}(i). \tag{88}$$

Now we may summarize the graphical LASSO (GLASSO) iterative algorithm as:

- In the initial step,
$$\mathbf{W} = \mathbf{R}_x + \lambda\mathbf{I}$$
is used. For each coordinate $j = 1, 2, \dots, N$, the matrix equation of form (87) is written. For each $j$ the reduced matrix $\mathbf{W}_{11}$ is formed by omitting the $j$th row and the $j$th column. Then the matrix $\mathbf{R}_x$ is rearranged appropriately.
- Equation (88) is solved.
- The matrix $\mathbf{W}$ is updated for each $j$ with
$$\mathbf{w}_{12} = \mathbf{W}_{11}\boldsymbol{\beta}.$$
- In the final iteration, for each $j$, the value of matrix $\mathbf{Q}$ is updated as
$$\mathbf{q}_{12} = -\boldsymbol{\beta}/q_{22}$$
with $1/q_{22} = w_{22} - \mathbf{w}_{12}^T - \boldsymbol{\beta}$.

## 9 Conclusion

An introduction to graph signal processing is presented. This chapter consists of three main parts. In the first part, a review of graphs is given. Next, the signal on graph definitions, basic properties, and systems for processing signals on graphs are reviewed. Finally, the graph topologies are discussed. The appendix provides some supplementary material for better understanding of the principles presented in the main part of the chapters. The topic of this book is the spectral localization through vertex-frequency analysis [39, 83–100], which will be presented in the next chapters.

## References

1. L. J. Grady and J. R. Polimeni, *Discrete calculus: Applied analysis on graphs for computational science.* Springer Science & Business Media, 2010.
2. S. S. Ray, *Graph theory with algorithms and its applications: in applied science and technology.* Springer Science & Business Media, 2012.
3. A. Marques, A. Ribeiro, and S. Segarra, "Graph signal processing: Fundamentals and applications to diffusion processes," in *Int. Conf. Accoustic, Speech and Signal Processing, (ICASSP), 2017*, IEEE, 2017.
4. H. Krim and A. B. Hamza, *Geometric methods in signal and image analysis.* Cambridge University Press, 2015.
5. A. Bunse-Gerstner and W. B. Gragg, "Singular value decompositions of complex symmetric matrices," *Journal of Computational and Applied Mathematics*, vol. 21, no. 1, pp. 41–54, 1988.
6. D. S. Grebenkov and B.-T. Nguyen, "Geometrical structure of laplacian eigenfunctions," *SIAM Review*, vol. 55, no. 4, pp. 601–667, 2013.
7. R. Bapat, "The laplacian matrix of a graph," *Mathematics Student-India*, vol. 65, no. 1, pp. 214–223, 1996.
8. S. O'Rourke, V. Vu, and K. Wang, "Eigenvectors of random matrices: a survey," *Journal of Combinatorial Theory, Series A*, vol. 144, pp. 361–442, 2016.
9. K. Fujiwara, "Eigenvalues of laplacians on a closed riemannian manifold and its nets," *Proceedings of the American Mathematical Society*, vol. 123, no. 8, pp. 2585–2594, 1995.
10. S. U. Maheswari and B. Maheswari, "Some properties of cartesian product graphs of cayley graphs with arithmetic graphs," *International Journal of Computer Applications*, vol. 138, no. 3, 2016.
11. D. M. Cvetković, M. Doob, and H. Sachs, *Spectra of graphs: theory and application*, vol. 87. Academic Pr, 1980.
12. D. M. Cvetković and M. Doob, "Developments in the theory of graph spectra," *Linear and Multilinear Algebra*, vol. 18, no. 2, pp. 153–181, 1985.
13. D. M. Cvetković and I. Gutman, *Selected topics on applications of graph spectra.* Matematicki Institut SANU, 2011.
14. A. E. Brouwer and W. H. Haemers, *Spectra of graphs.* Springer Science & Business Media, 2011.
15. F. Chung, *Spectral graph theory.* Providence, RI: AMS, 1997.
16. O. Jones, "Spectra of simple graphs," *Whitman college, May*, vol. 13, 2013.
17. D. Mejia, O. Ruiz-Salguero, and C. A. Cadavid, "Spectral-based mesh segmentation," *International Journal on Interactive Design and Manufacturing (IJIDeM)*, vol. 11, no. 3, pp. 503–514, 2017.
18. H. Lu, Z. Fu, and X. Shu, "Non-negative and sparse spectral clustering," *Pattern Recognition*, vol. 47, no. 1, pp. 418–426, 2014.
19. X. Dong, P. Frossard, P. Vandergheynst, and N. Nefedov, "Clustering with multi-layer graphs: A spectral perspective," *IEEE Transactions on Signal Processing*, vol. 60, no. 11, pp. 5820–5831, 2012.

20. R. Horaud, “A short tutorial on graph laplacians, laplacian embedding, and spectral clustering,” 2009.
21. R. Hamon, P. Borgnat, P. Flandrin, and C. Robardet, “Relabelling vertices according to the network structure by minimizing the cyclic bandwidth sum,” *Journal of Complex Networks*, vol. 4, no. 4, pp. 534–560, 2016.
22. M. Masoumi and A. B. Hamza, “Spectral shape classification: A deep learning approach,” *Journal of Visual Communication and Image Representation*, vol. 43, pp. 198–211, 2017.
23. M. Masoumi, C. Li, and A. B. Hamza, “A spectral graph wavelet approach for nonrigid 3d shape retrieval,” *Pattern Recognition Letters*, vol. 83, pp. 339–348, 2016.
24. J. M. Mouraaa, “Graph signal processing,” in *Cooperative and Graph Signal Processing*, pp. 239–259, Elsevier, 2018.
25. M. Vetterli, J. Kovačević, and V. Goyal, *Foundations of signal processing.* Cambridge University Press., 2014.
26. A. Sandryhaila and J. M. Moura, “Discrete signal processing on graphs,” *IEEE Transactions on Signal Processing*, vol. 61, no. 7, pp. 1644–1656, 2013.
27. V. N. Ekambaram, *Graph-Structured Data Viewed Through a Fourier Lens.* University of California, Berkeley, 2014.
28. A. Sandryhaila and J. M. Moura, “Discrete signal processing on graphs: Frequency analysis.,” *IEEE Transactions on Signal Processing*, vol. 62, no. 12, pp. 3042–3054, 2014.
29. A. Sandryhaila and J. M. Moura, “Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure,” *IEEE Signal Processing Magazine*, vol. 31, no. 5, pp. 80–90, 2014.
30. D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, “The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains,” *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 83–98, 2013.
31. R. Hamon, P. Borgnat, P. Flandrin, and C. Robardet, “Extraction of temporal network structures from graph-based signals,” *IEEE Transactions on Signal and Information Processing over Networks*, vol. 2, no. 2, pp. 215–226, 2016.
32. S. Chen, A. Sandryhaila, J. M. Moura, and J. Kovačević, “Signal denoising on graphs via graph filtering,” in *Signal and Information Processing (GlobalSIP), 2014 IEEE Global Conference on*, pp. 872–876, IEEE, 2014.
33. A. Gavili and X.-P. Zhang, “On the shift operator, graph frequency, and optimal filtering in graph signal processing,” *IEEE Transactions on Signal Processing*, vol. 65, no. 23, pp. 6303–6318, 2017.
34. A. Venkitaraman, S. Chatterjee, and P. Händel, “Hilbert transform, analytic signal, and modulation analysis for graph signal processing,” *arXiv preprint arXiv:1611.05269*, 2016.
35. A. Agaskar and Y. M. Lu, “A spectral graph uncertainty principle,” *IEEE Trans. Inf. Theory*, vol. 59, no. 7, pp. 4338–4356, 2013.
36. X. Yan, B. M. Sadler, R. J. Drost, P. L. Yu, and K. Lerman, “Graph filters and the z-laplacian,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, pp. 774–784, Sept 2017.
37. X. Wang, J. Chen, and Y. Gu, “Local measurement and reconstruction for noisy bandlimited graph signals,” *Signal Processing*, vol. 129, pp. 119–129, 2016.
38. S. Segarra and A. Ribeiro, “Stability and continuity of centrality measures in weighted graphs,” *IEEE Transactions on Signal Processing*, vol. 64, no. 3, pp. 543–555, 2016.
39. D. I. Shuman, B. Ricaud, and P. Vandergheynst, “Vertex-frequency analysis on graphs,” *Applied and Computational Harmonic Analysis*, vol. 40, no. 2, pp. 260–291, 2016.
40. S. Chen, A. Sandryhaila, and J. Kovačević, “Sampling theory for graph signals,” in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pp. 3392–3396, IEEE, 2015.
41. S. Chen, R. Varma, A. Sandryhaila, and J. Kovačević, “Discrete signal processing on graphs: Sampling theory,” *IEEE Transactions on Signal Processing*, vol. 63, no. 24, pp. 6510–6523, 2015.

42. S. Chen, A. Sandryhaila, J. M. Moura, and J. Kovačević, "Signal recovery on graphs: Variation minimization," *IEEE Transactions on Signal Processing*, vol. 63, no. 17, pp. 4609–4624, 2015.
43. S. Chen, R. Varma, A. Singh, and J. Kovačević, "Signal recovery on graphs: Fundamental limits of sampling strategies," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 2, no. 4, pp. 539–554, 2016.
44. M. Tsitsvero, S. Barbarossa, and P. Di Lorenzo, "Signals on graphs: uncertainty principle and sampling," *IEEE Trans. Signal Process.*, 2016. DOI 10.1109/TSP.2016.2573748.
45. X. Wang, P. Liu, and Y. Gu, "Local-set-based graph signal reconstruction," *IEEE Transactions on Signal Processing*, vol. 63, no. 9, pp. 2432–2444, 2015.
46. L. Stanković, E. Sejdić, S. Stanković, M. Daković, and I. Orović, "A tutorial on sparse signal reconstruction and its applications in signal processing," *Circuits, Systems, and Signal Processing*, pp. 1–58, 2018.
47. L. Stanković, "Digital signal processing with selected topics," 2015.
48. S. K. Narang and A. Ortega, "Downsampling graphs using spectral theory," in *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pp. 4208–4211, IEEE, 2011.
49. H. Q. Nguyen and M. N. Do, "Downsampling of signals on graphs via maximum spanning trees.," *IEEE Trans. Signal Processing*, vol. 63, no. 1, pp. 182–191, 2015.
50. S. K. Narang and A. Ortega, "Perfect reconstruction two-channel wavelet filter banks for graph structured data," *IEEE Transactions on Signal Processing*, vol. 60, no. 6, pp. 2786–2799, 2012.
51. S. Segarra, A. G. Marques, G. Leus, and A. Ribeiro, "Interpolation of graph signals using shift-invariant graph filters.," in *EUSIPCO*, pp. 210–214, 2015.
52. A. G. Marques, S. Segarra, G. Leus, and A. Ribeiro, "Sampling of graph signals with successive local aggregations.," *IEEE Trans. Signal Processing*, vol. 64, no. 7, pp. 1832–1843, 2016.
53. A. Anis, A. Gadde, and A. Ortega, "Efficient sampling set selection for bandlimited graph signals using graph spectral proxies," *IEEE Transactions on Signal Processing*, vol. 64, no. 14, pp. 3775–3789, 2016.
54. H. Behjat, U. Richter, D. Van De Ville, and L. Sörnmo, "Signal-adapted tight frames on graphs.," *IEEE Trans. Signal Process.*, vol. 64, no. 22, pp. 6017–6029, 2016.
55. Y. Tanaka and A. Sakiyama, "M-channel oversampled graph filter banks," *IEEE Trans. Signal Process.*, vol. 62, no. 14, pp. 3578–3590, 2014.
56. A. Sakiyama and Y. Tanaka, "Oversampled graph laplacian matrix for graph filter banks," *IEEE Transactions on Signal Processing*, vol. 62, no. 24, pp. 6425–6437, 2014.
57. N. Tremblay and P. Borgnat, "Subgraph-based filterbanks for graph signals," *IEEE Trans. Signal Process.*, vol. 64, no. 15, pp. 3827–3840, 2016.
58. J. Leskovec and C. Faloutsos, "Sampling from large graphs," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 631–636, ACM, 2006.
59. N. Perraudin and P. Vandergheynst, "Stationary signal processing on graphs," *IEEE Transactions on Signal Processing*, vol. 65, no. 13, pp. 3462–3477, 2017.
60. A. G. Marques, S. Segarra, G. Leus, and A. Ribeiro, "Stationary graph processes and spectral estimation," *IEEE Transactions on Signal Processing*, vol. 65, no. 22, pp. 5911–5926, 2017.
61. A. Loukas and N. Perraudin, "Stationary time-vertex signal processing," *arXiv preprint arXiv:1611.00255*, 2016.
62. S. P. Chepuri and G. Leus, "Subsampling for graph power spectrum estimation," in *Sensor Array and Multichannel Signal Processing Workshop (SAM), 2016 IEEE*, pp. 1–5, IEEE, 2016.
63. G. Puy, N. Tremblay, R. Gribonval, and P. Vandergheynst, "Random sampling of bandlimited signals on graphs," *Applied and Computational Harmonic Analysis*, 2016.
64. C. Zhang, D. Florêncio, and P. A. Chou, "Graph signal processing-a probabilistic framework," *Microsoft Res., Redmond, WA, USA, Tech. Rep. MSR-TR-2015-31*, 2015.
65. J. Friedman, T. Hastie, and R. Tibshirani, "Sparse inverse covariance estimation with the graphical lasso," *Biostatistics*, vol. 9, no. 3, pp. 432–441, 2008.

66. N. Meinshausen, P. Bühlmann, *et al.*, “High-dimensional graphs and variable selection with the lasso,” *The annals of statistics*, vol. 34, no. 3, pp. 1436–1462, 2006.
67. E. Pavez and A. Ortega, “Generalized laplacian precision matrix estimation for graph signal processing,” in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pp. 6350–6354, IEEE, 2016.
68. M. Pourahmadi, “Covariance estimation: The glm and regularization perspectives,” *Statistical Science*, pp. 369–387, 2011.
69. S. Epskamp and E. I. Fried, “A tutorial on regularized partial correlation networks.,” *Psychological methods*, 2018.
70. A. Das, A. L. Sampson, C. Lainscsek, L. Muller, W. Lin, J. C. Doyle, S. S. Cash, E. Halgren, and T. J. Sejnowski, “Interpretation of the precision matrix and its application in estimating sparse brain connectivity during sleep spindles from human electrocorticography recordings,” *Neural computation*, vol. 29, no. 3, pp. 603–642, 2017.
71. X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst, “Learning laplacian matrix in smooth graph signal representations,” *IEEE Transactions on Signal Processing*, vol. 64, no. 23, pp. 6160–6173, 2016.
72. C.-J. Hsieh, “Sparse inverse covariance estimation for a million variables,” 2014.
73. X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst, “Learning graphs from signal observations under smoothness prior.” June. 2015 [online]. Available: http://arXiv.org/abs/1406.7842.
74. M. Slawski and M. Hein, “Estimation of positive definite m-matrices and structure learning for attractive gaussian markov random fields,” *Linear Algebra and its Applications*, vol. 473, pp. 145–179, 2015.
75. S. Ubaru, J. Chen, and Y. Saad, “Fast estimation of tr(f(a)) via stochastic lanczos quadrature,” *SIAM Journal on Matrix Analysis and Applications*, vol. 38, no. 4, pp. 1075–1099, 2017.
76. T. S. Caetano, J. J. McAuley, L. Cheng, Q. V. Le, and A. J. Smola, “Learning graph matching,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, no. 6, pp. 1048–1058, 2009.
77. D. Thanou, D. I. Shuman, and P. Frossard, “Learning parametric dictionaries for signals on graphs,” *IEEE Trans. Signal Process.*, vol. 62, no. 15, pp. 3849–3862, 2014.
78. E. Camponogara and L. F. Nazari, “Models and algorithms for optimal piecewise-linear function approximation,” *Mathematical Problems in Engineering*, vol. 2015, 2015.
79. T. Zhao, H. Liu, K. Roeder, J. Lafferty, and L. Wasserman, “The huge package for high-dimensional undirected graph estimation in r,” *Journal of Machine Learning Research*, vol. 13, no. Apr, pp. 1059–1062, 2012.
80. Y. Yankelevsky and M. Elad, “Dual graph regularized dictionary learning,” *IEEE Transactions on Signal and Information Processing over Networks*, vol. 2, no. 4, pp. 611–624, 2016.
81. M. Zheng, J. Bu, C. Chen, C. Wang, L. Zhang, G. Qiu, and D. Cai, “Graph regularized sparse coding for image representation,” *IEEE Trans. Image Process.*, vol. 20, no. 5, pp. 1327–1336, 2011.
82. S. Segarra, A. G. Marques, G. Mateos, and A. Ribeiro, “Blind identification of graph filters with multiple sparse inputs.,” in *ICASSP*, pp. 4099–4103, 2016.
83. R. Rustamov and L. J. Guibas, “Wavelets on graphs via deep learning,” in *Advances in neural information processing systems*, pp. 998–1006, 2013.
84. L. Stanković, M. Daković, and T. Thayaparan, *Time-frequency signal analysis with applications.* Artech house, 2014.
85. I. Jestrović, J. L. Coyle, and E. Sejdić, “A fast algorithm for vertex-frequency representations of signals on graphs,” *Signal processing*, vol. 131, pp. 483–491, 2017.
86. L. Stanković, M. Daković, and E. Sejdić, “Vertex-frequency analysis: A way to localize graph spectral components [lecture notes],” *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 176–182, 2017.
87. L. Stanković, E. Sejdić, and M. Daković, “Vertex-frequency energy distributions,” *IEEE Signal Processing Letters*, 2017.
88. L. Stanković, E. Sejdić, and M. Daković, “Reduced interference vertex-frequency distributions,” *IEEE Signal Processing Letters*, 2018.

89. J. Lefèvre, D. Germanaud, J. Dubois, F. Rousseau, I. de Macedo Santos, H. Angleys, J.-F. Mangin, P. S. Hüppi, N. Girard, and F. De Guio, "Are developmental trajectories of cortical folding comparable between cross-sectional datasets of fetuses and preterm newborns?," *Cerebral cortex*, vol. 26, no. 7, pp. 3023–3035, 2015.
90. R. M. Rustamov, "Average interpolating wavelets on point clouds and graphs," *arXiv preprint arXiv:1110.2227*, 2011.
91. A. Golbabai and H. Rabiei, "Hybrid shape parameter strategy for the rbf approximation of vibrating systems," *International Journal of Computer Mathematics*, vol. 89, no. 17, pp. 2410–2427, 2012.
92. D. I. Shuman, C. Wiesmeyr, N. Holighaus, and P. Vandergheynst, "Spectrum-adapted tight graph wavelet and vertex-frequency frames," *IEEE Transactions on Signal Processing*, vol. 63, no. 16, pp. 4223–4235, 2015.
93. D. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Appl. Comput. Harmon. Anal.*, vol. 30, no. 2, pp. 129–150, 2011.
94. H. Behjat, N. Leonardi, L. Sörnmo, and D. Van De Ville, "Anatomically-adapted graph wavelets for improved group-level fMRI activation mapping," *NeuroImage*, vol. 123, pp. 185–199, 2015.
95. I. Ram, M. Elad, and I. Cohen, "Redundant wavelets on graphs and high dimensional data clouds," *IEEE Signal Process. Lett.*, vol. 19, no. 5, pp. 291–294, 2012.
96. A. Sakiyama, K. Watanabe, and Y. Tanaka, "Spectral graph wavelets and filter banks with low approximation error," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 2, no. 3, pp. 230–245, 2016.
97. T. Cioaca, B. Dumitrescu, and M.-S. Stupariu, "Graph-based wavelet representation of multi-variate terrain data," in *Computer Graphics Forum*, vol. 35, 1, pp. 44–58, Wiley Online Library, 2016.
98. T. Cioaca, B. Dumitrescu, and M.-S. Stupariu, "Lazy wavelet simplification using scale-dependent dense geometric variability descriptors," *Journal of Control Engineering and Applied Informatics*, vol. 19, no. 1, pp. 15–26, 2017.
99. A. Dal Col, P. Valdivia, F. Petronetto, F. Dias, C. T. Silva, and L. G. Nonato, "Wavelet-based visual analysis of dynamic networks," *IEEE transactions on visualization and computer graphics*, 2017.
100. P. Valdivia, F. Dias, F. Petronetto, C. T. Silva, and L. G. Nonato, "Wavelet-based visualization of time-varying data on graphs," in *Visual Analytics Science and Technology (VAST), 2015 IEEE Conference on*, pp. 1–8, IEEE, 2015.

## More detailed consideration of the presented topics may be found in:

L. Stankovic, D. Mandic, M. Dakovic, M. Brajovic, B. Scalzo-Dees, S. Li, and Anthony G. Constantinides, “Data Analytics on Graphs - Part III: Machine Learning on Graphs, from Graph Topology to Applications,” Foundations and Trends in Machine Learning, Vol. 13: No. 4, 2020, pp. 332-530. http://dx.doi.org/10.1561/2200000078-3. (Artificial Intelligence Q1, SCImago Journal Rank SJR 5.12)LJ.

L. Stankovic, D. Mandic, M. Dakovic, M. Brajovic, B. Scalzo-Dees, and Anthony G. Constantinides, “Data Analytics on Graphs Part II: Signals on Graphs,” Foundations and Trends in Machine Learning, Vol. 13: No. 2-3, 2020, pp. 158-331. http://dx.doi.org/10.1561/2200000078-2 (Artificial Intelligence Q1, SCImago Journal Rank SJR 5.12)

L. Stankovic, D. Mandic, M. Dakovic, M. Brajovic, B. Scalzo-Dees, and Anthony G. Constantinides, “Data Analytics on Graphs – Part I: Graphs and Spectra on Graphs,” Foundations and Trends in Machine Learning, Vol. 13: No. 1, 2020, pp 1-157. http://dx.doi.org/10.1561/2200000078-1. (Artificial Intelligence Q1, SCImago Journal Rank SJR 5.12)

and downloaded from

http://www.tfsa.me/PapersByAuthor.php?id=1

or from

http://www.tfsa.ac.me/PapersByAuthor.php?id=1