2 Node Domain Processing

Before developing a frequency representation for graph signals (Chapter 3), in this chapter we introduce tools to process graph signals in the node (or vertex) domain. Node-centric processing is a natural choice for graph signals, allowing us to process data at each node on the basis of information from its neighbors. In particular, this will be a useful tool to infer local information within a large graph or to split processing across multiple processors. This chapter will continue building on the idea of variation introduced in Chapter 1. We start by introducing several basic definitions. Next, we discuss more formally the notion of graph locality (Section 2.2) and introduce algebraic representations of graphs (Section 2.3) and how these can be used for processing by introducing graph filters (Section 2.4). Finally, we develop a more formal understanding of graph signals based on a chosen fundamental graph operator (Section 2.5).

2.1 Basic Definitions

In this book we focus on **simple** graphs[1] with at most one edge between any two vertices, but we study both undirected and directed graphs.

Definition 2.1 (Graph) A graph $\mathcal{G}(\mathcal{V},\mathcal{E})$ is defined by a set of nodes $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$ and a set of edges $\mathcal{E} = \{e_1, e_2, \ldots, e_M\}$. A directed edge ij in $\mathcal{E}$ goes from node v_j to node v_i (see below for a discussion of this convention).

If edge ij can take any real positive weight a_{ij} then the graph is **weighted**. **Unweighted** graphs have edges with weights all equal to 1. In both cases, if the edge ij does not exist then $a_{ij} = 0$. If ji exists whenever ij exists and $a_{ij} = a_{ji}$ then the graph is **undirected**. Otherwise the graph is a **directed** graph (also called a digraph): ij and ji may both exist, but in general $a_{ij} \neq a_{ji}$.

Directed graph convention For directed graphs we adopt the same convention as in [15]: a_{ij} is non-zero if there exists an edge from v_j to v_i, which will be denoted by an arrow from j to i (see Figure 1.2). We make this choice so that arrows in the graph are consistent with the flow of information as reflected by the corresponding matrix operations (see Section 2.3). If $a_{ij} \neq 0$, but $a_{ji} = 0$, then observations at v_i will directly depend on observations at v_j, but not the other way around. In the example of Figure 1.2,

[1] Hypergraphs are defined as graphs where there can be multiple edges between any two nodes.

the traffic at node B directly depends on the traffic at node A, but the link between the traffic at A and the traffic at B is not direct. Therefore, on the graph there is an arrow from A to B but not from B to A.

Paths and cycles Paths and cycles (Section 1.3) are important graphs that allow us to make a link with conventional signal processing. A path is a simple graph where two nodes are adjacent if and only if they are consecutive in a list (see Figure 1.8 for an example). A cycle is a simple graph with an equal number of nodes and edges, and where two nodes are adjacent if and only if they appear consecutively along a circle. Alternatively, we can think of a cycle graph as a path graph with N nodes, to which an edge from v_N to v_1 has been added (see Figure 1.9).

Complement graph Given an unweighted graph $\mathcal{G}$, its complement $\mathcal{G}^c$ is a graph with the same nodes but with connections only between nodes that were not connected in the original graph; this means that a signal with low variation in $\mathcal{G}$ will have high variation in $\mathcal{G}^c$ (see Chapter 3).

DEFINITION 2.2 (COMPLEMENT) The **complement** of an unweighted graph $\mathcal{G}$ without self-loops is a graph $\mathcal{G}^c$ with same node set $\mathcal{V}$ but with edge set $\mathcal{E}^c$, the complement of $\mathcal{E}$ in the set of possible edges, i.e., an edge $ij \in \mathcal{E}^c$ if and only if $ij \notin \mathcal{E}$. The set of all possible edges corresponds to all the edges in a **complete** graph.

Self-loops In some cases we use graphs with self-loops or node weights, i.e., edges going from v_i to v_i. Edge weights quantify the similarity between nodes. Since nodes should be maximally similar to themselves, self-loops can be interpreted by considering their weight *relative to the weight of the other edges* (see Section 3.2.2).

Node neighborhood We will define locality in a graph (Section 2.2.2) on the basis of how the nodes are connected. To do so we define the neighborhood of a node.

DEFINITION 2.3 (NODE NEIGHBORHOOD) In an undirected graph, if v_i and v_j are endpoints of an edge, we say these nodes are one-hop neighbors or simply neighbors. We denote by $\mathcal{N}(i)$ the **set of neighbors** of v_i and, by the definition of an undirected graph, if $v_j \in \mathcal{N}(i)$ then $v_i \in \mathcal{N}(j)$.

In a directed graph, we define **in-neighbors** and **out-neighbors**. If ij is a directed edge from v_j to v_i then v_i is an out-neighbor of v_j, while v_j is an in-neighbor of v_i.

In Figure 1.2, A is an in-neighbor of B, while B is an out-neighbor of A.

Node degree, regularity and sparsity Graphs can be characterized by the number of neighbors that each node has (for an unweighted undirected graph this is the same as the node degree), by how much this quantity changes between nodes (the regularity of the graph), and by how it relates to the number of nodes (the density or sparsity).

Definition 2.4 (Node degree) The **degree** d_i of node v_i is the total weight of the edges connecting to this node. For an undirected graph this is

$$d_i = \sum_j a_{ij}.$$

For unweighted undirected graphs, d_i is the number of neighbors of node i. For a directed graph we can define **in-degrees** and **out-degrees**:

$$d_i^{\text{in}} = \sum_{j=1}^{N} a_{ij}, \qquad d_i^{\text{out}} = \sum_{j=1}^{N} a_{ji},$$

which correspond to the weights of all the edges that end or that start at i, respectively.

In a directed graph, a node i with $d_i^{\text{in}} = 0$ and $d_i^{\text{out}} \neq 0$ is a **source**, while $d_i^{\text{in}} \neq 0$ and $d_i^{\text{out}} = 0$ corresponds to a **sink**.

A graph is **regular** if all its nodes have the same degree. The cycle graph in Figure 1.9 is exactly regular, with all nodes having degree 2, while the path graph in Figure 1.8 is nearly regular (two of its nodes have degree 1, the others have degree 2). Likewise the grid graph Figure 1.10 is not regular; most nodes have degree 4, while the nodes along the sides and at the corners have degrees 3 and 2, respectively. None of the other graphs from Chapter 1 is exactly regular. A graph with N nodes is **dense** if the number of edges for most nodes is close to N, and **sparse** if the number of edges for any node is much smaller than N.

Most graphs encountered in practice are not exactly regular. In this book, we will often use the terms regularity and sparsity qualitatively. For example, we may say that a graph is more regular than another if it shows less variation in its degree distribution. Similarly, if a graph is "nearly regular," its corresponding frequency definitions are more likely to resemble those used for conventional signals.

Subgraphs Subgraphs of a given graph contain a subset of nodes and edges and are important in the context of graph signal processing.

Definition 2.5 (Subgraph) Given a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ a subgraph $\mathcal{G}_1(\mathcal{V}_1, \mathcal{E}_1)$ of $\mathcal{G}$ is such that $\mathcal{V}_1 \subset \mathcal{V}$ and $\mathcal{E}_1 \subset \mathcal{E}$, that is, if $a, b \in \mathcal{V}_1$ and $ab \in \mathcal{E}_1$ then we must have that $ab \in \mathcal{E}$.

There are several scenarios where using a subgraph can be preferable to using the original graph. A subgraph that contains all the nodes in the original graph but only some of the edges leads to lower-complexity graph operations, and may have favorable properties (e.g., it may be bipartite). A graph with fewer nodes and edges can be used to replace the original graph signal with a lower-resolution approximation, as part of a multiresolution representation. This can be particularly useful for datasets such as 3D point clouds, where the nodes can number in the millions.

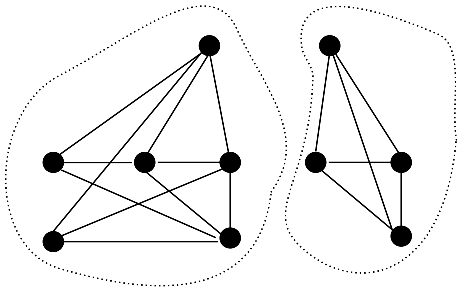


Figure 2.1 This graph consists of two components that are each connected. The corresponding subgraphs are not connected with each other.

Graph signals We have defined graph signals as data associated with a graph, e.g., measurements in a sensor network or information associated with users in an online social network (see examples in Chapter 1). More formally:

Definition 2.6 (Graph signal) A graph signal $\mathbf{x}$ is a real vector, $\mathbf{x} \in \mathbb{R}^N$, where the entry $x(v)$, $\forall v \in \mathcal{V}$, is the real scalar associated with node v. Unless otherwise stated we will consider only real graph signals, but extensions to complex and vector valued signals are also possible.

Recall that a graph signal, $\mathbf{x}$, can only be interpreted with respect to a specific graph (see Section 1.1.2). Therefore **signal smoothness is related to graph locality** (see Section 2.2), which depends on the graph topology. In particular, the same signal $\mathbf{x}$ will have different interpretations on a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ and on a subgraph $\mathcal{G}_1(\mathcal{V}, \mathcal{E}_1)$ of $\mathcal{G}$, where some edges are no longer included.

Box 2.1 Node indexing does not affect graph signal processing

The indexing of nodes on a graph is not important, as all the relevant information is captured by the connections between nodes (i.e., the set of edges). If v_i and v_j are connected and their indices are changed to v'_i and v'_j, respectively, then the new edge set $\mathcal{E}'$ will contain $v'_i v'_j$. Likewise, given a matrix representation for a graph (as described in Section 2.3), changing the indices corresponds to a permutation of the rows and columns of the matrix.

Connected graphs An *undirected* graph is **connected** if any node can be reached through a path from any other node. For *directed* graphs we consider two definitions. A graph is **strongly connected** if there is at least one directed path from any node to any other node; the path can have multiple hops. Note that graphs containing sinks or sources cannot be strongly connected, since no node is reachable from a sink, and a source cannot be reached from any node. A directed graph is **weakly connected** if we can build a connected undirected graph with the same nodes and where each edge connects two nodes if at least one directed edge existed between those two nodes in the original graph. Note that a strongly connected graph is always weakly connected, but the reverse is not true.

Connected components From a graph signal processing perspective, a graph with two connected components (i.e., two components that are internally connected but not connected to each other, as in Figure 2.1) can be viewed as two separate or independent graphs.

Box 2.2 A graph with two connected components is equivalent to two independent graphs

Without loss of generality we can assume that all graphs are connected. This is so because, in practice, a graph with two separately connected components that have no edges linking the components can be treated as two independent graphs. To see why, note that if a graph is disconnected it can be divided into two sets of nodes $\mathcal{V}_1$ and $\mathcal{V}_2$, with no edges connecting those two sets, as shown in the example of Figure 2.1. Since our definitions of variation and smoothness are based on signal changes *across edges*, it is clear that differences between values in $\mathcal{V}_1$ and $\mathcal{V}_2$ do not affect variation. Examples of disconnected graphs are as follows:

- two sets of sensors, $\mathcal{V}_1$, $\mathcal{V}_2$, that are sufficiently far apart that data measured by sensors in $\mathcal{V}_1$ is not correlated with measurements in $\mathcal{V}_2$;
- a social network, where no user in $\mathcal{V}_1$ has a connection to any user in $\mathcal{V}_2$;
- graphs learned from data under probabilistic models (see Section 6.3 and Box 6.4) where the lack of connection between nodes can be interpreted as conditional independence.

If a directed graph is not weakly connected then each of the connected components can be considered as a separate graph, following the arguments laid out in Box 2.2. However, if a graph is weakly connected this does not guarantee that a meaningful interpretation of graph signals exists. For example, the signal value of a source node is not affected by any other node, since there are no incoming edges, while the signal at a sink node has no effect on any other nodes, since there are no outgoing edges.

Keeping in mind that strong connectedness is indeed a much stricter condition, which may not be easily met by graphs of interest, the question of interpreting graph signals and their frequencies for the directed-graph case remains open. Indeed, in Section 2.5.4 and Example 2.9 we will see how graphs that are not strongly connected can lead to graph operators for which some important properties do not hold.

2.2 Locality

We motivate the importance of "local" processing (Section 2.2.1), define locality in the context of graphs (Section 2.2.2), and introduce elementary local operations (Section 2.2.3) that will serve as the basis for more complex local operations. Efficient distributed processing typically requires partitioning a graph so that each set of nodes is processed separately. We discuss two graph partitioning problems. *Identifying clusters and cuts* (Section 2.2.4) aims at grouping together nodes that are close. In contrast, *sam-*

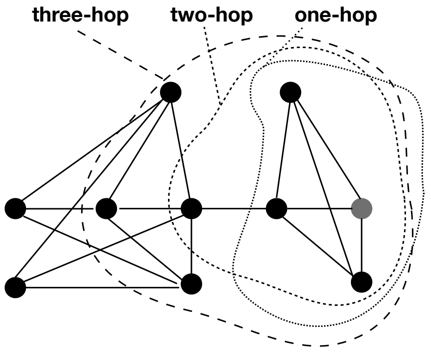


Figure 2.2 Example of locality on a graph. Each set contains neighbors of the gray node on the right. The one-hop, two-hop and three-hop neighborhoods of that node are shown.

pling and coloring (Section 2.2.5) assigns different labels to neighboring nodes, so that no connected nodes share the same label.

2.2.1 Importance of Locality

Localized methods are widely used for processing conventional signals such as audio, images or video. In audio coding, consecutive samples are grouped into **windows** and processed separately, so that the coding parameters adapt to local audio content, leading to increased coding efficiency. Similarly, most image and video coding systems divide images into non-overlapping **blocks**, so that each block (e.g., 8×8 pixels) is compressed separately, adapting to local image properties. Local processing is also useful for graph signals, since there are differences in behavior across the graph (e.g., separate areas monitored by a sensor network or different social groups in a social network).

A further benefit of local processing is computational efficiency. For audio processing, we can segment the signal into non-overlapping windows and encode the signal "on the fly," without having to wait for all of it to be available. Local operations make it easy to parallelize the processing: a large image can be divided into several sub-images, each of which can be assigned to a different processor. Similarly, a large graph can be split into subgraphs so that processing can be accomplished via distributed computations across multiple processors.

2.2.2 What Do We Mean by Local Graph Processing?

The notion of proximity can be generalized by extending Definition 2.3, corresponding to a one-hop neighborhood, and introducing the idea of a k-hop neighborhood.

Definition 2.7 (k-hop neighborhood) Given a node v_i, its k-hop neighborhood, $\mathcal{N}_k(i)$, is the set of all nodes that are part of a path with no more than k edges starting or ending at v_i.

In a k-hop neighborhood (Figure 2.2), for small enough k, "local" processing is possible, involving only a node and its closest neighbors. For example, in a social network, a two-hop neighborhood represents an extended circle of contacts, those we know directly and those to whom we could be introduced via one of our direct contacts.

When is a k-hop neighborhood truly local? To put locality in perspective, we can ask how small k should be for processing on a k-hop neighborhood to be considered local. This can be done by comparing the number of k-hop neighbors $|\mathcal{N}_k(i)|$ of node i with the size of the graph. If $|\mathcal{N}_k(i)| \ll N$ then processing in a k-hop neighborhood of i can be considered to be local. Alternative methods to quantify locality require defining the distance between nodes.

For an undirected and unweighted graph, the geodesic distance between two nodes is the length, in number of hops, of the shortest path between the two nodes. This idea can be extended to weighted graphs by assuming that the edge weights are a measure of node similarity; distance is then obtained as the reciprocal of similarity.[2] Thus, we can define the geodesic distance between any two nodes i and j in a connected graph as follows.

Definition 2.8 (Geodesic distance) If i, j are connected, $a_{ij} \neq 0$, define distance as $d_{ij} = 1/a_{ij}$, with greater similarity implying shorter distance. If i, j are not directly connected, let $p_{ij} = \{i, j_1, j_2, \ldots, j_k\}$ be a path connecting i and j. Then the geodesic distance between i and j is the minimum over all possible paths:

$$d(i, j) = \min_{p(i,j)} \left(\frac{1}{a_{ij_1}} + \frac{1}{a_{j_1 j_2}} + \cdots + \frac{1}{a_{j_k j}} \right).$$

In an unweighted graph $a_{ij} \in \{0, 1\}$, so that $d(i, j)$ is indeed the length (in number of hops) of the shortest path between i and j.

Note that geodesic distances are dependent on the graph structure. First, as illustrated in Example 1.3 the geodesic distance may not be symmetric, if the graph is directed. Second, the geodesic distance changes if the graph structure changes (e.g., the edge weights change). Thus if any of the edge weights along $p(i, j)$ does change, the distance following that path will change and $p(i, j)$ may no longer be the shortest path.

Definition 2.9 (Radius and diameter of a graph) For an arbitrary node v_i in a connected graph $\mathcal{G}$, define $D_i = \max_j d_{ij}$, the distance from v_i to the node farthest from v_i. In an unweighted and undirected graph, D_i is the maximum number of hops that can be taken away from v_i. The diameter and radius of the graph, $d(\mathcal{G})$ and $r(\mathcal{G})$, respectively, are:

$$d(\mathcal{G}) = \max_i D_i \quad \text{and} \quad r(\mathcal{G}) = \min_i D_i.$$

Intuitively, we expect $d(\mathcal{G})$ and $r(\mathcal{G})$ to be related to the graph's regularity. For instance, in the regular cycle graph of Figure 1.9, it is easy to see that $r(\mathcal{G}) = d(\mathcal{G}) = 4$.

[2] In this setting, similarity is given and distance is derived from it, but in other cases the distance between nodes is already well defined (e.g., when the nodes correspond to sensors deployed in the environment).

For an unweighted graph, radius and diameter are defined in terms of numbers of hops, so that we can define locality based on the following remark.

Remark 2.1 A k-hop localized node domain operation can only be considered truly “local” if k is significantly smaller than $r(\mathcal{G})$.

As an example, consider carrying out processing on a k-hop neighborhood for $k \geq r(\mathcal{G})$. This would imply that, for at least some nodes in $\mathcal{G}$, a k-hop operation would include **all** the nodes, which can hardly be described as local.

For a weighted graph it may be sufficient to compute the radius or diameter of the corresponding unweighted graph, i.e., the graph with the same topology as the original weighted graph but where all non-zero edge weights have been set to 1. But it is also important to distinguish between locality based strictly on connectivity and the localization implied by similarity. For example, let j be a node in a weighted graph that is exactly k hops away from i. If its similarity, $1/d_{ij}$, is very small, then its effect on a processing operation at i will be low in general. Thus, if all nodes that are k hops away have low similarity, we can view a k-hop operation as being nearly as localized as a $(k-1)$-hop operation. The topic of localization will be studied in more detail in Section 5.1.

2.2.3 Node-Centric Local Processing

Graph processing can be viewed as a series of *node-centric* operations. Consider a simple one-hop averaging operation:

$$y(i) = \frac{1}{d_i} \sum_{j \in \mathcal{N}(i)} a_{ij} x(j), \tag{2.1}$$

where d_i is the degree of node i and $x(i)$, $y(i)$ denote the ith entries of $\mathbf{x}$ and $\mathbf{y}$ respectively. This operation takes an input signal $\mathbf{x}$ and produces an output $\mathbf{y}$, where the ith entry is the average value of the neighbors of node i.

Computing the output $y(i)$ requires access to data from only the one-hop neighbors of i, $\mathcal{N}(i)$, which shows that this computation is *local* on the graph. Moreover, (2.1) is a simple forward operation, where the values in $\mathbf{x}$ are not modified, so that the outputs $y(i)$ can be computed in *any order*, without affecting the result. For example, we can compute $y(i)$ for $i \in 1, \ldots, N$ in increasing index order. Since labels associated with nodes are arbitrary (Box 2.1), this ordering (node 1, then 2, 3 etc.) does not necessarily have advantages for processing. This is in contrast with path or grid graphs (see Figure 1.8 or Figure 1.10), for which some natural ordering of nodes exists, e.g., sequential traversal, passing all nodes, from one end-point to the other end-point for a path graph. The following example illustrates further how computation efficiency is closely related to graph structure.

Example 2.1 For the computation of (2.1) assume that the graph is too large to store the weights a_{ij} and $\mathbf{x}$, $\mathbf{y}$ in the processor memory. Then, the input signal $\mathbf{x}$ has to be

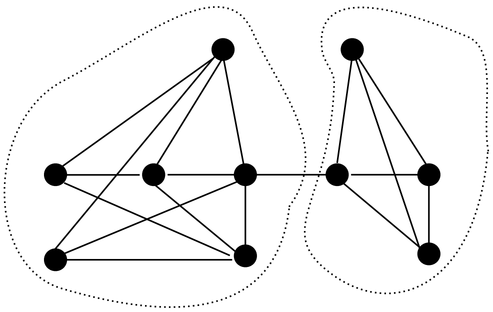


Figure 2.3 Example of clustering. Loosely speaking, for any node in a cluster, the number of connections within the cluster is much greater than connections to nodes outside the cluster. The graph cut is the set of edges which if removed would disconnect the two clusters. In this case only one edge forms the cut.

divided into two smaller signals $\mathbf{x}_1$ and $\mathbf{x}_2$, associated with a partitioning of $\mathcal{V}$ into two sets of nodes, $\mathcal{V}_1$ and $\mathcal{V}_2$, such that $\mathcal{V}_1 \cup \mathcal{V}_2 = \mathcal{V}$ and $\mathcal{V}_1 \cap \mathcal{V}_2 = \emptyset$. Then $\mathbf{x}_1$ and $\mathbf{x}_2$, along with information about the neigborhoods and weights of nodes in each set, are assigned to different processors that can operate in parallel. Assuming that the two subsets are required to have approximately the same size, what would be a good choice for $\mathcal{V}_1$ and $\mathcal{V}_2$?

Solution

Assume that $\mathcal{V}_1$ and $\mathcal{V}_2$ are chosen and the corresponding $\mathbf{x}_1$ and $\mathbf{x}_2$ have been loaded into the respective processors. The processing of $\mathbf{x}_1$ can be completed for all nodes in $\mathcal{V}_1$, *except* for those nodes in $\mathcal{V}_1$ having a neighbor in $\mathcal{V}_2$. If $j \in \mathcal{V}_2$ and $j \in \mathcal{N}(i)$ for $i \in \mathcal{V}_1$ then the two processors have to exchange data to complete the computation. Thus, to reduce this communication overhead, we should choose $\mathcal{V}_1$ and $\mathcal{V}_2$ so as to minimize the number of edges between $\mathcal{V}_1$ and $\mathcal{V}_2$.

A general version of the solution of Example 2.1 requires dividing the original graph into multiple subgraphs, while minimizing the number of edges between nodes assigned to different subgraphs. The problem of dividing a graph into subgraphs in such a way that the connections across subgraphs (graph cuts) meet some conditions is studied next.

2.2.4 Clustering and Graph Cuts

A favorable grouping for distributed processing (Example 2.1) leads to subgraphs that are not strongly connected to each other, as illustrated in Figure 2.3. If $\mathcal{V}_1$ and $\mathcal{V}_2$ are the respective node sets in the two subgraphs, then the nodes in $\mathcal{V}_1$ will have most of their neighbors in $\mathcal{V}_1$, and we can say that the nodes in $\mathcal{V}_1$ (and $\mathcal{V}_2$) form a **cluster**.

More formally, assume that we wish to identify $L \geq 2$ clusters. A specific solution can be described by assigning a label $l \in \{1, \ldots, L\}$ to each node. Denote by $l(i)$ the label assigned to node i. Because the idea of clustering is to group neighbors together, it is natural to define the elementary **cost** of the labeling needed between two nodes as

$$c_{ij} = a_{ij}\, \mathcal{I}(l(i) \neq l(j)),$$

where a_{ij} is the edge weight and $\mathcal{I}(l(i) \neq l(j))$ is an indicator function: $\mathcal{I}(l(i) \neq l(j)) = 1$ if $l(i) \neq l(j)$ (i and j are in different clusters) and $\mathcal{I}(l(i) \neq l(j)) = 0$ if $l(i) = l(j)$ (i and j are in the same cluster). The contribution to the overall cost is zero if two nodes are not connected ($a_{ij} = 0$). This allows us to define the clustering problem.

Definition 2.10 (Clustering) For $L \geq 2$, the L-class clustering problem consists of selecting labels $l(i) \in \{1, \ldots, L\}$ for each node i in a graph in such a way as to minimize the cost,

$$C = \sum_{i \sim j} c_{ij} = \sum_{i \sim j} a_{ij} \mathcal{I}(l(i) \neq l(j)), \tag{2.2}$$

where the summation is over all pairs of connected nodes (denoted by $i \sim j$). Variations of this problem may require the size of the clusters to be similar.

This is an NP-complete problem for which spectral techniques (Section 7.5.2) can provide efficient approximations. Clearly, the cost (2.2) can be made zero for a graph with exactly L connected components: in that case we assign each label to one of the connected components and there will be no connections between nodes having different labels. But, as discussed in Box 2.2, when we have multiple connected components we need to consider each of them as distinct graphs.

Graph cuts The problem of clustering can be related to the problem of finding a graph cut, where instead of grouping nodes we select edges.

Definition 2.11 (Graph cut) A graph cut is a set of edges $\mathcal{E}_c \subset \mathcal{E}$ in a connected graph $\mathcal{G}$ such that: (i) $\mathcal{G}$ is disconnected if all edges in $\mathcal{E}_c$ are removed and (ii) after removing all edges of any set $\mathcal{E}'$ such that $\mathcal{E}' \subset \mathcal{E}_c$, $\mathcal{G}$ remains connected. A graph cut is minimal, for given sizes of the resulting connected components, if it has the minimum number of edges, in an unweighted graph, or the minimum edge weights in a weighted graph.

Finding a **minimum graph cut** divides the graph into two sets of nodes, where the weights of edges connecting nodes in the two sets have minimal weight. We can see that the optimal solution to the problem of Definition 2.10 in the case $L = 2$ would correspond to a minimum cut as well. We will explore further the idea of clustering in Section 7.5.2, where we will consider how it connects with our definitions of graph frequencies. Similarly, finding a **maximum cut** also divides the graph into two sets of nodes, but this time the edge weights connecting those two sets are maximized. While a minimum cut can be used for clustering, a maximum cut is useful for coloring.

2.2.5 Graph Coloring

Graph coloring is a classical problem in graph theory, with interesting connections to graph signal processing. A color is a label associated with a set of nodes. A valid **graph coloring** is an assignment of colors to nodes such that no two connected nodes in the

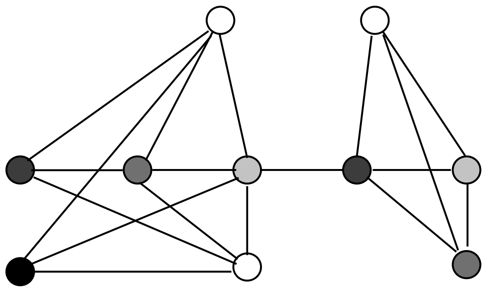


Figure 2.4 Example of coloring. The goal is to assign a color to each node in such a way that no two connected nodes have the same color. In this example, the graph has 10 nodes and is 5-colorable. Each color is represented by a different node shading.

graph are assigned the same color.[3] Note that clustering and coloring seek opposite goals: in clustering we try to minimize differences in labels between connected nodes, while a valid coloring requires labels to be different for connected nodes. A trivial coloring for a graph with N nodes is to assign N different colors, one to each node. Therefore the main challenge in graph coloring is to find the minimum number of colors, also called the **chromatic number**, for which a coloring exists. A k-colorable graph can be defined as follows.

Definition 2.12 (k-Colorable graph) A graph that is **k-colorable** or k-partite can be described using k sets of nodes, $\mathcal{V}_1, \mathcal{V}_2, \ldots, V_k$, where $\mathcal{V}_i \cap \mathcal{V}_j = \emptyset$ and $\mathcal{V}_1 \cup \mathcal{V}_2 \cup \cdots \cup \mathcal{V}_k = \mathcal{V}$ for $i \neq j$ and where all edges e are such that $e = v_{k_i} v_{k_j}$, given that v_{k_i} and v_{k_j} are in $\mathcal{V}_i$ and $\mathcal{V}_j$, respectively, for $i \neq j$.

Figure 2.4 shows a coloring (with five colors) for the graph of Figure 2.3. Such bipartite graphs ($k = 2$) are of particular interest for graph signal processing and indeed for conventional signal processing as well (path graphs are always bipartite, but cycle graphs are only bipartite if they have an even number of nodes). The definition of a bipartite graph follows directly from Definition 2.12.

Definition 2.13 (Bipartite graph) A **bipartite** or 2-colorable graph has two sets of nodes, $\mathcal{V}_1, \mathcal{V}_2$, where $\mathcal{V}_1 \cap \mathcal{V}_2 = \emptyset$ and $\mathcal{V}_1 \cup \mathcal{V}_2 = \mathcal{V}$ and any edge e can be written as $e = v_1 v_2$ or $e = v_2 v_1$, where $v_1 \in \mathcal{V}_1$ and $v_2 \in \mathcal{V}_2$.

Note that the four nodes forming the right-hand cluster in Figure 2.4 are all connected with each other, i.e., they form a **clique**. Because of this, four different colors are needed for those four nodes, so the chromatic number is 4. Similarly a complete graph with N nodes will have chromatic number N. More generally, since coloring has to avoid assigning the same color to two neighbors, the number of colors for a graph with N nodes will depend on the number of edges.

Trees Path graphs are bipartite and do not have any cycles. More generally, graphs that have no cycles are trees. Any node on a tree can be chosen as the **root** and assigned

[3] The term graph coloring originates from the problem of assigning colors to countries on a map, where for clarity two countries sharing a border should not be assigned the same color.

to the first **level**. Nodes connected to the root are its **descendants**, or **children**, and belong to the second level, and a node reached through a path with K edges is at level $K + 1$. A node at any level with no descendants is called a **leaf** node. Trees where every node has the same number of descendants are called **regular**, and if it is possible to choose a root node such that all leaf nodes are at the same level then the tree is called **balanced**. Note that every tree is bipartite, but every bipartite graph is not a tree, since bipartite graphs can have cycles; however, the cycles can only have even length.

Approximate *k*-partition From previous observations it follows that after removing edges from a graph, the chromatic number of the modified graph can only be the same or smaller. If a k-colorable graph for some prespecified k is needed (e.g., a bipartite graph as in Section 5.6.2) the graph can be modified (keeping all nodes, but removing some edges) to achieve the desired chromatic number. Clearly, we can always reach this goal by removing a sufficiently large number of edges, but this can lead to a graph that is very different from the original one. Thus, the more relevant (and challenging) problem is to find the best approximation, i.e., the graph having the desired chromatic number, while requiring the removal of the least number of edges or the removal of edges with the least total weight.

To illustrate this point, compare the cases of maximum and minimum cuts, which both divide a graph into two subgraphs and produce a bipartite approximation (where only edges that form the cut are kept, and all other edges are removed). However, the maximum cut would lead to the better approximation, since it selects edges with maximum weight across the two sets, and therefore the sum of edge weights within each set will be minimized. Since those are the edges to be removed in the bipartite approximation, minimizing their weight is a reasonable target. Graph approximations are studied in Chapter 6.

Coloring and signal variation We can use the concept of coloring to gain some insights about graph signal variation. By definition, two nodes that are assigned the same color cannot be immediate neighbors. Consider the bipartite graph of Figure 2.5, with two different signals (Figure 2.5(a) and (b)) defined on the graph. These two signals are similar: both have four positive and three negative values. Notice that the signal in Figure 2.5(b) assigns the same sign to all nodes corresponding to one color. Since the graph is bipartite, all edges connect nodes with values of different signs and the corresponding signal has high variation. In contrast, there are fewer sign changes in the signal of Figure 2.5(a) leading to overall less signal variation.

Coloring and sampling In graph signal sampling (Chapter 4), the goal is to select a subset of nodes (the sampling set) such that the signal obtained from these observed nodes can be used to estimate the signal at unobserved nodes. If we are going to select K out of N nodes it is desirable for these to be "spread out" so as to provide local information from all parts of the graph. This suggests that, after coloring a graph, one should sample all nodes with one color (since these are automatically kept separate). Thus, in the example of Figure 2.5 if three nodes are to be selected, the nodes on the

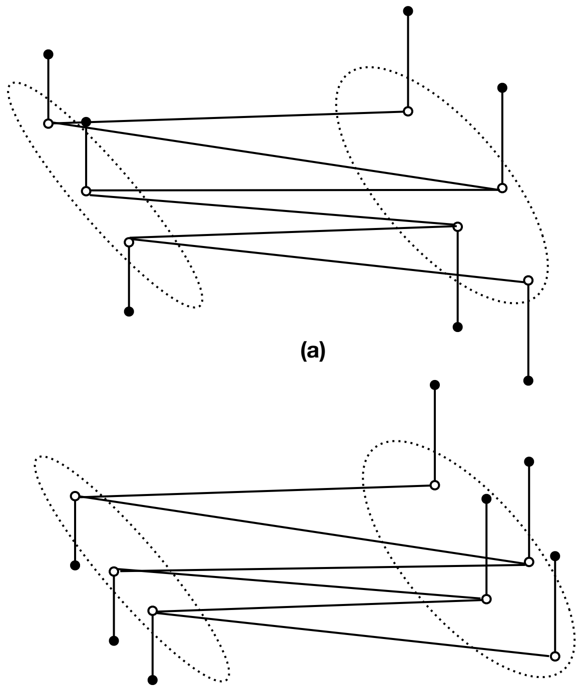


Figure 2.5 Example of bipartite graph with two different graph signals, indicated by the sets of vertical lines. A positive entry is denoted with a vertical line with a black circle above the node. The black circles are below the nodes for negative entries. Notice that both signals have three negative values and four positive values. (a) There is only one **sign change** across the edges of the graph. (b) There are sign changes across all edges.

left (corresponding to one color) would be a good choice: since they are not connected they are less likely to provide redundant information.

2.3 Algebraic Representations of Graphs

As just described, locality can be established on the basis of how nodes are connected. Local linear operations on graphs can be represented more compactly using matrix representations, leading to the definition of graph signal frequencies in Chapter 3.

2.3.1 Adjacency, Incidence and Degree Matrices

For a simple graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with sets of nodes $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$, and edges $\mathcal{E} = \{e_1, e_2, \ldots, e_M\}$, the adjacency matrix, of size $N \times N$, captures all the connectivity and edge weight information.[4]

[4] For directed graphs we follow the edge direction convention described in Section 2.1.

Definition 2.14 (Adjacency matrix) The adjacency matrix $\mathbf{A}$ is an $N \times N$ matrix where the entry a_{ij} corresponding to the ith row and jth column is equal to the weight of the edge from v_j to v_i, for $i \neq j$. If $a_{ii} = 0$ for all i then the graph has no self-loops, while $a_{ij} = a_{ji}$ indicates that the graph is undirected.

As an example, the adjacency matrix for a four-node unweighted path graph similar to that of Figure 1.8 can be written as

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

labeling the nodes from 1 to 4, left to right. As discussed in Box 2.3, different node labeling leads to a permutation of the adjacency matrix (and of the graph signals) and does not affect processing. Thus, if nodes in the undirected path graph were labeled (left to right) $1, 3, 2, 4$, the corresponding adjacency matrix would be

$$\mathbf{A}' = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{A} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

where $\mathbf{A}'$ is obtained by pre- and post-multiplying $\mathbf{A}$ by a permutation matrix representing the change in node indices (nodes 1 and 4 are unchanged, while 2 and 3 are exchanged with each other). We can now restate the ideas of Box 2.1 using permutation matrices.

Box 2.3 Node indexing and permutation

A change in node labeling or indexing can be written as a permutation of $\mathbf{A}$ which leaves unchanged the properties of interest. Let $\mathbf{P}$ be an $N \times N$ **permutation matrix** obtained by reordering the columns of the $N \times N$ identity matrix, $\mathbf{I}$. Given an input signal $\mathbf{x}$, $\mathbf{x}' = \mathbf{P}\mathbf{x}$ is a vector with the same entries as $\mathbf{x}$, but where the entries have been reordered. If the ith column of $\mathbf{I}$ is the jth column of $\mathbf{P}$ then $x'(j) = x(i)$. Then the adjacency matrix $\mathbf{A}'$, given by

$$\mathbf{A}' = \mathbf{P}^{\mathsf{T}}\mathbf{A}\mathbf{P},$$

has the same connections as $\mathbf{A}$ but with indices that are modified by permutation. Assume node a is connected to node b in $\mathbf{A}$ and that, after applying permutations, a and b become a' and b', respectively. Then there will be a connection between a' and b' in $\mathbf{A}'$.

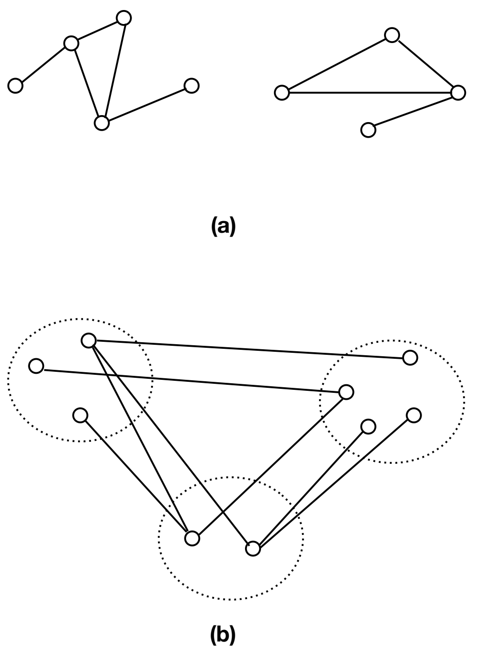


Figure 2.6 Examples of graphs. (a) A graph with two connected components leads to a block diagonal matrix with two non-zero blocks. (b) A 3-colorable graph leads to a matrix with three zero blocks along the diagonal.

Directed graphs For an unweighted directed (left to right) path graph with four nodes, the corresponding adjacency matrix is

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \tag{2.3}$$

so that multiplying a signal by **A** corresponds to *shifting* its entries between nodes along the direction of the edge arrows: if the input signal is $\mathbf{x} = [1, 0, 0, 0]^{\mathsf{T}}$, then $\mathbf{Ax} = [0, 1, 0, 0]^{\mathsf{T}}$.

Block-structured adjacency matrices Relabeling can be used to simplify the notation, in particular if a suitable permutation allows us to write adjacency matrices in block form. Two cases of interest are: (i) disconnected graphs, which can be written in block diagonal structure (one block for each connected component) and (ii) k-colorable graphs, which have identically zero diagonal blocks, corresponding to the nodes in each color. These two cases are illustrated in the following example.

Example 2.2 Write down the block form of the adjacency matrices corresponding to the graphs of Figure 2.6(a) and (b).

Solution

The graph in Figure 2.6(a) has two connected components, so its adjacency matrix **A** can be written as

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_2 \end{bmatrix}$$

where $\mathbf{A}_1$ and $\mathbf{A}_2$ are adjacency matrices that can have different sizes and the remaining two entries in **A** are all-zero matrices.

The graph in Figure 2.6(b) is 3-colorable, and its adjacency matrix can be written as

$$\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{A}_{12} & \mathbf{A}_{13} \\ \mathbf{A}_{21} & \mathbf{0} & \mathbf{A}_{23} \\ \mathbf{A}_{31} & \mathbf{A}_{32} & \mathbf{0} \end{bmatrix}$$

Incidence matrix In the adjacency matrix each column (or row) represents all connections of the corresponding node. However, in the **incidence matrix** each column represents one edge, and each row corresponds to a node.

Definition 2.15 (Incidence matrix – directed graph) The incidence matrix **B** of a graph with N nodes and M directed edges is a rectangular $N \times M$ matrix. If the kth edge is $e_k = v_i v_j$, from j to i with weight a_{ij}, then the kth column of **B**, $\mathbf{b}_k$, has only two non-zero entries, $b_{jk} = -\sqrt{a_{ij}}$ and $b_{ik} = \sqrt{a_{ij}}$. Note that by construction each column adds to zero:

$$\mathbf{1}^\mathsf{T}\mathbf{b}_k = 0 \quad \text{so that} \quad \mathbf{1}^\mathsf{T}\mathbf{B} = \mathbf{0} \quad \text{and} \quad \mathbf{B}^\mathsf{T}\mathbf{1} = \mathbf{0}, \tag{2.4}$$

where $\mathbf{1} = [1, \dots, 1]^\mathsf{T}$ is a vector with all entries equal to 1. Each row i contains the square root of the weights of the edges for which v_i is an end-point, where the sign of an entry is a function of the orientation of the edge.

Thus an incidence matrix **B** is such that: (i) each column of **B** represents an edge, with two non-zero entries with equal absolute values, the positive one corresponding to the end-point of the edge and (ii) each row of **B** represents a node, and non-zero row entries correspond to edges going in or out of that node.

Incidence and signal processing The incidence matrix of a directed graph allows us to represent the evolution of a graph signal in terms of **flows** between neighboring nodes: if nodes i and j are connected, a quantity flowing through the edge from j to i is added to i and subtracted from j, or shifted from j to i as in (2.3). Denote by $\mathbf{y} \in \mathbb{R}^M$ a vector of flows, where the kth entry of **y** represents the flow along the kth edge. Then, multiplying **y** by the incidence matrix produces

$$\mathbf{B}\mathbf{y} = \sum_{k=1}^{M} y_k \mathbf{b}_k, \tag{2.5}$$

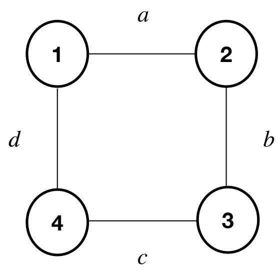


Figure 2.7 Graph for Example 2.3.

where y_k is the kth entry of $\mathbf{y}$ and $\mathbf{b}_k$ is the kth column of $\mathbf{B}$. Thus, if the kth edge goes from node j to node i then $y_k\sqrt{a_{ij}}$ will be added to i and $y_k\sqrt{a_{ij}}$ will be subtracted from j, corresponding to the kth vector in the summation in (2.5). If the original graph signal is $\mathbf{x}$ and a flow $\mathbf{y}$ is applied along the edges, then the resulting graph signal $\mathbf{x}'$ is

$$\mathbf{x}' = \mathbf{x} + \mathbf{B}\mathbf{y} \tag{2.6}$$

so that whatever is subtracted from node j is added to node i and, using (2.4), the sums of the entries of $\mathbf{x}'$ and $\mathbf{x}$ are the same:

$$\mathbf{1}^\mathsf{T}\mathbf{x}' = \mathbf{1}^\mathsf{T}\mathbf{x}.$$

Note that an undirected graph with M edges can be viewed as a directed graph where edges in both directions have equal weights, i.e., $a_{ji} = a_{ij}$, and we can define the corresponding incidence matrix as an $N \times 2M$ matrix where each edge in the graph appears twice, once in each direction. A more compact definition of the incidence matrix for the undirected case includes each edge just once.

Definition 2.16 (Incidence matrix – undirected graph) The incidence matrix $\mathbf{B}$ can also be defined as a rectangular $N \times M$ matrix where row i contains the square root of the weights of edges for which v_i is an end-point. If edge k connects i and j with weight a_{ij} then $b_{ik} = \sqrt{a_{ij}}$ and $b_{jk} = -\sqrt{a_{ij}}$, where the sign can be chosen arbitrarily as long as one of the two entries is negative and the other is positive.

Example 2.3 Write down the incidence matrix $\mathbf{B}$ for the undirected graph of Figure 2.7.

Solution

The label that we associate with each edge is not important, and, from Definition 2.16, we can choose arbitrary signs as long as each column sums to zero. Then we can write

$$\mathbf{B} = \begin{bmatrix} \sqrt{a} & 0 & 0 & \sqrt{d} \\ -\sqrt{a} & \sqrt{b} & 0 & 0 \\ 0 & -\sqrt{b} & \sqrt{c} & 0 \\ 0 & 0 & -\sqrt{c} & -\sqrt{d} \end{bmatrix}.$$

Degree matrix Given a graph with adjacency matrix $\mathbf{A}$ we can define the degree matrix, which will allow us to characterize the graph's regularity.

DEFINITION 2.17 (DEGREE MATRIX) The degree matrix $\mathbf{D}$ of an undirected graph is an $N \times N$ diagonal matrix such that each diagonal term represents the degree (number of edges or sum of edge weights) for the corresponding node, i.e., $d_{ii} = \sum_j a_{ij}$, where a_{ij} is the i, j entry of $\mathbf{A}$. If the graph is directed then we can define in-degree and out-degree matrices by summing only the weights of the incoming and outgoing edges, respectively.

The degree matrix for an undirected graph can be written more compactly as

$$\mathbf{D} = \text{diag}(\mathbf{A1}),$$

where diag($\mathbf{x}$) is a diagonal matrix with the entries $\mathbf{x}$ along the diagonal. For directed graphs we can obtain the *in-degree* and *out-degree* matrices in a similar way:

$$\mathbf{D}_{\text{out}} = \text{diag}(\mathbf{1}^{\mathsf{T}}\mathbf{A}) \quad \text{and} \quad \mathbf{D}_{\text{in}} = \text{diag}(\mathbf{A1}),$$

since the ith row represents the weights of edges ending in node i (Definition 2.4).

As was the case for the incidence matrix, the adjacency and degree matrices define operations on graph signals. That is, we can multiply a graph signal $\mathbf{x}$ by one of those matrices to obtain a new graph signal. Since $\mathbf{D}$ is diagonal, $\mathbf{Dx}$ is simply a scaling of the signal values at each node, while $\mathbf{Ax}$ can be interpreted as a "diffusion operation" where the ith entry of $\mathbf{Ax}$ is obtained as a weighted sum of the entries in $\mathbf{x}$ corresponding to neighbors of i on the graph. General node domain operations constructed with these matrices and the graph Laplacian will be studied in Section 2.4.

Matrix computations and large graphs Since graphs can be large, it is important to note that their algebraic representation is useful conceptually, but also may be practical for the storage of computation. For storage a straightforward representation of an adjacency matrix is as a 2D array where each element of the array corresponds to one edge, or contains a zero if no edge exists. As an alternative, an *adjacency list* representation stores for each node a linked list of all neighboring nodes so that total required storage increases with the number of edges $|E|$, which can be significantly smaller than N^2 [16]. This approach is much more efficient for very large and sparse graphs. Also note that software packages such as `Matlab` include special representations for sparse matrices.

2.3.2 Graph Laplacians

We introduce several types of graph Laplacians for undirected graphs, which will be used to develop the concepts of graph frequency in Chapter 3.

Definition 2.18 (Combinatorial graph Laplacian matrix) The combinatorial graph Laplacian matrix $\mathbf{L}$ of an undirected graph with adjacency and degree matrices $\mathbf{A}$ and $\mathbf{D}$ is the $N \times N$ matrix

$$\mathbf{L} = \mathbf{D} - \mathbf{A}. \tag{2.7}$$

Notice that the Laplacian does not include any additional information that was not already available, since $\mathbf{D}$ is given once $\mathbf{A}$ is known. We can get some insight on how to interpret $\mathbf{L}$ by considering the following example.

Example 2.4 Compute $\mathbf{B}\mathbf{B}^\mathsf{T}$ for the incidence matrix of Example 2.3 and express it in terms of $\mathbf{A}$.

Solution
We can easily compute

$$\mathbf{B}\mathbf{B}^\mathsf{T} = \begin{bmatrix} a+d & -a & 0 & -d \\ -a & a+b & -b & 0 \\ 0 & -b & b+c & -c \\ -d & 0 & -c & c+d \end{bmatrix}.$$

Recall that

$$\mathbf{A} = \begin{bmatrix} 0 & a & 0 & d \\ a & 0 & b & 0 \\ 0 & b & 0 & c \\ d & 0 & c & 0 \end{bmatrix} \text{ and } \mathbf{D} = \begin{bmatrix} a+d & 0 & 0 & 0 \\ 0 & a+b & 0 & 0 \\ 0 & 0 & b+c & 0 \\ 0 & 0 & 0 & c+d \end{bmatrix}$$

and therefore $\mathbf{B}\mathbf{B}^\mathsf{T} = \mathbf{D} - \mathbf{A} = \mathbf{L}$.

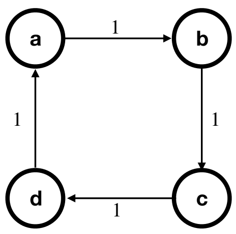
The result of Example 2.4 holds in general.

Proposition 2.1 For any undirected graph, with incidence matrix $\mathbf{B}$, we have

$$\mathbf{L} = \mathbf{D} - \mathbf{A} = \mathbf{B}\mathbf{B}^\mathsf{T}. \tag{2.8}$$

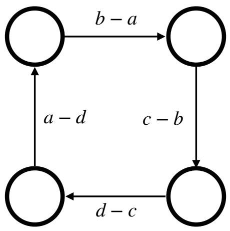
Proof Let $\mathbf{b}_k$ be the kth column of $\mathbf{B}$ which has non-zero entries $-\sqrt{a_{ij}}$ and $\sqrt{a_{ij}}$, corresponding to rows i and j, respectively. Then we can write $\mathbf{B}\mathbf{B}^\mathsf{T}$ as a sum of rank-1 matrices:

$$\mathbf{B}\mathbf{B}^\mathsf{T} = \sum_k \mathbf{b}_k \mathbf{b}_k^\mathsf{T} \quad \text{with} \quad \mathbf{L}_k = \mathbf{b}_k \mathbf{b}_k^\mathsf{T} = \begin{bmatrix} & \vdots & & \vdots & \\ \cdots & a_{ij} & \cdots & -a_{ij} & \cdots \\ & \vdots & & \vdots & \\ \cdots & -a_{ij} & \cdots & a_{ij} & \cdots \\ & \vdots & & \vdots & \end{bmatrix}, \tag{2.9}$$



$$\mathbf{B} = \begin{bmatrix} -1 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix}$$

$$\mathbf{B}^{\mathsf{T}} = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \\ 1 & 0 & 0 & -1 \end{bmatrix}$$



$$\begin{bmatrix} b-a \\ c-b \\ d-c \\ a-d \end{bmatrix} = \mathbf{B}^{\mathsf{T}} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \qquad \begin{bmatrix} 2a-b-d \\ 2b-a-c \\ 2c-b-d \\ 2d-a-c \end{bmatrix} = \mathbf{B} \begin{bmatrix} b-a \\ c-b \\ d-c \\ a-d \end{bmatrix}$$

Figure 2.8 Interpretation of incidence matrix **B**.

where only the non-zero terms of $\mathbf{L}_k$ are shown. Therefore the sum of all the $\mathbf{L}_k$ has diagonal terms equal to the degree of each node and (since there is only one edge between nodes i and j) the off-diagonal terms correspond to the negative of the adjacency matrix, from which the result follows. □

With $\mathbf{L}_k = \mathbf{b}_k \mathbf{b}_k^{\mathsf{T}}$ and denoting $\mathbf{x}_k = \mathbf{L}_k \mathbf{x}$ we can write

$$\mathbf{L}\mathbf{x} = \left(\sum_k \mathbf{L}_k \right) \mathbf{x} = \sum_k \mathbf{x}_k, \tag{2.10}$$

where each entry of $\mathbf{x}_k$ can be written as follows:

$$x_k(i) = a_{ij}\,(x(i) - x(j)), \quad x_k(j) = a_{ij}\,(x(j) - x(i)) \quad \text{and} \quad x_k(n) = 0, \forall n \neq i, j, \tag{2.11}$$

where we see that $x_k(i) + x_k(j) = 0$ and thus the sum of the entries of $\mathbf{x}_k$ is 0 ($\mathbf{1}^{\mathsf{T}}\mathbf{x}_k = 0$). Thus, adding $\mathbf{x}_k = \mathbf{L}_k\mathbf{x}$ can be viewed as the result of a flow along the kth edge, where the amount added to node i is the amount subtracted from j and we have

$$\mathbf{1}^{\mathsf{T}}(\mathbf{x} + \mathbf{L}\mathbf{x}) = \mathbf{1}^{\mathsf{T}}\mathbf{x}. \tag{2.12}$$

This interpretation is valid for both directed and undirected graphs: the operation $\mathbf{B}\mathbf{B}^{\mathsf{T}}$ first creates a flow $\mathbf{y}$ from $\mathbf{x}$ ($\mathbf{y} = \mathbf{B}^{\mathsf{T}}\mathbf{x}$) and then maps it back to produce an output $\mathbf{x}' = \mathbf{B}\mathbf{y}$. See Figure 2.8 for a directed graph example.

2.3.3 Normalized Graph Operators

Consider a weighted graph with adjacency matrix $\mathbf{A}$ and denote the output of the corresponding one-hop operation by $\mathbf{y} = \mathbf{A}\mathbf{x}$. Then the new signal value at node i is

$$y(i) = \sum_{j \in \mathcal{N}(i)} a_{ij} x(j), \tag{2.13}$$

where node i contributes $a_{ij}x(i)$ to node j, while node j contributes $a_{ij}x(j)$ to node i. The ith column of $\mathbf{A}$ provides the weights of all outgoing edges from node i, while the jth row includes the weights of all incoming edges for node j. The one-hop operation (2.13) needs to be modified (normalized) in order to provide a suitable model for some specific systems. We consider two important scenarios where normalization is required.

Graph normalization strategies In a **physical distribution network**, the signal at each node represents some physical quantity (e.g., goods being exchanged between nodes) so that the amounts "sent" through all links, i.e., the **outflows**, have to sum up to the original quantity (i.e., each of the items can be sent through only one of the links). Thus the signal at a given node has to be "distributed" across links. In this case, the total outflow of node i, with initial value $x(i)$, can at most be $x(i)$. Since by definition the sum of all outgoing weights is $d(i)$, where $d(i)$ is the degree of node i, normalized outflows are achieved by computing

$$\mathbf{y} = \mathbf{A}\mathbf{D}^{-1}\mathbf{x},$$

where each column of $\mathbf{A}\mathbf{D}^{-1}$ adds to 1.

Conversely, in a **consensus network**, a node receiving inputs from multiple nodes, i.e., **inflows**, may not simply add them. Instead, it could "summarize" them by computing a consensus or average. This can be achieved with using the one-hop average operation of (2.1), which is written in matrix form as

$$\mathbf{y} = \mathbf{D}^{-1}\mathbf{A}\mathbf{x},$$

where each row of $\mathbf{D}^{-1}\mathbf{A}$ adds to 1.

Normalized Laplacians The same normalizations can be applied when the Laplacian $\mathbf{L} = \mathbf{B}\mathbf{B}^{\mathsf{T}}$ is chosen as the one-hop operator, leading to one-hop operators $\mathbf{L}\mathbf{D}^{-1}$ and $\mathbf{D}^{-1}\mathbf{L}$ for the physical and consensus networks, respectively. These two normalizations can be seen as column or row normalization of the rank-1 matrix $\mathbf{L}_k$ in (2.9), leading to

$$\mathbf{L}_k\mathbf{D}^{-1} = \begin{bmatrix} & \vdots & & \vdots & \\ \cdots & \dfrac{a_{ij}}{d_i} & \cdots & \dfrac{-a_{ij}}{d_j} & \cdots \\ & \vdots & & \vdots & \\ \cdots & \dfrac{-a_{ij}}{d_i} & \cdots & \dfrac{a_{ij}}{d_j} & \cdots \\ & \vdots & & \vdots & \end{bmatrix}, \quad \mathbf{D}^{-1}\mathbf{L}_k = \begin{bmatrix} & \vdots & & \vdots & \\ \cdots & \dfrac{a_{ij}}{d_i} & \cdots & \dfrac{-a_{ij}}{d_i} & \cdots \\ & \vdots & & \vdots & \\ \cdots & \dfrac{-a_{ij}}{d_j} & \cdots & \dfrac{a_{ij}}{d_j} & \cdots \\ & \vdots & & \vdots & \end{bmatrix}. \tag{2.14}$$

The second normalization in (2.14) leads to

$$\mathbf{D}^{-1}\mathbf{B}\mathbf{B}^{\mathsf{T}} = \mathbf{D}^{-1}\mathbf{L} = \mathbf{D}^{-1}(\mathbf{D} - \mathbf{A}) = \mathbf{I} - \mathbf{D}^{-1}\mathbf{A},$$

which we define as the random walk Laplacian:

Definition 2.19 (Random walk graph Laplacian matrix) The random walk graph Laplacian matrix $\mathcal{T}$ is an $N \times N$ matrix

$$\mathcal{T} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{A}. \tag{2.15}$$

Note that these two forms of normalization lead to matrices that are not symmetric, as can be observed in (2.14).

The rows of $\mathcal{Q} = \mathbf{D}^{-1}\mathbf{A}$ and the columns of $\mathcal{P} = \mathbf{A}\mathbf{D}^{-1}$ respectively add to 1, showing that they are row and column **stochastic matrices**, but the same is not true in general for the columns of $\mathcal{Q}$ and the rows of $\mathcal{P}$.

Symmetric normalized Laplacian An alternative approach, which provides normalization while preserving the symmetry in $\mathbf{L}$, is the symmetric normalized Laplacian, where the columns of $\mathbf{B}^{\mathsf{T}}$ and the rows of $\mathbf{B}$ are normalized by the same factor $\mathbf{D}^{-1/2}$.

Definition 2.20 (Symmetric normalized graph Laplacian matrix) The symmetric normalized graph Laplacian matrix $\mathcal{L}$ is an $N \times N$ matrix

$$\mathcal{L} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}. \tag{2.16}$$

Note that since we can write $\mathcal{L} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$, $\mathcal{L}$ would appear to have the form of a graph Laplacian with degree matrix $\mathbf{I}$ and normalized adjacency matrix $\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$. But in fact the degree matrix for this normalized graph does not equal $\mathbf{I}$, that is, in general,

$$\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\mathbf{1} \neq \mathbf{1},$$

and thus, unless all nodes have the same degree, $\mathcal{L}\mathbf{1} \neq \mathbf{0}$, so that nodes in the normalized graph do not have degree equal to 1. Also note that

$$(\mathcal{L})_{i,j} = -a_{ij}\frac{1}{\sqrt{d_i}\sqrt{d_j}}, \quad i \neq j, \tag{2.17}$$

where a_{ij} is the original weight between i and j, and d_i, d_j are the degrees of i and j. We compare $\mathcal{T}$ and $\mathcal{L}$ in the following example.

Example 2.5 Find $\mathcal{T}$ and $\mathcal{L}$ for

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

Solution

We have

$$\mathcal{T} = \begin{bmatrix} 1 & -\frac{1}{2} & 0 & -\frac{1}{2} \\ -\frac{1}{3} & 1 & -\frac{2}{3} & 0 \\ 0 & -\frac{1}{2} & 1 & -\frac{1}{2} \\ -\frac{1}{3} & 0 & -\frac{2}{3} & 1 \end{bmatrix} \text{ and } \mathcal{L} = \begin{bmatrix} 1 & -\frac{1}{\sqrt{6}} & 0 & -\frac{1}{\sqrt{6}} \\ -\frac{1}{\sqrt{6}} & 1 & -\frac{1}{\sqrt{3}} & 0 \\ 0 & -\frac{1}{\sqrt{3}} & 1 & -\frac{1}{\sqrt{3}} \\ -\frac{1}{\sqrt{6}} & 0 & -\frac{1}{\sqrt{3}} & 1 \end{bmatrix},$$

where we can see that $\mathcal{T}$ is not symmetric but has rows that add to zero, while $\mathcal{L}$ is symmetric but its rows do not add up to zero. Thus, while we can create a graph with edge weights obtained with the normalization in Definition 2.20, $\mathcal{L}$ is not the combinatorial Laplacian of that graph.

Matrix representations of graphs are summarized in Table 2.2. In what follows we study how these matrices are used to process graph signals.

2.4 Node Domain Graph Filters

From Definition 2.6, a graph signal is a vector $\mathbf{x} \in \mathbb{R}^N$. In analogy with conventional signal processing, a **linear graph filter** is a linear operator (i.e., a transformation) that can be applied to any signal $\mathbf{x}$ to obtain an output $\mathbf{y} \in \mathbb{R}^N$:

$$\mathbf{y} = \mathbf{H}\mathbf{x}, \tag{2.18}$$

where $\mathbf{H}$ is an $N \times N$ matrix. We starting by defining filtering operations based on the one-hop graph operators of Section 2.3.

2.4.1 Simple One-Hop Filters and Their Interpretation

Adjacency matrix Letting $\mathbf{A}$ be the adjacency matrix of an undirected graph, the ith entry of the filter output $\mathbf{y} = \mathbf{A}\mathbf{x}$, i.e., the output at node i, is

$$y(i) = \sum_{j \in \mathcal{N}(i)} a_{ij} x(j), \tag{2.19}$$

where the sum is over all the neighbors of node i. Therefore we can view $\mathbf{A}$ as a one-hop operator, since the output at any one node depends only on the graph signal value at the neighboring nodes. We can modify the operation in (2.19) and replace the sum of the neighboring values by their average, leading to:

$$y'(i) = \frac{1}{d_i} \sum_{j \in \mathcal{N}(i)} a_{ij} x(j), \tag{2.20}$$

where $d_i = |\mathcal{N}(i)|$ if the graph is unweighted. More compactly,

$$\mathbf{y}' = \mathbf{D}^{-1}\mathbf{A}\mathbf{x} = \mathcal{Q}\mathbf{x},$$

so that the random walk matrix, $\mathcal{Q} = \mathbf{D}^{-1}\mathbf{A}$, can be seen as a one-hop operator that replaces the signal entry at node i by the average of its neighbors' values.

Random walk Laplacian The random walk Laplacian of Definition 2.19, $\mathcal{T} = \mathbf{I} - \mathcal{Q}$, can be interpreted as subtracting from the signal at node i the average of its one-hop neighborhood:

$$y''(i) = x(i) - \frac{1}{d_i} \sum_{j \in \mathcal{N}_i} a_{ij} x(j) = x(i) - y'(i), \tag{2.21}$$

where the second term in the equation, corresponding to $\mathcal{Q}\mathbf{x}$, computes a weighted average in which those nodes that are the most similar (with the largest a_{ij}) are given more weight. Thus, $y''(i)$ can be viewed as a **prediction error**, i.e., if $x(i)$ is close to the neighborhood average then $y''(i)$ will be close to zero. In Chapter 3 we will develop further the link between the properties of graph signals and the output of operators such as $\mathcal{T}$. Signals for which $\mathcal{T}\mathbf{x}$ is close to zero will be deemed smooth or low frequency, while signals for which $|y''(i)|$ is larger will be considered non-smooth or high frequency.

Weighted and unweighted prediction errors If we think of (2.21) as computing a prediction error, it is interesting to note that this expression does not provide any other information about how reliable the prediction is. In particular this prediction error does not depend on the number of neighbors or their weights. Example 2.6 illustrates why this lack of information may be problematic.

Example 2.6 Consider an unweighted and undirected graph and recall that $d_i = |\mathcal{N}(i)|$ (see Definition 2.3). Let two nodes i and j be such that (i) $|\mathcal{N}(i)| = 1$, $|\mathcal{N}(j)| = k > 1$, (ii) $x(i) = x(j) = 2$ and (iii) $\forall l \in \mathcal{N}(i) \cup \mathcal{N}(j)$ we have $x(l) = 1$. Use (2.21) to compute $y''(i)$ and $y''(j)$. For a smooth signal the prediction error (2.21) will be small. Let $\mathbf{x}$ be a noisy version of a signal $\mathbf{x}_0$ which is not observed but is assumed to be smooth. We would like to determine whether the observed values at nodes i and j are equally "reliable." That is, if we take $x(i)$ and $x(j)$ as estimates of $x_0(i)$ and $x_0(j)$, respectively, do we have the same confidence on both estimates? Discuss how the reliability of $x(j)$ may depend on k.

Solution

From (2.21) we can see easily that $y''(i) = y''(j) = 1$. Thus the prediction error is the same in both cases. However, if we assume that the nodes in $\mathcal{N}(i)$ and $\mathcal{N}(j)$ can provide predictions for the signals at i and j, respectively, we might perhaps expect a more reliable estimate when the neighborhood has more nodes.

Nevertheless, since v_i has only one neighbor, while v_j has k, a prediction error of 1 may be considered to be worse for j, since the number of neighbors of v_j is greater. Mathematically, we can define a weighted error $z(i) = |\mathcal{N}(i)|y''(i)$, which then leads to

$$z(i) = 1, \quad z(j) = k,$$

which makes it explicit that the error is actually worse for node j.

As a concrete illustration of the ideas of Example 2.6, consider temperature measurements, where nodes represent sensors and each sensor is connected on a graph to other sensors that are within a certain distance. Then, if v_i and its single neighbor have different measurements, it is possible that the temperature changes relatively quickly in the environment of v_i. On the other hand, if all neighbors of v_j have the same temperature as each other but different from that of v_i, it may be that the measurement $x(j)$ is incorrect and there is a problem with the corresponding sensor. Thus we expect the prediction error to be more informative (e.g., as an indication of a sensor malfunction) if more nodes are used to compute the prediction (i.e., the node has more neighbors).

Combinatorial Laplacian On the basis of Example 2.6 we define a **weighted prediction error** $z(i)$, which gives more weight to errors corresponding to nodes with more neighbors (or, more generally, with higher degrees):

$$z(i) = d_i \left(x(i) - \frac{1}{d_i} \sum_{j \in \mathcal{N}(i)} a_{ij} x(j) \right) \tag{2.22}$$

which can be written in matrix form as

$$\mathbf{z} = \mathbf{D}\mathcal{T}\mathbf{x} = \mathbf{D}(\mathbf{I} - \mathbf{D}^{-1}\mathbf{A})\mathbf{x} = (\mathbf{D} - \mathbf{A})\mathbf{x} = \mathbf{L}\mathbf{x}. \tag{2.23}$$

Thus, the combinatorial graph Laplacian, $\mathbf{L}$ and the random walk Laplacian, $\mathcal{T}$, are one-hop prediction operators using different weights.

Symmetric normalized Laplacian Using the edge weights for the symmetric normalized Laplacian of (2.17) we can write $\mathbf{z}' = \mathcal{L}\mathbf{x}$ as a prediction error at node i:

$$z'(i) = x(i) - \sum_{j \in \mathcal{N}(i)} \frac{a_{ij}}{\sqrt{d_i}\sqrt{d_j}} x(j), \tag{2.24}$$

where, similarly to (2.21), the prediction error does not depend on the size of the neighborhood, but where the weights have been normalized. Also, unlike in (2.21), the **prediction weights** do not add to 1.

2.4.2 Polynomials of One-Hop Operators

From this point forward, denote by $\mathbf{Z}$ a generic one-hop operator, which could be one among those considered so far ($\mathbf{A}$, $\mathbf{D}^{-1}\mathbf{A}$, $\mathcal{T} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{A}$, $\mathcal{L}$ and $\mathbf{L}$). We now make a key observation that will help us interpret more complex operations based on a chosen $\mathbf{Z}$.

Remark 2.2 If $\mathbf{Z}$ is a one-hop graph operator then $\mathbf{Z}^2$ is a two-hop graph operator and, more generally, $\mathbf{Z}^k$ is a k-hop graph operator.

Proof Let $\boldsymbol{\delta}_i$ be a signal that is zero everywhere except at node i. By definition, one-hop operators are such that $\mathbf{Z}\boldsymbol{\delta}_i$ can only have non-zero entries at node i and at nodes $j \in \mathcal{N}(i)$. Therefore we will be able to find some scalars α_j such that

$$\mathbf{Z}\boldsymbol{\delta}_i = \sum_{j \in \mathcal{N}(i)} \alpha_j \boldsymbol{\delta}_j,$$

and then, by linearity, we can write

$$\mathbf{Z}^2\boldsymbol{\delta}_i = \mathbf{Z} \sum_{j \in \mathcal{N}(i)} \alpha_j \boldsymbol{\delta}_j = \sum_{j \in \mathcal{N}(i)} \alpha_j \mathbf{Z}\boldsymbol{\delta}_j,$$

which only has non-zero values in the one-hop neighbors of $j \in \mathcal{N}(i)$; by definition these constitute the two-hop neighborhood of i, $\mathcal{N}_2(i)$. The same argument can be applied iteratively to show that $\mathbf{Z}^k\boldsymbol{\delta}_i$ has non-zero values only in $\mathcal{N}_k(i)$. Then, using superposition, it is easy to see that the output of $\mathbf{Z}^2\mathbf{x}$ at node i depends only on the values at the nodes j in the two-hop neighborhood of i. A value at node $k \notin \mathcal{N}_2(i)$ will not affect the output $\mathbf{Z}^2\mathbf{x}$ at node i. □

More generally, from Remark 2.2 it follows that $p(\mathbf{Z})$, an arbitrary polynomial of $\mathbf{Z}$ of degree K, will produce localized outputs. That is, $p(\mathbf{Z})\boldsymbol{\delta}_i$ will be identically zero for nodes that do not belong to $\mathcal{N}_K(i)$. Note that in our initial definition of a graph filtering operation in (2.18) we simply required the operator $\mathbf{H}$ to be linear, so that any arbitrary $N \times N$ matrix could have been chosen for $\mathbf{H}$. Instead, we now restrict ourselves to filters that can be written as polynomials of $\mathbf{Z}$.

Definition 2.21 (Graph filters) For a given one-hop operator $\mathbf{Z}$, a **graph filter** $\mathbf{H}$ is an $N \times N$ matrix that can be written as a polynomial $p(\mathbf{Z})$ of $\mathbf{Z}$:

$$\mathbf{H} = p(\mathbf{Z}) = \sum_{k=0}^{K} a_k \mathbf{Z}^k, \tag{2.25}$$

where $\mathbf{Z}^0 = \mathbf{I}$ and the scalars a_k are the coefficients of the polynomial.

With this definition, the degree of the polynomial defines the localization of the operator, with the caveat that K-hop localization may lead to a global operation if the radius or the diameter of the graph is small (see Section 2.2.2).

Polynomial graph filters commute While matrix multiplication is not commutative in general, matrices that are polynomials of a given $\mathbf{Z}$ do commute. This property is analogous to that of shift invariance in conventional signal processing (Box 2.4).

Theorem 2.1 (Commutativity of polynomial operators) Let $p(\mathbf{Z})$ be an arbitrary polynomial of $\mathbf{Z}$; then $p(\mathbf{Z})\,\mathbf{Z} = \mathbf{Z}\,p(\mathbf{Z})$, so that $\mathbf{Z}$ and $p(\mathbf{Z})$ commute, and, for any polynomial $q(\mathbf{Z})$

$$p(\mathbf{Z})\,q(\mathbf{Z}) = q(\mathbf{Z})\,p(\mathbf{Z}).$$

Proof Let $p(\mathbf{Z}) = \sum_{i=0}^{p-1} \alpha_i \mathbf{Z}^i$; then

$$\begin{aligned}\mathbf{Z}\, p(\mathbf{Z}) &= \sum_{i=0}^{p-1} \alpha_i \mathbf{Z}^{i+1} = \left(\sum_{i=0}^{p-1} \alpha_i \mathbf{Z}^i\right)\mathbf{Z}\\ &= p(\mathbf{Z})\,\mathbf{Z}\end{aligned}$$

and the commutativity of two polynomials follows directly from this result. □

Box 2.4 Graph operators versus graph shift operators

Focusing on only those graph filters that can be written in the polynomial form of Definition 2.21 is fundamental to the development of GSP. This also leads to strong analogies with conventional signal processing [15]. Specifically, in a linear shift invariant system, we can define an elementary shift or delay operator, z^{-1}, and the z-transforms of a filter $H(z)$ and of an input signal, $X(z)$. Then, the output $Y(z)$ of a linear shift-invariant system can be written as [17]

$$Y(z) = H(z)X(z).$$

The shift-invariance property for conventional discrete-time linear systems is expressed as

$$\underbrace{H(z)\left(z^{-1}X(z)\right)}_{(A)} = \underbrace{z^{-1}\left(H(z)X(z)\right)}_{(B)}$$

which states that output of the system when a shift is applied to the input (A) can also be obtained by shifting the output of the original, non-delayed, system (B). Notice that this implies that

$$H(z)z^{-1} = z^{-1}H(z),$$

which clearly shows the analogy with Theorem 2.1, where $p(\mathbf{Z})$ and $\mathbf{Z}$ commute. On the basis of this analogy, $\mathbf{Z}$ is often called the **graph shift operator**, and the property of Theorem 2.1 is described as **shift invariance** [15]. The correspondence between $\mathbf{Z}$ and z is indeed exact when we consider a directed graph cycle [15].

While this analogy is interesting and insightful, in this book we do not use the term "graph shift operator" to describe $\mathbf{Z}$ and instead call $\mathbf{Z}$ the **fundamental graph operator**, **one-hop graph operator** or just **graph operator** for short. This is simply a difference in names, and in all other respects our definition of $\mathbf{Z}$ is the same as that commonly used in the graph signal processing literature [15, 11].

We choose not to use the term "shift" to avoid potential confusion, given the differences that exist between $\mathbf{Z}$ and z. In particular, for a time-based signal $X(z)$, we can always reverse the delay, i.e., $X(z) = z^{-1}zX(z)$, while in general the graph operators $\mathbf{Z}$ are rarely invertible. Thus if we compute $\mathbf{y} = \mathbf{Z}\mathbf{x}$, we cannot guarantee that this operation can be reversed.

2.5 Graph Operators and Invariant Subspaces

In this section, we develop tools to characterize the output of a graph operator (Definition 2.21) applied to an arbitrary graph signal. We work with polynomials of $\mathbf{Z}$, a one-hop fundamental graph operator (Box 2.4) and make use of the basic linear algebra concepts reviewed in Appendix A.This section addresses two fundamental questions:

- For a graph with N nodes, with $\mathbf{Z}$ of size $N \times N$, how many **degrees of freedom**[5] are there to construct filters in the form of Definition 2.21? Essentially, the answer to this question will tell us what is the maximum degree of a polynomial that has the form of Definition 2.21. We will see that, when we are constructing polynomial operators, the degree can be no greater than N, and is often less than N.
- Can we express $\mathbf{Zx}$ as a function of the outputs $\mathbf{Zu}_i$ for a series of elementary signals $\mathbf{u}_i$? The answer is obviously yes, since $\mathbf{Z}$ is a linear operator and the response can be expressed as a linear combination of responses to elementary signals forming a basis for the space. We can construct **basis sets obtained from invariant subspaces** (where the signals in each subspace E_i are invariant under multiplication by $\mathbf{Z}$: $\mathbf{Zu}_i \in E_i$ if $\mathbf{u}_i \in E_i$). When $\mathbf{Z}$ is diagonalizable each of these subspaces corresponds to one of the N linearly independent eigenvectors.

While the answer to these questions builds on elementary linear algebra concepts (refer to Appendix A for a review), we introduce them step by step so as to highlight their interpretation in the context of graph signal filtering.

2.5.1 Minimal Polynomial of a Vector

As a starting point, consider an arbitrary vector $\mathbf{x} \in \mathbb{R}^N$ and apply the operator $\mathbf{Z}$ successively to this vector. This will produce a series of vectors,

$$\mathbf{x},\ \mathbf{Zx},\ \mathbf{Z}^2\mathbf{x},\ \ldots,\ \mathbf{Z}^{p-1}\mathbf{x}.$$

Are these p vectors linearly independent? The answer depends on both $\mathbf{x}$ and p. First, note that if $p > N$ then these vectors must be linearly dependent: in a space of dimension N, any set of more than N vectors must be linearly dependent.[6]

Next, for a given $\mathbf{x}$, find the smallest p such that $\mathbf{x}, \mathbf{Zx}, \mathbf{Z}^2\mathbf{x}, \ldots, \mathbf{Z}^{p-1}\mathbf{x}, \mathbf{Z}^p\mathbf{x}$ are linearly dependent. For this p, by the definition of linear dependence, we can find $a_0, a_1, \ldots, a_{p-1}$ such that

$$\mathbf{Z}^p\mathbf{x} = \sum_{k=0}^{p-1} a_k \mathbf{Z}^k \mathbf{x}, \tag{2.26}$$

where as before $\mathbf{Z}^0 = \mathbf{I}$. Therefore,

$$p_x(\mathbf{Z})\mathbf{x} = \left(-\sum_{k=0}^{p-1} a_k \mathbf{Z}^k + \mathbf{Z}^p \right) \mathbf{x} = \mathbf{0}, \tag{2.27}$$

[5] By degrees of freedom we mean the number of different parameters that can be selected.
[6] The **span** of $\mathbf{x}, \mathbf{Zx}, \mathbf{Z}^2\mathbf{x}, \ldots, \mathbf{Z}^p\mathbf{x}$ is the order-p Krylov subspace generated by $\mathbf{Z}$ and $\mathbf{x}$.

where $p_x(\mathbf{Z}) = -\sum_{k=0}^{p-1} a_k \mathbf{Z}^k + \mathbf{Z}^p$ is the minimal polynomial of $\mathbf{x}$.

Definition 2.22 (Minimal polynomial of a vector) For a given non-zero graph signal $\mathbf{x}$ and a graph operator $\mathbf{Z}$, the minimal polynomial $p_x(\mathbf{Z})$ of $\mathbf{x}$ is the lowest-degree non-trivial polynomial of $\mathbf{Z}$ such that

$$p_x(\mathbf{Z})\mathbf{x} = \mathbf{0}.$$

The minimal polynomial $p_x(\mathbf{Z})$ allows us to simplify operations on $\mathbf{x}$. Any $p(\mathbf{Z})$ can be written as

$$p(\mathbf{Z}) = q(\mathbf{Z})p_x(\mathbf{Z}) + r(\mathbf{Z}) \tag{2.28}$$

where $q(\mathbf{Z})$ and $r(\mathbf{Z})$ are the quotient and residue polynomials, respectively, and $r(\mathbf{Z})$ has degree less than that of $p_x(\mathbf{Z})$. The polynomials $q(\mathbf{Z})$ and $r(\mathbf{Z})$ can be obtained using long division as shown below in Example 2.7. Then, by the definition of $p_x(\mathbf{Z})$, we have

$$p(\mathbf{Z})\mathbf{x} = q(\mathbf{Z})p_x(\mathbf{Z})\mathbf{x} + r(\mathbf{Z})\mathbf{x} = r(\mathbf{Z})\mathbf{x}.$$

Example 2.7 Let $\mathbf{x}$ be a vector with minimal polynomial $p_x(\mathbf{Z}) = \mathbf{Z}^2 + 2\mathbf{Z} + \mathbf{I}$. Find $q(\mathbf{Z})$ and $r(\mathbf{Z})$ such that $p(\mathbf{Z}) = \mathbf{Z}^4$ can be written as in (2.28): $p(\mathbf{Z}) = q(\mathbf{Z})p_x(\mathbf{Z}) + r(\mathbf{Z})$.

Solution

We can express $\mathbf{Z}^4\mathbf{x}$ as a function of $p_x(\mathbf{Z})$ using long division. In the first step we approximate $p(\mathbf{Z})$ by $\mathbf{Z}^2 p_x(\mathbf{Z})$ to cancel out the highest-degree term in $p(\mathbf{Z})$ and compute the resulting residue,

$$r_1(\mathbf{Z}) = \mathbf{Z}^4 - \mathbf{Z}^2(\mathbf{Z}^2 + 2\mathbf{Z} + \mathbf{I}) = -2\mathbf{Z}^3 - \mathbf{Z}^2;$$

then we approximate $r_1(\mathbf{Z})$ to find

$$r_2(\mathbf{Z}) = (-2\mathbf{Z}^3 - \mathbf{Z}^2) + 2\mathbf{Z}(\mathbf{Z}^2 + 2\mathbf{Z} + \mathbf{I}) = 3\mathbf{Z}^2 + 2\mathbf{Z}$$

and finally

$$r(\mathbf{Z}) = (3\mathbf{Z}^2 + 2\mathbf{Z}) - 3\mathbf{I}(\mathbf{Z}^2 + 2\mathbf{Z} + \mathbf{I}) = -4\mathbf{Z} - 3\mathbf{I}$$

so that we have

$$\mathbf{Z}^4 = (\mathbf{Z}^2 - 2\mathbf{Z} + 3\mathbf{I})(\mathbf{Z}^2 + 2\mathbf{Z} + \mathbf{I}) - 4\mathbf{Z} - 3\mathbf{I},$$

where $q(\mathbf{Z}) = \mathbf{Z}^2 - 2\mathbf{Z} + 3\mathbf{I}$. Given that $p_x(\mathbf{Z})$ is the minimal polynomial of $\mathbf{x}$ we have

$$\mathbf{Z}^4\mathbf{x} = (\mathbf{Z}^2 - 2\mathbf{Z} + 3\mathbf{I})p_x(\mathbf{Z})\mathbf{x} - (4\mathbf{Z} + 3\mathbf{I})\mathbf{x} = -(4\mathbf{Z} + 3\mathbf{I})\mathbf{x}.$$

2.5.2 Eigenvectors and Eigenvalues

Clearly, the lowest degree of $p_x(\mathbf{Z})$ for a non-zero vector $\mathbf{x}$ is $p = 1$. Let $\mathbf{u}$ be a vector having minimal polynomial, $p_u(\mathbf{Z})$, of degree 1. Then, from (2.26), $\mathbf{u}$ and $\mathbf{Zu}$ are linearly dependent and so there exists a scalar λ such that $\mathbf{Zu} = \lambda\mathbf{u}$. Thus $\mathbf{u}$ is an eigenvector of $\mathbf{Z}$, with λ the corresponding eigenvalue.

Definition 2.23 (Eigenvectors and eigenvalues) Let $\mathbf{Z}$ be a square matrix. A vector $\mathbf{u}$ and a scalar λ are an eigenvector and an eigenvalue of $\mathbf{Z}$, respectively, if we have

$$\mathbf{Zu} = \lambda\mathbf{u}.$$

An eigenvalue and a corresponding eigenvector form an eigenpair $(\lambda, \mathbf{u})$.

If $\mathbf{u}$ is an eigenvector, the minimal polynomial of $\mathbf{Z}$ with respect to $\mathbf{u}$ is

$$p_u(\mathbf{Z}) = \mathbf{Z} - \lambda\mathbf{I} \tag{2.29}$$

and multiplication by $\mathbf{Z}$ simply scales $\mathbf{u}$. By linearity, any vector along the direction $\mathbf{u}$ is scaled in the same way, so that $\mathbf{Z}(\alpha\mathbf{u}) = \alpha\mathbf{Zu} = \lambda(\alpha\mathbf{u})$. Then $E_u = \text{span}(\mathbf{u})$ is called an **eigenspace** associated with $\mathbf{Z}$ (see Definition A.1 and Definition A.2). Given $|\lambda|$, the magnitude of the eigenvalue λ, if $|\lambda| > 1$ then all vectors in E_u are amplified by the transformation, while if $|\lambda| < 1$ they are attenuated. When $\lambda = 0$, $\mathbf{Zu} = \mathbf{0}$ and $\mathbf{u}$ belongs to the null space of the transformation $\mathbf{Z}$.

Characteristic polynomial By definition of $p_u(\mathbf{Z})$ in (2.29) we have that $p_u(\mathbf{Z})\mathbf{u} = \mathbf{0}$ and, since $\mathbf{u}$ is assumed to be non-zero, we will need to have $\det(\mathbf{Z} - \lambda\mathbf{I}) = 0$, so that the null space of $\mathbf{Z} - \lambda\mathbf{I}$, $\mathcal{N}(\mathbf{Z} - \lambda\mathbf{I})$, contains the non-zero vector $\mathbf{u}$. To find the eigenvectors $\mathbf{u}$ we need to find all the eigenvalues λ, scalars such that $\mathbf{Z} - \lambda\mathbf{I}$ is a singular matrix, i.e.,

$$\det(\lambda\mathbf{I} - \mathbf{Z}) = 0.$$

This determinant can be written out as a polynomial of the variable λ, leading to the characteristic polynomial of $\mathbf{Z}$:

$$p_c(\lambda) = \lambda^N + c_{N-1}\lambda^{N-1} + \cdots + c_1\lambda + c_0\lambda_0, \tag{2.30}$$

where the c_i are known values that depend on $\mathbf{Z}$ and result from the determinant computation. By the fundamental theorem of algebra, $p_c(\lambda)$ will have N roots in $\mathbb{C}$, the space of complex numbers. Given these roots (real or complex, simple or multiple), we can write

$$p_c(\lambda) = \prod_i (\lambda - \lambda_i)^{k_i}, \tag{2.31}$$

where the roots of this polynomial, $\lambda_1, \lambda_2, \ldots$, are the eigenvalues of $\mathbf{Z}$. From the same theorem, calling k_i the **algebraic multiplicity** of eigenvalue λ_i, we have that

$$\sum_i k_i = N.$$

Invariant subspaces As noted earlier, each subspace E_u corresponding to an eigenvector $\mathbf{u}$ is **invariant** under $\mathbf{Z}$. This idea can be easily extended, so that any subspace of $\mathbb{R}^N$ (or $\mathbb{C}^N$) having a basis formed by a set of eigenvectors of $\mathbf{Z}$ is also invariant. More generally, for any non-zero vector $\mathbf{x}$ we can construct an invariant subspace E_x.

Proposition 2.2 (Invariant subspace for $\mathbf{x}$) For a given vector $\mathbf{x}$ (not necessarily an eigenvector of $\mathbf{Z}$), with minimal polynomial $p_x(\mathbf{Z})$ of degree p, a subspace E_x, defined as

$$E_x = \text{span}(\mathbf{x}, \mathbf{Zx}, \mathbf{Z}^2\mathbf{x}, \dots, \mathbf{Z}^{p-1}\mathbf{x})$$

contains $\mathbf{x}$ and is invariant under multiplication by $\mathbf{Z}$, that is,

$$\forall \mathbf{y} \in E_x, \quad \mathbf{Zy} \in E_x,$$

where we emphasize that $\mathbf{Zy}$ is simply required to be in E_x and in general $\mathbf{Zy} \neq \alpha\mathbf{y}$.

Proof By the definition of the span of a set of vectors (Definition A.2), any $\mathbf{y} \in E_x$ can be written as

$$\mathbf{y} = a_0\mathbf{x} + a_1\mathbf{Zx} + \cdots + a_{p-1}\mathbf{Z}^{p-1}\mathbf{x}, \quad \text{so that} \quad \mathbf{Zy} = a_0\mathbf{Zx} + a_1\mathbf{Z}^2\mathbf{x} + \cdots + a_{p-1}\mathbf{Z}^p\mathbf{x}.$$

However, from (2.26), if $p_x(\mathbf{Z})$ has degree p then $\mathbf{x}, \mathbf{Zx}, \mathbf{Z}^2\mathbf{x}, \dots, \mathbf{Z}^{p-1}\mathbf{x}, \mathbf{Z}^p\mathbf{x}$ are linearly dependent and $a_{p-1}\mathbf{Z}^p\mathbf{x}$ can be written as a function of $\mathbf{x}, \mathbf{Zx}, \mathbf{Z}^2\mathbf{x}, \dots, \mathbf{Z}^{p-1}\mathbf{x}$, so that $\mathbf{Zy} \in E_x$, which proves the space is invariant. □

Since E_x is invariant under multiplication by $\mathbf{Z}$ the following properties hold for any $\mathbf{y} \in E_x$ and any polynomial $p(\mathbf{Z})$: (i) E_x is invariant under multiplication by $p(\mathbf{Z})$, i.e., $p(\mathbf{Z})\mathbf{y} \in E_x$ and (ii) $p(\mathbf{Z})\mathbf{y}$ can be simplified as in Example 2.7 given that[7] $p_x(\mathbf{Z})\mathbf{y} = \mathbf{0}$.

Complex eigenvalues Assume $\mathbf{Z}$ is real and μ is a complex root of its characteristic polynomial with multiplicity 1. Since $\mathbf{Z}$ is real, the characteristic polynomial has real coefficients, so that both μ and μ^* are eigenvalues. If $\mathbf{Zu} = \mu\mathbf{u}$, by conjugating both sides we have $(\mathbf{Zu})^* = \mathbf{Zu}^* = (\mu)^*\mathbf{u}^*$, which shows that $(\mu, \mathbf{u})$ and $(\mu^*, \mathbf{u}^*)$ are both eigenpairs of $\mathbf{Z}$. Thus we have a subspace $E_\mu = \text{span}(\mathbf{u}, \mathbf{u}^*) \subset \mathbb{C}^N$ that is invariant under multiplication by $\mathbf{Z}$, since the basis vectors are eigenvectors of $\mathbf{Z}$.

Note that for vectors in span($\mathbf{u}$) and span($\mathbf{u}^*$) the minimal polynomials are $\mathbf{Z} - \mu\mathbf{I}$ and $\mathbf{Z} - \mu^*\mathbf{I}$, respectively. From this we can see that, for any $\mathbf{z} \in E_\mu$, $p_\mu(\mathbf{Z})\mathbf{z} = \mathbf{0}$ with

$$p_\mu(\mathbf{Z}) = (\mathbf{Z} - \mu\mathbf{I})(\mathbf{Z} - \mu^*\mathbf{I}) = \mathbf{Z}^2 - (\mu + \mu^*)\mathbf{Z} + |\mu|^2\mathbf{I},$$

where $p_\mu(\mathbf{Z})$ is a real polynomial. Thus, we can construct a basis $(\mathbf{z}, \mathbf{Zz})$ for E_μ by choosing $\mathbf{z} \in E_\mu$ such that $\mathbf{z} \notin \text{span}(\mathbf{u}) \cup \text{span}(\mathbf{u}^*)$, where we see that the basis vectors are no longer eigenvectors. Also, from the definition of $p_\mu(\mathbf{Z})$

$$\mathbf{Z}^2\mathbf{v} = (\mu + \mu^*)\mathbf{Zv} - |\mu|^2\mathbf{v}.$$

If we are processing real-valued signals with real-valued polynomials of $\mathbf{Z}$, we can also define a real basis for the space of real vectors in E_μ ($E_\mu \cap \mathbb{R}^N$). This is shown in Box 2.5.

[7] Note that $p_x(\mathbf{Z})\mathbf{y} = \mathbf{0}$ but $p_y(\mathbf{Z}) \neq p_x(\mathbf{Z})$.

Box 2.5 Real invariant spaces for complex eigenvalues

From the above discussion, $(\mathbf{x}, \mathbf{Zx})$ is a real basis for $E_\mu \cap \mathbb{R}^N$ for any non-zero and real $\mathbf{x} \in E_\mu$. Alternative real-valued basis choices are possible. If $\mathbf{x} \in E_\mu$, $\mathbf{x} = \alpha\mathbf{u}+\beta\mathbf{u}^*$ and $\mathbf{x}$ is real we have that $\mathbf{x}^* = \mathbf{x}$ and therefore

$$\alpha\mathbf{u} + \beta\mathbf{u}^* = (\alpha\mathbf{u} + \beta\mathbf{u}^*)^* = \beta^*\mathbf{u} + \alpha^*\mathbf{u}^*,$$

so that $\alpha = \beta^*$. Thus, any real vector $\mathbf{x}$ in E_μ can be written as

$$\mathbf{x} = (\alpha_1 + j\alpha_2)\mathbf{u} + (\alpha_1 - j\alpha_2)\mathbf{u}^* = \alpha_1(\mathbf{u} + \mathbf{u}^*) + \alpha_2(j(\mathbf{u} - \mathbf{u}^*)),$$

where both α_1 and α_2 are real and $\mathbf{v}_1 = \mathbf{u}+\mathbf{u}^*$ and $\mathbf{v}_2 = j(\mathbf{u}-\mathbf{u}^*)$ are real vectors, that form a basis for $E_\mu \cap \mathbb{R}^N$ (where $\cap$ denotes the intersection). Since $\mathbf{u} = \frac{1}{2}(\mathbf{v}_1 - j\mathbf{v}_2)$ and $\mathbf{u}^* = \frac{1}{2}(\mathbf{v}_1 + j\mathbf{v}_2)$, we have

$$\mathbf{Zv}_1 = \mu\mathbf{u} + \mu^*\mathbf{u}^* = \frac{1}{2}\mu(\mathbf{v}_1 - j\mathbf{v}_2) + \mu^*\frac{1}{2}(\mathbf{v}_1 + j\mathbf{v}_2) = \mathrm{Re}(\mu)\mathbf{v}_1 + \mathrm{Im}(\mu)\mathbf{v}_2$$

and similarly

$$\mathbf{Zv}_2 = j(\mu\mathbf{u} - \mu^*\mathbf{u}^*) = \frac{1}{2}\mu(j\mathbf{v}_1 + \mathbf{v}_2) - \mu^*\frac{1}{2}(j\mathbf{v}_1 - \mathbf{v}_2) = \mathrm{Im}(\mu)\mathbf{v}_1 + \mathrm{Re}(\mu)\mathbf{v}_2.$$

Then for any real vector in the space $E_\mu \cap \mathbb{R}^N$ we will have

$$\mathbf{Zx} = \mathbf{Z}(\alpha_1\mathbf{v}_1 + \alpha_2\mathbf{v}_2) = (\alpha_1\mathrm{Re}(\mu) + \alpha_2\mathrm{Im}(\mu))\mathbf{v}_1 + (\alpha_1\mathrm{Im}(\mu) + \alpha_2\mathrm{Re}(\mu))\mathbf{v}_2,$$

showing explicitly that $E_\mu \cap \mathbb{R}^N$ is invariant under multiplication by $\mathbf{Z}$.

2.5.3 Minimal Polynomial of Z and Invariant Subspaces

So far we have defined minimal polynomials of individual vectors $\mathbf{x}$ and of eigenspaces. Now we study the minimal polynomial for *all* vectors in $\mathbb{R}^N$ (or $\mathbb{C}^N$). This will allow us to characterize all polynomial filtering operations for graph signals. We start by defining the Schur decomposition, which allows us to develop a series of invariant subspaces such as those introduced in Proposition 2.2 but with the added advantage that they lead to a representation for any signal in the space.

Theorem 2.2 (Schur decomposition) Any $N \times N$ (complex) matrix $\mathbf{Z}$ can be written as

$$\mathbf{Z} = \mathbf{U}^{\mathsf{H}}\mathbf{TU}, \tag{2.32}$$

where $\mathbf{U}$ is unitary, $\mathbf{U}^{\mathsf{H}}\mathbf{U} = \mathbf{UU}^{\mathsf{H}} = \mathbf{I}$ (the superscript H denotes the Hermitian conjugate) and $\mathbf{T}$ is upper triangular, with diagonal elements that are the eigenvalues of $\mathbf{Z}$. A decomposition can be obtained for any ordering of the eigenvalues, leading to different $\mathbf{U}$ and $\mathbf{T}$ in each case.

Proof Let $\lambda_1, \lambda_2, \ldots, \lambda_N$ be an ordering of the eigenvalues of $\mathbf{Z}$ (note that eigenvalues of multiplicity greater than 1 are repeated). Let $\mathbf{u}_1$ be an eigenvector corresponding to λ_1, $\mathbf{u}_1^{\mathsf{H}}\mathbf{u}_1 = 1$. Choose a unitary matrix using $\mathbf{u}_1$ as its first column so that $\mathbf{U} = (\mathbf{u}_1\ \mathbf{U}_2)$, then

$$\mathbf{U}^{\mathsf{H}}\mathbf{Z}\mathbf{U} = \begin{bmatrix} \mathbf{u}_1^{\mathsf{H}} \\ \mathbf{U}_2^{\mathsf{H}} \end{bmatrix} \mathbf{Z} \begin{bmatrix} \mathbf{u}_1 & \mathbf{U}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{u}_1^{\mathsf{H}}\mathbf{Z}\mathbf{u}_1 & \mathbf{u}_1^{\mathsf{H}}\mathbf{Z}\mathbf{U}_2 \\ \mathbf{U}_2^{\mathsf{H}}\mathbf{Z}\mathbf{u}_1 & \mathbf{U}_2^{\mathsf{H}}\mathbf{Z}\mathbf{U}_2 \end{bmatrix} = \begin{bmatrix} \lambda_1 & \mathbf{u}_1^{\mathsf{H}}\mathbf{Z}\mathbf{U}_2 \\ \mathbf{0} & \mathbf{U}_2^{\mathsf{H}}\mathbf{Z}\mathbf{U}_2 \end{bmatrix},$$

where the last equality uses the facts that $\mathbf{u}_1$ is an eigenvector, so that $\mathbf{u}_1^{\mathsf{H}}\mathbf{Z}\mathbf{u}_1 = \lambda_1\mathbf{u}_1^{\mathsf{H}}\mathbf{u}_1 = \lambda_1$, and that $\mathbf{U}_2^{\mathsf{H}}\mathbf{u}_1 = \mathbf{0}$ because $\mathbf{U}$ is unitary. Next denote $\mathbf{Z}_2 = \mathbf{U}_2^{\mathsf{H}}\mathbf{Z}\mathbf{U}_2$ and observe that $\mathbf{Z}$ and $\mathbf{U}^{\mathsf{H}}\mathbf{Z}\mathbf{U}$ have the same eigenvalues. From this it follows that $\mathbf{Z}_2$ must have eigenvalues $\lambda_2, \lambda_3, \ldots, \lambda_N$, so that we can apply recursively the approach above (define $\mathbf{u}_2$ as an eigenvector, find a unitary matrix, etc.) until we obtain the decomposition (2.32). □

Let $\mathbf{U}$ be a unitary matrix given by the Schur decomposition of $\mathbf{Z}$ in (2.32). This decomposition is not unique: a different $\mathbf{U}$ is obtained for each possible ordering of the eigenvalues[8] of $\mathbf{Z}$. Denote by $\mathbf{u}_i$ the ith column of $\mathbf{U}$ and let $E_i = \text{span}(\mathbf{u}_i)$. Then, because $\mathbf{U}$ is unitary, we have that $\mathbb{C}^N = \bigoplus_{i=1}^N E_i$, i.e., $\mathbb{C}^N$ is the direct sum of the E_i subspaces and the following property holds.

Proposition 2.3 (Invariant subspaces from Schur decomposition) The subspaces $F_k = \bigoplus_{i=1}^k E_i$, $k = 1, \ldots, N$ are invariant under $\mathbf{Z}$, that is, if $\mathbf{x} \in F_k$ then $\mathbf{Z}\mathbf{x} \in F_k$.

Proof Choose an arbitrary $\mathbf{x} \in F_k$. By the definition of F_k that means that $\mathbf{x} = \mathbf{U}^{\mathsf{H}}\mathbf{y}$, where $\mathbf{y}$ is such that its last $N - k$ entries are zero. Then

$$\mathbf{Z}\mathbf{x} = \mathbf{U}^{\mathsf{H}}\mathbf{T}\mathbf{U}\mathbf{U}^{\mathsf{H}}\mathbf{y} = \mathbf{U}^{\mathsf{H}}\mathbf{T}\mathbf{y} = \mathbf{U}^{\mathsf{H}}\mathbf{y}',$$

where the last $N - k$ entries of $\mathbf{y}'$ are also zero, because $\mathbf{T}$ is upper triangular, and thus $\mathbf{U}^{\mathsf{H}}\mathbf{y}' \in F_k$. □

As in Proposition 2.2, the invariance in Proposition 2.3 is not as strong as that associated with eigensubspaces (i.e., there is no scalar α such that $\mathbf{Z}\mathbf{x} = \alpha\mathbf{x}$), but it is important because it applies to any square $\mathbf{Z}$, including a $\mathbf{Z}$ lacking a complete set of eigenvectors.

The Schur decomposition of Theorem 2.2 can be used to identify minimal polynomials for all vectors in the space. First, we write a matrix polynomial for an arbitrary $N \times N$ matrix, $\mathbf{X}$, with the same coefficients as the characteristic polynomial of (2.30):

$$p_c(\mathbf{X}) = \mathbf{X}^N + c_{N-1}\mathbf{X}^{N-1} + \cdots + c_1\mathbf{X} + c_0\mathbf{I}. \tag{2.33}$$

For a given $\mathbf{X}$, $p_c(\mathbf{X})$ is also an $N \times N$ matrix. The Cayley–Hamilton theorem states that $\mathbf{Z}$ is a root of the characteristic polynomial.

[8] In addition, for a given ordering of the eigenvalues, the representation may not be unique if there are eigenvalues of multiplicity greater than 1.

Theorem 2.3 (Cayley–Hamilton) Let $p_c(\lambda)$ be the characteristic polynomial of the matrix $\mathbf{Z}$. For an arbitrary square matrix $\mathbf{X}$ let the corresponding matrix polynomial be $p_c(\mathbf{X})$, as defined in (2.33). Then, we have that $\mathbf{Z}$ is a root of $p_c(\mathbf{X})$, i.e., for **any** $\mathbf{x} \in \mathbb{R}^N$,

$$p_c(\mathbf{Z})\mathbf{x} = \mathbf{0}$$

so that

$$p_c(\mathbf{Z}) = \mathbf{0}.$$

Proof From Theorem 2.2 we can write $\mathbf{Z} = \mathbf{U}^\mathsf{H}\mathbf{T}\mathbf{U}$ with $\mathbf{U}$ unitary and $\mathbf{T}$ upper triangular with λ_i along the diagonals. Since $\mathbf{T}$ and $\mathbf{Z}$ are similar they will have the same characteristic polynomial. To show that $\mathbf{Z}$ is a root of $p_c(\mathbf{X})$, we need to show that $p_c(\mathbf{Z})\mathbf{x} = \mathbf{0}$ for any $\mathbf{x}$. Since

$$p_c(\mathbf{Z})\mathbf{x} = \mathbf{U}^\mathsf{H} p_c(\mathbf{T})\mathbf{U}\mathbf{x},$$

all we need to do is show that $P_c(\mathbf{T}) = \mathbf{0}$. Recall that we can write

$$p_c(\mathbf{T}) = \prod_i (\mathbf{T} - \lambda_i \mathbf{I})^{k_i} \tag{2.34}$$

and note that the entries in the first column of $\mathbf{T} - \lambda_1 \mathbf{I}$ are all zero, so that the first column of the product of matrices in (2.34) is $\mathbf{0}$. Then, the product of the first two matrices in $p_c(\mathbf{T})$ is

$$\begin{bmatrix} 0 & t_{21} & \cdots \\ 0 & \lambda_2 - \lambda_1 & \cdots \\ 0 & 0 & \\ \vdots & \vdots & \end{bmatrix} \begin{bmatrix} 0 & t_{21} & \cdots \\ 0 & \lambda_2 - \lambda_2 & \cdots \\ 0 & 0 & \\ \vdots & \vdots & \end{bmatrix} = \begin{bmatrix} 0 & t_{21} & \cdots \\ 0 & \lambda_2 - \lambda_1 & \cdots \\ 0 & 0 & \\ \vdots & \vdots & \end{bmatrix} \begin{bmatrix} 0 & t_{21} & \cdots \\ 0 & 0 & \cdots \\ 0 & 0 & \\ \vdots & \vdots & \end{bmatrix}$$

which shows that the second column is $\mathbf{0}$. Applying the same reasoning for successive columns shows that they are all $\mathbf{0}$ and thus $p_c(\mathbf{T}) = \mathbf{0}$ and $p_c(\mathbf{Z}) = \mathbf{0}$. □

As a consequence of Theorem 2.3 we can factor $p(\mathbf{Z})$ as follows:

$$p_c(\mathbf{Z}) = \prod_i (\mathbf{Z} - \lambda_i \mathbf{I})^{k_i} = \mathbf{0}, \tag{2.35}$$

where $k_i \geq 1$ is the multiplicity of the root λ_i of $p_c(\lambda)$. Note that Theorem 2.3 implies that there exist $c_0, c_1, \ldots, c_{N-1}$ such that for *any* $\mathbf{x}$

$$\mathbf{Z}^N\mathbf{x} = -\left(\sum_{i=0}^{N-1} c_i \mathbf{Z}^i\right)\mathbf{x}. \tag{2.36}$$

Thus, we can write $\mathbf{Z}^N\mathbf{x}$ as a linear combination of N vectors, $\mathbf{x}$, $\mathbf{Z}\mathbf{x}$, ..., $\mathbf{Z}^{N-1}\mathbf{x}$. From Theorem 2.3 $p_c(\mathbf{Z})\mathbf{x} = \mathbf{0}$ for *any* $\mathbf{x}$. Then we can ask what is the lowest degree (minimal) polynomial for a *specific* $\mathbf{x}$, $p_x(\mathbf{Z})$. The degree of the minimal polynomial $p_x(\mathbf{Z})$ depends on $\mathbf{x}$ and in general it can be lower than the degree of p_c. For example, if $\mathbf{u}$ is an eigenvector of $\mathbf{Z}$ with eigenvalue λ then we will have $p_c(\mathbf{Z})\mathbf{u} = \mathbf{0}$ and $(\mathbf{Z} - \lambda\mathbf{I})\mathbf{u} = \mathbf{0}$, so that $p_u(\mathbf{Z}) = (\mathbf{Z} - \lambda \mathbf{I})$ is the minimal polynomial for $\mathbf{u}$. This leads us to ask the following question: what is the minimal-degree polynomial $p_{\min}(\mathbf{Z})$, for which $p_{\min}(\mathbf{Z})\mathbf{x} = \mathbf{0}$ for *all* $\mathbf{x}$?

Minimal polynomial The minimal polynomial of $\mathbf{Z}$, $p_{\min}(\cdot)$, is the polynomial of minimal degree for which $\mathbf{Z}$ is a root. In other words, for any $\mathbf{x} \in \mathbb{R}^N$ we have that

$$p_{\min}(\mathbf{Z})\mathbf{x} = \left(\mathbf{Z}^p + \sum_{i=0}^{p-1} a_i \mathbf{Z}^i\right)\mathbf{x} = \mathbf{0}.$$

The importance of the minimal polynomial for node domain processing (graph signal filtering) is that it determines the maximum degree of any polynomial operator on $\mathbf{Z}$ (see Box 2.6 below). The minimal polynomial is closely related to the characteristic polynomial.

Proposition 2.4 (Roots of minimal polynomial) The polynomial $p_{\min}(\mathbf{Z})$ divides $p_c(\mathbf{Z})$ without residue and therefore can be factored as follows:

$$p_{\min}(\mathbf{Z}) = \prod_i (\mathbf{Z} - \lambda_i \mathbf{I})^{p_i}, \tag{2.37}$$

where $p_i \leq k_i$ and $\sum p_i = p \leq N$. For each eigenvalue λ_i, k_i is the **algebraic multiplicity** of the eigenvalue.

Proof By definition of the minimal and characteristic polynomials we must have that $p_{\min}(\mathbf{Z}) = \mathbf{0}$ and $p_c(\mathbf{Z}) = \mathbf{0}$. Since $p_{\min}(\mathbf{X})$ corresponds to the minimal polynomial of $\mathbf{Z}$, $p_{\min}(\lambda)$, its degree has to be $p \leq N$, because $p_c(\lambda)$, and thus $p_c(\mathbf{X})$, has degree N by construction. Then, we can use long division as in Example 2.7 to express the characteristic polynomial as a function of the minimal polynomial and a remainder:

$$p_c(\mathbf{X}) = q(\mathbf{X})p_{\min}(\mathbf{X}) + r(\mathbf{X}),$$

where the degree of $r(\mathbf{X})$ has to be less than p. Then, by Theorem 2.3,

$$\mathbf{0} = p_c(\mathbf{Z}) = q(\mathbf{Z})p_{\min}(\mathbf{Z}) + r(\mathbf{Z});$$

but, given that $p_{\min}(\mathbf{Z}) = 0$ and is the minimal-degree polynomial, the polynomial $r(\mathbf{Z})$ must be zero otherwise there would be a non-zero polynomial of lower degree than $p_{\min}$ for which $r(\mathbf{Z}) = \mathbf{0}$. Therefore, we can write

$$p_c(\mathbf{Z}) = q(\mathbf{Z})p_{\min}(\mathbf{Z}),$$

which means that $p_{\min}(\mathbf{Z})$ has the same roots as $p_c(\mathbf{Z})$, and therefore (2.37) follows. □

The main implication of the existence of $p_{\min}$ is that some polynomial operations on the graph can be simplified, which allows us to answer the first question we posed at the beginning of this section: the number of degrees of freedom is determined by the degree of the minimal polynomial.

Box 2.6 Maximum-degree polynomials for a given graph

Let $p(\mathbf{Z})$ be an arbitrary polynomial of $\mathbf{Z}$ of degree greater than that of $p_{\min}(\mathbf{Z})$. Then we can write

$$p(\mathbf{Z}) = q(\mathbf{Z})p_{\min}(\mathbf{Z}) + r(\mathbf{Z}),$$

where $q(\mathbf{Z})$, the quotient, and $r(\mathbf{Z})$, the remainder, are polynomials of $\mathbf{Z}$ that can be obtained by long division (see Example 2.7). Then for any input signal $\mathbf{x}$ it is easy to see that

$$p(\mathbf{Z})\mathbf{x} = (q(\mathbf{Z})p_{\min}(\mathbf{Z}) + r(\mathbf{Z}))\mathbf{x} = r(\mathbf{Z})\mathbf{x}, \tag{2.38}$$

since $p_{\min}(\mathbf{Z}) = \mathbf{0}$. An important consequence of this observation is that the locality of polynomial operations on the graph depends on both the graph's radius and diameter (as discussed in Section 2.4.2) and also on the algebraic characteristics of the one-hop operator (i.e., the degree of its minimal polynomial).

Given $p_c(\mathbf{Z})$, in order to identify $p_{\min}(\mathbf{Z})$ all we need to do is find the powers $p_i \leq k_i$ associated with each of the factors in (2.37) that guarantee that $p_{\min}(\mathbf{Z})\,\mathbf{x} = \mathbf{0}$ for all $\mathbf{x}$. Recall that polynomials of $\mathbf{Z}$ commute (Theorem 2.1) and thus we can write the factors in (2.37) in any order. Note that if the factor $(\mathbf{Z} - \lambda_i\mathbf{I})^{p_i}$ is present in (2.37), any vector $\mathbf{x} \in \mathcal{N}((\mathbf{Z} - \lambda_i\mathbf{I})^{p_i})$, the null space of $(\mathbf{Z} - \lambda_i\mathbf{I})^{p_i}$, will be such that $p(\mathbf{Z})\mathbf{x} = \mathbf{0}$ for any polynomial $p(\mathbf{Z})$ including a term $(\mathbf{Z} - \lambda_i\mathbf{I})^{p_i}$. Thus, the problem of finding the minimal polynomial $p_{\min}(\mathbf{Z})$ is the problem of finding p_i such that we can construct a basis for $\mathbb{C}^N$ where each basis vector is in one of the null spaces $\mathcal{N}((\mathbf{Z} - \lambda_i\mathbf{I})^{p_i})$. The choice of p_i thus depends on the dimension of $\mathcal{N}((\mathbf{Z} - \lambda_i\mathbf{I})^{p_i})$. We consider this next.

2.5.4 Algebraic and Geometric Multiplicities and Minimal Polynomials

We have seen already that the algebraic multiplicity k_i of eigenvalue λ_i is the power of the corresponding factor in the characteristic polynomial of (2.35), i.e., $(\mathbf{Z} - \lambda_i\mathbf{I})^{k_i}$. Since λ_i is an eigenvalue there must be at least one non-zero vector (an eigenvector) in $\mathcal{N}(\mathbf{Z} - \lambda_i\mathbf{I})$. The **geometric multiplicity** m_i of λ_i is the **dimension** of $\mathcal{N}(\mathbf{Z} - \lambda_i\mathbf{I})$, i.e., the maximum size of a set of linearly independent vectors in $\mathcal{N}(\mathbf{Z} - \lambda_i\mathbf{I})$.

Consider a specific case where $(\mathbf{Z} - \lambda_i\mathbf{I})^{k_i}$ is a term in the characteristic polynomial and $(\mathbf{Z} - \lambda_i\mathbf{I})^{p_i}$ is the corresponding term in the minimal polynomial. Denote by E_i the subspace associated with eigenvalue λ_i. As discussed above our goal is to represent any $\mathbf{x}$ as follows:

$$\mathbf{x} = \sum_i \mathbf{x}_i$$

where $\mathbf{x}_i \in E_i = \mathcal{N}((\mathbf{Z} - \lambda_i\mathbf{I})^{p_i})$. For each of the following two cases, depending on whether k_i and m_i are equal or not, we discuss how p_i can be obtained.

Case 1 $m_i = k_i,\ k_i \geq 1$

In this case $E_i = \mathcal{N}(\mathbf{Z} - \lambda_i \mathbf{I})$ has dimension k_i and we are able to express all vectors in E_i in terms of a basis comprising $m_i = k_i$ linearly independent eigenvectors, so that $E_i = \text{span}(\mathbf{u}_{i,1}, \mathbf{u}_{i,2}, \ldots, \mathbf{u}_{i,m_i})$, where each $\mathbf{u}_{i,j}$ is one of the eigenvectors associated with the eigenvalue λ_i. Thus, for any $\mathbf{x} \in E_i$ we have

$$\mathbf{x} = \sum_j \alpha_j \mathbf{u}_{i,j}$$

and as a consequence

$$(\mathbf{Z} - \lambda_i \mathbf{I})\mathbf{x} = \mathbf{0}, \quad \forall \mathbf{x} \in E_i,$$

so that $\mathbf{Z} - \lambda_i \mathbf{I}$ is the minimal polynomial for vectors in E_i, and $p_i = 1$. Moreover

$$\mathbf{Z}\mathbf{x} = \lambda_i \sum_j \alpha_j \mathbf{u}_{i,j} \in E_i,$$

which shows that E_i is **invariant** under multiplication by any polynomial of $\mathbf{Z}$. Denote by $\mathbf{U}_i$ the $N \times m_i$ matrix of the linearly independent eigenvectors that form a basis for E_i and let $\tilde{\mathbf{U}}_i$ be $N \times m_i$ and such that $\tilde{\mathbf{U}}_i^{\mathsf{H}} \mathbf{U}_i = \mathbf{I}_{m_i}$, where $\mathbf{I}_{m_i}$ is the identity matrix of size $m_i \times m_i$. For any $\mathbf{x} \in E_i$ there is a vector $\mathbf{a} = [\alpha_1, \ldots, \alpha_{m_i}]^{\mathsf{T}}$ such that:[9]

$$\mathbf{x} = \mathbf{U}_i \mathbf{a}, \quad \text{where} \quad \mathbf{a} = \tilde{\mathbf{U}}_i^{\mathsf{H}} \mathbf{x},$$

and the graph operator for $\mathbf{x} \in E_i$ can be written as

$$\mathbf{Z}\mathbf{x} = \mathbf{U}_i (\lambda_i \mathbf{I}_{m_i}) \tilde{\mathbf{U}}_i^{\mathsf{H}} \mathbf{x}.$$

If in addition $m_i = k_i$ for *all* i, then $\mathbf{Z}$ is **diagonalizable** and can be written as:

$$\mathbf{Z} = \mathbf{U} \Lambda \mathbf{U}^{-1}$$

where each column of $\mathbf{U}$ is one of N linearly independent eigenvectors and Λ is diagonal, with each diagonal entry an eigenvalue.

Case 2 $m_i < k_i,\ k_i > 1$

In this scenario, $\mathbf{Z}$ is **defective** or **non-diagonalizable**: m_i, the dimension of $\mathcal{N}(\mathbf{Z} - \lambda_i \mathbf{I})$, is less than k_i, the dimension of $\mathcal{N}((\mathbf{Z} - \lambda_i \mathbf{I})^{k_i})$. Note that $\mathbf{x} \in \mathcal{N}(\mathbf{Z} - \lambda_i \mathbf{I})$ implies that $\mathbf{x} \in \mathcal{N}((\mathbf{Z} - \lambda_i \mathbf{I})^2)$, so that $\mathcal{N}(\mathbf{Z} - \lambda_i \mathbf{I}) \subseteq \mathcal{N}((\mathbf{Z} - \lambda_i \mathbf{I})^2)$.

The term in the minimal polynomial $(\mathbf{Z} - \lambda_i \mathbf{I})^{p_i}$ has to set to zero all the vectors in $E_i = \mathcal{N}((\mathbf{Z} - \lambda_i \mathbf{I})^{k_i})$. Thus we are looking for the minimal p_i such that $\mathcal{N}((\mathbf{Z} - \lambda_i \mathbf{I})^{p_i}) = \mathcal{N}((\mathbf{Z} - \lambda_i \mathbf{I})^{k_i})$. Then $E_i = \text{span}(\mathbf{u}_{i,1}, \mathbf{u}_{i,2}, \ldots, \mathbf{u}_{i,k_i})$, where the linearly independent vectors $\mathbf{u}_{i,j}$ span $\mathcal{N}((\mathbf{Z} - \lambda_i \mathbf{I})^{p_i}) = \mathcal{N}((\mathbf{Z} - \lambda_i \mathbf{I})^{k_i})$, so that

$$(\mathbf{Z} - \lambda_i \mathbf{I})^{k_i} \mathbf{x} = (\mathbf{Z} - \lambda_i \mathbf{I})^{p_i} \mathbf{x} = \mathbf{0}, \quad \forall \mathbf{x} \in E_i. \tag{2.39}$$

Note that in Case 1 the basis for E_i was formed with linearly independent eigenvectors

[9] As discussed in Appendix A, $\tilde{\mathbf{U}}_i$ is the dual basis of $\mathbf{U}_i$, which always exists and is unique because the column vectors in $\mathbf{U}_i$ are linearly independent. If $\mathbf{U}$ has columns forming a basis for $\mathbb{R}^N$ then $\tilde{\mathbf{U}} = \mathbf{U}^{-1}$.

corresponding to λ_i. In contrast, in Case 2 the basis vectors cannot all be eigenvectors and the basis for E_i can be constructed in different ways. One approach for basis construction is illustrated in Example 2.8 below.

Example 2.8 Basis for invariant subspaces of a defective matrix Let $\mathbf{Z}$ be a defective matrix having an eigenvalue λ of algebraic multiplicity $k = 2$ and geometric multiplicity $m = 1$. Find a basis for the space E of all vectors $\mathbf{x}$ such that $(\mathbf{Z} - \lambda\mathbf{I})^2\mathbf{x} = \mathbf{0}$.

Solution

If $\mathbf{u}^{(0)}$ is an eigenvector corresponding to λ, we have that $(\mathbf{Z} - \lambda\mathbf{I})\mathbf{u}^{(0)} = \mathbf{0}$, by definition, and thus $(\mathbf{Z} - \lambda\mathbf{I})^2\mathbf{u}^{(0)} = \mathbf{0}$ so that $\mathbf{u}^{(0)} \in E$. Since the algebraic multiplicity is 2, the invariant space E has dimension 2 but its geometric multiplicity is 1, so that we cannot find two linearly independent eigenvectors. Thus, we need to find a vector $\mathbf{u}^{(1)}$ such that $\mathbf{u}^{(0)}$ and $\mathbf{u}^{(1)}$ are linearly independent and $(\mathbf{Z} - \lambda\mathbf{I})^2\mathbf{u}^{(1)} = \mathbf{0}$. Choose $\mathbf{u}^{(1)}$ such that

$$(\mathbf{Z} - \lambda\mathbf{I})\mathbf{u}^{(1)} = \mathbf{u}^{(0)}. \tag{2.40}$$

For any such $\mathbf{u}^{(1)}$, we have $(\mathbf{Z} - \lambda\mathbf{I})^2\mathbf{u}^{(1)} = (\mathbf{Z} - \lambda\mathbf{I})\mathbf{u}^{(0)} = \mathbf{0}$ and, since $\mathbf{u}^{(0)}$ is non-zero, this means that $\mathbf{u}^{(1)}$ does not belong to $\mathcal{N}(\mathbf{Z} - \lambda\mathbf{I})$ and therefore $\mathbf{u}^{(0)}$ and $\mathbf{u}^{(1)}$ are linearly independent.

Jordan canonical form The construction sketched in Example 2.8 forms the basis for the Jordan canonical form. For any λ_i of algebraic multiplicity $k_i > 1$, if the geometric multiplicity is $m_i < k_i$ then we can find m_i linearly independent eigenvectors. For each of these eigenvectors we can follow the procedure sketched in Example 2.8 to create a *Jordan chain*. As in Example 2.8, $\mathbf{u}^{(1)}$ can be found by first solving (2.40); the next step is to find $\mathbf{u}^{(2)}$ such that

$$(\mathbf{Z} - \lambda\mathbf{I})\mathbf{u}^{(2)} = \mathbf{u}^{(1)}, \quad (\mathbf{Z} - \lambda\mathbf{I})^2\mathbf{u}^{(2)} = \mathbf{u} \quad \text{and} \quad (\mathbf{Z} - \lambda\mathbf{I})^3\mathbf{u}^{(2)} = \mathbf{0},$$

or, equivalently, such that

$$\mathbf{Z}\mathbf{u}^{(0)} = \lambda\mathbf{u}^{(0)}, \quad \mathbf{Z}\mathbf{u}^{(1)} = \lambda\mathbf{u}^{(1)} + \mathbf{u}^{(0)} \quad \text{and} \quad \mathbf{Z}\mathbf{u}^{(2)} = \lambda\mathbf{u}^{(2)} + \mathbf{u}^{(1)}.$$

For any vector $\mathbf{x} \in \text{span}(\mathbf{u}^{(2)}, \mathbf{u}^{(1)}, \mathbf{u}^{(0)})$, so that $\mathbf{x} = x_2\mathbf{u}^{(2)} + x_1\mathbf{u}^{(1)} + x_0\mathbf{u}^{(0)}$, multiplication by $\mathbf{Z}$ can be written in a compact form:

$$\mathbf{y} = \mathbf{Z}\mathbf{x} = \mathbf{Z}(x_2\mathbf{u}^{(2)} + x_1\mathbf{u}^{(1)} + x_0\mathbf{u}^{(0)}) = \lambda x_2\mathbf{u}^{(2)} + (\lambda x_1 + x_2)\mathbf{u}^{(1)} + (\lambda x_0 + x_1)\mathbf{u}^{(0)},$$

where we can see that $\mathbf{y}$ and $\mathbf{x}$ are in the same subspace (since both are linear combinations of $\mathbf{u}^{(2)}$, $\mathbf{u}^{(1)}$, $\mathbf{u}^{(0)}$) and therefore span$(\mathbf{u}^{(2)}, \mathbf{u}^{(1)}, \mathbf{u}^{(0)})$ is invariant under multiplication by $\mathbf{Z}$; $\mathbf{y}$ can then be written as

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} \lambda & 1 & 0 \\ 0 & \lambda & 1 \\ 0 & 0 & \lambda \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}. \tag{2.41}$$

The matrix in (2.41) is an example of a Jordan block. A Jordan chain starts with each of the m_i eigenvectors and p_i will have the length of the longest Jordan chain.

Example 2.9 Defective graph As a concrete example, consider a directed path graph with four nodes, leading to the adjacency matrix

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

where we note that one node has only one outgoing link, while another only has an incoming link. Prove that $\mathbf{A}$ is defective, find three independent vectors in the invariant subspace and interpret their behavior.

Solution

First note that, since the determinant of an upper triangular matrix is the product of its diagonal terms, we have $\det(\lambda\mathbf{I} - \mathbf{A}) = \lambda^4$, where $\lambda = 0$ is an eigenvalue with multiplicity 4. Then, solving for an eigenvector leads to $u(2) = u(3) = u(4) = 0$, so that $\mathbf{u}_1 = \mathbf{e}_1 = [1, 0, 0, 0]^\mathsf{T}$ is the only eigenvector. Notice that $\mathbf{u}_2 = \mathbf{e}_2$ is such that $\mathbf{A}\mathbf{u}_2 = \mathbf{u}_1$, and therefore $\mathbf{A}^2\mathbf{u}_2 = 0$, so that $\mathbf{u}_2$ is not an eigenvector but belongs to the invariant space with polynomial $\mathbf{A}^4$. Likewise $\mathbf{e}_3$ and $\mathbf{e}_4$ are also in that invariant subspace. For any vector in $\mathbb{R}^4$ we have $\mathbf{A}^4\mathbf{x} = \mathbf{0}$. We can also observe directly that $\mathbf{A}$ is a Jordan block as in (2.41) with diagonal values $\lambda = 0$. In conclusion, since $\lambda = 0$ has algebraic multiplicity 4, but there is only one eigenvector, $\mathbf{e}_1$, corresponding to this eigenvalue, the geometric multiplicity is smaller than the algebraic multiplicity and $\mathbf{A}$ is defective.

Diagonalization and invariance Our goal in this section is to understand graph filters, polynomials of $\mathbf{Z}$, from the perspective of their invariance properties. It is important to emphasize that invariance under multiplication by $\mathbf{Z}$ exists whether or not $\mathbf{Z}$ can be diagonalized.

Remark 2.3 Let eigenvalue λ_i have algebraic multiplicity k_i, and define $E_i = \mathcal{N}\left((\mathbf{Z} - \lambda_i\mathbf{I})^{k_i}\right)$. Then E_i is invariant under multiplication by $\mathbf{Z}$.

Proof By definition $\mathbf{x} \in E_i$ if $(\mathbf{Z} - \lambda_i\mathbf{I})^{k_i}\mathbf{x} = \mathbf{0}$, so that, using the commutativity of polynomials (Theorem 2.1),

$$(\mathbf{Z} - \lambda_i\mathbf{I})^{k_i}\mathbf{Z}\mathbf{x} = \mathbf{Z}(\mathbf{Z} - \lambda_i\mathbf{I})^{k_i}\mathbf{x} = \mathbf{0},$$

which shows that $\mathbf{Z}\mathbf{x} \in E_i$ and proves that the space is invariant. □

Any vector in $\mathbb{R}^N$ can be written as a linear combination of vectors belonging to subspaces $E_1, E_2, \ldots, E_M$, where M is the number of distinct eigenvalues. The only difference between the diagonalizable and defective cases is that in the former we can form a basis for E_i consisting of eigenvectors, while this is not possible for all E_i in the latter case. For a given subspace E_i the power p_i of the term $(\mathbf{Z} - \lambda_i \mathbf{I})^{p_i}$ in the minimal polynomial is the minimal value such that $(\mathbf{Z} - \lambda_i \mathbf{I})^{p_i}\mathbf{x} = \mathbf{0}$ for any $\mathbf{x} \in E_i$. These ideas are summarized in Table 2.1.

Table 2.1 Summary: Diagonalizable and non-diagonalizable operators

	Diagonalizable	Non-diagonalizable
Multiplicity	$m_i = k_i, \forall i$	$m_i < k_i$ for at least one i
$\dim(\mathcal{N}((\mathbf{Z} - \lambda_i \mathbf{I})^{k_i}))$	$\dim(\mathcal{N}(\mathbf{Z} - \lambda_i \mathbf{I}))$	$\dim(\mathcal{N}((\mathbf{Z} - \lambda_i \mathbf{I})^{p_i}))$
Invariant subspaces	N	$M = \sum_i m_i$
$p_{\mathrm{c}}(\mathbf{Z})$	$\prod_i (\mathbf{Z} - \lambda_i \mathbf{I})^{k_i}$	$\prod_i (\mathbf{Z} - \lambda_i \mathbf{I})^{k_i}$
$p_{\min}(\mathbf{Z})$	$\prod_i (\mathbf{Z} - \lambda_i \mathbf{I})$	$\prod_i (\mathbf{Z} - \lambda_i \mathbf{I})^{p_i}$
$\mathbf{U}$	N eigenvectors	$\sum_i m_i$ eigenvectors
Λ	Diagonal	Block diagonal

Schur decomposition for defective matrices The Jordan canonical form for a defective $\mathbf{Z}$ has well-known numerical problems. An alternative approach is to start from the Schur decomposition (see Theorem 2.2) and obtain a block diagonal form from it. See for example [18, 19] for a general description. The idea of using the Schur decomposition as an alternative to the Jordan form was first proposed in the context of GSP in [20] and also implemented in the `GraSP Matlab` toolbox [21] described in Appendix B. Denote $\mathbf{U}_i$ as a matrix with k_i columns forming a basis for E_i and let $\tilde{\mathbf{U}}_i$ be $N \times k_i$ and such that $\tilde{\mathbf{U}}_i^{\mathsf{H}} \mathbf{U}_i = \mathbf{I}_{k_i}$, where $\mathbf{I}_{k_i}$ is the identity matrix of size $k_i \times k_i$. In this case we again have, for $\mathbf{x}_i \in E_i$,

$$\mathbf{x}_i = \mathbf{U}_i \mathbf{a}, \tag{2.42}$$

and therefore

$$\mathbf{a} = \tilde{\mathbf{U}}_i^{\mathsf{H}} \mathbf{x}_i \tag{2.43}$$

so that

$$\mathbf{Z}\mathbf{x}_i = \mathbf{U}_i \mathbf{\Lambda}_i \tilde{\mathbf{U}}_i^{\mathsf{H}} \mathbf{x}_i \tag{2.44}$$

where the main difference with respect to Case 1 is that Λ_i is not diagonal.

Importance of the defective case In some cases making use of the original graph is important and so it will be necessary to work with a defective $\mathbf{Z}$. In other cases we may consider changing the graph. Informally, it is always possible to find a diagonalizable matrix "close" to any non-diagonalizable one. Thus, if $\mathbf{Z}$ cannot be be diagonalized, we could look for an alternative, $\bar{\mathbf{Z}}$, that: (i) can be diagonalized, (ii) is close to $\mathbf{Z}$ (e.g., in terms of the Frobenius norm of the difference) and (iii) represents a graph (e.g., $\bar{\mathbf{Z}}$ is a valid adjancency matrix).

If selecting a new $\bar{\mathbf{Z}}$ is an option, several methods can be used. Recalling that symmetric matrices can always be diagonalized, one approach would be to convert the directed graph into an undirected one. For a directed $\mathbf{Z} = \mathbf{A}$, a natural symmetrization would be $\bar{\mathbf{Z}} = \mathbf{A} + \mathbf{A}^\mathsf{T}$, with the bibliometric symmetrization $\bar{\mathbf{Z}} = \mathbf{A}\mathbf{A}^\mathsf{T} + \mathbf{A}^\mathsf{T}\mathbf{A}$ as a possible alternative [5]. Other approaches that preserve the directed nature of the graph can be based on structural criteria, which modify the graph for a specific application. An example of this approach is the teleportation idea in PageRank [4]. Moreover, since some graph structures (such as path sources or path sinks) lead to a defective $\mathbf{Z}$, removing those can lead to graphs with better behavior.

More generally, as discussed in Chapter 6, in many cases the choice of graph is in itself a problem, and thus, when deciding what graph to choose in a particular case, the relevant properties of the corresponding graph operator $\mathbf{Z}$ should be taken into account.

2.5.5 Practical Implications for Graph Filter Design

To conclude this chapter we summarize the key consequences of our study of polynomials $\mathbf{Z}$ for the design of practical graph filters.

Localization depends on polynomial degree and graph topology As discussed in detail in Section 2.2, a degree-k polynomial corresponds to k-hop localized processing around every node in the graph. But the choice of an appropriate parameter k should be a function of prior knowledge about the graph signal of interest (e.g., over what size subgraphs can we expect to see locally similar behavior) as well as properties of the graph topology, such as radius or diameter. It is clear that a polynomial of degree $k_1 < k_2$ will provide more localized processing than one of degree k_2, but just how localized this is depends on the graph properties. This is particularly important in applications where graphs are reduced (see Section 6.1.5) and become denser as the number of nodes is decreased. Thus, if a k-hop filter is appropriate for the original graph, it may not be the right choice for a smaller graph derived from the original one.

Invariant subspaces and graph signal analysis We have described in detail subspaces of vectors that are invariant under multiplication by $\mathbf{Z}$. From a graph signal analysis perspective the corresponding minimal polynomials, which by definition produce a zero output for any signal in that subspace, may be useful. If E is an invariant subspace, and if $P_E(\mathbf{Z})$ is the corresponding minimal polynomial, then $P_E(\mathbf{Z})\mathbf{x} = \mathbf{0}$ for any $\mathbf{x} \in E$. Therefore, when analyzing some arbitrary signals $\mathbf{y} \in \mathbb{R}^N$ we can use $P_E(\mathbf{Z})$ to eliminate any component of such a signal that belongs to E. The signal $P_E(\mathbf{Z})\mathbf{y}$ will contain no information corresponding to subspace E.

In Chapter 3 we will discuss the problem of filter design, with an emphasis on the case where the graph operator $\mathbf{Z}$ can be diagonalized. But, more generally, even for defective $\mathbf{Z}$ the effect of a polynomial $P(\mathbf{Z})$ can be completely characterized by its effect on each invariant subspace. In particular, the dimension of each of these invariant subspaces is important, since for spaces of dimension greater than 1 the choice of basis functions is not unique, as discussed below in Box 2.7.

Box 2.7 Non-uniqueness of bases for invariant subspaces

If an eigenvalue λ_i is not simple, $k_i > 1$, we have a fundamental ambiguity. The invariant subspace E_i has dimension greater than 1 and thus an infinite number of basis functions can be selected for E_i. Since we are characterizing signals by their response to $\mathbf{Z}$ all the vectors in E_i have the same properties and same minimal polynomial $(\mathbf{Z} - \lambda_i \mathbf{I})^{p_i}$. Thus, any bases for vectors in E_i should be equivalent.

As noted in [22], unless there is a standardized way of selecting a basis for an invariant space E_i with $\dim(E_i) > 1$ we may not be able to compare representations of the same signal $\mathbf{x}$ produced with different implementations. The alternative proposed in [22] is to use the expressions (2.42) and (2.43). Letting $\mathbf{x}$ be any vector in $\mathbb{R}^N$ we choose its (non-orthogonal) projection $\mathbf{x}_i$ onto E_i as follows:

$$\mathbf{x}_i = \mathbf{U}_i \tilde{\mathbf{U}}_i^{\mathsf{H}} \mathbf{x}. \tag{2.45}$$

The vector $\mathbf{x}_i$ is independent of the basis chosen to represent E_i. This is not an orthogonal projection: the approximation error is orthogonal to the dual vectors, $\tilde{\mathbf{U}}_i$, rather than the spanning vectors, $\mathbf{U}_i$, as would be the case for an orthogonal projection:

$$\tilde{\mathbf{U}}_i^{\mathsf{H}}(\mathbf{x} - \mathbf{x}_i) = \tilde{\mathbf{U}}_i^{\mathsf{H}}(\mathbf{I} - \mathbf{U}_i \tilde{\mathbf{U}}_i^{\mathsf{H}})\mathbf{x} = (\tilde{\mathbf{U}}_i^{\mathsf{H}} - \tilde{\mathbf{U}}_i^{\mathsf{H}})\mathbf{x} = \mathbf{0}$$

where we have used the fact that $\tilde{\mathbf{U}}_i^{\mathsf{H}} \mathbf{U}_i = \mathbf{0}$, from the definition of biorthogonal bases (Definition A.8), and the conjugate transpose appears because the basis may be complex.

In summary, given invariant subspaces E_i the representation

$$\mathbf{x} = \sum_i \mathbf{x}_i,$$

where $\mathbf{x}_i$ is computed as in (2.45) and there is one term $\mathbf{x}_i$ per subspace E_i, which is unique and eliminates any ambiguity due to the choice of basis vectors for each E_i.

Chapter at a Glance

In this chapter we started by introducing basic definitions associated with graphs and using those to develop an understanding of node domain processing. Since graphs are characterized by one-hop connections between nodes, we explored how these induce a notion of locality, and how it can be extended so that a one-hop connection induces a k-hop neighborhood around a node. A natural follow-up step is to consider neighborhoods around multiple close nodes, leading to the notion of cuts and clustering. We then introduced algebraic representations of graphs and discussed how these can be viewed as one-hop operations (see Table 2.2). This was then generalized to define graph filters as polynomials of an elementary graph operator $\mathbf{Z}$. We showed that the algebraic properties of $\mathbf{Z}$ determine what operations can be performed on a graph. In particular, vectors that are invariant under multiplication by $\mathbf{Z}$ can be used to understand the effect

Table 2.2 Summary of one-hop operators

	$\mathbf{Z}$	$\mathbf{y} = \mathbf{Z}\mathbf{x}$
Adjacency matrix	$\mathbf{A}$	$y(i) = \sum_{j\in\mathcal{N}(i)} a_{ij}x(j)$
Random walk	$\mathcal{Q} = \mathbf{D}^{-1}\mathbf{A}$	$y(i) = \frac{1}{d_i}\sum_{j\in\mathcal{N}(i)} a_{ij}x(j)$
Combinatorial Laplacian	$\mathbf{L} = \mathbf{D} - \mathbf{A} = \mathbf{B}\mathbf{B}^\mathsf{T}$	$y(i) = d_i x(i) - \sum_{j\in\mathcal{N}(i)} a_{ij}x(j)$
Normalized Laplacian	$\mathcal{L} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$	$y(i) = x(i) - \sum_{j\in\mathcal{N}(i)} \frac{a_{ij}}{\sqrt{d_i}\sqrt{d_j}}x(j)$
Random walk Laplacian	$\mathcal{T} = \mathbf{I} - \mathcal{Q} = \mathbf{D}^{-1}\mathbf{B}\mathbf{B}^\mathsf{T}$	$y(i) = x(i) - \frac{1}{d_i}\sum_{j\in\mathcal{N}(i)} a_{ij}x(j)$

of polynomials of $\mathbf{Z}$ on arbitrary vectors. When $\mathbf{Z}$ is diagonalizable there exists a full set of linear independent invariant vectors (i.e., eigenvectors). When $\mathbf{Z}$ is not diagonalizable, we can instead find invariant subspaces and use bases for those subspaces to create a complete representation for graph signals in the space. In either case every vector in the space can be written as a linear combination of elementary vectors, each belonging to a space invariant under multiplication by $\mathbf{Z}$.

Further Reading

Graph theory is a classical topic, with many textbooks available to cover topics such as graph coloring, graph cuts and so on [23]. Texts more focused on spectral graph theory, such as [8], also provide basic graph definitions and cover the elementary algebraic operators described in this chapter. For the development of graph signal processing from the perspective of a polynomial of an elementary one-hop operator and its connection with conventional signal processing see [15, 10] as well as the general algebraic signal processing framework of [24, 25]. The representation of a space in terms of a basis derived from invariance under multiplication by a matrix $\mathbf{Z}$ is developed in [26]. A detailed discussion of graph Fourier transforms in the context of non-diagonalizable $\mathbf{Z}$ is provided by [22]. The main difference in our presentation with respect to that of [22] is that we show that the Jordan blocks, which we also used, are only one of the possible block diagonal representations that could be chosen. Because of the well-known numerical issues associated with the Jordan form, other block diagonal representations may be preferable.

`GraSP` Examples

Section B.3 provides several examples of node domain filtering operations to illustrate the different approaches that can be used for a specific filter defined as a function of a fundamental graph operator. A direct approach, where a full matrix is computed and applied to the signal, is shown first. This approach may not be efficient for a large graph, however. Thus, a second example considers implementation via successive applications of the fundamental graph operator, where the filter is a polynomial of the operator.

Problems

2.1 *Coloring*

Finding a graph coloring can be simple. If we have N nodes we can assign a different color to each node and the graph will be N-colorable. The challenge is finding the minimum coloring, i.e., the one with the minimum number of colors.

Find the minimum coloring for the following graphs:

a. a complete unweighted graph with N nodes;

b. a star graph with N nodes;

c. a tree with K levels.

2.2 *Coloring and number of edges*

(True/False) Given an unweighted graph with N nodes and chromatic number C (see Section 2.2.5), if we add an edge to connect two nodes that are not yet connected, then the chromatic number always increases.

2.3 *Trees*

In this problem we consider an unweighted tree with K levels. Discuss whether your answers depend on specific tree properties (balanced or not, minimum and maximum number of children per node, etc.).

a. Find its diameter and radius.

b. Is it possible to remove an edge in such a way that the diameter is increased while the graph remains connected?

c. Is it possible to add one edge to the graph (which then may no longer be a tree) in such a way that the diameter is reduced while the radius remains unchanged?

d. What is the chromatic number of the tree?

2.4 *Connectedness*

In this problem we consider an unweighted directed graph with N nodes with corresponding adjacency matrix $\mathbf{A}$. Assume that $\mathbf{A}$ is reducible, so that it is possible to find a permutation of $\mathbf{A}$ to put it into block upper triangular form:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_{12} \\ \mathbf{0} & \mathbf{A}_2 \end{bmatrix}$$

where $\mathbf{A}_1$ and $\mathbf{A}_2$ are square matrices of sizes $N_1 \times N_1$ and $N_2 \times N_2$ with $N_1 + N_2 = N$.

a. Prove that the corresponding graph is not strongly connected.

b. Assuming that the graph is weakly connected, what would be the minimum number of edges to be added in order to make it strongly connected?

c. Consider the special case where only entries directly above the diagonal are non-zero. What is this graph? Is it weakly connected?

2.5 *Processing signals on a tree*

Consider a balanced binary tree with K levels.

a. Assuming there are two processors, what is the best way to split the tree for processing into two roughly equal subgraphs?

b. Assuming we are performing a k-hop localized graph filter on this tree, how much data has to be exchanged between the two processors?

2.6 *Normalization*

In Section 2.3.3 we discussed two normalization strategies, one where the goal was to achieve consensus, the other where some physical goods were transported. Here we consider an extension of the latter case, where physical goods are transported but where a percentage of the goods is lost while being stored at a node before being distributed. Assume only a fraction $0 < \alpha < 1$ of the goods at any node is actually transported.

a. Let $\mathbf{L}$ be the combinatorial Laplacian of the original graph. For a given α propose a normalization matrix $\mathbf{K}$ as an alternative for the above approach using normalization strategies mentioned above.

b. In the case in part **a**, given an initial signal $\mathbf{x}^{(0)}$ with all positive entries and defining $N_k = \mathbf{1}^{\mathsf{T}}\mathbf{x}^{(k)}$, find k such that $N_k/N_0 < \epsilon$ for a given $0 < \epsilon < 1$.

c. Repeat parts **a** and **b** for the case where the loss is node-dependent, that is, there is an $0 < \alpha_i < 1$ associated with each node i.

2.7 *Adjacency matrices and permutations*

a. Write the adjacency matrix of an eight-node undirected and unweighted cycle graph with consecutive labeling of the nodes along the cycle.

b. Repeat this for the case where the labels along the cycle are $1, 3, 5, 7, 2, 4, 6, 8$.

c. With the first labeling the adjacency matrix is circulant, that is, each row can be obtained by shifting by one entry the row immediately above (and the first row can be obtained from the eighth). Assuming we add an edge from node 1 to node 4, what other edges should be added so that the resulting adjacency matrix is still circulant?

2.8 *Minimal polynomials of vectors*

Let $\mathbf{x}_1$ and $\mathbf{x}_2$ be two graph signals with respective minimal polynomials $p_1(\mathbf{Z})$ and $p_2(\mathbf{Z})$ having degrees n_1 and n_2. Assume that $\mathbf{Z}$ is $n \times n$ and that all its eigenvalues are simple.

a. What is the minimal polynomial for the vectors in the set $S = \text{span}(\mathbf{x}_1, \mathbf{x}_2)$?

b. Letting $p_{\min}(\mathbf{Z})$ be the minimal polynomial of $\mathbf{Z}$, is it possible for the minimal polynomial of S to be $p_{\min}(\mathbf{Z})$? If not, justify why this is so. If on the other hand it is possible then give an example of two vectors $\mathbf{x}_1$, $\mathbf{x}_2$ for which this is true.

2.9 *Non-diagonalizable* $\mathbf{Z}$

Consider a directed unweighted path graph of length 4.

a. Prove that the the operator $\mathbf{Z} = \mathbf{A}$ is not diagonalizable.

b. What are the invariant subspaces of $\mathbf{Z}$ as defined in **a**?

2.10 *Directed acyclic graphs*

Let $\mathbf{A}$ be the adjacency matrix of a directed acyclic graph (DAG). Prove that $\mathbf{A}$ is not diagonalizable.

2.11 *Path sinks and path sources*

In a directed graph $\mathcal{G}$ we define a path sink (or source) with n nodes as a structure where a node i_0 is connected to the rest of the graph by only one incoming edge (for a sink) or one outgoing edge (for a source) and where the connection to the rest of the graph is via a directed path connecting nodes $(i_1, i_2, \ldots, i_n)$.

a. Using a suitable permutation, write the adjacency matrices for two graphs having a single path sink and a single path source, respectively, with both paths having n nodes.

b. Prove that for $n > 1$ the corresponding adjacency matrices cannot be diagonalized.

2.12 *Nilpotent* $\mathbf{Z}$

Let $\mathbf{Z}$ be nilpotent, such that there is a $k > 1$ for which $\mathbf{Z}^k = \mathbf{0}$ while $\mathbf{Z}^l \neq \mathbf{0}$ for $1 \leq l < k$. Prove that $\mathbf{Z}$ cannot be diagonalized.

2.13 *Symmetrization*

Let $\mathbf{A}$ be the adjacency matrix of a directed graph, with the corresponding incidence matrix $\mathbf{B}$, where there are $M = |E|$ edges and $\mathbf{b}_k$ denotes the kth column of $\mathbf{B}$.

a. Find the incidence matrix of $\mathbf{A} + \mathbf{A}^\mathsf{T}$.

b. Find the incidence matrices for $\mathbf{A}\mathbf{A}^\mathsf{T}$, $\mathbf{A}^\mathsf{T}\mathbf{A}$ and $\mathbf{A}\mathbf{A}^\mathsf{T} + \mathbf{A}^\mathsf{T}\mathbf{A}$.

c. Provide a qualitative comparison of the symmetrizations $\mathbf{A}_1 = \mathbf{A} + \mathbf{A}^\mathsf{T}$ and $\mathbf{A}_2 = \mathbf{A}\mathbf{A}^\mathsf{T} + \mathbf{A}^\mathsf{T}\mathbf{A}$ with the corresponding incidence matrices $\mathbf{B}_1$ and $\mathbf{B}_2$ in terms of their number of edges (sparsity).