Part III

Decentralized Optimization and Learning

6 Graphs and their properties

Graphs play a critical role in decentralized learning settings as a tool to model constraints on interactions between pairs of agents. In this chapter, we review fundamental properties of graphs and related quantities that will turn out to be critical to the development and analysis of decentralized learning algorithms.

6.1 GRAPHS

An directed, unweighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consists of a pair of sets $\mathcal{N}$ and $\mathcal{E}$, where $\mathcal{N}$ denotes a set of *vertices* or *nodes*, and $\mathcal{E}$ denotes a set of directed *edges* linking pairs of vertices in $\mathcal{N}$. Each edge in $\mathcal{E}$ is represented as a pair (i.e., a 2-tuple) of vertices in $\mathcal{N}$. In principle, the elements of $\mathcal{V}$ can be arbitrary objects. For simplicity, it is generally sufficient number the vertices in $\mathcal{N}$ beginning at one through $|\mathcal{N}|$ and represent the elements through their respective index. In most chapters, vertices will be "agents", or "learners", terms which we use interchangeably in our presentation.

Example 6.1 (**A Simple Graph**) We consider a collection of 4 vertices, numbered 1 through 4. The vertices are collected in the set:

$$\mathcal{V} = \{1, 2, 3, 4\} \tag{6.1}$$

We construct a sample graph using the following set of edges:

$$\mathcal{E} = \{(1,2), (2,1), (2,3), (3,2), (3,4), (4,3), (2,4), (4,2)\} \tag{6.2}$$

The resulting graph $\mathcal{G} \triangleq (\mathcal{V}, \mathcal{E})$ is displayd in Fig. 6.1.

Figure 6.1 A simple graph consisting of four vertices.

6.2 GRAPH TOPOLOGY

6.2.1 Neighborhoods

Figure 6.2 shows the example of a graph consisting of K connected agents, labeled $k = 1, 2, \ldots, K$. The agents are represented by vertices in the graph with edges linking them. An edge that connects an agent to itself is called a *self-loop*. The neighborhood of an agent k is denoted by $\mathcal{N}_k$ and consists of all agents that are connected to k by an edge, in addition to agent k itself. Any two neighboring agents k and ℓ have the ability to share information over the edge connecting them. In later chapters, each agent k will also be assigned a risk function, denoted by $J_k(w)$.

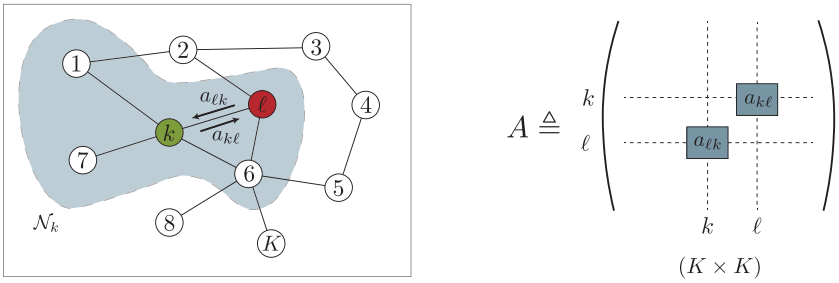

Figure 6.2 Agents are linked by a graph topology and can share information over their shared edges. The neighborhood of an agent is the collection of all agents linked to it. The neighborhood of agent k is marked by the highlighted area and denoted by $\mathcal{N}_k$.

6.2.2 Directed, undirected, and symmetric graphs

We say that a graph is *undirected*, if the presence of a link from agent k agent ℓ, implies that there is a link agent ℓ to agent k. In other words:

$$k \in \mathcal{N}_\ell \iff \ell \in \mathcal{N}_k \tag{6.3}$$

If there are pairs of vertices with connections only in one direction, we will refer to the graph as *directed*.

We assign a pair of nonnegative weights, $\{a_{k\ell}, a_{\ell k}\}$, to the edge connecting k and ℓ. The scalar $a_{\ell k}$ will be used by agent k to scale data it receives from agent ℓ; this scaling can be interpreted as a measure of the confidence level that agent k assigns to its interaction with agent ℓ. Likewise, $a_{k\ell}$ is used by agent ℓ to scale the data it receives from agent k. Whether agents linked by edges will exchange information, and whether the exchange will be unidirectional, bidirectional, or non-existent, will depend on the values of the weights $\{a_{k\ell}, a_{\ell k}\}$. In principle, the weights $\{a_{k\ell}, a_{\ell k}\}$ can be different so that the exchange of information between the neighboring agents $\{k, \ell\}$ need not be symmetrical. One or both weights can

also be zero. Although many of the decentralized algorithms described in this chapter can operate under such asymmetric conditions, it is generally assumed that $a_{k\ell} = a_{\ell k}$. Whenever $a_{k\ell} = a_{\ell k}$, we refer to the graph as *symmetric*, while the presence of at least one pair of agents with $a_{k\ell} \neq a_{\ell k}$ makes the graph *asymmetric*.

6.2.3 Strong-Connectedness

The graph is said to be *connected* if paths with *nonzero* scaling weights can be found linking any two distinct agents in *both* directions, either directly when they are neighbors or by passing through intermediate agents when they are not neighbors. In this way, information can flow in both directions between any two agents in the graph, although the forward path from an agent k to some other agent ℓ need not be the same as the backward path from ℓ to k. A *strongly-connected* network is a connected network with at least one non-trivial *self-loop*, meaning that $a_{kk} > 0$ for some agent k. Intuitively, this means that there exists at least one agent in the network that trusts its own information and will assign some positive weight to it. If $a_{kk} = 0$ for all k, then this means that all agents will be ignoring their individual information and will be relying instead on information received from other agents. The terminology of "strongly-connected networks" is perhaps somewhat excessive because it may convey wrongly the impression that the graph needs to have more connectivity than is actually necessary. In fact, the graph can be rather sparse.

Observe that we are defining connectivity over graphs not in terms of whether paths can be found connecting their vertices but in terms of whether these paths allow for the *meaningful* exchange of information between the vertices. This is because of the requirement that all scaling weights over the edges must be *positive* over at least one of the paths connecting any two distinct vertices. The assumption of a connected graph ensures that information will be flowing between any two arbitrary agents in the network and that this flow of information is bidirectional: information flows from k to ℓ and from ℓ to k, although the paths over which the flows occur need not be the same and the manner by which information is scaled over these paths can also be different.

The strong connectivity of a graph translates into a useful property on the combination weights. Assume we collect the coefficients $\{a_{\ell k}\}$ into a $K \times K$ matrix $A = [a_{\ell k}]$, such that the entries on the $k-$th column of A contain the coefficients used by agent k to scale data arriving from its neighbors $\ell \in \mathcal{N}_k$; we set $a_{\ell k} = 0$ if $\ell \notin \mathcal{N}_k$. In this way, the row index in (ℓ, k) designates the *source* agent and the column index designates the *sink* agent (or destination). We refer to A as the *combination* matrix or policy. It turns out that combination matrices that correspond to strongly-connected networks are *primitive* — a $K \times K$ matrix A with nonnegative entries is said to be primitive if there exists some finite integer

$n_o > 0$ such that all entries of A^{n_o} are strictly positive:

$$[A^{n_o}]_{\ell,k} > 0, \quad \text{uniformly for all } (\ell, k) \tag{6.4}$$

LEMMA 6.1 (**Primitive combination matrix**). *The $K \times K$ combination matrix of a strongly-connected graph with K vertices is a primitive matrix.*

Proof: See Problem 6.1.

□

One important consequence of the primitiveness of A is that a famous result in matrix theory, known as the *Perron-Frobenius* theorem, characterizes the eigenstructure of A. In particular, it will hold that:

(a) The matrix A has a *single* eigenvalue at one.
(b) All other eigenvalues of A are *strictly inside* the unit circle so that $\rho(A) = 1$.
(c) With proper sign scaling, all entries of the right-eigenvector of A corresponding to the single eigenvalue at one are *positive*. Let p denote this right-eigenvector, with its entries $\{p_k\}$ normalized to add up to one, i.e.,

$$\boxed{Ap = p, \quad \mathbb{1}^{\mathsf{T}} p = 1, \quad p_k > 0, \;\; k = 1, 2, \ldots, K} \tag{6.5}$$

We refer to p as the *Perron vector* or eigenvector of A. All other eigenvectors of A associated with the other eigenvalues will have at least one negative or complex entry.
(d) The Jordan decomposition $A = V_\epsilon J V_\epsilon^{-1}$ has a very particular structure:

$$V_\epsilon = \begin{bmatrix} p & V_R \end{bmatrix}, \quad J = \begin{bmatrix} 1 & 0 \\ 0 & J_\epsilon \end{bmatrix}, \quad V_\epsilon^{-1} = \begin{bmatrix} \mathbb{1}^{\mathsf{T}} \\ V_L^{\mathsf{T}} \end{bmatrix} \tag{6.6}$$

where J_ϵ is a block Jordan matrix with the eigenvalues $\lambda_2(A)$ through $\lambda_N(A)$ on the diagonal and ϵ on the first lower sub-diagonal. In particular, $\rho(J_\epsilon) < 1$.
(e) It holds that (see Problem 6.2):

$$\lim_{i \to \infty} A^i = p^{\mathsf{T}} \mathbb{1} \tag{6.7}$$

6.2.4 Graph Laplacian

We associate a Laplacian matrix with every graph; it is a symmetric matrix with useful properties that reveal the connectedness of the graph.

Thus, consider again a graph consisting of K vertices and E edges connecting these vertices to each other. We only need to focus on the edges that connect distinct nodes to each other; we can ignore any self-loops that may exist in the graph. In other words, when we refer to the E edges of the graph, we are excluding self-loops from this set; but we are still allowing loops of at least length 2 (i.e., loops generated by paths covering at least 2 edges).

The degree of a node k, which we denote by n_k, is defined as the positive integer that is equal to the size of its neighborhood, $\mathcal{N}_k$:

$$n_k \triangleq |\mathcal{N}_k| \tag{6.8}$$

Since by convention $k \in \mathcal{N}_k$, it holds that $n_k \geq 1$. The entries of the $K \times K$ Laplacian matrix, denoted by L, are defined by:

$$[L]_{k\ell} = \begin{cases} n_k - 1, & \text{if } k = \ell \\ -1, & \text{if } k \neq \ell \text{ and nodes } k \text{ and } \ell \text{ are neighbors} \\ 0, & \text{otherwise} \end{cases} \tag{6.9}$$

Note that the term $n_k - 1$ measures the number of edges that are incident on node k, and the locations of the $-1's$ on row k indicate the nodes that are connected to node k. We also associate with the graph a $K \times E$ *incidence matrix*, denoted by $\mathcal{I}$. The entries of $\mathcal{I}$ are defined as follows. Every column of $\mathcal{I}$ represents one edge in the graph. Each edge connects two nodes and its column will display two nonzero entries at the rows corresponding to these nodes: one entry will be $+1$ and the other entry will be -1. For directed graphs, the choice of which entry is positive or negative can be used to identify the nodes from which edges emanate (source nodes) and the nodes to which edges arrive (sink nodes). Since we are dealing with undirected graphs, we will simply assign positive values to lower indexed nodes and negative values to higher indexed nodes:

$$[\mathcal{I}]_{ke} = \begin{cases} +1, & \text{if node } k \text{ is the lower-indexed node connected to edge } e \\ -1, & \text{if node } k \text{ is the higher-indexed node connected to edge } e \\ 0, & \text{otherwise} \end{cases} \tag{6.10}$$

Figure 6.3 shows the example of a network with $K = 6$ nodes and $E = 8$ edges. Its Laplacian and incidence matrices are also shown and these have sizes 6×6 and 6×8, respectively. Consider, for example, column 6 in the incidence matrix. This column corresponds to edge 6, which links nodes 3 and 5. Therefore, at location $\mathcal{I}_{3,6}$ we have a $+1$ and at location $\mathcal{I}_{5,6}$ we have -1. The other columns of $\mathcal{I}$ are constructed in a similar manner.

Observe that the Laplacian and incidence matrices of a graph are related as follows:

$$L = \mathcal{I}\,\mathcal{I}^{\mathsf{T}} \tag{6.11}$$

The Laplacian matrix conveys useful information about the topology of the graph. The following is a classical result from graph theory.

LEMMA 6.2 (**Laplacian and graph connectivity**). *Let $\theta_1 \geq \theta_2 \geq \ldots \geq \theta_K$ denote the ordered eigenvalues of L. Then the following properties hold:*

(a) L is symmetric nonnegative-definite so that $\theta_k \geq 0$, for $k = 1, 2, \ldots, K$.

(b) The entries on each row of L add up to zero so that $\mathcal{L}\mathbb{1} = 0$. This means that $\mathbb{1}$ is a right eigenvector of L corresponding to the eigenvalue at zero.

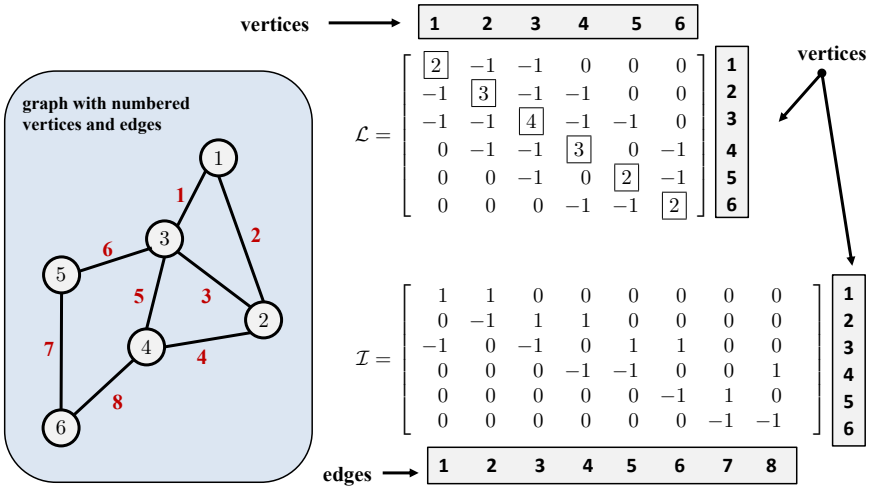

Figure 6.3 A graph with $K = 6$ vertices and $E = 8$ edges. The vertices are numbered 1 through 6 and the edges are also numbered 1 through 8. The corresponding Laplacian and incidence matrices L and $\mathcal{I}$ are 6×6 and 6×8, respectively.

(c) *The smallest eigenvalue is always zero, i.e., $\theta_K = 0$. The second smallest eigenvalue, θ_{K-1}, is called the algebraic connectivity of the graph.*

(d) *The number of times that zero is an eigenvalue of L (i.e., its multiplicity) is equal to the number of connected subgraphs.*

(e) *The algebraic connectivity of a connected graph is nonzero, i.e., $\theta_{K-1} \neq 0$. In other words, a graph is connected if, and only if, its algebraic connectivity is nonzero.*

Proof: Property (a) follows from the identity $L = \mathcal{I}\,\mathcal{I}^T$, which shows that L is symmetric and nonnegative-definite. Property (b) follows from the definition of L. Note that for each row of L, the entries on the row add up to zero. Property (c) follows from properties (a) and (b) since property (a) implies that all eigenvalues of L are nonnegative and property (b) implies that zero is one of the eigenvalues of L.

For part (d), assume the network consists of two separate connected subgraphs. Then, the Laplacian matrix would have a block diagonal structure, say, of the form $L = \text{diag}\{L_1, L_2\}$, where L_1 and L_2 are the Laplacian matrices for the smaller subgraphs. The smallest eigenvalue of each of these Laplacian matrices would in turn be zero and unique by property (e). More generally, if the graph consists of m connected subgraphs, then the multiplicity of zero as an eigenvalue of L must be m.

To establish property (e), first observe that if the algebraic connectivity is nonzero then it is obvious that the graph must be connected. Otherwise, if the graph were disconnected, then its Laplacian matrix would be block diagonal and the algebraic multiplicity of zero as an eigenvalue of L would be larger than one so that θ_{K-1} would be zero, which is a contradiction. For the converse statement, assume the graph is connected and let x denote an arbitrary eigenvector of L corresponding to the eigenvalue at zero, i.e., $Lx = 0$. We already know that $L\mathbb{1} = 0$ from property (b). Let us verify that x must be proportional to the vector $\mathbb{1}$ so that the algebraic multiplicity of the

eigenvalue at zero is one. Thus note that $x^{\mathsf{T}}Lx = 0$. If we denote the individual entries of x by x_k, then this identity implies that for each node k:

$$\sum_{\ell \in \mathcal{N}_k} (x_k - x_\ell)^2 \;=\; 0 \tag{6.12}$$

It follows that the entries of x within each neighborhood have equal values. But since the graph is connected, we conclude that all entries of x must be equal. It follows that the eigenvector x is proportional to the vector $\mathbb{1}$, as desired.

□

6.2.5 Weight Matrices

Most of the time, the weights $\{a_{\ell k}\}$ will be nonnegative scalars that satisfy the following condition for each agent $k = 1, 2, \ldots, K$:

$$a_{\ell k} \geq 0, \;\; a_{\ell k} = a_{k\ell}, \;\; \sum_{\ell=1}^{K} a_{\ell k} = 1, \;\; \text{and} \;\; a_{\ell k} = 0 \text{ if } \ell \notin \mathcal{N}_k \tag{6.13}$$

Condition (6.13) implies that the combination matrix $A = [a_{\ell k}]$ is symmetric and satisfies $A^{\mathsf{T}}\mathbb{1} = \mathbb{1}$, where $\mathbb{1}$ denotes the vector with all entries equal to one. We say that A is *doubly-stochastic* since the entries on each of its rows and on each of its columns add up to one:

$$A^{\mathsf{T}}\mathbb{1} = \mathbb{1}, \;\; A = A^{\mathsf{T}}, \qquad \text{(doubly-stochastic)} \tag{6.14}$$

It is useful to note that when A is doubly-stochastic, its Perron vector p defined in (6.5) is given by

$$p = \frac{1}{K}\mathbb{1} \tag{6.15}$$

If, on the other hand, A is not symmetric (i.e., if condition $a_{\ell k} = a_{k\ell}$ does not hold), then it would amount to a *left-stochastic* matrix and only the entries on each of its rows will add up to one:

$$A^{\mathsf{T}}\mathbb{1} = \mathbb{1}, \qquad \text{(left-stochastic)} \tag{6.16}$$

We will be dealing mainly with doubly-stochastic matrices in this chapter.

There are many choices for the matrix A. Table 6.1 lists some examples for graphs with K nodes. These examples are defined in terms of the degrees n_k of the various agents, with $n_{\max}$ refering to the maximum degree across the graph:

$$n_{\max} \;\triangleq\; \max_{1 \leq k \leq K} \; n_k \tag{6.17}$$

The Laplacian rule, which appears in the second row of the table, relies on the use of the Laplacian matrix L of the network and a positive scalar β. The Laplacian matrix was defined earlier in (6.9). The Laplacian rule can be reduced to other forms through the selection of the positive parameter β. One choice is $\beta = 1/n_{\max}$, while another choice is $\beta = 1/K$ and leads to the maximum-degree rule. Obviously, it always holds that $n_{\max} \leq K$ so that $1/n_{\max} \geq 1/K$.

Therefore, the choice $\beta = 1/n_{\max}$ ends up assigning larger weights to neighbors than the choice $\beta = 1/K$. The averaging rule in the first row of the table is one of the simplest combination rules whereby nodes simply average data from their neighbors.

Table 6.1 Examples of potential combination matrices $A = [a_{\ell k}]$. Other choices are of course possible.

Entries of combination matrix A	**Type of** A
1. Averaging rule: $a_{\ell k} = \begin{cases} 1/n_k, & \text{if } k \neq \ell \text{ are neighbors or } k = \ell \\ 0, & \text{otherwise} \end{cases}$	left-stochastic
2. Laplacian rule: $A = I_K - \beta\, L,\ \beta > 0$	doubly-stochastic
3. Laplacian rule using $\beta = 1/n_{\max}$: $a_{\ell k} = \begin{cases} 1/n_{\max}, & \text{if } k \neq \ell \text{ are neighbors} \\ 1 - (n_k - 1)/n_{\max}, & k = \ell \\ 0, & \text{otherwise} \end{cases}$	doubly-stochastic
4. Laplacian rule using $\beta = 1/K$(maximum-degree rule) : $a_{\ell k} = \begin{cases} 1/K, & \text{if } k \neq \ell \text{ are neighbors} \\ 1 - (n_k - 1)/K, & k = \ell \\ 0, & \text{otherwise} \end{cases}$	doubly-stochastic
5. Metropolis rule: $a_{\ell k} = \begin{cases} 1/\max\{n_k, n_\ell\}, & \text{if } k \neq \ell \text{ are neighbors} \\ 1 - \sum_{\ell \in \mathcal{N}_k \backslash \{k\}} a_{\ell k}, & \text{when } k = \ell \\ 0, & \text{otherwise} \end{cases}$	doubly-stochastic

Stochastic weight matrices A have some useful properties, which we will exploit in later sections. We establish them here for ease of reference.

LEMMA 6.3 (**Spectral norm of stochastic matrices**). *Let A be a $K \times K$ left (i.e., $A^{\mathsf{T}}\mathbb{1} = \mathbb{1}$), right (i.e., $A\mathbb{1} = \mathbb{1}$), or doubly-stochastic (i.e., $A^{\mathsf{T}}\mathbb{1} = A\mathbb{1} = \mathbb{1}$) matrix. Then, $\rho(A) = 1$ and, therefore, all eigenvalues of A lie inside the unit disc, i.e., $|\lambda(A)| \leq 1$.*

Proof: We prove the result for right-stochastic matrices; a similar argument applies to left or doubly-stochastic matrices. Let A be a right-stochastic matrix. Then, $A\mathbb{1} = \mathbb{1}$, so that $\lambda = 1$ is one of the eigenvalues of A. Moreover, for any matrix A, it holds

that $\rho(A) \leq \|A\|_\infty$, where $\|\cdot\|_\infty$ denotes the maximum absolute row sum of its matrix argument. But since all entries on each row of A add up to one, we have $\|A\|_\infty = 1$. Therefore, $\rho(A) \leq 1$. And since we already know that A has an eigenvalue at $\lambda = 1$, we conclude that $\rho(A) = 1$.

□

The above result asserts that the spectral radius of a stochastic matrix is equal to one and that A has an eigenvalue at $\lambda = 1$. The result, however, does not rule out the possibility of multiple eigenvalues at $\lambda = 1$, or even other (complex) eigenvalues with magnitude equal to one. If the stochastic matrix A happens to be *primitive*, then the Perron-Frobenius theorem mentioned before leads to a stronger characterization of the eigen-structure of A. In particular, it asserts that A has a single eigenvalue at one with multiplicity one, while all other eigenvalues have magnitude strictly less than one.

LEMMA 6.4 (**Useful properties of doubly-stochastic matrices**). *Let A be a $K \times K$ doubly-stochastic matrix. Then, the following properties hold:*

(a) $\rho(A) = 1$.
(b) $I_K - A$ *is symmetric and nonnegative-definite.*
(c) AA^{T} *and* $A^{\mathsf{T}}A$ *are doubly stochastic as well.*
(d) $\rho(AA^{\mathsf{T}}) = \rho(A^{\mathsf{T}}A) = 1$.
(e) The eigenvalues of AA^{T} *or* $A^{\mathsf{T}}A$ *are real and lie inside the interval* $[0, 1]$.
(f) $I_K - AA^{\mathsf{T}} \geq 0$ *and* $I_K - A^{\mathsf{T}}A \geq 0$.

Proof: Part (a) follows from Lemma 6.3. For part (b), since A is symmetric and $\rho(A) = 1$, we have that $\lambda(A) \in [-1, 1]$. That is, the eigenvalues of A are real and in the range $[-1, 1]$. It follows that $I - A$ is symmetric and also nonnegative-definite since its eigenvalues are in the range $[0, 2]$. For part (c), note that AA^{T} is symmetric and $AA^{\mathsf{T}}\mathbb{1} = A\mathbb{1} = \mathbb{1}$. Therefore, AA^{T} is doubly-stochastic. Likewise for $A^{\mathsf{T}}A$. Part (d) follows from part (a) since AA^{T} and $A^{\mathsf{T}}A$ are themselves doubly-stochastic matrices. For part (e), note that AA^{T} is symmetric and nonnegative-definite. Therefore, its eigenvalues are real and nonnegative. But since $\rho(AA^{\mathsf{T}}) = 1$, we must have $\lambda(AA^{\mathsf{T}}) \in [0, 1]$. Likewise for the matrix $A^{\mathsf{T}}A$. Part (f) follows from part (d).

□

When A happens to be additionally primitive, we get the following useful characterization of the nullspace of $I - A$.

LEMMA 6.5 (**Nullspace of $I - A$**). *Let A be a $K \times K$ doubly-stochastic and primitive matrix. Then, the nullspace of the matrix $I_K - A$ is spanned by $\mathbb{1}_K$:*

$$\text{nullspace}(I_K - A) = \text{span}(\mathbb{1}_K) \tag{6.18}$$

This means that every nonzero vector x such that $(I_K - A)x = 0$ is of the form $x = \alpha\mathbb{1}_K$, for some nonzero scalar α. Moreover, it also holds that

$$\text{nullspace}(I_K - A) \;=\; \text{nullspace}(I_K - A^2) \tag{6.19}$$

Proof: Let x be a vector in the nullspace of $I_K - A$ so that $(I_K - A)x = 0$, which is equivalent to $Ax = x$. Since A is primitive and doubly-stochastic, it has a single eigenvalue at one and $A\mathbb{1} = \mathbb{1}$. It follows that $x = \alpha\mathbb{1}$, for some α so that (6.18) is justified.

To establish (6.19), consider again any vector satisfying $(I_K - A)x = 0$. Then,

$$Ax = x \Longrightarrow A^2x = Ax = x \Longrightarrow (I_K - A^2)x = 0 \Longrightarrow x \in \mathcal{N}(I_K - A^2) \tag{6.20}$$

Conversely, the matrix A^2 is also doubly-stochastic and primitive. It therefore has a single eigenvalue at one and $A^2\mathbb{1} = \mathbb{1}$. Let x be any vector in the nullspace of $(I_K - A^2)$. Then, the equality $A^2x = x$ implies that $x = \alpha\mathbb{1}$ for some scalar α. This implies that x is also an eigenvector for A with eigenvalue one, which means $Ax = x$ so that $(I_K - A)x = 0$ and x lies in the nullspace of $(I_K - A)$.

□

Since the matrix $(I_K - A)$ is symmetric and nonnegative definite, we introduce the eigendecomposition

$$V \triangleq c(I_K - A) = U\Lambda U^{\mathsf{T}}, \qquad V^{1/2} \triangleq U\Lambda^{1/2}U^{\mathsf{T}} \tag{6.21}$$

for any scalar $c > 0$, and where U is a $K \times K$ orthogonal matrix ($UU^{\mathsf{T}} = U^{\mathsf{T}}U = I_K$) and Λ is a $K \times K$ diagonal matrix with nonnegative entries. Moreover, $\Lambda^{1/2}$ is a diagonal matrix with the nonnegative square-roots of the entries of Λ. We use the notation $V^{1/2}$ to refer to the "square-root" of the matrix V since $V^{1/2}(V^{1/2})^{\mathsf{T}} = V$. Observe that $V^{1/2}$ is a symmetric square-root since $V^{1/2} = (V^{1/2})^{\mathsf{T}}$ and, hence, $V = (V^{1/2})^2$. Since V has the same nullspace as $I_K - A$, it follows that $V^{1/2}$ and $(I_K - A)$ also have the same nullspace, i.e.,

$$(I_K - A)x = 0 \iff Vx = 0 \iff V^{1/2}x = 0 \tag{6.22}$$

Proof: Consider first any $x \in \mathcal{N}(V^{1/2})$. Then,

$$V^{1/2}x = 0 \Longrightarrow V^{1/2}\Big(V^{1/2}x\Big) = 0 \Longrightarrow Vx = 0 \Longrightarrow x \in \mathcal{N}(V) \tag{6.23}$$

Conversely, consider any $x \in \mathcal{N}(V)$. Then,

$$Vx = 0 \Longrightarrow x^{\mathsf{T}}Vx = 0 \Longrightarrow x^{\mathsf{T}}V^{1/2}\Big(V^{1/2}x\Big) = 0 \Longrightarrow \|V^{1/2}x\|^2 = 0 \Longrightarrow x \in \mathcal{N}(V^{1/2}) \tag{6.24}$$

□

6.3 PROBLEMS

6.1 Establish Lemma 6.1.

6.2 Establish (6.7).

6.3 Show that the Laplacian and Metropolis rules in Table 6.1 are doubly-stochastic.

7 Averaging over Graphs

In this chapter, we tackle a fundamental processing problem over graphs, namely coordination of a collection of agents to come to aggrement on an average value by relying soley on local interactions. While the resulting consensus problem has a number of applications in distributed and decentralized processing, the material in this chapter will primarily serve as a means of uncovering the implications of various graph properties explored in previous chapters as they relate to decentralized processing schemes. Additionally, as we will see in chapters further ahead, the averaging problem forms a critical building block of a number of more complex learning mechanisms. As such, a thorough understanding of the dynamics of the averaging problem over graphs forms a valuable foundation both for the understanding of existing decentralized algorithms, and the development of new schemes.

7.1 CENTRALIZED AVERAGING

We consider a collection of K vector-valued and potentially random and time-varying signals $\boldsymbol{g}_{k,i} \in \mathbb{R}^M$, and wish to efficiently compute the average:

$$\overline{\boldsymbol{g}}_i \triangleq \sum_{k=1}^{K} p_k \boldsymbol{g}_{k,i} \tag{7.1}$$

where $\{p_k\}_{k=1}^K$ denote scalar convex combination weights, such that $\sum_{k=1}^K p_k = 1$. If the collection of signals $\left\{\boldsymbol{g}_{k,i}\right\}_{k=1}^K$ are jointly available at some central location, and in the absence of computational constraints, implementing (7.1) is straightforward by simply averaging the available realizations. As we have seen in previous chapters, many learning problems in the context of multi-agent systems, and machine learning more broadly, boil down the problem of finding efficient ways to compute averages of the form (7.1), while being faced with constraints to computational resources or the availability of samples.

7.2 STATIC CONSENSUS

We consider a collection $\mathcal{N}$ of K agents, indexed by k, where each agent is assigned some vector-valued quantity $g_k \in \mathbb{R}^M$ as its "signal". The agents are arranged in a connected graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$. The objective of a consensus algorithm is to, in a decentralized manner, compute:

$$\{g_1, \ldots, g_K\} \Longrightarrow \overline{g} \triangleq \sum_{k=1}^{K} p_k g_k \tag{7.2}$$

Here, $\{p_k\}_{k=1}^K$ denote scalar convex combination weights, such that $\sum_{k=1}^K p_k = 1$. It is most common to let $p_k = \frac{1}{K}$, though we allow for arbitrary convex weights for generality.

7.2.1 Algorithm Development

Whenever studying the evolution of a collection of quantities over a network, it will be useful to introduce "network quantities" that summarize the state of the network in a compact form. We collect:

$$\mathcal{G} \triangleq \begin{pmatrix} g_1 \\ g_2 \\ \vdots \\ g_K \end{pmatrix} \tag{7.3}$$

It then follows that:

$$\overline{g} \triangleq \sum_{k=1}^{K} p_k g_k = \left(p^{\mathsf{T}} \otimes I_M\right) \mathcal{G} \tag{7.4}$$

If the network was fully connected, and every agent could exchange its signal with any other agent, each agent could directly compute:

$$w_k = \sum_{k=1}^{K} p_k g_k = \overline{g} \tag{7.5}$$

or in network quantities:

$$\mathcal{W} \triangleq \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_K \end{pmatrix} = \begin{pmatrix} \overline{g} \\ \overline{g} \\ \vdots \\ \overline{g} \end{pmatrix} = (\mathbb{1} \otimes I_M)\,\overline{g} = (\mathbb{1} \otimes I_M)\left(p^{\mathsf{T}} \otimes I_M\right)\mathcal{G} = \left(\mathbb{1}p^{\mathsf{T}} \otimes I_M\right)\mathcal{G} \tag{7.6}$$

The issue with this relation, of course, is the fact that averaging across the entire network requires a fully connected graph. We now exploit the spectral properties of weight matrices of connected graphs, established in Chapter 6, to construct

and iterative, but decentralized solutions. To this end, we associate with the graph $\mathcal{G}$ a left-stochastic, weighted adjacency matrix $A \in \mathbb{R}^{K\times K}$, such that:

$$a_{\ell k}\begin{cases} > 0, \text{ if } \ell \in \mathcal{N}_k, \\ = 0, \text{ otherwise.} \end{cases} \tag{7.7}$$

and

$$\sum_{\ell=1}^{K} a_{\ell k} = 1 \tag{7.8}$$

As long as A is primitive, it then follows from the Perron-Frobenius Theorem, that:

$$Ap = p \tag{7.9}$$

$$A^{\mathsf{T}}\mathbb{1} = \mathbb{1} \tag{7.10}$$

$$\lim_{i\to\infty} A^i = p\mathbb{1}^{\mathsf{T}} \tag{7.11}$$

where p denotes the Perron eigenvector of A (see Chapter 6 for a detailed discussion on the spectral properties of weight matrices of (strongly-)connected graphs). We can then write (7.6) equivalently as:

$$\mathcal{W} = \left(\mathbb{1}p^{\mathsf{T}} \otimes I_M\right)\mathcal{G} = \left(\lim_{i\to\infty}\left(A^{\mathsf{T}}\right)^i \otimes I_M\right)\mathcal{G} = \lim_{i\to\infty}\left(\mathcal{A}^{\mathsf{T}}\right)^i \mathcal{G} \tag{7.12}$$

where we defined:

$$\mathcal{A} \triangleq A \otimes I_M \tag{7.13}$$

While we are not able to implement (7.12) directly, the relation suggests a recursive implementation. We initialize $\mathcal{W}_0 = \mathcal{G}$, and iterate:

$$\mathcal{W}_i = \mathcal{A}^{\mathsf{T}}\,\mathcal{W}_{i-1} \tag{7.14}$$

so that:

$$\lim_{i\to\infty} \mathcal{W}_i = \left(\mathcal{A}^{\mathsf{T}}\right)^i \mathcal{G} = \left(\mathbb{1}p^{\mathsf{T}} \otimes I_M\right)\mathcal{G} \tag{7.15}$$

as desired. Unpacking the network recursion into agent-specific, local recursions yields the *consensus* algorithm:

$$w_{k,i} = \sum_{\ell\in\mathcal{N}_k} a_{\ell k} w_{\ell,i-1} \tag{7.16}$$

7.2.2 Convergence Analysis

While relation (7.15) indicates that all agents in the network will come to consensus on the (weighted) average value $\overline{g}$ *in the limit* as $i \to \infty$, we are now interested in the transient dynamics of the collection of iterates $w_{k,i}$ on their way to consensus. We establish the following theorem:

THEOREM 7.1 (**Linear convergence of the consensus algorithm**). *Each agent $w_{k,i}$ following the consensus protocol (7.16) converges linearly to the weighted mean $\overline{g} \triangleq \sum_{k=1}^{K} p_k g_k$:*

$$\sum_{k=1}^{K} \|\overline{g} - w_{k,i}\|^2 \leq \sigma_2(A)^{2i} \cdot \sum_{k=1}^{K} \|\overline{g} - s_k\|^2, \tag{7.17}$$

where the rate of convergence $\sigma_2(A)$ is dictated by the second-largest singular value of the combination matrix A:

$$\sigma_2(A) = \left\| 1p^{\mathsf{T}} \otimes I_M - \mathcal{A}^{\mathsf{T}} \right\| \tag{7.18}$$

Proof: Let us first examine the evolution of the weighted mean of iterates across the network $\overline{w}_i = \sum_{k=1}^{K} p_k w_{k,i}$. We have:

$$\begin{aligned}
\overline{w}_i &= \sum_{k=1}^{K} p_k w_{k,i} \\
&= \left(p^{\mathsf{T}} \otimes I_M\right) \mathcal{w}_i \\
&= \left(p^{\mathsf{T}} \otimes I_M\right) \mathcal{A}^{\mathsf{T}} \mathcal{w}_{i-1} \\
&= \left((Ap)^{\mathsf{T}} \otimes I_M\right) \mathcal{w}_{i-1} \\
&= \left(p^{\mathsf{T}} \otimes I_M\right) \mathcal{w}_{i-1} \\
&= \sum_{k=1}^{K} p_k w_{k,i-1} \\
&= \overline{w}_{i-1}
\end{aligned} \tag{7.19}$$

We observe that, as a consequence of the fact that p is an eigenvector for the combination matrix A, the weighted mean $\overline{w}_i$ is constant over time. We can conclude by iterating that:

$$\overline{w}_i = \overline{w}_{i-1} = \ldots = \overline{w}_0 \overset{(a)}{=} \sum_{k=1}^{K} p_k g_k \tag{7.20}$$

where step (a) follows since each local iterate $w_{k,0}$ is initialized at the local signal g_k. It follows that the weighted mean, which we also refer to as a *centroid*, is fixed, while the local iterates $w_{k,i}$ are scattered around it. We can then use the centroid as a reference point, and study to evolution of the network around it. Subtracting the consensus recusion (7.14) from the augmented mean $\mathbb{1}_K \otimes \overline{w}_i$, we find:

$$\begin{aligned}
\mathbb{1}_K \otimes \overline{w}_i - \mathcal{w}_i &= \mathbb{1}_K \otimes \overline{w}_i - \mathcal{A}^{\mathsf{T}} \mathcal{w}_{i-1} \\
&= \mathbb{1}_K \otimes \overline{w}_{i-1} - \mathcal{A}^{\mathsf{T}} \mathcal{w}_{i-1} \\
&= \mathbb{1}_K \otimes \left(\left(p^{\mathsf{T}} \otimes I_K\right) \mathcal{w}_{i-1}\right) - \mathcal{A}^{\mathsf{T}} \mathcal{w}_{i-1} \\
&= \left(\mathbb{1}p^{\mathsf{T}} \otimes I_M\right) \mathcal{w}_{i-1} - \mathcal{A}^{\mathsf{T}} \mathcal{w}_{i-1} \\
&= \left(\mathbb{1}p^{\mathsf{T}} \otimes I_M - \mathcal{A}^{\mathsf{T}}\right) \mathcal{w}_{i-1} \\
&\overset{(a)}{=} \left(\mathbb{1}p^{\mathsf{T}} \otimes I_M - \mathcal{A}^{\mathsf{T}}\right) (\mathcal{w}_{i-1} - \mathbb{1} \otimes \overline{w}_{i-1})
\end{aligned} \tag{7.21}$$

where (a) follows from:

$$\begin{aligned}
&\left(\mathbb{1}p^{\mathsf{T}} \otimes I_M - \mathcal{A}^{\mathsf{T}}\right)(\mathbb{1} \otimes \overline{w}_{i-1}) \\
&= \left(\mathbb{1}p^{\mathsf{T}} \otimes I_M\right)(\mathbb{1} \otimes \overline{w}_{i-1}) - \left(A^{\mathsf{T}} \otimes I_M\right)(\mathbb{1} \otimes \overline{w}_{i-1}) \\
&= \mathbb{1}p^{\mathsf{T}}\mathbb{1} \otimes \overline{w}_{i-1} - A^{\mathsf{T}}\mathbb{1} \otimes \overline{w}_{i-1} \\
&= 0
\end{aligned} \tag{7.22}$$

Returning to (7.21), upon taking norms we find:

$$\begin{aligned}
\|\mathcal{w}_i - \mathbb{1} \otimes \overline{w}_i\| &= \left\|\left(\mathbb{1}p^{\mathsf{T}} \otimes I_M - \mathcal{A}^{\mathsf{T}}\right)(\mathcal{w}_{i-1} - \mathbb{1} \otimes \overline{w}_{i-1})\right\| \\
&\leq \left\|\mathbb{1}p^{\mathsf{T}} \otimes I_M - \mathcal{A}^{\mathsf{T}}\right\| \|\mathcal{w}_{i-1} - \mathbb{1} \otimes \overline{w}_{i-1}\|
\end{aligned} \tag{7.23}$$

and hence after taking squares and iterating:

$$\begin{aligned}
\|\mathcal{w}_i - \mathbb{1} \otimes \overline{g}\|^2 &= \|\mathcal{w}_i - \mathbb{1} \otimes \overline{w}_i\|^2 \\
&\leq \left\|\mathbb{1}p^{\mathsf{T}} \otimes I_M - \mathcal{A}^{\mathsf{T}}\right\|^{2i} \|\mathcal{w}_0 - \mathbb{1} \otimes \overline{g}\|^2 \\
&\overset{(a)}{=} \sigma_2(A)^{2i} \cdot \left(\sum_{k=1}^{K} \|\overline{g} - g_k\|^2\right)
\end{aligned} \tag{7.24}$$

where (a) follows since:

$$\left\|\mathbb{1}p^{\mathsf{T}} \otimes I_M - \mathcal{A}^{\mathsf{T}}\right\| = \sigma_1\left(\mathbb{1}p^{\mathsf{T}} \otimes I_M - \mathcal{A}^{\mathsf{T}}\right) = \sigma_1\left(\mathbb{1}p^{\mathsf{T}} - A^{\mathsf{T}}\right) = \sigma_2\left(A^{\mathsf{T}}\right) = \sigma_2(A) \tag{7.25}$$

□

Theorem 7.1 clarifies the manner in which the combination matrix A chosen to perform repeated local averaging in (7.16) influences both the limiting points, as well as the transient behavior of the local iterates $w_{k,i}$. In particular, the limiting point $\overline{g}$ is given by the weighted average of the signals g_k chosen to initialize the consensus recursion, with weights, that depend on the Perron eigenvector $p = Ap$. In other words, the elements $\{p_k\}$ of the Perron vector p quantify the *influence* that agent each agent can exert over the limiting behavior of the network. For this reason the elements of p a also referred to as *agent centralities*. The transient behavior on the other hand, and in particular the rate at which the network approaches its limiting value, is captured in the second-largest singular value of the combination matrix A. For this reason $\sigma_2(A)$ is commonly denoted as the *mixing rate* of the graph.

Definition 7.1 (Centralities and Mixing Rate). For a graph $\mathcal{G}$ with adjacency matrix A, we refer to the elements of the Perron eigenvector:

$$Ap = p \tag{7.26}$$

as the *agent centrality*, and the second-largest singular value of the matrix A:

$$\sigma_2(A) = \left\|A - p\mathbb{1}^{\mathsf{T}}\right\| \tag{7.27}$$

as the *mixing rate*.

7.3 Combination Policy Design

The result of Theorem 7.1 shows that the consensus dynamics (7.16) depend strongly on the spectral properties of the combination matrix A, and in particular on the Perron vector of agent centralities p, and the mixing rate $\sigma_2(A)$. It is then natural formulate design problems in the pursuit of suitable or optimal combination policies. We will present several formulations for the pure consensus algorithm here, and present further policies in later chapters as the consensus mechanism is used as a building block in more elaborate structures.

7.3.1 Perron vector design

The Perron vector p of the combination matrix A quantifies the influence that any particular agent has over the evolution of the network. In many situations in practice, one would like the consensus algorithm to converge to a *particular* weighted average:

$$\overline{g} = \sum_{k=1}^{K} p_k g_k \tag{7.28}$$

for some particular weights $\{p_k\}$. The fact that the limiting point is influenced by the combination policy might, at first sight, appear as a limitation of the algorithm. However, it is in fact possible to construct the combination policy A, in a decentralized manner, to yield any desired Perron vector p, as long as the underlying communication graph is undirected as we demonstrate in the example below.

Example 7.1 (**Ensuring a desired Perron vector over undirected graphs**) We consider an undirected graph with $k \in \mathcal{N}_\ell \Leftrightarrow \ell \in \mathcal{N}_k$. Then, the following choice of combination weights $a_{\ell k}$ yields any desired Perron vector $p = Ap$:

$$a_{\ell k} = \begin{cases} 0, & \text{if } \ell \notin \mathcal{N}_k, \\ p_\ell, & \text{if } \ell \in \mathcal{N}_k \backslash \ell, \\ 1 - \sum_{m \in \mathcal{N}_k \backslash \ell} a_{mk}, & \text{when } \ell = k. \end{cases} \tag{7.29}$$

By definition, we have $a_{\ell k} = 0$ whenever $\ell \notin \mathcal{N}_k$ and $a_{\ell k} = p_\ell > 0$ when $\ell \in \mathcal{N}_k \backslash \ell$. For $\ell = k$, i.e., a_{kk} we have:

$$a_{kk} = 1 - \sum_{m \in \mathcal{N}_k \backslash k} a_{mk} = 1 - \sum_{m \in \mathcal{N}_k \backslash k} p_m \geq 1 - \sum_{m \neq k} p_m = 1 - (1 - p_k) = p_k > 0 \tag{7.30}$$

We conclude that A satisfies Eq. (1) and is primitive. We now proceed to verify that $Ap = p$. Note that if the underlying graph topology is undirected, we can impose $k \in \mathcal{N}_\ell \Leftrightarrow \ell \in \mathcal{N}_k$. Then, if $k \in \mathcal{N}_\ell \Leftrightarrow \ell \in \mathcal{N}_k$, we have:

$$a_{\ell k} p_k = p_\ell p_k = p_\ell a_{k\ell} \tag{7.31}$$

On the other hand, when $k \notin \mathcal{N}_\ell \Leftrightarrow \ell \notin \mathcal{N}_k$, it also holds that:

$$a_{\ell k} p_k = 0 \cdot p_k = p_\ell \cdot 0 = p_\ell a_{k\ell} \tag{7.32}$$

so that A satisfies the balance condition

$$a_{\ell k} p_k = p_\ell a_{k\ell} \tag{7.33}$$

for all k, ℓ. It then follows that:

$$\sum_{k=1}^{N} a_{\ell k} p_k = \sum_{k=1}^{N} a_{k\ell} p_\ell = \left(\sum_{k=1}^{N} a_{k\ell} \right) p_\ell = p_\ell \tag{7.34}$$

which implies that $Ap = p$. We conclude that *any* choice of convex $\{p_k\}$ can be induced over undirected graphs using a simple, decentralized protocol which relies only on the additional exchange of scalar quantities over neighborhoods.

7.4 DYNAMIC CONSENSUS

As the results from the previous section showed, the consensus recursion (7.16) allows for a collection of agents to effectively come to agreement on a (weighted) mean of a static signal $\{g_k\}_{k=1}^K$, dispersed across a network of agents. The resulting algorithm initializes estimates w_k at the signal g_k, and then averages repeatedly over neighborhoods until convergence. In many situations of practical interest, and most settings considered in this text, the signal of interest will instead be *time-varying* and *random*, taking the form $\boldsymbol{g}_{k,i}$. Here, the subscript i indicates the time-varying nature of $\boldsymbol{g}_{k,i}$, while it's bolface font indicates that it is now a random variable. It is hence desirable to design consensus algorithms that are able to process streaming realizations of a random signal $\boldsymbol{g}_{k,i}$, and track drifts in its distribution. To this end, it will be useful to formulate an optimization problem:

$$\overline{\mathrm{g}}_i = \arg\min_{\boldsymbol{w}} \sum_{k=1}^{K} p_k \|\boldsymbol{w}_k - \boldsymbol{g}_{k,i}\|^2, \text{ subject to } \boldsymbol{w}_k = \boldsymbol{w}_\ell \ \forall \ k, \ell. \tag{7.35}$$

To verify that (7.35) indeed defines the weighted average $\overline{\boldsymbol{g}}_i$, we note that under the constraint $\boldsymbol{w}_k = \boldsymbol{w}_\ell \ \forall \ k, \ell$, (7.35) is equivalent to:

$$\overline{\boldsymbol{g}}_i = \arg\min_{\boldsymbol{w}} \sum_{k=1}^{K} p_k \|\boldsymbol{w} - \boldsymbol{g}_{k,i}\|^2 \tag{7.36}$$

By differentiating, we find:

$$\sum_{k=1}^{K} p_k (\boldsymbol{w} - \boldsymbol{g}_{k,i}) = 0 \Longleftrightarrow \boldsymbol{w} = \sum_{k=1}^{K} p_k \boldsymbol{g}_{k,i} \tag{7.37}$$

This fact establishes that (7.35) is equivalent to simply pursuing the mean directly as in (7.37). However, the same drawbacks as discussed in Section 7.1 continue to hold: Evaluating $\overline{\boldsymbol{g}}_i$ directly, whether through (7.37), or by solving the optimization (7.35), require aggregation of all signals $\boldsymbol{g}_{k,i}$ at every time instance.

We will now examine algorithms for dynamic consensus by reformulating (7.35) in a manner which makes it more amenable to decentralized computations. To this end, we note that, as long as the graph $\mathcal{G}$ is connected, the constraint $\boldsymbol{w}_k = \boldsymbol{w}_\ell \; \forall \; k, \ell$ can be simplified to the equivalent constraint $\boldsymbol{w}_k = \boldsymbol{w}_\ell \; \forall \; \ell \in \mathcal{N}_k$, where $\mathcal{N}_k$ denotes the neighborhood of node k. This is because for any connected graph, there exists a sequence of edges leading from any node k, to an arbitrary node ℓ, yielding a string of equality constraints leading from node k all the way to node ℓ. It then follows that (7.35) is equivalent to:

$$\overline{\mathcal{G}}_i = \arg\min_{\mathcal{W}} \sum_{k=1}^{K} p_k \|\boldsymbol{w}_k - \boldsymbol{g}_{k,i}\|^2, \text{ subject to } \boldsymbol{w}_k = \boldsymbol{w}_\ell \; \forall \; \ell \in \mathcal{N}_k. \tag{7.38}$$

Furthermore, we can write equivalently:

$$\overline{\mathcal{G}}_i = \arg\min_{\mathcal{W}} \sum_{k=1}^{K} p_k \|\boldsymbol{w}_k - \boldsymbol{w}_{k,i}\|^2, \text{ subject to } \sum_{k=1}^{K}\sum_{\ell=1}^{K} c_{\ell k} \|\boldsymbol{w}_k - \boldsymbol{w}_\ell\|^2 = 0. \tag{7.39}$$

where $c_{\ell k}$ are non-negative weights with:

$$c_{\ell k} = \begin{cases} > 0 & \text{if } \ell \in \mathcal{N}_k, \\ = 0 & \text{otherwise.} \end{cases} \tag{7.40}$$

To verify that (7.38) and (7.39) are equivalent, we observe that:

$$\sum_{k=1}^{K}\sum_{\ell=1}^{K} c_{\ell k} \|\boldsymbol{w}_k - \boldsymbol{w}_\ell\|^2 = 0 \Longleftrightarrow \|\boldsymbol{w}_k - \boldsymbol{w}_\ell\|^2 = 0 \; \forall \ell \in \mathcal{N}_k \Longleftrightarrow \boldsymbol{w}_k = \boldsymbol{w}_\ell, \forall \ell \in \mathcal{N}_k. \tag{7.41}$$

7.4.1 Penalty-Based Dynamic Consensus

We begin with a penalty-based extension of the static consensus algorithm (7.16) to stochastic and dynamic environments by means of penalty optimization. To this end, rather than solve (7.39) with the exact constraint $\sum_{k=1}^{K}\sum_{\ell=1}^{K} c_{\ell k}\|\boldsymbol{w}_k - \boldsymbol{w}_\ell\|^2 = 0$, we penalize deviation from the constraint as in:

$$\overline{\mathcal{G}}_i^{\eta} = \arg\min_{\mathcal{W}} \sum_{k=1}^{K} p_k \|\boldsymbol{w}_k - \boldsymbol{g}_{k,i}\|^2 + \eta \sum_{\ell=1}^{K} c_{\ell k} \|\boldsymbol{w}_k - \boldsymbol{w}_\ell\|^2 \tag{7.42}$$

where $\eta > 0$ denotes a new penalty parameter. It is important to note that, while (7.35) through (7.39) were *equivalent* reformulations of the consensus problem, relation (7.42) is only an approximation for finite η. Letting $\eta \to \infty$ ensures $\sum_{\ell=1}^{K} c_{\ell k}\|\boldsymbol{w}_k - \boldsymbol{w}_\ell\|^2 = 0$, and hence exact consensus. In practice one chooses a finite, albeit large value of η. In particular, this means that the minimizing argument $\overline{\mathcal{G}}_i^{\eta}$ is a function of the penalty parameter η, and in general distinct from the true network average $\overline{\mathcal{G}}_i$. In the sequel, we will quantify this difference,

and present an algorithm for pursuing $\overline{\mathcal{G}}_i^{\eta}$. To this end, it will be useful to write the penalized cost (7.42) in terms of network level quantities. We introduce:

$$P = \begin{pmatrix} p_1 & 0 & \cdots & 0 \\ 0 & p_2 & \cdots & 0 \\ 0 & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & p_K \end{pmatrix} = \mathrm{diag}\{p\} \tag{7.43}$$

$$\mathcal{P} = P \otimes I_M \tag{7.44}$$

Then:

$$\sum_{k=1}^{K} p_k \|\boldsymbol{w}_k - \boldsymbol{g}_{k,i}\|^2 = \|\,\mathcal{w} - \mathcal{G}_i\|_{\mathcal{P}}^2 \tag{7.45}$$

For the second term in (7.42), we can verify (see Problem 7.1), that:

$$\eta \sum_{k=1}^{K} \sum_{\ell \in \mathcal{N}_k} c_{\ell k} \|\boldsymbol{w}_k - \boldsymbol{w}_\ell\|^2 = \eta\, \mathcal{w}^{\mathsf{T}}\, \mathcal{L}\, \mathcal{w} \tag{7.46}$$

where $\mathcal{L} = L \otimes I_M$ and we defined:

$$L = \mathrm{diag}\,\{C\mathbb{1}\} - C \tag{7.47}$$

Hence, problem (7.42) is equivalent to:

$$\overline{\mathcal{G}}_i^{\eta} = \arg\min_{\mathcal{w}} \frac{1}{2}\|\,\mathcal{w} - \mathcal{G}_i\|_{\mathcal{P}}^2 + \frac{\eta}{2}\,\mathcal{w}^{\mathsf{T}}\,\mathcal{L}\,\mathcal{w} \tag{7.48}$$

where we added a factor of $\frac{1}{2}$ to simplify the recursions that follow, which does not affect the minimizing argument. Note that this objective function is time-varying. Nevertheless, we can pursue its minimizing argument via (online) gradient descent, pre-conditioned by the positive-definite matrix $\mathcal{P}^{-1}$:

$$\begin{aligned} \mathcal{w}_i &= \mathcal{w}_{i-1} - \mu\mathcal{P}^{-1}\left(\mathcal{P}\,(\mathcal{w}_{i-1} - \mathcal{G}_i) + \mu\eta\mathcal{L}\,\mathcal{w}_{i-1}\right) \\ &= \mathcal{w}_{i-1} - \mu\,(\mathcal{w}_{i-1} - \mathcal{G}_i) - \mu\eta\mathcal{P}^{-1}\mathcal{L}\,\mathcal{w}_{i-1} \\ &= \left((1-\mu)I - \mu\eta\mathcal{P}^{-1}\mathcal{L}\right)\mathcal{w}_{i-1} + \mu\mathcal{G}_i \end{aligned} \tag{7.49}$$

If we define:

$$\mathcal{A}^{\mathsf{T}} \triangleq I - \mu\eta\mathcal{P}^{-1}\mathcal{L} \tag{7.50}$$

we can equivalently write:

$$\mathcal{w}_i = \left(\mathcal{A}^{\mathsf{T}} - \mu I\right)\mathcal{w}_{i-1} + \mu\mathcal{G}_i = \mathcal{A}^{\mathsf{T}}\,\mathcal{w}_{i-1} + \mu\,(\mathcal{G}_i - \mathcal{w}_{i-1}) \tag{7.51}$$

It is instructive to examine the spectral structure of $\mathcal{A}^{\mathsf{T}}$ defined as in (7.50). Since $\mathcal{L}\mathbb{1} = 0$, we have:

$$A^{\mathsf{T}}\mathbb{1} = \left(I - \mu\eta P^{-1}L\right)\mathbb{1} = \mathbb{1} - \mu\eta P^{-1}L\mathbb{1} = \mathbb{1} \tag{7.52}$$

$$Ap = \left(I - \mu\eta LP^{-1}\right)p = p - \mu\eta LP^{-1}p = p - \mu\eta L\mathbb{1} = p \tag{7.53}$$

We conclude that $\mathbb{1}$ and p are left and right eigenvectors corresponding to an eigenvalue at one respectively. Returning to node level quantities, we arrive at:

$$\boldsymbol{w}_{k,i} = \sum_{\ell \in \mathcal{N}_k} a_{\ell k} \boldsymbol{w}_{\ell,i-1} + \mu \left(\boldsymbol{g}_{k,i} - \boldsymbol{w}_{k,i-1} \right) \tag{7.54}$$

Analysis

To begin with, we will examine the performance of the penalty-based dynamic consensus algorithm (7.51) in the case when $\boldsymbol{g}_{k,i} \triangleq g_k$ is deterministic and constant. We then have:

$$\mathcal{W}_i = \left(\mathcal{A}^{\mathsf{T}} - \mu I \right) \mathcal{W}_{i-1} + \mu \mathcal{G} \tag{7.55}$$

Iterating we find:

$$\mathcal{W}_i = \left(\mathcal{A}^{\mathsf{T}} - \mu I \right)^i \mathcal{W}_0 + \mu \left(\sum_{j=0}^{i-1} \left(\mathcal{A}^{\mathsf{T}} - \mu I \right)^j \right) \mathcal{G} \tag{7.56}$$

In the limit, we then have:

$$\lim_{i \to \infty} \mathcal{W}_i = \mu \left(\sum_{j=0}^{i-1} \left(\mathcal{A}^{\mathsf{T}} - \mu I \right)^j \right) \mathcal{G} = \mu \left(I - \left(\mathcal{A}^{\mathsf{T}} - \mu I \right) \right)^{-1} \mathcal{G} = \mu \left((1+\mu) I - \mathcal{A}^{\mathsf{T}} \right)^{-1} \mathcal{G} \tag{7.57}$$

To evaluate this expression more precisely, we will again exploit the spectral properties of the combination matrix A, established in 6. In particular, $A = V_\epsilon J V_\epsilon^{-1}$ is guaranteed to have a particular Jordan decomposition, given by:

$$V_\epsilon = \begin{bmatrix} p & V_R \end{bmatrix}, \quad J = \begin{bmatrix} 1 & 0 \\ 0 & J_\epsilon \end{bmatrix}, \quad V_\epsilon^{-1} = \begin{bmatrix} \mathbb{1}^{\mathsf{T}} \\ V_L^{\mathsf{T}} \end{bmatrix} \tag{7.58}$$

We then have:

$$
\begin{aligned}
&\lim_{i\to\infty} \mathcal{W}_i \\
&= \mu\left((1+\mu)I - \left(\mathcal{V}_\epsilon{}^{-1}\right)^{\mathsf{T}}\mathcal{J}^{\mathsf{T}}\mathcal{V}_\epsilon{}^{\mathsf{T}}\right)^{-1}\mathcal{G} \\
&= \mu\left((1+\mu)\left(\mathcal{V}_\epsilon{}^{-1}\right)^{\mathsf{T}}\mathcal{V}_\epsilon{}^{\mathsf{T}} - \left(\mathcal{V}_\epsilon{}^{-1}\right)^{\mathsf{T}}\mathcal{J}^{\mathsf{T}}\mathcal{V}_\epsilon{}^{\mathsf{T}}\right)^{-1}\mathcal{G} \\
&= \mu\Big(\left(\mathcal{V}_\epsilon{}^{-1}\right)^{\mathsf{T}}\left((1+\mu)I - \mathcal{J}^{\mathsf{T}}\right)\mathcal{V}_\epsilon{}^{\mathsf{T}}\Big)^{-1}\mathcal{G} \\
&= \mu\left(\mathcal{V}_\epsilon{}^{-1}\right)^{\mathsf{T}}\left((1+\mu)I - \mathcal{J}^{\mathsf{T}}\right)^{-1}\mathcal{V}_\epsilon{}^{\mathsf{T}}\,\mathcal{G} \\
&= \mu\left(\mathcal{V}_\epsilon{}^{-1}\right)^{\mathsf{T}}\begin{bmatrix} (1+\mu)I_M - I_M & 0 \\ 0 & (1+\mu)I_M - \mathcal{J}_\epsilon^{\mathsf{T}} \end{bmatrix}^{-1}\mathcal{V}_\epsilon{}^{\mathsf{T}}\,\mathcal{G} \\
&= \mu\left(\mathcal{V}_\epsilon{}^{-1}\right)^{\mathsf{T}}\begin{bmatrix} \mu^{-1}I_M & 0 \\ 0 & \left((1+\mu)I_M - \mathcal{J}_\epsilon^{\mathsf{T}}\right)^{-1} \end{bmatrix}\mathcal{V}_\epsilon{}^{\mathsf{T}}\,\mathcal{G} \\
&= \begin{bmatrix} \mathbb{1}\otimes I_M & \mathcal{V}_L \end{bmatrix}\begin{bmatrix} I_M & 0 \\ 0 & \mu\left((1+\mu)I_M - \mathcal{J}_\epsilon^{\mathsf{T}}\right)^{-1} \end{bmatrix}\begin{bmatrix} p^{\mathsf{T}}\otimes I_M \\ \mathcal{V}_R^{\mathsf{T}} \end{bmatrix}\mathcal{G} \\
&= \left(\mathbb{1}p^{\mathsf{T}}\otimes I_M\right)\mathcal{G} + \mu\mathcal{V}_L\left((1+\mu)I_M - \mathcal{J}_\epsilon^{\mathsf{T}}\right)^{-1}\mathcal{V}_R^{\mathsf{T}}\,\mathcal{G} \\
&= \mathcal{G} + \mu\mathcal{V}_L\left((1+\mu)I_M - \mathcal{J}_\epsilon^{\mathsf{T}}\right)^{-1}\mathcal{V}_R^{\mathsf{T}}\,\mathcal{G}
\end{aligned}
\tag{7.59}
$$

We rearrange and find:

$$
\sum_{k=1}^{K}\left\|\lim_{i\to\infty} w_{k,i} - g_k\right\|^2 = \left\|\lim_{i\to\infty}\mathcal{W}_i - \mathcal{G}\right\|^2 \le \mu^2\|\mathcal{V}_L\|^2\left\|\left((1+\mu)I_M - \mathcal{J}_\epsilon^{\mathsf{T}}\right)^{-1}\right\|^2\|\mathcal{V}_R^{\mathsf{T}}\|^2 \tag{7.60}
$$

We observe that, even when the signal g_k is constant, a residual error, which is proportional to μ^2 remains. It can be gradually reduced, at the expense of performance in dynamic environments, until we recover the static consensus algorithm as $\mu \to 0$. Despite the persistent steady-state error, as we will see in later chapters, a number of algorithms for decentralized optimization and learning are based on the penalty-based consensus implementation.

7.4.2 Dynamic Consensus

The previous section motivates the question, whether it is possible to design a consensus algorithm over a graph, which is able to average a constant signal perfectly, while having some ability to track slowly time varying signals. It turns out that this is possible, though the derivation is slightly more involved, and will be presented in a future chapter. For the time being, we simply list the exact dynamic consensus algorithm, which takes the form:

$$
\boldsymbol{w}_{k,i} = \sum_{\ell\in\mathcal{N}_k} a_{\ell k}\boldsymbol{w}_{\ell,i-1} + \boldsymbol{g}_{k,i} - \boldsymbol{g}_{k,i-1} \tag{7.61}
$$

or in network level quantities:

$$\boldsymbol{\mathcal{W}}_i = \mathcal{A}^{\mathsf{T}}\,\boldsymbol{\mathcal{W}}_{i-1} + \mathcal{G}_i - \mathcal{G}_{i-1} \tag{7.62}$$

We begin by establishing the ability of the dynamic consensus (7.62) to compute exactly the mean of a constant signal over the graph. To this end, let $\mathcal{G}_i \triangleq \mathcal{G}$, and we find:

$$\boldsymbol{\mathcal{W}}_i = \mathcal{A}^{\mathsf{T}}\,\boldsymbol{\mathcal{W}}_{i-1} + \mathcal{G} - \mathcal{G} = \mathcal{A}^{\mathsf{T}}\,\boldsymbol{\mathcal{W}}_{i-1} \tag{7.63}$$

which corresponds precisely to the static consensus algorithm (7.14). Under the initialization $\boldsymbol{\mathcal{W}}_0 \triangleq \mathcal{G}$, we conclude that the evolution of the dynamic consensus algorithm (7.62) matches that of the static consensus algorithm (7.14), provided that the signal of interest $\mathcal{G}_{i-1}$ is constant. The question now is whether, and how well, the dynamic variant of the consensus algorithms is able to track the mean of the signal of interest when it is time-varying. To quantify the tracking performance, we will employ the simple random-walk model:

$$\mathcal{G}_i = \mathcal{G}_{i-1} + \boldsymbol{\mathcal{V}}_i \tag{7.64}$$

where $\boldsymbol{\mathcal{V}}_i \in \mathbb{R}^{MK}$ denotes a random driving variable with mean $\mu_v = \mathbb{E}\boldsymbol{\mathcal{V}}_i$ and variance $\sigma_v^2 = \mathbb{E}\|\boldsymbol{\mathcal{V}}_i - \mu_v\|^2$. We note that, while the driving term $\boldsymbol{\mathcal{V}}_i$ is presumed stationary, the resulting dynamics of $\mathcal{G}_i$ are not. Under this model, we have:

$$\boldsymbol{\mathcal{W}}_i = \mathcal{A}^{\mathsf{T}}\,\boldsymbol{\mathcal{W}}_{i-1} + \mathcal{G}_i - \mathcal{G}_{i-1} = \mathcal{A}^{\mathsf{T}}\,\boldsymbol{\mathcal{W}}_{i-1} + \boldsymbol{\mathcal{V}}_i \tag{7.65}$$

We proceed similarly to the analysis of the static consensus algorithm, and decompose the evolution of the local estimates into the network centroid $\boldsymbol{w}_{c,i} \triangleq \sum_{k=1}^{K} p_k \boldsymbol{w}_{k,i}$ and the deviations from the centroid $\boldsymbol{w}_{k,i} - \boldsymbol{w}_{c,i}$. We begin with the centroid, which evolves according to:

$$\boldsymbol{w}_{c,i} = \boldsymbol{w}_{c,i-1} + \sum_{k=1}^{K} p_k \boldsymbol{v}_{k,i} \tag{7.66}$$

In comparison, for the mean of the time varying signal $\mathcal{G}_i$, we have:

$$\begin{aligned}
\boldsymbol{g}_{c,i} &\triangleq \sum_{k=1}^{K} p_k \boldsymbol{g}_{k,i} = \left(p^{\mathsf{T}} \otimes I_M\right) \mathcal{G}_i \\
&\overset{(7.64)}{=} \left(p^{\mathsf{T}} \otimes I_M\right) \left(\mathcal{G}_{i-1} + \boldsymbol{\mathcal{V}}_i\right) \\
&= \sum_{k=1}^{K} p_k \boldsymbol{g}_{k,i-1} + \sum_{k=1}^{K} p_k \boldsymbol{v}_{k,i} \\
&= \boldsymbol{g}_{c,i-1} + \sum_{k=1}^{K} p_k \boldsymbol{v}_{k,i}
\end{aligned} \tag{7.67}$$

Comparing the expression (7.66) for $\boldsymbol{w}_{c,i}$ with (7.67) for $\boldsymbol{g}_{c,i}$, we find that the recursions match exactly. We can hence conclude that, as long as the $\{\boldsymbol{w}_{k,0}\}_{k=1}^{K}$

are initialized appropriately, namely to $\boldsymbol{w}_{k,0} = \boldsymbol{g}_{k,0}$, we have with probability one:

$$\boldsymbol{w}_{c,i} = \boldsymbol{g}_{c,i} \Longleftrightarrow \sum_{k=1}^{K} p_k \boldsymbol{w}_{k,i} = \sum_{k=1}^{K} p_k \boldsymbol{g}_{k,i} \tag{7.68}$$

This fact is analogous to relation (7.19) for the static consensus algorithm, with the important advantage that the correction term $\mathcal{G}_{i-1} - \mathcal{G}_{i-2}$ in (7.62) allows the dynamic algorithm to track a *time-varying* centroid. Perhaps surprisingly, the network centroid $\boldsymbol{w}_{c,i}$ under this construction tracks the signal centroid $\boldsymbol{g}_{c,i}$ perfectly, and with probablility one, independently of the power or variance of the random variable $\boldsymbol{v}_i$ driving the random walk. This fact of course does not imply that any given agent $\boldsymbol{w}_{k,i}$ is necessarily able to track the signal centroid $\boldsymbol{g}_{c,i}$ well. We now proceed to study the deviation of local estimates $\boldsymbol{w}_{k,i}$ around the network centroid $\boldsymbol{w}_{c,i}$ under the random walk model (7.64) and the dynamic consensus algorithm (7.62) to establish this stronger result. We have:

$$\begin{aligned}
&\mathcal{W}_i - \mathbb{1} \otimes \boldsymbol{w}_{c,i} \\
&= \mathcal{W}_i - \left(\mathbb{1}p^{\mathsf{T}} \otimes I_M\right) \mathcal{W}_i \\
&= \mathcal{A}^{\mathsf{T}} \mathcal{W}_{i-1} + \boldsymbol{v}_i - \left(\mathbb{1}p^{\mathsf{T}} \otimes I_M\right) \left(\mathcal{A}^{\mathsf{T}} \mathcal{W}_{i-1} + \boldsymbol{v}_i\right) \\
&= \left(\mathcal{A}^{\mathsf{T}} - \mathbb{1}p^{\mathsf{T}} \otimes I_M\right) \mathcal{W}_{i-1} + \left(I_{MK} - \mathbb{1}p^{\mathsf{T}} \otimes I_M\right) \boldsymbol{v}_i
\end{aligned} \tag{7.69}$$

We take squares and expectations and find:

$$\begin{aligned}
&\mathbb{E}\|\mathcal{W}_i - \mathbb{1} \otimes \boldsymbol{w}_{c,i}\|^2 \\
&= \mathbb{E}\left\|\left(\mathcal{A}^{\mathsf{T}} - \mathbb{1}p^{\mathsf{T}} \otimes I_M\right) \mathcal{W}_{i-1} + \left(I_{MK} - \mathbb{1}p^{\mathsf{T}} \otimes I_M\right) \boldsymbol{v}_i\right\|^2 \\
&\leq \frac{1}{\sigma_2} \mathbb{E}\left\|\left(\mathcal{A}^{\mathsf{T}} - \mathbb{1}p^{\mathsf{T}} \otimes I_M\right) \mathcal{W}_{i-1}\right\|^2 + \frac{1}{1-\sigma_2} \mathbb{E}\left\|\left(I_{MK} - \mathbb{1}p^{\mathsf{T}} \otimes I_M\right) \boldsymbol{v}_i\right\|^2 \\
&= \frac{1}{\sigma_2} \mathbb{E}\left\|\left(\mathcal{A}^{\mathsf{T}} - \mathbb{1}p^{\mathsf{T}} \otimes I_M\right) \left(I_{MK} - \mathbb{1}p^{\mathsf{T}} \otimes I_M\right) \mathcal{W}_{i-1}\right\|^2 + \frac{1}{1-\sigma_2} \mathbb{E}\left\|\left(I_{MK} - \mathbb{1}p^{\mathsf{T}} \otimes I_M\right) \boldsymbol{v}_i\right\|^2 \\
&\leq \frac{1}{\sigma_2} \left\|\mathcal{A}^{\mathsf{T}} - \mathbb{1}p^{\mathsf{T}} \otimes I_M\right\|^2 \mathbb{E}\|\mathcal{W}_{i-1} - \mathbb{1} \otimes \boldsymbol{w}_{c,i-1}\|^2 + \frac{1}{1-\sigma_2} \mathbb{E}\left\|\left(I_{MK} - \mathbb{1}p^{\mathsf{T}} \otimes I_M\right) \boldsymbol{v}_i\right\|^2 \\
&= \sigma_2 \mathbb{E}\|\mathcal{W}_{i-1} - \mathbb{1} \otimes \boldsymbol{w}_{c,i-1}\|^2 + \frac{1}{1-\sigma_2} \sigma_{\overline{v}}^2
\end{aligned} \tag{7.70}$$

Upon iterating and taking the limit as $i \to \infty$, we have:

$$\lim_{i \to \infty} \mathbb{E}\|\mathcal{W}_i - \mathbb{1} \otimes \boldsymbol{w}_{c,i}\|^2 \leq \frac{1}{1-\sigma_2} \left(\sum_{j=0}^{\infty} \sigma_2^j\right) \sigma_{\overline{v}}^2 = \frac{\sigma_{\overline{v}}^2}{(1-\sigma_2)^2} \tag{7.71}$$

7.5 PROBLEMS

7.1 Establish relation (7.46).

7.2 We saw at several points throughout this chapter that the mixing rate of an matrix A, quantified through its second-largest singular value $\sigma_2(A)$ plays a key role in quantifying the rate at which averages can be computed. In this problem, we will verify Theorem 7.1 in code. To this end, generate a random collection of signals $\{g_k\}_{k=1}^K$ using a statistical model of your choice. For the graph, we generate Erdos-Renyi graphs with varying edge probability p_{edge}. Any pair of agents ℓ and k is linked with probability p_{edge}, independently of all other edges. Construct the adjacency matrix A following the Metropolis rule of Chapter 6. For different choices of p_{edge}, compute the associated mixing rate $\sigma_2(A)$. Subsequently implement the static consensus algorithm (7.16), plot the evolution of $\sum_{k=1}^K \|\overline{g} - w_{k,i}\|^2$ and discuss the relationship between edge probablities p_{edge}, $\sigma_2(A)$ and the rate of convergence.