Multi-Agent Optimization and Learning
Lecture 3: Graphs and their Role in Decentralized Processing

Stefan Vlaski[†] and Ali H. Sayed[⋆]

[†]Department of Electrical and Electronic Engineering, Imperial College London, UK
[⋆]Adaptive Systems Laboratory, École Polytechnique Fédérale de Lausanne, Switzerland

IEEE ICASSP 2024 Short Course

Imperial College London

EPFL

Objective of this lecture

- In Lecture 2 we presented a number of fusion-center based algorithms for multi-agent learning optimization and learning.
- We also developed performance bounds, and investigated the dependence of these bounds on different factors including:
  - The quality of local gradient approximations.
  - The level of heterogeneity in the system.
  - Other algorithm parameters such as participation rate and number of local updates.
- For the remainder of the course, we will develop fully decentralized architectures, where interactions occur over a graph.
- In this lecture, we will introduce graphs and their properties, and illustrate some key dynamics in a simplified setting, namely the problem of computing averages over a graph.
- In the next lecture, we will generalize to optimization and learning over graphs.

Graphs

An directed, unweighted graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ 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 $\mathcal{N}$ 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 lectures, vertices will be "agents", or "learners", terms which we use interchangeably in our presentation. We generally denote the number of agents $|\mathcal{N}|$ by K.

Example

We consider a collection of 4 nodes, numbered 1 through 4. The nodes are collected in the set:

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

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

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

The resulting graph $\mathcal{G} \triangleq (\mathcal{N}, \mathcal{E})$ is displayed below.

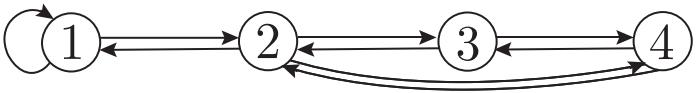


Figure: A simple graph consisting of four vertices.

Adjacency Matrix

Instead of describing a graph via its set of nodes $\mathcal{N}$ and vertices $\mathcal{E}$, we can equivalently describe it through its adjacency matrix $C \in \{0,1\}^{K \times K}$. Here, the element $c_{\ell k} = [C]_{\ell,k}$ is set to one when there is an edge from node ℓ to node k, and to zero otherwise. Formally:

$$c_{\ell k} = 1 \iff (\ell, k) \in \mathcal{E} \tag{3}$$

$$c_{\ell k} = 0 \iff (\ell, k) \notin \mathcal{E} \tag{4}$$

In this manner, we can find for the graph from the earlier example:

$$C = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \tag{5}$$

Weighted Adjacency Matrix

In many situations, it will be useful to associate a weight with edges in a graph to quantify the *strength* of the link between a pair of nodes ℓ and k. We can readily capture these weights by allowing $c_{\ell k}$ to take non-negative real values, resulting in an adjacency matrix $C \in \mathbb{R}^{K \times K}$. We allow only *non-zero* weights to be associated with an edge to avoid degenerate situations where nodes are connected by an edge, but the associated weight is zero. In this manner, it follows from $c_{\ell k} > 0$ that there is an edge from agent ℓ to agent k, while $c_{\ell k} = 0$ implies that there is no edge from ℓ to k.

$$C = \begin{pmatrix} 1 & 2 & 0 & 0 \\ 0.5 & 0 & 4 & 2 \\ 0 & 2 & 0 & 1 \\ 0 & 0.2 & 1 & 0 \end{pmatrix} \tag{6}$$

Neighborhoods

For a node k, we define the *in*-neighborhood as the set of nodes $\mathcal{N}_k^{\text{in}}$, for which an edge *from* ℓ *to* k exists. Formally:

$$\mathcal{N}_k^{\text{in}} = \{\ell : (\ell, k) \in \mathcal{E}\} = \{\ell : c_{\ell k} > 0\} \tag{7}$$

Alternatively, we can define the *out*-neighborhood $\mathcal{N}_k^{\text{out}}$ as the set of nodes, for which an edge *to* ℓ *from* k exists:

$$\mathcal{N}_k^{\text{out}} = \{\ell : (k, \ell) \in \mathcal{E}\} = \{\ell : c_{k\ell} > 0\} \tag{8}$$

For graphs where an edge $c_{\ell k}$ describes the influence of agent ℓ on agent k, we can interpret $\mathcal{N}_k^{\text{in}}$ as the set of agents that k is influenced by, while $\mathcal{N}_k^{\text{out}}$ denotes the set of nodes that are influenced by k.

An edge that connects an agent to itself is called a self-loop. This means that an agent k can be contained in its own neighborhood, provided that it has a self-loop.

Undirected Graphs

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

$$k \in \mathcal{N}_\ell^{\text{in}} \iff \ell \in \mathcal{N}_k^{\text{in}} \tag{9}$$

It can be readily verified that this is equivalent to requiring the in-neighborhoods and out-neighborhoods of each node k to coincide:

$$\mathcal{N}_k^{\text{in}} = \mathcal{N}_k^{\text{out}} \triangleq \mathcal{N}_k \tag{10}$$

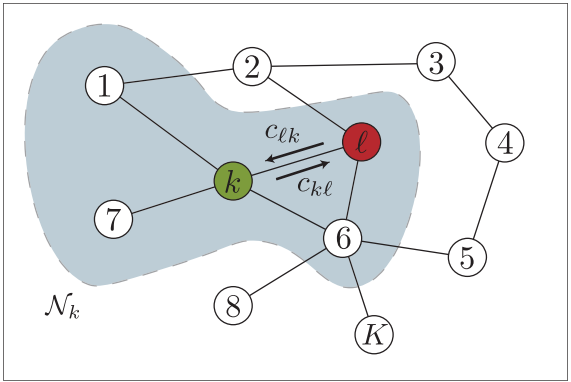
This means that for undirected graphs, there is no need to distinguish between its in-degrees and out-degrees, and we can just refer to the neighborhood $\mathcal{N}_k$.

Symmetric Graphs

We say that a graph is symmetric, if the weights linking pairs of agents are symmetric, i.e., when the adjacency matrix is symmetric.

$$C = C^{\mathsf{T}} \tag{11}$$

Adjacency Matrix



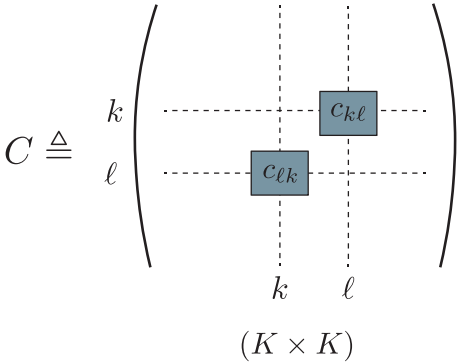


Figure: An undirected graph with the neighborhood $\mathcal{N}_k$ of node highlighted. On the right, we see the corresponding adjacency matrix.

Strongly-Connected Graphs

We say that a graph is strongly-connected if:

- It is connected.
- There exists at least one self-loop, i.e., $c_{kk} > 0$ for at least one k.

Strong-connectivity of a graph implies that its adjacency matrix is primitive, which means that there exists some finite integer $n_o > 0$ such that all entries of C^{n_o} are strictly positive:

$$[C^{n_o}]_{\ell,k} > 0, \quad \text{uniformly for all } (\ell, k) \tag{12}$$

Perron-Frobenius Theorem

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

(a) The matrix C has a *single* leading eigenvalue $\lambda_1 \in \mathbb{R}$, which is positive and real.

(b) The magnitudes of all other (potentially complex) eigenvalues of C are *strictly less* than λ_1, so that $\rho(A) = \lambda_1$.

(c) The leading eigenvalue λ_1 has algebraic multiplicity one, which implies that there is a single eigenvector p associated with λ_1. Furthermore, p can be normalized so that all of its entries are strictly positive and add up to one, i.e.,

$$Cp = \lambda_1 p, \quad \mathbb{1}^{\mathsf{T}} p = 1, \quad p_k > 0, \;\; k = 1, 2, \ldots, K \tag{13}$$

We refer to p as the *Perron vector* of C.

Jordan Decomposition of Primitive Matrices

Owing to the multiplicity of the leading eigenvalue, the Jordan decomposition $C = V_\epsilon J V_\epsilon^{-1}$ has a very particular structure:

$$V_\epsilon = \begin{bmatrix} p & V_R \end{bmatrix}, \quad J = \begin{bmatrix} \lambda_1 & 0 \\ 0 & J_\epsilon \end{bmatrix}, \quad V_\epsilon^{-1} = \begin{bmatrix} q^{\mathsf{T}} \\ V_L^{\mathsf{T}} \end{bmatrix} \tag{14}$$

where J_ϵ is a block Jordan matrix with the eigenvalues $\lambda_2(C)$ through $\lambda_K(C)$ on the diagonal and ϵ on the first lower sub-diagonal. In particular, $\rho(J_\epsilon) < \lambda_1$. One consequence of this decomposition is that:

$$\rho\left(A - pq^{\mathsf{T}}\right) < \lambda_1 \tag{15}$$

and

$$\lim_{i\to\infty} \frac{C^i}{\lambda_1^i} = pq^{\mathsf{T}} \tag{16}$$

Graph Laplacian

An alternative description of a graph is sometimes given through its graph Laplacian:

$$L \triangleq \operatorname{diag}\{C\mathbb{1}\} - C \tag{17}$$

When C is symmetric, several useful facts about a graph can be extracted from its Laplacian. Let $\theta_1 \geq \theta_2 \geq \ldots \geq \theta_K$ denote the ordered eigenvalues of L. Then:

(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 $L\mathbb{1} = 0$. This means that $\mathbb{1}$ is a right eigenvector of L corresponding to the eigenvalue at zero.

(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.

Example

If we consider again the graph from our earlier example, we find:

$$
\begin{aligned}
L &= \operatorname{diag}\{C\mathbb{1}\} - C \\
&= \operatorname{diag}\left\{ \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \right\} - \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \\
&= \operatorname{diag}\left\{ \begin{pmatrix} 2 \\ 3 \\ 2 \\ 2 \end{pmatrix} \right\} - \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \\
&= \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix} - \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & -1 & 0 & 0 \\ -1 & 3 & -1 & -1 \\ 0 & -1 & 2 & -1 \\ 0 & -1 & -1 & 2 \end{pmatrix}
\end{aligned} \tag{18}
$$

Interpretation as a Differential Operator

The Laplacian carries a useful interpretation because it allows us to compute efficiently the variation of a signal over a graph. Specifically, for a symmetric $C = C^{\mathsf{T}}$ and a signal $x \in \mathbb{R}^K$, it holds that:

$$\frac{1}{2}\sum_{k=1}^{K}\sum_{\ell=1}^{K} c_{\ell k}\|x_k - x_\ell\|^2 = x^{\mathsf{T}} L x \tag{19}$$

Combination/Weight Matrices

Adjacency matrices will appear frequently in future lectures as a means for describing weighting assigned by agents to information received from their neighbors. In this context, we will impose additional conditions on the entries of the adjacency matrix, and we will refer to matrices satisfying these conditions as *combination matrices* or *weight matrices*. Combination weights $\{a_{\ell k}\}$ will be nonnegative scalars that satisfy the following condition for each agent $k = 1, 2, \ldots, K$:

$$a_{\ell k} \geq 0, \quad \sum_{\ell=1}^{K} a_{\ell k} = 1, \quad \text{and} \quad a_{\ell k} = 0 \text{ if, and only if, } \ell \notin \mathcal{N}_k \tag{20}$$

Condition (20) implies that the combination matrix $A = [a_{\ell k}]$ is *left-stochastic*, which means that:

$$\mathbb{1}^{\mathsf{T}} A = \mathbb{1}^{\mathsf{T}} \iff A^{\mathsf{T}} \mathbb{1} = \mathbb{1}, \quad \text{(left-stochastic)} \tag{21}$$

Spectral Properties of Combination Matrices

Since the spectral radius of a matrix is bounded by any of its norms, it follows that:

$$\rho(A) \leq \|A\|_1 = 1 \tag{22}$$

From (21) we observe that A has an eigenvalue at one and hence $\rho(A) = 1$. We also recognize that A is an adjacency matrix. Hence, if the underlying graph is strongly-connected, it follows that A is primitive, and we can conclude from Perron-Frobenius, that the eigenvalue at one has multiplicity one with corresponding Perron vector p:

$$Ap = p \tag{23}$$

If A is additionally symmetric, it follows that:

$$A^{\mathsf{T}}\mathbb{1} = \mathbb{1}, \;\; A = A^{\mathsf{T}}, \qquad \text{(doubly-stochastic)} \tag{24}$$

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 It is useful to note that when A is doubly-stochastic, its Perron vector p defined in (22) is given by

$$p = \frac{1}{K}\mathbb{1} \tag{25}$$

Constructing Combination Matrices

We will frequently be provided with an C, whose entries are positive but do not necessarily add up to one, and seek to transform it into a valid left- or doubly-stochastic combination matrix A. There are some popular constructions, which we list here. These constructions start with an undirected, unweighted graph with associated adjacency matrix C and Laplacian L. We also denote the degrees of the various agents by n_k and refer to the maximum degree across the graph by $n_{\max}$.

Constructing Combination Matrices

- Averaging rule (left-stochastic):

$$a_{\ell k} = \begin{cases} 1/n_k, & \text{if } k \neq \ell \text{ are neighbors or } k = \ell \\ 0, & \text{otherwise} \end{cases} \tag{26}$$

- Laplacian rule (doubly-stochastic):

$$A = I_K - \beta\, L,\ \beta > 0 \tag{27}$$

- Metropolis rule (doubly-stochastic):

$$a_{\ell k} = \begin{cases} 1/\max\{n_k, n_\ell\}, & \text{if } k \neq \ell \text{ are neighbors} \\ 1 - \displaystyle\sum_{\ell \in \mathcal{N}_k \backslash \{k\}} a_{\ell k}, & \text{when } k = \ell \\ 0, & \text{otherwise} \end{cases} \tag{28}$$

Averaging over Graphs

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 $\mathcal{G} = (\mathcal{N}, \mathcal{E})$. Our objective is to, in a decentralized manner, compute:

$$\{g_1, \ldots, g_K\} \Longrightarrow \overline{g} \triangleq \sum_{k=1}^{K} p_k g_k \tag{29}$$

- 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.
- If the collection of signals $\{g_k\}_{k=1}^K$ are jointly available at some central location, and in the absence of computational constraints, implementing (29) is straightforward by simply averaging the available realizations.

Recap and Outlook

- We have introduced various concepts relating to graphs, as well as their properties.
- One key theme was that properties of the graph in many cases imply spectral properties on the adjacency or Laplacian matrix.
  - Hence, we can study graphs through linear algebra.
- We will now see how we can apply these concepts by performing a simple task over networks, namely computing averages.

Averaging over Graphs

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:

$$\{g_1, \ldots, g_K\} \Longrightarrow \overline{g} \triangleq \sum_{k=1}^{K} p_k g_k \tag{30}$$

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.

The Average Consensus Algorithm

We claim that we can pursue the average value $\overline{g} = \sum_{k=1}^{K} p_k g_k$ by initializing $w_{k,0} = g_k$ and then iterating:

$$w_{k,i} = \sum_{\ell \in \mathcal{N}_k} a_{\ell k} w_{\ell,i-1} \tag{31}$$

where the weights $a_{\ell k}$ are elements of a left-stochastic combination matrix $A \in \mathbb{R}^{K \times K}$ with:

$$a_{\ell k} \begin{cases} > 0, \text{ if } \ell \in \mathcal{N}_k, \\ = 0, \text{ otherwise.} \end{cases} \qquad \text{and} \qquad \sum_{\ell=1}^{K} a_{\ell k} = 1 \tag{32}$$

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

$$Ap = p, \qquad A^{\mathsf{T}} \mathbb{1} = \mathbb{1}, \qquad \lim_{i \to \infty} A^i = p \mathbb{1}^{\mathsf{T}} \tag{33}$$

where p denotes the Perron eigenvector of A. Assume for now that the Perron entries p_k coincide with the desired averaging weights (more on design later).

Network Quantities

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:

$$\mathcal{w}_i \triangleq \begin{pmatrix} w_{1,i} \\ w_{2,i} \\ \vdots \\ w_{K,i} \end{pmatrix} \tag{34}$$

and define

$$\mathcal{A} \triangleq A \otimes I_M \tag{35}$$

We can then write the consensus averaging recursion compactly as:

$$\mathcal{w}_i = \mathcal{A}^{\mathsf{T}} \mathcal{w}_{i-1} \tag{36}$$

Limiting Dynamics

Iterating the network recursion:

$$\lim_{i\to\infty} \mathcal{W}_i = \lim_{i\to\infty} \left(\mathcal{A}^{\mathsf{T}}\right)^i \mathcal{W}_0 = \left(\lim_{i\to\infty} \left(A^{\mathsf{T}}\right)^i \otimes I_M\right) \mathcal{W}_0 \tag{37}$$

But we know from Perron-Frobenius that:

$$\lim_{i\to\infty} A^i = p\mathbb{1}^{\mathsf{T}} \Longrightarrow \lim_{i\to\infty} \left(A^{\mathsf{T}}\right)^i = \mathbb{1}p^{\mathsf{T}} \tag{38}$$

and hence:

$$\lim_{i\to\infty} \mathcal{W}_i = \left(\mathbb{1}p^{\mathsf{T}} \otimes I_M\right) \mathcal{W}_0 = \begin{pmatrix} p^{\mathsf{T}} \otimes I_M \\ \vdots \\ p^{\mathsf{T}} \otimes I_M \end{pmatrix} \begin{pmatrix} g_1 \\ \vdots \\ g_K \end{pmatrix} = \begin{pmatrix} \sum_{k=1}^K p_k g_k \\ \vdots \\ \sum_{k=1}^K p_k g_k \end{pmatrix} \tag{39}$$

We conclude that $\lim_{i\to\infty} w_{k,i} = \overline{g}$ for all k.

Convergence of the consensus averaging algorithm

Linear convergence of the consensus averaging algorithm

Each agent $w_{k,i}$ following the consensus protocol (31) converges linearly to the weighted mean $\overline{g} \triangleq \sum_{k=1}^{K} p_k g_k$:

$$\sum_{k=1}^{K} \|\overline{g} - w_{k,i}\|^2 \leq \left\|p\mathbb{1}^{\mathsf{T}} - A\right\|^{2i} \times \sum_{k=1}^{K} \|\overline{g} - w_{k,0}\|^2, \tag{40}$$

where the rate of convergence is dictated by $\left\|p\mathbb{1}^{\mathsf{T}} - A\right\|$, and we are free to choose the norm.

Sketch of Proof: Step 1 – Centroid is Stationary

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:

$$\begin{aligned}
\overline{w}_i &= \sum_{k=1}^{K} p_k w_{k,i} = \left(p^{\mathsf{T}} \otimes I_M\right) \mathcal{W}_i = \left(p^{\mathsf{T}} \otimes I_M\right) \mathcal{A}^{\mathsf{T}} \mathcal{W}_{i-1} = \left((Ap)^{\mathsf{T}} \otimes I_M\right) \mathcal{W}_{i-1} \\
&= \left(p^{\mathsf{T}} \otimes I_M\right) \mathcal{W}_{i-1} \\
&= \sum_{k=1}^{K} p_k w_{k,i-1} \\
&= \overline{w}_{i-1}
\end{aligned} \tag{41}$$

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:

$$\overline{w}_i = \overline{w}_{i-1} = \ldots = \overline{w}_0 = \sum_{k=1}^{K} p_k g_k \tag{42}$$

Sketch of Proof: Step 2 – Deviations around the Centroid Contract

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 recursion from the augmented mean $\mathbb{1}_K \otimes \overline{w}_i$, we find:

$$
\begin{aligned}
\mathbb{1}_K \otimes \overline{w}_i - \mathcal{W}_i &= \mathbb{1}_K \otimes \overline{w}_i - \mathcal{A}^{\mathsf{T}} \mathcal{W}_{i-1} \\
&= \mathbb{1}_K \otimes \overline{w}_{i-1} - \mathcal{A}^{\mathsf{T}} \mathcal{W}_{i-1} \\
&= \mathbb{1}_K \otimes \Big(\Big(p^{\mathsf{T}} \otimes I_M \Big) \mathcal{W}_{i-1} \Big) - \mathcal{A}^{\mathsf{T}} \mathcal{W}_{i-1} \\
&= \Big(\mathbb{1} p^{\mathsf{T}} \otimes I_M \Big) \mathcal{W}_{i-1} - \mathcal{A}^{\mathsf{T}} \mathcal{W}_{i-1} \\
&= \Big(\mathbb{1} p^{\mathsf{T}} \otimes I_M - \mathcal{A}^{\mathsf{T}} \Big) \mathcal{W}_{i-1} \\
&= \Big(\mathbb{1} p^{\mathsf{T}} \otimes I_M - \mathcal{A}^{\mathsf{T}} \Big) (\mathcal{W}_{i-1} - \mathbb{1} \otimes \overline{w}_{i-1})
\end{aligned} \tag{43}
$$

The result follows after taking norms and and applying the sub-multiplicative property of norms.

Centralities and Mixing Rate

For a graph $\mathcal{G}$ with combination matrix A, we refer to the elements of the Perron eigenvector:

$$Ap = p \tag{44}$$

as the *agent centrality*, and the quantity:

$$\lambda_2 \triangleq \left\| A - p\mathbb{1}^{\mathsf{T}} \right\| \tag{45}$$

as the *mixing rate*. Note that for a symmetric combination matrix $A = A^{\mathsf{T}}$, by choosing $\|\cdot\|$ to be the spectral norm, it follows that $p = \frac{1}{K}\mathbb{1}$ and:

$$\left\| A - \frac{1}{K}\mathbb{1}\mathbb{1}^{\mathsf{T}} \right\| = \max\left\{ |\lambda_2(A)|, |\lambda_K(A)| \right\} \tag{46}$$

Designing Perron Vectors and Mixing Rates

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 $p = Ap$:

$$a_{\ell k} = \begin{cases} 0, & \text{if } \ell \notin \mathcal{N}_k, \\ p_\ell, & \text{if } \ell \in \mathcal{N}_k \backslash \ell, \\ 1 - \sum_{m \in \mathcal{N}_k \backslash \ell} a_{mk}, & \text{when } \ell = k. \end{cases} \tag{47}$$

Combination matrices can also be designed to optimize $\rho\left(A - \frac{1}{K}\mathbb{1}\mathbb{1}^{\mathsf{T}}\right)$ [Boyd, Diaconis, and Xiao 2004].

Dynamic Consensus

We consider now 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:

$$\overline{\boldsymbol{g}}_i \triangleq \sum_{k=1}^{K} p_k \boldsymbol{g}_{k,i} \tag{48}$$

where $\{p_k\}_{k=1}^K$ denote scalar convex combination weights, such that $\sum_{k=1}^K p_k = 1$.

Dynamic Consensus

To this end, it will be useful to formulate an optimization problem:

$$\overline{\boldsymbol{g}}_i = \arg\min_{\boldsymbol{\mathcal{W}}} \sum_{k=1}^{K} p_k \| \boldsymbol{w}_k - \boldsymbol{g}_{k,i} \|^2, \text{ subject to } \boldsymbol{w}_k = \boldsymbol{w}_\ell \ \forall \ k, \ell. \tag{49}$$

To verify that (49) indeed defines the weighted average $\overline{\boldsymbol{g}}_i$, we note that under the constraint $\boldsymbol{w}_k = \boldsymbol{w}_\ell \ \forall \ k, \ell$, (49) is equivalent to:

$$\overline{\boldsymbol{g}}_i = \arg\min_{\boldsymbol{w}} \sum_{k=1}^{K} p_k \| \boldsymbol{w} - \boldsymbol{g}_{k,i} \|^2 \tag{50}$$

By differentiating, we find:

$$\sum_{k=1}^{K} p_k (\boldsymbol{w} - \boldsymbol{g}_{k,i}) = 0 \Longleftrightarrow \boldsymbol{w} = \sum_{k=1}^{K} p_k \boldsymbol{g}_{k,i} \tag{51}$$

Dynamic Consensus

We will now examine algorithms for dynamic consensus by reformulating (49) 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 (49) is equivalent to:

$$\overline{\boldsymbol{g}}_i = \arg\min_{\boldsymbol{\mathcal{W}}} \sum_{k=1}^{K} p_k \|\boldsymbol{w}_k - \boldsymbol{g}_{k,i}\|^2, \ \text{subject to} \ \boldsymbol{w}_k = \boldsymbol{w}_\ell \ \forall \ \ell \in \mathcal{N}_k. \tag{52}$$

Dynamic Consensus

Furthermore, we can write equivalently:

$$\overline{\boldsymbol{\varrho}}_i = \arg\min_{\boldsymbol{\mathcal{W}}} \sum_{k=1}^{K} p_k \|\boldsymbol{w}_k - \boldsymbol{w}_{k,i}\|^2, \text{ subject to } \sum_{k=1}^{K}\sum_{\ell=1}^{K} c_{\ell k} \| \boldsymbol{w}_k - \boldsymbol{w}_\ell \|^2 = 0. \tag{53}$$

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

$$c_{\ell k} = \begin{cases} > 0 & \text{if } \ell \in \mathcal{N}_k, \\ = 0 & \text{otherwise.} \end{cases} \tag{54}$$

To verify that (52) and (53) are equivalent, we observe that:

$$\sum_{k=1}^{K}\sum_{\ell=1}^{K} c_{\ell k} \| \boldsymbol{w}_k - \boldsymbol{w}_\ell \|^2 = 0 \Longleftrightarrow \| \boldsymbol{w}_k - \boldsymbol{w}_\ell \|^2 = 0 \ \forall \ell \in \mathcal{N}_k \Longleftrightarrow \boldsymbol{w}_k = \boldsymbol{w}_\ell, \forall \ell \in \mathcal{N}_k. \tag{55}$$

Penalty-Based Dynamic Consensus

Rather than solve (53) 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:

$$\overline{\boldsymbol{g}}_i^{\eta} = \arg\min_{\boldsymbol{\mathcal{W}}} \sum_{k=1}^{K} p_k \|\boldsymbol{w}_k - \boldsymbol{g}_{k,i}\|^2 + \frac{\eta}{4}\sum_{\ell=1}^{K} c_{\ell k}\| \boldsymbol{w}_k - \boldsymbol{w}_\ell \|^2 \tag{56}$$

where $\eta > 0$ denotes a new penalty parameter. It is important to note that, while (49) through (53) were *equivalent* reformulations of the consensus problem, relation (56) 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.

Penalty-Based Dynamic Consensus

In the sequel, we will quantify this difference, and present an algorithm for pursuing $\overline{\boldsymbol{\mathcal{G}}}_i^{\eta}$. To this end, it will be useful to write the penalized cost (56) in terms of network level quantities. We introduce:

$$P = \begin{pmatrix} p_1 & 0 & \cdots & 0 \\ 0 & p_2 & \cdots & 0 \\ 0 & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & p_K \end{pmatrix} = \mathrm{diag}\{p\} \tag{57}$$

$$\mathcal{P} = P \otimes I_M \tag{58}$$

Then:

$$\sum_{k=1}^{K} p_k \|\boldsymbol{w}_k - \boldsymbol{g}_{k,i}\|^2 = \|\boldsymbol{\mathcal{W}} - \boldsymbol{\mathcal{G}}_i\|_{\mathcal{P}}^2 \tag{59}$$

Penalty-Based Dynamic Consensus

For the second term in (56), we can verify, that:

$$\frac{\eta}{2}\sum_{k=1}^{K}\sum_{\ell\in\mathcal{N}_k} c_{\ell k}\|\,\boldsymbol{w}_k-\boldsymbol{w}_\ell\,\|^2=\eta\,\boldsymbol{\mathcal{W}}^{\mathsf{T}}\,\mathcal{L}\,\boldsymbol{\mathcal{W}} \tag{60}$$

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

$$L=\operatorname{diag}\{C\mathbb{1}\}-C \tag{61}$$

Hence, problem (56) is equivalent to:

$$\overline{\boldsymbol{\mathcal{G}}}_i^{\eta}=\arg\min_{\boldsymbol{\mathcal{W}}}\frac{1}{2}\|\,\boldsymbol{\mathcal{W}}-\boldsymbol{\mathcal{G}}_i\|_{\mathcal{P}}^2+\frac{\eta}{2}\,\boldsymbol{\mathcal{W}}^{\mathsf{T}}\,\mathcal{L}\,\boldsymbol{\mathcal{W}} \tag{62}$$

Penalty-Based Dynamic Consensus

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 $\mathcal{P}^{-1}$:

$$\begin{aligned}\boldsymbol{\mathcal{W}}_i &= \boldsymbol{\mathcal{W}}_{i-1} - \mu\mathcal{P}^{-1}\left(\mathcal{P}\left(\boldsymbol{\mathcal{W}}_{i-1} - \boldsymbol{\mathcal{G}}_i\right) + \mu\eta\mathcal{L}\,\boldsymbol{\mathcal{W}}_{i-1}\right) \\ &= \boldsymbol{\mathcal{W}}_{i-1} - \mu\left(\boldsymbol{\mathcal{W}}_{i-1} - \boldsymbol{\mathcal{G}}_i\right) - \mu\eta\mathcal{P}^{-1}\mathcal{L}\,\boldsymbol{\mathcal{W}}_{i-1} \\ &= \left((1-\mu)I - \mu\eta\mathcal{P}^{-1}\mathcal{L}\right)\boldsymbol{\mathcal{W}}_{i-1} + \mu\boldsymbol{\mathcal{G}}_i \\ &= \left(I - \mu\eta\mathcal{P}^{-1}\mathcal{L}\right)\boldsymbol{\mathcal{W}}_{i-1} + \mu\left(\boldsymbol{\mathcal{G}}_i - \boldsymbol{\mathcal{W}}_{i-1}\right)\end{aligned} \tag{63}$$

We set $\mu = \eta^{-1}$ and define:

$$\mathcal{A}^{\mathsf{T}} \triangleq I - \mathcal{P}^{-1}\mathcal{L} \tag{64}$$

Penalty-Based Dynamic Consensus

We can then equivalently write:

$$\boldsymbol{\mathcal{W}}_i = \mathcal{A}^{\mathsf{T}}\,\boldsymbol{\mathcal{W}}_{i-1} + \mu\,(\boldsymbol{\mathcal{G}}_i - \boldsymbol{\mathcal{W}}_{i-1}) \tag{65}$$

It is instructive to examine the spectral structure of $\mathcal{A}^{\mathsf{T}}$ defined as in (64). Since $\mathcal{L}\mathbb{1} = 0$, we have:

$$A^{\mathsf{T}}\mathbb{1} = \left(I - \mu\eta P^{-1}L\right)\mathbb{1} = \mathbb{1} - \mu\eta P^{-1}L\mathbb{1} = \mathbb{1} \tag{66}$$

$$Ap = \left(I - \mu\eta LP^{-1}\right)p = p - \mu\eta LP^{-1}p = p - \mu\eta L\mathbb{1} = p \tag{67}$$

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:

$$\boldsymbol{w}_{k,i} = \sum_{\ell\in\mathcal{N}_k} a_{\ell k}\,\boldsymbol{w}_{\ell,i-1} + \mu\left(\boldsymbol{g}_{k,i} - \boldsymbol{w}_{k,i-1}\right) \tag{68}$$

Bias of Penalty-Based Algorithms

The dynamic consensus algorithm we derived relies on a penalty-reformulation of the networked averaging problem, which is merely an approximation unless $\eta \to \infty$ which implies $\mu \to 0$. We can verify that even when applying this algorithm to a constant set of signal, this will result in a bias compared to the classical consensus averaging algorithm. Thus, consider the case when $\boldsymbol{g}_{k,i} \triangleq g_k$ is deterministic and constant. We then have:

$$\mathcal{W}_i = \left(\mathcal{A}^{\mathsf{T}} - \mu I\right) \mathcal{W}_{i-1} + \mu \mathcal{G} \tag{69}$$

Iterating we find:

$$\mathcal{W}_i = \left(\mathcal{A}^{\mathsf{T}} - \mu I\right)^i \mathcal{W}_0 + \mu \left(\sum_{j=0}^{i-1} \left(\mathcal{A}^{\mathsf{T}} - \mu I\right)^j\right) \mathcal{G} \tag{70}$$

Bias of Penalty-Based Algorithms

In the limit, we then have:

$$\lim_{i\to\infty} \mathcal{W}_i = \mu \left(\sum_{j=0}^{i-1} \left(\mathcal{A}^{\mathsf{T}} - \mu I \right)^j \right) \mathcal{G} = \mu \left(I - \left(\mathcal{A}^{\mathsf{T}} - \mu I \right) \right)^{-1} \mathcal{G} = \mu \left((1+\mu) I - \mathcal{A}^{\mathsf{T}} \right)^{-1} \mathcal{G} \tag{71}$$

By exploiting the Jordan decomposition of the primitive matrix A, we can expand this expression and conclude:

$$\sum_{k=1}^{K} \left\| \lim_{i\to\infty} w_{k,i} - g_k \right\|^2 = O(\mu^2) \tag{72}$$

Which means that we pay for the ability to track time-varying signals with a bias that is $O(\mu^2)$ when the signals are constant.

Exact Dynamic Consensus Algorithm

The previous discussion 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, and this algorithm can be derived using control-theoretic tools [Kia et al 2019] or primal-dual techniques we will develop in the next lecture. For the time being, we simply list the exact dynamic consensus algorithm, which takes the form:

$$\boldsymbol{w}_{k,i} = \sum_{\ell \in \mathcal{N}_k} a_{\ell k}\, \boldsymbol{w}_{\ell,i-1} + \boldsymbol{g}_{k,i} - \boldsymbol{g}_{k,i-1} \tag{73}$$

or in network level quantities:

$$\boldsymbol{\mathcal{W}}_i = \mathcal{A}^{\mathsf{T}}\, \boldsymbol{\mathcal{W}}_{i-1} + \boldsymbol{\mathcal{G}}_i - \boldsymbol{\mathcal{G}}_{i-1} \tag{74}$$

Constant Signals

We begin by establishing the ability of the dynamic consensus (74) to compute exactly the mean of a constant signal over the graph. To this end, let $\boldsymbol{\mathcal{G}}_i \triangleq \mathcal{G}$, and we find:

$$\boldsymbol{\mathcal{W}}_i = \mathcal{A}^{\mathsf{T}} \boldsymbol{\mathcal{W}}_{i-1} + \mathcal{G} - \mathcal{G} = \mathcal{A}^{\mathsf{T}} \boldsymbol{\mathcal{W}}_{i-1} \tag{75}$$

which corresponds precisely to the static consensus algorithm (36). Under the initialization $\boldsymbol{\mathcal{W}}_0 \triangleq \mathcal{G}$, we conclude that the evolution of the dynamic consensus algorithm (74) matches that of the static consensus algorithm (36), provided that the signal of interest $\boldsymbol{\mathcal{G}}_{i-1}$ is constant.

Tracking Random Walks

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:

$$\boldsymbol{\mathcal{G}}_i = \boldsymbol{\mathcal{G}}_{i-1} + \boldsymbol{v}_i \tag{76}$$

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

$$\boldsymbol{\mathcal{W}}_i = \mathcal{A}^{\mathsf{T}}\,\boldsymbol{\mathcal{W}}_{i-1} + \boldsymbol{\mathcal{G}}_i - \boldsymbol{\mathcal{G}}_{i-1} = \mathcal{A}^{\mathsf{T}}\,\boldsymbol{\mathcal{W}}_{i-1} + \boldsymbol{v}_i \tag{77}$$

Tracking Analysis: Centroid

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:

$$\boldsymbol{w}_{c,i} = \boldsymbol{w}_{c,i-1} + \sum_{k=1}^{K} p_k \boldsymbol{v}_{k,i} \tag{78}$$

Tracking Analysis: Centroid

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

$$\begin{aligned}
\boldsymbol{g}_{c,i} &\triangleq \sum_{k=1}^{K} p_k \boldsymbol{g}_{k,i} = \left(p^{\mathsf{T}} \otimes I_M\right) \boldsymbol{\mathcal{G}}_i \\
&\overset{(76)}{=} \left(p^{\mathsf{T}} \otimes I_M\right) \left(\boldsymbol{\mathcal{G}}_{i-1} + \boldsymbol{\mathcal{V}}_i\right) \\
&= \sum_{k=1}^{K} p_k \boldsymbol{g}_{k,i-1} + \sum_{k=1}^{K} p_k \boldsymbol{v}_{k,i} \\
&= \boldsymbol{g}_{c,i-1} + \sum_{k=1}^{K} p_k \boldsymbol{v}_{k,i}
\end{aligned} \tag{79}$$

Comparing the expression (78) for $\boldsymbol{w}_{c,i}$ with (79) for $\boldsymbol{g}_{c,i}$, we find that the recursions match exactly.

Tracking Analysis: Centroid

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:

$$\boldsymbol{w}_{c,i} = \boldsymbol{g}_{c,i} \iff \sum_{k=1}^{K} p_k \boldsymbol{w}_{k,i} = \sum_{k=1}^{K} p_k \boldsymbol{g}_{k,i} \tag{80}$$

This fact is analogous to relation (41) for the static consensus algorithm, with the important advantage that the correction term $\boldsymbol{g}_{i-1} - \boldsymbol{g}_{i-2}$ in (74) 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 probability one, independently of the power or variance of the random variable $\boldsymbol{v}_i$ driving the random walk.

Tracking Analysis: Deviation from the Centroid

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 (76) and the dynamic consensus algorithm (74) to establish this stronger result. We have:

$$\begin{aligned}
&\boldsymbol{\mathcal{W}}_i - \mathbb{1} \otimes \boldsymbol{w}_{c,i} \\
&= \boldsymbol{\mathcal{W}}_i - \left(\mathbb{1} p^{\mathsf{T}} \otimes I_M\right) \boldsymbol{\mathcal{W}}_i \\
&= \mathcal{A}^{\mathsf{T}} \boldsymbol{\mathcal{W}}_{i-1} + \boldsymbol{\mathcal{V}}_i - \left(\mathbb{1} p^{\mathsf{T}} \otimes I_M\right) \left(\mathcal{A}^{\mathsf{T}} \boldsymbol{\mathcal{W}}_{i-1} + \boldsymbol{\mathcal{V}}_i\right) \\
&= \left(\mathcal{A}^{\mathsf{T}} - \mathbb{1} p^{\mathsf{T}} \otimes I_M\right) \boldsymbol{\mathcal{W}}_{i-1} + \left(I_{MK} - \mathbb{1} p^{\mathsf{T}} \otimes I_M\right) \boldsymbol{\mathcal{V}}_i
\end{aligned} \tag{81}$$

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

$$\lim_{i \to \infty} \mathbb{E} \|\boldsymbol{\mathcal{W}}_i - \mathbb{1} \otimes \boldsymbol{w}_{c,i}\|^2 \leq \frac{1}{1 - \lambda_2} \left(\sum_{j=0}^{\infty} \lambda_2^j\right) \sigma_{\mathcal{V}}^2 = \frac{\sigma_{\mathcal{V}}^2}{(1 - \lambda_2)^2} \tag{82}$$

Simulation Study

A key take-away of this lecture is that the rate at which average consensus is achieved over a graph is determined by its mixing rate. We illustrate this numerically here. To this end, we generate first an unweighted Erdos-Renyi graph, where $c_{\ell k} = 1$ with probability p_{edge}. To ensure the graph is undirected, we set:

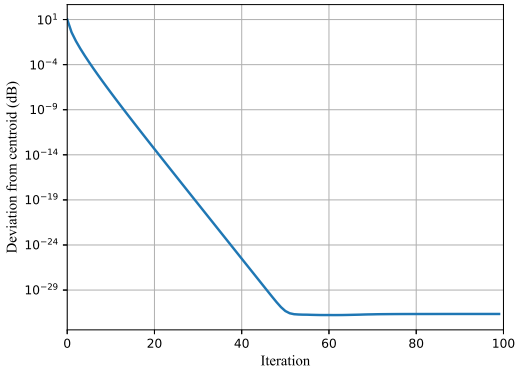
$$C \Longleftarrow \frac{1}{2}\left(C + C^{\mathsf{T}}\right) \tag{83}$$

Given this symmetric adjacency matrix, we can then construct a symmetric combination matrix $A = A^{\mathsf{T}}$ using the Metropolis-Rule. The mixing rate in that case is given by:

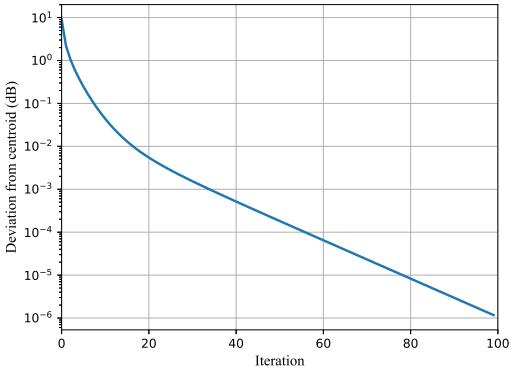
$$\lambda_2 = \rho\left(A - \frac{1}{K}\mathbb{1}\mathbb{1}^{\mathsf{T}}\right) \tag{84}$$

We can control λ_2 indirectly by adjusting the edge probability p_{edge}.

Convergence of the Consensus Averaging Algorithm for Differing Network Connectivities



$p_{\text{edge}} = 0.3,\ \lambda_2 = 0.498$



$p_{\text{edge}} = 0.05,\ \lambda_2 = 0.950$

Conclusion

- We introduced graphs and their (spectral) properties. To key quantities are:
  - The mixing rate: Quantifies how well connected the graph is, and how quickly information spreads.
  - The agent centrality: Quantifies the influence an agent has over the limiting behavior of the algorithm.
- We illustrated these in the context of averaging over graphs, a simple task that nevertheless captures some of the fundamental trade-offs.
- We also developed dynamic consensus algorithms, which allow for the tracking of averages of time-varying signals.

References and Further Reading

- General references and surveys:
  - A. H. Sayed, *Inference and Learning from Data*, Cambridge University Press, 2022.
  - A. H. Sayed, "Adaptation, learning, and optimization over networks," *Foundations and Trends in Machine Learning*, vol. 7, no. 4-5, pp. 311–801, July 2014.
- Averaging over graphs:
  - S. Boyd, P. Diaconis, and L. Xiao, "Fastest Mixing Markov Chain on a Graph", *SIAM Review*, vol. 46, no. 4, 2004.
  - S. S. Kia, B. Van Scoy, J. Cortes, R. A. Freeman, K. M. Lynch and S. Martinez, "Tutorial on Dynamic Average Consensus: The Problem, Its Applications, and the Algorithms," in *IEEE Control Systems Magazine*, vol. 39, no. 3, pp. 40-72, June 2019.