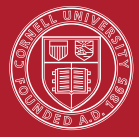




# Lecture 5 - Algebra of Networks: Network Flows and The Laplacian Matrix

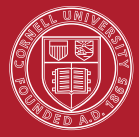
## Graph-Based Data Science For Networked Systems
## ECE 5260 – ORIE 5735

Anna Scaglione

Cornell Tech

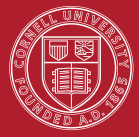
February 4, 2026

# Content



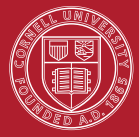
Cornell University

# Outline



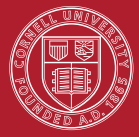
Cornell University

# Outline



Cornell University

# Incidence matrix definition





- Let $e = 1, \ldots, |\mathcal{E}|$ be the index of an edge $(u, v) \in \mathcal{E}$.
- The incidence matrix $\boldsymbol{B}$ of a weighted directed graph is a $|\mathcal{V}| \times |\mathcal{E}|$ matrix:

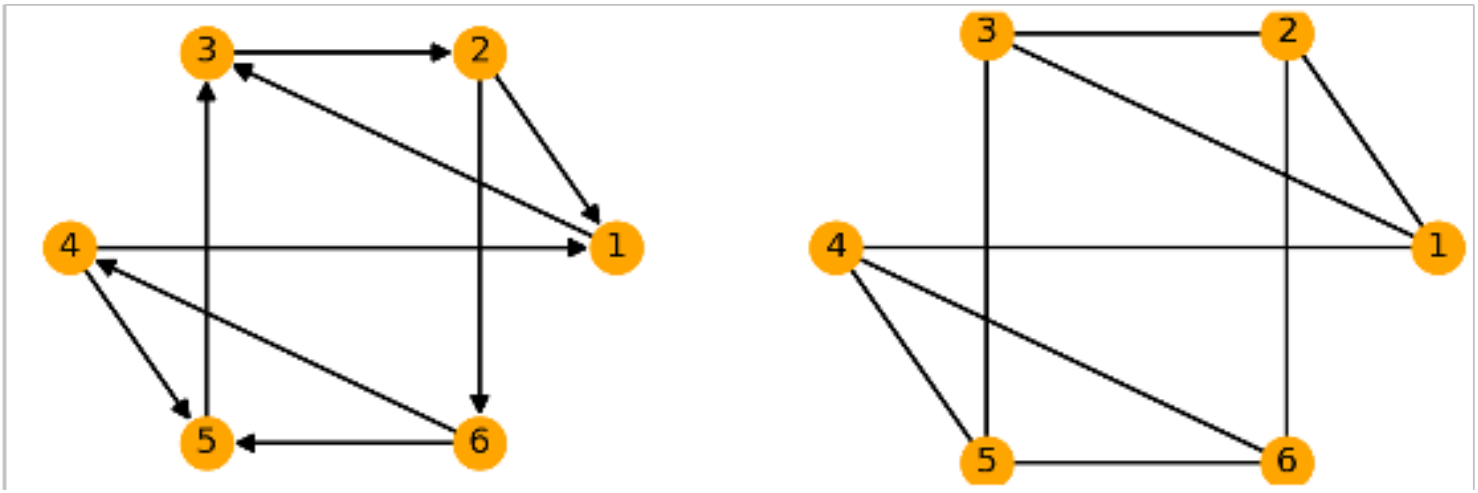
$$B_{ue} = \begin{cases} 1 & \text{if } e = (u, v) \in \mathcal{E}, \text{ i.e. enters node } u \\ -1 & \text{if } e = (v, u) \in \mathcal{E}, \text{ i.e. leaves node } u \\ 0 & \text{else.} \end{cases}$$

- For an weighted graph use the weight $w_e$ instead of $1$.

# Incidence matrix - Visualization

Cornell University

$$\boldsymbol{B} = \overbrace{\begin{bmatrix} 1. & 1. & 0. & 0. & 1. & 0. & 0. & 0. & 0. \\ 0. & 1. & 1. & 1. & 0. & 0. & 0. & 0. & 0. \\ 1. & 0. & 0. & 1. & 0. & 0. & 1. & 0. & 0. \\ 0. & 0. & 0. & 0. & 1. & 1. & 0. & 1. & 0. \\ 0. & 0. & 0. & 0. & 0. & 1. & 1. & 0. & 1. \\ 0. & 0. & 1. & 0. & 0. & 0. & 0. & 1. & 1. \end{bmatrix}}^{\text{Unoriented}} \quad \boldsymbol{B} = \overbrace{\begin{bmatrix} -1. & -1. & -1. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 1. & 0. & 1. & 1. & 0. & 0. & 0. & 0. \\ 1. & 0. & 0. & 0. & 1. & -1. & 0. & 0. & 0. \\ 0. & 0. & 1. & 0. & 0. & 0. & -1. & -1. & 0. \\ 0. & 0. & 0. & 0. & 0. & 1. & 1. & 0. & -1. \\ 0. & 0. & 0. & 1. & 0. & 0. & 0. & 1. & 1. \end{bmatrix}}^{\text{Oriented}}$$



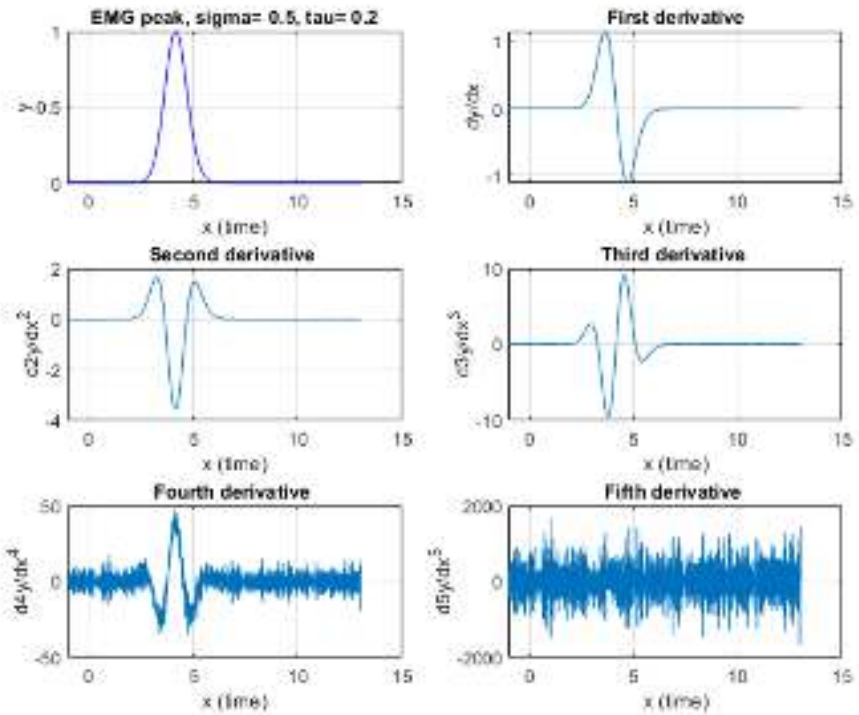

The oriented version is typically more useful....

# Derivative



- One very useful notion in analyzing time-series is to have an estimate of the *variation* of the signal.
- In continuous time we have the derivative:

$$\dot{x}(t) = \lim_{dt \to 0} \frac{x(t) - x(t - dt)}{dt}$$



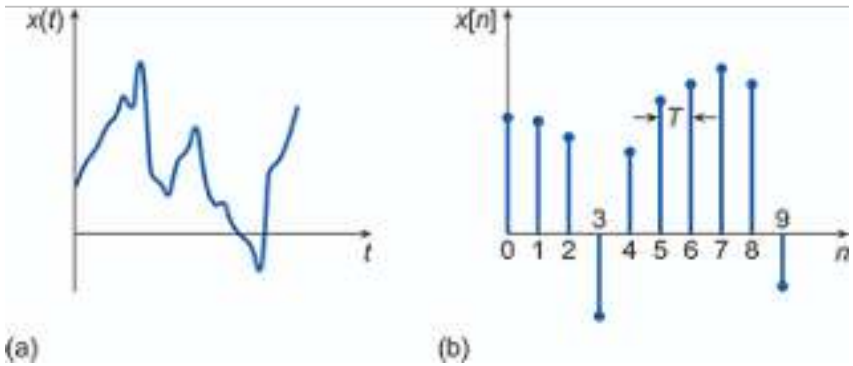
# Discrete derivative



- In discrete time, we use the finite difference signal:

$$dx(t) = x(t) - x(t-1), \quad \text{or} \quad dx(t) = (x(t) - x(t-1))w$$

if the spacing between samples is $T = 1/w$.




- You can view $\boldsymbol{x} = (x(0), \ldots, x(N-1))$ as the signal on a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{0, \ldots, N-1\}$ and $\mathcal{E} = \{(0,1), (1,2), \ldots, (N-2, N-1)$.

# Interpretation

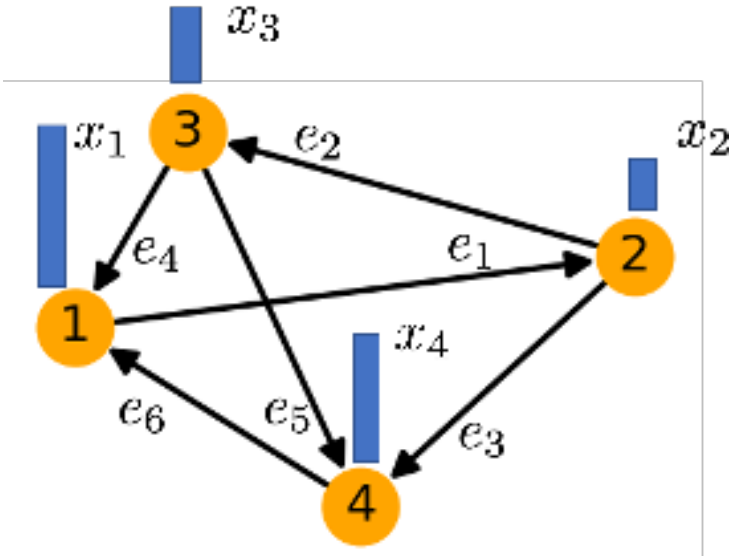


- Let $\boldsymbol{B}$ be the oriented weighted incidence matrix of $\mathcal{G}(\mathcal{V}, \mathcal{E})$ and let $\boldsymbol{x} \in \mathbb{R}^{|\mathcal{V}|}$ be a graph signal, i.e. a vector whose index set is $\mathcal{V}$.
- We can define the edge signal indexed by the set $\mathcal{E}$:

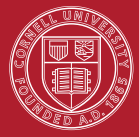
$$\boldsymbol{e} = \boldsymbol{B}^\top \boldsymbol{x}.$$

**Taking "derivatives" on a graph?**

The vector $\boldsymbol{e} = \boldsymbol{B}^\top \boldsymbol{x}$ contains the differences of all nodes values $(x_v - x_u)$ $\forall (u, v) \in \mathcal{E}$.



# Outline



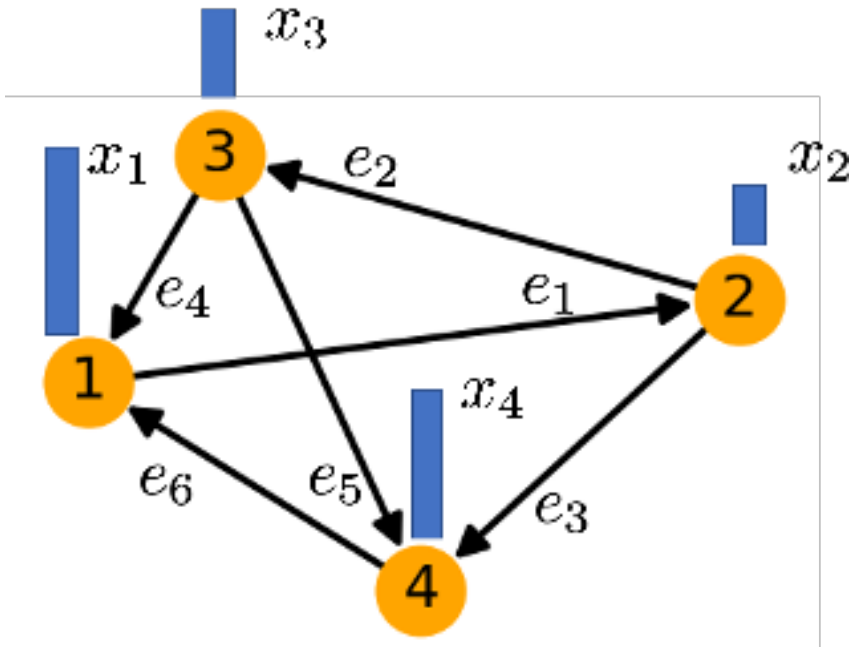
Cornell University

# Flows and node signals

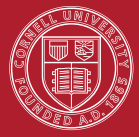


- Let $\boldsymbol{B}$ be the oriented weighted incidence matrix of $\mathcal{G}(\mathcal{V}, \mathcal{E})$.
- Let $\boldsymbol{x} \in \mathbb{R}^{|\mathcal{V}|}$ be a graph signal, i.e. a vector whose index set is $\mathcal{V}$.
- We can define the edge signal indexed by the set $\mathcal{E}$:

$$\boldsymbol{e} = \boldsymbol{B}^\top \boldsymbol{x}.$$



# The difference of node signals



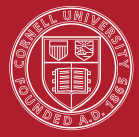


- The entries of $\boldsymbol{e}$ are the (weighted) difference between the values of $\boldsymbol{x}$ connected by the incident edge corresponding to the row of $\boldsymbol{B}^\top$ (or column of $\boldsymbol{B}$).
- Using the notation $a_k := [\boldsymbol{a}]_k$, specifically, for edge $k = (u, v)$:

$$e_k = [\boldsymbol{B}^\top \boldsymbol{x}]_k = w_{uv}(x_u - x_v) \quad k = (u, v) \in \mathcal{E}$$

- Note that, if $\boldsymbol{x} = \mathbf{1}$, then $\boldsymbol{e} = \boldsymbol{B}^\top \boldsymbol{x} = \mathbf{0}$. But, as long as the graph is connected, the null space of $\boldsymbol{B}^\top$ contains only vectors $\propto \mathbf{1}$.

# Cycles and the incidence matrix





- In general the rank of a matrix is less or equal than the minimum between the number of rows and column.
- Whenever it is lower than that we say that the matrix looses rank. Note that $\boldsymbol{B}$ is $|\mathcal{V}| \times |\mathcal{E}|$.
- In meshed networks often $|\mathcal{E}| \geq |\mathcal{V}|$ ($\boldsymbol{B}$ is often "fat"). A non obvious fact is that:

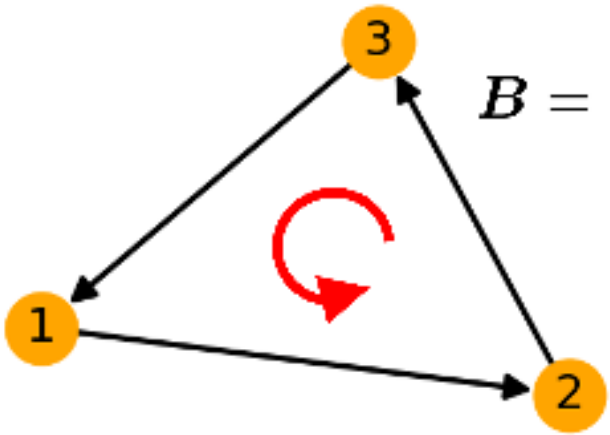
**Lemma**

:The rank of rank($\boldsymbol{B}$) is equal to $|\mathcal{E}| - |\mathcal{V}| + c$ where $c$ is the number of connected components of the graph.

# Cycles - Kirchoff's voltage law



- We can check what happens when the graph has cycles looking at the smallest possible example:




$$\boldsymbol{B} = \begin{bmatrix} -1. & 0. & 1. \\ 1. & -1. & 0. \\ 0. & 1. & -1. \end{bmatrix} \Rightarrow \boldsymbol{e} = \boldsymbol{B}^T \boldsymbol{x} = \begin{bmatrix} x_2 - x_1 \\ x_3 - x_2 \\ x_1 - x_3 \end{bmatrix}$$

$$\Rightarrow \; e_1 + e_2 + e_3 = 0$$

## Kichoff's voltage law

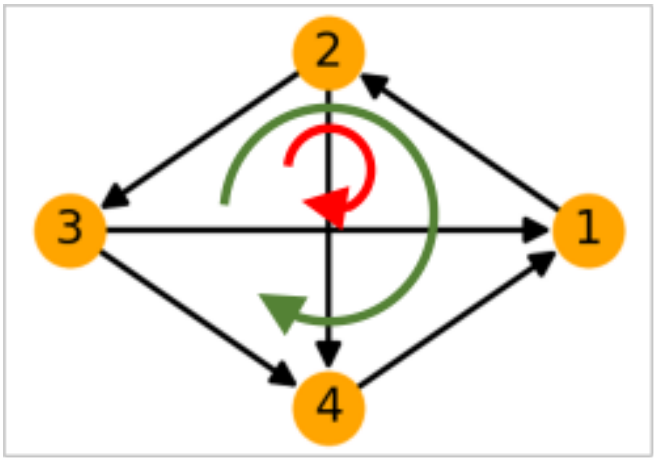
For graph signals $\boldsymbol{x}$, differences of nodal values $x_i$ (a.k.a. potentials) around a cycle add up to zero.

- This is not specific to electrical networks - it is due to the fact that there is no residual "charge" allowed to remain at the node (no sink, no injection)

# Larger graphs and cycles

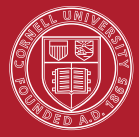


- Let us check a case with multiple cycles, longer than 3 hops.
- We can observe from the following example that, again, the set of columns in $B$ corresponding to edges of a loop add up to zero.




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

# Flows





- Let us now consider a signal $\boldsymbol{f}$ that is indexed by the edge set $\mathcal{E}$.

$$y_u = [\boldsymbol{B}\boldsymbol{f}]_u = \overbrace{\sum_{e \in \mathcal{N}_u^+} f_e}^{\text{incoming flow } f_u^{\text{in}}} - \overbrace{\sum_{e \in \mathcal{N}_u^-} f_e}^{\text{outgoing flow } f_u^{\text{out}}}$$

**Kirchoff's current law**

In an electrical network flows add up to zero at every node:

$$\boldsymbol{y} = \boldsymbol{B}\boldsymbol{f} = \boldsymbol{0} \quad \text{(conservation of charge)}$$

# Network flows - KVL and KCL + Ohm's law

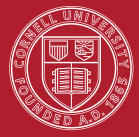


## Network flows

If the graph is connected and $\boldsymbol{x}$ is such that $\boldsymbol{f} = \text{diag}(\boldsymbol{w})\boldsymbol{B}^\top \boldsymbol{x} \neq 0$ (Ohm's law) then $\boldsymbol{y} = \boldsymbol{B}\boldsymbol{f} = \boldsymbol{B}\text{diag}(\boldsymbol{w})\boldsymbol{B}^\top \boldsymbol{x} \neq \boldsymbol{0}$.

- $\boldsymbol{B}\boldsymbol{f} = \boldsymbol{0}$ is true because there is no *external* injection or withdrawal source at the nodes (i.e. think of a battery).

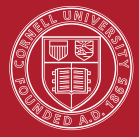
# Flows





- Incidence matrices naturally appear in describing flows not only of electric currents information, transportation and pipelines, which are instances of flow networks.
- Kirchoff's law holds also for streets intersections or pipelines junctions that do not leak (nor nor are pumping or drawing flows).
- If vehicles stop in a street or packets are held at a router in a buffer then Kirchoff's law does not hold
- In general, in a connected network, unbalanced potential ($e = B^T x \neq \mathbf{0}$) is compatible with unbalanced flows ($f_u^{\text{in}} \neq f_u^{\text{out}}$).

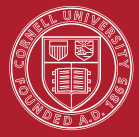
# Flows





- Incidence matrices naturally appear in describing flows not only of electric currents information, transportation and pipelines, which are instances of flow networks.
- Kirchoff's law holds also for streets intersections or pipelines junctions that do not leak (nor nor are pumping or drawing flows).
- If vehicles stop in a street or packets are held at a router in a buffer then Kirchoff's law does not hold
- In general, in a connected network, unbalanced potential ($e = B^T x \neq 0$) is compatible with unbalanced flows ($f_u^{\text{in}} \neq f_u^{\text{out}}$).

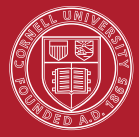
# Flows





- Incidence matrices naturally appear in describing flows not only of electric currents information, transportation and pipelines, which are instances of flow networks.
- Kirchoff's law holds also for streets intersections or pipelines junctions that do not leak (nor nor are pumping or drawing flows).
- If vehicles stop in a street or packets are held at a router in a buffer then Kirchoff's law does not hold
- In general, in a connected network, unbalanced potential ($e = B^T x \neq 0$) is compatible with unbalanced flows ($f_u^{\text{in}} \neq f_u^{\text{out}}$).

# Flows



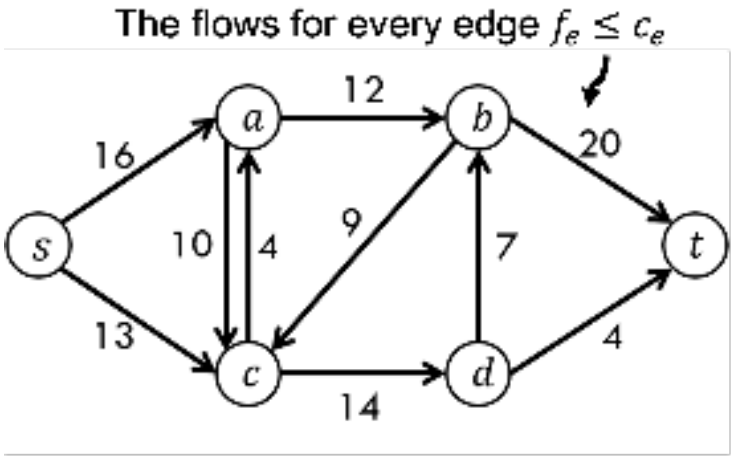


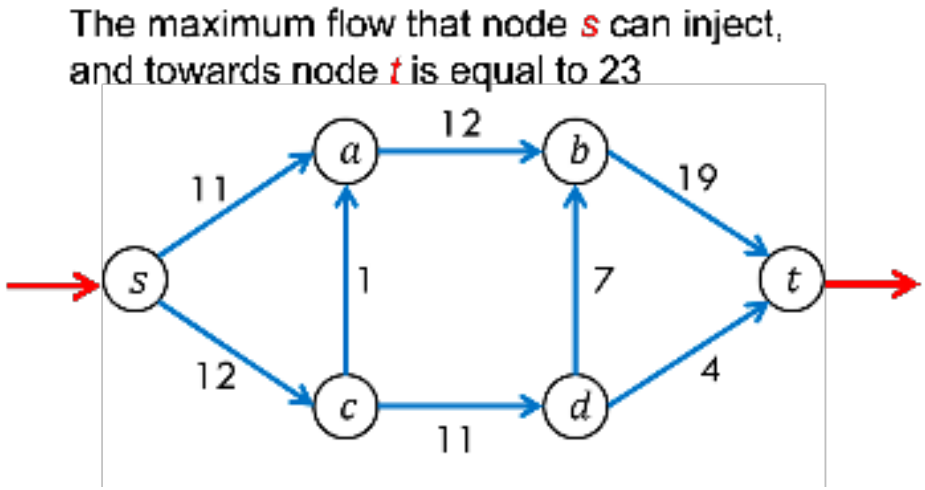
- Incidence matrices naturally appear in describing flows not only of electric currents information, transportation and pipelines, which are instances of flow networks.
- Kirchoff's law holds also for streets intersections or pipelines junctions that do not leak (nor nor are pumping or drawing flows).
- If vehicles stop in a street or packets are held at a router in a buffer then Kirchoff's law does not hold
- In general, in a connected network, unbalanced potential ($\boldsymbol{e} = \boldsymbol{B}^T \boldsymbol{x} \neq \mathbf{0}$) is compatible with unbalanced flows ($f_u^{\text{in}} \neq f_u^{\text{out}}$).

# Network flows problems

Cornell University

- This is the case in network flow problems, where a source node $s$ is transferring a "mass" $y_s$ to a destination $t$.
- In such problems there are link capacity limits, $\overline{f}_e$ meaning that: $f_e \leq \overline{f}_e$
- Intermediate nodes obey Kirchoff's law $f_u^{\text{in}} = f_u^{\text{out}} \rightarrow y_u = 0$ while $y_s = f_s^{\text{out}} - f_s^{\text{in}} > 0$ while $y_t = f_t^{\text{out}} - f_t^{\text{in}} = -y_s < 0$

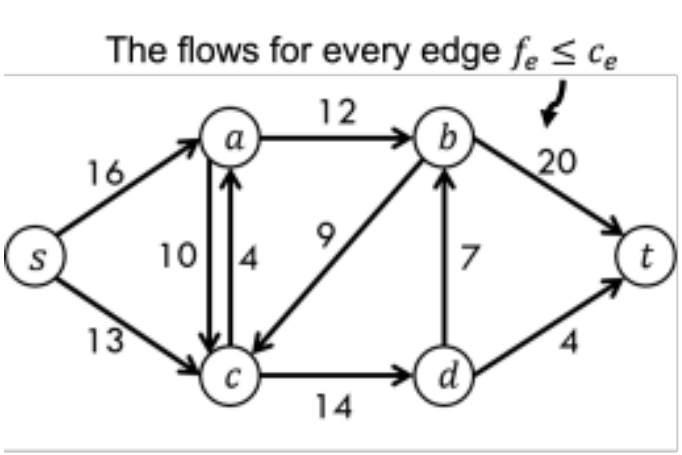


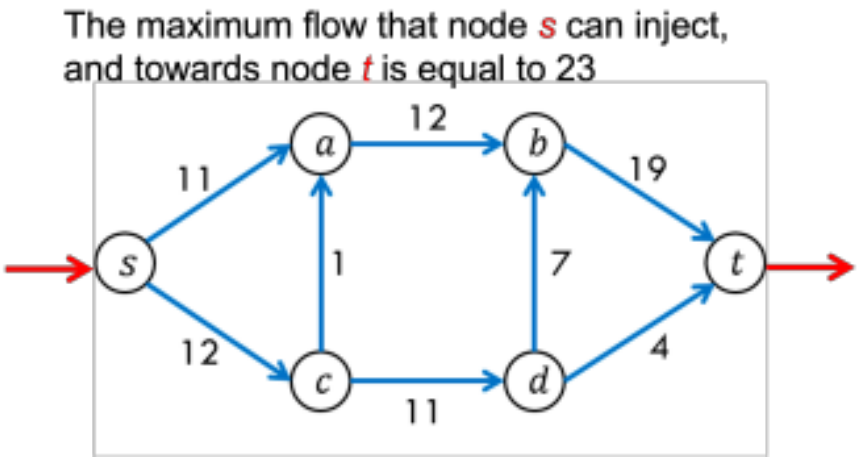



# Network flows



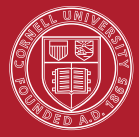
- Note that in the example $f_u^{\text{in}} = f_u^{\text{out}}$ for all $u \in \{a, b, c, d\}$




- However, $y_s = f_s^{\text{out}} - \overbrace{f_s^{\text{in}}}^{=0} = 23$ and $y_t = \overbrace{f_t^{\text{out}}}^{=0} - f_t^{\text{in}} = -23$.
- Since both $y_s$ and $y_t$ are non zero, this implies that for this example: $\boldsymbol{y} = \boldsymbol{B}\boldsymbol{f} \neq 0$.

# Max-flow problem



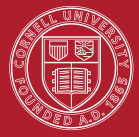


- Maximize the total amount of flow subject to flow constraints, i.e.: $y_s$ from $s$ to $t$ to the flow constraints.

$$\begin{aligned} \max \ y_s \ \text{ subject to} \\ & [\boldsymbol{B}\boldsymbol{f}]_u = 0, u \in \mathcal{V}/\{s,t\}, \\ & [\boldsymbol{B}\boldsymbol{f}]_s = -[\boldsymbol{B}\boldsymbol{f}]_t = y_s, \\ & 0 \leq \boldsymbol{f} \leq \overline{\boldsymbol{f}}, \quad , y_s \geq 0 \end{aligned}$$

- This is a linear program (LP), one can use in principle a simplex method for the answer.

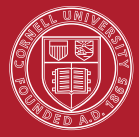
# Max-flow and min-cut





- An equivalent formulation that is more computationally efficient is based on identifying the bottleneck for the network:
  1. Set as a cost for removing an edge the capacity $\overline{f}_e$
  2. Find the minimum edge cut, such that $\mathcal{E}_{\text{cut}} \subset \mathcal{E}$ disconnects $s$ and $t$ with minimum cost.
- Flows that on a path from $s$ to $t$ will include an edge in $\mathcal{E}_{\text{min-cut}}$.
- Across all edges in the path the flow must be $\leq \overline{f}_e \in \mathcal{E}_{\text{min-cut}}$.

# Ford-Fulkerson algorithm



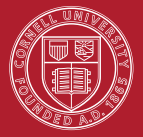


- The algorithm based on a proven property of the solution. Its complexity is $O(|\mathcal{V}| + |\mathcal{E}|)$.
  - A flow $\boldsymbol{f}$ can always be decomposed in two components $\boldsymbol{f} = \boldsymbol{f}_p + \boldsymbol{f}_c$, where:
    1. $\boldsymbol{f}_p$ is a sum of flows that run along paths connecting $s$ and $d$
    2. $\boldsymbol{f}_c$ is a sum *circulations*, i.e. $\boldsymbol{B}\boldsymbol{f}_c = \boldsymbol{0}$
  - The optimum solution has always $\boldsymbol{f}_c = \boldsymbol{0}$.
  - The flow components of $\boldsymbol{f}_p$ cannot be larger than the capacity of the edge $e \in \mathcal{E}$_min-cut that they cross.
  - So all is needed is to find all paths[1], identify $\mathcal{E}_{\text{min-cut}}$, and

$$\max y_s = \sum_{e \in \mathcal{E}_{\text{min-cut}}} \overline{f}_e$$

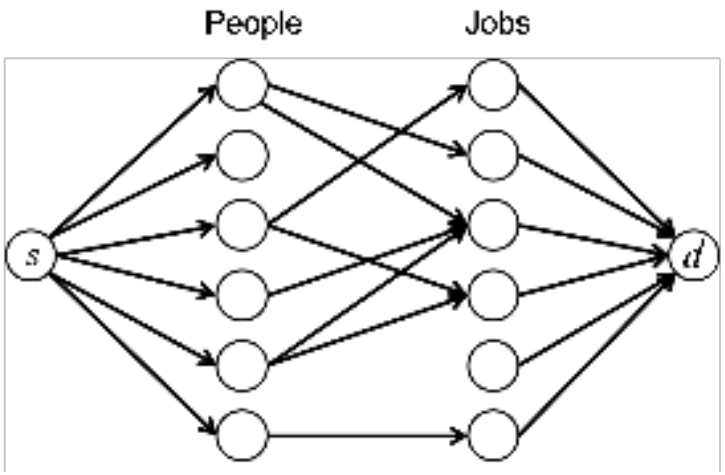
[1] We can use the Depth First Search (DFS) algorithm for that

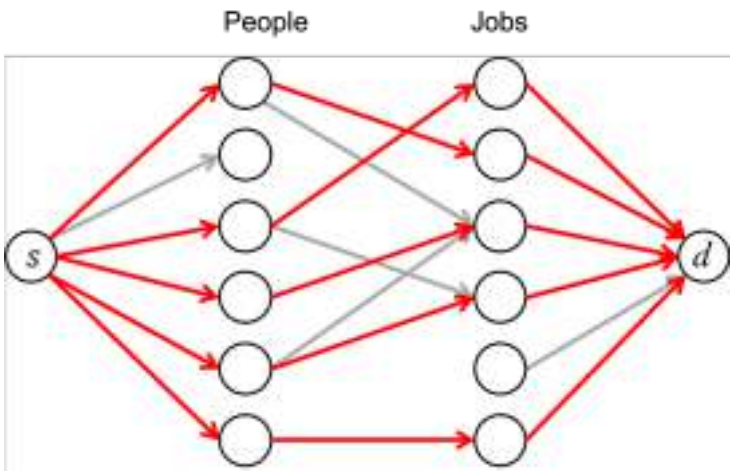
# Maximum Bipartite matching




Cornell University

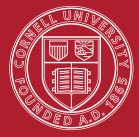
- The same algorithm can be used to solve another problem called *maximum bipartite matching*[2] where the bipartite graph sets are $\mathcal{A}$ and $\mathcal{B}$, say $\mathcal{A}$ are people and $\mathcal{B}$ jobs.
- One adds a source and a destination and capacities (e.g. the number of jobs available for a position).
- The active intermediate links in the optimum flow solution are going to be the maximum number of matches.



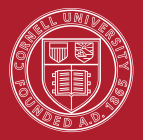

[2] See https://www.cs.cmu.edu/~ckingsf/bioinfo-lectures/matching.pdf

# Outline



Cornell University

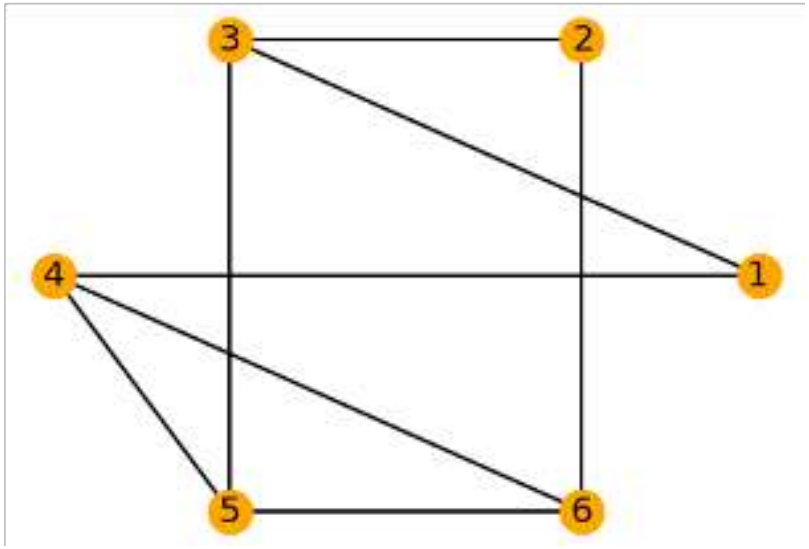
# Unweighted graph Laplacian matrix



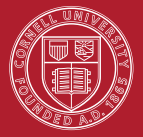

- Let $\boldsymbol{d}$ be the vector of nodal degrees, i.e. $d_v = |\mathcal{N}_v|, \quad \forall v \in \mathcal{V}$.
- The Laplacian is defined as ($\boldsymbol{B}$ is the oriented incidence matrix for the unweighted graph):

$$\boldsymbol{L} = \text{diag}(\boldsymbol{d}) - \boldsymbol{A} = \boldsymbol{B}\boldsymbol{B}^\top$$

$$\boldsymbol{L} = \begin{bmatrix} 2 & 0 & -1 & -1 & 0 & 0 \\ 0 & 2 & -1 & 0 & 0 & -1 \\ -1 & -1 & 3 & 0 & -1 & 0 \\ -1 & 0 & 0 & 3 & -1 & -1 \\ 0 & 0 & -1 & -1 & 3 & -1 \\ 0 & -1 & 0 & -1 & -1 & 3 \end{bmatrix}$$



# Weighted graph Laplacian matrix

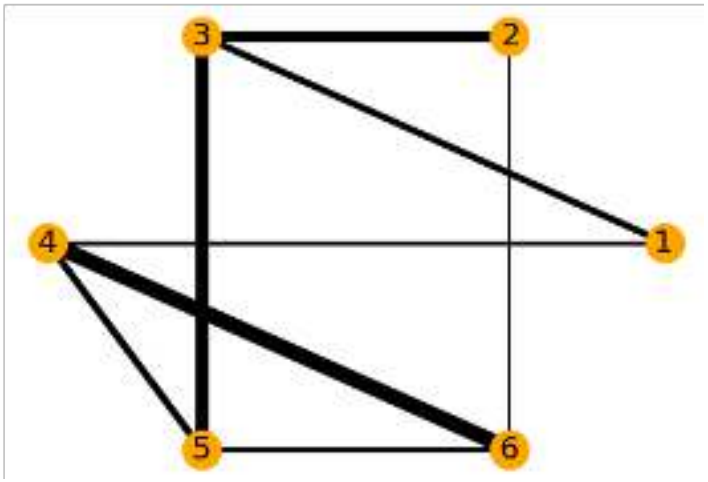



Cornell University

- For a weighted graph, the definition relies on the weighted adjacency matrix, or the oriented incidence matrix as follows:

$$\boldsymbol{L} = \text{diag}(\boldsymbol{A}\mathbf{1}) - \boldsymbol{A} = \boldsymbol{B}\text{diag}(\boldsymbol{w})\boldsymbol{B}^\top.$$

$$\boldsymbol{L} = \begin{bmatrix} 7.6 & 0. & -4.5 & -3.1 & 0. & 0. \\ 0. & 9.35 & -7.25 & 0. & 0. & -2.1 \\ -4.5 & -7.25 & 20.25 & 0. & -8.5 & 0. \\ -3.1 & 0. & 0. & 17.95 & -4.75 & -10.1 \\ 0. & 0. & -8.5 & -4.75 & 16.5 & -3.25 \\ 0. & -2.1 & 0. & -10.1 & -3.25 & 15.45 \end{bmatrix}$$




- Note that the Laplacian (unweighted or weighted) starts from an undirected graph $\boldsymbol{A} = \boldsymbol{A}^\top$ and is $\boldsymbol{L} = \boldsymbol{L}^\top$, so it is symmetric.

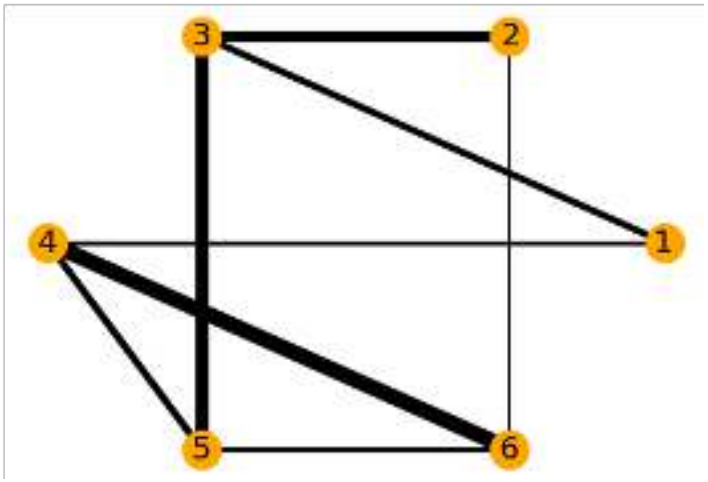
# Weighted graph Laplacian matrix



- For a weighted graph, the definition relies on the weighted adjacency matrix, or the oriented incidence matrix as follows:

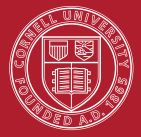
$$\boldsymbol{L} = \text{diag}(\boldsymbol{A}\mathbf{1}) - \boldsymbol{A} = \boldsymbol{B}\text{diag}(\boldsymbol{w})\boldsymbol{B}^\top.$$

$$\boldsymbol{L} = \begin{bmatrix} 7.6 & 0. & -4.5 & -3.1 & 0. & 0. \\ 0. & 9.35 & -7.25 & 0. & 0. & -2.1 \\ -4.5 & -7.25 & 20.25 & 0. & -8.5 & 0. \\ -3.1 & 0. & 0. & 17.95 & -4.75 & -10.1 \\ 0. & 0. & -8.5 & -4.75 & 16.5 & -3.25 \\ 0. & -2.1 & 0. & -10.1 & -3.25 & 15.45 \end{bmatrix}$$




- Note that the Laplacian (unweighted or weighted) starts from an undirected graph $\boldsymbol{A} = \boldsymbol{A}^\top$ and is $\boldsymbol{L} = \boldsymbol{L}^\top$, so it is symmetric.

# Rescaling the Laplacian matrix





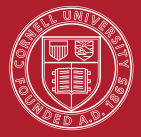
- Normalized Laplacian: It is symmetric and defined as:

$$\begin{aligned}\boldsymbol{L}^{\text{norm}} &= \text{diag}(\boldsymbol{d})^{-\frac{1}{2}}\boldsymbol{L}\text{diag}(\boldsymbol{d})^{-\frac{1}{2}} \\ &= \overbrace{\text{diag}(\boldsymbol{d})^{-\frac{1}{2}}\text{diag}(\boldsymbol{d})\text{diag}(\boldsymbol{d})^{-\frac{1}{2}}}^{\boldsymbol{I}} - \text{diag}(\boldsymbol{d})^{-\frac{1}{2}}\boldsymbol{A}\text{diag}(\boldsymbol{d})^{-\frac{1}{2}} \\ &= \boldsymbol{I} - \text{diag}(\boldsymbol{d})^{-\frac{1}{2}}\boldsymbol{A}\text{diag}(\boldsymbol{d})^{-\frac{1}{2}}\end{aligned}$$

$$L_{uv}^{\text{norm}} = \begin{cases} 1 & \text{if } u = v \text{ and } d_u \neq 0 \\ -\frac{1}{\sqrt{d_u}\sqrt{d_v}} & \text{if } (u,v) \in \mathcal{E} \\ 0 & \text{else} \end{cases}$$

- Also $\boldsymbol{L}^{\text{norm}} = (\boldsymbol{L}^{\text{norm}})^\top$ is symmetric.

# Rescaling the Laplacian matrix



Cornell University

- Normalized Laplacian: It is symmetric and defined as:

$$\begin{aligned}\boldsymbol{L}^{\text{norm}} &= \text{diag}(\boldsymbol{d})^{-\frac{1}{2}}\boldsymbol{L}\text{diag}(\boldsymbol{d})^{-\frac{1}{2}} \\ &= \overbrace{\text{diag}(\boldsymbol{d})^{-\frac{1}{2}}\text{diag}(\boldsymbol{d})\text{diag}(\boldsymbol{d})^{-\frac{1}{2}}}^{\boldsymbol{I}} - \text{diag}(\boldsymbol{d})^{-\frac{1}{2}}\boldsymbol{A}\text{diag}(\boldsymbol{d})^{-\frac{1}{2}} \\ &= \boldsymbol{I} - \text{diag}(\boldsymbol{d})^{-\frac{1}{2}}\boldsymbol{A}\text{diag}(\boldsymbol{d})^{-\frac{1}{2}}\end{aligned}$$

$$L_{uv}^{\text{norm}} = \begin{cases} 1 & \text{if } u = v \text{ and } d_u \neq 0 \\ -\frac{1}{\sqrt{d_u}\sqrt{d_v}} & \text{if } (u,v) \in \mathcal{E} \\ 0 & \text{else} \end{cases}$$

- Also $\boldsymbol{L}^{\text{norm}} = (\boldsymbol{L}^{\text{norm}})^\top$ is symmetric.

# Rescaling the Laplacian matrix: weighted graph

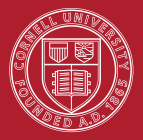


Normalized Laplacian: In this case replace $\boldsymbol{d} = \boldsymbol{A}\mathbf{1}$, where $A_{uv} = w_{uv}$ if $(u, v) \in \mathcal{E}$ and 0 else.

$$\boldsymbol{L}^{\text{norm}} = \text{diag}(\boldsymbol{d})^{-\frac{1}{2}} \boldsymbol{L} \text{diag}(\boldsymbol{d})^{-\frac{1}{2}} = \boldsymbol{I} - \text{diag}(\boldsymbol{d})^{-\frac{1}{2}} \boldsymbol{A} \text{diag}(\boldsymbol{d})^{-\frac{1}{2}}$$

$$L_{uv}^{\text{norm}} = \begin{cases} 1 & \text{if u=v and } d_u \neq 0 \\ -\dfrac{w_{uv}}{\sqrt{\sum_{v \in \mathcal{N}_u} w_{uv}} \sqrt{\sum_{u \in \mathcal{N}_v} w_{vu}}} & \text{if } (u, v) \in \mathcal{E} \\ 0 & \text{else} \end{cases}$$

# Another Laplacian matrix rescaling





Random walk Laplacian: Is defined as:

$$\boldsymbol{L}^{\text{rw}} = \text{diag}(\boldsymbol{d})^{-1}\boldsymbol{L} = \overbrace{\text{diag}(\boldsymbol{d})^{-1}\text{diag}(\boldsymbol{d})}^{\boldsymbol{I}} - \text{diag}(\boldsymbol{d})^{-1}\boldsymbol{A}$$
$$\equiv \boldsymbol{I} - \text{diag}(\boldsymbol{d})^{-1}\boldsymbol{A}$$

$$L_{uv}^{\text{rw}} = \begin{cases} 1 & \text{if u=v and } d_u \neq 0 \\ -\frac{1}{d_u} & \text{if } (u,v) \in \mathcal{E} \\ 0 & \text{else} \end{cases} \quad \text{unweighted}$$

$$L_{uv}^{\text{rw}} = \begin{cases} 1 & \text{if u=v and } d_u \neq 0 \\ -\frac{1}{\sum_{v \in \mathcal{N}_u} w_{uv}} & \text{if } (u,v) \in \mathcal{E} \\ 0 & \text{else} \end{cases} \quad \text{weighted}$$

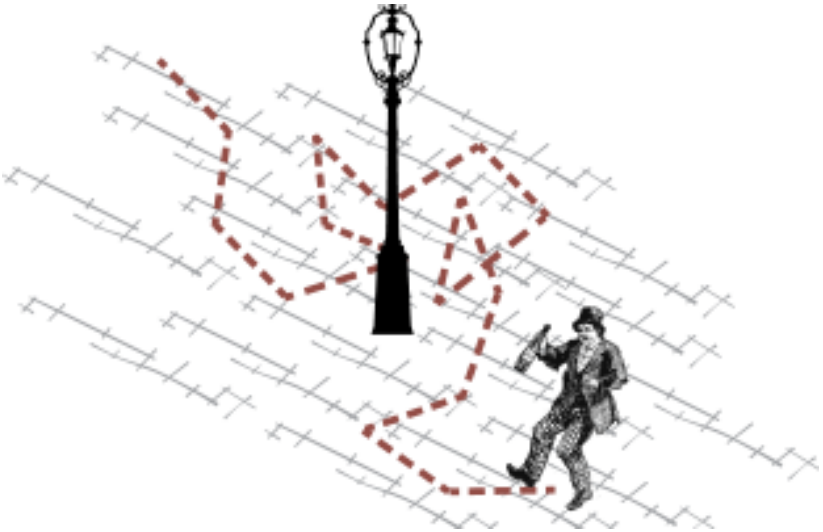
This definition holds in general, for directed and undirected graphs.

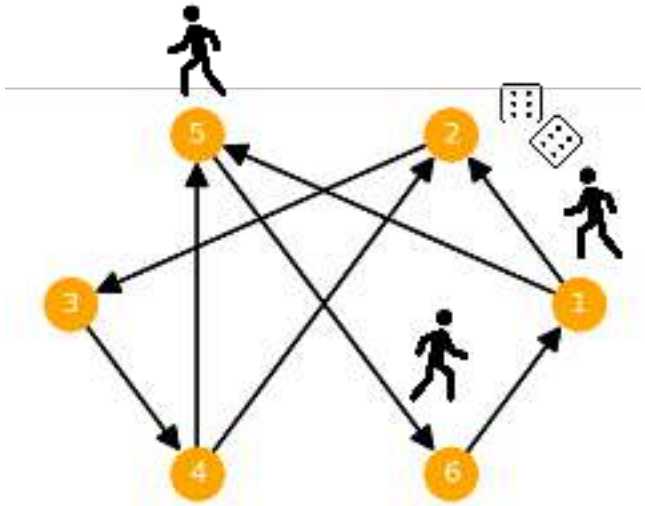
# The Random walk Laplacian


- The namesake of random walk Laplacian is one of the most well known network dynamics: **the random walk process**

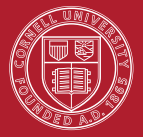
## Random Walk

A random walk starting at node $v$ is a walk where the edges are chosen uniformly at random, by selecting the next node to land on randomly in the (out) neighborhood. The weights $w_{uv}$ from $\boldsymbol{W} = \boldsymbol{I} - \alpha \boldsymbol{L}^{\text{rw}}$ are the probabilities of going from $v$ to $u$. (... more in a few lectures).





# Observations on $L$





- We said $\boldsymbol{L}$ is symmetric, $\boldsymbol{L} = \boldsymbol{L}^\top$ and real so it has an eigenvalue decomposition $\boldsymbol{L} = \boldsymbol{U}\text{diag}(\boldsymbol{\lambda})\boldsymbol{U}^\top$
- It is easy to verify based on the definition that $\boldsymbol{L}\mathbf{1} = \mathbf{0}$:

$$\boldsymbol{L}\mathbf{1} = (\text{diag}(\boldsymbol{A}\mathbf{1}) - \boldsymbol{A})\mathbf{1} = \text{diag}(\boldsymbol{A}\mathbf{1})\mathbf{1} - \boldsymbol{A}\mathbf{1} = \boldsymbol{A}\mathbf{1} - \boldsymbol{A}\mathbf{1} \equiv \mathbf{0}$$

since $\text{diag}(\boldsymbol{x})\mathbf{1} = \text{diag}(\mathbf{1})\boldsymbol{x} = \boldsymbol{I}\boldsymbol{x} = \boldsymbol{x}$

- It is easy to verity that the property is valid also for the normalized Laplacians, since

$$\boldsymbol{L}^{\text{rw}}\mathbf{1} = \text{diag}(\boldsymbol{d})^{-1}\boldsymbol{L}\mathbf{1} = \text{diag}(\boldsymbol{d})^{-1}\mathbf{0}.$$

- This means that $\mathbf{1}$ is a null-vector of $\boldsymbol{L}$. Thus, with eigenvalues in decreasing order, $\lambda_1 \geq \lambda_2 \geq \ldots$, at least the smallest eigenvalue must be $\lambda_N = 0$, by definition.

# Observations on $L$



- We said $\boldsymbol{L}$ is symmetric, $\boldsymbol{L} = \boldsymbol{L}^\top$ and real so it has an eigenvalue decomposition $\boldsymbol{L} = \boldsymbol{U}\text{diag}(\boldsymbol{\lambda})\boldsymbol{U}^\top$
- It is easy to verify based on the definition that $\boldsymbol{L}\mathbf{1} = \mathbf{0}$:

$$\boldsymbol{L}\mathbf{1} = (\text{diag}(\boldsymbol{A}\mathbf{1}) - \boldsymbol{A})\mathbf{1} = \text{diag}(\boldsymbol{A}\mathbf{1})\mathbf{1} - \boldsymbol{A}\mathbf{1} = \boldsymbol{A}\mathbf{1} - \boldsymbol{A}\mathbf{1} \equiv \mathbf{0}$$

since $\text{diag}(\boldsymbol{x})\mathbf{1} = \text{diag}(\mathbf{1})\boldsymbol{x} = \boldsymbol{I}\boldsymbol{x} = \boldsymbol{x}$

- It is easy to verity that the property is valid also for the normalized Laplacians, since

$$\boldsymbol{L}^{\text{rw}}\mathbf{1} = \text{diag}(\boldsymbol{d})^{-1}\boldsymbol{L}\mathbf{1} = \text{diag}(\boldsymbol{d})^{-1}\mathbf{0}.$$

- This means that $\mathbf{1}$ is a null-vector of $\boldsymbol{L}$. Thus, with eigenvalues in decreasing order, $\lambda_1 \geq \lambda_2 \geq \ldots$, at least the smallest eigenvalue must be $\lambda_N = 0$, by definition.

# Observations on $\boldsymbol{L}$



- We said $\boldsymbol{L}$ is symmetric, $\boldsymbol{L} = \boldsymbol{L}^\top$ and real so it has an eigenvalue decomposition $\boldsymbol{L} = \boldsymbol{U}\text{diag}(\boldsymbol{\lambda})\boldsymbol{U}^\top$
- It is easy to verify based on the definition that $\boldsymbol{L}\mathbf{1} = \mathbf{0}$:

$$\boldsymbol{L}\mathbf{1} = (\text{diag}(\boldsymbol{A}\mathbf{1}) - \boldsymbol{A})\mathbf{1} = \text{diag}(\boldsymbol{A}\mathbf{1})\mathbf{1} - \boldsymbol{A}\mathbf{1} = \boldsymbol{A}\mathbf{1} - \boldsymbol{A}\mathbf{1} \equiv \mathbf{0}$$

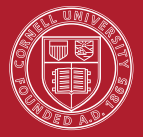
since $\text{diag}(\boldsymbol{x})\mathbf{1} = \text{diag}(\mathbf{1})\boldsymbol{x} = \boldsymbol{I}\boldsymbol{x} = \boldsymbol{x}$

- It is easy to verity that the property is valid also for the normalized Laplacians, since

$$\boldsymbol{L}^{\text{rw}}\mathbf{1} = \text{diag}(\boldsymbol{d})^{-1}\boldsymbol{L}\mathbf{1} = \text{diag}(\boldsymbol{d})^{-1}\mathbf{0}.$$

- This means that $\mathbf{1}$ is a null-vector of $\boldsymbol{L}$. Thus, with eigenvalues in decreasing order, $\lambda_1 \geq \lambda_2 \geq \ldots$, at least the smallest eigenvalue must be $\lambda_N = 0$, by definition.

# Observations on $\boldsymbol{L}$





- We said $\boldsymbol{L}$ is symmetric, $\boldsymbol{L} = \boldsymbol{L}^\top$ and real so it has an eigenvalue decomposition $\boldsymbol{L} = \boldsymbol{U}\text{diag}(\boldsymbol{\lambda})\boldsymbol{U}^\top$
- It is easy to verify based on the definition that $\boldsymbol{L}\mathbf{1} = \mathbf{0}$:

$$\boldsymbol{L}\mathbf{1} = (\text{diag}(\boldsymbol{A}\mathbf{1}) - \boldsymbol{A})\mathbf{1} = \text{diag}(\boldsymbol{A}\mathbf{1})\mathbf{1} - \boldsymbol{A}\mathbf{1} = \boldsymbol{A}\mathbf{1} - \boldsymbol{A}\mathbf{1} \equiv \mathbf{0}$$

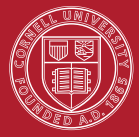
since $\text{diag}(\boldsymbol{x})\mathbf{1} = \text{diag}(\mathbf{1})\boldsymbol{x} = \boldsymbol{I}\boldsymbol{x} = \boldsymbol{x}$

- It is easy to verity that the property is valid also for the normalized Laplacians, since

$$\boldsymbol{L}^{\text{rw}}\mathbf{1} = \text{diag}(\boldsymbol{d})^{-1}\boldsymbol{L}\mathbf{1} = \text{diag}(\boldsymbol{d})^{-1}\mathbf{0}.$$

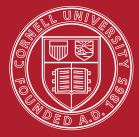
- This means that $\mathbf{1}$ is a null-vector of $\boldsymbol{L}$. Thus, with eigenvalues in decreasing order, $\lambda_1 \geq \lambda_2 \geq \ldots$, at least the smallest eigenvalue must be $\lambda_N = 0$, by definition.

# Outline



Cornell University

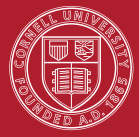
# Spectral graph theory





- Laplacians properties are extensively studied in *spectral graph theory*.
- By spectrum of the graph Laplacian we are referring to the properties of the eigenvectors and eigenvalues from $\boldsymbol{L} = \boldsymbol{U}\text{diag}(\boldsymbol{\lambda})\boldsymbol{U}^\top$, where we assume the eigenvalues are in decreasing order $\lambda_1 \geq \lambda_2 \ldots \geq \lambda_{N-1} \geq \lambda_N = 0$.
- $\lambda_N = 0$ because $\boldsymbol{L}\mathbf{1} = 0$.
- Are there any more zero eigenvalues in general, or when is $\lambda_{N-k} = \lambda_{N-k+1} = \ldots = \lambda_N = 0$?

# Connected components





- We already mentioned this without proof in Lecture 3. The theorem is as follows:

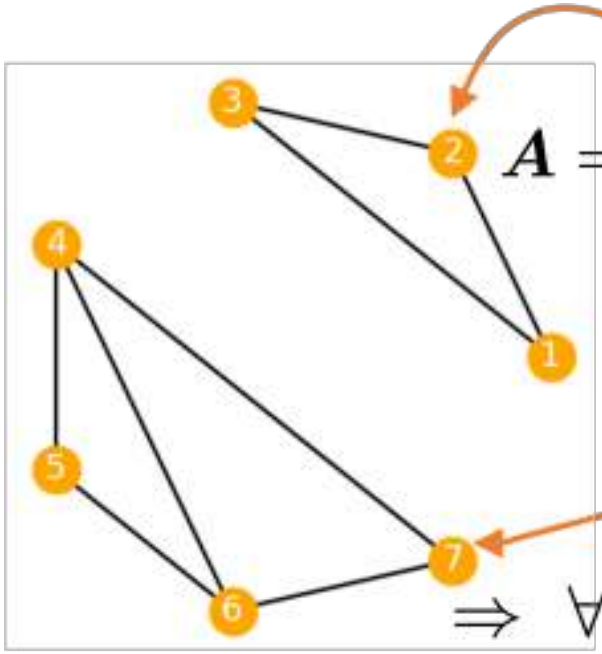
**Theorem (M. Fiedler)**

The nullity (number of zero eigenvalues) of $\boldsymbol{L}$ equals the number of connected components of a graph.

# Laplacian matrix for disconnected graphs




- It is not hard to see that it is the case.

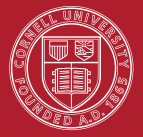



$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_1 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{A}_2 \end{bmatrix} \Rightarrow \boldsymbol{L} = \begin{bmatrix} \boldsymbol{L}_1 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{L}_2 \end{bmatrix}$$

Another null vector

$$\Rightarrow \forall \alpha \quad \boldsymbol{L} \begin{bmatrix} \alpha \boldsymbol{1} \\ \boldsymbol{1} \end{bmatrix} = \begin{bmatrix} \alpha \boldsymbol{L}_1 \boldsymbol{1} \\ \boldsymbol{L}_2 \boldsymbol{1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix} = \boldsymbol{0}$$

# Laplacian matrix and connectivity



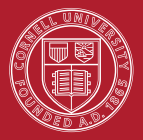
Cornell University

- The property is generalized to the case of $k$ components, since sorting appropriately the nodes, the matrix is a block diagonal

$$\boldsymbol{L} = \begin{bmatrix} \boldsymbol{L}_1 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{L}_2 & \dots & \mathbf{0} \\ \vdots & \ddots & \ddots & \vdots \\ \mathbf{0} & \dots & \mathbf{0} & \boldsymbol{L}_k \end{bmatrix}$$

- Let the eigenvalue decomposition of each Laplacian be $\boldsymbol{L}_k = \boldsymbol{U}_k \boldsymbol{\Lambda}_k \boldsymbol{U}_k^T$. We know each $\boldsymbol{L}_k$ has nullity 1, so this $\boldsymbol{L}$ has nullity $k$, which proves Fiedler's theorem.
- The eigenvalue decomposition of a block diagonal matrix has block diagonal eigenvector matrices. Shown in the next slide ....

# Laplacian matrix and connectivity



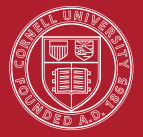
Cornell University

- The property is generalized to the case of $k$ components, since sorting appropriately the nodes, the matrix is a block diagonal

$$\boldsymbol{L} = \begin{bmatrix} \boldsymbol{L}_1 & \boldsymbol{0} & \dots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{L}_2 & \dots & \boldsymbol{0} \\ \vdots & \ddots & \ddots & \vdots \\ \boldsymbol{0} & \dots & \boldsymbol{0} & \boldsymbol{L}_k \end{bmatrix}$$

- Let the eigenvalue decomposition of each Laplacian be $\boldsymbol{L}_k = \boldsymbol{U}_k \boldsymbol{\Lambda}_k \boldsymbol{U}_k^T$. We know each $\boldsymbol{L}_k$ has nullity 1, so this $\boldsymbol{L}$ has nullity $k$, which proves Fiedler's theorem.
- The eigenvalue decomposition of a block diagonal matrix has block diagonal eigenvector matrices. Shown in the next slide ....

# Laplacian matrix and connectivity



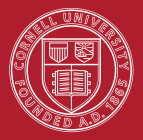


- The property is generalized to the case of $k$ components, since sorting appropriately the nodes, the matrix is a block diagonal

$$\boldsymbol{L} = \begin{bmatrix} \boldsymbol{L}_1 & \boldsymbol{0} & \dots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{L}_2 & \dots & \boldsymbol{0} \\ \vdots & \ddots & \ddots & \vdots \\ \boldsymbol{0} & \dots & \boldsymbol{0} & \boldsymbol{L}_k \end{bmatrix}$$

- Let the eigenvalue decomposition of each Laplacian be $\boldsymbol{L}_k = \boldsymbol{U}_k \boldsymbol{\Lambda}_k \boldsymbol{U}_k^T$. We know each $\boldsymbol{L}_k$ has nullity 1, so this $\boldsymbol{L}$ has nullity $k$, which proves Fiedler's theorem.
- The eigenvalue decomposition of a block diagonal matrix has block diagonal eigenvector matrices. Shown in the next slide ....

# Laplacian matrix and connectivity





$$\boldsymbol{L} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\top = \begin{bmatrix} \boldsymbol{U}_1\boldsymbol{\Lambda}_1\boldsymbol{U}_1^\top & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{U}_2\boldsymbol{\Lambda}_2\boldsymbol{U}_2^\top & \dots & \mathbf{0} \\ \vdots & \ddots & \ddots & \vdots \\ \mathbf{0} & \dots & \mathbf{0} & \boldsymbol{U}_k\boldsymbol{\Lambda}_k\boldsymbol{U}_k^\top \end{bmatrix}$$

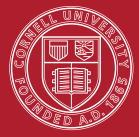
$$= \begin{bmatrix} \boldsymbol{U}_1 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{U}_2 & \dots & \mathbf{0} \\ \vdots & \ddots & \ddots & \vdots \\ \mathbf{0} & \dots & \mathbf{0} & \boldsymbol{U}_k \end{bmatrix} \begin{bmatrix} \boldsymbol{\Lambda}_1 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Lambda}_2 & \dots & \mathbf{0} \\ \vdots & \ddots & \ddots & \vdots \\ \mathbf{0} & \dots & \mathbf{0} & \boldsymbol{\Lambda}_k \end{bmatrix} \begin{bmatrix} \boldsymbol{U}_1^\top & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{U}_2^\top & \dots & \mathbf{0} \\ \vdots & \ddots & \ddots & \vdots \\ \mathbf{0} & \dots & \mathbf{0} & \boldsymbol{U}_k^\top \end{bmatrix}$$

# Laplacian matrix and connectivity



- If we already have ordered the nodes so that respective components have adjacent numbers, the Laplacian appears as block diagonal and the vectors in the null space are $(\mathbf{1}, \mathbf{0} \ldots, \mathbf{0}), (\mathbf{0}, \mathbf{1}, \mathbf{0} \ldots, \mathbf{0}), \ldots$.
- Even the nodes are shuffled, and the block diagonal structure is not immediately apparent, the sparsity of the $k$ null-vectors will point to the nodes that belong to a component.
- This fact will be exploited to find *graph clusters*, i.e. approximately disconnected groups of nodes

# Laplacian matrix and connectivity





- If we already have ordered the nodes so that respective components have adjacent numbers, the Laplacian appears as block diagonal and the vectors in the null space are $(\mathbf{1}, \mathbf{0} \ldots, \mathbf{0}), (\mathbf{0}, \mathbf{1}, \mathbf{0} \ldots, \mathbf{0}), \ldots$.
- Even the nodes are shuffled, and the block diagonal structure is not immediately apparent, the sparsity of the $k$ null-vectors will point to the nodes that belong to a component.
- This fact will be exploited to find *graph clusters*, i.e. approximately disconnected groups of nodes

# Laplacian matrix and connectivity

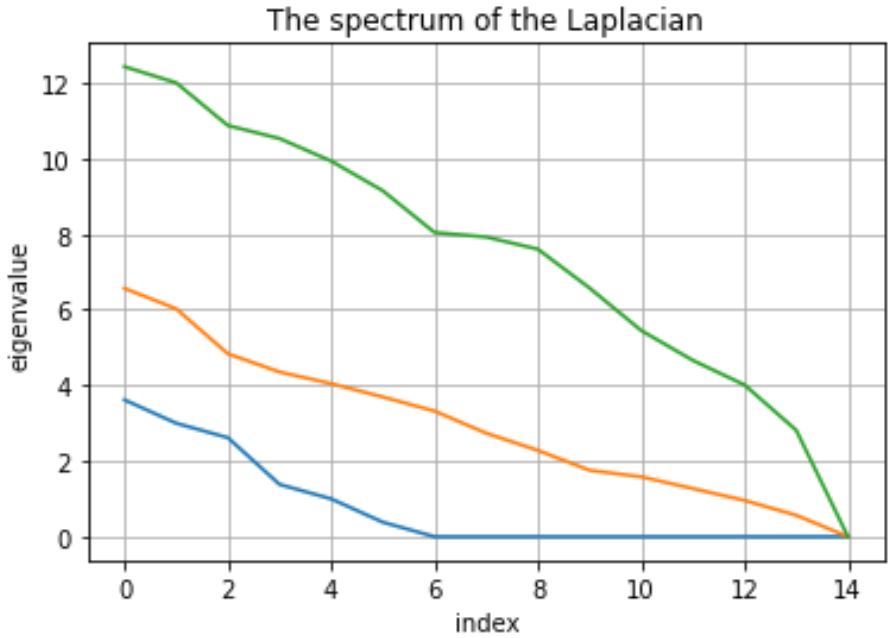


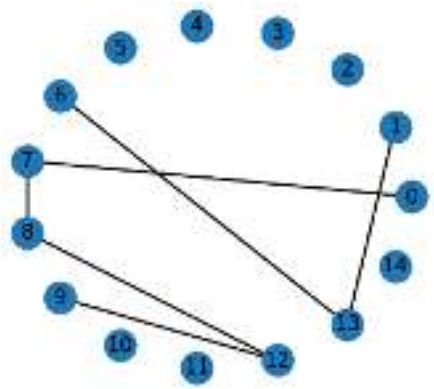
- If we already have ordered the nodes so that respective components have adjacent numbers, the Laplacian appears as block diagonal and the vectors in the null space are $(\mathbf{1}, \mathbf{0} \ldots, \mathbf{0}), (\mathbf{0}, \mathbf{1}, \mathbf{0} \ldots, \mathbf{0}), \ldots$.
- Even the nodes are shuffled, and the block diagonal structure is not immediately apparent, the sparsity of the $k$ null-vectors will point to the nodes that belong to a component.
- This fact will be exploited to find *graph clusters*, i.e. approximately disconnected groups of nodes

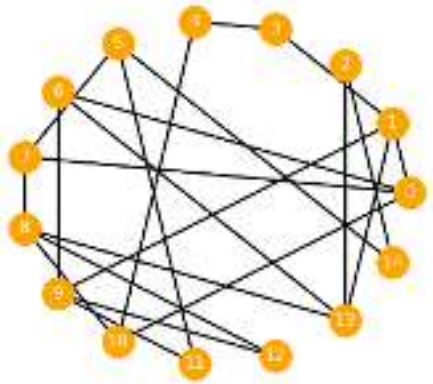
# Laplacian matrix and connectivity

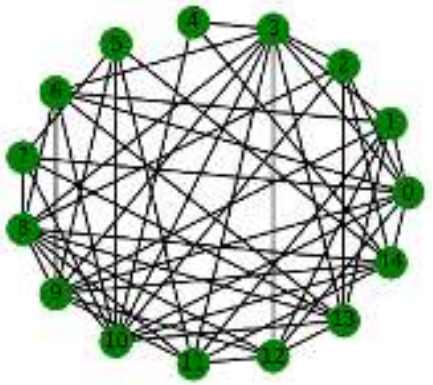
Cornell University

- Even if a graph is connected it may be sparse. Can we characterize the connectivity?
- Algebraic connectivity (a.k.a. Fiedler's eigenvalue): A measure of how well connected is a graph is the second smallest eigenvalue $\lambda_{N-1}(\boldsymbol{L})$.

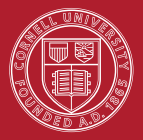




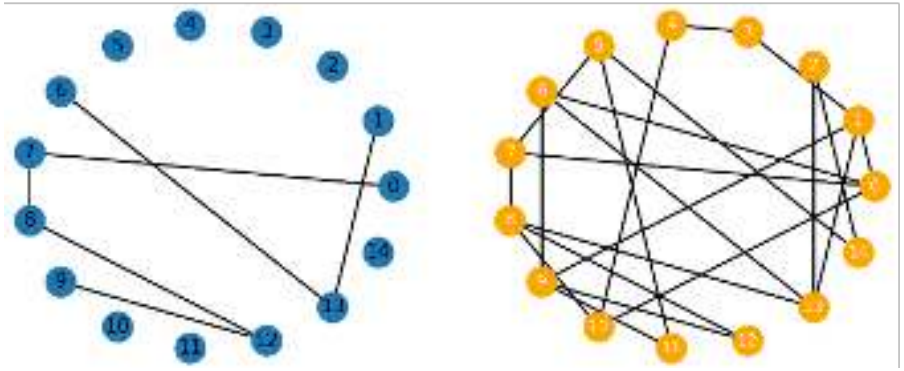


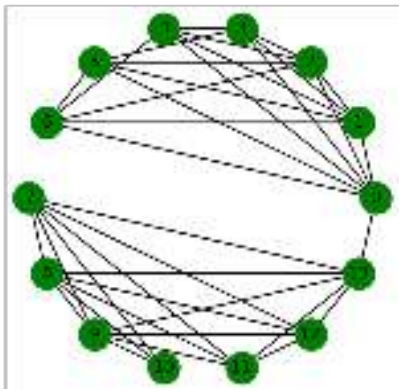
# What does well connected mean?




Cornell University

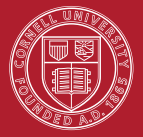
- There really are two kinds of situation where $\lambda_{N-1}(\boldsymbol{L})$ is small. Networks that:
  1. are pathologically sparse (cycle, path, tree networks)
  2. those that feature some dense but loosely connected clusters.





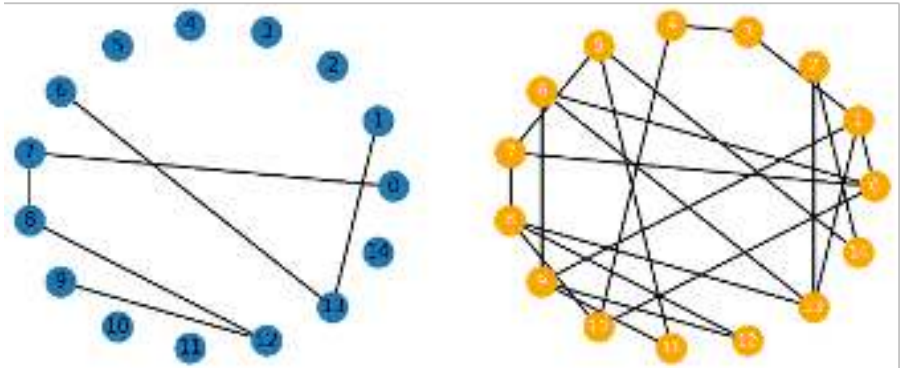
- On the left and middle two sparse graphs, on the right a graph with two dense but loosely connected components

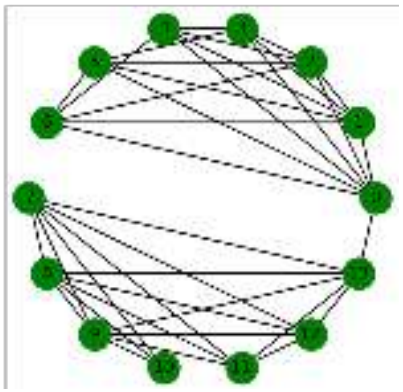
# What does well connected mean?




Cornell University

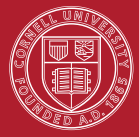
- There really are two kinds of situation where $\lambda_{N-1}(\boldsymbol{L})$ is small. Networks that:
  1. are pathologically sparse (cycle, path, tree networks)
  2. those that feature some dense but loosely connected clusters.





- On the left and middle two sparse graphs, on the right a graph with two dense but loosely connected components

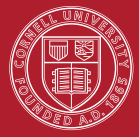
# What does sparse mean?





- A sparse array is one that has very few non zero elements compared to its size.
- Let $\|a\|_0$ be the zero norm of the array $w$: i.e. the number of non zero elements of $a$, also called its **support**.
- When $\|a\|_0$ is small relative to the dimension of $a$, the vector is said to be **sparse**.
- If the graph density $\delta = 2|\mathcal{E}| \ll N(N-1) \ll 1$ then we call the graph sparse.
- For a sparse network $\|\boldsymbol{A}\|_0 \ll |\mathcal{V}|^2$. That is not necessarily true when there are dense clusters that are loosely connected.

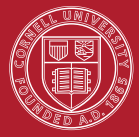
# What does sparse mean?





- A sparse array is one that has very few non zero elements compared to its size.
- Let $\|\boldsymbol{a}\|_0$ be the zero norm of the array $\boldsymbol{w}$: i.e. the number of non zero elements of $\boldsymbol{a}$, also called its **support**.
- When $\|\boldsymbol{a}\|_0$ is small relative to the dimension of $\boldsymbol{a}$, the vector is said to be **sparse**.
- If the graph density $\delta = 2|\mathcal{E}| \ll N(N-1) \ll 1$ then we call the graph sparse.
- For a sparse network $\|\boldsymbol{A}\|_0 \ll |\mathcal{V}|^2$. That is not necessarily true when there are dense clusters that are loosely connected.

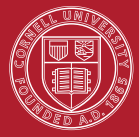
# What does sparse mean?





- A sparse array is one that has very few non zero elements compared to its size.
- Let $\|\boldsymbol{a}\|_0$ be the zero norm of the array $\boldsymbol{w}$: i.e. the number of non zero elements of $\boldsymbol{a}$, also called its **support**.
- When $\|\boldsymbol{a}\|_0$ is small relative to the dimension of $\boldsymbol{a}$, the vector is said to be **sparse**.
- If the graph density $\delta = 2|\mathcal{E}| \ll N(N-1) \ll 1$ then we call the graph sparse.
- For a sparse network $\|\boldsymbol{A}\|_0 \ll |\mathcal{V}|^2$. That is not necessarily true when there are dense clusters that are loosely connected.

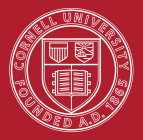
# What does sparse mean?





- A sparse array is one that has very few non zero elements compared to its size.
- Let $\|\boldsymbol{a}\|_0$ be the zero norm of the array $\boldsymbol{w}$: i.e. the number of non zero elements of $\boldsymbol{a}$, also called its **support**.
- When $\|\boldsymbol{a}\|_0$ is small relative to the dimension of $\boldsymbol{a}$, the vector is said to be **sparse**.
- If the graph density $\delta = 2|\mathcal{E}| \ll N(N-1) \ll 1$ then we call the graph sparse.
- For a sparse network $\|\boldsymbol{A}\|_0 \ll |\mathcal{V}|^2$. That is not necessarily true when there are dense clusters that are loosely connected.

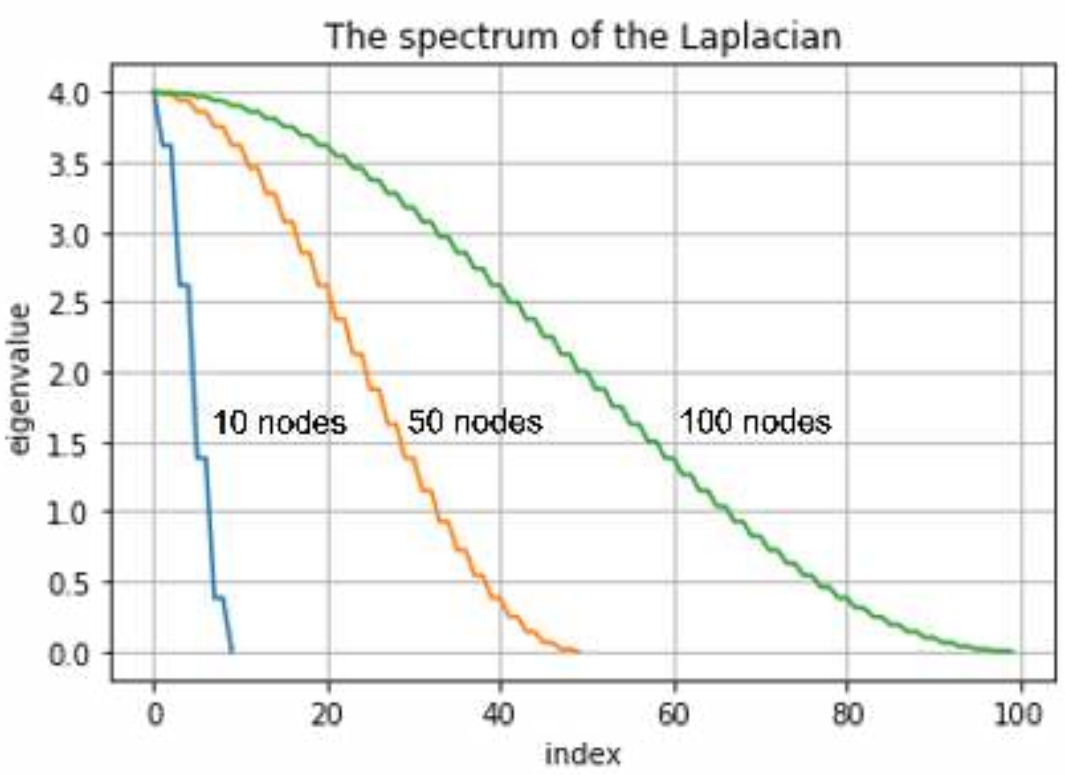
# Example of sparse network: cycle graph



Cornell University

- For the cycle graph Laplacian of the spectrum rapidly drops with its size.
- E.g., for a cycle graph with $N = |\mathcal{V}|$ nodes.

$$\boldsymbol{L} = \begin{bmatrix} 2 & -1 & 0 & 0 & -1 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ -1 & 0 & 0 & -1 & 2 \end{bmatrix}$$




- The eigenvectors and eigenvalues of $\boldsymbol{L}$ are known. They are, for $k = 0, \ldots, N-1$:

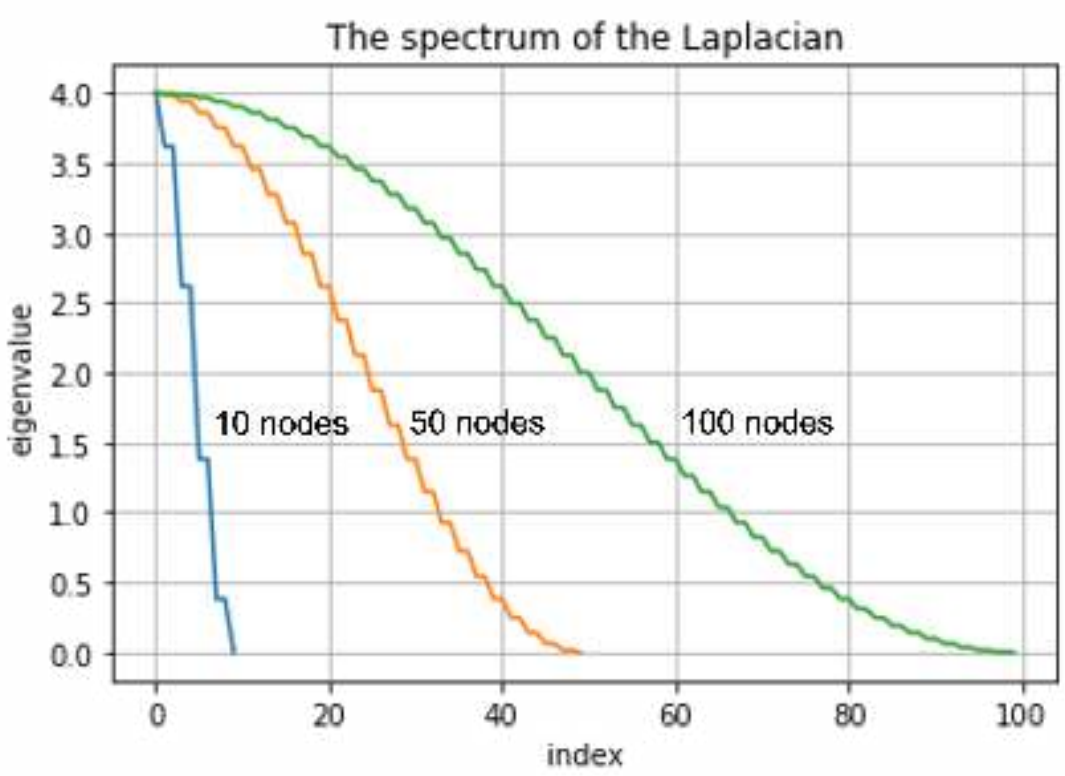
$$\boldsymbol{u}_k = (1, x_k, \ldots, x_k^{N-1})^\top, \quad x_k = \frac{1}{\sqrt{N}} e^{-j\frac{2\pi k}{N}}, \quad \lambda_k = 2(1-\cos(2\pi k/N))$$

# Example of sparse network: cycle graph



- For the cycle graph Laplacian of the spectrum rapidly drops with its size.
- E.g., for a cycle graph with $N = |\mathcal{V}|$ nodes.

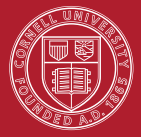
$$\boldsymbol{L} = \begin{bmatrix} 2 & -1 & 0 & 0 & -1 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ -1 & 0 & 0 & -1 & 2 \end{bmatrix}$$




- The eigenvectors and eigenvalues of $\boldsymbol{L}$ are known. They are, for $k = 0, \ldots, N-1$:

$$\boldsymbol{u}_k = (1, x_k, \ldots, x_k^{N-1})^\top, \quad x_k = \frac{1}{\sqrt{N}} e^{-j\frac{2\pi k}{N}}, \quad \lambda_k = 2(1-\cos(2\pi k/N))$$

# Graphs with circular symmetry





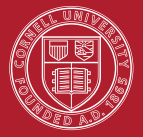
- Note that $\boldsymbol{u}_k, k = 1, \ldots, N$ is the Discrete Fourier Transform (DFT) basis. Not just the cycle graph, but any Laplacian that has a Toeplitz circulant structure has this eigenvector basis.
- That is, the $\boldsymbol{L}$ for such graphs have the structure:

$$\boldsymbol{L} = \begin{bmatrix} L(0) & L(1) & L(2) & \ldots & L(N-1) \\ L(1) & L(0) & \ddots & \ddots & L(N-2) \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ L(N-1) & \ddots & \ddots & \ddots & L(0) \end{bmatrix}$$

- Observe that $\boldsymbol{L}$ is defined by $N$ values $L(0), \ldots, L(N-1)$ so that:

$$[\boldsymbol{L}]_{uv} = L(u - v \pmod N), \quad u = 1, \ldots, N, v = 1, \ldots, N$$

# Graphs with circular symmetry





- Note that $\boldsymbol{u}_k, k = 1, \ldots, N$ is the Discrete Fourier Transform (DFT) basis. Not just the cycle graph, but any Laplacian that has a Toeplitz circulant structure has this eigenvector basis.
- That is, the $\boldsymbol{L}$ for such graphs have the structure:

$$\boldsymbol{L} = \begin{bmatrix} L(0) & L(1) & L(2) & \ldots & L(N-1) \\ L(1) & L(0) & \ddots & \ddots & L(N-2) \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ L(N-1) & \ddots & \ddots & \ddots & L(0) \end{bmatrix}$$

- Observe that $\boldsymbol{L}$ is defined by $N$ values $L(0), \ldots, L(N-1)$ so that:

$$[\boldsymbol{L}]_{uv} = L(u - v \pmod N), \quad u = 1, \ldots, N, v = 1, \ldots, N$$

# Circular matrices and circular convolutions



- Note that a circulant matrix $\boldsymbol{A}$, with $[\boldsymbol{A}]_{uv} = A(u - v \,(\text{mod } N))$, which is entirely defined by the $N$ values $A(\ell),\ \ell = 0, \dots, N-1$
- The graphs we are discussing have an adjacency matrix $\boldsymbol{A}$ that is also circulant, not just $\boldsymbol{L}$
- The multiplication of a vector $\boldsymbol{x} \in \mathbb{R}^N$ and a circulant matrix $\boldsymbol{A}$ is equivalent to what in digital signal processing (DSP) is called a circular convolution.

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}, \quad \Leftrightarrow \quad \overbrace{y(t)}^{y_{t+1}} = \sum_{\ell=0}^{N-1} A(t - \ell \ (\text{mod } N)) \overbrace{x(\ell)}^{x_{\ell+1}}$$

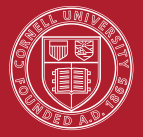
# Circular matrices and circular convolutions



- Note that a circulant matrix $\boldsymbol{A}$, with $[\boldsymbol{A}]_{uv} = A(u - v \,(\text{mod } N))$, which is entirely defined by the $N$ values $A(\ell),\ \ell = 0, \ldots, N-1$
- The graphs we are discussing have an adjacency matrix $\boldsymbol{A}$ that is also circulant, not just $\boldsymbol{L}$
- The multiplication of a vector $\boldsymbol{x} \in \mathbb{R}^N$ and a circulant matrix $\boldsymbol{A}$ is equivalent to what in digital signal processing (DSP) is called a circular convolution.

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}, \quad \Leftrightarrow \quad \overbrace{y(t)}^{y_{t+1}} = \sum_{\ell=0}^{N-1} A(t - \ell \ (\text{mod } N)) \overbrace{x(\ell)}^{x_{\ell+1}}$$

# Graphs with circular symmetry



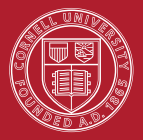


- Let $\ell = u - v + 1$. Next we show that:

$$\boldsymbol{L}\boldsymbol{u}_k = \lambda_k \boldsymbol{u}_k, \quad \text{where} \quad \lambda_k = \overbrace{\tilde{L}(e^{j\omega_k})}^{\text{DFT of } L(\ell)} = \sum_{\ell=0}^{N-1} L(\ell) e^{-\frac{j2\pi k\ell}{N}}$$

- The eigenvalues of the Laplacian for this type of graphs that have circular symmetry are the DFT of $L(\ell)$, i.e. $\lambda_k = \tilde{L}(e^{j\omega_k})$, for $\omega_k = 2\pi k/N$

# Graphs with circular symmetry





- Let us ignore the normalization by $\frac{1}{\sqrt{N}}$

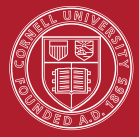
$$[\boldsymbol{L}\boldsymbol{u}_k]_u = \sum_{v=1}^{N} [\boldsymbol{L}]_{uv} [\boldsymbol{u}_k]_v = \sum_{v=1}^{N} [\boldsymbol{L}]_{uv} e^{-j\frac{2\pi k(v-1)}{N}}$$

$$\overset{(a)}{=} \sum_{v'=0}^{N-1} L(u'-v') \overbrace{e^{-j\frac{2\pi k(v'-u'+u')}{N}}}^{e^{-j\frac{2\pi kv'}{N}}} = e^{-j\frac{2\pi ku'}{N}} \sum_{v'=0}^{N-1} L(u'-v') e^{-j\frac{2\pi k(v'-u')}{N}}$$

$$\overset{(b)}{=} e^{-j\frac{2\pi ku}{N}} \overbrace{\sum_{\ell=0}^{N-1} L(\ell) e^{-j\frac{2\pi k\ell}{N}}}^{\text{DFT}} = [\boldsymbol{u}_k]_u \lambda_k$$

Where (a) is the substitution $u' = u - 1, v' = v - 1$ and (b) is $\ell = u - v = u' - v'$

# Outline



Cornell University

# Concluding remarks



- We talked about network flows in relation to the incidence matrix
- We introduced the Laplacian matrix
- We talked about Spectral Graph Theory and the key graph feature called *algebraic connectivity* or Fiedler eigenvalue
- Next time we will leverage this insight to search for clusters in a network
- We will also talk about generalization of the Laplacian.

# Concluding remarks



- We talked about network flows in relation to the incidence matrix
- We introduced the Laplacian matrix
- We talked about Spectral Graph Theory and the key graph feature called *algebraic connectivity* or Fiedler eigenvalue
- Next time we will leverage this insight to search for clusters in a network
- We will also talk about generalization of the Laplacian.