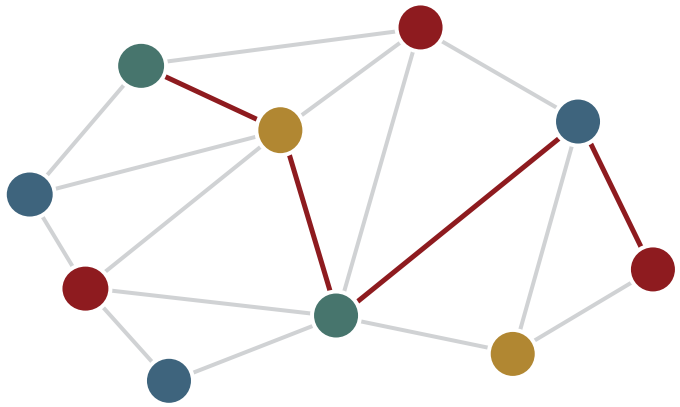
Foundations of Graph Signal Processing and Graph Machine Learning



August 27

Zhiyang Wang · Electrical & Systems Engineering · WashU

Class 2 · Intro to graph signal processing

Office hours: Thus 2:30 - 4 PM from next week McKelvey Hall 2052

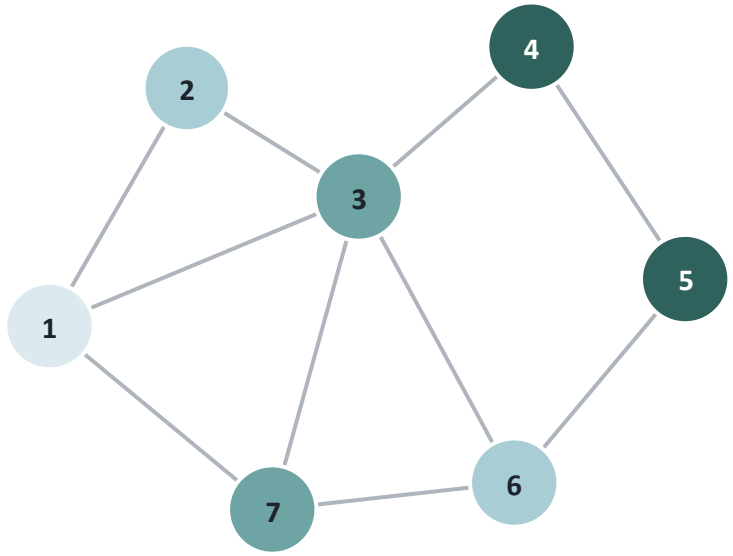
Paper presentation & Projects can be individual or in pair.
Sign-ups for paper presentation will be posted after the roster settles.

Three parts: from local diffusion to graph frequency

01	**GRAPH DIFFUSION**	Define a graph shift operator; use adjacency or Laplacian for local updates; repeated shifts lead to multi-hop diffusions
02	**GRAPH LAPLACIAN**	Measure local mismatch at one node vs its connected nodes; total variation across weighted edges.
03	**GRAPH FOURIER TRANSFORM**	Use Laplacian eigenvectors as oscillation modes; eigenvalues represent edge variation levels; compare with DCT and DFT

Core model: a graph is a triplet $G = (V, E, W)$

Undirected graph

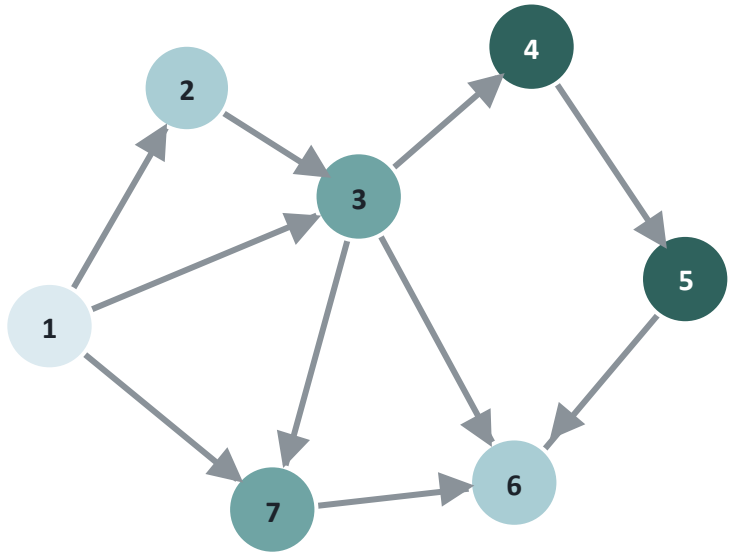


V	vertices: sensors, locations, people, or other entities
E	edges: unordered pairs of related vertices
$w_{ij} = w_{ji}$	symmetric nonnegative strength between vertices i and j
x_i	signal value at vertex i

The graph defines which vertices interact and how strongly.

Core model: a graph is a triplet $G = (V, E, W)$

Directed graph

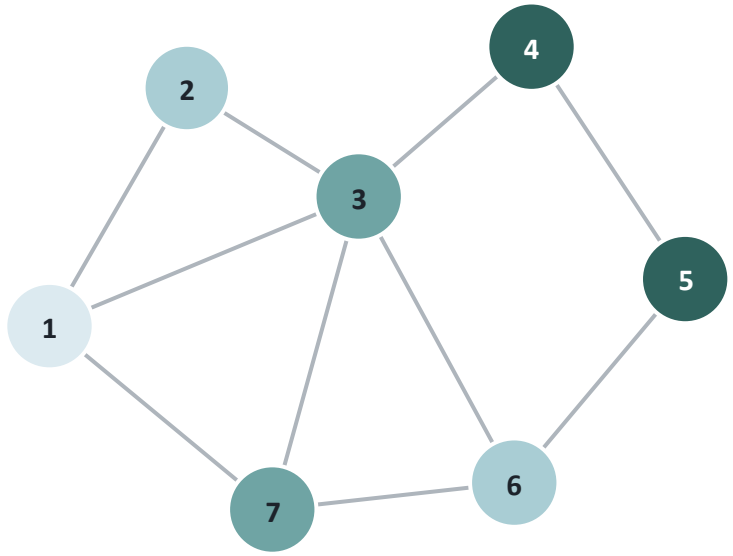


V	vertices
E	edges: ordered pairs of related vertices
$w_{ij} \neq w_{ji}$	nonnegative strength between vertices i and j
x_i	signal value at vertex i

Direction is a separate modeling and can carry important information.

Core model: a graph is a triplet $G = (V, E, W)$

Unweighted graph

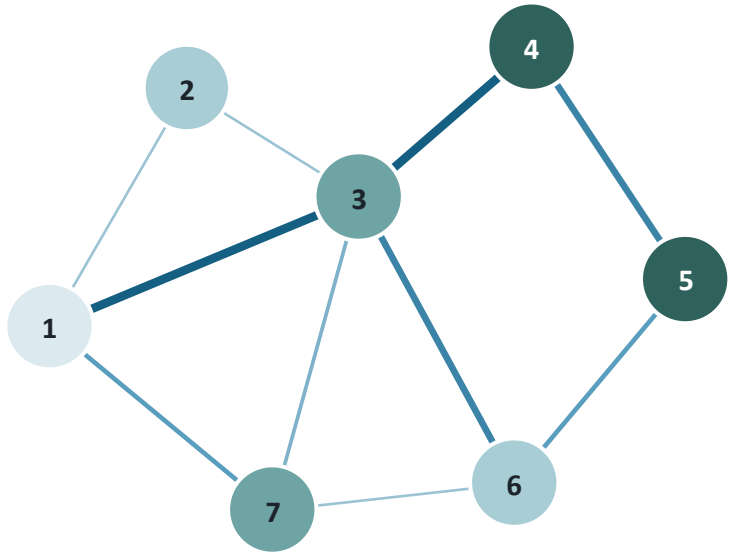


V	vertices
E	edges: unordered pairs of related vertices
$w_{ij} = 0$ or 1	e.g. citation networks, friendship networks…
x_i	signal value at vertex i

Edges represent absence or presence. Independent modeling choice with directions.

Core model: a graph is a triplet $G = (V, E, W)$

Weighted graph

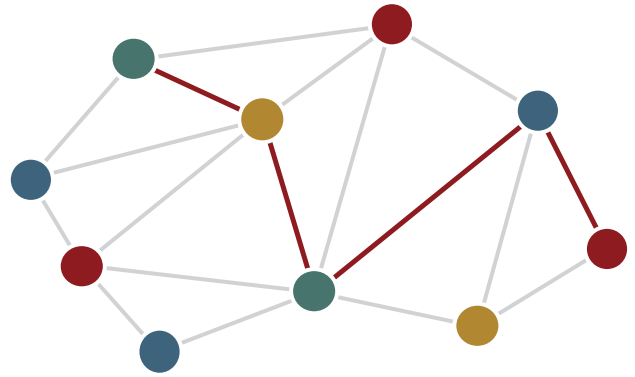


V	vertices
E	edges: unordered pairs of related vertices
$w_{ij} \in \mathbb{R}$	e.g. road networks, user networks …
x_i	signal value at vertex i

Edges carry numerical strengths.

PART I

Graph diffusion uses local graph shifts.



01 Define the graph shift operator

02 Choose adjacency or Laplacian

03 Apply one local update

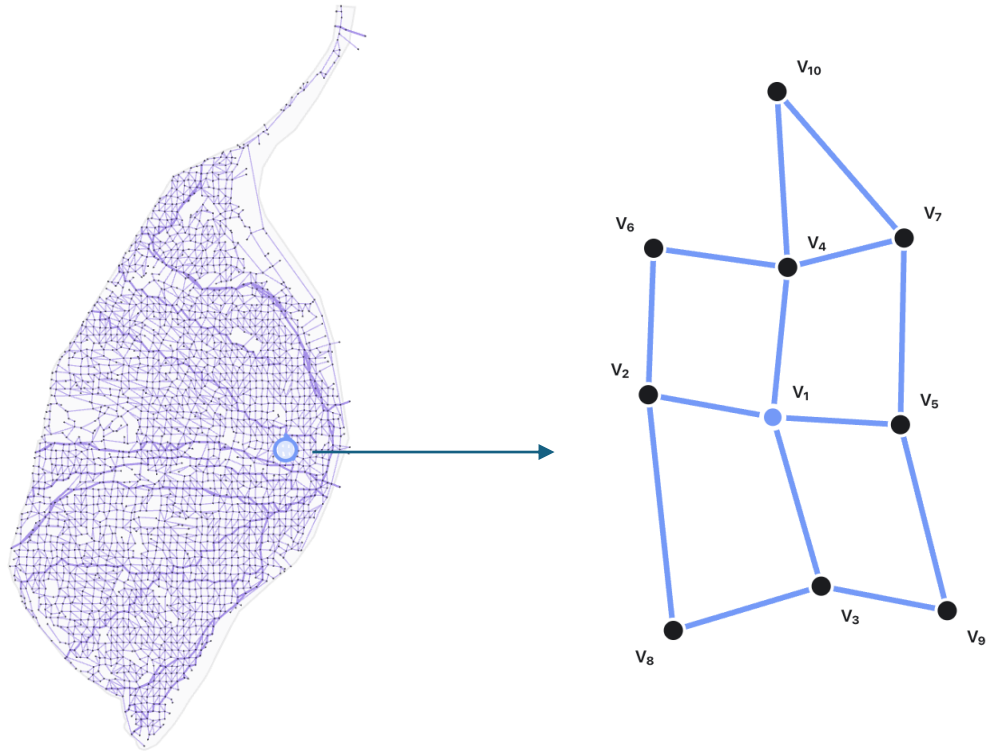
04 Combine several hop ranges

Repeated local shifts build global diffusion.

Graph signals and Diffusion process

- **Graph signal**: values supported over graph nodes $\boldsymbol{x} \in \mathbb{R}^n$

  How do graph signal $\boldsymbol{x}$ interact with graph $\boldsymbol{G}$? ---- Through **diffusion process**

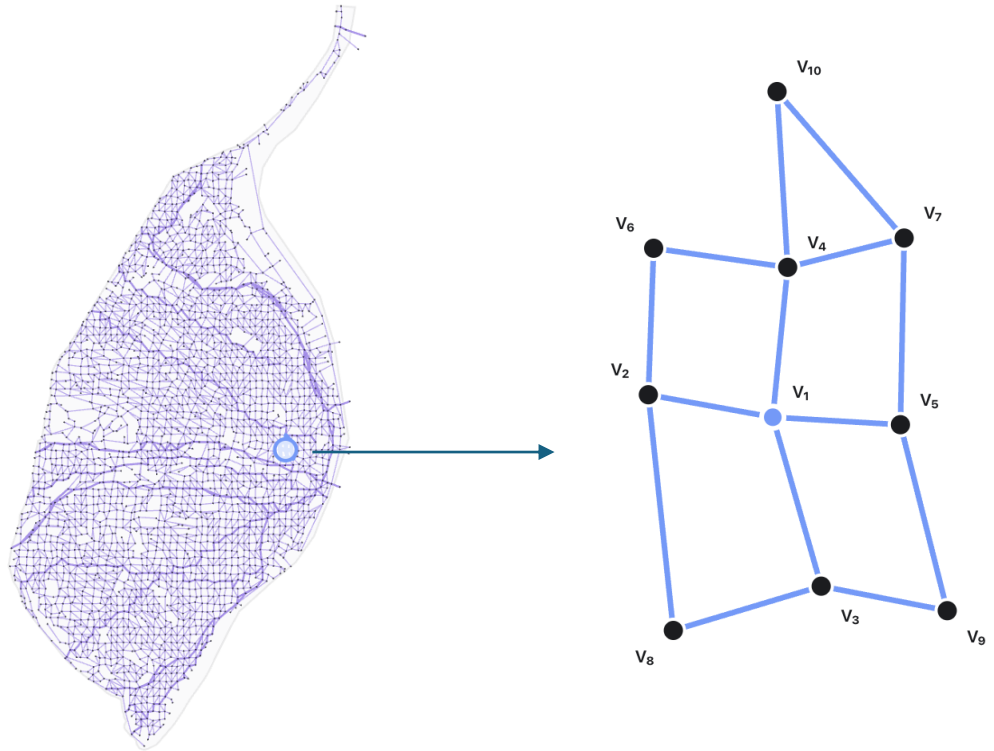


- $\boldsymbol{G}$ as directed and weighted transportation network
- Graph signal $\boldsymbol{x}$ as traffic volume at each node
- $\sum_j w_{ij} = 1$ for each node i – probability of traffic moving from i to j
- Given $\boldsymbol{x}(t)$, how to predict $\boldsymbol{x}(t+1)$
- Simplified model and fit for other similar problems

Graph signals and Diffusion process

- **Graph signal**: values supported over graph nodes $\boldsymbol{x} \in \mathbb{R}^n$

  How do graph signal $\boldsymbol{x}$ interact with graph $\boldsymbol{G}$? ---- Through **diffusion process**



- Define $\mathcal{N}_i^{in} = \{j \in V, (j,i) \in E\}$ and

  $\mathcal{N}_i^{out} = \{j \in V, (i,j) \in E\}$ directed neighbor set
- Estimation of traffic at node i at time $t+1$:

$$\hat{x}_i(t+1) = x_i(t) + \sum_{j \in \mathcal{N}_i^{in}} w_{ji}\, x_j(t) - \sum_{j \in \mathcal{N}_i^{out}} w_{ij}\, x_i(t)$$

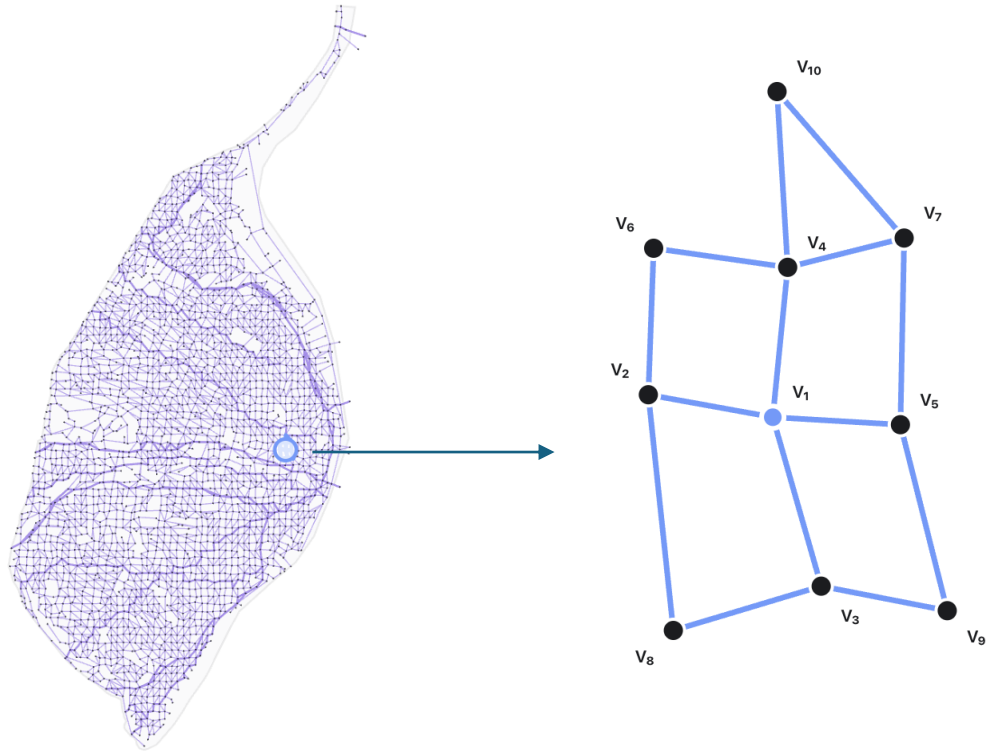
- Diffusion as local processes on each node

Graph signals and Diffusion process

- **Graph signal**: values supported over graph nodes $\boldsymbol{x} \in \mathbb{R}^n$

  How do graph signal $\boldsymbol{x}$ interact with graph $\boldsymbol{G}$? ---- Through **diffusion process**

Graph Shift Operator



- Consider network adjacency matrix

$$[\boldsymbol{A}]_{ij} = \begin{cases} w_{ji} & \text{if } (j,i) \in E \\ 0 & \text{otherwise} \end{cases}$$

- Estimation of traffic at time t+1:

$$\hat{\boldsymbol{x}}(t+1) = \boldsymbol{A}\boldsymbol{x}(t)$$

$$[\boldsymbol{A}\boldsymbol{x}(t)]_i = \sum_j [\boldsymbol{A}]_{ij}\,[\boldsymbol{x}(t)]_j = \sum_{j \in \mathcal{N}_i^{in}} w_{ji}\, x_j(t)$$

- Diffusion as global process with adjacency matrix

Graph shift operators: adjacency or Laplacian

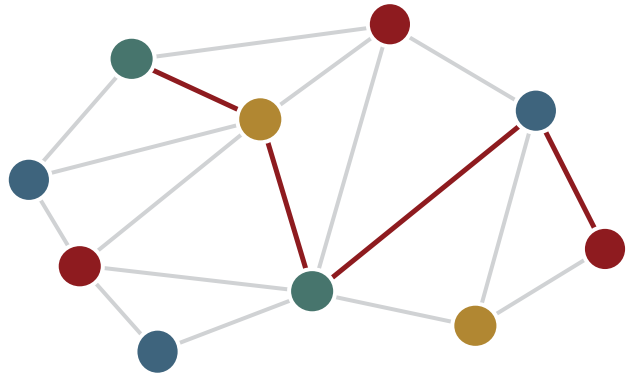
GRAPH SHIFT OPERATOR	GENERAL TASK	LOCAL ACTION	GRAPH LOCALITY
$\boldsymbol{z = Sx}$	**Define one graph shift**	one graph-local update	$[\boldsymbol{S}]_{ij} \neq 0$ if $(i,j) \in \boldsymbol{E}$
$\boldsymbol{A}$	**Aggregate connected values**	weighted neighbor sum	one-hop neighbor mixing
$\boldsymbol{L = D - A}$	**Measure local variation**	node value minus neighbors	one-hop graph difference

$D = diag(A\ 1)$

Adjacency aggregates neighboring values. Laplacian measures their mismatch.

PART II

The graph Laplacian measures change across edges.



01 Read the Laplacian as a graph difference

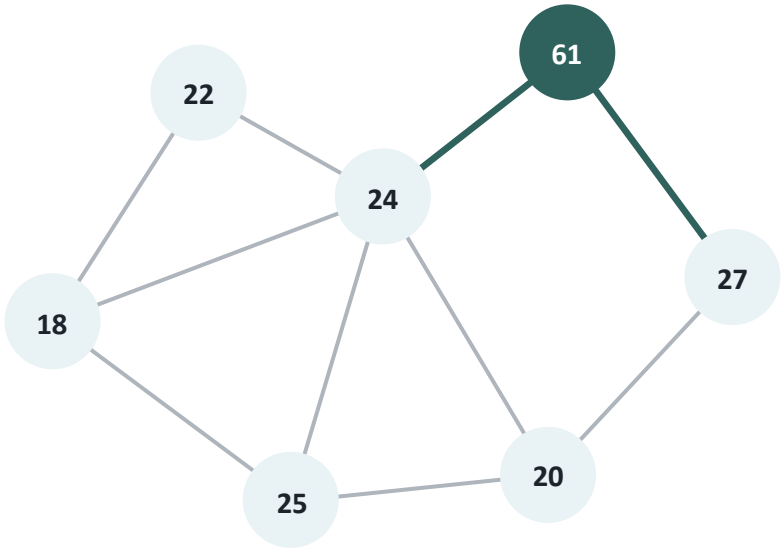
02 Recall DSP total variation on a sample chain

03 Generalize the definition to weighted graph edges

04 Use total variation to order graph-frequency modes

Total variation connects graph differences to graph frequency.

The graph Laplacian measures local mismatch



$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$$

$$(\boldsymbol{L}\boldsymbol{x})_i = \sum_{j \in \mathcal{N}_i} w_{ij}\left(x_i - x_j\right)$$

Each node compares its own value with each neighbor's — ***Local variation detector***

One step of smoothing: $\boldsymbol{x} \leftarrow (\boldsymbol{I} - \epsilon \boldsymbol{L})\boldsymbol{x}$

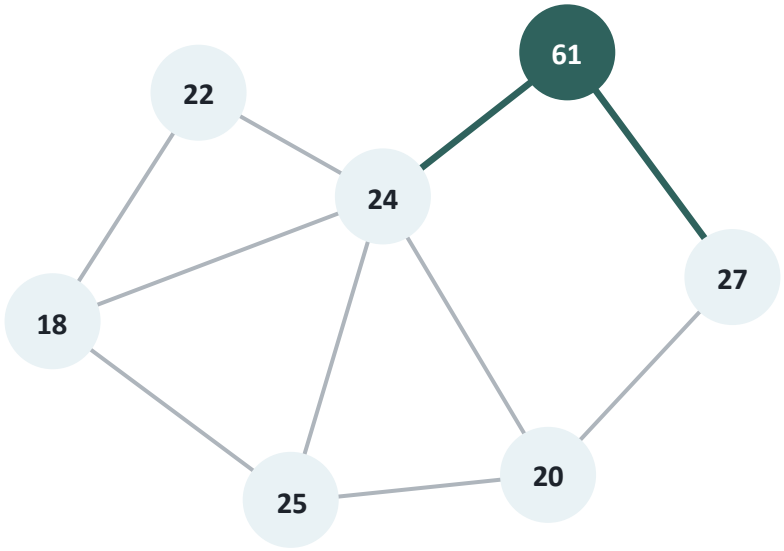
$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$ sums weighted differences

L is a local variation detector

- Small where the signal is flat, large at the jump.
- Constant signal: $L\mathbf{1} = 0$ exactly — no disagreement anywhere.
- Alternating signal: large output at every node.

Lx = per-node disagreement with the neighborhood.

Total variation characterizes global smoothness



Local variation can be detected as a **vector**

*How to measure **global variation**?*

$$TV(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{L}\boldsymbol{x} = \sum_{(i,j)\in \boldsymbol{E}} w_{ij}\left(x_i - x_j\right)^2$$

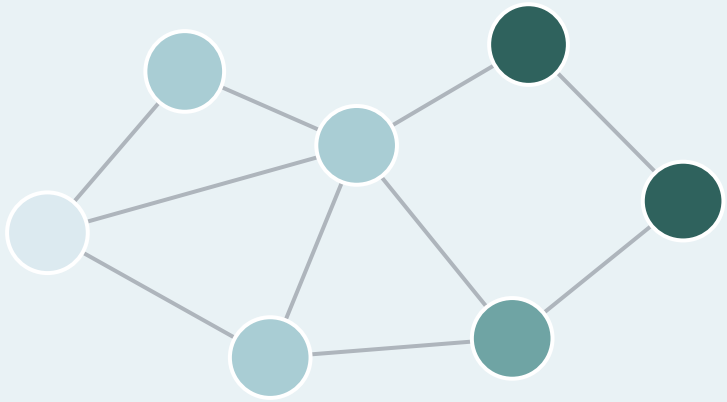
$$\nabla\left(\frac{1}{2}\boldsymbol{x}^T \boldsymbol{L}\boldsymbol{x}\right) = \boldsymbol{L}\boldsymbol{x}$$ leads to the smoothing step $$\boldsymbol{x} \leftarrow (\boldsymbol{I} - \epsilon \boldsymbol{L})\boldsymbol{x}$$

TV also seen as Dirichlet Energy or Laplacian quadratic form.

You have seen $x^{\mathrm{T}}Lx$ before: Laplacian regularization.

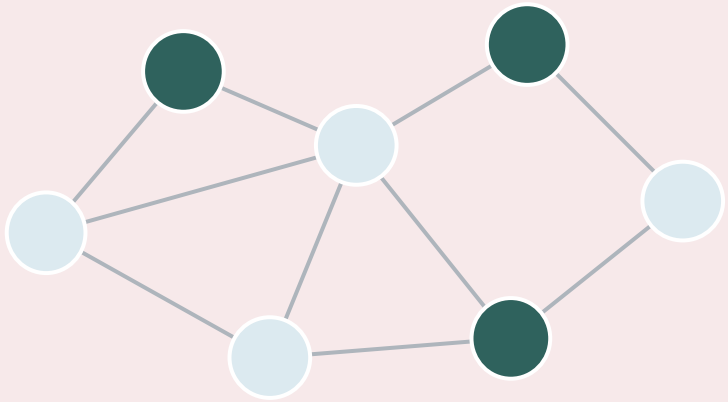
The same graph can carry low- or high-variation signals

LOW TOTAL VARIATION -- homophily



gradual color changes across most edges

HIGH TOTAL VARIATION -- heterophily

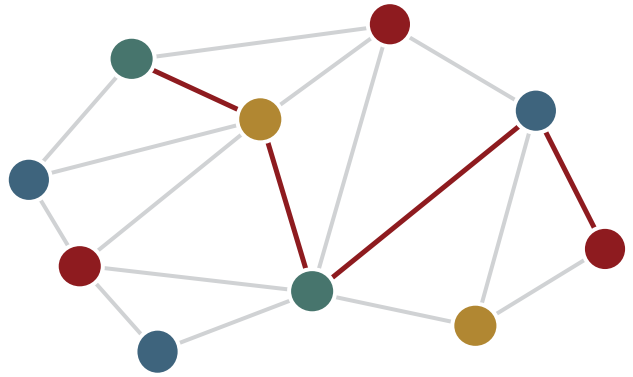


a few dark–light edges create large changes

Graph energy tells us about compatibility with the connectivity.
The scientific task tells us whether to preserve or suppress it.

PART III

Graph frequency comes from total variation.



01 Normalize each Laplacian eigenvector

02 Its total variation equals its eigenvalue

03 Order modes from low to high variation

04 Project graph signals onto these modes

· Small eigenvalue: smooth mode
· Large eigenvalues: rapidly varying mode

What are the lowest and highest TV(x)?

$TV(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{L}\boldsymbol{x}$ Assume $\|\boldsymbol{x}\| = 1$

Rayleigh quotient
--variation per unit energy $\dfrac{\boldsymbol{x}^T \boldsymbol{L}\boldsymbol{x}}{\boldsymbol{x}^T \boldsymbol{x}}$

- **The easy extreme: the constant signal**

$$\boldsymbol{u}_1 = \frac{1}{\sqrt{n}}\mathbf{1}, \qquad TV(\boldsymbol{u}_1) = 0$$

It is already secretly an eigenpair: $\mathbf{L1} = 0 = 0 \cdot \mathbf{1}$

Multiplicity of 0 means the number of connected components

Minimizing TV forces the eigen-equation

- Smoothest signal that isn't constant:

$$\min_{\boldsymbol{x}} \boldsymbol{x}^\top \boldsymbol{L}\boldsymbol{x} \qquad s.t. \|\boldsymbol{x}\| = 1, \boldsymbol{x} \perp \mathbf{1}$$

- Trade the constraint for a multiplier — the Lagrangian: $\mathcal{L}(x;\lambda) = \boldsymbol{x}^\top \boldsymbol{L}\boldsymbol{x} - \lambda(\boldsymbol{x}^\top \boldsymbol{x} - \mathbf{1})$
- Stationarity — set the gradient to zero: $\nabla_{\mathbf{x}}\mathcal{L} = 2\boldsymbol{L}\boldsymbol{x} - 2\lambda\boldsymbol{x} = 0 \Rightarrow \boldsymbol{L}\boldsymbol{x} = \lambda\boldsymbol{x}$
- Plug $\|\boldsymbol{x}\| = 1$ back in — multiply by $\boldsymbol{x}^{\mathrm{T}}$:

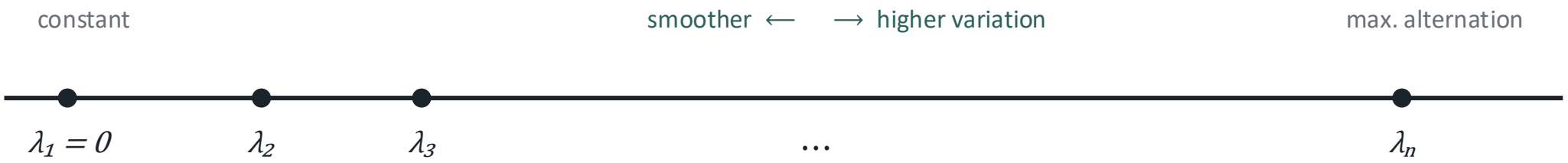
$$\boldsymbol{L}\boldsymbol{x} = \lambda\boldsymbol{x} \qquad and \qquad \lambda = \boldsymbol{x}^\top \boldsymbol{L}\boldsymbol{x} = TV(\boldsymbol{x})$$

The eigenvalue is the smoothness score of its eigenvector.

Eigenpairs are answers to the contest.

The smoothness ruler

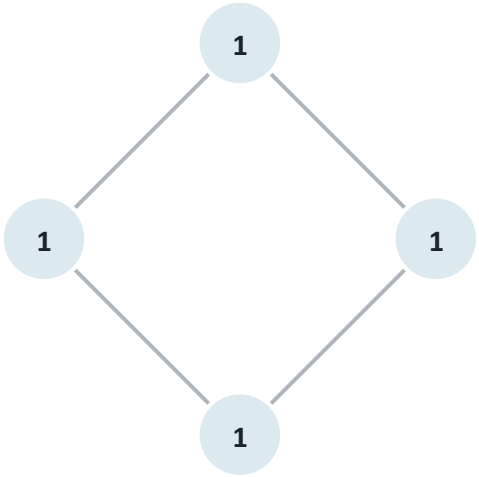
- Repeat “smoothest unit signal orthogonal to everything so far”: $\boldsymbol{u}_2, \boldsymbol{u}_3, \ldots$
- Maximize instead of minimize: $\boldsymbol{u}_{\mathrm{n}}$, the largest variational unit signal.



The ordered spectrum $0 = \lambda_1 \leq \cdots \leq \lambda_{\mathrm{n}}$ of graph Laplacian is a smoothness axis --- ordered graph frequency.

Laplacian eigenvectors are graph oscillation modes

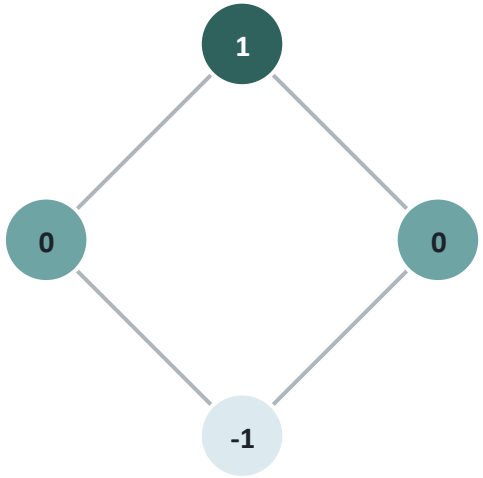
u_1



$\lambda_1 = 0$

same level everywhere

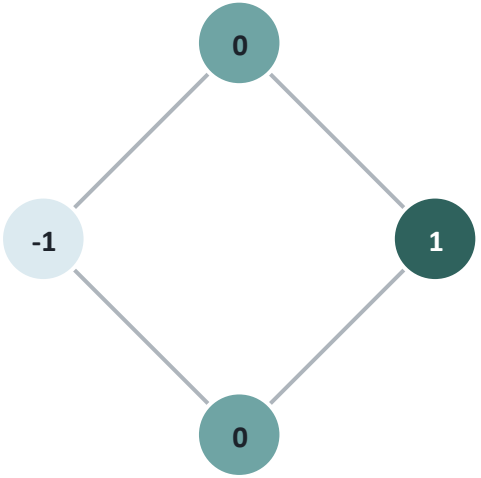
u_2



$\lambda_2 = 2$

one graph partition differs

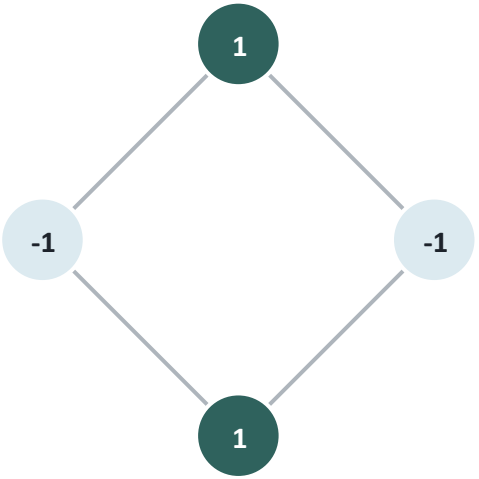
u_3



$\lambda_3 = 2$

another partition differs

u_4



$\lambda_4 = 4$

values alternate across edges

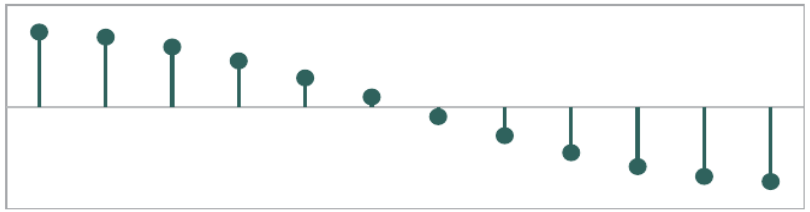
$$\boldsymbol{L}\boldsymbol{u}_k = \lambda_k \boldsymbol{u}_k$$

Small λ_k -- smooth; large λ_k-- rapid variation

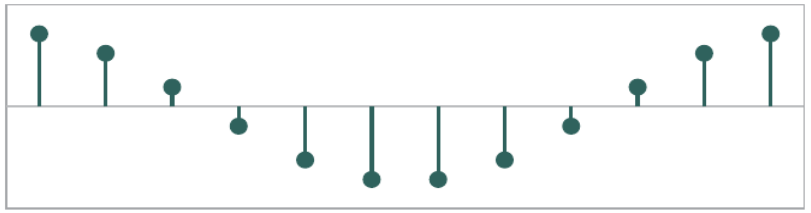
Eigenvectors look like sampled sinusoids

v_1: $\lambda \approx 0$, 0 sign changes

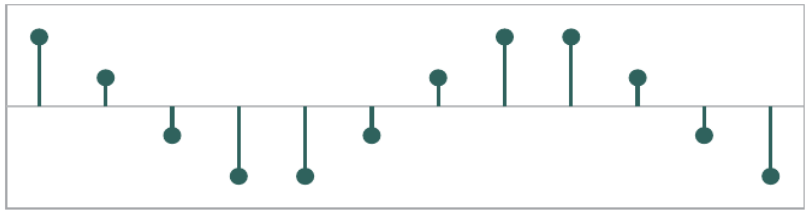
v_2: $\lambda \approx 0.07$, 1 sign changes



v_3: $\lambda \approx 0.27$, 2 sign changes



v_4: $\lambda \approx 0.59$, 3 sign changes



Eigenvalue order = sign-change count = oscillation. (Path: DCT basis · cycle: DFT basis.)

A graph signal is a mixture of oscillation modes

$\boldsymbol{L}$ is symmetric -- Eigendecomposition: $\boldsymbol{L = U\Lambda U^T}$ **with** $\boldsymbol{U^T U = I}$

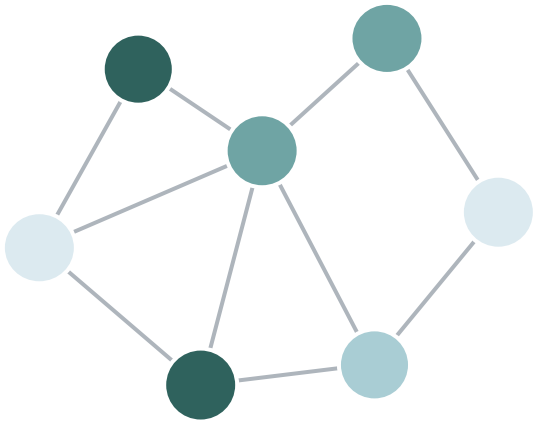
- Each direction carries its smoothness score λ_k from the ruler
- For the path graph these are exactly the DCT vectors — classical Fourier is the special case

Graph Fourier Transform (GFT) – mapping graph signals onto the eigenbasis

$$\hat{x}_k = \boldsymbol{u}_k^T \boldsymbol{x}, \quad \hat{\boldsymbol{x}} = \boldsymbol{U}^T \boldsymbol{x} \qquad \boldsymbol{x} = \boldsymbol{U}\hat{\boldsymbol{x}} = \sum_{k=1}^{n} \hat{x}_k \, \boldsymbol{u}_k$$

We can represent any graph signal x in this basis. This also holds to other GSOs.

The GFT measures the strength of each graph pattern

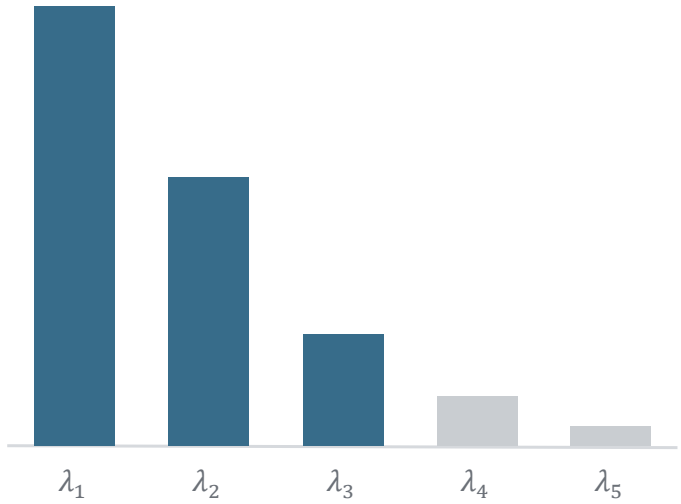


graph signal

Projection onto the eigenbasis

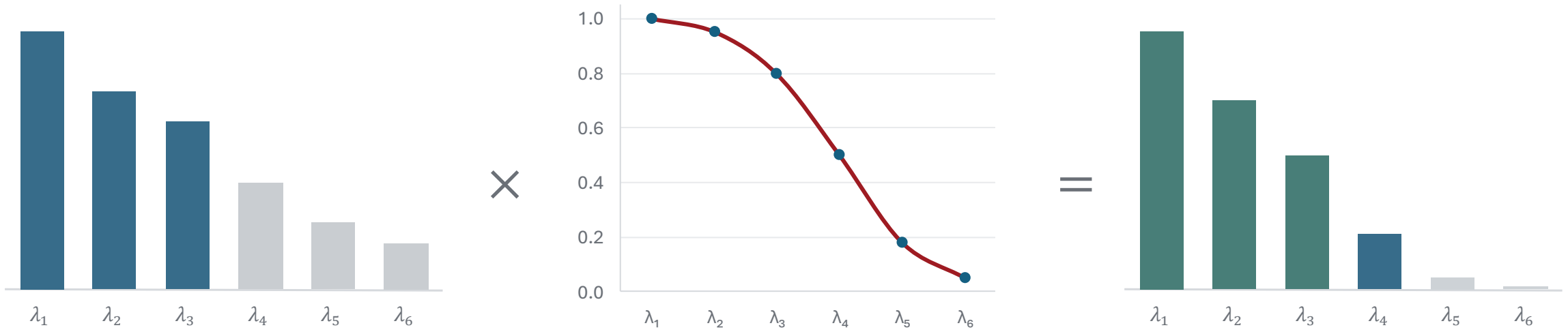
$$\hat{\boldsymbol{x}} = \boldsymbol{U}^{\boldsymbol{T}}\boldsymbol{x}$$

compare $\boldsymbol{x}$ with every Laplacian eigenvector



Low eigenvalue – smooth graph content
High eigenvalue– varied across edges content

A graph low-pass filter keeps smooth components



input GFT coefficients **spectral gains** **filtered coefficients**

Smooth modes × large gain; Rapidly varying modes × small gain

Depend on what is your task and goal

Filtering: one knob per frequency

Transform, reweight each frequency by $h(\lambda_k)$, transform back:

$$\boldsymbol{y} = h(\boldsymbol{L})\boldsymbol{x} \longrightarrow \boldsymbol{y} = \boldsymbol{U}h(\boldsymbol{\Lambda})\boldsymbol{U}^T\boldsymbol{x} = \sum_{k=1}^{n} h(\lambda_k)\hat{x}_k\boldsymbol{u}_k$$

- You already own a filter — smoothing from Part I is the low-pass $h(\lambda) = 1 - \varepsilon\lambda$.
- Low-pass denoises · high-pass finds boundaries · band-pass isolates scales.

Next class: h as a polynomial in $\boldsymbol{L}$ — filters that are local, cheap, and learnable. That road leads to graph neural networks.

The GFT turns "operate on a graph signal" into "choose a function of λ."

On a path, Laplacian eigenvectors are DCT modes

A finite sequence forms a path

no edge from sample 7 to sample 0

$$x[-1] = x[0] \qquad x[N] = x[N-1]$$

dashed nodes show the conceptual reflection

One DCT-II mode

$$c_k[n] = \alpha_k cos[\frac{\pi}{N}(n+\frac{1}{2})k] \qquad \alpha_0 = \frac{1}{\sqrt{N}} \qquad \alpha_k = \sqrt{\frac{2}{N}} \quad \text{for } k > 0$$

The basis is fully real; reflection replaces periodic wrap-around.

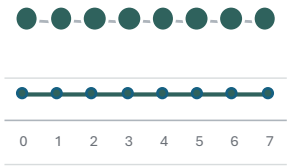
$$L_{P_N} c_k = \lambda_k c_k \qquad \lambda_k = 2 - 2cos(\frac{\pi k}{N})$$

$$a[k] = c_k^T x \qquad x = \sum_k a[k] c_k$$

Examples on an 8-node path

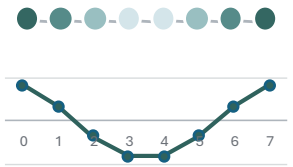
nodes show sign · plots show normalized $c_k[n]$ vs n

k = 0 · constant



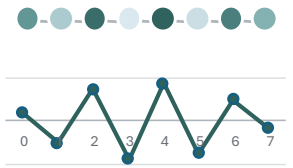
$\lambda_0 = 0$

k = 2 · two sign changes



$\lambda_2 = 2 - \sqrt{2} \approx 0.586$

k = 7 · almost alternating

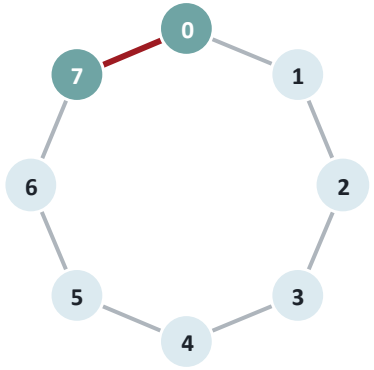


$\lambda_7 \approx 3.848$

DCT coefficients measure how much of each path mode appears in x.

On a cycle, Laplacian eigenvectors are DFT modes

Periodic samples form a cycle



sample 7 ↔ sample 0

$$x[n+N] = x[n]$$

$$(L_{C_N}x)[n] = 2x[n] - x[n-1] - x[n+1]$$

indices are taken modulo N

One DFT mode

$$f_k[n] = \frac{1}{\sqrt{N}} exp(j\frac{2\pi kn}{N}) \qquad j^2 = -1$$

Each edge advances the phase by 2πk/N.

$$L_{C_N} f_k = \lambda_k f_k$$

$$\lambda_k = 2 - 2cos(\frac{2\pi k}{N})$$

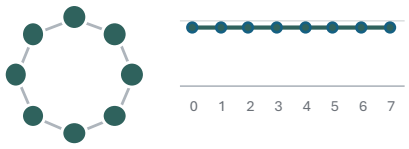
$$\hat{x}[k] = f_k^H x$$

$$x = \Sigma_k \hat{x}[k] f_k$$

Examples on an 8-node cycle

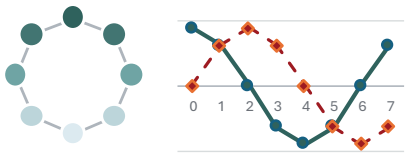
nodes show $Ref_k[n]$· plots vs n: teal Re, red Im

k = 0 · constant



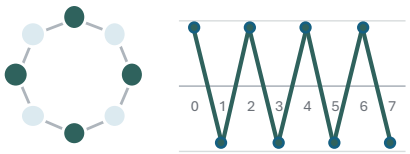
$\lambda_0 = 0$

k = 1 · one turn



$\lambda_1 = 2 - \sqrt{2} \approx 0.586$

k = 4 · alternating



$\lambda_4 = 4$

DFT coefficients measure how much of each cycle mode appears in x.

DFT and GFT use parallel concepts

IDEA	DFT / CLASSICAL DSP	GFT / GRAPH SIGNAL PROCESSING
Measurement	audio sample at time n	signal value at vertex i
Neighbor	previous / next time sample	vertex joined by an undirected edge
Shift	move one time step	apply a graph shift operator
Low frequency	slow change over time	small variation across strongly weighted edges
High frequency	rapid alternation in time	large variation across edges
Filter	smooth temporal noise	attenuate selected graph frequencies

GSP keeps the DSP workflow and replaces the regular grid with graph connectivity.

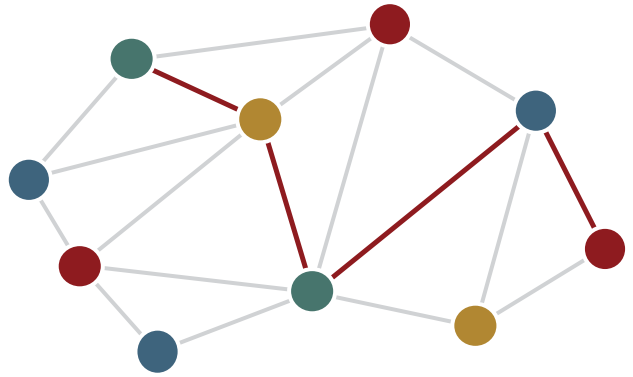
The graph defines locality and frequency

01	ORDER	sample 8 follows sample 7	vertex labels need not encode geometry
02	NEIGHBOR	adjacent sample index	connected by an edge
03	LOW FREQ.	slow change along time	signal changes slowly across edges
04	HIGH FREQ.	rapid temporal oscillation	signal changes rapidly across edges

Changing the graph changes the modes—even when the signal values stay the same.

TAKEAWAYS

Three ideas organize Graph Signal Processing.



01 Graph shifts use adjacency or Laplacian.

02 The Laplacian measures graph total variation.

03 Its eigenvectors are graph oscillation modes.

04 The GFT parallels the DFT on a cycle graph.

Diffusion → variation → frequency.