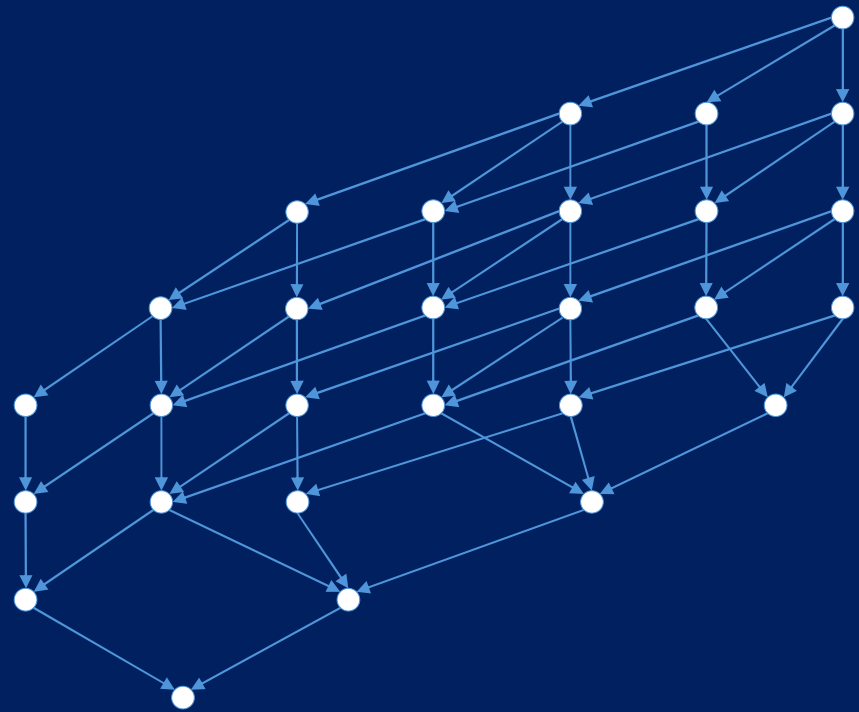


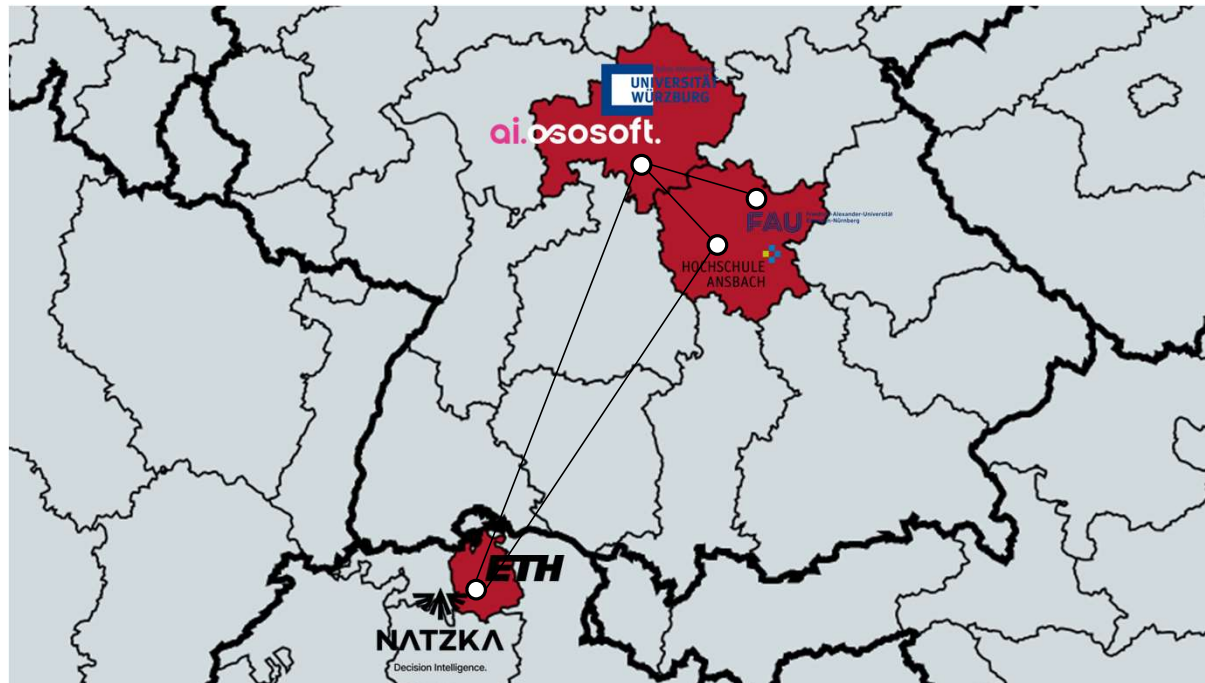
# Graph Signal Processing

Bastian Seifert

Sommersemester 2025



# Lecturer: Dr. Bastian Seifert



**Q: What is your background?**

# Organizational matters

## **Graded student research project (Studienarbeit):**

- Topics: Your own or one from list (will be distributed by mail)
- Use AI (!), but the right one (e.g., <https://notebooklm.google.com/> to make sure you have no hallucinated sources etc.)

## **Deliverables:**

- ~4 pages, IEEE conference format (will distribute template)
- Code used to implement solutions

## **Exercises:**

- Used to deepen knowledge
- You solve between lectures – we talk about possible solutions

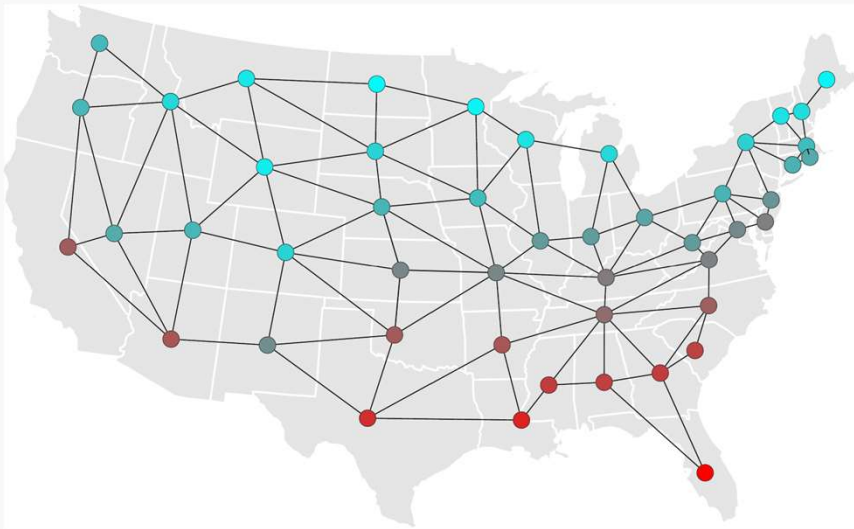
## **Lecture dates (Wednesday, 13:15-14:45, exercise 15:00-16:30):**

19.3 (no exercise), 2.4, 16.4, 30.4, 14.5, 28.5 (no exercise), 11.6, 25.6

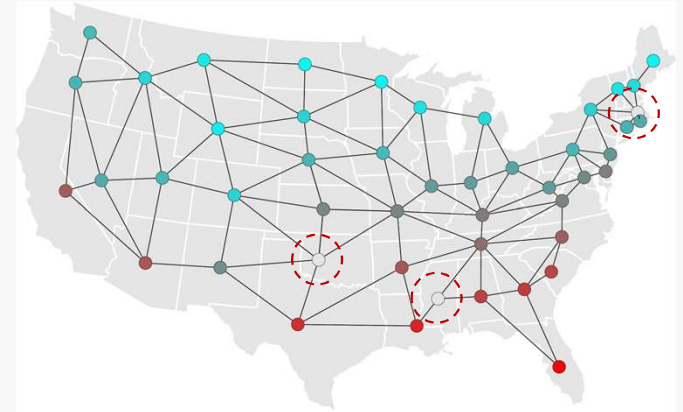
## **Github repository:**

<https://github.com/bastian-seifert/gsp-lecture>

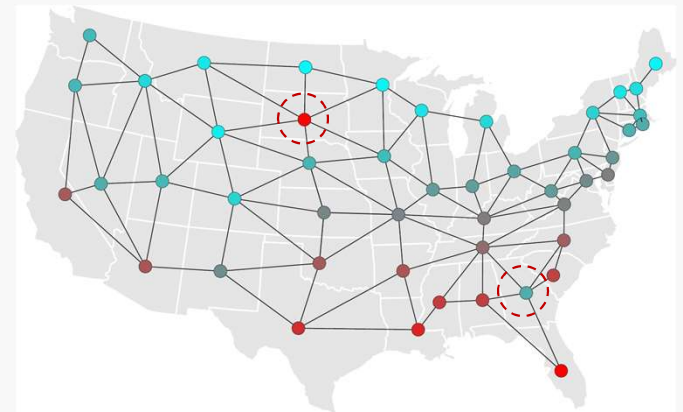
# What we want to investigate



Fill in missing data  
(signal imputation)



Detect outliers

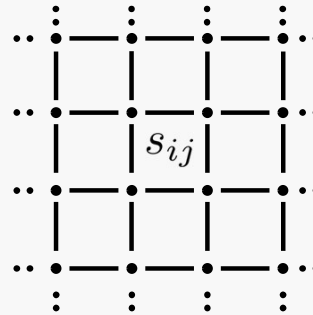


## Classical Signal Process.



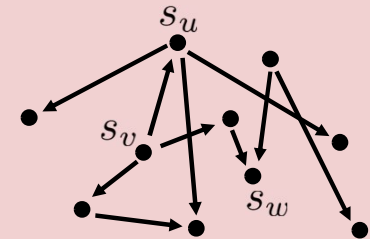
**Signals indexed by time  
(discrete)**

## Image Processing



**Signals indexed by space  
(discrete)**

## Graph Signal Processing



**Signals indexed by graph**

*Shumann 2012  
Sandryhaila 2013*

**Goal: Analyze, process, learn with data supported on graph**

## Graph

$$G = \{V, E\}$$

Nodes (vertices): a finite set

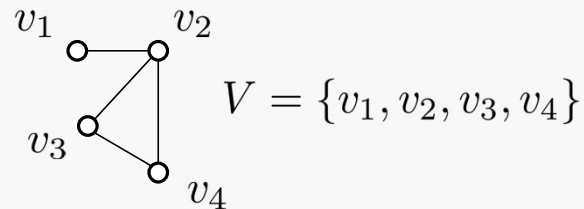
$$V = \{v_1, \dots, v_n\}$$

Edges (links):

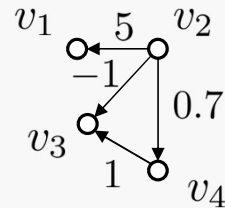
$$E = \{\{v_i, v_k\}, \dots, \{v_j, v_h\}\}$$

Note that if a graph is disconnected, we consider it in this lecture as two graphs.

## Example



$$E = \{\{v_1, v_2\}, \{v_2, v_3\}, \{v_2, v_4\}, \{v_3, v_4\}\}$$



## More structure

### Directed graph

edges have direction

$$E = \{(v_i, v_k), \dots, (v_h, v_j)\}$$

note: there are now pairs instead of a set, i.e.,  $(v_i, v_k) \neq (v_k, v_i)$

### Weighted graph

edges (directed or undirected) have weights

$$W = \{w_{v_i, v_k} \mid (v_i, v_k) \in E\}$$

**Q: Can you think of some real-world examples?**

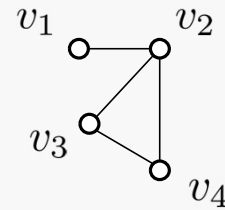
## Adjacency matrix

Represents a graph as matrix

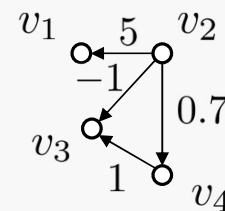
$$A = (A_{v_i, v_j}) = \begin{cases} 1 & \text{if } (v_i, v_j) \in E \\ 0 & \text{otherwise} \end{cases}$$

it has 1 if the edge between nodes at index i and j exists, and 0 otherwise.

## Examples



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



$$A = \begin{bmatrix} 0 & 5 & 0 & 0 \\ 0 & 0 & -1 & 0.7 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

## Sparse matrices

It's very inefficient (and sometimes impossible) to store all these zeros. Hence when working with graphs, we use **sparse matrices**, special data structures for matrices which avoid storing the zeros.

In this lecture you will probably have contact with **Coordinate list (COO)** for creating and **Compressed sparse column (CSC)** for applications formats (lookup *sparse* for Scipy/Numpy or Matlab).

## Laplacian matrix

- The **degree** of a node, is the number of incoming edges
- Collect all degrees in diagonal matrix

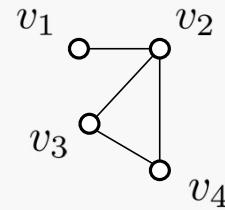
$$D = \text{diag}(\deg(v_1), \dots, \deg(v_n))$$

- The Laplacian matrix is

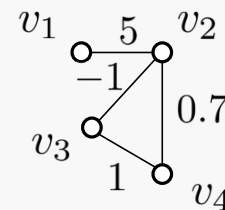
$$L = D - A$$

- Many choices for directed graphs, often one just uses the undirected Laplacian

## Examples



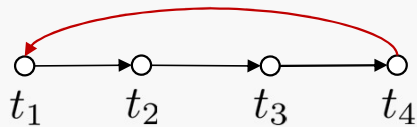
$$L = \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 3 & -1 & -1 \\ 0 & -1 & 2 & -1 \\ 0 & -1 & -1 & 2 \end{bmatrix}$$



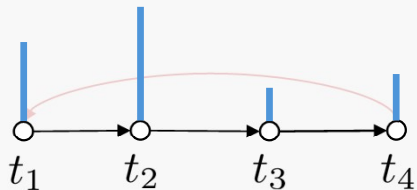
$$L = \begin{bmatrix} 5 & -5 & 0 & 0 \\ -5 & 4.7 & 1 & -0.7 \\ 0 & 1 & 0 & -1 \\ 0 & -0.7 & -1 & 1.7 \end{bmatrix}$$



## Time signals (as graphs)



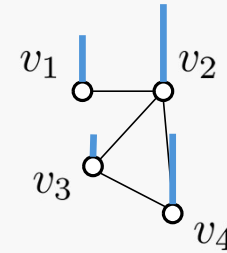
Time graph, representing the flow of time.  
*Red edge is there, as one considers periodic signals (or approximates with periodic signals).*



Numbers associated to each time step are the time signal

$$\mathbf{s} = \begin{bmatrix} s_{t_1} \\ s_{t_2} \\ s_{t_3} \\ s_{t_4} \end{bmatrix} = (s_t)_{t \in T}$$

## Graph signals



Numbers associated to each *node* are the **graph signal**

$$\mathbf{s} = \begin{bmatrix} s_{v_1} \\ s_{v_2} \\ s_{v_3} \\ s_{v_4} \end{bmatrix} = (s_v)_{v \in V}$$

## Social Network

- Nodes = Users
- Edges = Friendship
- Signal = Number of interactions



## 3D point cloud

- Nodes = Location in 3D space
- Edges = Nearest neighborhood
- Signal = Color of voxel



## Sensor Network

- Nodes = Sensors
- Edges = Connection
- Signal = Sensor measurements



**Q: Can you think of more examples?**

## Time shift

$$(Ts)_i = s_{(i-k) \bmod 4}$$

Consider z-transform

$$\mathbf{s} \mapsto s_0 z^0 + s_1 z^{-1} + s_2 z^{-2} + s_3 z^{-3}$$

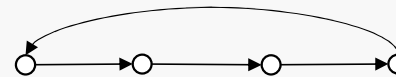
shifting of signals is done by circular delay

$$\begin{aligned} z^{-1} \cdot \mathbf{s} &= s_0 z^{-1} + s_1 z^{-2} + s_2 z^{-3} + s_3 z^0 \\ &= s_3 z^0 + s_0 z^{-1} + s_1 z^{-2} + s_2 z^{-3} \end{aligned}$$

## Matrix representation

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} = \begin{bmatrix} s_3 \\ s_0 \\ s_1 \\ s_2 \end{bmatrix}$$

is adjacency matrix of graph



**The time shift/delay is the central building block of discrete time signal processing**

## Space shift

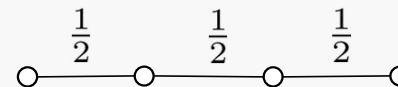
$$(T\mathbf{s})_i = \frac{1}{2}(s_{i-1} + s_{i+1})$$

We will see in later lectures how variations of this shift are connected to discrete cosine/sine transformations

## Matrix representation

$$\begin{bmatrix} 0 & \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} & 0 \end{bmatrix} \cdot \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} = \begin{bmatrix} \frac{1}{2}s_1 \\ \frac{1}{2}(s_0 + s_2) \\ \frac{1}{2}(s_1 + s_3) \\ \frac{1}{2}s_2 \end{bmatrix}$$

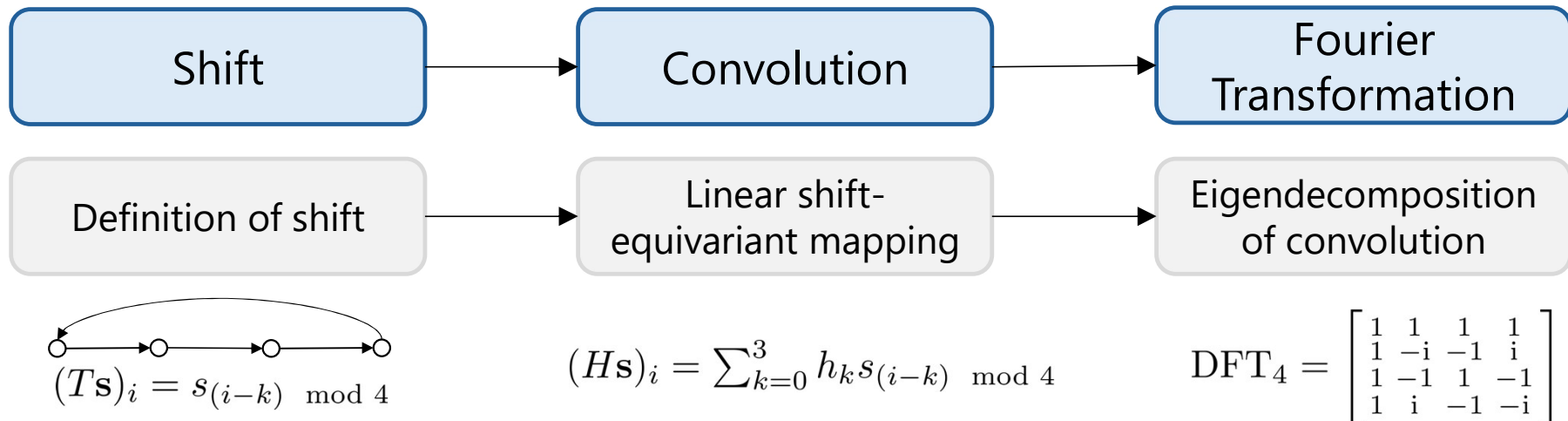
is adjacency matrix of graph



**The space shift can be used to define image processing concepts.**

# Algebraic Signal Processing Theory

Püschel & Moura, 2008, IEEE Trans. Signal Proc.



**Q: How to define shift(s) on Graphs?**

## Shift = adjacency matrix

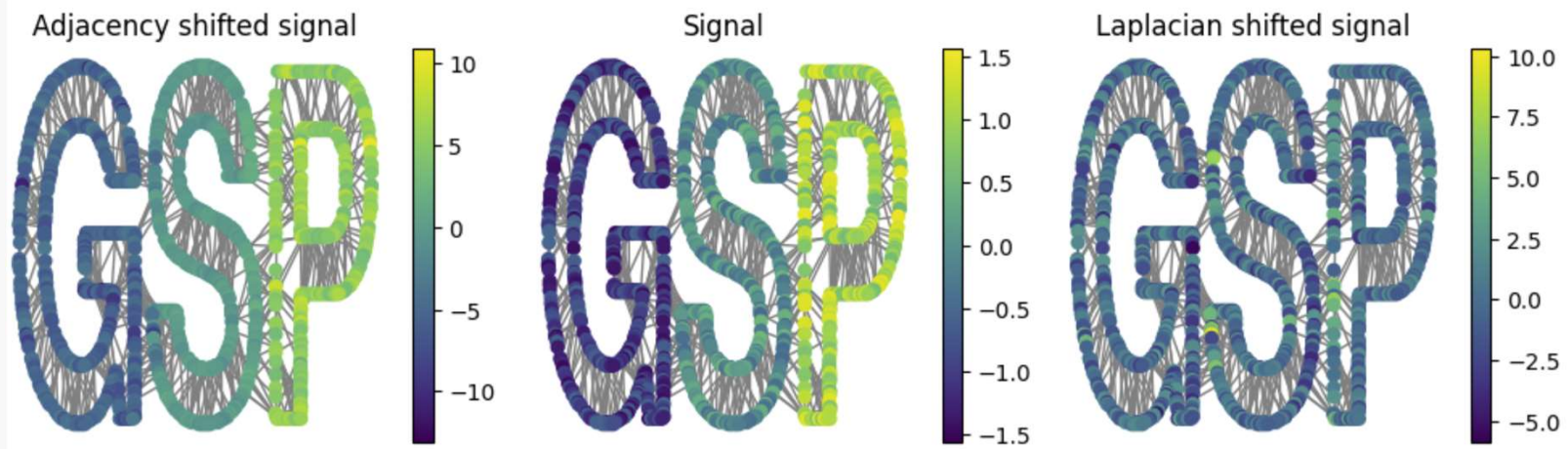
Sandryhaila & Moura 2013, IEEE Trans. Sign. Proc.

- Linear combination of one-hop neighbours
- Generalization of delay

## Shift = graph Laplacian

Shuman & Ricaud & Vandergheynst, IEEE SSP Workshop 2012

- Discrete difference operator
- Generalization of diffusion
- Hard to generalize to directed graphs



**You have to test which shift is better suited for your application.**

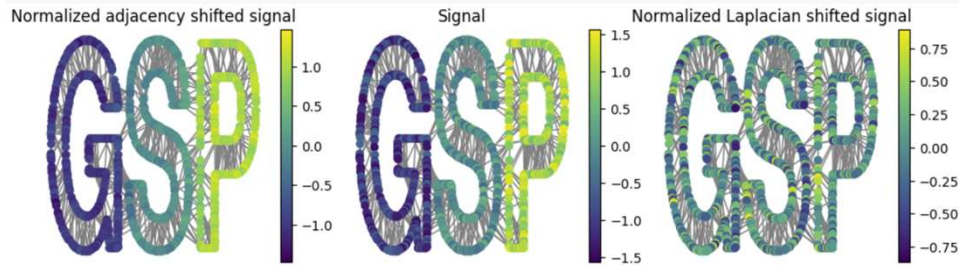
## Shifts, normalized

- Normalization is done via

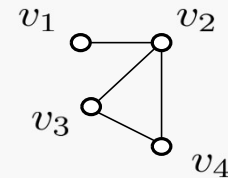
$$A_{\text{norm}} = D^{-1}A$$

$$L_{\text{norm}} = D^{-1}L$$

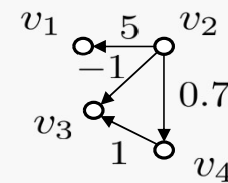
- For Laplacian other normalizations are possible (see exercises)



## Examples



$$A_{\text{norm}} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3} \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 \end{bmatrix}$$



$$A_{\text{norm-indegree}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$A_{\text{norm-outdegree}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ \frac{5}{3} & 0 & -\frac{1}{3} & 0.7 \cdot \frac{1}{3} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

**For directed graphs it matters if you normalize to indegrees or outdegrees!**