

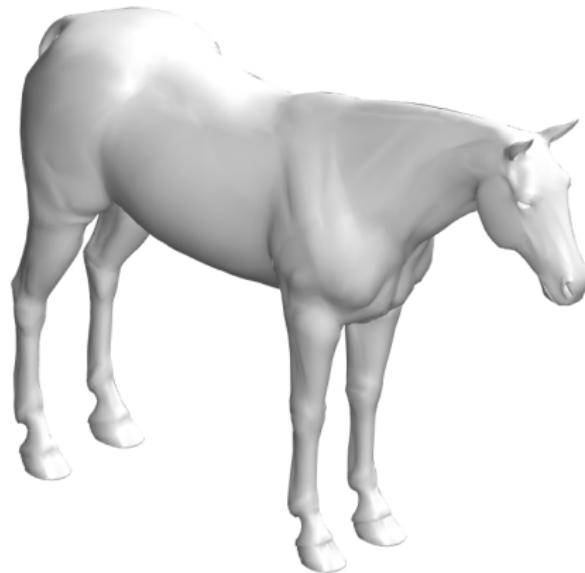
Algorithms

Graph analysis

Emanuele Rodolà
rodola@di.uniroma1.it



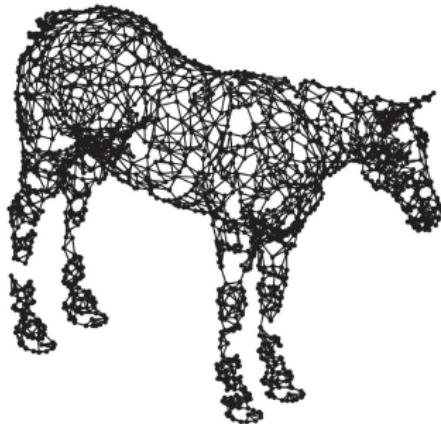
Discrete domains



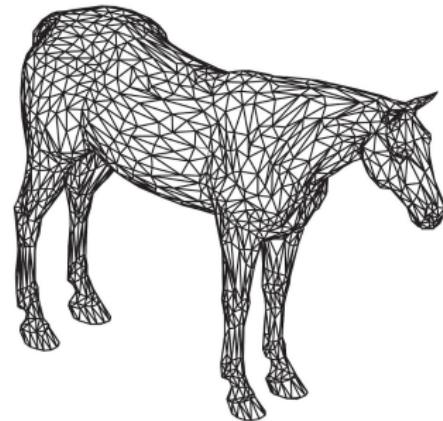
smooth surface

(continuous domain with **infinitely many** points)

Discrete domains



Nearest neighbor [graph](#)



Triangular [mesh](#)

Vertices $\mathcal{V} = \{1, \dots, n\}$

Edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$

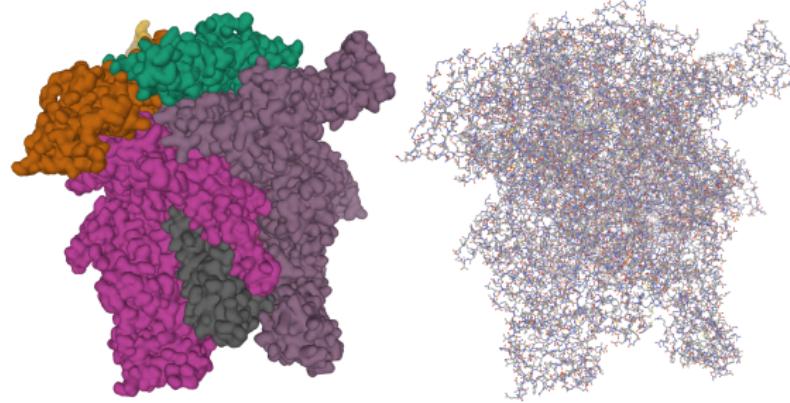
Vertices $\mathcal{V} = \{1, \dots, n\}$

Edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$

Faces $\mathcal{F} = \{(i, j, k) \in \mathcal{V} \times \mathcal{V} \times \mathcal{V} : (i, j), (j, k), (k, i) \in \mathcal{E}\}$

Discrete domains

Escherichia coli RNA polymerase
(<https://www.rcsb.org/3d-view/7KHI>)

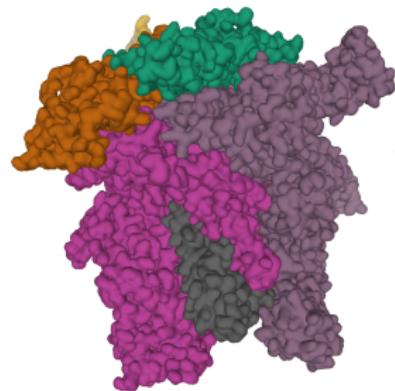


coarse surface

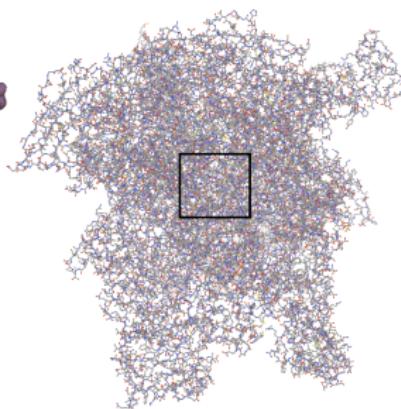
atomic detail

Discrete domains

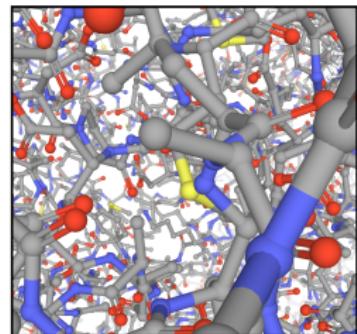
Escherichia coli RNA polymerase
(<https://www.rcsb.org/3d-view/7KHI>)

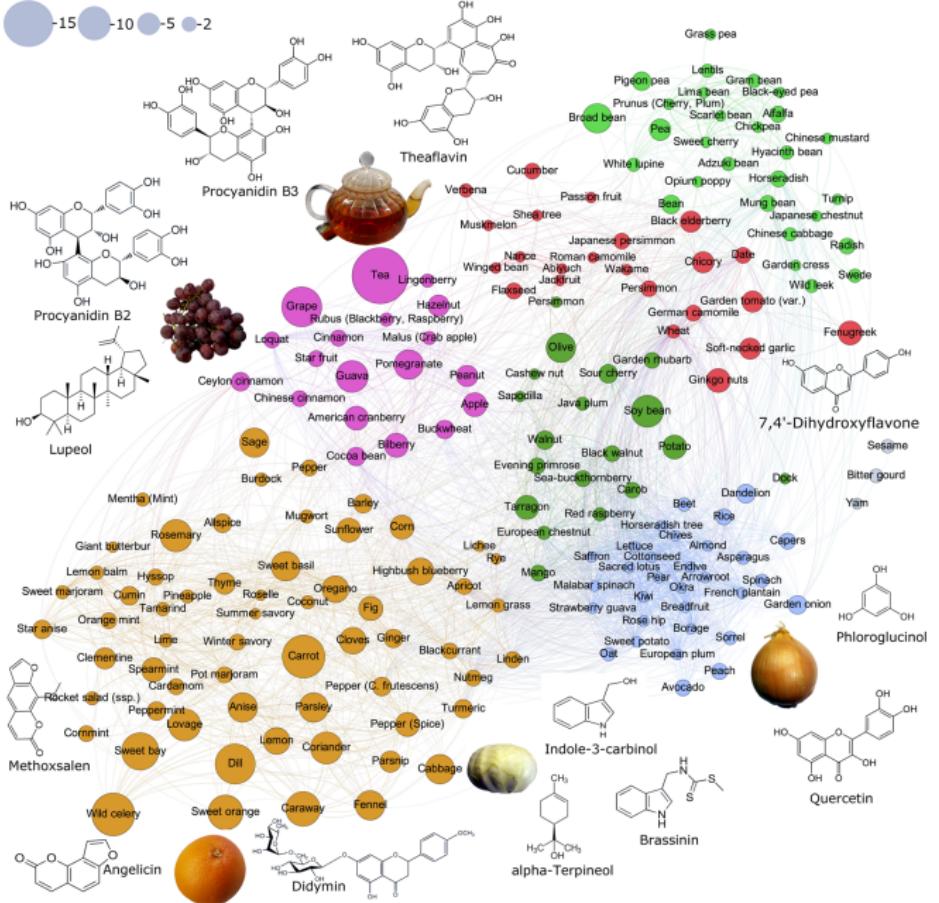


coarse surface



atomic detail

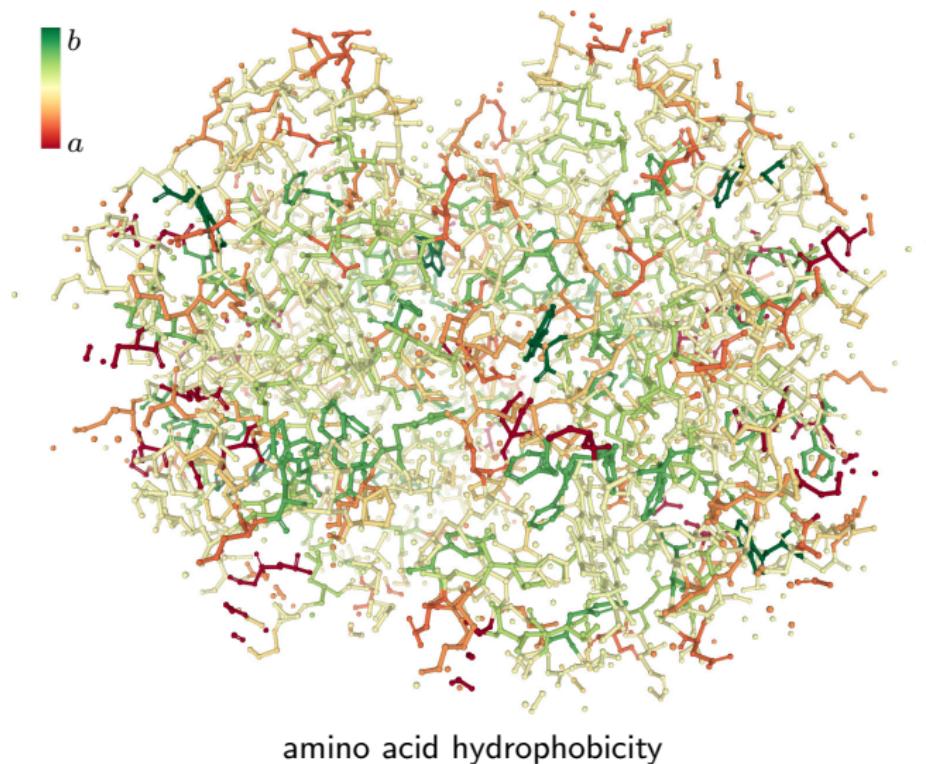




Veselkov et al, "HyperFoods: Machine intelligent mapping of cancer-beating molecules in foods", Nature 2019

Attributed graphs

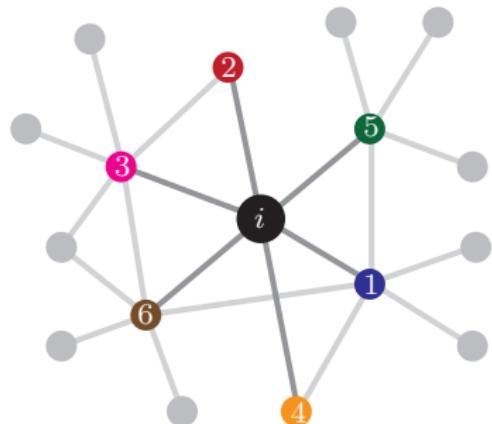
Often we have additional **attributes** on top of the graph structure.



Attributed graphs

Often we have additional **attributes** on top of the graph structure.

Per-node attributes can be encoded as **vectors**.



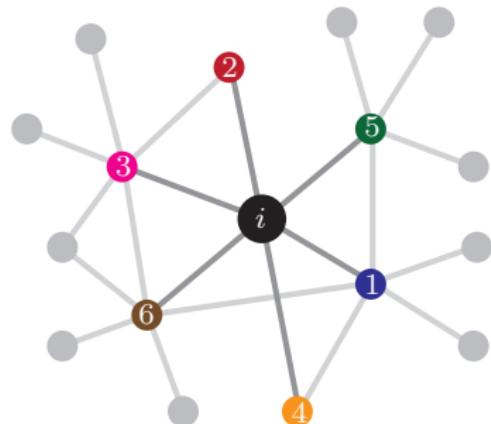
$$\mathbf{x} = \begin{matrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ \vdots \end{matrix}$$

$$\mathbf{x} \in \mathbb{R}^n$$

Attributed graphs

Often we have additional **attributes** on top of the graph structure.

Per-node attributes can be encoded as **vectors**.



$$\mathbf{x} = \begin{matrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ \vdots \end{matrix}$$

$$\mathbf{x} \in \mathbb{R}^n$$

The vertex ordering is **arbitrary** and can be permuted.

Adjacency matrices

Graph connectivity can be encoded in **adjacency matrices**

Let $|V| = n$, $|E| = e$, $|F| = m$ for a mesh $M = (V, E, F)$

Adjacency matrices: Vertex-to-vertex

Graph connectivity can be encoded in **adjacency matrices**

Let $|V| = n$, $|E| = e$, $|F| = m$ for a mesh $M = (V, E, F)$

The **vertex-to-vertex** adjacency is defined as the $n \times n$ binary matrix:

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & \cdots & 1 \\ \vdots & \cdots & \cdots & \cdots & \vdots \\ \vdots & \cdots & \cdots & \cdots & \vdots \\ 1 & 0 & 1 & \cdots & 0 \end{pmatrix}$$

where $a_{ij} = 1$ if vertex v_i is connected to v_j (that is, $e_{ij} \in E$)

Adjacency matrices: Vertex-to-vertex

Graph connectivity can be encoded in **adjacency matrices**

Let $|V| = n$, $|E| = e$, $|F| = m$ for a mesh $M = (V, E, F)$

The **vertex-to-vertex** adjacency is defined as the $n \times n$ binary matrix:

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & \cdots & 1 \\ \vdots & \cdots & \cdots & \cdots & \vdots \\ \vdots & \cdots & \cdots & \cdots & \vdots \\ 1 & 0 & 1 & \cdots & 0 \end{pmatrix}$$

where $a_{ij} = 1$ if vertex v_i is connected to v_j (that is, $e_{ij} \in E$)

- The **diagonal** is always 0

Adjacency matrices: Vertex-to-vertex

Graph connectivity can be encoded in **adjacency matrices**

Let $|V| = n$, $|E| = e$, $|F| = m$ for a mesh $M = (V, E, F)$

The **vertex-to-vertex** adjacency is defined as the $n \times n$ binary matrix:

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & \cdots & 1 \\ \vdots & \cdots & \cdots & \cdots & \vdots \\ \vdots & \cdots & \cdots & \cdots & \vdots \\ 1 & 0 & 1 & \cdots & 0 \end{pmatrix}$$

where $a_{ij} = 1$ if vertex v_i is connected to v_j (that is, $e_{ij} \in E$)

- The **diagonal** is always 0
- **A** is **symmetric** (assuming undirected graphs)

Adjacency matrices: Vertex-to-vertex

Graph connectivity can be encoded in **adjacency matrices**

Let $|V| = n$, $|E| = e$, $|F| = m$ for a mesh $M = (V, E, F)$

The **vertex-to-vertex** adjacency is defined as the $n \times n$ binary matrix:

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & \cdots & 1 \\ \vdots & \cdots & \cdots & \cdots & \vdots \\ \vdots & \cdots & \cdots & \cdots & \vdots \\ 1 & 0 & 1 & \cdots & 0 \end{pmatrix}$$

where $a_{ij} = 1$ if vertex v_i is connected to v_j (that is, $e_{ij} \in E$)

- The **diagonal** is always 0
- **A** is **symmetric** (assuming undirected graphs)
- Each row and column has at least one 1 (that is, $\sum_{ij} a_{ij} = 2e$)

Adjacency matrices as operators

From a linear algebra perspective:

- Per-node attributes are **functions** $f : V \rightarrow \mathbb{R}^k$.
- Adjacency matrices as **operators** that can be applied to functions.

Adjacency matrices as operators

From a linear algebra perspective:

- Per-node attributes are **functions** $f : V \rightarrow \mathbb{R}^k$.
- Adjacency matrices as **operators** that can be applied to functions.

For example, $\mathbf{g} = \mathbf{A}\mathbf{f}$ represents a new function g defined as:

$$\begin{pmatrix} g_1 \\ g_2 \\ \vdots \\ g_n \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & \cdots & 1 \\ \vdots & \dots & \dots & \dots & \vdots \\ \vdots & \dots & \dots & \dots & \vdots \\ 1 & 0 & 1 & \cdots & 0 \end{pmatrix} \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{pmatrix}$$

Adjacency matrices as operators

From a linear algebra perspective:

- Per-node attributes are **functions** $f : V \rightarrow \mathbb{R}^k$.
- Adjacency matrices as **operators** that can be applied to functions.

For example, $\mathbf{g} = \mathbf{A}\mathbf{f}$ represents a new function g defined as:

$$g(v_i) = \sum_{e_{ij} \in E} f(v_j)$$

Adjacency matrices as operators

From a linear algebra perspective:

- Per-node attributes are **functions** $f : V \rightarrow \mathbb{R}^k$.
- Adjacency matrices as **operators** that can be applied to functions.

For example, $\mathbf{g} = \mathbf{A}\mathbf{f}$ represents a new function g defined as:

$$g(v_i) = \sum_{e_{ij} \in E} f(v_j)$$

One can construct new operators such as $\mathbf{I} - \mathbf{A}$:

$$g(v_i) = f(v_i) - \sum_{e_{ij} \in E} f(v_j)$$

And similarly for more complex (but still **linear**) operations.

Graph Laplacian

Given an **attributed graph** $G = (V, E)$, consider this condition on node v_i :

$$f(v_i) - \frac{1}{d_i} \sum_{j:(i,j) \in E} f(v_j) = 0$$

where d_i is the **degree** of v_i .

Graph Laplacian

Given an **attributed graph** $G = (V, E)$, consider this condition on node v_i :

$$f(v_i) - \underbrace{\frac{1}{d_i} \sum_{j:(i,j) \in E} f(v_j)}_{\substack{\text{average of } f \\ \text{on the neighbors of } v_i}} = 0$$

where d_i is the **degree** of v_i .

Graph Laplacian

Given an **attributed graph** $G = (V, E)$, consider this condition on node v_i :

$$f(v_i) - \underbrace{\frac{1}{d_i} \sum_{j:(i,j) \in E} f(v_j)}_{\substack{\text{average of } f \\ \text{on the neighbors of } v_i}} = 0$$

where d_i is the **degree** of v_i .

The equation is satisfied if $f(v_i)$ equals the average of its neighbors.

Graph Laplacian

Given an attributed graph $G = (V, E)$, consider this condition on node v_i :

$$f(v_i) - \underbrace{\frac{1}{d_i} \sum_{j:(i,j) \in E} f(v_j)}_{\text{average of } f \text{ on the neighbors of } v_i} = 0$$

where d_i is the degree of v_i .

The equation is satisfied if $f(v_i)$ equals the average of its neighbors.

In matrix notation, we define the $n \times n$ matrix \mathbf{L} as:

$$L_{ij} = \begin{cases} 1 & \text{if } i = j \\ -\frac{1}{d_i} & \text{if } e_{ij} \in E \\ 0 & \text{otherwise} \end{cases}$$

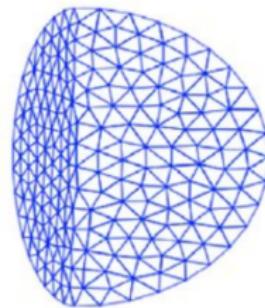
called the Laplacian of G .

Example

Consider the **planar** graph $G = (V, E)$ with per-node attributes:

$$f(v_i) = \{x_i, y_i\},$$

which are 2D coordinates for all the nodes.



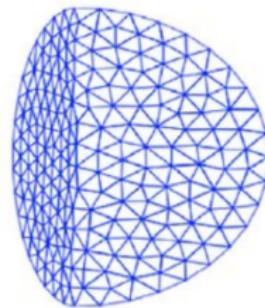
Example

Consider the **planar** graph $G = (V, E)$ with per-node attributes:

$$f(v_i) = \{x_i, y_i\},$$

which are 2D coordinates for all the nodes.

Encode the attributes in a $n \times 2$ matrix \mathbf{F} .



Example

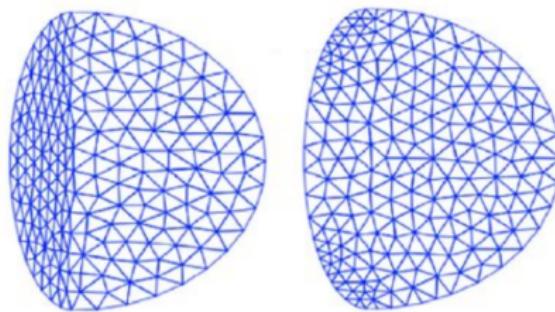
Consider the **planar** graph $G = (V, E)$ with per-node attributes:

$$f(v_i) = \{x_i, y_i\},$$

which are 2D coordinates for all the nodes.

Encode the attributes in a $n \times 2$ matrix \mathbf{F} .

The equation $\mathbf{LF} = \mathbf{0}$ is satisfied if all the v_i are in the **barycenter** of their immediate neighbors (except for the boundary vertices).



Laplacian eigen-decomposition

Consider the **spectral decomposition** of the Laplacian:

$$\mathbf{L}\mathbf{u} = \lambda\mathbf{u}$$

Laplacian eigen-decomposition

Consider the **spectral decomposition** of the Laplacian:

$$\mathbf{L}\mathbf{u} = \underbrace{\lambda}_{\text{eigenvalue}} \quad \underbrace{\mathbf{u}}_{\text{eigenvector}}$$

Laplacian eigen-decomposition

Consider the **spectral decomposition** of the Laplacian:

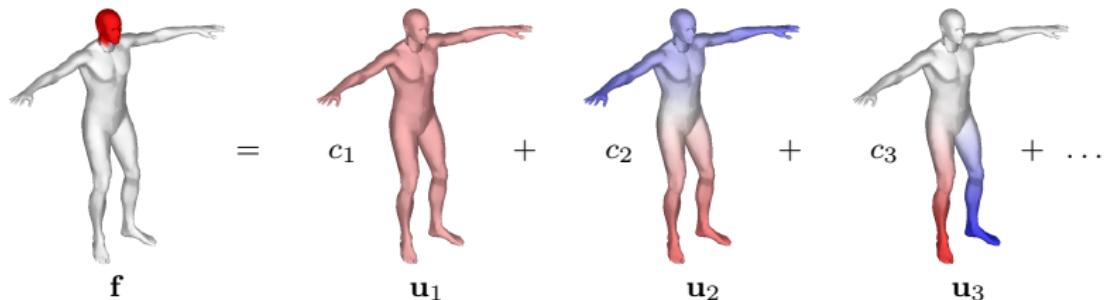
$$\mathbf{L}\mathbf{u}_i = \lambda_i \mathbf{u}_i, \quad i = 1, \dots, n$$

Laplacian eigen-decomposition

Consider the **spectral decomposition** of the Laplacian:

$$\mathbf{L}\mathbf{u}_i = \lambda_i \mathbf{u}_i, \quad i = 1, \dots, n$$

Any given function $f : V \rightarrow \mathbb{R}$ can be written as a unique **linear combination** of the Laplacian eigenvectors $\{\mathbf{u}_i\}$:

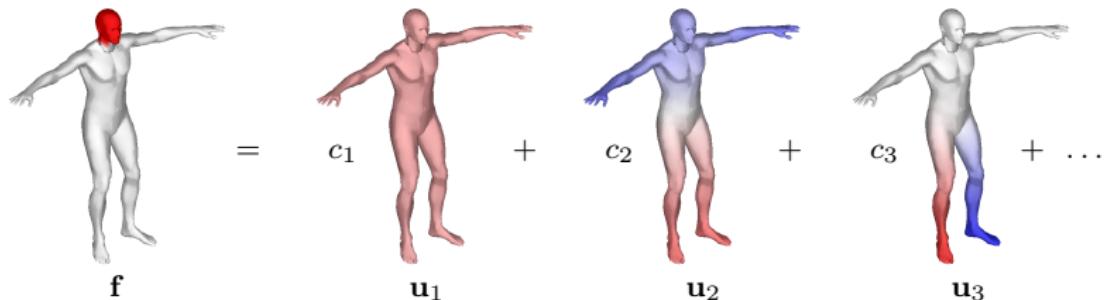


Laplacian eigen-decomposition

Consider the **spectral decomposition** of the Laplacian:

$$\mathbf{L}\mathbf{u}_i = \lambda_i \mathbf{u}_i, \quad i = 1, \dots, n$$

Any given function $f : V \rightarrow \mathbb{R}$ can be written as a unique **linear combination** of the Laplacian eigenvectors $\{\mathbf{u}_i\}$:



Since each eigenvector \mathbf{u}_i can be interpreted as a function $u_i : V \rightarrow \mathbb{R}$, they are also called **eigenfunctions**.

Spectrum

The **spectrum** is the set of Laplacian eigenvalues.

It has a canonical ordering, and the **first eigenvalue** is always = 0:

$$0 = \lambda_1 < \lambda_2 \leq \lambda_3 \leq \dots \rightarrow 2$$

Spectrum

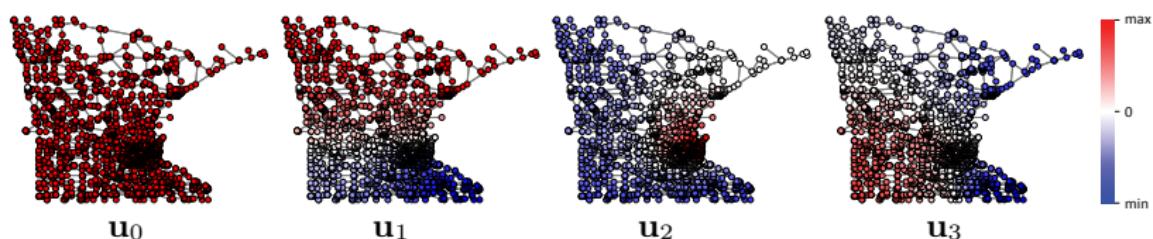
The **spectrum** is the set of Laplacian eigenvalues.

It has a canonical ordering, and the **first eigenvalue** is always $= 0$:

$$0 = \lambda_1 < \lambda_2 \leq \lambda_3 \leq \dots \rightarrow 2$$

The eigenfunctions follow the ordering of the eigenvalues.

Thus we can talk of the “first” k eigenfunctions:



First 4 eigenfunctions of a graph Laplacian

Band-limited approximation

Each eigenvalue quantifies the **oscillations** of its associated eigenfunction.

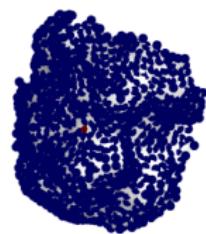
Small eigenvalues are associated to **smooth** eigenfunctions.

Band-limited approximation

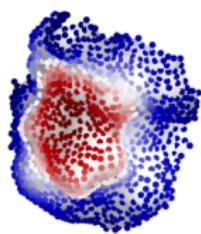
Each eigenvalue quantifies the **oscillations** of its associated eigenfunction.

Small eigenvalues are associated to **smooth** eigenfunctions.

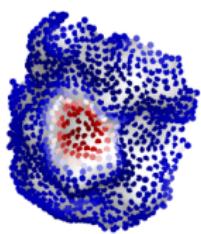
Using the first k eigenfunctions to approximate a given function:



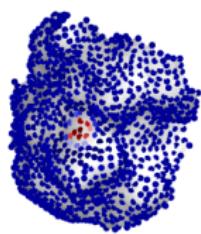
$k = n$



$k = 10$



$k = 30$



$k = 100$

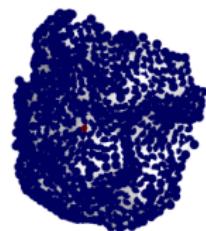
Band-limited approximation

Each eigenvalue quantifies the **oscillations** of its associated eigenfunction.

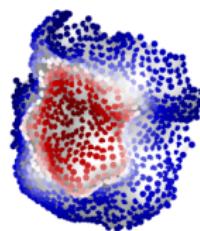
Small eigenvalues are associated to **smooth** eigenfunctions.

Using the first k eigenfunctions to approximate a given function:

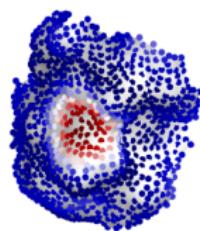
- Reduces high-frequency **noise**.



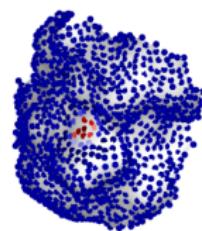
$k = n$



$k = 10$



$k = 30$



$k = 100$

Band-limited approximation

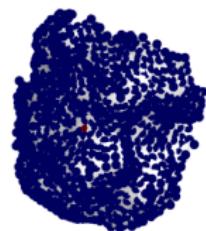
Each eigenvalue quantifies the **oscillations** of its associated eigenfunction.

Small eigenvalues are associated to **smooth** eigenfunctions.

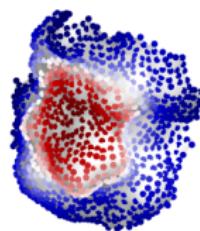
Using the first k eigenfunctions to approximate a given function:

- Reduces high-frequency **noise**.
- Enables **efficient** algorithms.

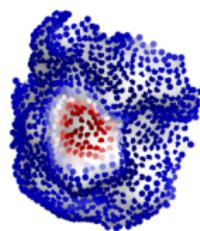
Manipulate and store $k \ll n$ coefficients instead of n function values.



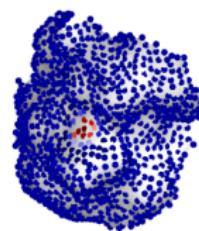
$k = n$



$k = 10$



$k = 30$



$k = 100$

Band-limited approximation

Each eigenvalue quantifies the **oscillations** of its associated eigenfunction.

Small eigenvalues are associated to **smooth** eigenfunctions.

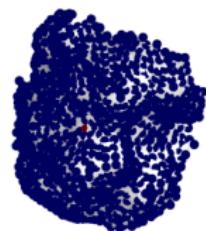
Using the first k eigenfunctions to approximate a given function:

- Reduces high-frequency **noise**.

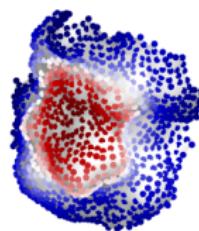
- Enables **efficient** algorithms.

Manipulate and store $k \ll n$ coefficients instead of n function values.

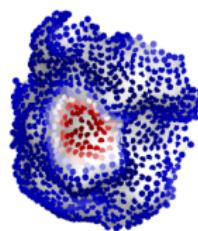
- Guarantees invariance to **deformations**.



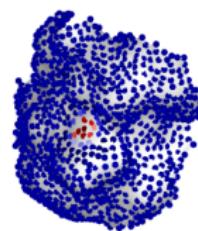
$k = n$



$k = 10$



$k = 30$



$k = 100$

Multiplicity

For a given graph G , an eigenvalue might appear multiple times, e.g.:

$$\lambda_{16} = \lambda_{17} = 1.1451$$

In this example, we say that eigenvalue 1.1451 has [multiplicity 2](#).

Multiplicity is known to be related to [symmetries](#) in the graph.

Multiplicity

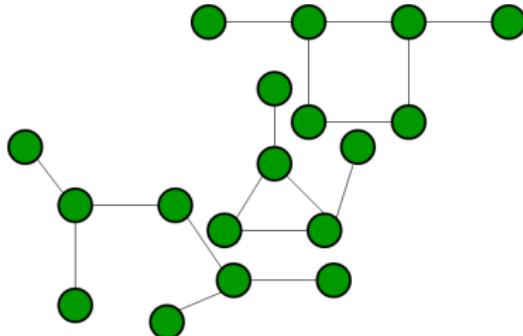
For a given graph G , an eigenvalue might appear multiple times, e.g.:

$$\lambda_{16} = \lambda_{17} = 1.1451$$

In this example, we say that eigenvalue 1.1451 has **multiplicity 2**.

Multiplicity is known to be related to **symmetries** in the graph.

The multiplicity of $\lambda_1 = 0$ counts the **connected components** in G :



$$\lambda_1 = \lambda_2 = \lambda_3 = 0$$

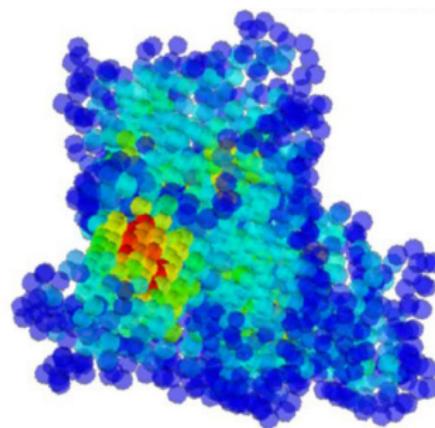
Informativity

Spectral quantities encode a whole lot of **information** about the underlying structure (a surface or a graph).

Informativity

Spectral quantities encode a whole lot of **information** about the underlying structure (a surface or a graph).

For example, eigenfunctions **localize** around hot spots of vibrational energy (active sites in the case of enzymes).



glutamate synthase

Chalopin et al, "Universality of fold-encoded localized vibrations in enzymes", Nature 2019