4 GRAPH FOURIER TRANSFORM

4.1 THE FOURIER TRANSFORM

The Fourier transform is a fundamental tool in mathematical analysis that was introduced by Joseph Fourier in to solve the heat equation that in the one dimensional case reads

$$\partial_t u(x,t) = \alpha\, \partial_x^2 u(x,t), \tag{4.1}$$

The heat equation

where α is the thermal diffusivity. The Fourier transform was introduced with the scope of solving this differential equation and it consists of moving to the "space of frequencies" that, in some cases, like this one, can be very convenient. This powerful tool of signal processing, in fact, allows one to define simple operations such as filtering and convolutions in the frequency domain that would otherwise be very hard to perform in the original one. Given its power, we want to extend its definition to non-Euclidean domains and in particular to functions that are defined on the vertices of a graph.

As a first step, we here re-derive the definition of Fourier transform, starting from Equation 4.1 to then attempt to generalize it to the graph domain.

4.1.1 DERIVING THE FOURIER TRANSFORM

In Equation 4.1, the second derivative can be seen as an operator that acts on the function $u(x,t)$. The simplest way to derive the Fourier transform is to see it as a decomposition of $u(x,t)$ on the eigenfunctions of that operator. We start from these two basic properties

The eigenfunctions of the second derivative

$$\partial_x^2 \, e^{ikx} = -k^2 e^{ikx}, \tag{4.2}$$

$$\frac{1}{2\pi}\int_{\mathbb{R}} dk \, e^{ikx} = \delta(x). \tag{4.3}$$

These two equations imply that e^{ikx} are the eigenfunctions of the second derivative and they form an orthogonal basis. For a given function $f(x)$ we can thus write

The Fourier transform

$$\begin{aligned} f(x) &= \int_{\mathbb{R}} dy \, f(y)\delta(x-y) \\ &\overset{(a)}{=} \frac{1}{2\pi}\int_{\mathbb{R}} dy \, f(y) \int_{\mathbb{R}} dk \, e^{ik(x-y)} \\ &\overset{(b)}{=} \frac{1}{2\pi}\int_{\mathbb{R}} dk \, e^{ikx} \underbrace{\int_{\mathbb{R}} dy \, f(y)e^{-iky}}_{\hat{f}(k)} \\ &\overset{(c)}{=} \frac{1}{2\pi}\int_{\mathbb{R}} dk \, \hat{f}(k)e^{ikx}, \end{aligned}$$

where in (a) we used the definition of the δ function given in Equation (4.3), in (b) we inverted the order of the two integrals and in (c) we introduced the Fourier transform definition as $\hat{f}_k$. The function $f(x)$ is then expressed as combination of the eigenfunctions of the second derivative – Laplacian, in higher dimensions – operator. Here, the $\hat{f}(k)$ plays the role of the "weight coefficient" of each eigenfunction.

Repeating these steps, we now derive a the graph Fourier transform .

4.2 THE GRAPH FOURIER TRANSFORM

We here derive the *Graph Fourier transform* (GFT) definition, first writing the heat equation on a graph, then decomposing a signal on the basis of the new Laplacian operator as done before.

4.2.1 HEAT DIFFUSION ON GRAPHS

Consider a graph $\mathcal{G}(\mathcal{V},\mathcal{E})$ and let $u_i(t)$ be a signal defined on node $i \in \mathcal{V}$ at time t. We suppose that this signal evolves with a diffusive process on the

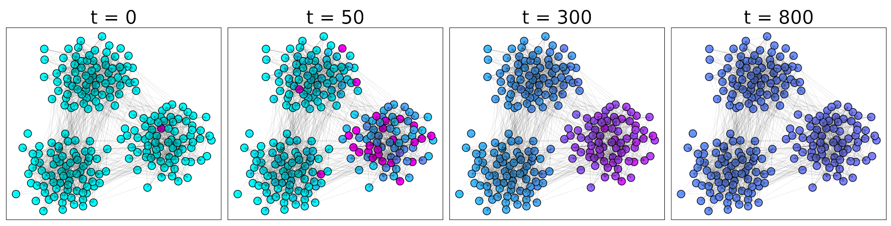

Figure 4.1: **Diffusion on a graph with communities**. In color code we have the value $u_i(t)$ and the title indicates the four different time-steps.

graph and that, at each time-step, a i exchanges a fraction α of information $u_i(t)$ with each of its neighbors. In equations, this becomes

$$u_i(t+dt) = u_i(t) + \alpha \sum_{j\in\mathcal{V}} A_{ij}\big(\underbrace{u_j(t)}_{j\to i} - \underbrace{u_i(t)}_{i\to j}\big).$$

This allows us to rewrite the diffusion equation in vector form as

Heat diffusion on a graph

$$\partial_t \boldsymbol{u}(t) = -\alpha(D-A)\boldsymbol{u}(t) := -\alpha L\boldsymbol{u}(t),$$

where we introduced the graph Laplacian matrix L. In Figure 4.1we show a simple example of diffusion on a graph with three groups of nodes – called communities – that are more densely connected among themselves than with other. We initialize $u_i(t) = 0$ for all nodes except and perform the simulation. The result reported at three successive time-steps evidences a diffusion process that follows a concept of proximity on the graph: first the signal is propagated in the node in the same community and to few of the other communities ($t = 50$), then it progressively tends to a homogeneous distribution. Let us now formally define the graph Laplacian matrix

The graph Laplacian matrix

The graph Laplacian matrix

Given an undirected graph $\mathcal{G}(\mathcal{V},\mathcal{E})$ with $A \in [0,1]^{n\times n}$ as adjacency matrix and $D = \mathrm{diag}(A\mathbf{1}_n)$ the diagonal degree matrix, the graph Laplacian associated to $\mathcal{G}(\mathcal{V},\mathcal{E})$ is

$$L = D - A \tag{4.4}$$

Given this matrix, we now use it to decompose the signal $\boldsymbol{u}$ on its basis and show that it how we can relate this decomposition to a “frequency domain”.

4.2.2 DECOMPOSE A SIGNAL ON THE GRAPH FOURIER MODES

Following the same procedure as above, to define the graph Fourier transform we decompose the signal on the orthonormal basis defined by the eigenvectors of L. Since L is a Hermitian matrix, these eigenvectors form and orthonormal basis and they play the same role as e^{ikx} in the classical Fourier transform. We denote with $\boldsymbol{x}_k$ the eigenvector k associated with λ_k, the k smallest eigenvalue of L *i.e.* $L\boldsymbol{x}_k = \lambda_k \boldsymbol{x}_k$, then

The eigenvectors of L form an orthonormal basis

$$I_n = \sum_{k=1}^{n} \boldsymbol{x}_k \boldsymbol{x}_k^T.$$

For any vector $\boldsymbol{u} \in \mathbb{R}^n$ defined on the vertices of $\mathcal{G}(\mathcal{V}, \mathcal{E})$, we can write

Projecting $\boldsymbol{u}$ on the basis of L

$$\boldsymbol{u} = I_n \boldsymbol{u} = \sum_{k=1}^{n} \boldsymbol{x}_k \underbrace{\boldsymbol{x}_k^T \boldsymbol{u}}_{\hat{u}_k} = \sum_{k=1}^{n} \boldsymbol{x}_k \hat{u}_k.$$

By analogy with the classical Fourier transform, we define $\hat{\boldsymbol{u}} = X^T \boldsymbol{u}$ is the GFT, having denoted with $X \in \mathbb{R}^{n \times n}$ the matrix with the eigenvectors of L in its columns.

The graph Fourier transform

The graph Fourier transform

Consider an undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ with L the graph Laplacian matrix of Equation (4.4), and a signal $\boldsymbol{u} \in \mathbb{R}^n$ defined on the set $\mathcal{V}$. Let $L = X \Lambda X^T$ be the eigenvector decomposition of L, with $X \in \mathbb{R}^{n \times n}$ the matrix with the eigenvectors of L in its columns. We define the GFT $\hat{\boldsymbol{u}}$ and its inverse as

$$\hat{\boldsymbol{u}} = X^T \boldsymbol{u} \tag{4.5}$$

$$\boldsymbol{u} = X \hat{\boldsymbol{u}}. \tag{4.6}$$

The expression of the inverse GFT easily comes from the property $XX^T = I_n$. Now that we introduced a definition for the GFT we want to see how it actually relates to some concept of frequency of the signal on the graph.

4.2.3 FREQUENCIES ON GRAPHS

The best way to understand the relation between the graph Laplacian and the concept of frequency is through this lemma that we will then prove.

Lemma 4.1. *Let $\boldsymbol{u} \in \mathbb{R}^n$ be a vector defined on the vertices of a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ with graph Laplacian matrix L. then*

$$\boldsymbol{u}^T L \boldsymbol{u} = \frac{1}{2} \sum_{i,j \in \mathcal{V}} A_{ij} (u_i - u_j)^2. \tag{4.7}$$

Proof.

$$\begin{aligned} \boldsymbol{u}^T L \boldsymbol{u} &= \sum_{i,j \in \mathcal{V}} u_i L_{ij} u_j \\ &\overset{(a)}{=} \sum_{i,j \in V} u_i D_{ij} u_j - \sum_{i,j \in \mathcal{V}} u_i A_{ij} u_j \end{aligned}$$

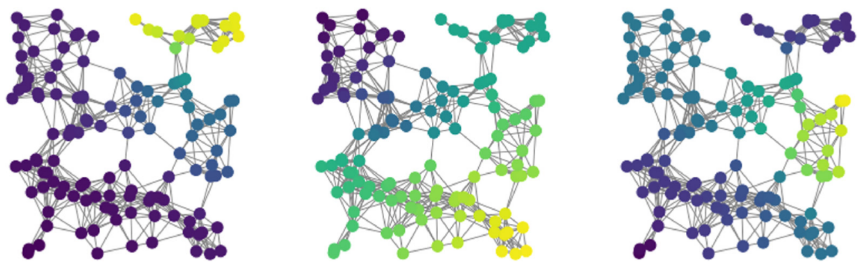

Figure 4.2: **Some graph Fourier modes on a graph**. From left to right: the eigenvector associated to the second, third and fourth smallest eigenvalue of L. Color-map: positive values in yellow, negative ones in blue. Picture taken from 10.1016/j.crhy.2019.08.003.

$$
\begin{aligned}
&= \sum_{i,j\in V} u_i d_i \delta_{ij} u_j - \sum_{i,j\in\mathcal{V}} u_i A_{ij} u_j \\
&= \sum_{i\in V} u_i^2 d_i - \sum_{i,j\in\mathcal{V}} u_i A_{ij} u_j \\
&\overset{(b)}{=} \sum_{i,j\in V} u_i^2 A_{ij} - \sum_{i,j\in\mathcal{V}} u_i A_{ij} u_j \\
&\overset{(c)}{=} \frac{1}{2}\left(\sum_{i,j\in V} u_i^2 A_{ij} + \sum_{i,j\in V} u_j^2 A_{ji}\right) - \sum_{i,j\in\mathcal{V}} u_i A_{ij} u_j \\
&\overset{(d)}{=} \frac{1}{2}\left(\sum_{i,j\in V} u_i^2 A_{ij} + \sum_{i,j\in V} u_j^2 A_{ij}\right) - \sum_{i,j\in\mathcal{V}} u_i A_{ij} u_j \\
&= \frac{1}{2}\sum_{i,j\in V} A_{ij}(u_i^2 + u_j^2 - 2u_i u_j) \\
&= \frac{1}{2}\sum_{i,j\in V} A_{ij}(u_i - u_j)^2,
\end{aligned}
$$

where in (a) we used the definition of graph Laplacian $L = D - A$, in (b) we rewrote the degree $d_i = \sum_{j\in\mathcal{V}} A_{ij}$, in (c) we exploited the fact that i, j are dummy variables that can be inverted and in (d) we used $A_{ij} = A_{ji}$. □

The consequence of Lemma 4.1 is that the eigenvalues of L can be interpreted as a "frequency" of the corresponding eigenvector, in the sense that they quantify how fast $\boldsymbol{x}_k$ changes on the graph.

$$\lambda_k = \boldsymbol{x}_k^T L \boldsymbol{x}_k = \frac{1}{2}\sum_{i,j\in\mathcal{V}} (x_{k,i} - x_{k,j})^2,$$

so if the eigenvector $\boldsymbol{x}_k$ changes *smoothly* on the graph, *i.e.* is has similar value for neighboring nodes, the corresponding eigenvalue will be small. Figure 4.2 shows in color code the frequency of the second, third and fourth eigenvectors of L on a graph.

Using the GFT notation we can write, for a generic vector $\boldsymbol{u}$

$$\boldsymbol{u}^T L \boldsymbol{u} = \boldsymbol{u} \left(\sum_{k=1}^{n} \lambda_k \boldsymbol{x}_k \boldsymbol{x}_k^T \right) \boldsymbol{u} = \sum_{k=1}^{n} \lambda_k \hat{u}_k^2.$$

The term $\hat{u}_k$ is a weight that accounts for how much $\boldsymbol{u}$ is aligned with $\boldsymbol{x}_k$ and λ_k is the corresponding frequency. The scalar $\boldsymbol{u}^T L \boldsymbol{u}$ can hence be used to quantify how fast the signal $\boldsymbol{u}$ changes on the graph.

To summarize, the eigenvectors of the graph Laplacian matrix are the Fourier eigenmodes and by projecting a signal over them we are in the Fourier space of frequencies that are the eigenvalues of the same matrix. We now detail some basic but relevant property of the eigenvalues of the graph Laplacian matrix that is fundamental to understand the GFT.

4.2.4 SOME BASIC SPECTRAL PROPERTIES OF THE GRAPH LAPLACIAN

We here list and prove three basic facts about the graph Laplacian eigenvalues. We first formally state that L does not have any negative eigenvalues.

Corollary 4.1. *The graph Laplacian matrix L is positive semi-definite,* i.e. *it does not have negative eigenvalues. The all one vector* $\mathbf{1}_n$ *is an eigenvector of L with eigenvalue* 0.

Proof. This is a corollary to Lemma 4.1. Let $\boldsymbol{x}_k$ be an eigenvector of L with eigenvalue λ_k, then

$$\lambda_k = \boldsymbol{x}_k^T L \boldsymbol{x}_k = \frac{1}{2} \sum_{i,j \in \mathcal{V}} (x_{k,i} - x_{k,j})^2 \geq 0.$$

The equality is reached for $\boldsymbol{x}_1 = \mathbf{1}_n$ that is an eigenvector because $D\mathbf{1}_n = \boldsymbol{d}$ and $A\mathbf{1}_n = \boldsymbol{d}$, where $\boldsymbol{d}$ denotes the degree vector. □

The second property concerns the multiplicity of the 0 eigenvalue and its relation to the connectedness of the graph.

Corollary 4.2. *The multiplicity of the* 0 *eigenvalue of graph Laplacian matrix L equals the number of connected components of the graph.*

Proof. Also in this case the proof is a straightforward consequence of Lemma 4.1. Consider a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ with c connected components $\{\mathcal{V}_a\}_{a=1,\dots,c}$ so that

$$\cup_{a=1,\dots,c} \mathcal{V}_a = \mathcal{V}$$
$$\forall\, a \neq b \quad \mathcal{V}_a \cap \mathcal{V}_b = \emptyset.$$

Denote with $\boldsymbol{y}^{(a)}$ the vector with entries $y_i^{(a)} = 1$ if $i \in \mathcal{V}_a$ and 0 otherwise. Then

$$\begin{aligned}
(L\boldsymbol{y}^{(a)})_i &= \sum_{j\in\mathcal{V}} (D-A)_{ij} y_j^{(a)} \\
&= \sum_{b=1}^{c} \sum_{j\in\mathcal{V}_b} (D-A)_{ij} y_j^{(a)} \\
&\overset{(a)}{=} \sum_{j\in\mathcal{V}_a} (D-A)_{ij} \\
&\overset{(b)}{=} y_i^{(a)} \cdot (d_i - d_i) = 0,
\end{aligned}$$

where in (a) we used the fact that $y_j^{(a)} = 0$ for all $j \notin \mathcal{V}_a$, while in (b) that all the connections each node has are within the same connected component. Given their definition, the indicator vectors are orthogonal, *i.e.* $(\boldsymbol{y}^{(a)})^T \boldsymbol{y}^{(b)} = \delta_{ab}$ and thus are different eigenvectors of L, concluding the proof. □

Finally, we provide a result on the *largest* eigenvalue of L that will be useful for the next sections.

Lemma 4.2. *Letting $d_{\max}$ be the largest degree of a graph, The spectral radius of L satisfies the following inequality.*

$$\rho(L) \leq 2d_{\max}.$$

Proof. Let $\boldsymbol{x}$ be the eigenvector associated with the largest eigenvalue of L, so that $L\boldsymbol{x} = \rho(L)\boldsymbol{x}$. Consider the index i satisfying $|x_i| \geq |x_j|$ for all j. Then we can write

$$\begin{aligned}
|\rho(L)x_i| &= |(L\boldsymbol{x})_i| \\
&= \left| \sum_{j\in\mathcal{V}} L_{ij} x_j \right| \\
&\overset{(a)}{\leq} \sum_{j\in\mathcal{V}} |L_{ij}||x_j| \\
&\overset{(b)}{\leq} |x_i| \sum_{j\in\mathcal{V}} |L_{ij}| \\
&= |x_i| 2d_i \\
&\leq |x_i| 2d_{\max},
\end{aligned}$$

where in (a) we used the triangle inequality and in (b) we exploited the property of the index i. □

As one can see from the proof, this is not a very tight bound and better results exist. Yet, this is a simple bound we can find without the need to explicitly compute the largest eigenvalue. We now proceed in our discussion with some more practical applications of the GFT to graph signal processing.

4.3 GRAPH SIGNAL PROCESSING

4.3.1 TIKHONOV REGULARIZATION

We formulate here a semi-supervised learning problem of on a graph that we solve exploiting the concept of GFT. We suppose that there is a signal $\boldsymbol{u} \in \mathbb{R}^n$ defined on the nodes of a graph but only some entries of this signal are known. In particular $\mathcal{Q}$ is the set of measured nodes meaning that we know u_i for all $i \in \mathcal{Q}$ and our problem is to guess u_i for all $i \in \mathcal{V} \setminus \mathcal{Q}$. We attempt to reconstruct the signal on the whole graph by identifying a vector $\boldsymbol{v} \in \mathbb{R}^n$ that is at the same time close to $\boldsymbol{u}$ for all $i \in \mathcal{Q}$ and that is smooth on the graph.[1] For all $i \notin \mathcal{Q}$ we hence make a sort of interpolation with the known signal values. We let $\tilde{\boldsymbol{v}}, \tilde{\boldsymbol{u}} \in \mathbb{R}^{|\mathcal{Q}|}$ be two vectors defined only on the set of labeled vertices of $\mathcal{G}(\mathcal{V}, \mathcal{E})$and we define the following loss function

$$\mathcal{L}(\boldsymbol{v}, \boldsymbol{u}) = \|\tilde{\boldsymbol{u}} - \tilde{\boldsymbol{v}}\|^2 + \gamma \boldsymbol{v}^T L \boldsymbol{v}.$$

The first term simply imposes that the loss is minimized when $\boldsymbol{v}$ is equal to $\boldsymbol{u}$ for all $i \in \mathcal{V}$. The second term instead represents instead the "frequency" of $\boldsymbol{v}$ that we want to minimize to ensure the signal changes smoothly over the graph. The factor γ is a weight that balances these two terms: large γ will enforce high regularization, while small γ will tend to force a matching result on the labeled nodes. To explicitly formulate the optimization problem, we let $Q \in \mathbb{R}^{n \times n}$ be a matrix defined as follows

$$Q_{ij} = \delta_{ij} \mathbb{1}_{i \in \mathcal{Q}}. \tag{4.8}$$

The reconstructed signal is then $\boldsymbol{u}^*$, the solution to Tikhonov regularization.

Tikhonov regularization

Tikhonov regularization

Consider a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ with Laplacian matrix L; a matrix Q as per Equation 4.8; a signal $\boldsymbol{u} \in \mathbb{R}^n$ defined on $\mathcal{V}$; and a positive scalar γ. We define $\boldsymbol{u}^*$ the solution of Tikhonov regularization as

$$\boldsymbol{u}^* = \underset{\boldsymbol{v} \in \mathbb{R}^n}{\arg\min}\ (\boldsymbol{u} - \boldsymbol{v})^T Q (\boldsymbol{u} - \boldsymbol{v}) + \gamma \boldsymbol{v}^T L \boldsymbol{v}. \tag{4.9}$$

1 For instance, the values u_i can be the temperatures measured by weather stations at different locations. If we want to have a guess of the temperature in places where no station is available, we may solve this problem on a spatial graph in which each node corresponds to a point on the Earth and edges connect nodes with a distance below a given threshold.

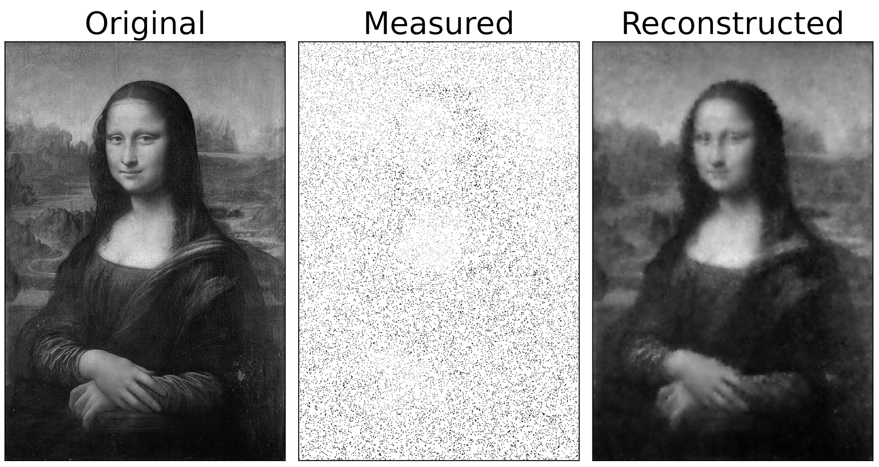

Figure 4.3: **Tikhonov regularization for image reconstruction**. Left: the original black and white image. Center: the measured image in which only 10% of the pixels are known. Right: The reconstructed image using Tikhonov regularization for $\gamma = 0.1$.

The optimization of Equation (4.9) can be solved analytically, in fact,

$$\nabla \boldsymbol{v} \mathcal{L}(\boldsymbol{u}, \boldsymbol{v}) = -2Q(\boldsymbol{u} - \boldsymbol{v}) + 2\gamma L \boldsymbol{v}.$$

Setting $\nabla_{\boldsymbol{v}} = 0$, we get[2]

$$\boldsymbol{u}^* = (Q + \gamma L)^{-1} Q \boldsymbol{u}.$$

In Figure 4.3 we show an example of application of this algorithm to reconstruct an image given that only some pixels are known. We naturally define a grid graph that connects each pixel to its neighbors and show the input image against the reconstructed one. This example clearly shows the power of Tikhonov regularization.

4.3.2 FILTERING

We now address a closely related problem to the one above, that is filtering. Suppose that the signal $\boldsymbol{u} \in \mathbb{R}^n$ is observed on all the graph vertices but that there is some noise. Our goal is to obtain a cleaner version of the signal, removing the noise by exploiting the fact that the signal changes smoothly over the graph, while noise does not. Equation (4.9) can still be used to achieve this task letting $Q = I_n$. Let us look more closely at the solution of the smoothed vector $\boldsymbol{u}^*$

$$\begin{aligned}\boldsymbol{u}^* &= (I + \gamma L)^{-1} \boldsymbol{u} \\ &= \sum_{k=1}^{n} \frac{1}{1 + \gamma \lambda_k} \boldsymbol{x}_k \boldsymbol{x}_k^T \boldsymbol{u}\end{aligned}$$

2 Note that inverting the matrix $Q + \gamma L$ is not computationally efficient nor necessary and the problem can be solved in a faster way finding the solution to $(Q + \gamma L)\boldsymbol{u}^* = Q\boldsymbol{u}$ with an appropriate solver.

$$= \sum_{k=1}^{n} \underbrace{\boldsymbol{x}_k}_{\text{anti-transform}} \underbrace{\frac{1}{1+\gamma\lambda_k}}_{\text{filter in frequency demain}} \underbrace{\hat{u}_k}_{\text{Fourier transform}} .$$

The order in which we should see the operations is from right to left. First we project the signal $\boldsymbol{u}$ on the Laplacian eigenvectors, thus moving to the Fourier space. Here we re-weight every mode with the function $f(x) = (1+\gamma x)^{-1}$. This is a low-pass filter because the smallest eigenvalue $\lambda_1 = 0$ gets a weight equal to one and it corresponds to the eigen-mode with smallest frequency. As we consider larger values of k (hence larger frequencies), $f(\lambda_k)$ decreases, thus filtering out the contribution of high frequency states. Finally, we make the anti-transform and get back to the original space.

Now that we have introduced the concept of filtering, we can design any filter that acts in the frequency domain so to get a good result. Considering a general filtering function f, we can write

Filtering in the Fourier space

$$\boldsymbol{u}^* = \sum_{k=1}^{n} \boldsymbol{x}_k f(\lambda_k)\hat{u}_k. \tag{4.10}$$

A relevant problem is that to solve exactly this problem, we must compute *all* the eigenvalues of L which requires $\mathcal{O}(n^3)$ operations and is unfeasible for large networks. If we use a polynomial filter (or the polynomial approximation of the filtering function), however, we can greatly simplify things. Suppose that

Polynomial filter

$$f(x) = \sum_{a=0}^{p} \alpha_a x^a,$$

then we can rewrite Equation (4.10) as

$$\begin{aligned}
\boldsymbol{u}^* &= \sum_{k=1}^{n}\sum_{a=1}^{p} \boldsymbol{x}_k a_p \lambda_k^p \hat{u}_k \\
&\overset{(a)}{=} \sum_{k=1}^{n}\sum_{a=1}^{p} a_p L^p \boldsymbol{x}_k \hat{u}_k \\
&\overset{(b)}{=} \sum_{k=1}^{n}\sum_{a=1}^{p} a_p L^p \boldsymbol{x}_k \boldsymbol{x}_k^T \boldsymbol{u} \\
&\overset{(c)}{=} \sum_{a=1}^{p} a_p L^p \boldsymbol{u},
\end{aligned}$$

where in (a) we exploited the fact that $\boldsymbol{x}_k$ is an eigenvector of L^p with eigenvalue λ_k^p and in (b) we explicitly rewrote $\hat{u}_k = \boldsymbol{x}_k^T \boldsymbol{u}$ and in (c) we used $I_n = \sum_{k=1}^{n} \boldsymbol{x}_k \boldsymbol{x}_k^T$. From the last equation we can see that the smoothed function can be computed using a polynomial of L, without the need of diagonalizing the matrix. One can still argue, however, that for large p, the matrix L^p is quite dense and thus the computational complexity to perform this operation is still very high. What has to be noticed is that we do not need to compute L^p but actually only $L^p\boldsymbol{x}$ and this can be done efficiently.

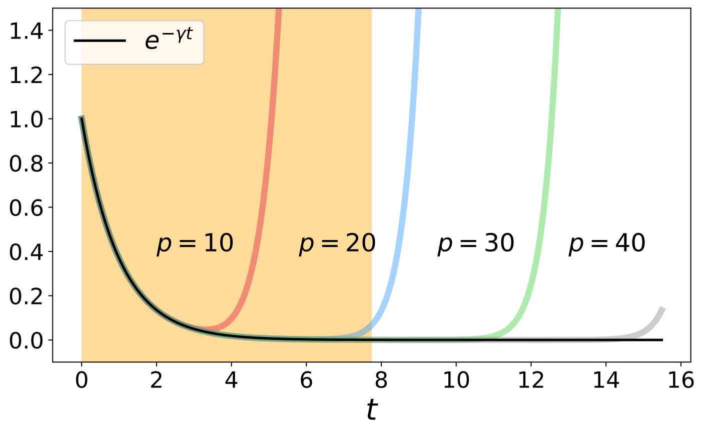

Figure 4.4: **Polinomial approximation of the exponential decay**. The exponential decay function is shown in black for $\gamma = 1$, while in color we have the p order polynomial approximation for different values of p. The orange background indicates the values of t for which the approximation must not diverge and it spans from 0 to $\gamma\rho(L)$.

In fact, let $y_p = L^p \boldsymbol{x}$, then $L^{p+1}\boldsymbol{x} = L\boldsymbol{y}_p$. This is the product between a (presumably sparse) matrix and e vector. Iterating this for all p, we see that the calculation can be performed very efficiently.

We now conclude with an example. We consider the NYC taxi dataset containing several information about taxi drives, including the region (of NYC) of origin and destination and fare of the ride. These data can be found here. The city is divided in approximately 300 regions and we compute the average fare of the destination for each origin. This signal signal is strongly auto-correlated, meaning that nearby regions have similar average fares to be paid. We now introduce some noise in this dataset, taking a random sample including 25% of the total number of regions and randomly reassigning them the average fare amount. Our goal is to reconstruct the original signal by filtering out the high frequency components introduced by noise and exploiting geometrical proximity to smooth the signal.

From the dataset we can extract the coordinates of the centroid of each region and use that to build a graph. We generate it using a fast k nearest neighbors algorithm that connects each node i to its k closest nodes. Note that the matrix we get in this way is not symmetric: Palermo and Roma are (probably) the closest provinces to Sassari, but the opposite is not true. We then must first symmetrize the adjacency matrix to get the correct representation of our graph. We then design an exponential filter $e^{-\gamma x}$, exploiting its polynomial approximation, *i.e.*

$$f(x) = \sum_{a=1}^{p} \frac{(-\gamma x)^a}{a!},$$

for some p. As we know, the modulus of a polynomial function tends to infinity for $x \to \infty$ and is thus not a good approximation of the exponential function. Yet, the argument of the function is bounded by $\gamma\rho(L)$ and

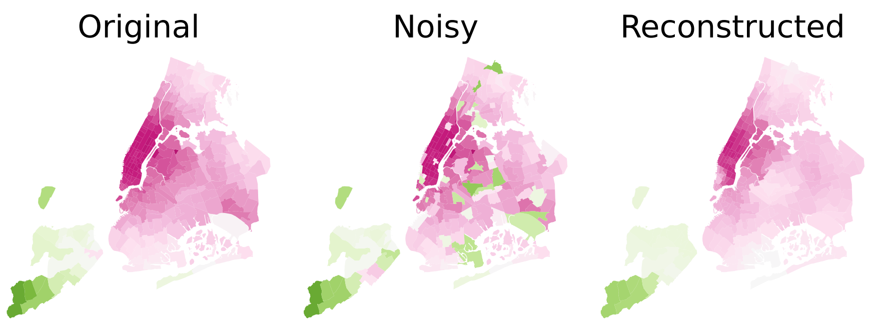

Figure 4.5: **Signal filtering of taxi fares in New York**. Left panel: original signal of the average fare of the region of destination. Center panel: noisy version of the signal. Right panel: smoothed signal using the exponential filter and $p = 30$.

we must choose the order of the polynomial p so that is well approximates the exponential function for all $x \in [0, \gamma\rho(L)]$. Note that, from 4.2 we do not need to explicitly compute $\rho(L)$ as we have a bound as per Lemma 4.2. Figure 4.4 shows the function $f(x)$ for different values of p: the colored background denotes the region in which the filter must well approximate the exponential, hence $p = 30$ is a reasonable choice to proceed. Figure 4.5 shows in color code the result of this experiment.

4.4 CONCLUSION

In this chapter we introduced the concept of Fourier transform on graphs and showcased to example os application to signal reconstruction and denoising. The GFT is an important tool because it allows one to introduce the concept of "frequency" of a signal on a graph *i.e.*, of how fast it changes of the neighbors. Even if we did not discuss it, the relevance of the GFT goes well beyond these two examples and is at the basis of the convolutional graph neural networks. Note that the convolution of two function f, g can be written exploiting their Fourier transforms $\hat{f}, \hat{g}$ as

$$(f * g)(x) = \frac{1}{2\pi} \int_{\mathbb{R}} dk\, \hat{f}(k)\hat{g}(k)e^{ikx},$$

i.e. the Fourier transform of the convolution of two functions if the product of their Fourier transforms. Consequently, this allows one to define the convolution on a graph exploiting the definition of GFT and, in fact, the graph Laplacian matrix is a fundamental building block of convolutional neural networks. As a final remark, we would like to stress that, even though we only mentioned the relation between the matrix L and the GFT, other matrices can similarly be used to define the concept of Fourier transform on graphs. One of the most commonly adopted ones is the *normalized Laplacian*

$$L_{\mathrm{n}} = I_n - D^{-1/2}AD^{-1/2}. \qquad (4.11)$$

4.5 REFERENCES

- Ricaud, Borgnat, Tremblay, Gonçalves, Vandergheynst: *Fourier could be a data scientist: From graph Fourier transform to signal processing on graphs*
  This is a quite pedagogical review on GFT.
- Tremblay, Gonçalves, Borgnat: *Design of graph filters and filterbanks.*
  This is a more advanced reference with the focus on graph filters.