Graph Signal Processing - Part II: Processing and Analyzing Signals on Graphs

McGill

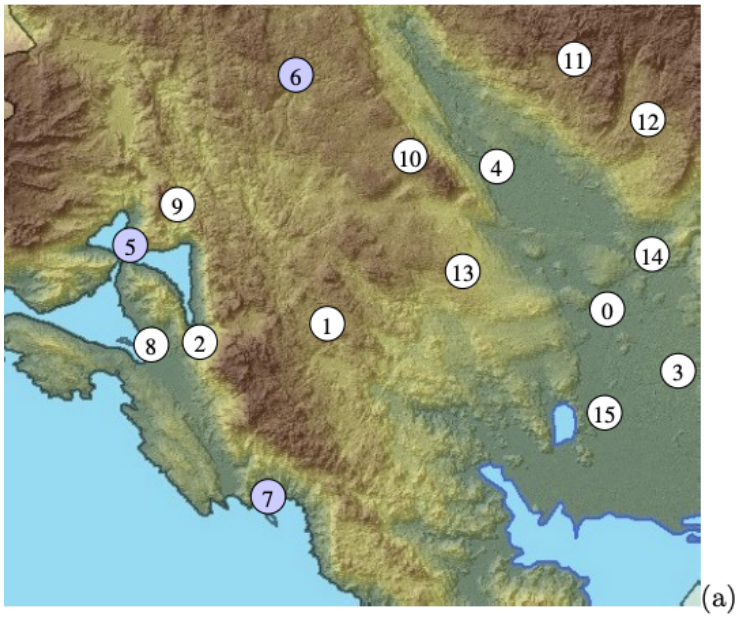
Jiaqi Zhu

- Motivation: Case Study

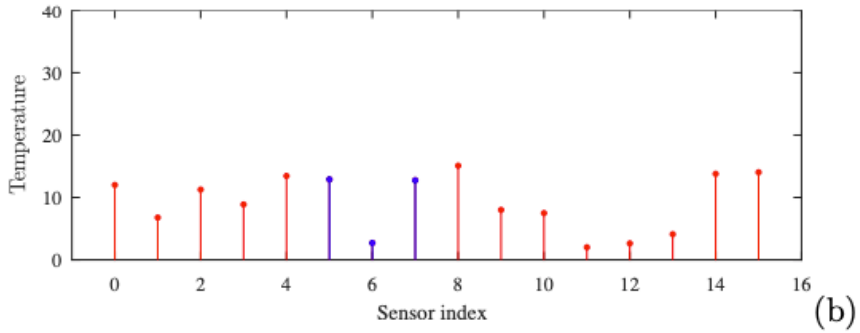
Motivation: Case Study

Problem Setup

- Consider a multi-sensor setup for measuring a temperature field in a region of interest.
- The temperature sensing locations are chosen according to the significance of a particular geographic area to local users (with N = 16 sensing points in total, as shown in figure (a))
- The temperature field is denoted by {x(n)}, with n as the sensor index, while a snapshot of its values is given in figure (b).
- Each measured sensor signal can then be mathematically expressed as $x(n) = s(n) + \varepsilon(n)$, $n = 0, 1, \ldots, 15$, where s(n) is the true temperature that would have been obtained in ideal measuring conditions, and ε(n) is the noise, where $\varepsilon(n) \in N(0, 4)$. This yields the signal-to-noise ratio in in x(n) of $SNR_{in} = 14.2$ dB



(a)



(b)

Motivation: Case Study

Goal:

- Denoise and increase SNR
  - **5 dB to 10 dB**: is below the minimum level to establish a connection, due to the noise level being nearly indistinguishable from the desired signal (useful information).
  - **10 dB to 15 dB**: is the accepted minimum to establish an unreliable connection.
  - **15 dB to 25 dB**: is typically considered the minimally acceptable level to establish poor connectivity.
  - **25 dB to 40 dB**: is deemed to be good.
  - **41 dB or higher**: is considered to be excellent.

(https://resources.pcb.cadence.com/blog/2020-what-is-signal-to-noise-ratio-and-how-to-calculate-it)

Motivation: Case Study

Noise Reduction Scheme:

- "Situation-Aware": local signal average operator

Cumulative Temperature:

$$y(n) = \sum_{m \text{ at and around } n} x(m)$$

Example:
n = 3 (low land)
n = 6(mountains)
n = 8 (coast)

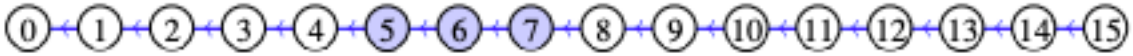
y(3) = x(3) + x(0) + x(14) + x(15)
y(6) = x(6) + x(9) + x(10)

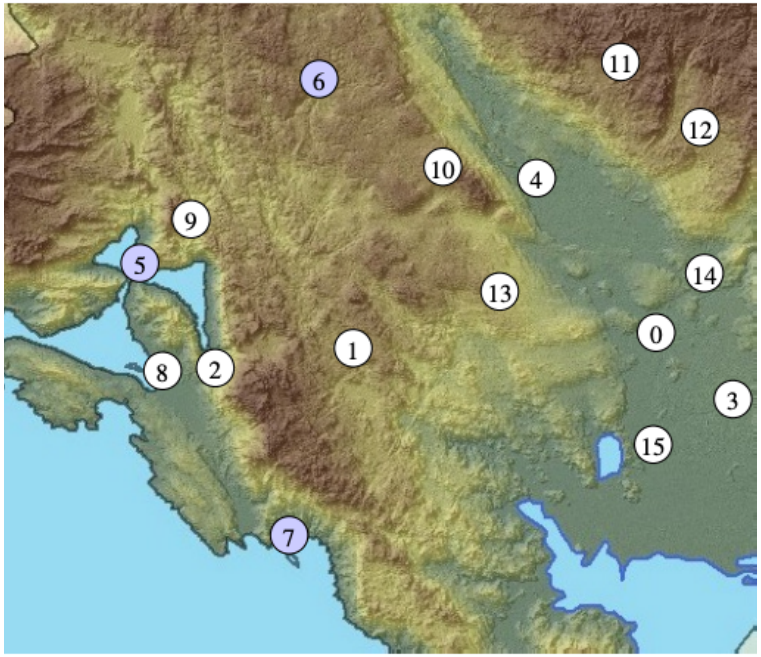
y = x + Ax
A is the *adjacency matrix*

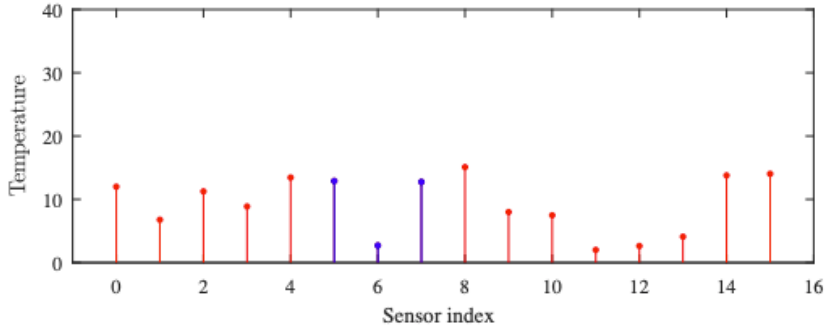
Motivation: Case Study

Classical Representation

- Linear structure







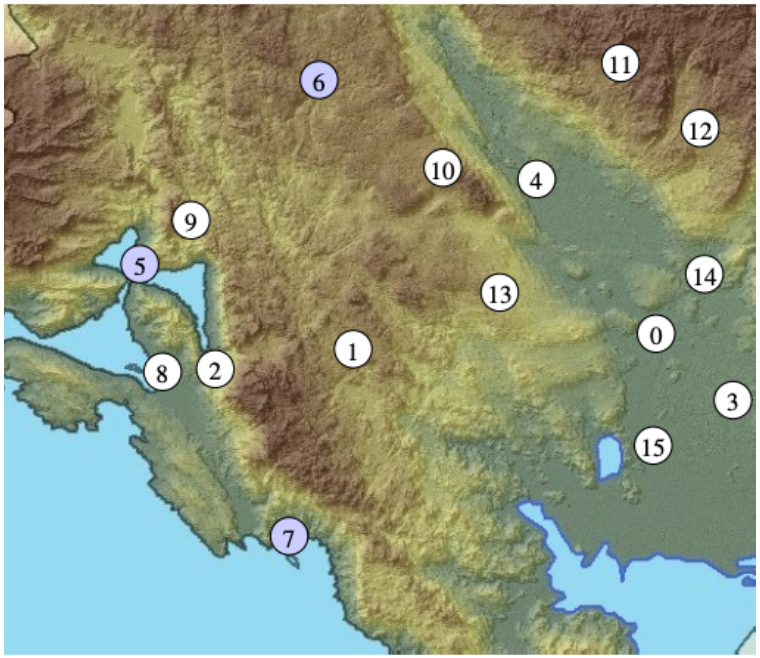
Motivation: Case Study

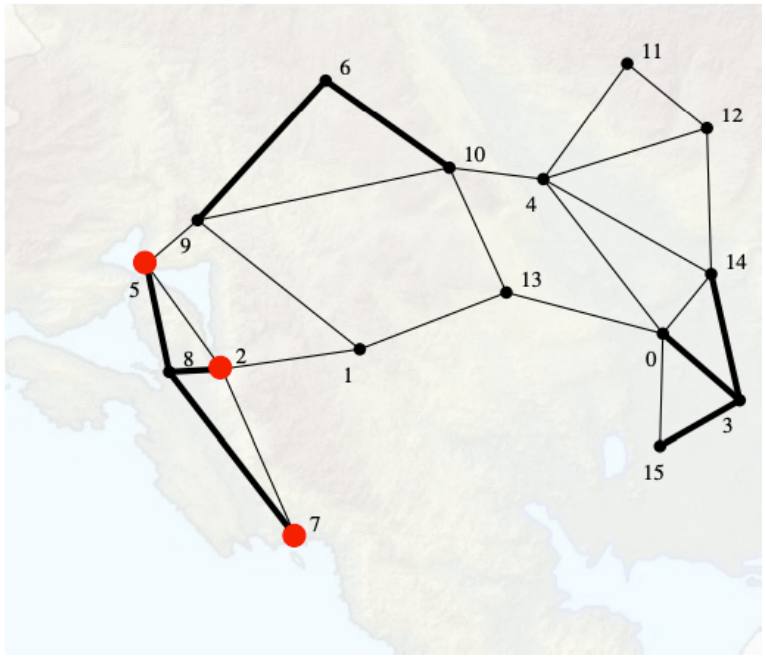
Graph Representation:

- Construct a graph
  - Vertices: sensing points where the signal is measured
  - Edges: Vertex-to-vertex lines which indicate physically meaningful connectivity among the sensing points

  We will use graph (rather than a standard vector of sensing points) to analyze and process data, since it exhibits both spatial and physical domain awareness

Motivation: Case Study

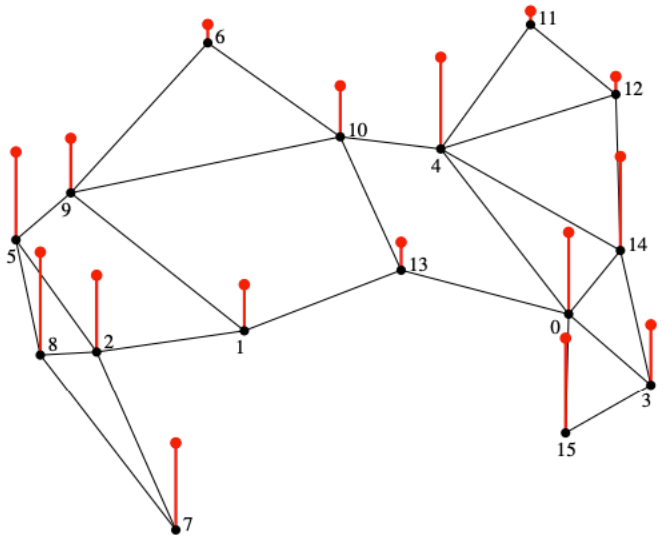


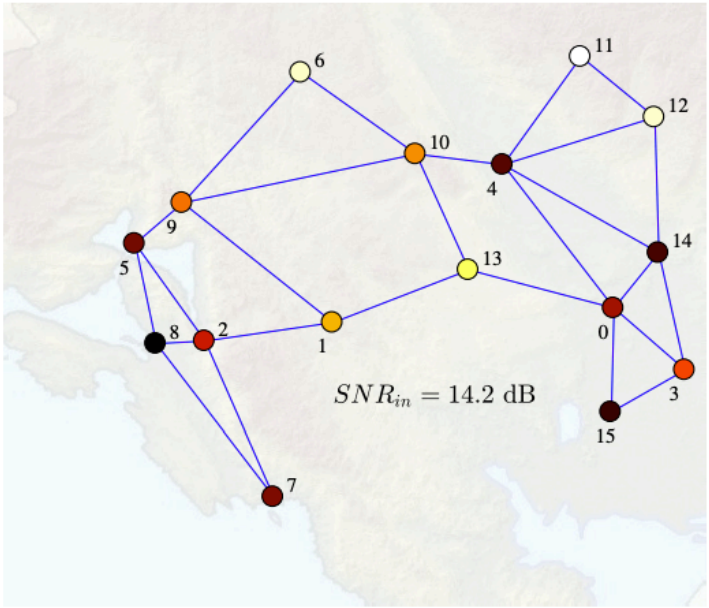


Motivation: Case Study

Graph Representation:

- Signal on Graph
  - The measured temperatures are now interpreted as signal samples on graph
  - Similar to traditional signal processing, this new graph signal may have many realizations on the same graph and may comprise noise





Motivation: Case Study

Graph Representation:

- System on Graph
  - $\mathbf{y} = \mathbf{x} + \mathbf{A}\mathbf{x}$ defines a simple system on a graph for local signal averaging
  - To model sensor relevance, we can also adapt a weighting scheme

$$y(n) = x(n) + \sum_{m \neq n} W_{nm} x(m)$$

where W_{nm} are the elements of the weighting matrix, W

  - Three ways to define graph edges and their corresponding weights (will be introduced in the next series of lectures)
    - Already physically well defined edges and weights
    - **Definition of edges and weights based on the geometry of vertex positions**
    - Data similarity based methods for learning the underlying graph topology

$$W_{mn} = e^{-\alpha r_{mn} - \beta h_{mn}}$$

where α and β are suitable constants.

Motivation: Case Study

Graph Representation:

- System on Graph
  - Weighted graph signal estimator

$$\mathbf{y} = \mathbf{x} + \mathbf{W}\mathbf{x}$$

  - Normalized estimator: weighting coefficients sum to 1 for each y(n)

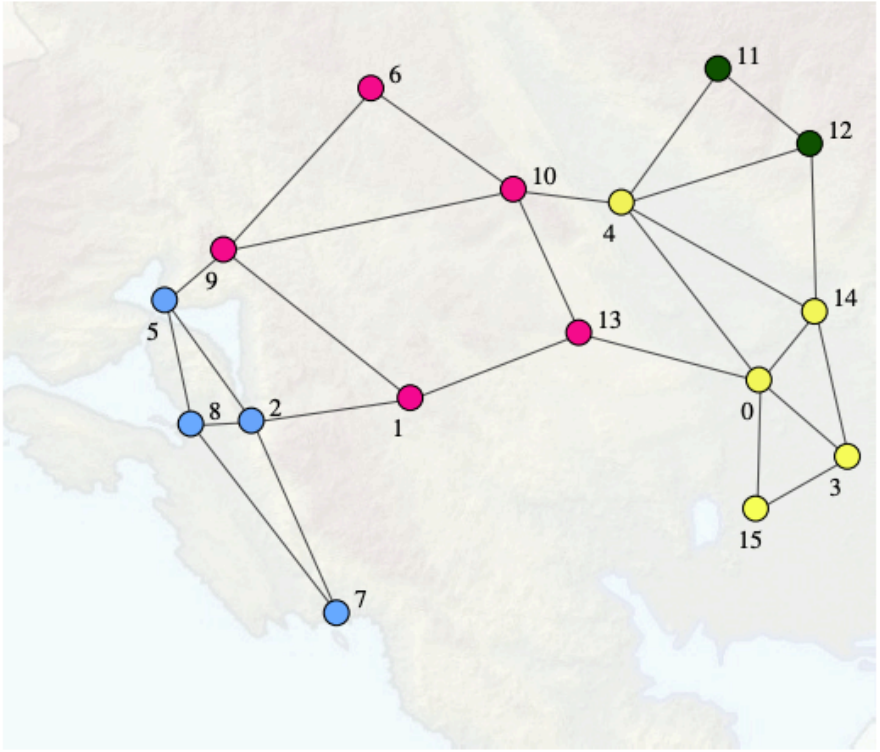
$$\mathbf{y} = \frac{1}{2}(\mathbf{x} + \mathbf{D}^{-1}\mathbf{W}\mathbf{x})$$

where D is the degree matrix, $D_{nn} = \sum_m W_{nm}$

Motivation: Case Study

Graph Representation:

- Vertices Clustering



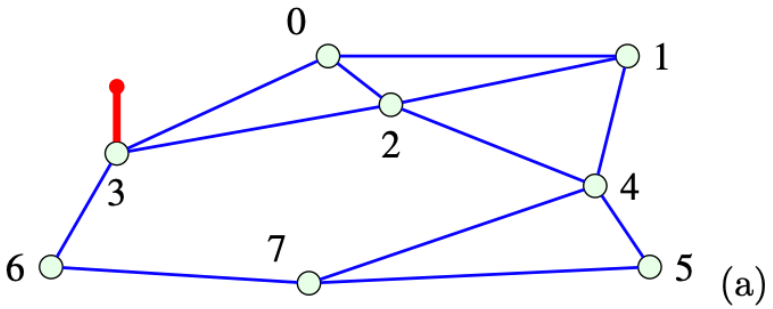
- Signals and Systems on Graphs

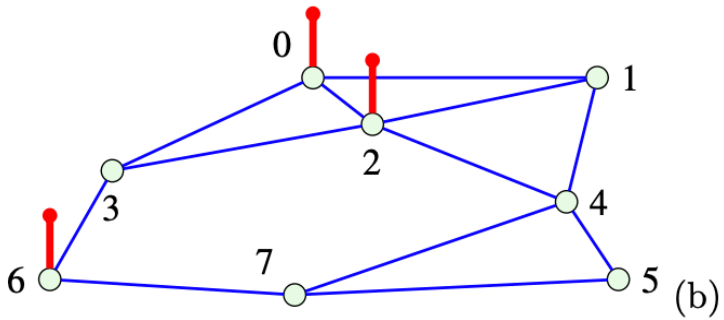
Signals and Systems on Graphs

- Basic Concepts:
  - Signals on Graphs:

$$\mathbf{x} = [x(0), x(1), \ldots, x(N-1)]^T$$

  - Graph Signal Shift:
    - Walk with length 1: $\mathbf{x}_1 = \mathbf{A}\mathbf{x}$
    - Walk with length 2: $\mathbf{x}_2 = \mathbf{A}\mathbf{x}_1 = \mathbf{A}(\mathbf{A}\,\mathbf{x}) = \mathbf{A}^2\,\mathbf{x}$
    - Walk with length m: $\mathbf{x}_m = \mathbf{A}\mathbf{x}_{m-1} = \mathbf{A}^m\,\mathbf{x}$
  - Graph shift operator: a local operator that replaces a signal value at each node of a graph with a linear combinations of the signal values at the neighborhood of that value.





Signals and Systems on Graphs

- Basic Concepts
  - Systems on Graphs with Shift Operator as Adjacency Matrix:

$$\mathbf{y} = h_0\mathbf{A}^0\mathbf{x} + h_1\mathbf{A}^1\mathbf{x} + \cdots + h_{M-1}\mathbf{A}^{M-1}\mathbf{x} = \sum_{m=0}^{M-1} h_m\mathbf{A}^m\mathbf{x}$$

where $\mathbf{A}^0 = \mathbf{I}$, by definition, and $h_0, h_1, \ldots, h_{M-1}$ are the system coefficients.

    - Remark: a physically meaningful system order $(M-1)$ should be lower than the number of vertices N, that is is, $M \leq N$
  - General System of Graphs (in vertex domain):

$$\mathbf{y} = H(\mathbf{A})\mathbf{x}$$

where H(A) is a vertex domain system (filter) function

    - Linearity: $H(\mathbf{A})(a_1\mathbf{x}_1 + a_2\mathbf{x}_2) = a_1\mathbf{y}_1 + a_2\mathbf{y}_2$
    - Shift Invariance: $H(\mathbf{A})[\mathbf{A}\mathbf{x}] = \mathbf{A}[H(\mathbf{A})\mathbf{x}] = \mathbf{A}\mathbf{y}$

Signals and Systems on Graphs

- Spectral Domain of the Adjacency Matrix
  - Graph discrete Fourier $\mathbf{X} = \mathbf{U}^{-1}\mathbf{x}$ DFT)

    where X denotes a vector of the GDFT coefficients, and U is a matrix whose columns represent the eigenvectors of the adjacency matrix

    - For undirected g

$$X(k) = \sum_{n=0}^{N-1} x(n) u_k(n)$$

  - Inverse graph discrete Fourier transform (IGDFT)

$$\mathbf{x} = \mathbf{U}\,\mathbf{X}$$

$$x(n) = \sum_{k=0}^{N-1} X(k) u_k(n)$$

Signals and Systems on Graphs

- System on a graph in the GDFT domain

$$\mathbf{y} = h_0\mathbf{A}^0\mathbf{x} + h_1\mathbf{A}^1\mathbf{x} + \cdots + h_{M-1}\mathbf{A}^{M-1}\mathbf{x} = \sum_{m=0}^{M-1} h_m\mathbf{A}^m\mathbf{x}$$

$$\Rightarrow \quad \mathbf{y} = H(\mathbf{A})\mathbf{x} = \left(h_0\mathbf{A}^0 + h_1\mathbf{A}^1 + \cdots + h_{M-1}\mathbf{A}^{M-1}\right)\mathbf{x}$$

$$\Rightarrow \quad \mathbf{y} = \left(h_0\mathbf{U}\boldsymbol{\Lambda}^0\mathbf{U}^{-1} + h_1\mathbf{U}\boldsymbol{\Lambda}^1\mathbf{U}^{-1} + \cdots + h_{M-1}\mathbf{U}\boldsymbol{\Lambda}^{M-1}\mathbf{U}^{-1}\right)\mathbf{x}$$

$$= \mathbf{U}(h_0\boldsymbol{\Lambda}^0 + h_1\boldsymbol{\Lambda}^1 + \cdots + h_{M-1}\boldsymbol{\Lambda}^{M-1})\mathbf{U}^{-1}\mathbf{x}$$

$$= \mathbf{U}\,H(\boldsymbol{\Lambda})\mathbf{U}^{-1}\mathbf{x}$$

with $H(\boldsymbol{\Lambda}) = h_0\boldsymbol{\Lambda}^0 + h_1\boldsymbol{\Lambda}^1 + \cdots + h_{M-1}\boldsymbol{\Lambda}^{M-1}$

$$\Rightarrow \quad \mathbf{U}^{-1}\mathbf{y} = H(\boldsymbol{\Lambda})\mathbf{U}^{-1}\mathbf{x} \quad \Leftrightarrow \quad \mathbf{Y} = H(\boldsymbol{\Lambda})\,\mathbf{X}$$

$$Y(k) = (h_0 + h_1\lambda_k + \cdots + h_{M-1}\lambda_k^{M-1})X(k)$$

The output graph signal in the vertex domain can then be calculated as

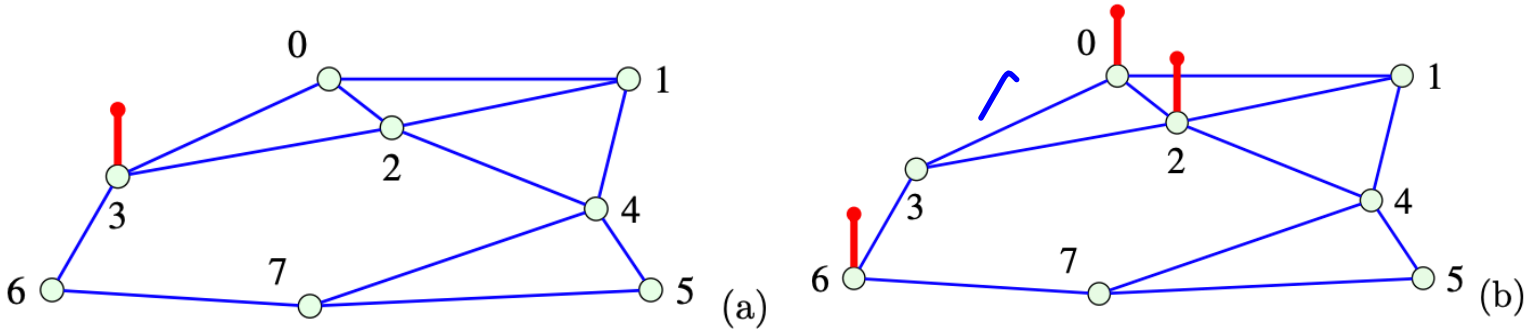
$$\mathbf{y} = H(\mathbf{A})\mathbf{x} = \mathrm{IGDFT}\{H(\boldsymbol{\Lambda})\,\mathbf{X}\}$$

Transfer function of a system on a graph:

$$H(\lambda_k) = \frac{Y(k)}{X(k)} = h_0 + h_1\lambda_k + \cdots + h_{M-1}\lambda_k^{M-1}$$

Signals and Systems on Graphs

- Graph signal filtering
  - Energy of a graph shift $\|\mathbf{x}_1\|_2^2 = \|\mathbf{A}\mathbf{x}\|_2^2$



In general $\|\mathbf{A}\mathbf{x}\|_2^2 \neq \|\mathbf{x}\|_2^2$

$$\max\{\frac{\|\mathbf{A}\mathbf{x}\|_2^2}{\|\mathbf{x}\|_2^2}\} = \max\{\frac{\mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x}}{\|\mathbf{x}\|_2^2}\} = \lambda_{\max}^2 \text{, where } \lambda_{\max} = \max_k |\lambda_k|,\ k = 0, 1, \ldots, N-1.$$

Signals and Systems on Graphs

- Graph signal filtering
  - Normalization of the Adjacency of Matrix

$$\mathbf{A}_{norm} = \frac{1}{\lambda_{\max}}\mathbf{A} \quad \Longrightarrow \quad \|\mathbf{A}_{norm}\mathbf{x}\|_2^2 \leq \|\mathbf{x}\|_2^2$$

  - Spectral Ordering: low-varying and high-varying eigenvectors
    - First graph difference

$$\boldsymbol{\Delta}\mathbf{x} = \mathbf{x} - \mathbf{x}_1 = \mathbf{x} - \mathbf{A}_{norm}\mathbf{x}.$$

    - Energy of signal change

$$E_{\Delta x} = \|\mathbf{x} - \mathbf{A}_{norm}\mathbf{x}\|_2^2 = \left\|\mathbf{x} - \frac{1}{\lambda_{\max}}\mathbf{A}\mathbf{x}\right\|_2^2$$

$$\mathrm{x} = \mathrm{u} \quad \Downarrow$$

$$E_{\Delta u} = \left\|\mathbf{u} - \frac{1}{\lambda_{\max}}\lambda\mathbf{u}\right\|_2^2 = \left|1 - \frac{\lambda}{\lambda_{\max}}\right|^2$$

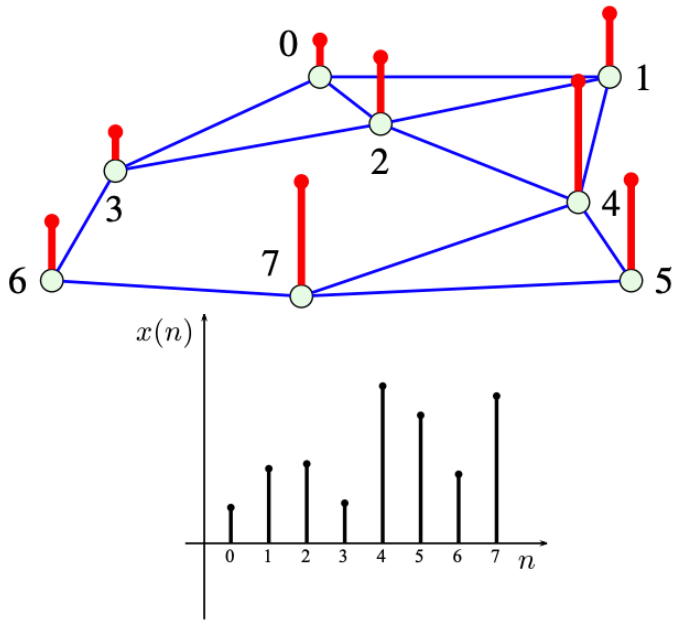
two-norm total variation of a basis function/eigenvector.

The eigenvectors corresponding to large λ_k correspond to the low-pass part of a graph signal

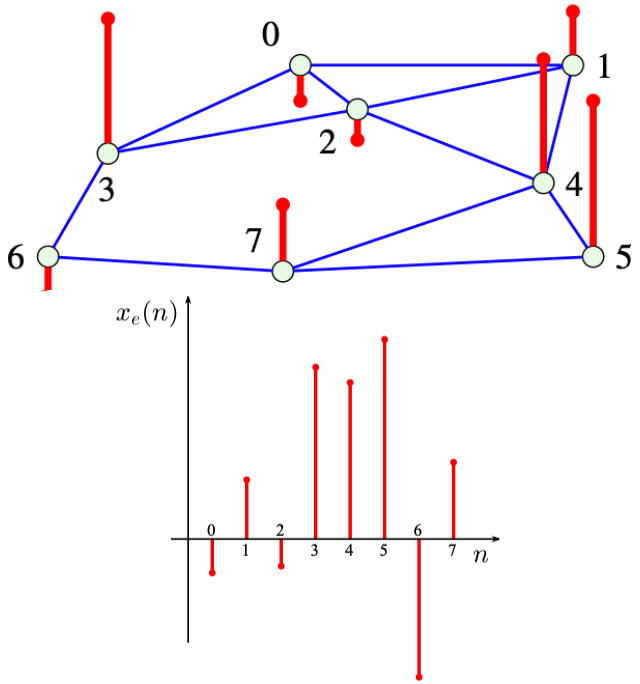
Signals and Systems on Graphs

- Graph signal filtering
  - Ideal low-pass filter on a graph

$$f(\lambda) = \begin{cases} 1, & \text{for } \lambda > \lambda_c, \\ 0, & \text{for other } \lambda. \end{cases}$$



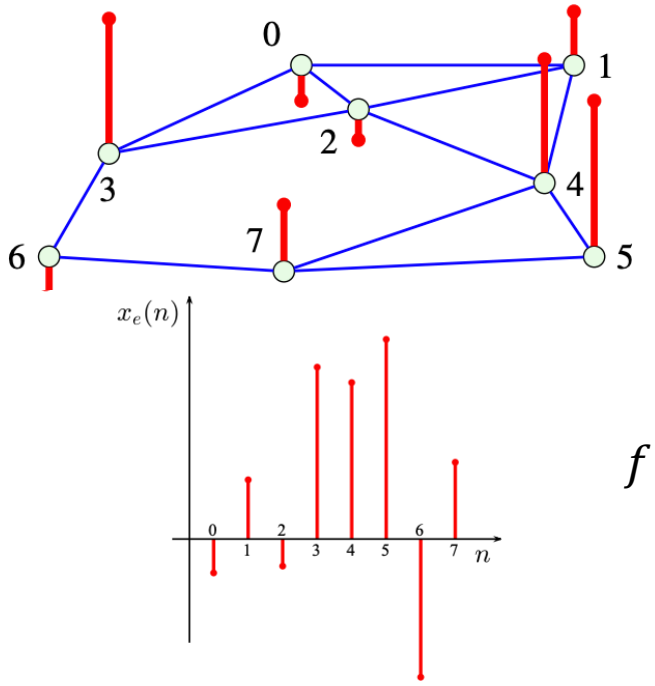
(a) original signal, $\mathbf{x} = 3.2\mathbf{u}_7 + 2\mathbf{u}_6$



(b) noisy signal, $\mathbf{x}_\varepsilon = \mathbf{x} + \boldsymbol{\varepsilon}$

Signals and Systems on Graphs

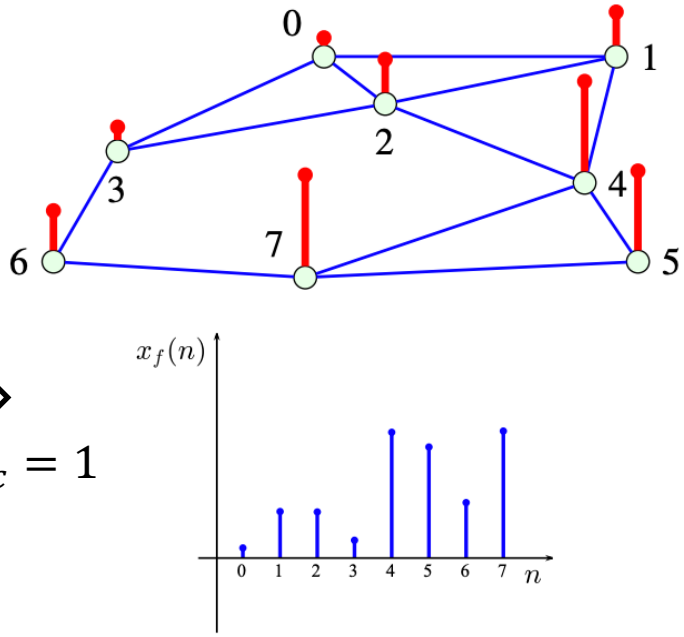
$SNR_{in} = 2.7dB$



(b) noisy signal, $\mathbf{x}_\varepsilon = \mathbf{x} + \boldsymbol{\varepsilon}$

$f(\lambda)\ with\ \lambda_c = 1$

$SNR_{in} = 18.8dB$



(c) filtered signal

Signals and Systems on Graphs

- Spectral domain filter design
  - Let $G(\Lambda)$ denote desired graph transfer function of a system defined on a graph. A system with this transfer function can be implemented either in the spectral domain or in the vertex domain
  - Spec

1. Calculate the GDFT of the input graph signal, $\mathbf{X} = \mathbf{U}^{-1}\mathbf{x}$,
2. Multiply the GDFT of the input graph signal by the graph transfer function, $G(\mathbf{\Lambda})$, to obtain the output spectral form, $\mathbf{Y} = G(\mathbf{\Lambda})\mathbf{X}$, and
3. Calculate the output graph signal as the inverse GDFT of $\mathbf{Y}$ in Step 2, that is, $\mathbf{y} = \mathbf{U}\mathbf{Y}$.

    - Computationally demanding for large graphs
    - More convenient to implement the desired filter (or its close approximation) directly in the vertex domain

Signals and Systems on Graphs

- Spectral domain filter design
  - Let $G(\Lambda)$ denote desired graph transfer function of a system defined on a graph. A system with this transfer function can be implemented either in the spectral domain or in the vertex domain
  - Vertex domain implementation: find the coefficients (cf. standard impulse respo

$$\mathbf{y} = h_0\mathbf{A}^0\mathbf{x} + h_1\mathbf{A}^1\mathbf{x} + \cdots + h_{M-1}\mathbf{A}^{M-1}\mathbf{x} = \sum_{m=0}^{M-1} h_m\mathbf{A}^m\mathbf{x}$$

such that their spectral representation, H(Λ), is equal to the G(Λ)

i.e.

$$H(\lambda_k) = \frac{Y(k)}{X(k)} = h_0 + h_1\lambda_k + \cdots + h_{M-1}\lambda_k^{M-1} = G(\lambda_k)$$

for k = 0, 1, . . . , N − 1

Signals and Systems on Graphs

- Spectral domain filter design
  - Vertex domain implementation (cont'd)

$$h_0 + h_1\lambda_0^1 + \cdots h_{M-1}\lambda_0^{M-1} = G(\lambda_0)$$
$$h_0 + h_1\lambda_1^1 + \cdots + h_{M-1}\lambda_1^{M-1} = G(\lambda_1)$$
$$\vdots$$
$$h_0 + h_1\lambda_{N-1}^1 + \cdots + h_{M-1}\lambda_{N-1}^{M-1} = G(\lambda_{N-1})$$

$$\Downarrow$$

$$\mathbf{V}_\lambda\,\mathbf{h} = \mathbf{g}\,(1) \quad \text{with} \quad \mathbf{g} = [G(\lambda_0), G(\lambda_1), \ldots, G(\lambda_{N-1})]^T = \mathrm{diag}(G(\mathbf{\Lambda}))$$

$$\mathbf{h} = [h_0, h_1, \ldots, h_{M-1}]^T$$

$$\mathbf{V}_\lambda = \begin{bmatrix} 1 & \lambda_0^1 & \cdots & \lambda_0^{M-1} \\ 1 & \lambda_1^1 & \cdots & \lambda_1^{M-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \lambda_{N-1}^1 & \cdots & \lambda_{N-1}^{M-1} \end{bmatrix}$$

Signals and Systems on Graphs

- Spectral domain filter design
  - Vertex domain implementation (cont'd)

    Solution of (1):
    - Consider the case with N vertices and with all distinct eigenvalues of the adjacency
      a) If the filter order, M, is such that M = N, then the solution to (1) is unique
      b) If the filter order, M, is such that M < N, then the system in (1) is overdetermined. Therefore, the solution to (1) can only be obtained using least square approximation

Signals and Systems on Graphs

- Spectral domain filter design
  - Vertex domain implementation (cont'd)

    Solution of (1):
    - If some of the eigenvalues are of a degree higher than one, the system in (1) can be reduced into a system of N_m linear equations
      a) If the filter order, M, is such that $N_m < M \leq N$, the system in (1) is underdetermined, $(M - N_m)$ filter coefficients are free variables. The system has an infinite number of solutions
      b) If the filter order is such that $M = N_m$, the solution to the system in (1) is unique
      c) If the filter order is such that M < Nm, the system in (1) is overdetermined and the solution is obtained in the least squares sense

Signals and Systems on Graphs

- Spectral domain filter design
  - Vertex domain implementation (cont'd)
    - Exact Soluti $\mathbf{h} = \mathbf{V}_\lambda^{-1}\mathbf{g}$ r M = N_m)

    - Least-squares solution (M < N_m)

      We want to minimize $e = \|\mathbf{V}_\lambda \mathbf{h} - \mathbf{g}\|_2^2$

      From $\partial e / \partial \mathbf{h}^T = \mathbf{0}$ we then have

      $$\hat{\mathbf{h}} = (\mathbf{V}_\lambda^T \mathbf{V}_\lambda)^{-1} \mathbf{V}_\lambda^T \mathbf{g} = \text{pinv}(\mathbf{V}_\lambda)\mathbf{g}$$

      Since this solution may not satisfy $V_\lambda h = g$, the designed coefficient vector, $\hat{g}$, obtained from

      $$\mathbf{V}_\lambda \hat{\mathbf{h}} = \hat{\mathbf{g}}$$

      in general, differs from the desired system coefficients, g

Signals and Systems on Graphs

- Spectral domain filter design
  - Polynomial approximation of the system on a graph transfer function:
    - Why: Multiplication with the eigenvector matrix U takes $O(N^2)$
    - How
      - Without loss of generality, it can be considered that the desired transfer function, $g = [G(\lambda_0), G(\lambda_1), \ldots, G(\lambda_{N-1})]^T$, consists of samples taken from a continuous function of λ within the interval $\lambda_{min} \leq \lambda \leq \lambda_{max}$, where λ_{min} and λ_{max} denote the minimum and maximum value of $\{\lambda_0, \lambda_1, \ldots, \lambda_{N-1}\}$. The system on graph uses only the values at discrete points $\lambda \in \{\lambda_0, \lambda_1, \ldots, \lambda_{N-1}\}$
      - We find to find an approximating polynomial $P(\lambda)$, which has the smallest maximum absolute error from the desired function value, i.e error at the points within $\lambda_{min} \leq \lambda \leq \lambda_{max}$, is bounded and sufficiently small => minmax approximation

Signals and Systems on Graphs

- Spectral domain filter design
  - Polynomial approximation of the system on a graph transfer function (cont'd)
    - Chebyshe

$$P_{M-1}(z) = \frac{c_0}{2} + \sum_{m=1}^{M-1} c_m T_m(z)$$

Chebyshev polynomials

$$\begin{aligned} T_0(z) &= 1, \\ T_1(z) &= z, \\ T_2(z) &= 2z^2 - 1, \\ T_3(z) &= 4z^3 - 3z, \\ &\vdots \\ T_m(z) &= 2zT_{m-1}(z) - T_{m-2}(z), \end{aligned}$$

Chebyshev coefficients

$$\begin{aligned} c_m &= \frac{2}{\pi} \int_{-1}^{1} G(z) T_m(z) \frac{dz}{\sqrt{1-z^2}} \\ &= \frac{2}{\pi} \int_{0}^{\pi} cos(m\theta) G(cos(\theta)) d\theta. \end{aligned}$$

Normalized Input

$$z = \frac{2\lambda - (\lambda_{\max} + \lambda_{\min})}{\lambda_{\max} - \lambda_{\min}}$$

$$\lambda = \frac{1}{2}\Big(z(\lambda_{\max} - \lambda_{\min}) + \lambda_{\max} + \lambda_{\min}\Big)$$

Signals and Systems on Graphs

- Spectral domain filter design
  - Polynomial approximation of the system on a graph transfer function (cont'd)
    - Graph signal filtering applied in the vertex domain:

$$\mathbf{y} = \bar{P}_{M-1}(\mathbf{A})\mathbf{x}$$

    - Computational complexity:
      - If the number of nonzero elements in the adjacency matrix, A, is N_A, then the number of arithmetic operations (additions) to calculate Ax is of N_A order. The same number of operations is required to calculate $A^2x = A(Ax)$ using the available Ax
      - This means that the total number arithmetic operations (additions) to calculate all Ax, $A^2x,\ldots,A^{M-1}x$ is of order MN_A.
      - Adding these terms requires additional MN_A arithmetic operations (additions)
      - Calculation of all terms of the form c_mA^mx requires an order of MN_A multiplications by constants c_m

    => An order of $2MN_A$ additions and MN_A multiplications is needed

Signals and Systems on Graphs

- Spectral Domain of the Laplacian Matrix
  - Graph discrete Fourier $\mathbf{X} = \mathbf{U}^{-1}\mathbf{x}$ DFT)

    where X denotes a vector of the GDFT coefficients, and U is a matrix whose columns represent the eigenvectors of the Laplacian matrix

    - For undirected g $$X(k) = \sum_{n=0}^{N-1} x(n) u_k(n)$$
  - Inverse graph discrete Fourier transform (IGDFT)

$$\mathbf{x} = \mathbf{U}\,\mathbf{X}$$

$$x(n) = \sum_{k=0}^{N-1} X(k) u_k(n)$$

Signals and Systems on Graphs

- System on a graph in the GDFT domain

$$\mathbf{y} = h_0\mathbf{L}^0\,\mathbf{x} + h_1\mathbf{L}^1\,\mathbf{x} + \cdots + h_{M-1}\mathbf{L}^{M-1}\,\mathbf{x}$$
$$= \sum_{m=0}^{M-1} h_m\mathbf{L}^m\,\mathbf{x}.$$

$$\Rightarrow \quad \mathbf{y} = \mathbf{U}\mathbf{Y} = \sum_{m=0}^{M-1} h_m\mathbf{L}^m\,\mathbf{x} = H(\mathbf{L})\mathbf{x}$$
$$= \mathbf{U}H(\mathbf{\Lambda})\mathbf{U}^T\mathbf{x} = \mathbf{U}H(\mathbf{\Lambda})\mathbf{X}$$

with $H(\mathbf{\Lambda}) = \sum_{m=0}^{M-1} h_m\mathbf{\Lambda}^m$

$$\Rightarrow \quad \mathbf{Y} = H(\mathbf{\Lambda})\mathbf{X} \quad Y(k) = (h_0 + h_1\lambda_k + \cdots + h_{M-1}\lambda_k^{M-1})X(k)$$

Signals and Systems on Graphs

The n^{th} element of the output signal, $y = UH(\Lambda)U^T x$, of a system on a graph is given by

$$y(n) = \sum_{k=0}^{N-1} \sum_{i=0}^{N-1} x(i) u_k(i) H(\lambda_k) u_k(n) = \sum_{i=0}^{N-1} x(i) h_n(i),$$

where $h_n(i) = \sum_{k=0}^{N-1} H(\lambda_k) u_k(n) u_k(i) = \mathcal{T}_n\{h(i)\}$. (graph impulse response)

Transfer function of a system on a graph:

$$H(\lambda_k) = h_0 + h_1 \lambda_k + \cdots + h_{M-1} \lambda_k^{M-1}$$

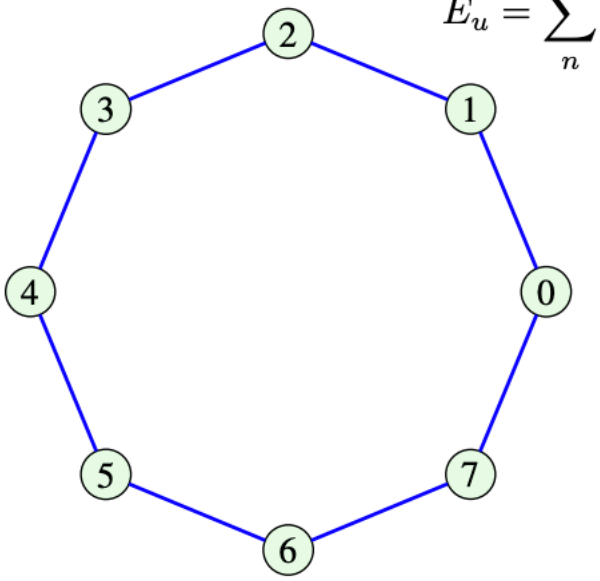
Signals and Systems on Graphs

- Graph signal filtering
  - Spectral Ordering: low-varying and high-varying eigenvectors
    - Second order finite difference on circular graph

$$y(n) = -u(n-1) + 2u(n) - u(n+1) \iff y = \lambda u = Lu$$

    - Energy of signal change on circular graph

$$E_u = \sum_n \left[\left(u(n) - u(n-1) \right)^2 + \left(u(n) - u(n+1) \right)^2 \right] / 2$$



$$\Updownarrow$$

$$E_u = u^T L u$$

$$\Updownarrow$$

$$\mathbf{u}^T \mathbf{L} \mathbf{u} = \lambda \mathbf{u}^T \mathbf{u} = \lambda = E_u$$

Signals and Systems on Graphs

- Graph signal filtering
  - Spectral Ordering (cont'd)

$$\mathbf{u}_k^T \mathbf{L} \mathbf{u}_k = \lambda_k = \frac{1}{2} \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} W_{nm} \Big(u_k(n) - u_k(m) \Big)^2 \geq 0$$

⇩

a small $u_k^T L u_k = \lambda_k$ corresponds to slow eigenvector variation $W_{nm}(u_k(n) - u_k(m))^2$ within the neighboring vertices

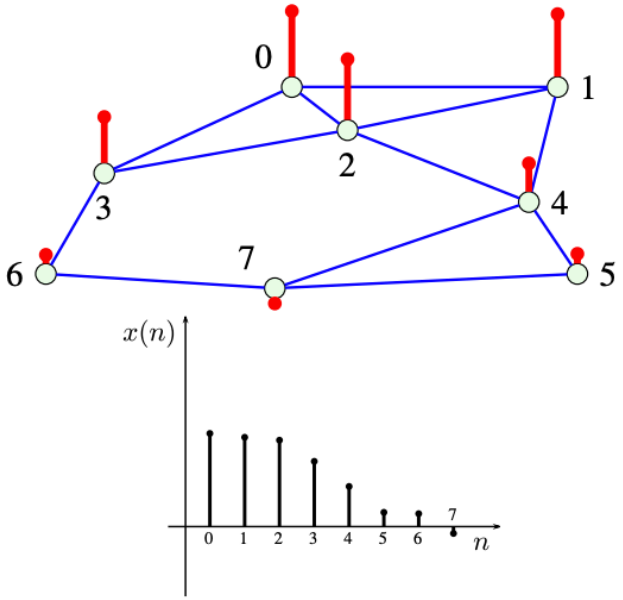
⇩

the eigenvectors corresponding to small λ_k correspond to the low-pass part of a graph signal i.e. smoother.

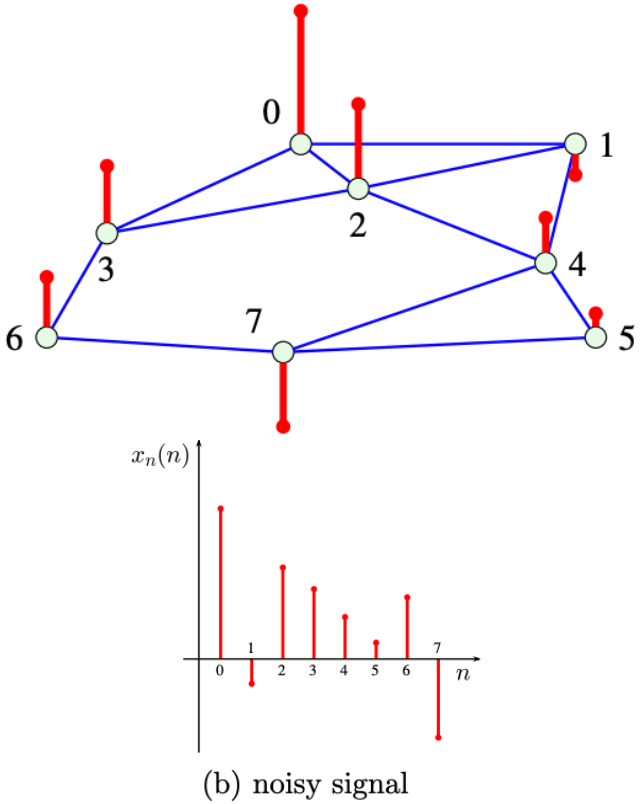
Signals and Systems on Graphs

- Graph signal filtering
  - Ideal low-pass filter on a graph

$$f(\lambda) = \begin{cases} 1, & \text{for } \lambda < \lambda_c \\ 0, & \end{cases}$$



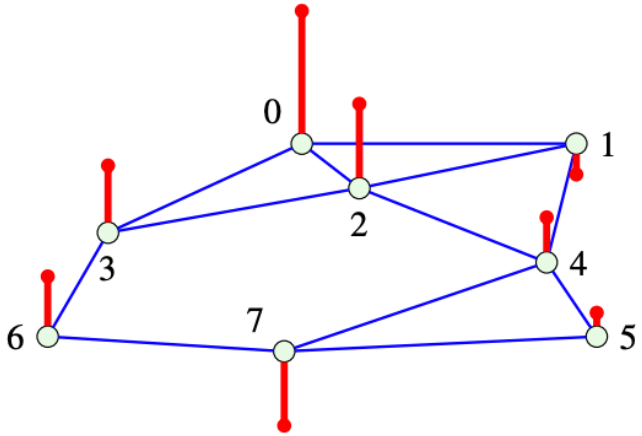
(a) original signal, $x = 2u_0 + 1.5u_1$

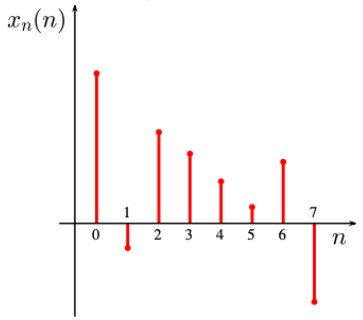


(b) noisy signal

Signals and Systems on Graphs

SNR_{in} = -1.76 dB

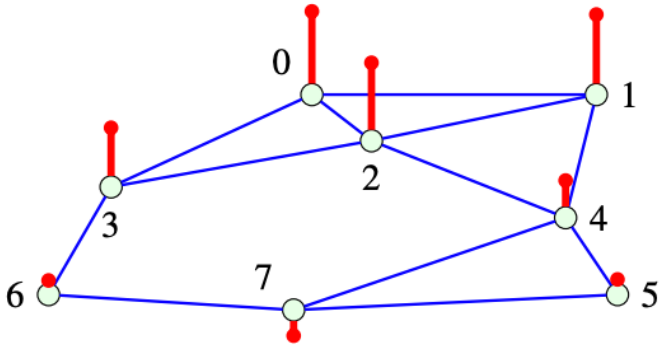


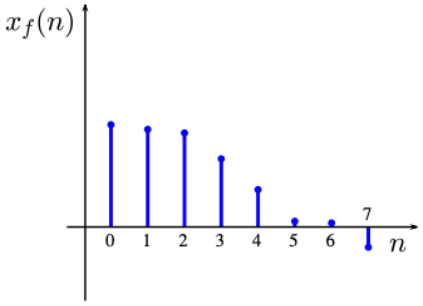


(b) noisy signal

$f(\lambda)$ *with* $\lambda_c = 0.25$

SNR_{in} = 21.29 dB





(c) filtered signal

Signals and Systems on Graphs

- Convolution of signals on a graph
  - Definition: Consider two signals, **x** and **y**, defined on a graph. The corresponding GFTs are denoted by **X** and **Y**. A generalized convolution, **z** , of signals **x** and **y** can then be defined in the GFT domain as

$$z = x * y = IGDFT(Z)$$

where

$$Z = GDFT(x * y) = XY$$

Signals and Systems on Graphs

- Shift on a graph
  - Consider the graph signal, h(n), and the delta function located at a vertex m,

$$\delta_m(n) = \begin{cases} 1, & \text{for } m = n \\ 0, & \text{for } m \neq n \end{cases}$$

GDFT: $\Delta(k) = \sum_{n=0}^{N-1} \delta_m(n) u_k(n) = u_k(m)$

  - The shifted signal is obtained as a convolution of the original signal and an appropriately shifted delta function

$h(n) * \delta_m(n)$ with GDFT $H(k)u_k(m)$

$h_m(n)$: the shifted version of the graph signal, h(n), "toward" a vertex m

$$h_m(n) = h(n) * \delta_m(n) = \sum_{k=0}^{N-1} H(k) u_k(m) u_k(n)$$

Signals and Systems on Graphs

- Shift on a graph
  - Output signal

$$
\begin{aligned}
y(n) &= \sum_{k=0}^{N-1} X(k)H(k)u_k(n) \\
&= \sum_{k=0}^{N-1}\sum_{m=0}^{N-1} x(m)u_k(m)H(k)u_k(n) \\
&= \sum_{m=0}^{N-1} x(m)h_m(n) = x(n) * h(n),
\end{aligned}
$$

Where, $h_m(n) = \sum_{k=0}^{N-1} H(k)u_k(m)u_k(n) = T_m\{h(n)\}$

Signals and Systems on Graphs

- Optimal Denoising
  - Problem: Consider a measurement, x, composed of a slow-varying graph signal, s, and a fast changing disturbance, ε: $x = s + \varepsilon$. The aim is to design a filter for disturbance suppression (denoising), the output is denoted by y = H(x)
  - Solution:

$$J = \frac{1}{2}\|\mathbf{y} - \mathbf{x}\|_2^2 + \alpha \mathbf{y}^T \mathbf{L} \mathbf{y}$$

$$\frac{\partial J}{\partial \mathbf{y}^T} = \mathbf{y} - \mathbf{x} + 2\alpha \mathbf{L}\mathbf{y} = \mathbf{0}$$

$$\Rightarrow \quad \mathbf{y} = (\mathbf{I} + 2\alpha \mathbf{L})^{-1}\mathbf{x}.$$

$$\Rightarrow \quad \mathbf{Y} = (\mathbf{I} + 2\alpha \mathbf{\Lambda})^{-1}\mathbf{X}.$$ (The Laplacian spectral domain form)

$$H(\lambda_k) = \frac{1}{1 + 2\alpha\lambda_k}$$ (graph filter transfer function)

Signals and Systems on Graphs

- Systems on a Graph Defined Using **Random Walk Laplacian**
  - We can also use random walk matrix as graph shift operator

$$\mathbf{S} = \mathbf{D}^{-1}\mathbf{W}$$

  - A generalized form of the output from a system on a graph can then be written as

$$\mathbf{y} = h_0\mathbf{S}^0\,\mathbf{x} + h_1\mathbf{S}^1\,\mathbf{x} + \cdots + h_{M-1}\mathbf{S}^{M-1}\,\mathbf{x} = \sum_{\substack{m=0 \\ m=0}}^{M-1} h_m\mathbf{S}^m\,\mathbf{x},$$

Where $S_0 = I$, and $h_0, h_1, \ldots, h_{M-1}$ are the system coefficients

  - An unbiased version of the random walk shift with asymptotically power preserving property (asymptotically preserving the signal power over shifts with an increase in the number of edges and vertices) is defined as

$$\mathbf{S} = (\mathbf{I} + \mathbf{D})^{-1}(\mathbf{I} + \mathbf{W})$$

Signals and Systems on Graphs

- Systems on a Graph Defined Using **Random Walk Laplacian**
  - Asymptotically power preserving property proof (cont'd):
    - With the assumption that the random graph signal, x, follows the general random walk (GRW) model, it will exhibits the following properties:
      - **Graph Markov property**: the random process is dependent only of its shifted state

$$P\left(\mathbf{x}\middle|\bigcap_{m>0}\mathbf{S}^m\mathbf{x}\right)=P\left(\mathbf{x}\middle|\mathbf{S}\mathbf{x}\right)$$

      - **Graph Martingale property:** at a particular instant, the conditional expectation of the next value in a sequence, given all prior values, is equal to the present value (shifted state)

$$E\left\{\mathbf{x}\middle|\bigcap_{m>0}\mathbf{S}^m\mathbf{x}\right\}=\mathbf{S}\mathbf{x}$$

Signals and Systems on Graphs

- Systems on a Graph Defined Using **Random Walk Laplacian**
  - Asymptotically power preserving property proof:
    - In this way, the random walk can be described by a Markov matrix, $P \in R_{N\times N}$, with its (m, n)-th element defined as the transition probability, P_{mn}, of going from vertex m to vertex n
    - By setting S = P, this shift operator is unbiased, since each row in P sums up to unity. Furthermore, from the above property, we can derive that

$$\mathbf{Sx} = \mathbf{Px} = E\{\mathbf{x}\}$$

    - Then it can be shown

$$\lim_{N\to\infty} \|\mathbf{Sx}\|^2 = E\left\{\|\mathbf{x}\|^2\right\}$$

    - Since $\mathbf{y} = \sum_{m=0}^{M} h_m \mathbf{S}^m \mathbf{x}$ then $\lim_{N\to\infty} \|\mathbf{y}\|^2 \le \sum_{m=0}^{M-1} |h_m|^2 E\left\{\|\mathbf{x}\|^2\right\}$

Signals and Systems on Graphs

- Systems on a Graph Defined Using **Random Walk Laplacian**
  - In practice, the actual probabilities of vertex transition are often unknown but can be inferred from the available information of the graph topology, implied by the weight matrix, W.
  - Donsker's theorem states that the GRW has a probability density which convergences to that of the Wiener process.
  - In the graph setting, for a walker at a vertex m, the central limit theorem asserts that after a sufficiently large number of independent steps, the probability of walker's position is Gaussian distributed, $P_{mn} \propto e^{-r_{mn}^2}$, where r_{mn} is a measure of physical distance between vertices m and n.

$$\tilde{W}_{mn} = \begin{cases} e^{-r_{mn}^2}, & (m,n) \in \mathcal{E}, \\ 1, & m = n, \\ 0, & (m,n) \notin \mathcal{E}. \end{cases}$$

Signals and Systems on Graphs

- Systems on a Graph Defined Using **Random Walk Laplacian**
  - The the elements of the GRW weight matrix is

$$\tilde{W}_{mn} = \begin{cases} e^{-r_{mn}^2}, & (m,n) \in \mathcal{E}, \\ 1, & m = n, \\ 0, & (m,n) \notin \mathcal{E}. \end{cases}$$

  - To ensure that the transition probabilities sum up to unity, we therefore need to normalise the GRW weights to obtain

$$P_{mn} = \tilde{W}_{mn} / \tilde{D}_{mm}$$

  - Notice that the standard weight matrix, W, has zeros on the diagonal, $\widetilde{W} = (I + W)$ and $\widetilde{D} = (I + D)$
  - Thus $S = P = (I + D)^{-1}(I + W)$

- Subsampling, Compressed Sensing and Re-construction

Subsampling, Compressed Sensing and Reconstruction

- Motivation:

  In real world, graphs may comprise of a very large number of vertices, of the order of millions or even higher, which will raise computational and storage issues. This brings to the consideration of potential advantages of subsampling and compressive sensing defined on graphs

Subsampling, Compressed Sensing and Reconstruction

- Simplest case: subsampling of low-pass graph signal

  Problem setup:

  - We will consider graph signal of a low pass nature, such a signal can be expressed as a linear combination of K < N eigenvectors of the graph Laplacian which exhibit the lowest smoothness indices.

$$x(n) = \sum_{k=0}^{K-1} X(k) u_k(n), \;\; n = 0, 1, \ldots, N-1$$

  - The GDFT domain coefficients of this signal in the GDFT domain are of the following form

$$\mathbf{X} = [X(0), X(1), \ldots, X(K-1), 0, 0, \ldots, 0]^T$$

  - We want to reconstruct x from a smaller set of signal samples

Subsampling, Compressed Sensing and Reconstruction

- Simplest case: subsampling of low-pass graph signal (cont'd)

  Solution:

  - The smallest number of graph signal samples, M, needed to recover the sparse signal is $M = K < N$ (for stability of reconstruction, it is common to employ $K \leq M < N$ graph signal samples).
  - The vector of available graph signal samples will be referred to as the **measurement vector**, we will denote it as y.
  - Set of vertices (a random subset of $V = \{0, 1, 2, \ldots, N-1\}$) over which the samples of graph signal are available is denoted by

$$\mathbb{M} = \{n_1, n_2, \ldots, n_M\}$$

Subsampling, Compressed Sensing and Reconstruction

- Simplest case: subsampling of low-pass graph signal (cont'd)

  Solution:

  - The measurement matrix A_{MN} can now be defined using the IGDFT, x =UX

$$\begin{bmatrix} x(n_1) \\ x(n_2) \\ \vdots \\ x(n_M) \end{bmatrix} = \begin{bmatrix} u_0(n_1) & u_1(n_1) & \dots & u_{N-1}(n_1) \\ u_0(n_2) & u_1(n_2) & \dots & u_{N-1}(n_2) \\ \vdots & \vdots & \ddots & \vdots \\ u_0(n_M) & u_1(n_M) & \dots & u_{N-1}(n_M) \end{bmatrix} \begin{bmatrix} X(0) \\ X(1) \\ \vdots \\ X(N-1) \end{bmatrix} \qquad \mathbf{y} = \mathbf{A}_{MN}\mathbf{X}$$

⇩ low-pass nature

$$\begin{bmatrix} x(n_1) \\ x(n_2) \\ \vdots \\ x(n_M) \end{bmatrix} = \begin{bmatrix} u_0(n_1) & u_1(n_1) & \dots & u_{K-1}(n_1) \\ u_0(n_2) & u_1(n_2) & \dots & u_{K-1}(n_2) \\ \vdots & \vdots & \ddots & \vdots \\ u_0(n_M) & u_1(n_M) & \dots & u_{K-1}(n_M) \end{bmatrix} \begin{bmatrix} X(0) \\ X(1) \\ \vdots \\ X(K-1) \end{bmatrix} \qquad \mathbf{y} = \mathbf{A}_{MK}\mathbf{X}_K$$

Subsampling, Compressed Sensing and Reconstruction

- Simplest case: subsampling of low-pass graph signal (cont'd)

  Solution:

  - For M = K independent measurements, this system can be solved uniquely
  - For M > K the system is overdetermined and the solution is found in the least squares (LS) sense

$$\mathbf{X}_K = (\mathbf{A}_{MK}^T \mathbf{A}_{MK})^{-1} \mathbf{A}_{MK}^T \mathbf{y} = \text{pinv}(\mathbf{A}_{MK})\mathbf{y}$$

  - After X_K is calculated, all GDFT values follow directly as X = [X(0), X(1), . . . , X(K −1), 0, 0, . . . , 0]T , where the assumed zero values are added for X(K), X(K + 1), . . . , X(N −1)
  - The graph signal is then recovered at all vertices using x = UX
  - Recovery Condition: $A_{MK}^T A_{MK}$ is nonsingular

Subsampling, Compressed Sensing and Reconstruction

- Subsampling of Sparse Graph Signals (in GDFT domain)
  - Known coefficient position
    - The previous analysis holds not only for a low-pass type of the graph signal, x, and its corresponding GDFT, X, but also for case of GDFT, X, with K nonzero values at arbitrary, but known spectral positions:

$$X(k) = 0 \text{ for } k \notin \mathbb{K} = \{k_1, k_2, \ldots, k_K\}$$

    - The measurement matrix A_{MK} can also be defined using the IGDFT as:

$$\begin{bmatrix} x(n_1) \\ x(n_2) \\ \vdots \\ x(n_M) \end{bmatrix} = \begin{bmatrix} u_{k_1}(n_1) & u_{k_2}(n_1) & \ldots & u_{k_K}(n_1) \\ u_{k_1}(n_2) & u_{k_2}(n_2) & \ldots & u_{k_K}(n_2) \\ \vdots & \vdots & \ddots & \vdots \\ u_{k_1}(n_M) & u_{k_2}(n_M) & \ldots & u_{k_K}(n_M) \end{bmatrix} \begin{bmatrix} X(k_1) \\ X(k_2) \\ \vdots \\ X(k_K) \end{bmatrix}$$

    - We can then follow the same procedures as the subsampling of low-pass signal to solve for the nonzero spectral values X(k)

Subsampling, Compressed Sensing and Reconstruction

- Subsampling of Sparse Graph Signals (in GDFT domain)
  - Support Matrices, Subsampling and Upsampling
    - Assume that a graph signal, x, is subsampled in such way that it is available on a subset of vertices $n \in M = \{n_1, n_2, \ldots, n_M\}$, rather than on the full set of vertices. For this **subsampled signal**, we can define its **upsampled version**, $\mathbf{x_s}$, by adding zeros at the vertices where the signal is not available

$$\mathbf{x}_s = \mathbf{B}\mathbf{x}$$

where the support matrix B is an N × N diagonal matrix with ones at the diagonal positions which correspond to $M = \{n_1, n_2, \ldots, n_M\}$ and zeros elsewhere.

    - For graph signals which are also sparse in the GDFT domain, the additional constraint is that the signal, x, has only $K \leq M$ nonzero coefficients in the GDFT domain, $X = U^T x$, at $k \in K = \{k_1, k_2, \ldots, k_K\}$,

$$\mathbf{X} = \mathbf{C}\mathbf{X}$$

where the support matrix C is an N × N diagonal matrix with ones at the diagonal positions which correspond to $K = \{k_1, k_2, \ldots, k_K\}$, and zeros elsewhere

Subsampling, Compressed Sensing and Reconstruction

- Subsampling of Sparse Graph Signals (in GDFT domain)
  - Support Matrices, Subsampling and Upsampling (cont'd)
    - The reconstruction formula then follows from

$$\mathbf{x}_s = \mathbf{B}\mathbf{x} = \mathbf{BUX} = \mathbf{BUCX}$$

$$\Downarrow$$

$$\mathbf{X} = \mathbf{CX} = \text{pinv}(\mathbf{BUC})\mathbf{x}_s$$

    - Recovery condition: for K nonzero coefficients of CX, the rank of BUC is K (if there are K linearly independent equations), that is

$$rank(BUC) = K$$

Subsampling, Compressed Sensing and Reconstruction

- Subsampling of Sparse Graph Signals (in GDFT domain)
  - Unknown coefficient position
    - The positions of nonzero spectral coefficients $K = \{k_1, k_2, \ldots, k_K\}$ are not known. This case has been addressed within standard compressive sensing theory and can be formulated as

$$\min \|\mathbf{X}\|_0 \text{ subject to } \mathbf{y} = \mathbf{A}_{MN}\mathbf{X}$$

    - Simple Solutions:
      - Estimate the positions $K = \{k_1, k_2, \ldots, k_K\}$ of the nonzero coefficients using $M > K$ signal samples
      - Reconstruct the nonzero coefficients of X at the estimated positions K, along with the signal x at all vertices, using the methods for the reconstruction with the known nonzero coefficient positions, described previously. The nonzero coefficients at positions K are calculated as $X_K = \text{pinv}(A_{MK})y$.

Subsampling, Compressed Sensing and Reconstruction

- Subsampling of Sparse Graph Signals (in GDFT domain)
  - Unknown coefficient position (cont'd)
    - Estimation of nonzero positions: through the projection of measurements (available signal samples), y, on the measurement matrix

$$\mathbf{A}_{MN} = \begin{bmatrix} u_0(n_1) & u_1(n_1) & \dots & u_{N-1}(n_1) \\ u_0(n_2) & u_1(n_2) & \dots & u_{N-1}(n_2) \\ \vdots & \vdots & \ddots & \vdots \\ u_0(n_M) & u_1(n_M) & \dots & u_{N-1}(n_M) \end{bmatrix}$$

$$\Downarrow$$

$$\mathbf{X}_0 = \mathbf{A}_{MN}^T \mathbf{y}$$

where the positions of K largest values in X_0 are used as an estimate of the nonzero positions, K.

Subsampling, Compressed Sensing and Reconstruction

- Subsampling of Sparse Graph Signals (in GDFT domain)
  - Unknown coefficient position (cont'd)
    - Estimation of nonzero positions with unknown k: iterative method
      - In the first iteration we assume K = 1 and proceed to estimate the largest spectral component in the graph signal. Upon determining its position as $\mathbf{k_1 = argmax|A^T}_{\mathbf{MN}}\ \mathbf{y|}$, the initially empty set of the nonzero positions becomes $\mathbf{K = \{k_1\}}$. The reconstructed vector $\mathbf{y_1 = A_1X_1}$, where $\mathbf{X_1 = pinv(A_{M1})y}$, is then removed from the measurements, y. In this case, the matrix A_{M1} is a column of the matrix A_{MN} defined by the index k1. The difference e = y − y1 is used as the measurement vector in the next step

Subsampling, Compressed Sensing and Reconstruction

- Subsampling of Sparse Graph Signals (in GDFT domain)
  - Unknown coefficient position (cont'd)
    - Estimation of nonzero positions: iterative method
      - The position of the second largest spectral component in the graph signal is estimated by solving $\mathbf{k_2 = argmax|A^T_{MN}\, e|}$. The set of nonzero positions now becomes $\mathbf{K = \{k_1, k_2\}}$. The first and the second component of the graph signal are now estimated as $\mathbf{X_2 = pinv(A_{M2})y}$, where the matrix A_{M2} is a submatrix of the measurement matrix, A_{MN} , which consists of the columns defined by the indices k_1 and k_2. The reconstructed vector $y_2 = A_2X_2$, is removed from the measurements, y, with the error, e = y − y2, now acting as the new measurement vector.

Subsampling, Compressed Sensing and Reconstruction

- Subsampling of Sparse Graph Signals (in GDFT domain)
  - Unknown coefficient position (cont'd)
    - Estimation of nonzero positions with unknown k: iterative method
      - The procedure is iteratively repeated K times or until the remaining measurement values in e are negligible. In the cases when the sparsity, K, is unknown, the procedure is iterated until $\|e\|_2 < \varepsilon$, where ε is a predefined precision.

Subsampling, Compressed Sensing and Reconstruction

- Measurements as Linear Combinations of Samples
  - If some spectrum coefficients of a graph signal are strongly related to only a few of the signal samples, then these signal samples may not be good candidates for the measurements. Such measurements are linear combinations of all signal samples.

$$\begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(M) \end{bmatrix} = \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1N} \\ b_{21} & b_{12} & \dots & b_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ b_{M1} & b_{M2} & \dots & b_{MN} \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ \vdots \\ x(N-1) \end{bmatrix} \iff \mathbf{y} = \underbrace{\mathbf{B}_{MN}}_{\text{Weight coefficients}} \mathbf{x}$$

Subsampling, Compressed Sensing and Reconstruction

- Measurements as Linear Combinations of Samples (cont'd)
  - For reconstruction, the sparsity of a graph signal, x, should be again assumed in the GDFT domain.

$$\mathbf{y} = \mathbf{B}_{MN}\mathbf{x} = \mathbf{B}_{MN}\mathbf{U}\mathbf{X} = \mathbf{A}_{MN}\mathbf{X}.$$

The reconstruction is now obtained as a solution to

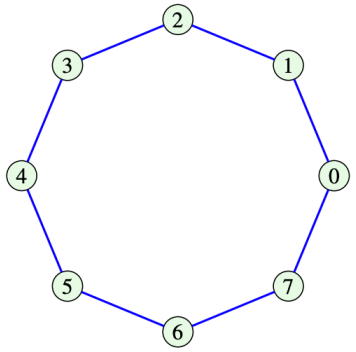
$$\min \|\mathbf{X}\|_0 \text{ subject to } \mathbf{y} = (\mathbf{B}_{MN}\mathbf{U})\mathbf{X}$$

Or alternatively

$$\min \|\mathbf{X}\|_1 \text{ subject to } \mathbf{y} = (\mathbf{B}_{MN}\mathbf{U})\mathbf{X}$$

Subsampling, Compressed Sensing and Reconstruction

- Aggregate Sampling
  - A specific form of a linear combination of graph signals is referred to as **aggregate sampling**
  - Circular graph
    - Consider a graph signal, x, at a vertex n. If the signal is observed at this vertex/instant only, then its value is $\mathbf{y_0(n) = x(n)}$.
    - Applying the graph shift operator, we have $\mathbf{y_1 = Ax}$, then for the same vertex, n, we have $\mathbf{y_1(n) = x(n-1)}$.
    - If we continue this "shift and observe" operation on the directed circular graph N times at the same vertex/instant, n, we will eventually have all signal values $\mathbf{x(n), x(n-1), \ldots, x(n-N+1)}$ observed at vertex n.



Subsampling, Compressed Sensing and Reconstruction

- Aggregate Sampling
  - Circular graph (cont'd)
    - To proceed with signal reconstruction, if the shifts are stopped after $M < N$ steps, the available signal samples will be $\mathbf{x(n), x(n-1), \ldots, x(n-M+1)}$. From this reduced set of measurements/samples we can still recover the full graph signal, x, using compressive sensing based reconstruction methods, if the appropriate reconstruction conditions are met.

Subsampling, Compressed Sensing and Reconstruction

- Aggregate Sampling
  - Arbitrary graph
    - The same procedure can be applied to a signal observed in the same way on an arbitrary graph. Assume that we observe the graph signal at only one vertex, n, and obtain one graph signal sample

$$y_0(n) = x(n) \iff y(0) = y_0(n).$$

    - With one-step signal shift on a graph $\mathbf{y_1} = \mathbf{Ax}$, the sample of a graph signal at vertex n will now be a sum of all signal samples that have shifted to this vertex.

$$y_1(n) = \sum_m A_{nm} x(m), \iff y(1) = y_1(n)$$

    - One more signal shift on the graph yields

$$y_2(n) = \sum_m A^{(2)}_{nm} x(m)$$ where $A^{(2)}_{mn}$ are the elements of matrix $A^2 = AA$

Subsampling, Compressed Sensing and Reconstruction

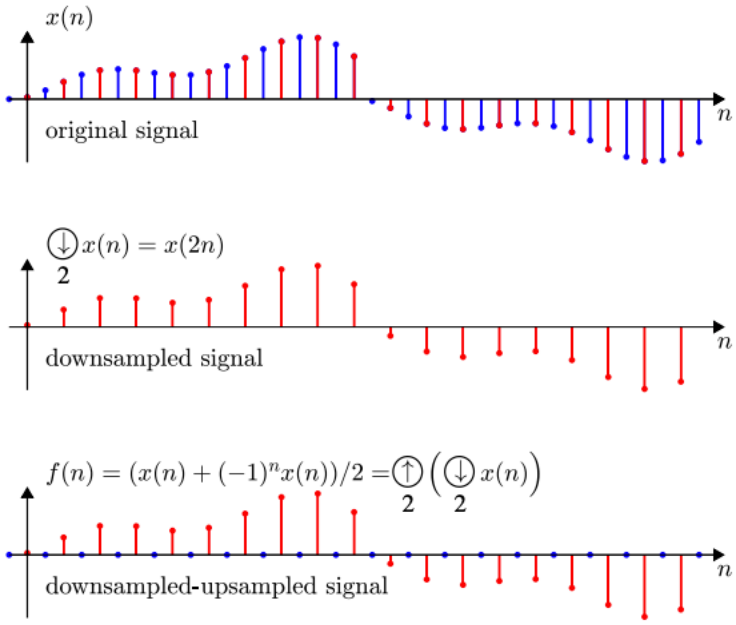
- Aggregate Sampling
  - Arbitrary graph (cont'd)
    - If we proceed with shifts M = N times, we will obtain a system of N linear equations, from which all signal values, x(n), can be calculated

$$\mathbf{y} = \mathbf{B}_{MN}\mathbf{x}$$

    - If we stop at M < N, the signal can still be recovered using compressive sensing based reconstruction methods if the signal is sparse and the reconstruction conditions are met
    - **Note:** instead of M signal samples (instants) at one vertex, we may use, for example, P samples at vertex n and (M − P) samples from a vertex m. Other combinations of vertices and samples may be also used to obtain M measurements and to fully reconstruct a signal.

Subsampling, Compressed Sensing and Reconstruction

- Filter Bank on Graph
  - **Filter bank**: A filter bank is a system that divides the input signal x(n) into a set of analysis signals $x_1(n)$, $x_2(n)$, … , each of which corresponds to a different region in the spectrum of x(n)
  - Subsampling and upsampling are the two standard operators used to alter the scale at which the signal is processed.



Example:
subsampling of a signal by a factor of 2, followed by the corresponding upsampling

$$f(n) = \frac{1}{2}\Big(x(n) + (-1)^n x(n)\Big) = \frac{1}{2}\Big((1 + (-1)^n)x(n)\Big)$$

Subsampling, Compressed Sensing and Reconstruction

- Filter Bank on Graph (cont'd)
  - Subsampling and upsampling on graphs
    - Consider a graph with the set of vertices V. Any set of vertices can be considered as a union of two disjoint subsets E and H, such that

$$\mathbf{V = E \cup H \text{ and } E \cap H = \emptyset}$$

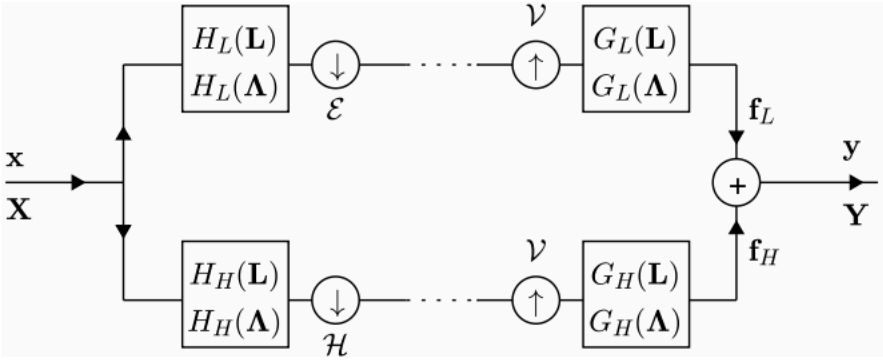
    - The subsampling-upsampling procedure:
      1. Subsample the signal on a graph by keeping only signal values on the vertices n ∈ E, while not altering the original graph topology
      2. Upsample the graph signal by setting the signal values for the vertices n /∈ E to zero
    - Resulting graph signal:

$$f(n) = \frac{1}{2}\left(1 + (-1)^{\beta_{\mathcal{E}}(n)}\right)x(n) \quad \text{where} \quad \beta_{\mathcal{E}}(n) = \begin{cases} 0, & \text{if } n \in \mathcal{E} \\ 1, & \text{if } n \in \mathcal{H}. \end{cases}$$

$$\mathbf{f} = \frac{1}{2}(\mathbf{x} + \mathbf{J}_{\mathcal{E}}\mathbf{x}) = \frac{1}{2}(\mathbf{I} + \mathbf{J}_{\mathcal{E}})\mathbf{x}, \quad \text{where} \quad \mathbf{J}_{\mathcal{E}} = \text{diag}((-1)^{\beta_{\mathcal{E}}(n)}),\ n \in \mathcal{V}.$$

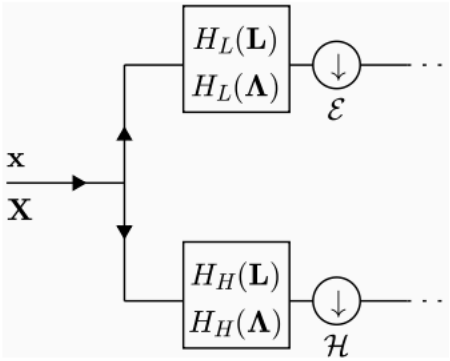
Subsampling, Compressed Sensing and Reconstruction

- Filter Bank on Graph
  - Subsampling and upsampling on graphs (cont'd)
    - Two-channel wavelet filter bank on graph
      - Provides decomposition of a graph signal into the corresponding low-pass (smooth) and high-pass (fast-varying) constituents



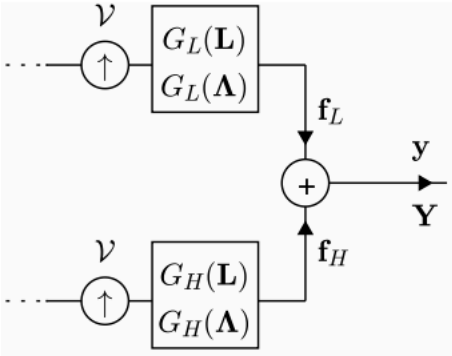
Subsampling, Compressed Sensing and Reconstruction

- Filter Bank on Graph
  - Subsampling and upsampling on graphs (cont'd)
    - Two-channel wavelet filter bank on graph
      - Analysis side – signal decomposition:
        1. Decompose signals into low-pass and high-pass parts
           - operator HL(L) acts as a low-pass filter, transferring the low-pass components of the graph signal, while operator HH(L) does the opposite
        2. Downsampling:
           - For the low-pass filter, $H_L(H)$, keeps only the graph signal values, x, at the vertices $n \in E$
           - For the high-pass filtering with the operator $H_H(L)$, is subsequently followed by a downsampling to the vertices $n \in H$.



Subsampling, Compressed Sensing and Reconstruction

- Filter Bank on Graph
  - Subsampling and upsampling on graphs (cont'd)
    - Two-channel wavelet filter bank on graph
      - Synthesis side – signal reconstruction:
        1. Upsampling: adding zeros to the complementary sets of vertices
        2. Perform the graph signal reconstruction based on the upsampled versions.



Subsampling, Compressed Sensing and Reconstruction

- Filter Bank on Graph
  - Subsampling and upsampling on graphs (cont'd)
    - Two-channel wavelet filter bank on graph
      - Consider a graph signal x, if the graph signal , passes through a lowpass analysis filter, $H_L(L)$, the output signal is $\mathbf{H_L(L)x}$. The downsampled-upsampled form of the output signal is $\frac{1}{2}(\mathbf{I}+\mathbf{J}_{\mathcal{E}})H_L(\mathbf{L})\mathbf{x}$. After the syntheses filter, $G_L(L)$, the graph signal output becomes

$$\mathbf{f}_L = \frac{1}{2}G_L(\mathbf{L})(\mathbf{I}+\mathbf{J}_{\mathcal{E}})H_L(\mathbf{L})\mathbf{x}$$

      - The same holds for the high-pass part

$$\mathbf{f}_H = \frac{1}{2}G_H(\mathbf{L})(\mathbf{I}+\mathbf{J}_{\mathcal{H}})H_H(\mathbf{L})\mathbf{x}$$

where $\mathbf{J}_{\mathcal{H}} = -\mathbf{J}_{\mathcal{E}} = \mathrm{diag}((-1)^{1-\beta_{\mathcal{E}}(n)})$ and $\mathbf{J}_{\mathcal{H}} + \mathbf{J}_{\mathcal{E}} = \mathbf{0}$

Subsampling, Compressed Sensing and Reconstruction

- Filter Bank on Graph
  - Subsampling and upsampling on graphs (cont'd)
    - Two-channel wavelet filter bank on graph
      - The overall output is

$$\mathbf{y} = \mathbf{f}_L + \mathbf{f}_H = \frac{1}{2}(G_L(\mathbf{L})H_L(\mathbf{L}) + G_H(\mathbf{L})H_H(\mathbf{L}))\mathbf{x} + \frac{1}{2}(G_L(\mathbf{L})\mathbf{J}_{\mathcal{E}}H_L(\mathbf{L}) + G_H(\mathbf{L})\mathbf{J}_{\mathcal{H}}H_H(\mathbf{L}))\mathbf{x}$$

      - The perfect reconstruction condition, y = x, is then achieved if

$$G_L(\mathbf{L})H_L(\mathbf{L}) + G_H(\mathbf{L})H_H(\mathbf{L}) = 2\mathbf{I},$$
$$G_L(\mathbf{L})\mathbf{J}_{\mathcal{E}}H_L(\mathbf{L}) - G_H(\mathbf{L})\mathbf{J}_{\mathcal{E}}H_H(\mathbf{L}) = \mathbf{0}.$$

- Vertex-Frequency Representation

Vertex-Frequency Representation

- Motivation
  - Oftentimes in practical applications concerned with large graphs, we may not be interested in the analysis of the entire graph signal, but rather in its local behavior.
  - The Big Data paradigm has revealed the possibility of using smaller and localized subsets of the available information to enable reliable mathematical analysis and local characterization of subsets of data of interest
  - Vertex-frequency analysis can be used for graph signal estimation, filtering, and efficient representation

Vertex-Frequency Representation

- Localized Graph Fourier Transform (LGFT)
  - For a given vertex m and a spectral index k, the localized graph Fourier transform is defined as

$$S(m,k) = \sum_{n=0}^{N-1} x(n) h_m(n)\, u_k(n)$$

GDFT of signal x(n)

vertex localization window function, which in general, should localizes the signal content around the vertex m

Vertex-Frequency Representation

- Localized Graph Fourier Transform (LGFT)
  - The LGDFT can also be represented as a matrix form, **S**, which contains all elements, S(m, k), m = 0, 1, . . . , N −1, k = 0, 1, . . . , N − 1. The columns of S which correspond to a vertex m are given by

$$\mathbf{s}_m = \mathrm{GDFT}\{x(n)h_m(n)\} = \mathbf{U}^T\mathbf{x}_m$$

Matrix where each column is a eigenvector of graph Laplacian

A vector of which the elements are equal to the graph signal samples, x(n), multiplied by the window function, $h_m(n)$, centered at the vertex m

Vertex-Frequency Representation

- Vertex Domain Windows: $h_m(n)$
  - Spectral domain definition

$$h_m(n) = h(n) * \delta_m(n) = \sum_{k=0}^{N-1} H(k) u_k(m) u_k(n)$$

where

$$\delta_m(n) = \begin{cases} 1, & \text{for } m = n \\ 0, & \text{for } m \neq n \end{cases}$$

$$H(k) = C \exp(-\lambda_k \tau)$$

basic function of the window: where C denotes the "window amplitude" and $\tau > 0$ is a constant which determines the window width in the spectral domain

Vertex-Frequency Representation

- Vertex Domain Windows: $h_m(n)$
  - Spectral domain definition

$$S(m,k) = \sum_{n=1}^{N} x(n) h_m(n)\, u_k(n)$$

$$= \sum_{n=1}^{N} \sum_{p=1}^{N} x(n) \underbrace{H(p) u_p(m) u_p(n)}_{h_m(n)}\, u_k(n)$$

Vertex-Frequency Representation

- Vertex Domain Windows: $h_m(n)$
  - Spectral domain definition
    - **Vertex-frequency kernel**: the modulated (frequency shifted) version of the window centered at a vertex m and for a spectral index k

$$\mathcal{H}_{m,k}(n) = h_m(n)u_k(n) = \Big(\sum_{p=0}^{N-1} H(p)u_p(m)u_p(n) \Big) u_k(n)$$

    - LGDT with vertex-frequency kernel representation

$$S(m,k) = \langle \mathcal{H}_{m,k}(n), x(n) \rangle = \sum_{n=0}^{N-1} \mathcal{H}_{m,k}(n)x(n)$$

Vertex-Frequency Representation

- Signal localization in the spectral domain
  - GDFT of the original signal and a spectral domain window

$$S(m,k) = \sum_{p=0}^{N-1} X(p) H(k-p)\, u_p(m)$$

inverse GDFT of x(p)

spectral domain window, H(k − p), which is centered around spectral index k

Vertex-Frequency Representation

- LGFT with Band-Pass Function
  - Assume that the GDFT of the localization window, $h_m(n)$, corresponds to the transfer function of a bandpass system on a graph, centered at an eigenvalue, λ_k, and around it. This transfer function is defined in the form of a polynomial

$$H_k(\lambda_p) = h_{0,k} + h_{1,k}\lambda_p + \cdots + h_{M-1,k}\lambda_p^{M-1}$$

with (M −1) as the polynomial order and k = 0, 1, . . . , K, where K is the number of spectral bands

  - The vertex-frequency kernel is defined as

$$\mathcal{H}_{m,k}(n) = \sum_{p=0}^{N-1} H_k(\lambda_p) u_p(m) u_p(n).$$

Vertex-Frequency Representation

- LGFT with Band-Pass Function
  - The local vertex domain transform for a vertex m and a spectral index k is

$$S(m,k)=\sum_{n=0}^{N-1}\mathcal{H}_{m,k}(n)x(n)$$

$$=\sum_{n=0}^{N-1}\sum_{p=0}^{N-1}x(n)H_k(\lambda_p)u_p(m)u_p(n)=\sum_{p=0}^{N-1}X(p)H_k(\lambda_p)u_p(m)$$

  - In matrix form, the column k is

$$\mathbf{s}_k=\mathbf{U}H_k(\mathbf{\Lambda})\mathbf{U}^T\mathbf{x}=H_k(\mathbf{L})\mathbf{x}=\sum_{p=0}^{M-1}h_{p,k}\mathbf{L}^p\,\mathbf{x}$$

Vertex-Frequency Representation

- Vertex Domain Windows: $h_m(n)$
  - Vertex domain definition: the window function can be defined as a function of vertex distance, in the form of

$$h_m(n) = g(d_{mn})$$

    - d_{mn} is distance between vertices m and n, which is equal to the length of the shortest walk from vertex m to vertex n
    - g(d) corresponds to any basic window function in classical signal processing
      - Ex: Hann window:

$$h_m(n) = \frac{1}{2}\Big(1 + \cos(\pi d_{mn}/D)\Big), \text{ for } 0 \le d_{mn} < D$$

where D is the assumed window width

Vertex-Frequency Representation

- Vertex Domain Windows: $h_m(n)$
  - Vertex domain definition
    - Window functions for every vertex can be calculated in a matrix form as follows
      - For the vertices for which the distance is $d_{mn} = 1$, window functions are defined trough an **adjacency matrix** $A_1 = A$. In other words, the vertices which belong to the one-neighborhood of a vertex, m, are indicated by unit-value elements in the m^{th} row of the adjacency matrix A (in unweighted graphs). In weighed graphs, the corresponding adjacency matrix A can be obtained from the weighting matrix W as A = sign(W)
      - For the vertices for which the distance is dmn = 2

$$\mathbf{A}_2 = (\mathbf{A} \odot \mathbf{A}_1) \circ (\mathbf{1} - \mathbf{A}_1) \circ (\mathbf{1} - \mathbf{I}),$$

where $\odot$ denotes the logical (Boolean) matrix product, $\circ$ is the Hadamard (element-by-element) product, and **1** is a matrix with all elements equal to 1

$$** \; A = B \odot C \iff A_{ij} = \bigvee_{k=0}^{n} B_{i,k} \wedge C_{k,j}$$

Vertex-Frequency Representation

- Vertex Domain Windows: $h_m(n)$
  - Vertex domain definition
    - Window functions for every vertex can be calculated in a matrix form as follows (cont'd)
      - For the vertices for which the distance is $d_{mn} > 2$, we arrive at a recursive relation for the calculation of a matrix which will give the information about the vertices separated by the distance d. Such a matrix has the form

$$\mathbf{A}_d = (\mathbf{A} \odot \mathbf{A}_{d-1}) \circ (\mathbf{1} - \mathbf{A}_{d-1}) \circ (\mathbf{1} - \mathbf{I})$$

Vertex-Frequency Representation

- Vertex Domain Windows: $h_m(n)$
  - Vertex domain definition
    - The window matrix for an assumed graph window width, D, can now be defined as

$$\mathbf{P}_D = g(0)\mathbf{I} + g(1)\mathbf{A}_1 + \cdots + g(D-1)\mathbf{A}_{D-1}$$

    - A graph signal which is localized around a vertex m, may be formed based on this matrix, as

$$x_m(n) = h_m(n)x(n) = P_D(n,m)x(n)$$

    - The LGFT representation of a graph signal, x(n), then becomes

$$S(m,k) = \sum_{n=0}^{N-1} x(n)h_m(n)\, u_k(n) = \sum_{n=0}^{N-1} x(n)P_D(n,m)\, u_k(n)$$

Vertex-Frequency Representation

- Vertex Domain Windows: $h_m(n)$
  - Vertex domain definition
    - The vertex-frequency kernel is

$$\mathcal{H}_{m,k}(n) = h_m(n)u_k(n) = P_D(n,m)u_k(n).$$

    - LGFT in matrix form

$$\mathbf{S} = \mathbf{U}^T\left(\mathbf{P}_D \circ [\mathbf{x},\ \mathbf{x},\ \ldots,\ \mathbf{x}]\right)$$

where [x, x, . . . , x] is an N × N matrix, x is the signal vector

Vertex-Frequency Representation

- Inversion of the LGFT
  - Inversion by summation of LGFT
    - The reconstruction of a graph signal, x(n), from its local spectrum, S(m, k), can be performed through an inverse GDFT

$$x(n)h_m(n) = \sum_{k=0}^{N-1} S(m,k)\, u_k(n)$$

$$\Downarrow$$

$$x(n) = \frac{1}{\sum_{m=0}^{N-1} h_m(n)} \sum_{m=0}^{N-1} \sum_{k=0}^{N-1} S(m,k) u_k(n)$$

Vertex-Frequency Representation

- Inversion of the LGFT
  - Inversion by summation of LGFT (cont'd)
    - If the windows $h_m(n)$ satisfy the condition for every vertex, n

$$\sum_{m=0}^{N-1} h_m(n) = 1$$

    - then the reconstruction is vertex independent

$$x(n) = \sum_{m=0}^{N-1} \sum_{k=0}^{N-1} S(m,k) u_k(n) = \sum_{k=0}^{N-1} X(k) u_k(n)$$

where

$$X(k) = \sum_{m=0}^{N-1} S(m,k)$$

Vertex-Frequency Representation

- Inversion of the LGFT
  - Inversion by summation of LGFT (cont'd)
    - We can enforce the above condition by normalizing the elements of matrix A_d in prior to prior to the calculation of matrix P_D, in such a way that the sum of each of its columns is equal to 1, which allows us to arrive at

$$\sum_{m=0}^{N-1} h_m(n) = \sum_{m=0}^{N-1} P_D(n,m) = \sum_{d=1}^{D-1} g(d) = const$$

Vertex-Frequency Representation

- Inversion of LGFT
  - Kernel based inversion
    - For LGFT defined by

$$\sum_{m=0}^{N-1}\sum_{k=0}^{N-1} S(m,k)\mathcal{H}_{m,k}(n) = \sum_{m=0}^{N-1}\left(\sum_{k=0}^{N-1} S(m,k)h_m(n)u_k(n)\right)$$

$$= \sum_{m=0}^{N-1}\left(\sum_{i=0}^{N-1} \underset{k\to i}{\mathrm{IGDFT}}\{S(m,k)\}\underset{k\to i}{\mathrm{IGDFT}}\{h_m(n)u_k(n)\}\right)$$

$$= \sum_{m=0}^{N-1}\sum_{i=0}^{N-1}[x(i)h_m(i)][h_m(n)\delta(n-i)]$$

$$= \sum_{m=0}^{N-1} x(n)h_m^2(n) = x(n)\sum_{m=0}^{N-1} h_m^2(n)$$

By Parseval's theorem for graph signals

$$\sum_{n=0}^{N-1} x(n)y(n) = \sum_{k=0}^{N-1} X(k)Y(k)$$

$$x(n) = \frac{1}{\sum_{m=0}^{N-1} h_m^2(n)}\sum_{m=0}^{N-1}\sum_{k=0}^{N-1} S(m,k)\mathcal{H}_{m,k}(n)$$

TAHNK YOU