More advanced topics in GSP:

- **Topic 1: More about filters**

We defined graph filters as operators that are diagonal in the spectral (Fourier) domain, so that it weights linearly independently modes of different frequencies on graphs.

. $\mathcal{H}$ is a filter on a graph if it acts as: $\tilde{y}(\lambda_k) = h(\lambda_k)\,\tilde{x}(\lambda_k)$ **Def 1**

. with $\tilde{\underline{\underline{H}}} = \mathrm{diag}(h(\lambda_0), \dots h(\lambda_{N-1}))$: $\tilde{\underline{y}} = \tilde{\underline{\underline{H}}}\,\tilde{\underline{x}}$

Now, one can think about (linear) filters in the original space of nodes, the graph:

On G, $\mathcal{H}$ acts as a different (non diagonal) matrix $\underline{\underline{H}}$:

$$\underline{\underline{H}} = \underline{\underline{\chi}}\,\underline{\underline{\tilde{H}}}\,\underline{\underline{\chi}}^T$$

$\Leftrightarrow$ We could define Filters as operator $\underline{\underline{H}}$ (in matrix form) such that $\underline{\underline{\chi}}^T\,\underline{\underline{H}}\,\underline{\underline{\chi}}$ is diagonal. **Def 2**

Prop : If the eigenvalues of $\underline{\underline{L}}$ are of multiplicity 1, **Def 1 ≡ Def 2**

Why this restriction ? If two eigenvectors have the same eigenvalue λ_k, Def 1 imposes that $\mathcal{H}$ weights with the same $h(\lambda_k)$, while Def 2 does not require it.

A third definition of graph filters would be to put the emphasis on the idea that a filter is a function of the Laplacian. Hence, it should commute in the node domain with L.

Def Let $\mathcal{F}_G = \{ \underline{\underline{H}} \text{ such that } \underline{\underline{H}}\,\underline{\underline{L}} = \underline{\underline{L}}\,\underline{\underline{H}} \}$ — Def 3

Prop : $\mathcal{F}_G$ includes all graph filters and if $\underline{\underline{L}}$ has only simple eigenvalues, then $\mathcal{F}_G$ is equal to the set of filters.

$$\text{Def 3} = \text{Def 2} = \text{Def 1}$$

Proof : Let's consider $\underline{\underline{H}} \in \mathcal{F}_G$.

Using that $\underline{\underline{L}} = \underline{\underline{\chi}}\,\underline{\underline{\Lambda}}\,\underline{\underline{\chi}}^T$, we have : $H\chi\Lambda\chi^T = \chi\Lambda\chi^T H$

hence $\underbrace{\chi^T H \chi}_{\text{noted } Y}\Lambda\underbrace{\chi^T\chi}_{Id} = \underbrace{\chi^T\chi}_{Id}\Lambda\underbrace{\chi^T H\chi}_{Y}$

The constraint is that $\boxed{Y\Lambda = \Lambda Y}$.

Developing the products : $\begin{cases} (Y\Lambda)_{ij} = \sum_k Y_{ik}\Lambda_{kj} = \sum_k Y_{ik}\lambda_k\delta_{kj} = \lambda_j Y_{ij}, \\ (\Lambda Y)_{ij} = \lambda_i Y_{ij}, \end{cases}$

we have that $(\forall i,j) \quad \lambda_i Y_{ij} = \lambda_j Y_{ij}$.

Under the hyp of simple eigenvalues (all λ_k's are different), this imposes that Y is diagonal. Hence $\underline{\underline{\chi}}^T\underline{\underline{H}}\,\underline{\underline{\chi}}$ is diagonal and $\underline{\underline{H}}$ satisfies Def 2.

As a consequence, it would be easy to consider mostly filters that are written explicitely as operators commuting, in the node domain, with L. This would be the set of **polynomial filters**:

$\underline{\underline{H}}$ such that $\underline{\underline{H}} = \sum_{m=0}^{p} a_m \underline{\underline{L}}^m$ with $p \in \mathbb{N}$ (finite).

Def 4

Property: polynomial filters are included in $\mathcal{F}_G$.
if all the eigenvalues are simple, then the two sets are equal.

• Topic 2: Implementation of graph filters

We can now think about several ways (at least) to implement filters:

• **Direct application in the spectral domain:**
First, diagonalise the Laplacian matrix ; algorithm in O(N^3), memory in O(N^2) for the matrix $\underline{\underline{\chi}}$
Then apply directly the formula: $\underline{f}(\underline{x}) = \underline{\underline{\chi}}\, \underline{\underline{\hat{H}}}\, \underline{\underline{\chi}}^T \underline{x}$

This is costly!
We lose the advantage of having often sparse graphs G, and so sparse matrices L or A.

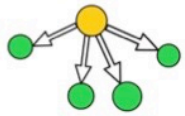
• **Polynomial approximation of graph filters:**
Using Def 4, we know that a filter can be expressed using a polynomial expression on L
(and if a non polynomial function is required (see previous examples: diffusion, Tikhonov denoising; or an ideal low- or high- pass filter), then a polynomial approximation can be computed)

From $h(\lambda) \simeq \sum_{m=0}^{p} a_m \lambda^m$ we have that $\underline{\underline{H}} \simeq \sum_{m=0}^{p} a_m \underline{\underline{L}}^m$.

This expression bypasses the diagonalisation, and a straightforward computation of this formula will require O(p|E|) operations only, because L **is non zero only on the edges of G.**

- **Distributed implementations of graph filters:**

  - Important: **operator** L **is local** on the graph, i.e., at node v, its uses only the values of the nodes adjacent to v



  - If h is a polynomial of order K (i.e., MA filter), a simple distributed implementation of $h(L) = \sum_{k=0}^{K} h_k L^k$:
    - Iterate $K+1$ times the operations at each node v
    - - stored from previous step in v: $(L^{k-1} \cdot x)_v$ and $h^{(k-1)}(L)_v$
    - - transmit this current value to all neighbors
    - - update: $(L^k \cdot x)_v = L \cdot (L^{k-1} \cdot x)$
    - - update $h^{(k}(L)_v = h_k (L^k \cdot x)_v + h^{(k-1)}(L)_v$
  - (More refined version: [Shuman et al., IEEE TSP 2011])

- **Better approximation bases:**

Depending of the function h to be approximated, a better basis can be more adapted.
Example: Chebyshev polynomials are often used [see Shuman et al., 2011]

- **Topic 3: Shift operators to define GSP**

Other authors prefer to define GSP not from harmonic analysis and the Laplacian, but from the definition of **a graph shift operator R,** usually taken to be 0 between 2 non-adjacent nodes, and something non zero if 2 nodes are connected.
Example: R could be A, or L, or the normalized Laplacian, or....

The sparsity pattern of R captures the topology of G.

Then, one can define almost the same theory from this operator:

- the eigendecomposition if R gives the Fourier transform on the graph

$$\begin{cases} \underline{\underline{R}} = \underline{\underline{U}}\,\underline{\underline{\Lambda}}\,\underline{\underline{U}}^T \text{ with } \underline{\underline{\Lambda}} \text{ diagonal} \\ \mathcal{F}_G \underline{x} = \underline{\underline{U}}^T \underline{x} \end{cases}$$

- Filters are defined as functions which commute with the Shift operator:

$$\underline{\underline{H}} \text{ s.t. } \underline{\underline{H}}\,\underline{\underline{R}} = \underline{\underline{R}}\,\underline{\underline{H}}$$

Like in classical Digital Signal Processing (H is then a linear shift invariant system).

- Consequence (as seen above): filters are polynomials of R and are diagonal in the Fourier domain.

- Advantages: it works naturally for directed graphs, using A (adjacency matrix),

And the shift operator A as a natural interpretation: it sends the value to the neighbors and they are then summed (possibly averaged is A is normalized properly)

- Drawback: the notion of **frequency** is only defined a posteriori, computing some variation function

$$\text{Ex: } V_{\underline{\underline{A}}}(\underline{x}) = \left\| \underline{x} - \frac{1}{|\max_k \lambda_k(\underline{\underline{A}})|}\,\underline{\underline{A}}\,\underline{x} \right\|$$

More information: [Sandryhaila and Moura, 2013 and 2014]

Topic 4: Random Graph Processes

Up to here, we have considered deterministic models of signals on graphs.
Like in classical Signal processing, we also want to consider **random processes** on graphs.

Classical random processes:

From $\underline{x}$ where each $\underline{x}_t$ is a random variable, we consider:

- its covariance: $\underline{\underline{C}}_x = \mathbb{E}\{\underline{x}\,\underline{x}^T\} - \underline{m}_x \underline{m}_x^T$ (with $\underline{m}_x = \mathbb{E}\{\underline{x}\}$)

or its autocorrelation $\mathbb{E}\{\underline{x}\,\underline{x}^T\}$

- its power spectral density: $\underline{\underline{S}}_x = \mathbb{E}\{|\mathbb{F}\underline{x}|^2\}$

Let's now define stationary processes (in the weak sense) such that $(\forall \tau)\quad (\underline{\underline{C}}_x)_{u,t} = (\underline{\underline{C}}_x)_{u+\tau,\,t+\tau}$

meaning that a shift in time does not change the process.

Now, the following results hold:

- a process is stationary iff $\underline{\underline{S}}_x$ is a diagonal matrix, hence $\underline{\underline{S}}_x$ has values $S_x(\nu)$ only (ν: freq.)
- In that case, we have that $\underline{\underline{C}}_x$ has a Toeplitz structure using the function $C_x(\tau) = \mathbb{E}\{x_t\, x_{t+\tau}\}$

and the Wiener-Khintchin theorem tells that $\boxed{\mathbb{F} C_x = S_x}$.

- (More elaborated) $\underline{x}$ can be represented as the output of a Linear Time-Shift-Invariant system, with a white noise input ($\mathbb{E}\{e_u e_t\} = \sigma\,\delta_{ut}$).
(Wold decomposition theorem).

- Being Toeplitz (and circulant), $\underline{\underline{C}}_x$ is diagonalized by Fourier transform

$$\boxed{\underline{\underline{C}}_x = \underline{\underline{\mathbb{F}}}\ \mathrm{diag}(\underline{\underline{S}}_x)\ \underline{\underline{\mathbb{F}}}^T}$$

Motivation

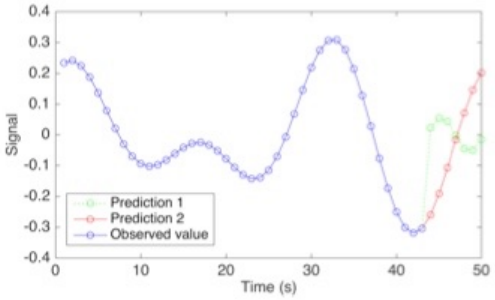


Figure: Signal prediction. The red curve is more likely to occur than the green curve because it respects the frequency statistics of the blue curve.

from [Perraudin, 2016]

Now in the case of GSP:
all these results hold in the same way

- White random process on $\mathcal{G}$: $\underline{n}$ if $\mathbb{E}\{\underline{n}\,\underline{n}^T\} = \underline{I_N}$
- Def $\underline{x}$ is weakly stationary on $\mathcal{G}$ with respect to $\underline{R}$ (shift operator) iif it can be written as the output of a linear shift-invariant graph filter $\underline{H} = \sum_{l=0}^{q} h_l \underline{R}^l$ to a zero-mean white input.

Hence $\underline{x} = \underline{H}\,\underline{n}$ and $\underline{C}_{\underline{x}} = \mathbb{E}\{\underline{x}\,\underline{x}^T\}$

$$= \mathbb{E}\{\underline{H}\,\underline{n}\,(\underline{H}\,\underline{n})^T\} = \underline{H}\,\mathbb{E}\{\underline{n}\,\underline{n}^T\}\,\underline{H}^T = \underline{H}\,\underline{H}^T$$

- Prop (could be Def) For a weakly stationary process on $\mathcal{G}$ w.r.t. $\underline{R}$, the 2 equivalent properties hold:
  1) $\underline{C}_{\underline{x}}$ and $\underline{R}$ are simultaneously diagonalizable
  2) $\mathbb{E}\{(\underline{R}^a\underline{x})((\underline{R}^T)^b\underline{x})^T\} = \mathbb{E}\{(\underline{R}^{a+c}\underline{x})((\underline{R}^T)^{b-c}\underline{x})^T\}$ for a,b,c in $\mathbb{N}$ and $c \le b$.

- Power Spectral Density: for such a process, with $\underline{R} = \underline{U}\,\underline{\Lambda}\,\underline{U}^T$ diagonalizable, the PSD is defined as $\underline{p} = \mathrm{diag}(\underline{U}^T\underline{C}_{\underline{x}}\underline{U})$ and is ≥ 0.

This means that $\underline{C}_{\underline{x}} = \underline{U}\,\mathrm{diag}(\underline{p})\,\underline{U}^T$.

The covariance matrix is diagonalized by the GFT.

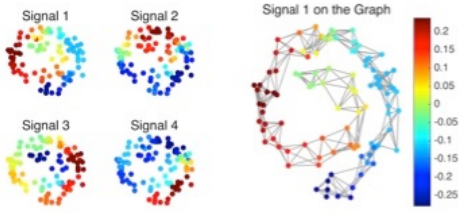
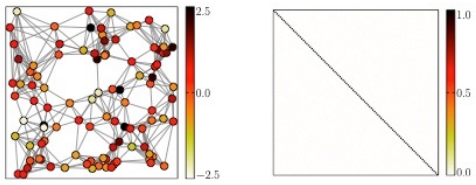


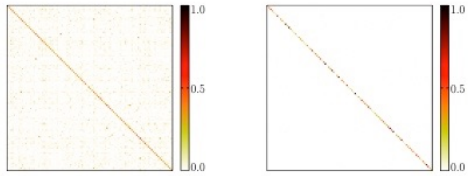
Figure: Example of a stationary process on a graph.

Some Illustrations: [Girault et al., EUSIPCO 2015]
On synthetic data



(a) A realisation. (b) Empirical S.

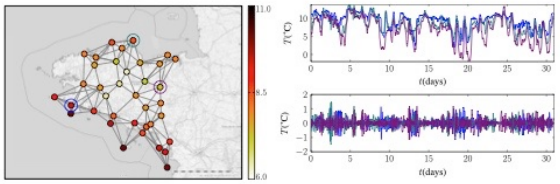
Fig. 1: White noise on a graph with $\sigma^2 = 1$ and zero mean. (a) shows a realisation of this white noise, and (b) the empirical spectral correlation matrix computed using 50k realisations.



(a) R. (b) S.

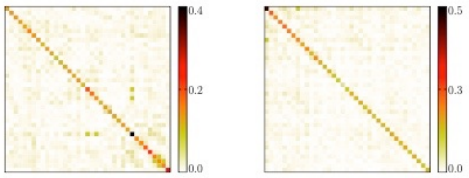
Fig. 2: Empirical second moments of a WSS signal with zero mean and non uniform spectral power with 50k realizations.

On real data



(a) Mean temperature. (b) Temperature variations.

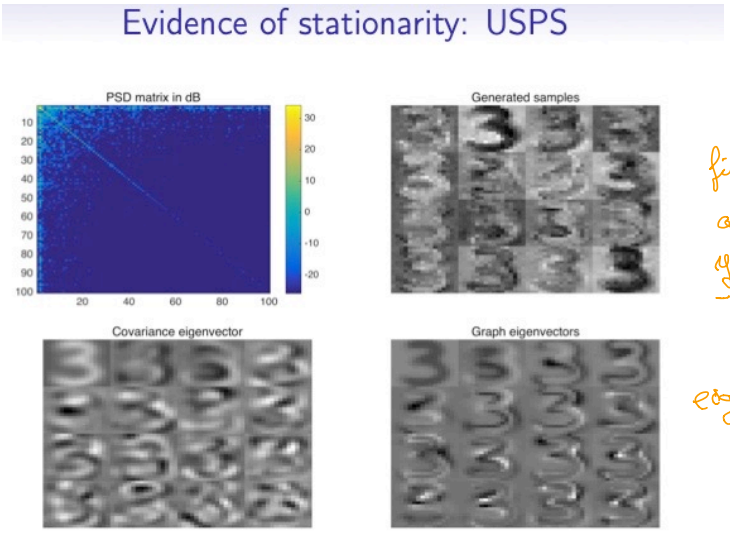
Fig. 3: Spatial and temporal variations of temperatures in Brittany in January 2014. (a) mean temperature of each station. (b) temporal variations of three stations circled in (a) on top, and first IMF of these time series below.



(a) $|R_{\mathbf{Y}}|$. (b) $|S_{\mathbf{Y}}|$.

Fig. 4: Empirical spectral correlation matrices of $\mathbf{Y}$.

[Perraudin et al., 2016] **Example for inpainting of data**
studying number 3 with 20-neighbours graph



$U^T C_x U$

Samples using filter $h(\lambda_k) = p(\lambda_k)$ on the diagonal, $y = U \Sigma$ (Σ iid Gaussian noise)

eig of C_x

eig of L

Instead of the grid of pixels.

Each pixel has feature $f \in \mathbb{R}^N$ (N images), then a NN graph w/ heat kernel on f's.

Inpainting = Interpolation between pixels or points, or nodes

Method: **Wiener filter** (or optimal filter)

if $\underline{y} = \underline{\underline{H}}\,\underline{x} + \underline{w}$. W.S.S with PSD $s(\lambda)$ for x

. $\underline{w}$ additive uncorrelated noise. or PSD $m(\lambda\ell)$

To estimate $\underline{x}$ optimally from $\underline{y}$, i.e. minimizing the expected quadratic error: $\mathbb{E}\{\|\hat{x} - x\|_2^2\}$ with a linear estimate $\hat{\underline{x}} = \underline{\underline{G}}\,\underline{y}$,

one finds the solution easily in the frequency domain.

Sum Error in frequency = Sum error in nodes

Then $e(\lambda_\ell) = \mathbb{E}\{|\mathbb{F}_G \hat{x}(\lambda\ell) - g(\lambda\ell)\,\tilde{y}(\lambda\ell)|^2\}$ and $\tilde{y}(\lambda\ell) = h(\lambda\ell)\tilde{x}(\lambda\ell) + \tilde{w}(\lambda\ell)$

$= (1 - g(\lambda_\ell)h(\lambda\ell))^2 \underbrace{\mathbb{E}\{|\tilde{x}(\lambda\ell)|^2\}}_{s(\lambda\ell)} + g(\lambda\ell)^2 \underbrace{\mathbb{E}\{|\tilde{w}(\lambda\ell)|^2\}}_{= m(\lambda_\ell)}$

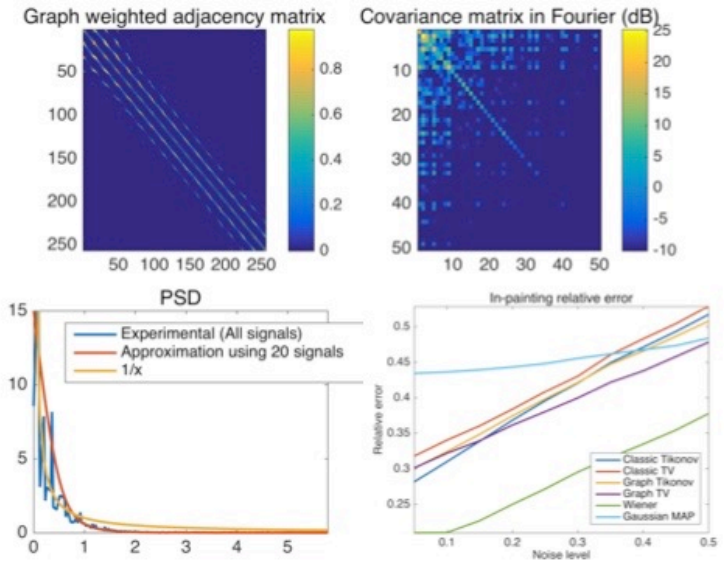
using orthogonality $\mathbb{E}\{\tilde{x}(\lambda\ell)\,\tilde{w}(\lambda\ell)\} = 0$

This leads to $\boxed{g(\lambda\ell) = \dfrac{h(\lambda\ell)\,s(\lambda\ell)}{h^2(\lambda\ell)\,s^2(\lambda\ell) + m(\lambda\ell)}}$ the **Wiener Filter**!

$$G(f) = \frac{H(f)^* S(f)}{|H(f)|^2 S(f) + N(f)}$$

with $SNR = \frac{S}{N}$

USPS digits inpainting



Second example of inpainting [Perraudin et al., 2016] (also [Girault, ENSL, 2015])

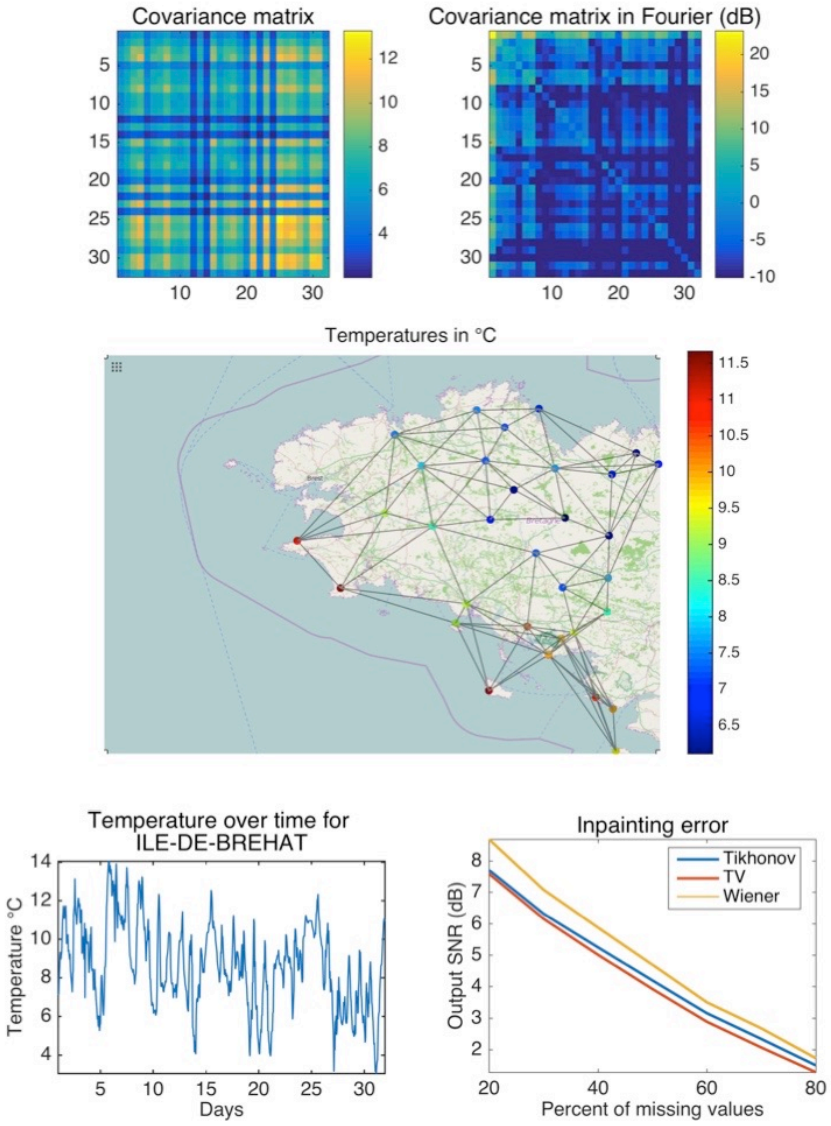


Figure 10: Experiments on the temperature of Molene. Top: Covariance matrices. Bottom left: A realization of the stochastic graph signal (first measure). Bottom center: the temperature of the Island of Brehat. Bottom right: Recovery errors for different noise levels.

Topic 5: Sampling of graphs

- Sampling and interpolation are cornerstone problems in classical SP
  - ⇒ How to find $\mathbf{x} \in \mathbb{R}^N$ using less than N observations?
  - ⇒ Need to limit the degrees of freedom: subspace, smoothness

- Graph signals: sampling thoroughly investigated
  - ⇒ Most works assume only a few values are observed
  - ⇒ [Anis14, Chen15, Tsitsvero15, Puy15, Wang15]

In a nutshell, to compare with classical signals: [Tanaka et al., Sampling signals on graphs, 2020]

Principle:

Illustrative example of sampling procedures

In Fig. 3 (left) standard discrete-time sampling is shown in both time and frequency domains. Point-wise sampling in the time domain corresponds to folding of the DFT spectrum [1]. Note that both sampling methods yield the same output after the inverse DFT of the frequency sampled signal. Fig. 3 (right) illustrates graph signal sampling in vertex and graph frequency domains (Definitions 2 and 3), which do not yield the same output, unlike their conventional signal counterparts of Fig. 3 (left).

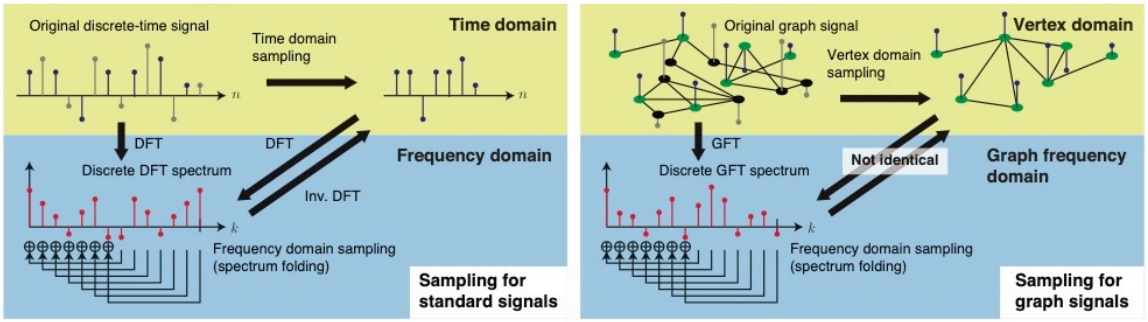


Fig. 3: Sampling procedures for standard and graph signals.

Example of sampling, bandlimited or not:

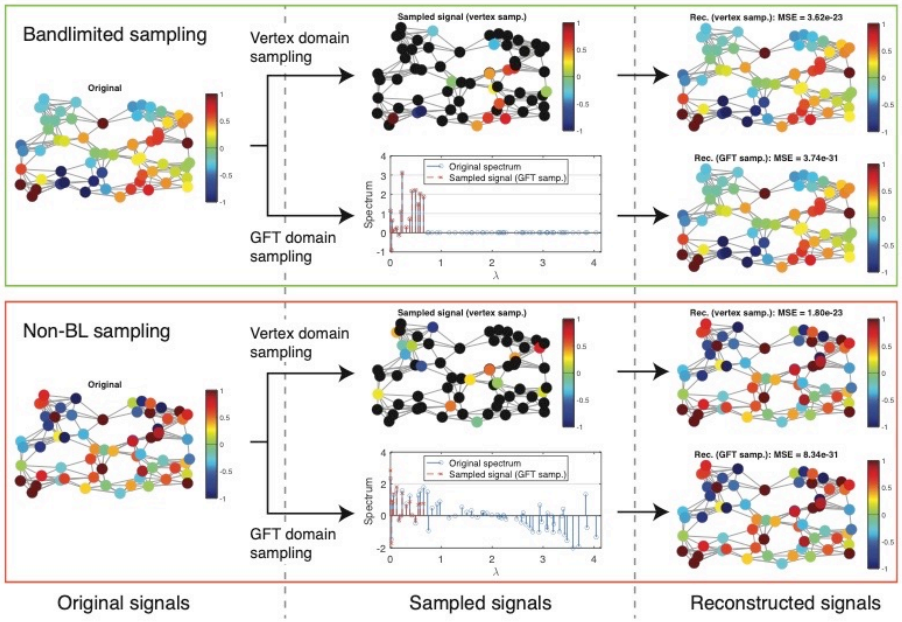


Fig. 4: Sampling examples for signals on a random sensor graph with $N = 64$. The sample $\mathbf{c}$ has length $M = 15$. Top: Bandlimited sampling and recovery, where the signal is bandlimited with $K = 15$ and the sampling filter is the identity matrix. Bottom: Sampling and recovery of the graph signal lying in a known subspace, where the original signal is generated by the PGS model [8] with generator function $\hat{a}(\lambda_i) = 1 - 2\lambda_i/\lambda_{\max}$ and the sampling function is $\hat{g}(\lambda_i) = \exp(-\lambda_i/2)$. Even in this case, the original signal is perfectly recovered from $\mathbf{c}$ by using both vertex and graph frequency domain sampling without changing the framework. GSPBOX (https://epfl-lts2.github.io/gspbox-html/) is used for obtaining the random sensor graph.

Random sampling:

Definition (Bandlimited graph signal) A k-bandlimited signal $\boldsymbol{x} \in \mathbb{R}^N$ on $\mathcal{G}$ is a signal that satisfies, for some $\boldsymbol{\alpha} \in \mathbb{R}^k$

$$\boldsymbol{x} = \mathrm{U}_k \boldsymbol{\alpha} = \sum_{i=1}^{k} \alpha_i \boldsymbol{u}_i$$

Problem (Nyquist-Shannon equivalent for sampling graph signals) : how to efficiently sample a (hopefully) small number of nodes for perfect reconstruction ?

A possible solution : random sampling [Puy '16]

1. Associate to each node i a probability $\boldsymbol{p}_i = \left\| \mathrm{U}_k^\intercal \boldsymbol{\delta}_i \right\|_2^2 / k$ to draw this node.
2. Draw the sampling set $\mathcal{S}$ of size $|\mathcal{S}|$ independently w/ replacement from $\boldsymbol{p}$.
3. Theorem (Restricted Isometry Property) With high probability, reconstruction (up to the noise) is perfect, provided that
$$|\mathcal{S}| \geqslant \mathcal{O}(k \log k).$$
4. Regularized decoder : $\boldsymbol{x}_{\text{rec}} = \min_{\boldsymbol{z} \in \mathbb{R}^N} \left\| \mathrm{P}_\Omega^{-1/2} (\mathrm{M}\boldsymbol{z} - \boldsymbol{y}) \right\|_2^2 + \gamma\, \boldsymbol{z}^\intercal \mathrm{L}^r \boldsymbol{z}$

Other sampling schemes exist : greedy [Chamon '17, Anis '16, Chen '16], loop-erased random walks [Tremblay '17a], DPP-based sampling [Tremblay '17b], etc.

Comparisons:

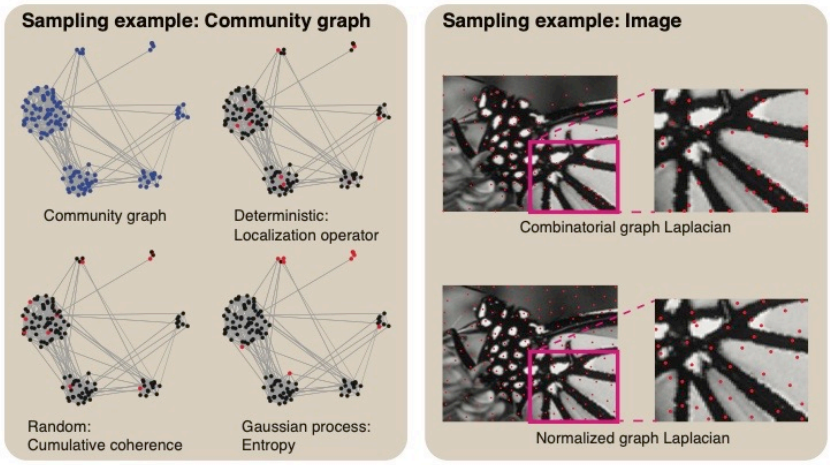


Fig. 5: Comparison of sampling sets. (Left) Sampling sets for a community graph with $N = 256$, where 10 nodes are selected. The sizes of the clusters are $\{4, 4, 8, 16, 32, 64\}$ from the top-left cluster in a clockwise direction. Top: The original graph and sampling set by a deterministic approach [11]. Bottom: Sampling set of one realization by a random approach [27] and sampling set by a Gaussian process-based method [30]. GSPBOX (https://epfl-lts2.github.io/gspbox-html/) is used for obtaining the cluster graph. (Right) Sampling sets for image pixels. Each pixel is a node, and edge weights are chosen based on the method in https://github.com/STAC-USC/NNK_Image_graph. Sampling set selection based on maximizing cut-off frequency [10] is used. Enlarged portions are shown for better visualization. Left and right: Sampling sets using the combinatorial and normalized graph Laplacians, respectively.