Stationary signal processing on graphs

Nathanaël Perraudin and Pierre Vandergheynst [*]

January 9, 2016

Abstract

Graphs are a central tool in machine learning and information processing as they allow to conveniently capture the structure of complex datasets. In this context, it is of high importance to develop flexible models of signals defined over graphs or networks. In this paper, we generalize the traditional concept of wide sense stationarity to signals defined over the vertices of arbitrary weighted undirected graphs. We show that stationarity is intimately linked to statistical invariance under a localization operator reminiscent of translation. We prove that stationary graph signals are characterized by a well-defined Power Spectral Density that can be efficiently estimated even for large graphs. We leverage this new concept to derive Wiener-type estimation procedures of noisy and partially observed signals and illustrate the performance of this new model for denoising and regression.

Index terms— Stationarity, graphs, spectral graph theory, power spectral density, Wiener filter, covariance

1 Introduction

Stationarity is a traditional hypothesis in signal processing used to represent a special type of statistical relationship between samples of a time series. For instance, wide-sense stationarity assumes that the first two statistical moments are invariant under translation. In other words, we can expect statistically similar consequences from the same causes. Stationarity is a corner stone of many signal analysis methods. The expected frequency content of stationary signals, called Power Spectral Density (PSD), provides an essential source of information used to build signal models, generate realistic surrogate data or perform predictions. In Figure 1, we present an example of a stationary process (blue curve) and two predictions (red and green curves). As the blue signal is a realization of a stationary process, the red curve is more probable than the green one because it respects the frequency content of the observed signal. Classical stationarity is a statement of statistical regularity under arbitrary translations and thus requires a regular structure (often "time"). However many signals do not live on such a regular structure. Imagine that instead of having one sensor returning a time-series, we have multiple sensors living in a two-dimensional space, each of which delivers only one value. In this case (see Figure 2 left), the

[*]EPFL, Ecole Polytechnique Fédérale de Lausanne, LTS2 Laboratoire de traitement du signal, CH-1015 Lausanne, Switzerland

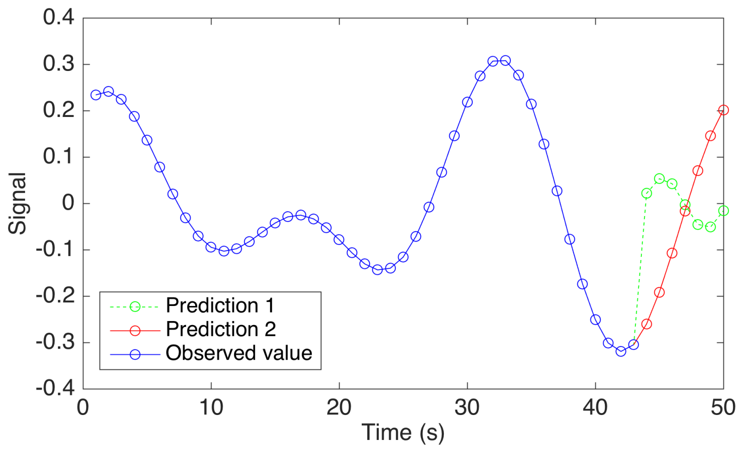


Figure 1: Signal prediction. The red curve is more likely to occur than the green curves because it respects the frequency statistics of the blue curve.

signal support is no longer regular. Since there exist an underlying continuum in this example (2D space), one could assume the existence of a 2D stationarity field and use Kriging [1] to interpolate observations to arbitrary locations. This technique thus generalizes stationarity for a regular domain but irregularly spaced samples. The goal of this contribution is to generalize stationarity for an irregular domain that is represented by a graph, without resorting to any underlying regular continuum. Instead we use a weak notion of translation invariance that captures the structure (if any) of the samples set. Graphs are convenient data structures able to capture complicated topologies and are used in numerous domains of applications such as social, transportation or neuronal networks. In this work a graph is composed of vertices connected by (weighted but undirected) edges. As we shall recall, a graph possess a localization operator that generalizes the classical translation. Signals are now scalar values observed at the vertices of the graph.

Figure 2 (left) presents an example of some data living in a 2-dimensional space. Seen as scattered samples of an underlying 2D process, one would (rightly) conclude it is not stationary. However, under closer inspection, the observed values look stationary *within* the spiral-like structure depicted by the graph in Figure 2 (right). The traditional Kriging interpolation technique would ignore this underlying structure and conclude that there are always rapid two dimensional variations in the underlying continuum process. This problem does not occur in the graph case where the statistical relationships inside the data follow the graph connections (edges) resulting in this case in signals oscillating smoothly over the graph at low frequency. A typical example of a stationary signal on a graph would be the result of a survey performed by the users of a social network. If there is a relationship between a user's answers and those of his neighbours, this relationship is expected to be constant among all users. Using stationarity on the graph, we could build the most probable answer for users that did not answer the survey.

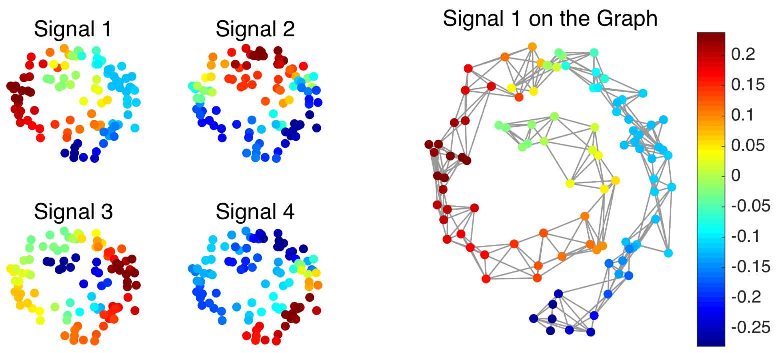


Figure 2: Example of a stationary process on a graph. The graph connections express relationships between the different elements of one signal. In this case, the signal varies smoothly along the snail shape of the graph.

1.1 Contributions

We use spectral graph theory to extend the notion of stationarity to a broader class of signals. To do so, we establish in Section 3 the theoretical basis of this extension. In order to evaluate the usefulness of our new model, we generalize Wiener filters [2, 3]. Based on our generalization of Wiener filters, we propose a new regularization term for graph signal optimization instead of the traditional Dirichlet prior. This term depends on the noise level and on the PSD of the signal. The new optimization scheme presented in Section 4 has two main advantages: 1) it allows to get rid of an arbitrary regularization parameter and 2) it adapts to the data. We also prove that the new optimization model is optimal under mild hypotheses.

Since Wiener filters use the power spectral density (PSD), we generalize the Welch method [4, 5] in Section 5 and obtain a scalable way to estimate the PSD.

Finally, in Sections 6 and 7, we show experimentally that common datasets closely follow our stationarity assumption. We exploit this fact to perform missing data imputation and we suggest that nearest neighbor graph construction often leads to stationarity.

1.2 Review of the literature

Graphs have been used as regularizers in data application for more than a decade [6, 7, 8, 9] and two of the most used models will be presented in Section A. The idea of graph filtering was hinted at by the machine learning community [10] but developed for the spectral graph wavelets proposed by Hammond et al. [11] and extended by Shuman et al. in [12]. Moura and collaborators have suggested to use the adjacency matrix rather than the Laplacian as the basis of graph filtering [13].

A notion of stationarity on graphs has been recently proposed in [14, 15]. However these contributions use a completely different translation operator, promoting unitarity over localization, and therefore end up with a fairly different framework. Finally, we also note that a probabilistic model using Gaussian random fields has been proposed in [16, 17]. In this model, signals are automatically graph stationary. A detailled explanation is given at the end of section 3.

2 Background results

2.1 Graph signal processing

Graph nomenclature A graph consists of two sets: $\mathcal{V}, \mathcal{E}$ and a weight function. $\mathcal{V}$ is the set of vertices representing the nodes of the graph and $\mathcal{E}$ is the set of edges that connect two nodes if there is a particular relation between them. In this work all graphs are undirected. To obtain a finer structure, this relation can be quantified by a weight function $\mathcal{W} : \mathcal{V} \times \mathcal{V} \to \mathbb{R}$ that reflects to what extent two nodes are related to each other. A graph is therefore a tuple denoted by $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{W}\}$. Let us index the nodes from $1, \dots, N = |\mathcal{V}|$ and construct the weight matrix $\mathbf{W} \in \mathbb{R}^{N \times N}$ by setting $W_{i,j} = \mathcal{W}(v_i, v_j)$ as the weight associated to the edge connecting the node i and the node j. When no edge exists between i and j, the weight is set to 0. For a node $v_i \in \mathcal{V}$, the degree $d(i)$ is defined as $d(i) = \sum_{j=1}^{N} \mathbf{W}(i,j)$. In this framework, a signal is defined as a function $f : \mathcal{V} \to \mathbb{R}$ (or $\mathbb{C}$) assigning a scalar value to each vertex. It is convenient to consider a signal f as a vector of size N with the n^{th} component representing the signal value at the n^{th} vertex.

The most fundamental operator in graph signal-processing is the (combinatorial) graph Laplacian, defined as:

$$\mathcal{L} = \mathbf{D} - \mathbf{W},$$

where $\mathbf{D}$ is the diagonal degree matrix ($\mathbf{D}_{ii} = d[i]$).

Spectral theory Since the Laplacian $\mathcal{L}$ is always a symmetric positive semi-definite matrix, we know from the spectral theorem that it possesses a complete set of orthonormal eigenvectors. We denote them by $\{u_\ell\}_{\ell=0,1,\dots,N-1}$. For convenience, we order the set of real, non-negative eigenvalues as follows: $0 = \lambda_0 < \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_{N-1} = \lambda_{\max}$. When the graph is connected[1], there is only one zero eigenvalue. In fact, the multiplicity of the zero eigenvalue is equal to the number of connected components. See for example [18, 19] for more details on spectral graph theory. The eigenvectors of the Laplacian are used to define a graph Fourier basis [6, 12] which will be denoted as U. The eigenvalues are considered as a generalization of squared frequencies. The Laplacian matrix can thus be decomposed as

$$\mathcal{L} = U \Lambda U^*,$$

where U^* denotes the transposed conjugate of U. The graph Fourier transform is written $\hat{f} = \mathcal{F}(f) = U^* f$ and its inverse $f = \mathcal{F}^{-1}(\hat{f}) = U\hat{f}$. This Graph Fourier Transform possesses interesting properties further studied in [12].

Graph filters The graph Fourier-transform plays a central role in graph signal processing since it allows a natural extension of filtering operations. In the classical setting, applying a filter to a signal is carried out with a convolution,

[1]a path connects each pair of nodes in the graph

which is simply a point-wise multiplication in the spectral domain. Similarly, filtering a graph signal is also a multiplication in the graph Fourier domain. A graph filter g is defined as a continuous function $g : \mathbb{R}_+ \to \mathbb{R}$. In the spectral domain, filtering a signal f with a filter g is therefore written as $\hat{f}'[\ell] = g(\lambda_\ell) \cdot \hat{f}[\ell]$ or in matrix notation

$$f' = Ug(\Lambda)U^* f = g(\mathcal{L})f,$$

where $\hat{f}'$ is the Fourier transform of the filtered signal f', g the filter and $\hat{f}$ the Fourier transform of the signal f. Equivalently, a filter defines the following matrix-valued function:

$$g(\mathcal{L}) := Ug(\Lambda)U^*,$$

where $g(\Lambda)$ is a diagonal matrix with entries $g(\lambda_\ell)$. It is worth noting that these formulas make explicit use of the Laplacian eigenvectors and thus its diagonalization. The complexity of this operation is in general $\mathcal{O}(N^3)$. In order to avoid this, there exist fast filtering algorithms based on Chebyshev polynomials or the Lanczos method [11, 20]. These methods scale with the number of edges $|E|$ and reduce the complexity to $\mathcal{O}(|E|)$, which is advantageous in the case of sparse graphs.

Localization operator Because most graphs do not possess a regular structure, we cannot translate a signal around the vertex set. This is problematic since translation plays a central role in stationarity. In order to overcome this issue, we use the localization operator defined in [11, 12], which is simply the convolution with a Kroneker delta. Localizing a filter g onto node i reads:

$$\mathcal{T}_i g[n] = \sum_{\ell=0}^{N-1} g(\lambda_\ell)\overline{u_\ell}[i]u_\ell[n] = \left(g(\mathcal{L})\right)_{in}, \tag{1}$$

where $\overline{u_\ell}$ is the complex conjugate of u_ℓ. For a sufficiently regular function g, it has been proved in [12] that this operation localizes the filter g around the vertex i. In the classical case, the concept of translation is equivalent to localization since the convolution with a Kroneker delta is precisely a translation. However, for irregular graphs, localization differs from translation because the shape of the localized signal adapts to the graph and varies as a function of its topology.

2.2 Stationarity for time series

Let $\mathbf{x}(t)$ be a time indexed stochastic process. Throughout this contribution, all stochastic variables will be written in bold fonts.

Definition 1 (Time Wide-Sense Stationarity)**.** *A signal is Time Wide-Sense Stationary (WSS) if its first two statistical moments are invariant under translation, i.e:*

1. $\mathbb{E}\big(\mathbf{x}(t)\big) = M,$
2. $\mathbb{E}\big(\mathbf{x}(t)\mathbf{x}^*(s)\big) = \gamma_{\mathbf{x}}(t-s)$

where $\gamma_{\mathbf{x}}$ *is the autocorrelation function of* $\mathbf{x}$.

For a WSS signal, the autocorrelation function depends only on one parameter, $t-s$, and is linked to the Power Spectral Density (PSD) through the Wiener-Khintchine Theorem [2]. The latter states that the PSD of the stochastic process $\mathbf{x}$ denoted $S_{\mathbf{x}}(\omega)$ is the Fourier transform of its auto-correlation :

$$S_{\mathbf{x}}(\omega) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} \gamma_{\mathbf{x}}(t)e^{-i\omega t}\mathrm{d}t. \tag{2}$$

As a consequence, when a signal is filtered, its PSD is multiplied by the energy of the filter: for $\mathbf{y} = h * \mathbf{x}$, we have

$$S_{\mathbf{y}}(\omega) = \left|\hat{h}(\omega)\right|^2 S_{\mathbf{x}}(\omega),$$

where $\hat{h}(\omega)$ is the Fourier transform of the filter. For more information about stationarity, we refer the reader to [21].

Note that equivalent definitions exist for continuous, discrete and periodic signals. When generalizing these concepts to graphs, the underlying structure for the stationarity will no longer be time but graph vertices.

3 Stationarity of graph signals

Let $\mathbf{x} \in \mathbb{R}^N$ be a stochastic process with a finite number of variables indexed by the vertices of a weighted undirected graph. The expected value of each variable is written $\mathbb{E}\big(\mathbf{x}[i]\big)$ and the covariance matrix of the process is $\Sigma_{\mathbf{x}}[i,j] = \mathbb{E}\big(\mathbf{x}^*[i]\mathbf{x}[j]\big)$. Note that the covariance $\Sigma_{\mathbf{x}}$ can obviously be estimated empirically with no information about the graph itself via

$$\Sigma_{\mathbf{x}} \approx \tilde{X}\tilde{X}^\top / N_s,$$

where X are N_s instances of the stochastic processes stacked together columnwise and $\tilde{X} = X - \mathrm{mean}(X)$ is the centered process. For discrete time WSS processes, the covariance matrix $\Sigma_{\mathbf{x}}$ is Toeplitz, or circulant for periodic boundary conditions, reflecting translation invariance. In that case, the covariance is diagonalized by the Fourier transform. We are now going to generalize this property to take into account the intricate graph structure.

Definition 2. *A stochastic process* $\mathbf{x}$ *defined on the vertices of a graph* $\mathcal{G}$ *is called Graph Wide-Sense (or second order) Stationary (GWSS), if and only if it satisfies the following properties :*

1. *its first moment is constant over the vertex set, i.e* $\mathbb{E}\big(\mathbf{x}[i]\big) = M$ *and*
2. *its covariance matrix* $\Sigma_{\mathbf{x}}[i,j] = \mathbb{E}\big(\mathbf{x}[i]\mathbf{x}[j]\big)$ *is jointly diagonalizable with the Laplacian of* $\mathcal{G}$.

The first part of the above definition is equivalent to the first property of time WSS signal. The requirement for the second moment is a natural generalization where we are imposing the use of the graph Fourier transform but seems to obscure any notion of translation invariance. We shall now see that this condition is equivalent to imposing that the covariance is invariant under the localization operator. The easiest way to observe this fact is to generalize the Wiener Khintchine Theorem 2, which makes the link between the localization operator and the covariance matrix.

Theorem 1. *If a signal is GWSS, its covariance matrix is given by graph localization of a kernel $\gamma_{\mathbf{x}}$, i.e:*

$$\mathbb{E}(\mathbf{x}[i]\mathbf{x}[j]) = \Sigma_{\mathbf{x}}[i,j] = \mathcal{T}_i\gamma_{\mathbf{x}}(j) = (\gamma_{\mathbf{x}}(\mathcal{L}))_{ij} \tag{3}$$

Proof. By Definition 2, the covariance matrix of a GWSS signal can be written as:

$$\Sigma_{\mathbf{x}} = U\Gamma_{\mathbf{x}}U^* \tag{4}$$

where $\Gamma_{\mathbf{x}}$ is a diagonal matrix. For any filter $\gamma_{\mathbf{x}}$ satisfying $\gamma_{\mathbf{x}}(\lambda_\ell) = \Gamma_{\mathbf{x}}(\ell,\ell)$, we observe that $\Sigma_{\mathbf{x}}$ is the graph localized version of $\gamma_{\mathbf{x}}$. $\square$

The choice of $\gamma_{\mathbf{x}}$ in this result is somewhat arbitrary but we shall soon see that we are interested in localized kernels. In that case, $\gamma_{\mathbf{x}}$ will be typically be the lowest degree polynomial satisfying the constraints and can be constructed using Lagrange interpolation for instance.

Theorem 1 provides a fundamental information about the covariance. The size of the correlation (distance over the graph) depends on the size of the support of localized the kernel $\mathcal{T}_i\gamma_{\mathbf{x}}$. In [12] (Theorem 1 and Corrolary 2), it has been proved that the size of $\mathcal{T}_i\gamma_{\mathbf{x}}$ depend on the regularity of $\gamma_{\mathbf{x}}$. For example, if $\gamma_{\mathbf{x}}$ is polynomial of degree K, it is exactly localized in a ball of radius K. Hence we will be mostly interested in such low degree polynomial kernels.

The graph power spectral density (PSD) matrix of a stochastic process is given by $\Gamma_{\mathbf{x}} = U^*\Sigma_{\mathbf{x}}U$. For a GWSS signal this matrix is diagonal and the graph PSD of $\mathbf{x}$ becomes:

$$\gamma_{\mathbf{x}}(\lambda_\ell) = (U^*\Sigma_{\mathbf{x}}U)_{\ell,\ell}\,. \tag{5}$$

Table 1 present the difference and the similarities between the classical and the graph case.

Example 1 (Gaussian noise)**.** *Gaussian noise is WSS for any graph. Indeed, the first moment is $\mathbb{E}\big(\mathbf{x}[k]\big) = 0$. Moreover, the covariance matrix can be written as $I = \Sigma_{\mathbf{x}} = UIU^*$ with any orthonormal matrix U and thus is diagonalizable with any graph Laplacian. We also observe that the PSD is constant, which implies that similar to the classical case, white noise contains all "graph frequencies".*

When $\gamma_{\mathbf{x}}$ is a bijective function, the covariance matrix contains an important part of the graph structure: the laplacian eigenvectors. On the contrary, if $\gamma_{\mathbf{x}}$ is not bijective, some of the graph structure is lost as it is not possible to recover all eigenvectors. This is for instance the case when the covariance matrix is low-rank. As another example, let us consider completely uncorrelated samples. In this case, the covariance matrix becomes $\Sigma_{\mathbf{x}} = I$ and loses all graph information, even if by definition the process remains stationary on the graph.

One of the crucial benefits of stationarity is that it is preserved by linear filtering, while the PSD is simply reshaped by the filter. The same property holds on graphs.

Theorem 2. *When a graph filter g is applied to a GWSS process, the result remains GWSS and the PSD satisfies:*

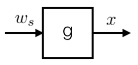
$$\gamma_{g(\mathcal{L})\mathbf{x}}[\ell] = g^2(\lambda_\ell)\cdot\gamma_{\mathbf{x}}[\ell] \tag{6}$$

Proof. The output of a filter g can be written $\mathbf{x}' = g(L)\mathbf{x}$. If the input signal $\mathbf{x}$ is GWSS, we can check easily that the first moment of the filter's output is constant, $\mathbb{E}(g(L)\mathbf{x}[i]) = g(0)M$. The computation of the second moment gives:

$$\begin{aligned}
\mathbb{E}\left(g(\mathcal{L})x\big(g(\mathcal{L})x\big)^*\right) &= g(\mathcal{L})\mathbb{E}\left(xx^*\right)g(\mathcal{L}) \\
&= g(\mathcal{L})\Sigma_{\mathbf{x}}g(\mathcal{L}) \\
&= Ug^2(\Lambda)\gamma_{\mathbf{x}}(\Lambda)U^*.
\end{aligned}$$

□

Theorem 2 provides a simple way to artificially produce stationary signals with a presecribed PSD by simply filtering white noise.



The resulting signal will be stationary with PSD g^2.

	Classical	Graph
Stationary with respect to	Translation	The localization operator
First Moment	$\mathbb{E}\big(\mathbf{x}[i]\big) = M$	$\mathbb{E}\big(\mathbf{x}[i]\big) = M$
Second Moment	$\Sigma_{\mathbf{x}}[i,n] = \mathbb{E}\big(\mathbf{x}[i])\mathbf{x}^*[n]\big) = \gamma_{\mathbf{x}}[t-s]$ $\Sigma_{\mathbf{x}}$ Toeplitz	$\Sigma_{\mathbf{x}}[i,n] = \mathbb{E}\big(\mathbf{x}[i]\mathbf{x}[n]\big) = \gamma_{\mathbf{x}}(\mathcal{L})_{i,n}$ $\Sigma_{\mathbf{x}}$ diagonalizable with $\mathcal{L}$
Wiener Khintchine	$S_{\mathbf{x}}[\ell] = \frac{1}{\sqrt{N}}\sum_{i=1}^{N}\gamma_{\mathbf{x}}[n]e^{-j2\pi\frac{n\ell}{N}}$	$\gamma_{\mathbf{x}}(\lambda_\ell) = (\Gamma_{\mathbf{x}})_{\ell,\ell} = (U^*\Sigma_{\mathbf{x}}U)_{\ell,\ell}$
Result of filtering	$\gamma_{g*\mathbf{x}}[\ell] = g^2[\lambda_\ell]\cdot\gamma_{\mathbf{x}}[\ell]$	$\gamma_{g(\mathcal{L})\mathbf{x}}[\ell] = g^2(\lambda_\ell)\cdot\gamma_{\mathbf{x}}[\ell]$

Table 1: Difference and similarities between classical and graph stationarity. In the classical case, we work with a N periodic discrete signal.

Gaussian random field interpretation The framework of stationary signals on graphs can be interpreted probabilistically using Gaussian Markov Random Field (GMRF). Generalizing the generative model of Gadde and Ortega ([16] section 3), the authors of [17] assume that the signal $\mathbf{x}$ is drawn from a distribution

$$p(\mathbf{x}) = \frac{1}{Z_p}\exp\left(-\mathbf{x}^T p(\mathcal{L})\mathbf{x}\right) \tag{7}$$

where $Z_p = \int \exp\left(-\mathbf{x}^T p(\mathcal{L})\mathbf{x}\right) \mathrm{d}\mathbf{x}$. This leads to a stationary process $\mathbf{x}$ with a covariance matrix given by:

$$\Sigma_{\mathbf{x}} = \left(p(\mathcal{L})\right)^{-1} = p^{-1}(\mathcal{L}).$$

Stationarity and the GMRF probabilistic model are thus linked by PSD $= p^{-1}$.

4 Graph Wiener filters and optimization framework

Using stationary signals, we can naturally extend the framework of Wiener filters [3] largely used in signal processing for Mean Square Error (MSE) optimal linear prediction. Since the construction of Wiener filters is very similar for non-graph and graph signals, we present only the latter here. The main difference is that the traditional frequencies are replaced by the graph Laplacian eigenvalues[2] λ_ℓ. The full Wiener recovery process is summarized in Figure 3.

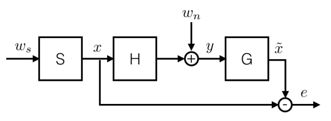


Figure 3: Wiener process model. w_s and w_n are white noises with different scaling, h is the convolution filter and g the Wiener filter. x is the original signal, y the measurements, $\tilde{x}$ the recovery and e the final error.

The Wiener filter can be used to produce a mean-square error optimal estimate of a stationary signal under a linear but noisy observation model. Let us consider a GWSS process $\mathbf{x}$ with PSD of $s^2(\lambda_\ell)$. The measurements $\mathbf{y}$ are given by:

$$y = h(\mathcal{L})x + w_n, \tag{8}$$

where h is a graph filter and w_n additive noise of PSD $n(\lambda_\ell)$.

To recover $\mathbf{x}$, we extend Wiener filters to the graph case:

$$g(\lambda_\ell) = \frac{h(\lambda_\ell)s^2(\lambda_\ell)}{h^2(\lambda_\ell)s^2(\lambda_\ell) + n(\lambda_\ell)}. \tag{9}$$

The expression above can be derived identically as in the classical case and minimizes the expected quadratic error, which can be written as:

$$e(\lambda_\ell) = \mathbb{E}\left(\hat{x}(\lambda_\ell) - \hat{\tilde{x}}(\lambda_\ell)\right)^2 = \mathbb{E}\left(\hat{x}(\lambda_\ell) - g(\lambda_\ell)\hat{y}(\lambda_\ell)\right)^2$$

Theorem 3 proves the optimality of this filter for the graph case.

Wiener optimization In this contribution, we would like to address a more general problem. Let us suppose that our measurements are generated as:

$$y = Ax_o + w_n, \tag{10}$$

where the original signal x_o has a PSD denoted $s^2(\lambda_\ell)$ and the noise w_n a PSD of $n(\lambda_\ell)$. Since the operator A is not assumed to be diagonalizable with $\mathcal{L}$, we cannot build a Wiener filter that constructs a direct estimation of the signal x. If the signal x varies smoothly on the graph, i.e is low frequency based, a classic optimization scheme would be the following:

$$\arg\min_x \|Ax - y\|_2^2 + \gamma x^t L x \tag{11}$$

[2]The graph eigenvalues are equivalent to classical squared frequencies.

This optimization scheme presents two main disadvantages. Firstly, the parameter γ must be tuned in order to remove the best amount of noise. Secondly, it does not take into account of the data structure charcterized by the PSD $s^2(\lambda_\ell)$.

Our solution to overcome these issues is to solve the following optimization problem (that we suggestively call Wiener optimization):

$$\tilde{x} = \arg\min_x \|Ax - y\|_2^2 + \|w(\mathcal{L})x\|_2^2, \tag{12}$$

where $w(\lambda_\ell)$ is the Fourier penalization weights. These weights are defined as

$$w(\lambda_\ell) = \left| \frac{\sqrt{n(\lambda_\ell)}}{s(\lambda_\ell)} \right| = \frac{1}{\sqrt{SNR(\lambda_\ell)}}.$$

Problem (12) generalizes Problem (11) which assumes implicitly a PSD of $\frac{1}{\lambda_\ell}$ and a constant noise level of γ across all frequencies. Note that this framework generalizes two main assumptions done on the data in practice:

1. The signal is smooth on the graph, i.e: the edge derivative has a small ℓ_2-norm. As seen before this is done by setting the PSD as $\frac{1}{\lambda_\ell}$.
2. The signal is band-limited, i.e: is it a linear combination of the k lowest graph Laplacian eigenvectors. This class of signal simply have a null PSD for $\lambda_\ell > \lambda_k$.

3 motivations for the optimization framework The first motivation is intuitive. The weight $w(\lambda_\ell)$ heavily penalizes frequencies associated to low SNR and vice versa.

The second and main motivation is theoretical. When A is diagonalizable, the solution to Problem (12) is the application of the corresponding Wiener filter.

Theorem 3. *If the operator A is diagonalizable with $\mathcal{L}$, (i.e: $A = a(\mathcal{L}) = Ua(\Lambda)U^*$), then problem 12 is optimal with respect of the weighting w in the sense that its solution minimizes the mean square error:*

$$\mathbb{E}\left(\|e\|_2^2\right) = \mathbb{E}\left(\|\dot{x} - x_o\|_2^2\right) = \mathbb{E}\left(\sum_{i=1}^{N} \left(\dot{x}(i) - x_o(i)\right)^2\right).$$

Additionally, the solution can be computed by the application of the corresponding Wiener filter.

The proof is given in Appendix B.

The last motivation is algorithmic and requires the knowledge of proximal splitting methods [22, 23]. Problem (12) can be solved by a splitting scheme that minimizes iteratively each of the terms. The minimization of the regularizer, i.e the proximal operator of $\|w(\mathcal{L})x\|_2^2$, becomes a Wiener de-noising operation:

$$\begin{aligned} \operatorname{prox}_{\frac{1}{2}\|w(L))x\|_2^2}(y) &= \arg\min_x \|w\mathcal{F}x\|_2^2 + \|x - y\|_2^2 \\ &= g(\mathcal{L})y \end{aligned}$$

with

$$g(\lambda_\ell) = \frac{1}{1 + w^2(\lambda_\ell)} = \frac{s^2(\lambda_\ell)}{s^2(\lambda_\ell) + n(\lambda_\ell)}.$$

Solving the problem Note that Problem (12) can be solved with a simple gradient descent. However, for a large number of nodes N, the matrix $w(\mathcal{L})$ requires $O(N^3)$ operations to be computed and $O(N^2)$ bits to be stored. This difficulty can be overcome by applying its corresponding filter operator at each iteration. As already mentioned, the cost of the approximation is $O(|E|)$ [20].

When $s(\lambda_\ell) \approx 0$ for some λ_ℓ the operator $w(\mathcal{L})$ becomes badly conditioned. To overcome this issue, Problem (12) can be solved efficiently using a forward-backward splitting scheme [24, 22, 23]. The proximal operator of $\|w(\mathcal{L})x\|_2^2$ has been given above and we use the term $\|Ax - y\|_2^2$ as the differentiable function. Algorithm 1 uses an accelerated forward backward scheme [25] to solve Problem (12) where γ is the step size (we use $\gamma = \frac{1}{2\lambda_{\max}(A)^2}$), ϵ the stopping tolerance, J the maximum number of iterations and δ is a very small number to avoid a possible division by 0.

Algorithm 1 Fast Wiener optimization to solve (12)

```
INPUT: $z_1 = x$, $u_0 = x$, $t_1 = 1$, $\epsilon > 0$
for $j = 1, \dots J$ do
    $v = A^*(Az_j - y)g(z_j)$
    $u_{j+1} = \gamma g(L)v$ with $g(\lambda) = \frac{s^2(\lambda)}{s^2(\lambda)+\gamma n(\lambda)}$
    $t_{j+1} = \frac{1+\sqrt{1+4t_j^2}}{2}$
    $z_{j+1} = z_j + \frac{t_j - 1}{t_{j+1}}(u_j - u_{j-1})$
    if $\frac{\|z_{j+1}-z_j\|_F^2}{\|z_j\|_F^2+\delta} < \epsilon$ then
        BREAK
    end if
end for
SOLUTION: $z_J$
```

5 Estimation of the signal PSD

As the PSD is central in our method, we need a reliable and scalable way to compute it. Equation (5) suggests a direct estimation method using the Fourier transform of the covariance matrix. Unfortunately, when the number of nodes is considerable, this method requires the diagonalization of the Laplacian, an operation whose complexity in the general case scales as $O(N^3)$ for the number of operations and $O(N^2)$ for memory requirements. Additionally, when the number of available realizations of the process is small, it is not possible to obtain a good estimate of the covariance matrix. To overcome these issues, inspired by Bartlett [5] and Welch [4], we propose to use a graph generalization of the Short Time Fourier transform [12] to construct a scalable estimation method.

Bartlett's method can be summarized as follows. The signal is first cut into equally sized segments without overlap. Then, the Fourier transform of each segment is computed. Finally, the PSD is obtained by averaging over segments the squared amplitude of the Fourier coefficients. Welch's method is a generalization that works with segment overlap.

A direct generalization of these method is complicated because it requires to cut the signal in the vertex domain. Additionally, it would not be scalable, since it still requires to compute the Fourier transform of the signal. In this contribution, we use another angle. Both Bartlett and Welch methods can be generalized with the idea that a PSD estimation is obtained by averaging the squared short time Fourier coefficients over the time.

We propose a method based on this last principle. Instead of a translated (rectangular) window, we use a kernel g localized over all the nodes of the graph. This kernel is then uniformly shifted in the spectral domain ($g_m(\lambda_\ell) = g(\lambda_\ell - m\tau)$) to obtain a generalization of the short time Fourier transform. This kind of filterbank has been largely used in classical signal processing and recently introduced for graphs [12]. Our algorithm simply consists in averaging the squared coefficients of this transform over the vertex set.

Because graphs have an irregular spectrum, we additionally need a normalization factor which is given by the norm of the window g_m: $\|g_m\|_2^2 = \sum_\ell g(\lambda_\ell - m\tau)^2$. Note that this norm will vary for the different m. Our final estimator reads :

$$\widehat{\gamma_x}(m\tau) = \frac{\|g_m(\mathcal{L})x\|_2^2}{\|g_m\|_2^2}, \tag{13}$$

where x is a single realization of the stationary process $\mathbf{x}$. Studying the bias of (13) reveals its interest :

$$\frac{\mathbb{E}\left(\|g_m(\mathcal{L})\mathbf{x}\|_2^2\right)}{\|g_m\|_2^2} = \frac{\sum_{\ell=0}^{N-1}\left(g(\lambda_\ell - m\tau)\right)^2 \gamma_{\mathbf{x}}(\lambda_\ell)}{\sum_{\ell=0}^{N-1}\left(g(\lambda_\ell - m\tau)\right)^2}, \tag{14}$$

where $\mathbf{x}$ is the graph stationary process. For a filter g well concentrated at the origin, (14) gives a smoothed estimate of $\gamma_{\mathbf{x}}(m\tau)$. This smoothing corresponds to the windowing operation in the vertex domain: the larger the kernel g (and the smoothing), the smaller the window in the vertex domain. It is very interesting to note we recover the traditional trade-off between bias and variance in non-parametric spectral estimation. Indeed, if g is very sharply localized on the spectrum, ultimately a Dirac delta, the estimator (13) is unbiased. Let us now study the variance. Intuitively, if the signal is correlated only over small regions of the vertex set, we could isolate them with localized windows of a small size and averaging those uncorrelated estimates together would reduce the variance. These small size windows on the vertex set correspond to wide filters g_m and therefore large bias. However, if those correlated regions are large, and this happens when the PSD is localized in low-frequencies, we cannot hope to benefit from vertex-domain averaging since the graph is finite. Indeed the corresponding windows g_m on the vertex set are so large that a single window spans the whole graph and there is no averaging effect: the variance increases precisely when we try to suppress the bias.

Our complete estimation procedure is as follows. First, we design a filterbank by choosing a mother function g (for example a Gaussian $g(x) = e^{-x^2/\sigma^2}$). A frame is then created by shifting uniformly M times g in the spectral domain: $g_m(x) = g(x - m\tau) = e^{-(x-m\tau)^2/\sigma^2}$. Second, we compute the estimator $\widehat{\gamma_x}(m\tau)$ from the stationary signal x. Note that if we have access to K realizations of the stationary process, we can of course average them to further reduce the variance using $\mathbb{E}\left(\|g_m(\mathcal{L})\mathbf{x}\|_2^2\right) \approx 1/K \sum_k \|g_m(\mathcal{L})x_k\|_2^2$. Third we use the following trick to quickly approximate $\|g_m\|_2^2$. Using K_2 randomly-generated Gaussian white signals, we estimate

$$\mathbb{E}\left(\|g_m(\mathcal{L})\mathbf{w}\|_2^2\right) = \|g_m\|_2^2.$$

Finally, the last step consists in computing the ratio between the two quantities and interpolating the discrete points $(k\tau, \big(g * \gamma\big)(m\tau))$.

Experimental assessment of the method Figure 4 shows the results of our PSD-estimation algorithm on a 10-nearest neighbors graph of $20'000$ nodes (sensor type) and only 1 signal. We compare the estimation using frames of 10, 20, 30, 100 Gaussian filters. The parameters σ and τ are adapted to the number of filters. For this experiment K_2 is set to 10 and the Chebysheff polynomial order is 30 (Except for $M = 100$ where we took 100). The estimated curves are smoothed versions of the PSD. Note that with 100 filters, the windows are very concentrated in the spectral domain and broad in the vertex domain. Thus, we loose the averaging effect of the algorithm resulting in a PSD looking like the Fourier transform of the original signal.

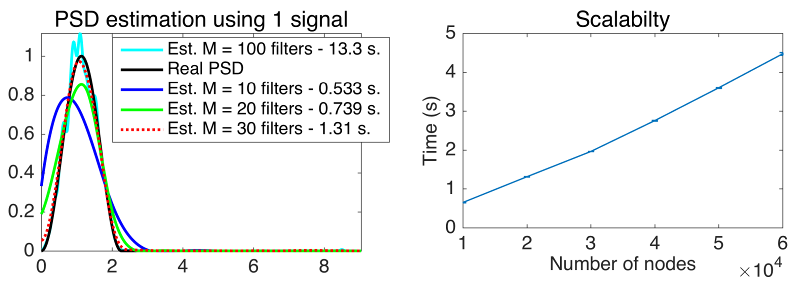


Figure 4: Left: PSD estimation on a graph of $20'000$ nodes with 1 measurements. Our algorithm is able to successively estimate the PSD of a signal. Right: Computation time versus size of the graph. We use $M = 30$ filters. The algorithm scales linearly with the number of edges.

Complexity analysis The approximation scales with the number of edges of the graph: $\mathcal{O}(|\mathcal{E}|)$, (which is proportional to N in many graphs). Precisely, our PSD estimation method necessitates $(K + K_2)M$ filtering operations (with M the number of shifts of g). A filtering operation costs approximatively $O_c|E|$, with O_c the order of the Chebysheff polynomial. The final computational cost of the method is thus $\mathcal{O}\left(O_c(K + K_2)M|\mathcal{E}|\right)$.

Error analysis The difference between the approximation and the exact PSD is caused by three different factors.

1. The inherent bias of the estimator, which is now directly controlled by the parameter σ.

2. We estimate the expected value using K signals (often $K = 1$). For large graphs $N >> K$ and a few filters $M << N$, this error is usually low because the variance of $\|g_m(\mathcal{L})\mathbf{x}\|_2^2$ is inversely proportional to bias. The estimation error improves as $\frac{1}{K}$.

3. We use a fast-filtering method based on a polynomial approximation of the filter. For a rough approximation, $\sigma >> \frac{\lambda_{\max}}{N}$, this error is usually negligible. However, in the other cases, this error may become large.

6 Illustrative experiment on USPS

Stationarity may not be an obvious hypothesis for a general dataset, since our intuition does not allow us to easily capture the kind of shift invariance that is really implied. In this section we give additional insights on stationarity from a more experimental point of view. To do so, we use the well-known USPS dataset.

Images can be considered as signals on the 2-dimensional euclidean plane and, naturally, when the signal is sampled, a grid graph is used as a discretization of this manifold. The corresponding eigenbasis is the 2 dimensional DCT[3]. Following Julesz's initial assumption [27] many papers have exploited the fact that natural texture images are stationary 2-dimensional processes, i.e stationary signals on the grid graph [28]. In [29], the authors go one step further and ask the following question: suppose that pixels of images have been permuted, can we recover their relative two-dimensional location? Amazingly, they answer positively adding that only a few thousand images are enough to approximatively recover the relative location of the pixels. The grid graph seems naturally encoded within images.

The observation of [29] motivate the following experiment involving stationarity on graphs. Let us select the USPS data set which contains 9298 digit images of 16×16 pixels. We create 5 classes of data: (a) the circularly shifted digits[4], (b) the original digits and (c), (d) and (e) the classes of digit 3, 7 and 9. For those 5 cases, we compute the covariance matrix Σ and its graph PSD,

$$\Gamma = U^* \Sigma U, \tag{15}$$

for 2 different graphs: (a) the grid and (b) the 20 nearest neighbors graph. In this latter case, each node is a pixel and is associated to a feature vector containing the corresponding pixel value of all images. We use the squared euclidean distance between feature vectors to define edge weights. We then compute the stationarity level of each class of data with both graph using the following measure:

$$s_r(\Gamma) = \left(\frac{\sum_\ell \Gamma_{\ell,\ell}^2}{\sum_{\ell_1} \sum_{\ell_2} \Gamma_{\ell_1,\ell_2}^2} \right)^{\frac{1}{2}} = \frac{\|\mathrm{diag}(\Gamma)\|_2}{\|\Gamma\|_F} \tag{16}$$

The closer $s_r(\Gamma)$ is to 1, the more diagonal the matrix Γ is and the more stationary the process. Table 2 shows the obtained stationarity measures. The less universal the data, the less stationary it is on the grid. Clearly, specificity inside the data requires a finer structure than a grid. This is confirmed by the behavior of the nearest neighbors graph. When only one digit class is selected the nearest neighbors graph still yields very stationary signals.

Let us focus on the digit 3. For this experiment, we build a 20 nearest neighbors graph with only 50 samples. Figure 5 shows the eigenvectors of the

[3]This is a natural extension of [26]

[4]We performed all possible shifts. Because of this, the covariance matrix becomes Toeplitz

Data \Graph	2-dimensional grid	20 nearest neighbors graph
Shifted all digits	0.86	1
All digits	0.73	0.84
Digit 3	0.53	0.97
Digit 7	0.44	0.97
Digit 9	0.49	0.97

Table 2: $s_r(\Gamma) = \frac{\|\mathrm{diag}(\Gamma)\|_2}{\|\Gamma\|_F}$ measures for different graphs and different datasets. The nearest neighbors graph adapts to the data. The individual digits are stationary with the nearest neighbor graph.

Laplacian and of the covariance matrix. Because of stationarity, they are very similar. Moreover, they have 3-like shape. Using the graph and the PSD, it is also possible generate samples by filtering Gaussian random noise with the following PSD based kernel: $g(\lambda_\ell) = \sqrt{\Gamma_{\ell,\ell}}$. The resulting digits have 3-like shape confirming the that the class is stationary on the nearest neighbors graph.

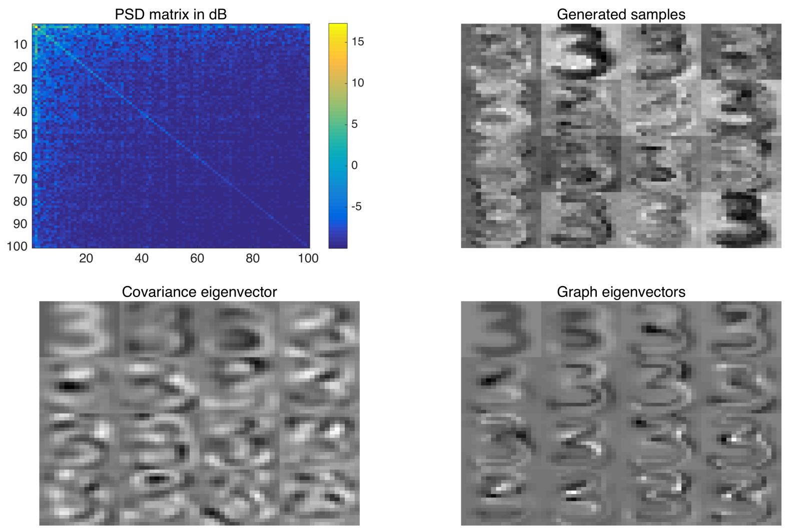


Figure 5: Studying the number 3 of USPS. Top left: PSD of the data (Note the diagonal shape of the matrix). Top right: generated samples by filtering Gaussian random noise on the graph. Bottom left: Covariance eigenvectors associated with the 16 highest eigenvalues. Bottom right: Laplacian eigenvectors associated to the 16 smallest non-zero eigenvalues. Because of stationarity, Laplacian eigenvectors are similar to the covariance eigenvectors.

To intuitively motivate the effectiveness of nearest neighbors at producing stationary signals, let us define the centering operator $J = I - \mathbf{1}\mathbf{1}^\top/N$. A straightforward calculation shows that the matrix of average squared distances between the centered features is directly proportional to the covariance matrix :

$$\frac{1}{d}\Sigma_{\mathbf{x}} = -\frac{1}{2} J D_{\mathbf{x}} J,$$

where $D_{ij} = \sum_k \left(x_k(i) - x_k(j)\right)^2$ and $\mathbb{E}D = D_{\mathbf{x}}$. The nearest-neighbors graph can be seen as an approximation of the original distance matrix, which pleads for using it as a good proxy destined at leveraging the spectral content of the covariance. Put differently, when using realizations of the process as features and computing the k-NN graph we are connecting strongly correlated variables via strong edge weights.

7 Experiments

The experiments were performed with the GSPBox [30] and the UNLocBoX [31] two open-source softwares. The code to reproduce all figures of the paper can be downloaded at: `https://lts2srv1.epfl.ch/rrp/stationarity`. As the stationary signals are random, the reader may obtain slightly different results. However, conclusions shall remain identical. The models used in our comparisons are detailed in the Appendix A for completeness.

7.1 Synthetic dataset

In order to obtain a first insight into applications using stationarity, we begin with some classical problems solved on a synthetic dataset. Compared to real data, this framework allows us to be sure that the signal is stationary on the graph.

Graph Wiener deconvolution We start with a de-convolution example on a random geometric graph. This can model an array of sensors distributed in space or simply a mesh. The signal is chosen with a low frequency band-limited PSD. To produce the measurements, the signal is convolved with the heat kernel $e^{-\tau x}$. Additionally, we add some Gaussian noise. The heat kernel is chosen because it simulates a diffusion process. Using de-convolution we aim at recovering the original signal before diffusion. For this experiment, we put ourselves in an ideal case and suppose that both the PSD of the input signal and the noise level are known.

Figure 6 presents the results. We observe that Wiener filtering is able to de-convolve the measurements. The second plot shows the reconstruction errors for three different methods: Tikonov presented in problem (18), TV in (20) and Wiener filtering in (9). Wiener filtering performs clearly much better than the other methods because it has a much better prior assumption.

Graph Wiener in-painting In our second example, we use Wiener optimization to solve an in-painting problem. This time, we suppose that the PSD of the input signal is unknown and we estimate it using 50 signals. Figure 7 presents quantitative results for the in-painting. Again, we compare three different optimization methods: Tikonov (17), TV (20) and Wiener (12). Wiener

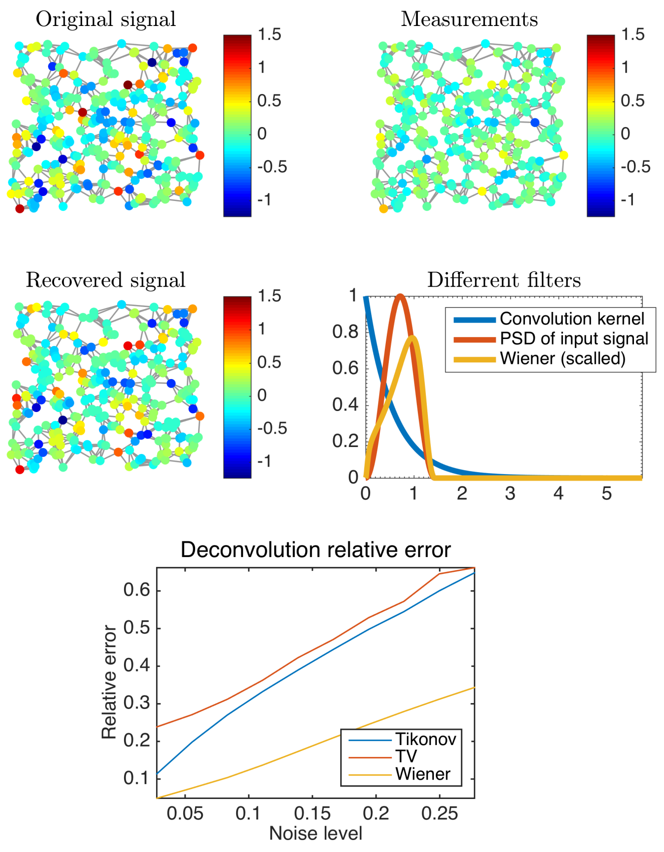


Figure 6: Graph de-convolution on a geometric random graph. The convolution kernel is $e^{-\frac{10x}{\lambda_{\max}}}$. Top: Signal and filters for a noise level of 0.16. Bottom: evolution of the error with respect of the noise.

optimization performs much better than traditional methods because the generated data fits the stationarity model. Moreover, we also observe that the PSD approximation does not affect the result.

7.2 Real dataset

We apply our methods to a weather measurements dataset, more precisely to the temperature and the humidity. Since these two quantities change smoothly across space, it suggests that they are more or less stationary on a nearest neighbour graph.

The French national meteorological service has published in open access a dataset[5] with hourly weather observations collected during the Month of January 2014 in the region of Brest (France). From these data, we wish to ascertain that our method still performs better than the two other models (TV and Tikonov) on real measurements. The graph is built from the coordinates of

[5] Access to the raw data is possible directly through our code.

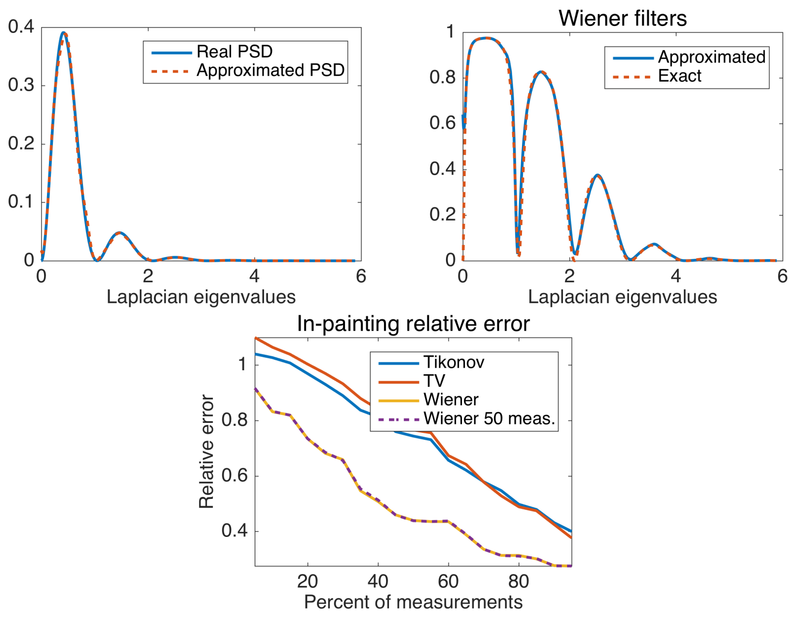


Figure 7: Wiener in-painting on a geometric graph of 400 nodes. Top: true VS approximated PSD and resulting Wiener filters. Bottom: in-painting relative error with respect to number of measurements.

the weather stations by connecting all the neighbours in a given radius with a weight function $w(i,j) = e^{-d^2\tau}$ where τ is adjusted to obtain a average degree around 3 (τ, however, is not a sensitive parameter). For our experiments, we consider every time step as an independent realization of a WSS process. As sole pre-processing, we remove the mean of the temperature. Thanks to the 744 time observation, we can estimate the covariance matrix and check wether the process is stationary on the graph.

In-painting - Temperature The result of the experiment with temperatures is displayed in Figure 8. The covariance matrix shows a strong correlation between the different weather stations. Diagonalizing it with the Fourier basis of the graph assesses that the meteorological instances are more or less stationary within the distance graph by highlighting its diagonal characteristic. Moreover this diagonal gives us access to the PSD of the data. In our experiment, we solve an in-painting/de-noising problem with a mask operator covering 50 per cent of measurements and various amount of noise. We then average the result over 744 experiments (corresponding to the 744 observations) to obtain the curves displayed in Figure 8. We observe that Wiener optimization performs significantly better when the noise level is high and equivalently well to the two other methods for low noise level.

In-painting - Humidity Using the same graph, we have performed another set of experiments on humidity observations. The testing framework is identical as for the temperature and the conclusions are similar.

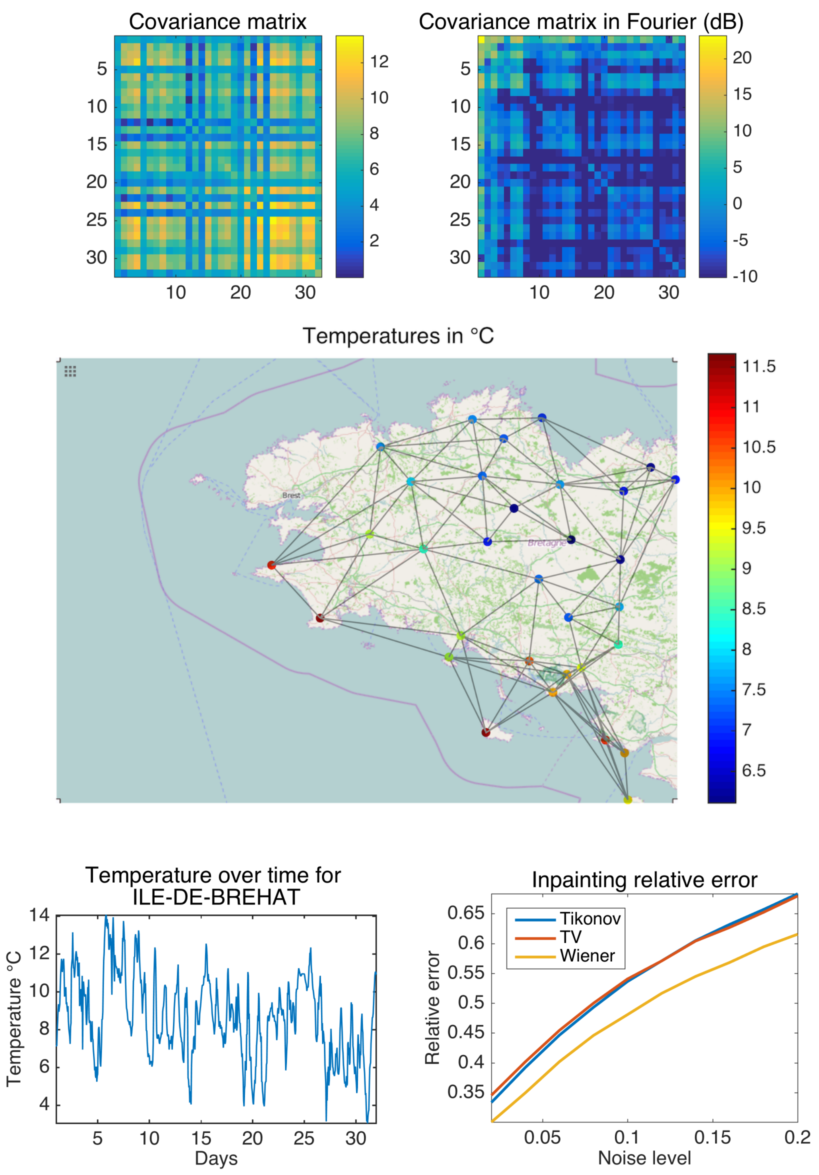


Figure 8: Top: Covariance matrices. Bottom left: An example of the process on the graph (first measure). Bottom center: the temperature of the Island of Brehat. Bottom right: Recovery errors for different noise levels.

7.3 USPS dataset

We perform the same kind of in-painting/de-noising experiment with the USPS dataset. We compute the graph using the first 300 digits and use 4349 of the remaining ones to test our algorithm. We use a mask covering 50 per cent of the pixel and various amount of noise. We then average the result over 4349 experiments (corresponding to the 4349 digits) to obtain the curves

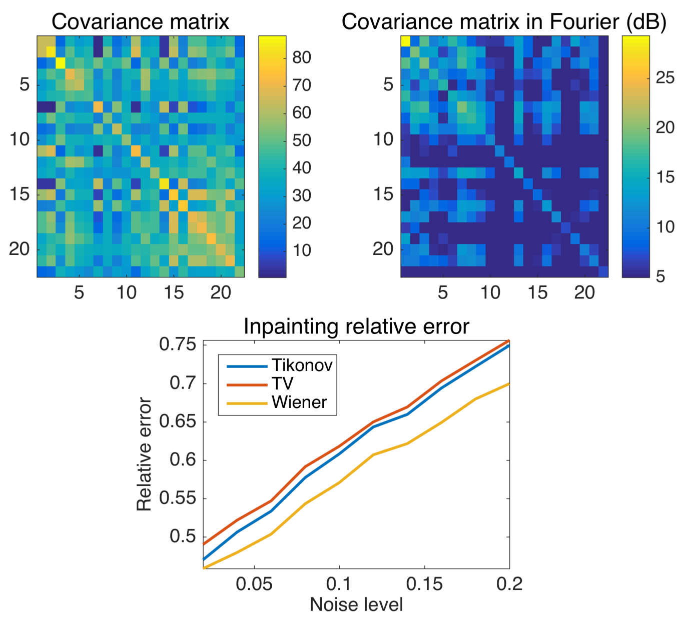


Figure 9: Top: Covariance matrices. Bottom: Recovery errors for different noise levels.

displayed in Figure 10. For this experiment, we also compare to traditional TV de-noising [32] and Tikonow de-noising. The optimization problems used are similar to 17. The results presented in Figure 10 show that graph optimization is outperforming classical techniques meaning that the grid is not the optimal graph for the USPS dataset. Moreover, Wiener once again outperforms the other graph-based models.

8 Conclusion

In this contribution, we have extended the common concept of stationarity to graph signals. Using this statistical model, we proposed a new regularization framework that leverages the stationarity hypothesis by using the Power Spectral Density (PSD) of the signal. Since the PSD can be efficiently estimated, even for large graphs, the proposed Wiener regularization framework offers a compelling way to solve traditional problems such as denoising, regression or semi-supervised learning. We believe that stationarity is a natural hypothesis for many signals on graphs and showed experimentally that it is deeply connected with the popular nearest neighbor graph construction. As future work, it would be very interesting to clarify this connection and explore if stationarity could be used to infer the graph structure from training signals, in the spirit of [33].

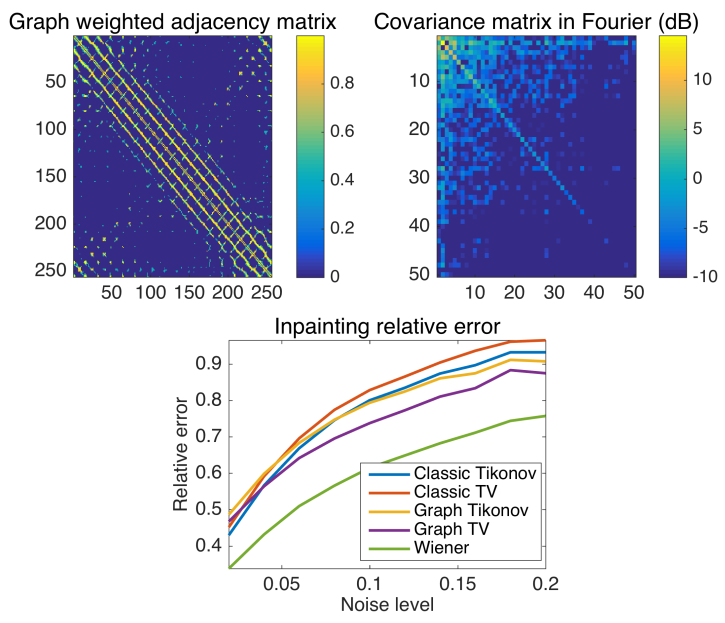


Figure 10: Top left: Weights matrix of the 20 nearest neighbor graph (The diagonal shape indicate the grid base topology of the graph). Top right: PSD matrix for the first 50 graph frequencies. Bottom: Recovery errors for different noise levels. Methods using the nearest neighbors graph performs better.

Acknowledgment

This work has been supported by the Swiss National Science Foundation research project *Towards Signal Processing on Graphs*, grant number: 2000_21/154350/1.

A Convex models

Convex optimization has recently become a standard tool for problems such as de-noising, de-convolution or in-painting. Graph priors have been used in this field for more than a decade [7, 8, 9]. The general assumption is that the signal varies smoothly along the edges, which is equivalent to saying that the signal is low-frequency-based. Using this assumption, one way to express mathematically an in-painting problem is the following:

$$\tilde{x} = \arg\min_x x^T \mathcal{L} x \qquad \text{s.t.} \qquad \|Mx - y\|_2 \leq \epsilon \tag{17}$$

where M is a masking operator and ϵ a constant computed thanks to the noise level. We could also rewrite the objective function as $x^T \mathcal{L} x + \gamma \|Mx - y\|_2^2$, but this implies a greedy search of the regularization parameter γ even when the level of noise is known. In general ϵ is easier to set. For example, when the noise is uniformly Gaussian, we have $\epsilon = n\sqrt{\#y}$, where $\#y$ is the number of elements of y.

Graph de-convolution can also be addressed with the same prior assumption leading to

$$\tilde{x} = \arg\min_x x^T \mathcal{L} x \qquad \text{s.t.} \qquad \|h(\mathcal{L})x - y\|_2 \leq \epsilon \tag{18}$$

where h is the convolution kernel. To be as generic as possible, we combine problems (17) and (18) together leading to a model capable of performing de-convolution, in-painting and de-noising at the same time:

$$\tilde{x} = \arg\min_x x^T \mathcal{L} x \qquad \text{s.t.} \qquad \|Mh(\mathcal{L})x - y\|_2 \leq \epsilon. \tag{19}$$

When the signal is piecewise smooth on the graph, another regularization term can be used instead of $x^T \mathcal{L} x = \|\nabla_G x\|_2^2$, which is the ℓ_2-norm of the gradient on the graph[6]. Using the ℓ_1-norm of the gradient favours a small number of major changes in signal and thus is better for piecewise smooth signals. The resulting model is:

$$\tilde{x} = \arg\min_x \|\nabla_G x\|_1 \qquad \text{s.t.} \qquad \|Mh(\mathcal{L})x - y\|_2 \leq \epsilon \tag{20}$$

In order to solve these problems, we use a subset of convex optimization tools called proximal splitting methods. Since we are not going to summarise them here, we encourage a novice reader to consult [22, 23] and the references therein for an introduction to the field.

B Proof of Theorem 3

The following is a generalization of the classical proof.

Proof. Because, by hypothesis $A = a(\mathcal{L}) = Ua(\Lambda)U^*$, we can rewrite the optimization problem 12 in the graph Fourier domain using Parseval relationship $\|x\|_2 = \|Ux\|_2 = \|\hat{x}\|_2$:

$$\dot{\hat{x}} = \arg\min_{\hat{x}} \|w(\Lambda)\hat{x}\|_2^2 + \|a(\Lambda)\hat{x} - \hat{y}\|_2^2.$$

Since the matrix Λ is diagonal, the solution of this problem satisfies for all graph eigenvalue λ_ℓ

$$w^2(\lambda_\ell)\hat{x}(\lambda_\ell) + a^2(\lambda_\ell)\hat{x}(\lambda_\ell) - a(\lambda_\ell)\hat{y}(\lambda_\ell) = 0 \tag{21}$$

For simplicity, we drop the notation (λ_ℓ). The previous equation is transformed in

$$\dot{\hat{x}} = \frac{a}{w^2 + a^2}\hat{y}$$

As a next step, we use the fact that $\hat{y} = a\hat{x}_o + \hat{w}_n$ to find:

$$\dot{\hat{x}} = \frac{a^2\hat{x}_o + a\hat{w}_n}{w^2 + a^2}.$$

The error performed by the algorithm becomes

$$\hat{e} = \dot{\hat{x}} - \hat{x}_o = \frac{-w^2\hat{x}_o}{w^2 + a^2} + \frac{a\hat{n}}{w^2 + a^2}$$

[6]The gradient on the graph is defined as $\nabla_G x(i,j) = \frac{1}{2}\sqrt{W(i,j)}\,(x(i) - x(j))$

The expectation of the error can thus be computed:

$$\begin{aligned}\mathbb{E}\left(\hat{e}^2\right) &= \frac{w^4\mathbb{E}\left(\hat{x}_o^2\right)}{\left(w^2+a^2\right)^2} + \frac{a^2\mathbb{E}\left(\hat{w}_n^2\right)}{\left(w^2+a^2\right)^2} - \frac{aw^2\mathbb{E}\left(\hat{x}_o\hat{w}_n\right)}{\left(w^2+a^2\right)^2} \\ &= \frac{w^4s^2+a^2n}{\left(w^2+a^2\right)^2}\end{aligned}$$

with s^2 the PSD of x_o and n the PSD of the noise w_n. Note that $\mathbb{E}\left(\hat{x}_o\hat{w}_n\right) = 0$ because x and w_n are uncorrelated. Let us now substitute w^2 by z and minimize the expected error (for each λ_ℓ) with respect of z

$$\begin{aligned}\frac{\partial}{\partial z}\mathbb{E}\left(\hat{e}^2\right) &= \frac{\partial}{\partial z}\frac{zs^2+a^2n}{\left(z+a^2\right)^2} \\ &= \frac{2zs^2\left(z+a^2\right)-2\left(z^2s^2+a^2n\right)}{\left(z+a^2\right)^3} = 0\end{aligned}$$

From the numerator, we get:

$$2zs^2a^2 - 2a^2n = 0$$

The three possible solution for z are $z_1 = \frac{n}{s^2}$, $z_2 = \infty$ and $z_3 = -\infty$. z_3 is not possible because z is required to be positive. z_2 leads to $\dot{x} = 0$ which is optimal only if $s^2 = 0$. This makes the optimal value of $z(\lambda_\ell) = \frac{n(\lambda_\ell)}{s^2(\lambda_\ell)}$, resulting in

$$w(\lambda_\ell) = \sqrt{\frac{n(\lambda_\ell)}{s^2(\lambda_\ell)}}.$$

This close the first part of the proof.

To show that the solution to (12) is a wiener filtering operation, we replace $w^2(\lambda_\ell)$ by $\frac{n(\lambda_\ell)}{s^2(\lambda_\ell)}$ in (21). We find

$$\hat{x}(\lambda_\ell) = \frac{s^2(\lambda_\ell)a(\lambda_\ell)\hat{y}(\lambda_\ell)}{a^2(\lambda_\ell)s^2(\lambda_\ell)+n(\lambda_\ell)},$$

which is the wiener filter associated to the convolution $a(\mathcal{L}) = A$. □

References

[1] C. Williams, "Prediction with Gaussian processes: From linear regression to linear prediction and beyond," *Learning in graphical models*, 1998.

[2] N. Wiener, "Generalized harmonic analysis," *Acta mathematica*, vol. 55, no. 1, pp. 117–258, 1930.

[3] ——, *Extrapolation, interpolation, and smoothing of stationary time series.* MIT press Cambridge, MA, 1949, vol. 2.

[4] P. Welch, "The use of fast fourier transform for the estimation of power spectra: a method based on time averaging over short, modified periodograms," *IEEE Transactions on audio and electroacoustics*, pp. 70–73, 1967.

[5] M. S. Bartlett, "Periodogram analysis and continuous spectra," *Biometrika*, pp. 1–16, 1950.

[6] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *Signal Processing Magazine, IEEE*, vol. 30, no. 3, pp. 83–98, 2013.

[7] A. J. Smola and R. Kondor, "Kernels and regularization on graphs," in *Learning theory and kernel machines.* Springer, 2003, pp. 144–158.

[8] D. Zhou and B. Schölkopf, "A regularization framework for learning from graph data," 2004.

[9] G. Peyré, S. Bougleux, and L. Cohen, "Non-local regularization of inverse problems," in *Computer Vision–ECCV 2008.* Springer, 2008, pp. 57–68.

[10] A. J. Smola and R. Kondor, "Kernels and Regularization on Graphs," in *Proc. Ann. Conf. Comp. Learn. Theory*, B. Schölkopf and M. Warmuth, Eds. Springer, 2003, pp. 144–158.

[11] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Applied and Computational Harmonic Analysis*, vol. 30, no. 2, pp. 129–150, 2011.

[12] D. I. Shuman, B. Ricaud, and P. Vandergheynst, "Vertex-frequency analysis on graphs," *arXiv preprint arXiv:1307.5708*, 2013.

[13] A. Sandryhaila and J. M. Moura, "Discrete signal processing on graphs," *IEEE transactions on signal processing*, vol. 61, pp. 1644–1656, 2013.

[14] B. Girault, P. Gonçalves, and É. Fleury, "Translation and stationarity for graph signals," Ph.D. dissertation, École Normale Supérieure de Lyon; Inria Rhône-Alpes, 2015.

[15] B. Girault, P. Goncalves, E. Fleury, and A. S. Mor, "Semi-supervised learning for graph to signal mapping: A graph signal wiener filter interpretation," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on.* IEEE, 2014, pp. 1115–1119.

[16] A. Gadde and A. Ortega, "A probabilistic interpretation of sampling theory of graph signals," *arXiv preprint arXiv:1503.06629*, 2015.

[17] C. Zhang, D. Florêncio, and P. A. Chou, "Graph signal processing–a probabilistic framework," 2015.

[18] F. R. Chung, *Spectral graph theory.* AMS Bookstore, 1997, vol. 92.

[19] F. Chung, "Laplacians and the cheeger inequality for directed graphs," *Annals of Combinatorics*, vol. 9, no. 1, pp. 1–19, 2005.

[20] A. Susnjara, N. Perraudin, D. Kressner, and P. Vandergheynst, "Accelerated filtering on graphs using lanczos method," *arXiv preprint arXiv:1509.04537*, 2015.

[21] A. Papoulis and S. U. Pillai, *Probability, random variables, and stochastic processes.* Tata McGraw-Hill Education, 2002.

[22] P. L. Combettes and J.-C. Pesquet, "Proximal splitting methods in signal processing," in *Fixed-point algorithms for inverse problems in science and engineering.* Springer, 2011, pp. 185–212.

[23] N. Komodakis and J.-C. Pesquet, "Playing with duality: An overview of recent primal-dual approaches for solving large-scale optimization problems," *arXiv preprint arXiv:1406.5429*, 2014.

[24] P. L. Combettes and V. R. Wajs, "Signal recovery by proximal forward-backward splitting," *Multiscale Modeling & Simulation*, vol. 4, no. 4, pp. 1168–1200, 2005.

[25] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM journal on imaging sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[26] G. Strang, "The discrete cosine transform," *SIAM review*, vol. 41, no. 1, pp. 135–147, 1999.

[27] B. Julesz, "Visual Pattern Discrimination," *Information Theory, IRE Transactions on*, vol. 8, no. 2, pp. 84–92, 1962.

[28] R. C. Dubes and A. K. Jain, "Random field models in image analysis," *Journal of applied statistics*, 1989.

[29] N. L. Roux, Y. Bengio, P. Lamblin, M. Joliveau, and B. Kégl, "Learning the 2-d topology of images," in *Advances in Neural Information Processing Systems*, 2008, pp. 841–848.

[30] N. Perraudin, J. Paratte, D. Shuman, V. Kalofolias, P. Vandergheynst, and D. K. Hammond, "GSPBOX: A toolbox for signal processing on graphs," *ArXiv e-prints*, Aug. 2014.

[31] N. Perraudin, D. Shuman, G. Puy, and P. Vandergheynst, "UNLocBoX A matlab convex optimization toolbox using proximal splitting methods," *ArXiv e-prints*, Feb. 2014.

[32] A. Chambolle, "An algorithm for total variation minimization and applications," *Journal of Mathematical imaging and vision*, vol. 20, no. 1-2, pp. 89–97, 2004.

[33] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst, "Learning Laplacian Matrix in Smooth Graph Signal Representations," *arXiv.org*, Jun. 2014.