ARE THERE APPROXIMATE FAST FOURIER TRANSFORMS ON GRAPHS?

Luc Le Magoarou, Rémi Gribonval
Inria
Centre Inria Rennes - Bretagne Atlantique

ABSTRACT

Signal processing on graphs is a recent research domain that seeks to extend classical signal processing tools such as the Fourier transform to irregular domains given by a graph. In such a graph setting, a way to rapidly apply the Fourier transform, i.e. a Fast Fourier Transform (FFT), is lacking. In this paper, we propose to leverage the recently introduced Flexible Approximate MUlti-layer Sparse Transforms (FAμST) in order to compute approximate FFTs on graphs. The approach is first described, then validated on several types of classical graphs and finally used for fast filtering, showing good potential.

Index Terms— Graphs signal processing, Fast Fourier Transform, matrix factorization.

1. INTRODUCTION

Graphs are ubiquitous in modern data processing. They are indeed a very convenient mathematical tool to represent complex relationships that arise naturally when dealing with networked data acquisition settings. Recently, methodological tools have been developed that generalize classical signal processing techniques to graph signals, i.e. signals that live on the vertices of a graph instead of a regular grid, as classically assumed in signal processing. Extensive surveys on this topic can be found in [1] and [2]. In particular, the Fourier transform of graph signals can be defined, but no fast algorithm has been discovered yet to apply it for general graphs.

In the following, we consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$, where $\mathcal{V}$ and $\mathcal{E}$ represent the vertex and edge sets of the graph, and $\mathbf{W} \in \mathbb{R}^{n\times n}$ represents the matrix of edge weights (w_{ij} being the weight of an edge connecting vertices i and j), with $n = |\mathcal{V}|$ denoting the number of vertices. We assume the graph is connected. We denote by $\mathbf{L} \in \mathbb{R}^{n\times n}$ the combinatorial graph Laplacian matrix, and by $\mathbf{U} \in \mathbb{R}^{n\times n}$ its eigenvector matrix, with associated eigenvalues in the diagonal matrix $\mathbf{\Sigma} \in \mathbb{R}^{n\times n}$.

A Fourier transform on a graph can be defined as the change of basis from the trivial node basis to the basis defined by the eigenvectors of $\mathbf{L}$, namely the columns of $\mathbf{U}$ (see [1] for a more precise definition). Considering a signal $\mathbf{x} \in \mathbb{R}^n$ on the graph, and its Fourier transform $\mathbf{y} \in \mathbb{R}^n$ we have:

$$\begin{aligned} \mathbf{y} = &\quad \mathbf{U}^T\mathbf{x} \\ \mathbf{x} = &\quad \mathbf{U}\mathbf{y}. \end{aligned}$$

The matrix $\mathbf{U}$ being dense in general, the change of basis costs $\mathcal{O}(n^2)$ arithmetic operations in both ways, if done using this basic matrix multiplication. However in the case of a regular domain (which can be viewed as a ring graph), it is possible to do the change of basis in $\mathcal{O}(n \log n)$ arithmetic operations, using the Fast Fourier Transform (FFT) [3], which is a linear algorithm (i.e. only made of scalar multiplications and additions) exploiting the factorizability of $\mathbf{U}$ into sparse factors[1],

$$\mathbf{U} = \prod_{j=1}^{J} \mathbf{S}_j. \tag{1}$$

This factorizability is a necessary and sufficient condition for a fast linear algorithm to exist [4]. In the case of the classical Fourier transform, $\mathbf{U}$ can be factorized into $J = \log_2(n)$ factors, each having $2n$ nonzero entries. Unfortunately, it is as of today unclear if such a factorization can be generalized to more complicated graphs structures. Relaxing the exact factorization constraint, and considering an approximate form of (1), one can wonder: *Are there approximate Fast Fourier Transforms on graphs?*

We recently proposed the Flexible Approximate MUlti-layer Sparse Transforms (FAμST) [5], an empirical approach to approximate matrices by such multi-layer sparse products. It allows to get computationally efficient approximations for matrices of interest. The approach is based on non-convex optimization techniques and amounts to a hierarchical factorization of the matrix to approximate. It has been applied to dictionary learning [6] and inverse problems [7], showing a good trade-off between accuracy and computational complexity.

Such an approach can be applied to the Fourier matrix $\mathbf{U}$ of a graph, in order to try and give an answer to the question asked in this paper. This could be of direct practical interest in applications where the same Fourier matrix has to be applied to a great number of signals. This situation occurs for example when time-varying signals defined on a static graph have to be manipulated in the frequency domain (see [8, section VI.A] for an example application of this type).

Objective and contributions. Here, our goal is to show that $\mathbf{U}$ can be approximated by a FAμST for a wide variety of graphs, thus defining an approximate FFT for the considered graph. This is done by modifying the previously proposed multi-layer sparse factorization approach [5] to handle the fact that the FAμST has to diagonalize approximately the Laplacian matrix $\mathbf{L}$. We propose in this paper an approach that allows to get FAμSTs with computational complexities $\mathcal{O}(n^\alpha)$, $1 < \alpha < 2$, approximating well the true Fourier transform of many classical families of graphs as shown in sections 3.1 and 3.2. Moreover, it shows potential for concrete applications such as fast approximate filtering, as attested in section 3.3.

2. OPTIMIZATION FRAMEWORK

2.1. Algorithm

Our goal is to compute a FAμST $\hat{\mathbf{U}} = \prod_{j=1}^{J} \mathbf{S}_j$ that approximates well the graph Fourier matrix $\mathbf{U}$. One possibility to do so is to directly factorize $\mathbf{U}$ into sparse factors, as is done in [7, Algorithm 1]

This work was supported in part by the European Research Council, PLEASE project (ERC-StG- 2011-277906).

[1]The product being taken from right to left: $\prod_{i=1}^{N} \mathbf{A}_i = \mathbf{A}_N \cdots \mathbf{A}_1$

in the case of inverse problems. We refer to this approach as *FAμST factorization of the Fourier matrix* $\mathbf{U}$ in the remaining of the paper. However, such a direct factorization does not take into account the fact that the approximate Fourier matrix $\hat{\mathbf{U}}$ has to diagonalize approximately the Laplacian $\mathbf{L}$. We propose next a *FAμST diagonalization of the Laplacian matrix* $\mathbf{L}$. The optimization problem at hand is the following:

$$\begin{array}{ll} \underset{\mathbf{S}_1,\ldots,\mathbf{S}_J,\mathbf{D}}{\text{minimize}} & \frac{1}{4}\left\|\mathbf{L}-\mathbf{S}_J\ldots\mathbf{S}_1\mathbf{D}\mathbf{S}_1^T\ldots\mathbf{S}_J^T\right\|_F^2 \\ \text{subject to} & \mathbf{S}_j\in\mathcal{S}_j,\ \forall j\in\{1,\ldots,J\} \\ & \mathbf{D}\in\mathcal{D}, \end{array} \tag{2}$$

where the $\mathcal{S}_j$s are sets of sparse matrices and $\mathcal{D}$ is a set of diagonal matrices. This problem actually amounts to try and diagonalize the Laplacian matrix $\mathbf{L}$ by a FAμST $\prod_{j=1}^{J}\mathbf{S}_j$. It is a non-convex and non-smooth problem, that can be tackled using a Proximal Alternating Linearized Minimization (PALM) [9] algorithm, with convergence guarantees to a stationary point. Such an algorithm updates alternatively each factor by a projected gradient descent step. However, the high number of local minima renders the problem very dependent to initialization.

As first introduced in our previous work [6] for a related problem, instead of directly tackling the problem (2), we propose to resort to a strategy in which the true Fourier matrix $\mathbf{U}$ is factorized hierarchically. The strategy is to start from the true Fourier matrix $\mathbf{U}\triangleq\mathbf{T}_0$, and factorize it in two factors: $\mathbf{U}\approx\mathbf{T}_1\mathbf{S}_1$, where $\mathbf{T}_1$ is called the residual and $\mathbf{S}_1$ is sparse. The residual is then iteratively factorized: $\mathbf{T}_{i-1}\approx\mathbf{T}_i\mathbf{S}_i$, where $\|\mathbf{T}_i\|_0+\|\mathbf{S}_i\|_0<\|\mathbf{T}_{i-1}\|_0$ in order to make complexity savings at each step ($\|\mathbf{A}\|_0$ being a shorthand for $\|\text{vec}(\mathbf{A})\|_0$). A global optimization step is inserted between each 2-factorization in order to stay close to diagonalizing the Laplacian matrix $\mathbf{L}$. This global optimization handles the following subproblem (for increasing values of i):

$$\begin{array}{ll} \underset{\mathbf{S}_1,\ldots,\mathbf{S}_i,\mathbf{T}_i,\mathbf{D}}{\text{minimize}} & \frac{1}{4}\left\|\mathbf{L}-\mathbf{T}_i\mathbf{S}_i\ldots\mathbf{S}_1\mathbf{D}\mathbf{S}_1^T\ldots\mathbf{S}_i^T\mathbf{T}_i^T\right\|_F^2 \\ \text{subject to} & \mathbf{S}_j\in\mathcal{S}_j,\ \forall j\in\{1,\ldots,i\} \\ & \mathbf{T}_i\in\mathcal{T}_i,\quad \mathbf{D}\in\mathcal{D}, \end{array} \tag{3}$$

with $\mathcal{T}_i$ a set of sparse matrices (sparser as i grows). The structure of the algorithm is given in Algorithm 1. The main steps of this algorithm (lines 3 and 4) use PALM, which amounts to alternate updates of the factors by projected gradient steps. Expressions of the projection operators associated to sets of sparse matrices are given in [5], as well as expressions of the gradients implicit in line 3. Expressions of the gradient implicit in line 4 are not given here for brevity reasons, but easily calculated with basic algebra.For a broader exposition of PALM and its application to FAμST, see [5].

Algorithm 1 FAμST diagonalization of the Laplacian matrix

```
Input: Fourier matrix $\mathbf{U}$; Laplacian $\mathbf{L}$; eigenvalues $\mathbf{\Sigma}$; desired
    number of factors $J$; constraint sets $\mathcal{S}_i$ and $\tilde{\mathcal{S}}_i$, $i\in\{1\ldots J-1\}$.
1: $\mathbf{T}_0\leftarrow\mathbf{U}, \mathbf{D}\leftarrow\mathbf{\Sigma}$
2: for $i=1$ to $J-1$ do
3:     Factorization of the residual in 2 factors using PALM [9]:
       $\mathbf{T}_{i-1}\approx\mathbf{T}_i\mathbf{S}_i$
4:     Global optimization with PALM to handle (3):
       $\mathbf{L}\approx\mathbf{T}_i\mathbf{S}_i\ldots\mathbf{S}_1\mathbf{D}\mathbf{S}_1^T\ldots\mathbf{S}_i^T\mathbf{T}_i^T$
5: end for
6: $\mathbf{S}_J\leftarrow\mathbf{T}_{J-1}$
Output: The estimated factorization: $\{\mathbf{S}_j\}_{j=1}^J$,$\mathbf{D}$.
```

2.2. Factorization setting

The usual Fourier transform on a 1D regular grid (corresponding to a ring graph) can be applied in $\mathcal{O}(n\log n)$ arithmetic operations although the Fourier matrix contains n^2 non-zero entries. This is because the Fourier matrix $\mathbf{U}$ can be factorized into $\log_2 n$ factors, each having $2n$ non-zero entries. In other words, in the 1D regular grid case we have $\mathbf{U}=\prod_{j=1}^{\log_2 n}\mathbf{S}_j$, with $\|\mathbf{S}_j\|_0=2n$. This factorization corresponds to the usual FFT [3]. In the context of a hierarchical factorization of the Fourier matrix with exponentially decreasing sparsity of the residual (as explained in [5]), we can hope to retrieve this factorization by setting the sparsity of the right factor to $2n$ and the sparsity of the residual to $\frac{n^2}{2^i}$ at the ith 2-factorization. This amounts to dividing the residual sparsity by two at each step, i.,e, a rate of decrease of 2. Such a hierarchical strategy would lead to the sparsity configuration depicted on Figure 1.

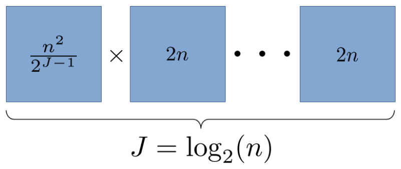


Fig. 1. Ideal factorization configuration in the 1D regular grid case. The sparsity of each factor is shown.

Here, our objective is to generalize the multi-layer sparse factorization of the Fourier matrix to domains given by an underlying graph, in order to find approximate FFTs for graph signals. Since in that case we do not know if a factorization of the Fourier matrix is possible with factors as sparse as in the 1D regular grid case, we propose to relax the sparsity assumptions in the three following ways:

- The number of factors J is reduced by a constant C_1 ($J=\log_2 n-C_1$). This leads to a higher accuracy in the factorization (more factors leading empirically to higher error [5]), and allows to reduce the factorization time which is proportional to the number of factors considering the hierarchical strategy of Algorithm 1.
- The rate of decrease of the residual sparsity is reduced to C_2 with $1<C_2\leq 2$. This amounts to dividing the sparsity of the residual by a number smaller than two at each step of Algorithm 1, leading to a better accuracy of factorization.
- The sparsity of each factor is further multiplied by a constant C_3 with $C_3>1$. Allowing more non-zero entries in the FAμST this way logically leads to higher accuracy.

The resulting relaxed factorization configuration is summarized in Figure 2. Such a factorization contains $C_3.(2n(\log_2 n-C_1)+n^2.C_2^{1+C_1-\log_2 n})$ non-zero entries, which is $\mathcal{O}(n^{2-\log_2 C_2})$ for $1<C_2<2$ and $\mathcal{O}(n\log_2 n)$ for $C_2=2$.

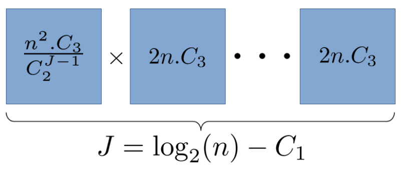


Fig. 2. Relaxed factorization configuration for graph settings.

3. EXPERIMENTS

In this section we assess experimentally the performance of the proposed method. We first try and give an answer to the main question of this paper, and show that we can get FAμSTs with complexity $\mathcal{O}(n^{\alpha})$ and good accuracy for various graphs. We then compare the proposed *FAμST diagonalization of the Laplacian matrix* $\mathbf{L}$ to the *FAμST factorization of the Fourier matrix* $\mathbf{U}$ on random sensor graphs. We finally use the approach in a fast filtering scenario on the Minnesota road graph, showing its applicability.

3.1. Approximate FFT for classical graphs

We perform here an experiment in order to show that it is possible to define approximate FFTs on a variety of standard families of graphs. The factorizations are done with different graphs of different sizes, in order to show that the $\mathcal{O}(n^{\alpha})$ settings that were introduced in section 2.2 are indeed appropriate. We consider five classical random graphs, generated with help of the Graph Signal Processing (GSP) toolbox [10], and taken with their default parameterization (a detailed description can be found in [10]). Are considered:

- **Erdős-Rényi:** a random graph with connection probability $p = 0.1$.
- **Community:** a random community graph made of $\sqrt{n}/2$ communities of random sizes.
- **Random sensor:** a random graph where nodes represent sensors that are placed randomly on the plane.
- **Swiss roll:** a graph where nodes are placed randomly on the Swiss roll manifold.
- **Random ring:** a ring graph with random distances between nodes.

Examples of considered graphs are shown in Figure 3.

The experiment amounts to applying Algorithm 1 on these graphs with a fixed setting: $C_1 = 3$, $C_2 = 1.67$ ($\alpha = 1.26$) and $C_3 = 1.4$, with a varying number of nodes $n \in \{64, 128, 256, 512\}$. The results in terms of relative error with respect to the factorized Laplacian are shown on Figure 4. It can be seen on this figure that for all considered kinds of graphs, the achieved error does not significantly depend on the graph dimension. The Relative Complexity (RC= $\sum_{j=1}^{J} \|\mathbf{S}_j\|_0 / n^2$) of the computed FA$\mu$STs decreases when the dimension increases, it follows indeed a $\mathcal{O}(n^{-0.74})$ behaviour for $C_2 = 1.67$. More specifically with our setting we have RC= 0.59 for $n = 64$, RC= 0.37 for $n = 128$, RC= 0.23 for $n = 256$ and RC= 0.14 for $n = 512$. This means that we can factorize the Fourier matrices corresponding to various graphs, with FAμSTs having $\mathcal{O}(n^{\alpha})$ complexities with $\alpha < 2$. In other words, the greater the dimension, the higher the complexity savings for the same error, and approximate FFTs can be defined for these graphs.

However, we can see differences in approximation quality between the considered graphs. Indeed , the relative error with respect to the Laplacian is about 10^{-5} for the Swiss roll and the random ring graph, and about 10^{-1} for the Erdős-Rényi graph. Such differences may indicate properties of graphs favorable to Fast Fourier Transforms. It seems indeed that graphs with more structure (typically the random ring, related to non-uniform FFT [11]) have Fourier matrices that can be better approximated by FAμSTs than unstructured graphs (typically Erdős-Rényi).

3.2. FAμST factorization of U vs FAμST diagonalization of L

The objective of this subsection is to compare Algorithm 1 with a FAμST factorization of the Fourier matrix $\mathbf{U}$. For this experiment,

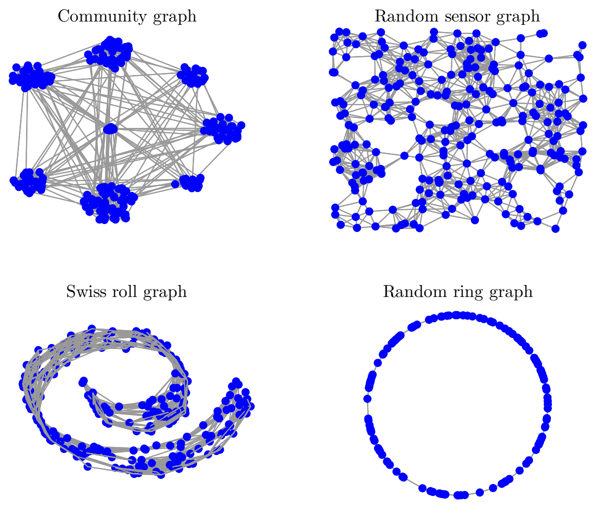


Fig. 3. Example of different graphs with $n = 256$ nodes (except the random ring that has $n = 128$ nodes).

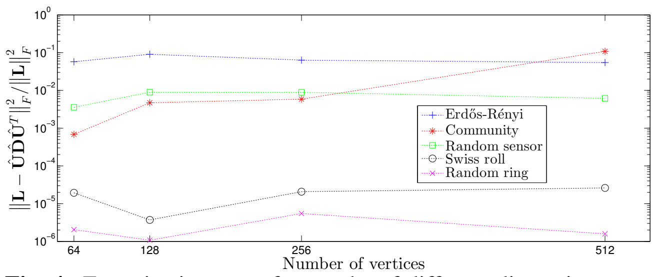


Fig. 4. Factorization error for graphs of different dimensions $n \in \{64, 128, 256, 512\}$. The mean over 10 independent trials is shown.

a random sensor graph with $n = 1024$ nodes is used. The Fourier matrix $\mathbf{U}$ is approximated using either Algorithm 1 (denoted *diagonalization* method in the sequel) or the factorization method that is described in details in [5] (denoted *factorization* method in the sequel). The parameters for both methods are set as follows:

- $C_1 = 3$
- $C_2 \in \{1.43, 1.67, 2\}$ (it corresponds to FAμSTs of complexity $\{\mathcal{O}(n^{1.48}), \mathcal{O}(n^{1.26}), \mathcal{O}(n \log_2 n)\}$)
- $C_3 \in \{1.3, 1.4, 1.5\}$.

We use the following performance measures:

- **Approximation of U:** we evaluate the closeness between the original Fourier matrix $\mathbf{U}$ and its FAμST approximation $\hat{\mathbf{U}}$ with the quantity $\|\hat{\mathbf{U}} - \mathbf{U}\|_F^2 / \|\mathbf{U}\|_F^2$ (this quantity is proportional to the cost function of the factorization).
- **Diagonalization of L:** we evaluate how close to diagonal would the Laplacian $\mathbf{L}$ be in the basis defined by the columns of $\hat{\mathbf{U}}$ if $\hat{\mathbf{U}}$ was orthogonal, by the quantity $\|\text{diag}(\hat{\mathbf{U}}^T\mathbf{L}\hat{\mathbf{U}})\|_F^2 / \|\hat{\mathbf{U}}^T\mathbf{L}\hat{\mathbf{U}}\|_F^2$.
- **Approximation of L:** we evaluate the closeness between the graph Laplacian $\mathbf{L}$ and its approximation by the quantity $\|\mathbf{L} - \hat{\mathbf{U}}\hat{\mathbf{D}}\hat{\mathbf{U}}^T\|_F^2 / \|\mathbf{L}\|_F^2$ (we consider that $\hat{\mathbf{D}} = \mathbf{\Sigma}$ in the factorization case, and this quantity is proportional to the cost function of the diagonalization).
- **Orthogonality of $\hat{\mathbf{U}}$:** we evaluate how close to orthogonal $\hat{\mathbf{U}}$ is with the quantity $\|\text{diag}(\hat{\mathbf{U}}\hat{\mathbf{U}}^T)\|_F^2 / \|\hat{\mathbf{U}}\hat{\mathbf{U}}^T\|_F^2$.

The results of these approximations are given in Figure 5. Several comments are in order. First of all, as expected, the approximation of $\mathbf{U}$ is better in the factorization case, compared to the diagonalization case, since it is the objective of the factorization. Similarly, the approximation of $\mathbf{L}$ is better in the diagonalization case. Second, the diagonalization of $\mathbf{L}$ is performed similarly by the two methods, except for low relative complexities where the diagonalization method seems better. Third, $\hat{\mathbf{U}}$ is closer to orthogonal using the diagonalization method, for any relative complexity. In conclusion, the diagonalization method yields approximate fast Fourier transforms $\hat{\mathbf{U}}$ that exhibit better properties in terms of orthogonality, approximation of the Laplacian and diagonalization of the Laplacian. On the other hand, they approximate the original Fourier matrix $\mathbf{U}$ less well than the FAμSTs computed using the factorization method, but this criterion seems less relevant than the others that measure properties that are intrinsically expected of the graph Fourier transform.

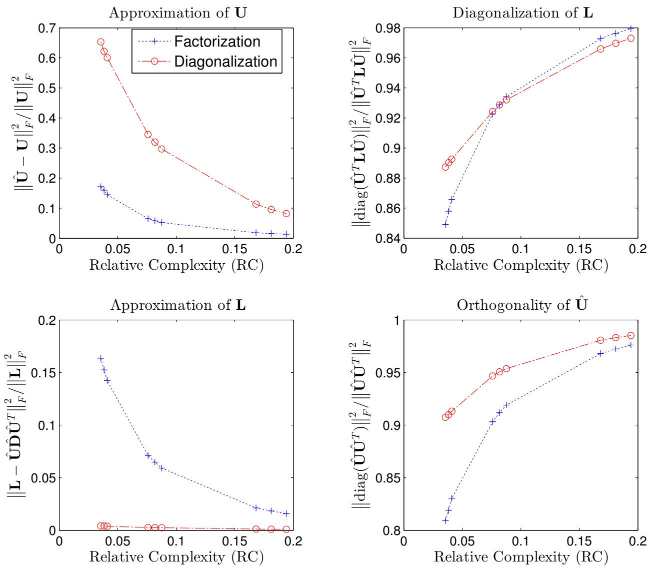


Fig. 5. Approximation results on the random sensor graph with $n = 1024$ nodes. The diagonalization and factorization methods are compared.

3.3. Fast filtering on the Minnesota road graph

The objective of this subsection is to use the FAμSTs approximations computed by Algorithm 1 and the factorization method of [5] for fast filtering on the Minnesota road graph, in order to show the applicability of our approach. This experiment should be taken as a proof of concept, showing that filtering signals on graphs using a FAμST $\hat{\mathbf{U}}$ instead of the true Fourier matrix $\mathbf{U}$ makes sense.

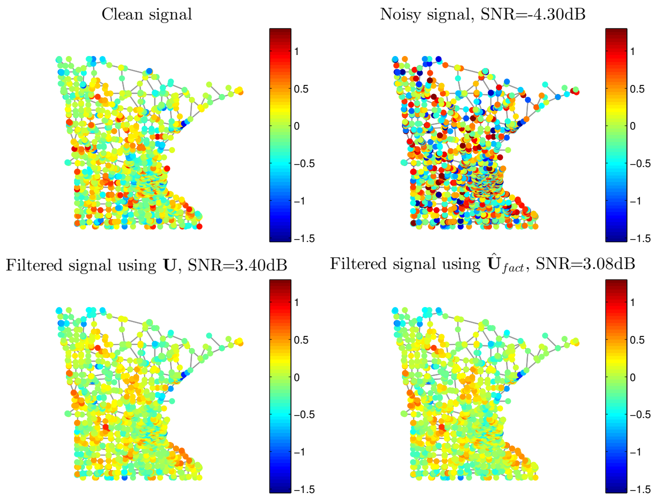
Signal model. For this experiment, a low-pass signal $\mathbf{x}$ is generated randomly in the graph frequency domain: the components of its spectrum are independent and follow a normal distribution of standard deviation $\theta = \exp(-f)$, where f is the frequency (eigenvalues of the Laplacian). This reference signal is then corrupted by a white Gaussian noise $\mathbf{n} \backsim \mathcal{N}(0, \sigma)$. This gives a noisy signal $\tilde{\mathbf{x}} = \mathbf{x} + \mathbf{n}$.

Filtering. The corrupted signal is low-pass filtered by a filter of frequency response $h(f) = 1/(1 + \gamma f)$ with $\gamma = 3$. The filtering is done using either the true Fourier transform matrix $\mathbf{U}$, a FAμST $\hat{\mathbf{U}}_{\mathrm{diag}}$ computed by diagonalization, or a FAμST $\hat{\mathbf{U}}_{\mathrm{fact}}$ computed by factorization. The setting presented in section 2.2 is used in both cases with $C_1 = 3$, $C_2 = 1.43$ and $C_3 = 1.5$. In this configuration, $\hat{\mathbf{U}}_{\mathrm{diag}}$ and $\hat{\mathbf{U}}_{\mathrm{fact}}$ exhibit a relative complexity of 0.13 (they are approximately eight times faster than $\mathbf{U}$).

Results. SNR results are given for different noise levels σ and in average over several realizations in Table 1. Filtering using a FAμST as an approximate Fast Fourier Transform on graph is shown to be almost as good as classical filtering using the actual Fourier matrix (less than one decibel of difference), although it is more computationally efficient (eight times). Moreover, the factorization technique seems to provide better filtering performance than the diagonalization. An example of filtering is shown in Figure 6.

	$\sigma = 0.3$	$\sigma = 0.4$	$\sigma = 0.5$	$\sigma = 0.6$
Noisy	1.82	-0.68	-2.65	-4.25
Filtered with $\mathbf{U}$	5.11	4.57	3.89	3.22
Filtered with $\hat{\mathbf{U}}_{\mathrm{diag}}$	4.04	3.62	3.11	2.60
Filtered with $\hat{\mathbf{U}}_{\mathrm{fact}}$	4.70	4.23	3.59	2.98

Table 1. Filtering results, the SNRs in decibels and in average over 100 independently drawn signals for each noise level are given.



Clean signal
Noisy signal, SNR=-4.30dB
Filtered signal using U, SNR=3.40dB
Filtered signal using $\hat{\mathbf{U}}_{fact}$, SNR=3.08dB

Fig. 6. Example of filtering on the Minnesota road graph. Classical filtering using $\mathbf{U}$ and filtering using a FAμST $\hat{\mathbf{U}}_{\mathrm{fact}}$ are shown.

4. DISCUSSION AND CONCLUSION

In this paper, we showed that many graphs admit an approximate fast Fourier transform, by approximately diagonalizing the graph Laplacian $\mathbf{L}$ with a FAμST $\hat{\mathbf{U}} = \prod_{j=1}^{J} \mathbf{S}_j$. The approach was validated on several random graphs, compared to a direct approximate factorization of the Fourier matrix $\mathbf{U}$ and tested on a fast graph signal filtering task with promising results.

Such an approach is well suited to situations where the graph is fixed, and the Fourier transform has to be applied rapidly a great number of times. This is to amortize the factorization cost. Such a situation corresponds for example to real-time monitoring of graph signals, where a threshold on the high-frequency components of the signal is set to detect anomalies, as is done in [8, section VI.A].

In future work, and in order to reduce the computation time, one could imagine to directly diagonalize the graph Laplacian without even requiring the true Fourier matrix as input. This would amount initialize $\mathbf{T}_0$ and $\mathbf{D}$ differently in Algorithm 1. Indeed, the initializations proposed in this paper ($\mathbf{T}_0 \leftarrow \mathbf{U}, \mathbf{D} \leftarrow \mathbf{\Sigma}$) require an exact diagonalization of the graph Laplacian prior to the hierarchical factorization. Another interesting perspective would be to take advantage of the versatility of the used optimization algorithm to enforce some properties on the factorization. For example, if the targeted application is filtering, one could imagine enforcing a better reconstruction of eigenvectors corresponding to small eigenvalues, used for low-pass filtering.

5. REFERENCES

[1] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre 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.

[2] Aliaksei Sandryhaila and Jose M.F. Moura, "Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure," *Signal Processing Magazine, IEEE*, vol. 31, no. 5, pp. 80–90, Sept 2014.

[3] James Cooley and John Tukey, "An algorithm for the machine calculation of complex Fourier series," *Mathematics of Computation*, vol. 19, no. 90, pp. 297–301, 1965.

[4] Jacques Morgenstern, "The linear complexity of computation," *J. ACM*, vol. 22, no. 2, pp. 184–194, Apr. 1975.

[5] Luc Le Magoarou and Rémi Gribonval, "Flexible multi-layer sparse approximations of matrices and applications," *CoRR*, vol. abs/1506.07300, 2015.

[6] Luc Le Magoarou and Rémi Gribonval, "Chasing butterflies: In search of efficient dictionaries," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, April 2015.

[7] Luc Le Magoarou, Rémi Gribonval, and Alexandre Gramfort, "FAμST: speeding up linear transforms for tractable inverse problems," in *EUSIPCO* , Nice, France, Aug. 2015.

[8] Aliaksei Sandryhaila and Jose M.F. Moura, "Discrete signal processing on graphs: Frequency analysis," *Signal Processing, IEEE Transactions on*, vol. 62, no. 12, pp. 3042–3054, 2014.

[9] Jérôme Bolte, Shoham Sabach, and Marc Teboulle, "Proximal alternating linearized minimization for nonconvex and nonsmooth problems," *Mathematical Programming*, vol. 146, no. 1-2, pp. 459–494, 2014.

[10] Nathanaël Perraudin, Johan Paratte, David Shuman, Vassilis Kalofolias, Pierre Vandergheynst, and David K. Hammond, "GSPBOX: A toolbox for signal processing on graphs," *ArXiv e-prints*, Aug. 2014.

[11] A. Dutt and V. Rokhlin, "Fast fourier transforms for nonequispaced data," *SIAM J. Sci. Comput.*, vol. 14, no. 6, pp. 1368–1393, Nov. 1993.