Analyzing the Approximation Error of the Fast Graph Fourier Transform

Luc Le Magoarou, Nicolas Tremblay, Rémi Gribonval

Analyzing the Approximation Error of the Fast Graph Fourier Transform

Luc LeMagoarou*, Nicolas Tremblay† and Rémi Gribonval‡
*b<>com, Rennes, France
†Univ. Grenoble Alpes, CNRS, GIPSA-lab, Grenoble, France
‡INRIA Rennes Bretagne-Atlantique, Rennes, France

***Abstract*—The graph Fourier transform (GFT) is in general dense and requires $\mathcal{O}(n^2)$ time to compute and $\mathcal{O}(n^2)$ memory space to store. In this paper, we pursue our previous work on the approximate fast graph Fourier transform (FGFT). The FGFT is computed via a truncated Jacobi algorithm, and is defined as the product of J Givens rotations (very sparse orthogonal matrices). The truncation parameter, J, represents a trade-off between precision of the transform and time of computation (and storage space). We explore further this trade-off and study, on different types of graphs, how is the approximation error distributed along the spectrum.**

I. Introduction

Recently, methods have been developed to analyze and process signals defined over the vertices of a graph [1], [2], instead of over a regular grid, as is classically assumed in discrete signal processing. The starting point of graph signal processing is to define a graph Fourier transform (GFT), via an analogy with classical signal processing. Depending on the preferred analogy, there exists several definitions of GFTs, all of them based on the diagonalisation of a graph operator, may it be the adjacency matrix [2], the Laplacian matrix [1], their degree-normalized versions, etc. We refer the reader to [3] for a recent review on different existing definitions of GFTs.

In this paper, we restrict ourselves to the study of the Laplacian-based Fourier matrix of undirected graphs. In this case, and denoting by n the number of vertices of the graph, the Laplacian $\mathbf{L} \in \mathbb{R}^{n\times n}$ (see the Methods Section for a precise definition) may be diagonalized as:

$$\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top, \tag{1}$$

where $\mathbf{U} \in \mathbb{R}^{n\times n}$ is an orthonormal matrix whose columns are the graph Fourier modes and $\mathbf{\Lambda} \in \mathbb{R}^{n\times n}$ is a non-negative diagonal matrix whose diagonal entries correspond to the graph's generalized frequencies.

The graph Fourier transform of a graph signal $\boldsymbol{x} \in \mathbb{R}^n$ is its scalar product with all Fourier modes, i.e., $\tilde{\boldsymbol{x}} = \mathbf{U}^\top\boldsymbol{x} \in \mathbb{R}^n$. This Fourier matrix $\mathbf{U}$ being in general dense, these n scalar products require $\mathcal{O}(n^2)$ computations. Nevertheless, in the classical signal processing case, the well-known Fast Fourier Transform (FFT) [4] allows to apply the Fourier transform in only $\mathcal{O}(n\log n)$ arithmetic operations. In fact, the FFT is a fast linear algorithm [5], which implies that the classical Fourier matrix can be factorized into sparse factors, as discussed in [6]. It is natural to wonder if this kind of factorization can be generalized to the graph Fourier transform, in order to tend towards a "fast" graph Fourier transform, scaling in $\mathcal{O}(n\log n)$, thus mimicking the classical case.

In [7], we propose such a factorization approach. More precisely, we develop a truncated Jacobi method that amounts to an approximate diagonalization of the Laplacian $\mathbf{L}$, as

$$\mathbf{L} \approx \mathbf{S}_1 \dots \mathbf{S}_J \hat{\mathbf{\Lambda}} \mathbf{S}_J^T \dots \mathbf{S}_1^T, \tag{2}$$

where the so-called Givens rotation matrices $\mathbf{S}_1, \dots, \mathbf{S}_J$ are both sparse *and* orthogonal, and $\hat{\mathbf{\Lambda}}$ is a diagonal matrix whose diagonal entries are approximations of the graph frequencies. J is typically chosen such as $J = o(n^2)$. The approximate fast graph Fourier transform of $\boldsymbol{x}$ reads $\hat{\mathbf{U}}^\top\boldsymbol{x} = \mathbf{S}_J^\top \dots \mathbf{S}_1^\top\boldsymbol{x}$. It is approximate because we stop the Jacobi algorithm before its convergence. It is fast because applying each Givens rotation $\mathbf{S}_i$ to a vector requires only 6 elementary operations (4 multiplications and 2 sums): computing $\hat{\mathbf{U}}^\top\boldsymbol{x}$ thus requires only $6J$ computations.

Contributions. In this paper, we build upon our previous work [7] and explore more precisely the approximation error due to the truncated Jacobi method: where in the spectrum are the errors localized, and what are the consequences from a graph signal processing point of view.

II. Methods

A. Notations, conventions and definitions

General notations. Matrices are denoted by bold upper-case letters: $\mathbf{A}$; vectors by bold lower-case letters: $\mathbf{a}$; the ith column of a matrix $\mathbf{A}$ by $\mathbf{a}_i$; its entry on the ith row and jth column by a_{ij}. Sets are denoted by calligraphic symbols: $\mathcal{A}$, and we denote by $\delta_{\mathcal{A}}$ the indicator function of the set $\mathcal{A}$ in the optimization sense ($\delta_{\mathcal{A}}(x) = 0$ if $x \in \mathcal{A}$, $\delta_{\mathcal{A}}(x) = +\infty$ otherwise). The standard vectorization operator is denoted $\mathrm{vec}(\cdot)$. The ℓ_0-pseudonorm is denoted $\|\cdot\|_0$ (it counts the number of non-zero entries), $\|\cdot\|_F$ denotes the Frobenius norm, and $\|\cdot\|_2$ the spectral norm. By abuse of notations, $\|\mathbf{A}\|_0 = \|\mathrm{vec}(\mathbf{A})\|_0$. The identity matrix is denoted $\mathbf{Id}$.

Graph Laplacian. We consider in this paper undirected weighted graphs, denoted $\mathcal{G} \triangleq \{\mathcal{V}, \mathcal{E}, \mathbf{W}\}$, where $\mathcal{V}$ represents the set of vertices (otherwise called nodes), $\mathcal{E} \subset \mathcal{V}\times\mathcal{V}$ is the set of edges, and $\mathbf{W}$ is the weighted adjacency matrix of the graph. We denote $n \triangleq |\mathcal{V}|$ the total number of vertices and the adjacency matrix $\mathbf{W} \in \mathbb{R}^{n\times n}$ is symmetric and such that $w_{ij} = w_{ji}$ is non-zero only if $(i,j) \in \mathcal{E}$ and represents

the strength of the connection between nodes i and j. We define the degree matrix $\mathbf{D} \in \mathbb{R}^{n\times n}$ as a diagonal matrix with $\forall i,\ d_{ii} \triangleq \sum_{j=1}^{n} w_{ij}$, and the combinatorial Laplacian matrix $\mathbf{L} \triangleq \mathbf{D} - \mathbf{W}$ (we only consider this type of Laplacian matrix in this paper, and hereafter simply call it the Laplacian).

Givens rotations. An n-dimensional Givens rotation [8] is a linear transformation that does not act on $n-2$ coordinates and rotates the two remaining by an angle $\theta \in [0; 2\pi[$. Noting p and q the indices of the two rotated coordinates, Givens rotations thus correspond to matrices of the following form,

$$\mathbf{G}_{p,q,\theta} = \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ & & & c & & -s & \\ & & & & 1 & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{pmatrix},$$

where $c \triangleq \cos(\theta)$ and $s \triangleq \sin(\theta)$. A Givens rotation only depends on three parameters: the two coordinates p and q and the rotation angle θ, hence the notation $\mathbf{G}_{p,q,\theta}$.

Input: matrix $\mathbf{L}$, target number J of Givens rotations.
1: $\mathbf{L}_1 \leftarrow \mathbf{L}$
2: **for** $j = 1$ to J **do**
3: $\quad \mathbf{S}_j \leftarrow \operatorname{argmin}_{\mathbf{S}\in\mathcal{S}} \left\|\mathbf{S}^T\mathbf{L}_j\mathbf{S}\right\|^2_{\text{offdiag}}$
4: $\quad \mathbf{L}_{j+1} \leftarrow \mathbf{S}_j^T\mathbf{L}_j\mathbf{S}_j$
5: **end for**
6: $\hat{\mathbf{\Lambda}} \leftarrow \operatorname{diag}(\mathbf{L}_{J+1})$
7: Sort diagonal entries of $\hat{\mathbf{\Lambda}}$ in increasing order. Reorder columns of $\mathbf{S}_J$ accordingly.
Output: sparse orthogonal factors $\mathbf{S}_1, \dots, \mathbf{S}_J$; diagonal factor $\hat{\mathbf{\Lambda}}$.

Fig. 1: **Truncated Jacobi algorithm**: Approximate diagonalization algorithm with prescribed complexity.

Input: matrix $\mathbf{L}_j$.
1: $(p,q) \leftarrow \operatorname{argmax}_{(r,s)\in[n]^2} |l^j_{rs}|$
2: $\theta \leftarrow \frac{1}{2}\arctan(\frac{l^j_{qq}-l^j_{pp}}{2l^j_{pq}}) + \frac{\pi}{4}$
3: $\mathbf{S}_j \leftarrow \mathbf{G}_{p,q,\theta}$
Output: matrix $\mathbf{S}_j = \operatorname{argmin}_{\mathbf{S}\in\mathcal{S}} \left\|\mathbf{S}^T\mathbf{L}_j\mathbf{S}\right\|^2_{\text{offdiag}}$.

Fig. 2: **Resolution of subproblem** (SP)

B. Objective

Our goal is to approximately diagonalize the Laplacian $\mathbf{L}$ with an efficient approximate eigenvector matrix $\hat{\mathbf{U}} = \mathbf{S}_1 \dots \mathbf{S}_J$, where the factors $\mathbf{S}_1, \dots, \mathbf{S}_J \in \mathbb{R}^{n\times n}$ are Givens rotations. Using the Frobenius norm to measure the quality of approximation, this objective can be stated as the following optimization problem:

$$\begin{aligned} \underset{\hat{\mathbf{\Lambda}},\mathbf{S}_1,\dots,\mathbf{S}_J}{\text{minimize}} \quad & \left\|\mathbf{L} - \mathbf{S}_1 \dots \mathbf{S}_J \hat{\mathbf{\Lambda}} \mathbf{S}_J^T \dots \mathbf{S}_1^T\right\|_F^2 \\ & + \textstyle\sum_{j=1}^{J} \delta_{\mathcal{S}}(\mathbf{S}_j) + \delta_{\mathcal{D}}(\hat{\mathbf{\Lambda}}), \end{aligned} \tag{DP}$$

where $\mathcal{D}$ is the set of diagonal matrices and $\mathcal{S}$ is the set of Givens rotations.

C. Optimization framework

The truncated Jacobi algorithm. To approximately solve problem (DP), one can rely on a truncated version of the classical Jacobi eigenvalues algorithm [9], [10]. The Jacobi algorithm is an iterative procedure where at each step one seeks the Givens rotation reducing the most the cost function. At the first step, this means setting $\mathbf{S}_1$ and $\hat{\mathbf{\Lambda}}$ as follows:

$$(\mathbf{S}_1, \hat{\mathbf{\Lambda}}) \leftarrow \underset{\mathbf{D}\in\mathcal{D},\mathbf{S}\in\mathcal{S}}{\operatorname{argmin}} \quad \left\|\mathbf{L} - \mathbf{S}\mathbf{D}\mathbf{S}^T\right\|_F^2,$$

which can be reformulated, given that the Frobenius norm is invariant under orthogonal transformations, as

$$(\mathbf{S}_1, \hat{\mathbf{\Lambda}}) \leftarrow \underset{\mathbf{D}\in\mathcal{D},\mathbf{S}\in\mathcal{S}}{\operatorname{argmin}} \quad \left\|\mathbf{S}^T\mathbf{L}\mathbf{S} - \mathbf{D}\right\|_F^2.$$

Since $\mathcal{D}$ is the set of all diagonal matrices, the optimal estimated generalized frequencies factor is simply $\hat{\mathbf{\Lambda}} = \operatorname{diag}(\mathbf{S}^T\mathbf{L}\mathbf{S})$. This allows to rule out this factor of the problem and to reformulate it as follows:

$$\mathbf{S}_1 \leftarrow \underset{\mathbf{S}\in\mathcal{S}}{\operatorname{argmin}} \quad \left\|\mathbf{S}^T\mathbf{L}\mathbf{S}\right\|^2_{\text{offdiag}},$$

where $\|\mathbf{A}\|^2_{\text{offdiag}}$ is the sum of the squared off-diagonal entries of $\mathbf{A}$. Once the factor $\mathbf{S}_1$ is set this way, and introducing the notation $\mathbf{L}_2 \triangleq \mathbf{S}_1^T\mathbf{L}\mathbf{S}_1$, the next step of the strategy is to choose $\mathbf{S}_2 \leftarrow \operatorname{argmin}_{\mathbf{S}\in\mathcal{S}} \left\|\mathbf{S}^T\mathbf{L}_2\mathbf{S}\right\|^2_{\text{offdiag}}$, and so on until the Jth and last step. The algorithm thus amounts to solve a sequence of J very similar subproblems of the form

$$\underset{\mathbf{S}\in\mathcal{S}}{\text{minimize}} \quad \left\|\mathbf{S}^T\mathbf{L}_j\mathbf{S}\right\|^2_{\text{offdiag}}, \tag{SP}$$

with $\mathbf{L}_j \triangleq \mathbf{S}_{j-1}^T\mathbf{L}_{j-1}\mathbf{S}_{j-1}$. This is summarized by the algorithm of Figure 1. Compared to the traditional Jacobi eigenvalues algorithm [9], [10], where new Givens rotations are chosen until a certain accuracy is attained, the main difference is that by prescribing the number J of Givens rotations we can adjust the trade-off between accuracy and computational efficiency of the product $\mathbf{S}_1 \dots \mathbf{S}_J$.

Remark II.1. *Note that in order for the approximate FGFT $\hat{\mathbf{U}} = \mathbf{S}_1 \dots \mathbf{S}_J$ to make sense in a graph signal processing context, we reorder its columns according to the estimated eigenvalues (line 7 of the algorithm). This way, the first columns of $\hat{\mathbf{U}} = \mathbf{S}_1 \dots \mathbf{S}_J$ "physically" correspond to low frequencies, and its last columns to high frequencies.*

Subproblem resolution. The algorithm requires to solve J times the optimization subproblem (SP) (at line 3 of the

algorithm of Figure 1). The solution of this subproblem is given by the Givens rotation $\mathbf{G}_{p,q,\theta}$, where the indices p and q correspond to the greatest entry of $\mathbf{L}_j$ in absolute value (denoted $|l^j_{pq}|$), and the rotation angle has the expression $\theta = \frac{1}{2}\arctan(\frac{l^j_{qq}-l^j_{pp}}{2l^j_{pq}}) + (2k+1)\frac{\pi}{4}$, $k \in \mathbb{Z}$. We then have $\|\mathbf{L}_{j+1}\|^2_{\text{offdiag}} = \|\mathbf{L}_j\|^2_{\text{offdiag}} - 2(l^j_{pq})^2$. For a proof as well as a review of the different implementations and variants of the Jacobi algorithm, see [11, pages 426-435]. Figure 2 details the algorithm to solve subproblem (SP).

Refinements. This truncated Jacobi algorithm requires $\mathcal{O}(n^2 + nJ)$ operations to compute. We proposed in [7] a parallel framework that reduces this complexity to $\mathcal{O}(nJ\log n)$. Nevertheless, in this paper, we will not concentrate on the cost of *obtaining* the sparse matrix factorization; we will rather concentrate on the tradeoff between the cost of *applying* $\hat{\mathbf{U}}$ to a graph signal, which, in both cases (truncated or parallel truncated Jacobi), is $\mathcal{O}(J)$, and the quality of the approximation of $\mathbf{U}$ by $\hat{\mathbf{U}}$. The codes of this truncated Jacobi algorithm are freely available at `https://faust.inria.fr/`.

III. Experiments

In the following, let us denote by $\mathbf{u}_k$ the k-th eigenvector of $\mathbf{L}$; i.e., $\mathbf{U} = (\mathbf{u}_1|\dots|\mathbf{u}_n)$. Let us also denote by $\hat{\mathbf{u}}^J_k$ the k-th approximate eigenvector of $\mathbf{L}$; i.e., $\hat{\mathbf{U}} = \mathbf{S}_1 \dots \mathbf{S}_J = (\hat{\mathbf{u}}^J_1|\dots|\hat{\mathbf{u}}^J_n)$. Given this approximate fast graph Fourier matrix $\hat{\mathbf{U}}$, a natural question is to investigate how well does this matrix approximate the real graph Fourier matrix $\mathbf{U}$ as a function of J, the number of Givens rotations[1]. To conduct this analysis, we propose the following experiments.

A. Types of graphs

We consider three types of random graphs:

- **Erdos-Renyi graphs**. We set the number of nodes n to 128. All pairs of nodes are connected with probability p. The average degree of such a model is $c = p(n-1)$.
- **Stochastic Block Model (SBM)**, a random community-structured graph model. We specifically look at graphs with m communities of same size n/m. In the SBM, the probability of connection between any two nodes i and j is q_1 if they are in the same community, and q_2 otherwise. One can show that the average degree reads $c = q_1\left(\frac{n}{k} - 1\right) + q_2\left(n - \frac{n}{k}\right)$. Thus, instead of providing the probabilities (q_1, q_2), one may characterize a SBM by considering $(\epsilon = \frac{q_2}{q_1}, c)$. The larger ϵ, the fuzzier the community structure. In fact, authors in [12] show that above the critical value $\epsilon_c = (c - \sqrt{c})/(c + \sqrt{c}(m-1))$, community structure becomes undetectable in the large n limit. In the following, we set $n = 128$ and $m = 8$.
- **Sensor graphs.** $n = 128$ sensors (points in two dimensions) are randomly and uniformly generated on the unit square. Two sensors i and j are connected if their interdistance is inferior to a given threshold τ. The obtained graph is binary. The average degree c of such a model is not explicit.

For the sensor graph, we set $\tau = 0.161$, which yields graphs with average degree $c = 10$. In experiments with Erdos-Renyi and SBM graphs, we enforce the same average degree by setting $p = c/(n-1) = 10/127$ for the Erdos-Renyi model, and $c = 10$ for the SBM model (besides, ϵ is set to $\epsilon_c/10$).

B. Analysis of the approximation errors

In [7], we concentrated on global approximation error measures such as:

$$\frac{\|\mathbf{U} - \hat{\mathbf{U}}\|_F}{\|\mathbf{U}\|_F}. \tag{3}$$

This measure does not differentiate *where* in the spectrum are the errors. The goal of this paper is to refine this analysis. First of all, note that:

$$\begin{aligned}\frac{\|\mathbf{U} - \hat{\mathbf{U}}\|^2_F}{\|\mathbf{U}\|^2_F} &= \frac{\|\mathbf{Id} - \mathbf{U}^\top\hat{\mathbf{U}}\|^2_F}{\|\mathbf{U}\|^2_F} \\ &= \frac{1}{\|\mathbf{U}\|^2_F}\sum_{k=1}^{n}\sum_{j=1}^{n}(\delta_{jk} - \mathbf{u}_j^\top\hat{\mathbf{u}}^J_k)^2 \\ &= \frac{1}{n}\sum_{k=1}^{n}\text{err}_1(k,J)^2,\end{aligned}$$

where we consider the fine-grain error measure

$$\text{err}_1(k,J)^2 := \left\|\boldsymbol{\delta}_k - \mathbf{U}^\top\hat{\mathbf{u}}^J_k\right\|^2_2 \tag{4}$$

that enables a finer analysis of the approximation error (3).

In Figure 3, we show $\text{err}_1(k,J)^2$ as a function of k and J for the three random models detailed above. We readily observe very different behaviors. Firstly, for small values of J, the approximation error is large and somewhat uniform over the whole spectrum, except for high values of k where the error drops. Moreover, as J increases, the approximation error decreases, but not uniformly: the error decreases faster for some values of k. In the Erdos-Renyi example, the approximation error decreases faster on the boundaries (small or large values of k) than in the middle. In the random sensor graph example, the approximation error decreases faster around $k = 10$ and for large values of k. For the SBM graph, the error decreases very fast around $k = 8$ (notice that this coincides with $k = m$, the number of communities), and faster on the boundaries than in the middle. In the following, we give tentative explanations accounting for these differences.

Normalized error measure. For $J = 0$ we have $\hat{\mathbf{U}} = \mathbf{Id}$, up to column permutation (cf Remark II.1) according to the diagonal of $\hat{\mathbf{U}}^\top\mathbf{L}\hat{\mathbf{U}} = \mathbf{L}$, i.e. with respect to $\mathbf{D}$, the degree matrix. Taking into account this ordering yields

$$\hat{\mathbf{U}} = \left(\boldsymbol{\delta}_{\sigma(1)}|\boldsymbol{\delta}_{\sigma(2)}|\dots|\boldsymbol{\delta}_{\sigma(n)}\right), \tag{5}$$

[1]Note that the orientation of $\mathbf{u}^J_i$ is arbitrary: both $\hat{\mathbf{u}}^J_i$ and $-\hat{\mathbf{u}}^J_i$ are equivalent from a diagonalisation point of view. For a fair comparison with $\mathbf{u}_i$, the orientation of $\hat{\mathbf{u}}^J_i$ is decided by settling that $\mathbf{u}_i^\top\hat{\mathbf{u}}^J_i$ should be positive.

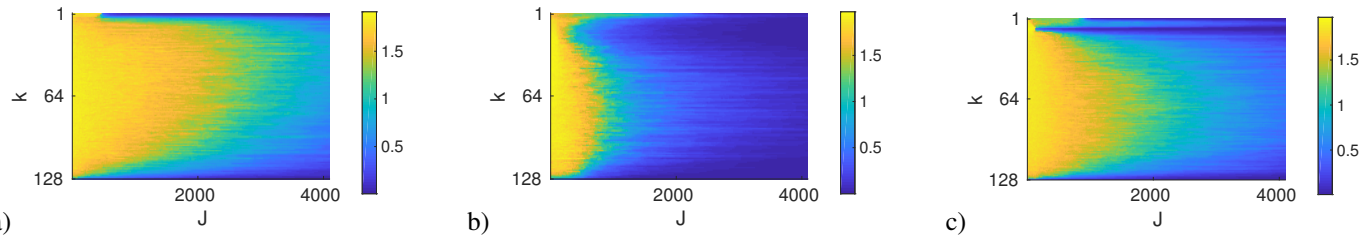

Fig. 3: Median of the error $\mathrm{err}_1(k,J)^2$ over 100 random draws: a) Erdos-Renyi graph, b) sensor graph, c) SBM graph.

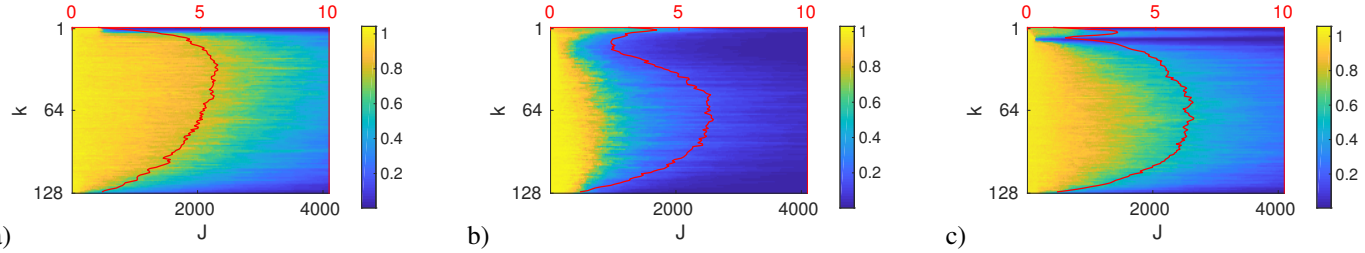

Fig. 4: Median of the (degree-corrected) normalized error $\widetilde{\mathrm{err}}_1(k,J)^2$ over 100 random draws: a) Erdos-Renyi graph, b) sensor graph, c) SBM graph. The red line represents a measure of eigenvalue density as a function of k (see Eq. (7)).

with σ a permutation such that $d_{\sigma(1),\sigma(1)} \leqslant d_{\sigma(2),\sigma(2)} \leqslant \ldots \leqslant d_{\sigma(n),\sigma(n)}$. In particular $\sigma(n)$ indexes the node with largest degree, and one has:

$$\mathrm{err}_1(k,0)^2 = \left\|\boldsymbol{\delta}_k - \mathbf{U}^\top \boldsymbol{\delta}_{\sigma(k)}\right\|_2^2.$$

As shown on Figure 5-a, this error is not evenly distributed along the spectrum: higher values of k show a comparatively lower approximation error. As $\mathbf{U}^\top \boldsymbol{\delta}_{\sigma(k)} = (u_1(\sigma(k)),\ldots,u_n(\sigma(k)))^\top$, this non-uniformity shows in fact that high degree nodes tend to contribute more to high-frequency Fourier modes than other nodes. In other words, high frequency Fourier modes tend to localize more on high degree nodes, at least on the three random graph models considered here.

To validate that this accounts for the non-uniformity observed for large k and small J on Figure 3, we define a normalized error measure as:

$$\widetilde{\mathrm{err}}_1(k,J)^2 = \frac{\mathrm{err}_1(k,J)^2}{\mathrm{err}_1(k,0)^2}. \tag{6}$$

As shown on Figure 4, the non-uniformity at low values of J is corrected, but the heterogeneity over the spectrum of the decrease of the error with respect to k is still unaccounted for.

Eigenvalue density. The remaining heterogeneity can be conjectured to be due to the inhomogeneity of the spectrum density: in the extreme case of eigenvectors associated to an eigenvalue of multiplicity greater than one, the error cannot be small. To elaborate, define an eigenvalue density measure:

$$f(k) = \#\{i \text{ s.t. } \lambda_i \in [\lambda_k - \Delta, \lambda_k + \Delta]. \tag{7}$$

If λ_k is isolated and there are no other eigenvalues in its vicinity (defined by $\pm\Delta$), then $f(k) = 1$. Whereas if λ_k is in a high-density part of the spectrum, then $f(k)$ is large. Superimposed on Figure 4 representing $\widetilde{\mathrm{err}}_1(k,J)$, we plot

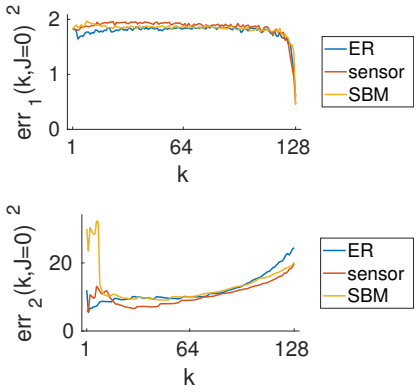

Fig. 5: Median error $\mathrm{err}_1(k,0)^2$ (top) [resp. $\mathrm{err}_2(k,0)^2$ (bottom)] over 100 draws.

$f(k)$ (computed with $\Delta = 0.25$) as a function of k. We indeed observe an interesting correlation between the approximation error for k and the density of the spectrum around λ_k: the higher the eigenvalue density, the larger the error.

To validate this correlation, we define an error measure that probes how well $\hat{\mathbf{u}}_k^J$ approximates $\mathbf{u}_k$ as an eigenvector of $\mathbf{L}$:

$$\mathrm{err}_2(k,J)^2 = \left\|\mathsf{L}\hat{\boldsymbol{u}}_k^J - \lambda_k \hat{\boldsymbol{u}}_k^J\right\|_2^2, \tag{8}$$

where λ_k is the exact k-th eigenvalue of $\mathbf{L}$. This measure may be rewritten as:

$$\mathrm{err}_2(k,J)^2 = \left\|\mathsf{U}\Lambda\mathsf{U}^\top \hat{\boldsymbol{u}}_k^J - \lambda_k \hat{\boldsymbol{u}}_k^J\right\|_2^2 = \left\|\Lambda\mathsf{U}^\top \hat{\boldsymbol{u}}_k^J - \lambda_k \mathsf{U}^\top \hat{\boldsymbol{u}}_k^J\right\|_2^2$$
$$= \sum_{j=1}^{n} (\lambda_j - \lambda_k)^2 (\boldsymbol{u}_j^\intercal \hat{\boldsymbol{u}}_k^J)^2.$$

To illustrate the difference between the two errors, say $\hat{\boldsymbol{u}}_k^J$ is a weighted sum of two true eigenvectors $\hat{\boldsymbol{u}}_k^J = \alpha \boldsymbol{u}_k + \beta \boldsymbol{u}_{k+1}$, with $\alpha^2 + \beta^2 = 1$. The first error measures $\mathrm{err}_1^2 =$

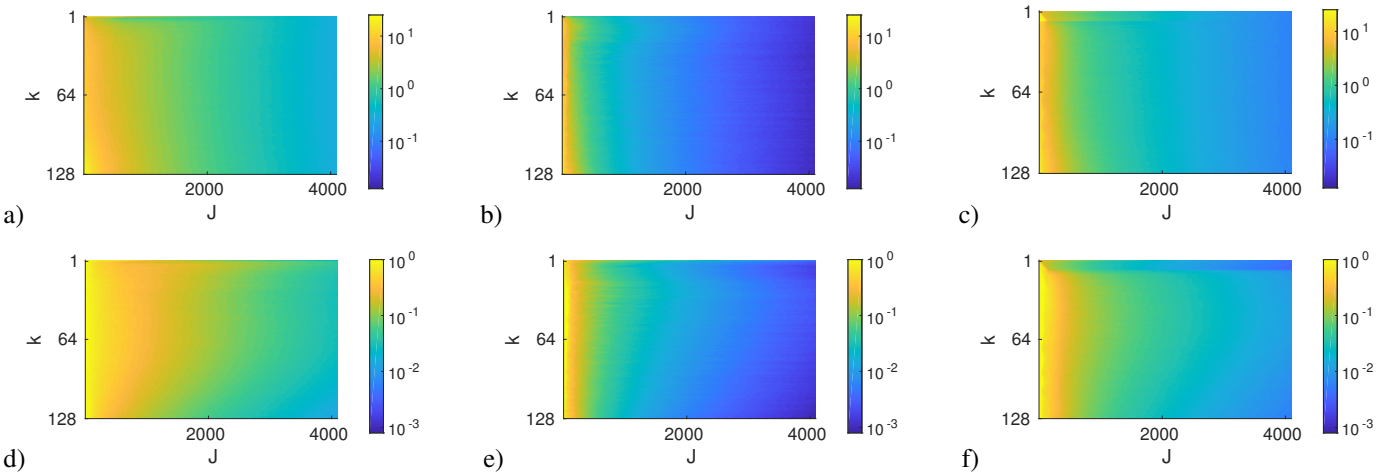

Fig. 6: The first (resp. second) line of figures represents $\mathrm{err}_2(k,J)^2$ (resp. $\widetilde{\mathrm{err}}_2(k,J)^2$). Note that the color scale is logarithmic. Results shown here are the median behaviour over 100 realisations of the underlying random model: a+d) Erdos-Renyi graph, b+e) sensor graph, c+f) SBM graph.

$2(1-\alpha)$ and does not depend on the distance between λ_{k+1} and λ_k, whereas the second error measures $\mathrm{err}_2^2 = (\lambda_{k+1} - \lambda_k)^2(1-\alpha^2)$, meaning that if $\hat{\boldsymbol{u}}_k^J$ is not exactly $\boldsymbol{u}_k$ but close to it from a "spectral point-of-view", then the error is small. Once again, one may normalize this error to correct for degree heterogeneities, using the notations of (5) to write:

$$\mathrm{err}_2(k,0)^2 = \left\| \mathbf{L}\boldsymbol{\delta}_{\sigma(k)} - \lambda_k \boldsymbol{\delta}_{\sigma(k)} \right\|_2^2 .$$

Figure 5-b shows how $\mathrm{err}_2(k,0)$ varies with k. We observe how it increases at extreme values of k (very small or very large). Interestingly, for the SBM, the large error at small k is due to the community structure that imposes the first $m = 8$ eigenvalues (because there are 8 communities in the model) to be very small. We checked (experiments not displayed here) that decreasing the strength of the community structure (by increasing ϵ to values closer to ϵ_c) increases the first eigenvalues, and decreases the error at $J = 0$.

Figure 6 displays $\mathrm{err}_2(k,J)$ and its normalized version $\widetilde{\mathrm{err}}_2(k,J)$. For visualisation purposes, the color scale is logarithmic. This new measure drastically decreases the inhomogeneity of the error over the spectrum.

IV. Conclusion

In this paper, the approximation capabilities of FGFTs were empirically studied. In order to do so, the FGFT of graphs pertaining to various families were computed and their approximation errors analyzed with respect to the number of used Givens rotations. Two main conclusions can be drawn. First, high frequency modes tend to be localized around high degree nodes, and are thus well approximated with few Givens rotations (that are very sparse matrices). Second, Fourier modes corresponding to dense regions of the spectrum are more difficult to disentangle, and thus need more Givens rotations to be well approximated. Stated differently, if one wishes to exactly recover the ordering of the eigenmodes, FGFTs perform poorly in the dense parts of the spectrum. On the other hand, if one only whishes to approximate eigenspaces in the vicinity of a given eigenvalue, the FGFT is an interesting tool to consider and leads to fast implementations.

V. Acknowledgments

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program Investissement d'avenir.

References

[1] 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, 2013.

[2] A. Sandryhaila and J. Moura, "Discrete signal processing on graphs," *Signal Processing, IEEE Transactions on*, vol. 61, no. 7, 2013.

[3] N. Tremblay, P. Goncalves, and P. Borgnat, "Design of graph filters and filterbanks," oct 2017. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01243889

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

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

[6] L. Le Magoarou and R. Gribonval, "Flexible multilayer sparse approximations of matrices and applications," *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 4, pp. 688–700, 2016.

[7] L. L. Magoarou, R. Gribonval, and N. Tremblay, "Approximate fast graph Fourier transforms via multi-layer sparse approximations," *IEEE Transactions on Signal and Information Processing over Networks*, vol. PP, no. 99, pp. 1–1, 2017.

[8] W. Givens, "Computation of plane unitary rotations transforming a general matrix to triangular form," *Journal of the Society for Industrial and Applied Mathematics*, vol. 6, no. 1, pp. pp. 26–50, 1958.

[9] C. G. J. Jacobi, "Über ein leichtes verfahren, die in der theorie der säkularstörungen vorkommenden gleichungen numerisch aufzulösen," *J. reine angew. Math.*, vol. 30, pp. 51–94, 1846.

[10] G. H. Golub and H. A. Van der Vorst, "Eigenvalue computation in the 20th century," *Journal of Computational and Applied Mathematics*, vol. 123, no. 1, pp. 35–65, 2000.

[11] G. H. Golub and C. F. Van Loan, *Matrix computations*. JHU Press, 2012, vol. 3.

[12] A. Decelle, F. Krzakala, C. Moore, and L. ZdeborovÃ¡, "Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications," *Phys. Rev. E*, vol. 84, no. 6, p. 066106, 2011.