Random Multi-Type Spanning Forests for Synchronization on Sparse Graphs *

Hugo Jaquard, Pierre-Olivier Amblard, Simon Barthelme, Nicolas Tremblay

Random Multi-Type Spanning Forests for Synchronization on Sparse Graphs*

Hugo Jaquard, Pierre-Olivier Amblard, Simon Barthelmé, Nicolas Tremblay
CNRS, Univ. Grenoble Alpes, Grenoble INP, GIPSA-lab, France

Abstract. Random diffusions are a popular tool in Monte-Carlo estimations, with well established algorithms such as Walk-on-Spheres (WoS) going back several decades. In this work, we introduce diffusion estimators for the problems of angular synchronization and smoothing on graphs, in the presence of a rotation associated to each edge. Unlike classical WoS algorithms, these estimators allow for *global* estimations by propagating along the branches of *multi-type spanning forests*, and we show that they can outperform standard numerical-linear-algebra solvers in challenging instances, depending on the topology and density of the graph.

Key words. RandNLA, graphs, random spanning forests, connection, graph signal processing, synchronization

AMS subject classifications. 68W20, 68Q87, 60G35, 62M99

Introduction. Data processing over graphs is of interest in many fields of research, such as discrete geometry, signal processing or graph machine learning for instance. A common setting is that of nodes supporting some kind of data, typically numerical values, whereas the edges describe the topology of the underlying space. In a number of situations, the edges can carry more precise geometric information regarding *how* two data points defined on adjacent nodes should compared, and thus processed. In geometrical parlance, this information is known as a *connection* (an idea first formalized in [13]).

We focus here on a connection describing rotations of angle $\theta_{i,j}$ associated to each edge (i, j), a setting which has found a growing number of applications over the last decade, including *angular synchronization* [63, 71], signal processing over directed graphs [24, 73], discrete geometry processing [60], or even direction of arrival estimation [48, 54]. Solving these problems requires computing the solution to large numerical-linear-algebra (NLA) problems, whose formulation relies on the *connection Laplacian* L_θ (the formal definition of L_θ is postponed to Section 1.2), a linear operator encoding both the topology of the graph *and* the rotations $\theta_{i,j}$.

Computing an exact solution to these problems (*e.g.*, the smoothing problem we will consider requires solving a linear system, with generic time complexity in $\mathcal{O}(n^3)$) remains prohibitive beyond moderately large graphs ($n \simeq 10^4$), so that approximations computed from iterative Krylov-subspaces-based algorithms are typically used instead [56, 57]. The main drawback of these methods is that their convergence can depend in a complex manner on the spectrum of the matrix at hand [37] (linear-system solvers for instance require good-quality preconditioners to ensure fast convergence).

Randomized numerical linear algebra (RandNLA) is a successful and modern alternative [21, 45]: Monte-Carlo estimators allow flexible schemes of computation (*e.g.*, parallelized or distributed) and can exhibit both advantageous complexities and state-of-the-art practical performances, *in spite of* their slow convergence rates in $O\left(\frac{\sigma}{m}\right)$ (with σ^2 the variance of the estimator and m the number of Monte-Carlo samples). Methods specialized in graphs, based

*This work is an extension of the conference proceedings [31].

on random walks and decompositions of the graph, have also appeared (*e.g.* [39, 51, 65]).

In this paper, we leverage novel connection-aware random decompositions of the graph, and propose RandNLA estimators for two connection-Laplacian-based problems:

- A graph Tikhonov smoothing problem in the presence of a connection.
- The angular synchronization problem on graphs.

Tikhonov smoothing. Connection-aware graph Tikhonov smoothing amounts to solving:

$$\underset{f\in\mathbf{C}^n}{\operatorname{argmin}}\ q\|f-g\|_2^2+\frac{1}{2}\sum_{i,j}|f(j)-e^{\iota\theta_{i,j}}f(i)|^2, \tag{0.1}$$

where $g\in\mathbf{C}^n$, $q\in\mathbf{R}_+^*$ is an *a priori* known regularization parameter, and the sum is over a subset of ordered pairs of indices (i,j) describing the edges of the graph. This problem appears in different contexts, such as vector field extension in discrete geometry processing [60], or directed graph signal processing [24]. Here, the regularization term penalizes functions that are not *locally coherent* with respect to the connection, and can be expressed using the connection Laplacian L_θ. The solution to this first problem takes the form $q(\mathsf{L}_\theta+q\mathsf{I})^{-1}g$, and can be leveraged in the second problem we consider: angular synchronization.

Angular Synchronization. The objective is to recover a set of n unknown angles $\omega=(\omega)_i\subseteq[0,2\pi)^n$ from measured pairwise offset measurements $\{\theta_{i,j}\}_{i,j}$ [63] :

$$\theta_{i,j}=\omega_j-\omega_i+\varepsilon_{i,j}\mod 2\pi, \tag{0.2}$$

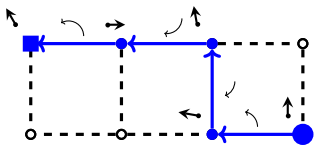
where $\varepsilon_{i,j}$ represents some unknown degradation of the measurement. This task appears in many structured signal processing problems, where it is often a key component in state-of-the-art recovery methods, such as perceived luminance reconstruction [70], ptychography [23], ranking [16], clock synchronization [25] or phase reconstruction [1], and also appears in the statistical physics literature (*e.g.* [14]).

In practice, we may only observe a subset of all such measurements $\theta_{i,j}$, and the problem is naturally formulated on a graph $\mathcal{G}$ with n nodes whose set of edges $\mathcal{E}$ is indexed by the number of measurements, and where edge (i,j) (resp. (j,i)) carries the offset $\theta_{i,j}$ (resp. $(\theta_{j,i}=-\theta_{i,j})$). If noiseless measurements $\theta_{i,j}=\omega_j-\omega_i$ are available, exact recovery can be trivially performed up to a global phase shift, by *propagating* values according to the offset measurements along a spanning tree of $\mathcal{G}$ (a procedure explained in Figures 0.1a and 0.1b).

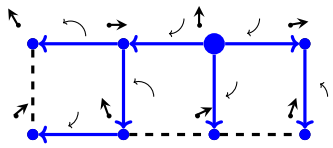
The problem becomes more involved when considering imperfect measurements $\theta_{i,j}$, with non-zero noise $\varepsilon_{i,j}$. In general, *exact* angular synchronization can no longer be performed in this noisy regime, as long as even one *incoherent* cycle is present in the graph (see Figure 0.1c). A common workaround, first introduced in [70], consists in quantifying the incoherence of an angular assignment $s\in[0,2\pi)^n$ as

$$\mathcal{I}(s)=\sum_{\{i,j\}\in\mathcal{E}}\left(2-2\cos\left((s_j-s_i)-\theta_{i,j}\right)\right), \tag{0.3}$$

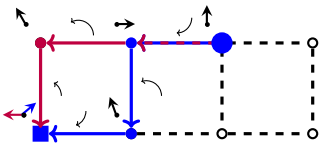
before minimizing this incoherence over all possible assignments. While this problem is non-convex and NP-hard in general [72] (see also [10] for a discussion), different techniques allowing to recover a solution have been proposed [10, 27, 50, 63, 71], be it approximately via relaxations or for specific noise regimes or topologies. Most of those techniques also rely on the introduction of the *connection Laplacian* L_θ (see Section 1.2).



(a) Propagation from the large blue node (bottom left) to the large blue square (top right) along the path in blue. A rotation is applied when traversing each edge.



(b) Synchronization by propagation from the large blue node to all other nodes in the graph, along the branches of the blue spanning tree (only possible if there is no incoherence).



(c) Propagation from the large blue node to the large blue rectangle (bottom left), along two different paths (blue and purple). Here, the rotations are incoherent and the result of the propagation depends on the specific path.

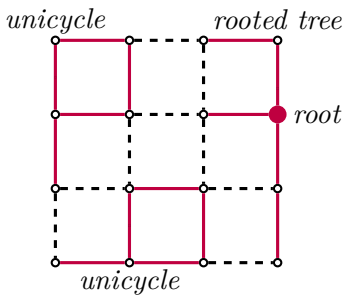
Figure 0.1: Propagations on the 4×2 grid graph, along different paths. The rotation angles $\theta_{i,j}$'s associated to each edge are represented as circular rotating arrows (only drawn along the paths we consider). Propagations always start from the large blue node, and propagated angles are represented as straight arrows (top left of each node). Left 0.1a: propagation along an arbitrary path. Center 0.1b: exact synchronization achieved by propagation along a spanning tree, in the absence of noise. Right 0.1c: synchronization impossible due to noise and incoherence along cycles.

Our contributions. We propose novel RandNLA estimators for the smoothing and angular synchronization problems. We rely on propagations along branches of *Multi-Type Spanning Forests* (MTSF): spanning subsets of both edges and nodes of a graph, whose connected components are either rooted trees or unicycles (connected subsets of edges containing *exactly one* cycle). Our main theorem concerns our estimator for the graph Tikhonov smoothing problem, and can be roughly described as follows (see Theorem 2.4 for a formal statement).

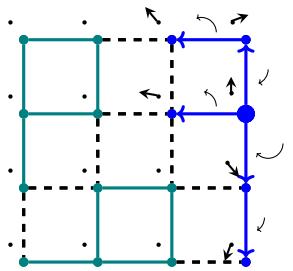
Theorem 0.1. *For MTSFs sampled from the probability distribution in Equation* (2.4)*, propagating the values from the roots of the trees to the other nodes yields an unbiased estimation of the solution of the connection-aware Tikhonov smoothing problem of Equation* (0.1)*.*

See Figure 0.2 for an illustration. We include a teaser runtime comparison with a standard deterministic solver in Figure 0.3, which shows that our estimators are *less sensitive* to the density of the graph, and can provide significant speed-ups for equivalent precision.
In practice, we obtain a *fast* and *scalable* algorithm. MTSF-sampling is achieved via a Wilson-like random-walk-based algorithm with runtime *linear* in the number of edges of the graph. From the theoretical side, our arguments rely on the theory of *Determinantal Point Processes* (DPPs) [29, 44], and generalize the combinatorial analyses of [35, 51].

A preliminary version of this work already appeared in [31], where we presented our smoothing estimator along with two variance-reduction techniques, and an application to a ranking problem for a simple synthetic data model. We improve and extend this work in a



(a) A MTSF in purple, with two unicycles (top left, bottom) and one tree (right). The large purple node is the root of the tree.



(b) Estimation: we propagate the values from the root (blue circle) to its tree via the blue arrows. The estimation is 0 on the unicycles (represented by arrowless dots at the top left of the nodes).

Figure 0.2: Theorem 0.1 illustrated on the 4×4 grid graph. Left: a MTSF. Right: estimation by propagations along the branches of the MTSF. The estimation in $\mathbf{C}$ is represented using angled arrows (top left of the nodes).

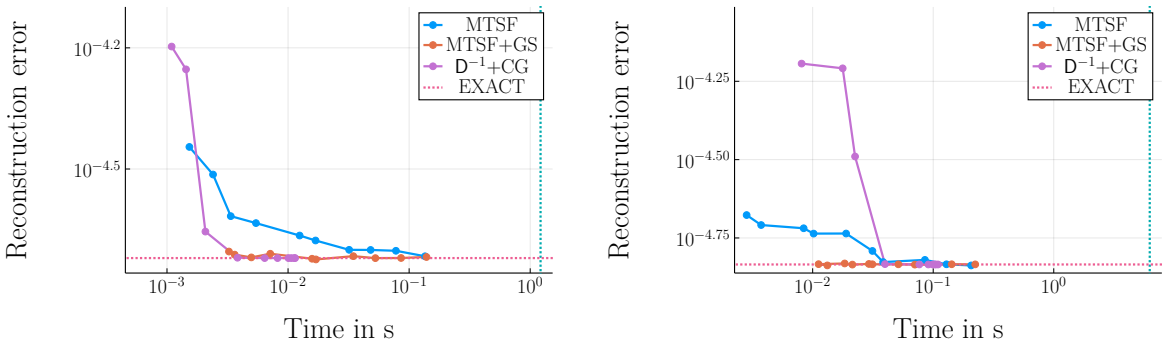
number of different manners, some of which we list below.

Theoretical results. In addition to Theorem 2.4, we derive a connection-aware Feynman-Kac formula (Proposition 2.2), resulting in a local (node-wise) random-walk based estimator for the smoothing problem, similar to walk-on-spheres algorithms [49, 58]. In comparison, our MTSF-based estimators allow for *global* estimation on all the nodes at once. We further relate these two estimators, which may pave the way to generalizations (see Section 13 of the Supplementary Material). We also improve one of our variance-reduction techniques to better handle heterogeneous degree distributions, with significant improvements.

Methodological and experimental contributions. We compare our estimators with standard Krylov-subspaces methods on synthetic data, for both the angular synchronization and smoothing problems. For the angular synchronization problem, we leverage our smoothing estimator as an iterative step in existing approaches [10, 63, 71]. We analyze and contrast the behavior of these methods on different graph topologies which, up to our knowledge, has never been studied in the literature before. Our results show that MTSF-based estimations can outperform standard deterministic methods whenever the graph is not *very sparse* (the gain getting bigger as the density increases). See Figure 0.3. The code used for these experiments is publically available[1].

Related Work. Our main result can be understood as a specialization of the approach in [19,20], based on DPPs, a class of distributions exhibiting negatively-correlated samples [29, 38], and resulting in unbiased estimators for least-squares problems. In contrast to this general work, the specific structure of MTSFs allows efficient sampling and practical algorithms.

[1] https://gricad-gitlab.univ-grenoble-alpes.fr/gaia/synchromtsf



(a) DC-SBM with mean degree $\overline{d} \simeq 30$.

(b) DC-SBM with mean degree $\overline{d} \simeq 420$.

Figure 0.3: Runtime-precision comparisons for the graph Tikhonov smoothing problem, comparing two MTSF-based estimators (MTSF and MTSF+GS) against a diagonally-preconditioned conjugate-gradient-descent (D^{-1}+CG) on degree-corrected stochastic block model graphs (DC-SBM, see Section 3 for a definition) with 10000 nodes. Results are averaged over 10 graphs generated from DC-SBM models with two different parametrizations, resulting in different average degrees $\overline{d}$: on the left, a DC-SBM with average degree $\overline{d} \simeq 30$ and, on the right, a DC-SBM with average degree $\overline{d} \simeq 420$. x-axis: average runtime. y-axis: average reconstruction error. Each data-point corresponds to a number $m \in \{1, 2, 3, 5, 8, 13, 22, 36, 60, 100\}$ of MTSFs (or of CG iterations) used in the estimation. The vertical (resp. horizontal) line is the computation-time (resp. error) of an exact Cholesky solver (see Section 3 for details).

There exists similar applications of random spanning forests distributions in RandNLA for problems on connection-free graphs [3, 4], such as the trace estimation of (regularized) inverse Laplacians [6] or, the Tikhonov regularization and interpolation of graphs signals [51]. Multi-Type Spanning Forests have also been used to build spectral sparsifiers for the (regularized) connection Laplacian in [22], resulting in randomized preconditioners. Our Feynman-Kac formula is inspired from a similar result in [34] for continuous-time random walks, but is specially tailored to the graph Tikhonov smoothing problem; discrete-time walks also exhibit intriguing links with propagations on MTSFs (see Section 13 of the Supplementary Material).

Organisation of the Paper. In Section 1, we introduce preliminary background on graphs and connections used in the rest of the paper. In Section 2, we present our Feynman-Kac formula (Section 2.1) and our MTSF-based estimators (Section 2.2), analyze an efficient sampling algorithm for MTSFs (Section 2.3), and describe our variance-reduction techniques (Section 2.4). We analyze the numerical behavior of our estimators in Section 3, and compare them with standard deterministic algorithms (Sections 3.1 and 3.2). In Section 4, we propose an iterative scheme leveraging our smoothing estimators to solve the angular synchronization problem (Section 4.1), illustrate its application to a denoising problem inspired from *cryogenic electron microscopy* (Section 4.2), and compare this scheme with standard deterministic methods (Section 4.3). Proofs are deferred to the Supplementary Material.

1. Background. A graph $\mathcal{G}$ is defined as a set on nodes $\mathcal{V}$ interconnected by a set of edges $\mathcal{E}$. The edges of $\mathcal{E}$ are *non-oriented* edges. However, for the purpose of this paper, we will also need to consider their *oriented* counterparts, denoted by $\overrightarrow{\mathcal{E}}$. The size of $\overrightarrow{\mathcal{E}}$ is twice the size of $\mathcal{E}$: each edge $e \in \mathcal{E}$ is associated to two oriented edges in $\overrightarrow{\mathcal{E}}$. Choosing arbitrarily an orientation for e and writing s_e and t_e its *source* and *target*, we have $e = (s_e, t_e) \in \overrightarrow{\mathcal{E}}$, and its reversely-oriented edge $e^* = (t_e, s_e) \in \overrightarrow{\mathcal{E}}$. We state our results for *unweighted* graphs, the weighted case is included in the Supplementary Material.

1.1. Combinatorial Laplacian. An elementary instance of graph-supported data is that of real values attached to the nodes of the graph, usually formalized as a vector $f \in \mathbf{R}^{\mathcal{V}}$, and called a *graph signal* [62]. The regularity of such a signal can be quantified using the *combinatorial Laplacian* $\mathsf{L} : \mathbf{R}^{\mathcal{V}} \to \mathbf{R}^{\mathcal{V}}$, a symmetric semi-definite positive operator, conveniently expressed as $\mathsf{L} = \mathsf{D} - \mathsf{A}$, where D is the diagonal degree matrix ($\mathsf{D}_{i,i} = d_i$ the degree of node i) and A the adjacency matrix of the graph (with $\mathsf{A}_{(i,j)} = 1$ if $(i,j) \in \overrightarrow{\mathcal{E}}$ and 0 otherwise) [15]. The quadratic form associated to L acts as:

$$\langle f, \mathsf{L}f \rangle = \frac{1}{2} \sum_{e \in \overrightarrow{\mathcal{E}}} |f(t_e) - f(s_e)|^2, \tag{1.1}$$

which associates a high value to functions with important local variations over the edges of the graph, and serves as a basis for the notion of *frequency* for graph signals [62, 67]. These operators are also related to combinatorial properties of the graph $\mathcal{G}$ [8], for instance:

(P1) $\ker \mathsf{L}$ is always non-empty, with dimension the number of connected components of $\mathcal{G}$. This observation forms one of the bases for spectral clustering on graphs [68].

(P2) $\mathsf{D}^{-1}\mathsf{A}$ is the matrix of the natural *random walk* on $\mathcal{G}$, which transitions from a node to one of its neighbors with uniform probability at each step, with $(\mathsf{D}^{-1}\mathsf{A})^l_{i,j}$ the probability that a path of length l starting at i ends up in node j.

1.2. Connection Laplacian: Definition and Basic Properties. *Unitary connections* are a practical way to add additional information to the graph's structure[2]. We describe in the following how we can capture additional rotations associated to edges in the form of a Laplacian-like operator. The main idea is to represent rotations as *unitary complex numbers*, which will naturally lead to consider *complex-valued* functions $f \in \mathbf{C}^{\mathcal{V}}$ when studying variational properties on $\mathcal{G}$. The entries of f should be understood as belonging to different copies $\mathbf{C}_v$ of the complex plane $\mathbf{C}$ associated to each node $v \in \mathcal{V}$, see Figure 1.1[3].

We will associate to each directed edge in $\overrightarrow{\mathcal{E}}$ a map representing the transformation along that edge, which is known as a *connection* [35]. A *unitary connection* Ψ on a graph $\mathcal{G}$ is a collection of unitary linear maps $(\psi_e)_{e \in \overrightarrow{\mathcal{E}}}$, with each map $\psi_e : \mathbf{C}_{s_e} \to \mathbf{C}_{t_e}$ acting as multiplication by a unitary complex number $e^{\iota\theta_e}$, so that $\psi_e(z) = e^{\iota\theta_e} \cdot z$, with ι the complex imaginary unit (we sometimes abuse notation and write $\psi_e = e^{\iota\theta_e}$ when there is no risk of confusion). In addition, we require that $\psi_{e^*} = \psi_e^*$ (*i.e.* $\psi_{(t_e,s_e)} = \psi^*_{(s_e,t_e)}$), so that the

[2] By describing explicit geometrical transformations between the signal values of the nodes of $\mathcal{G}$.

[3] This association of a copy of $\mathbf{C}$ is analogous to a fiber bundle over a manifold (*e.g.* its tangent bundle), and is known as a *complex line bundle* over $\mathcal{G}$.

transformation associated to an edge traversed in one direction or the other differs only by conjugation. This notion is illustrated in Figure 1.1.

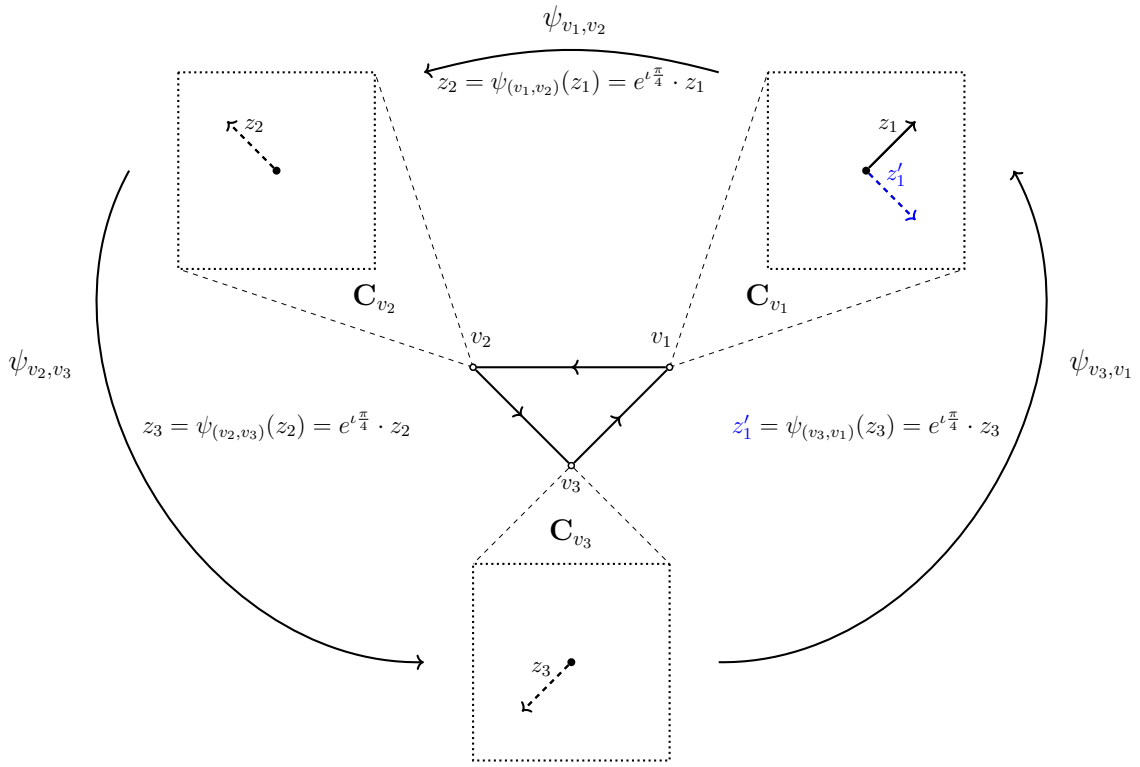


Figure 1.1: A connection Ψ on the triangle graph K_3 (in the center). The connections maps ψ_e are associated with angles $\theta_{(v_1,v_2)} = \theta_{(v_2,v_3)} = \theta_{(v_3,v_1)} = \frac{\pi}{4}$, and we represent their action on some $z_1 \in \mathbf{C}_{v_1}$. A cyclic path $C = ((v_1, v_2), (v_2, v_3), (v_3, v_1))$ is depicted using directed arrows, we denote by $\psi_C = \psi_{(v_3,v_1)} \circ \psi_{(v_2,v_3)} \circ \psi_{(v_1,v_2)}$ the composition of the connection maps along this cycle (a rotation of $\theta_C = \frac{3\pi}{4}$). Top right: $z_1 \in \mathbf{C}_{v_1}$ (bold arrow) and $z_1' = \psi_C(z_1) \in \mathbf{C}_{v_1}$ (blue dotted arrow). Top left, bottom: $z_2 = \psi_{(v_1,v_2)}(z_2) \in \mathbf{C}_{v_2}$ and $z_3 = \psi_{(v_2,v_3)}(z_2) \in \mathbf{C}_{v_3}$ (dotted arrows).

Remark 1.1. *In applications, such as angular synchronization, the θ_e's often represent* a priori *known rotations, and are frequently modulated by a scale parameter $\gamma \in \mathbf{R}_+$, resulting in connections such that $\psi_e(z) = e^{\iota\gamma\theta_e} z$. While we will forego this additional parameter in our theoretical results, it is easily recovered by replacing each θ_e with $\gamma\theta_e$.*

The connection Laplacian. To a graph endowed with a unitary connection, we associate the *connection Laplacian* $\mathsf{L}_\theta : \mathbf{C}^{\mathcal{V}} \to \mathbf{C}^{\mathcal{V}}$, defined by $\mathsf{L}_\theta = \mathsf{D} - \mathsf{A}_\theta$, with A_θ a connection-aware adjacency matrix such that $(\mathsf{A}_\theta)_{i,j} = e^{-\iota\theta_{(i,j)}}$ if $\{i, j\} \in \mathcal{E}$, and $\mathsf{A}_{i,j} = 0$ otherwise. This operator generalizes the Laplacian to non-trivial connections, and is associated to the

quadratic form:

$$\langle f, \mathsf{L}_\theta f \rangle = \frac{1}{2} \sum_{e \in \overrightarrow{\mathcal{E}}} |f(t_e) - e^{\iota\theta_e} f(s_e)|^2. \tag{1.2}$$

For the trivial connection, with $\psi_e = \mathrm{id}_{\mathbf{C}_{s_e}, \mathbf{C}_{t_e}}$ (*i.e.* $\theta_e = 0$) for all edges $e \in \overrightarrow{\mathcal{E}}$, we recover $\mathsf{L}_\theta = \mathsf{L}$ and the expression in Equation (1.1). Unlike this specific case, $\langle f, \mathsf{L}_\theta f \rangle$ also penalizes functions that are incoherent with respect to the connection, including constant functions.

Let us now answer a natural question: when is the kernel $\ker \mathsf{L}_\theta$ non-empty, and what are the functions $f \in \mathbf{C}^{\mathcal{V}}$ that are not penalized by $\langle f, \mathsf{L}_\theta f \rangle$?

(P1′) Angular synchronization and $\ker L_\theta$. When the entries f_i of the vector f are *unitary* complex numbers $f_i = e^{\iota s_i}$, we recover from Equation (1.2) the expression in Equation (0.3). In this case, $|f(j) - e^{\iota\theta_{(i,j)}} f(i)|^2 = 2 - 2\mathfrak{Re}((e^{\iota s_j})^* e^{\iota\theta_{(i,j)}} e^{\iota s_i})$, which, thanks to conjugation-invariance of the real part, results in:

$$\langle f, \mathsf{L}_\theta f \rangle = \sum_{\{i,j\} \in E} 2 - 2\cos((s_j - s_i) - \theta_{(i,j)}). \tag{1.3}$$

Even though this equality only holds for $f \in U(\mathbf{C})^{\mathcal{V}}$, note that $\langle f, \mathsf{L}_\theta f \rangle = 0$ if and only if $f_i = e^{\iota\theta_{(j,i)}} f_j$ for all $(i,j) \in \overrightarrow{\mathcal{E}}$, that is, if $s_i = s_j + \theta_{(j,i)}$. As a consequence, L_θ is not only semi-definite positive (as seen from Equation (1.2)), but *positive* definite, *unless* there is an *exact solution* $x \in U(\mathbf{C})^{\mathcal{V}}$ to the associated angular synchronization problem, in which case the kernel $\ker \mathsf{L}_\theta$ is generated by x.

This behavior of the smallest eigenpair of L_θ is in clear contrast to that of L (which is *always* zero, see property (P1)), and we explain in Section 2 how a generalization of property (P2) pertaining to *random propagations* can be used to solve connection-Laplacian-based problems.

2. Random Estimators for Tikhonov Smoothing under a Connection. The goal of the next Section is to smooth a complex signal on a graph, by way of random propagations. More precisely, we fix a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ endowed with a connection Ψ, and we aim at smoothing a signal $g \in \mathbf{C}^{\mathcal{V}}$ by solving:

$$\operatorname*{argmin}_{f \in \mathbf{C}^{\mathcal{V}}} \; q\|f - g\|_2^2 + \langle f, \mathsf{L}_\theta f \rangle, \tag{2.1}$$

with $q \in \mathbf{R}_+^*$. This is nothing but a re-writing of Equation (0.1), and $\langle f, \mathsf{L}_\theta f \rangle$ penalizes functions that are not coherent with respect to the connection Ψ. Denoting by f_* the optimal solution to this problem, we have.

Proposition 2.1. *The solution to Problem* (2.1) *can be expressed as:*

$$f_* = q(\mathsf{L}_\theta + q\mathsf{I})^{-1} g. \tag{2.2}$$

The proof of Proposition 2.1, based on straightforward **CR**-calculus, is detailed in Section 8 of the Supplementary Material.

Let us now describe how to recover the solution of this problem via local propagation over random paths on $\mathcal{G}$.

2.1. Local Estimation: a Feynman-Kac Formula. Feynman-Kac formulas express solutions of variational problems as the expectations of stochastic processes: we develop here an instance tailored to the Tikhonov smoothing problem of Equation (2.1). We start with a few definitions. A *path* p in $\mathcal{G}$ is an ordered sequence of oriented edges in $\overrightarrow{\mathcal{E}}$. We consider the two operations of concatenation pq of two paths p and q and of orientation-reversal p^* of a path p, and denote by P_i^j the set of paths from i to j in $\mathcal{G}$. Connection maps extend to paths inductively as $\psi_{pq} = \psi_q \circ \psi_p$, and we will for instance frequently encounter the map $\psi_{p^*} : \mathbf{C}_j \to \mathbf{C}_i$, acting by composition of the (inverse) rotations encountered along path $p \in P_i^j$ from i to j.

The stochastic process we consider is a random walk with a non-zero probability of being interrupted at any node. It is conveniently defined on an extended graph $\mathcal{G}^\Gamma$, with nodes $\mathcal{V} \cup \{\Gamma\}$ and edges $\mathcal{E} \cup \bigcup_{v\in\mathcal{V}}\{\{v,\Gamma\}\}$ (so that all the nodes of $\mathcal{V}$ are connected to Γ), where Γ is an additional *boundary* node. We then define the random walk $(u_t)_{t\geq 0}$, which starts at some fixed node $u_0 = i \in V$ and transitions to some other node at time t as follows:

- $u_{t+1} = \Gamma$ with probability $\frac{q}{d_{u_t}+q}$,
- $u_{t+1} = v \in \mathcal{V}$ with probability $\frac{\mathsf{A}_{u_t,v}}{d_{u_t}+q}$.

The process ends upon reaching node Γ. In other words, this is the natural random walk on $\mathcal{G}_\Gamma$, with a stopping criterion at the boundary Γ. Writing $u_l = \Gamma$ for some time l, this process results in a random path $((u_0,u_1),...,(u_{l-1},u_l))$. We denote by P_i^Γ the set of all possible paths obtainable via this process, and by ν_i the resulting probability measure on P_i^Γ.
We can then show that propagating along paths sampled according to ν_i converges in expectation to the solution of Problem (2.1) on node i. More precisely:

Proposition 2.2. *Denoting by $j = u_{l-1}$ the last node reached before p reaches Γ, we have:*

$$f_*(i) = \mathbf{E}_{p\sim\nu_i}\left(\psi_{p_\Gamma^*}(g_j)\right), \tag{2.3}$$

where $p = p_\Gamma e_\Gamma$ with e_Γ the last edge in p.

In other words, an unbiased estimate of $f_*(i)$ is obtained by drawing a random path from i to Γ, and retropropagating the value of g at the node just before Γ (j in Proposition 2.2), taking into account the rotations along the path. See Algorithm 2.1. This formula is a discrete-time analog of a Theorem in [34], and we prove a more general version in the Supplementary Material (Section 8, Proposition 8.1), building upon a connection-aware version of property (P2).

Algorithm 2.1 Feynman-Kac estimator (Proposition 2.2).

$f_i \leftarrow 0$
Repeat m times
  Sample a path $p \in P_i^\Gamma$ from ν_i ▷ By running an interrupted random walk on $\mathcal{G}$
  $f_i \leftarrow f_i + \psi_{p_\Gamma^*}(g_j)$ ▷ j the last node before interruption of p
Output $\frac{1}{m} f_i$

Algorithm 2.1 is simple but may be computationally expensive: estimation on one node requires sampling m paths of *unbounded length*, which needs to be repeated *for each* of the $|\mathcal{V}|$ nodes in $\mathcal{G}$ (a locality issue inherent to Walk-on-Spheres-type algorithms [49]). We address these limitations in the remainder of this Section and propose novel *propagation-based*

estimators of f_*, allowing to update the Monte-Carlo estimates on all the nodes at once, by sampling specific substructures of the graph, which is achieved with an expected number of steps of a random walk *linear* in the number of edges $|\mathcal{E}|$.

Remark 2.3. *The restriction to complex signals on $\mathcal{G}$ and unitary connections (encoding 2D rotations) is mostly artificial, and Proposition 2.2 generalizes to transformations in* e.g. $O(\mathbf{R}^d)$. *This setting is encountered for instance for 3D molecule alignement in cryogenic electron microscopy (cryo-EM) [64], or in structure-from-motion problems [2], where the ψ_e's act as 3D rotations (*i.e. $\psi_e \in SO(\mathbf{R}^3)$*).*

2.2. Multi-Type Spanning Forests: a first global estimator. To build our global estimators, we rely on decompositions of the graph into *Multi-Type Spanning Forests* (MTSFs) [22]. A MTSF $\phi \subseteq \mathcal{V} \cup \mathcal{E}$ decomposes $\mathcal{G}$ into disjoint components that are either *rooted trees*, or *unicycles*. More precisely, a MTSF ϕ must have fixed cardinality $|\phi| = |\mathcal{V}|$, and is divided into (maximal) components $c_\phi \subseteq \phi$, which must be of one amongst two types:

- A rooted tree $c_\phi \subseteq \mathcal{V} \cup \mathcal{E}$, such that $c_\phi \cap \mathcal{E}$ is a connected cycle-free subset of edges, and $c_\phi \cap \mathcal{V}$ is reduced to a single node, connected to $c_\phi \cap \mathcal{E}$.
- A unicycle $c_\phi \subseteq \mathcal{E}$, which is a connected subset of edges containing *exactly* one cycle.

This structure is illustrated in Figure 2.1. We denote by $\mathcal{M}(\mathcal{G})$ the set of all MTSFs over the graph $\mathcal{G}$.

Note that in the absence of unicycles, a MTSF is a *rooted spanning forest* [3, 4], whereas a tree-free MTSF is a *spanning forest of unicycles* [35].

We will consider the distribution $\mathcal{D}_\mathcal{M}$ over $\mathcal{M}(\mathcal{G})$ defined by:

$$\mathbf{P}_{\mathcal{D}_\mathcal{M}}(\phi) \propto q^{|\phi \cap \mathcal{V}|} \prod_{C \in \mathcal{C}(\phi)} (2 - 2\cos(\theta_C)), \tag{2.4}$$

where $\mathcal{C}(\phi)$ is the set of cycles belonging to the unicycles of ϕ, and θ_C is the argument of the unitary complex number associated connection map ψ_C obtained from a path traversing C *one time* (as in Figure 1.1). Note that, while ψ_C depends on the orientation of this path, $\cos(\theta_C)$ is insensitive to this orientation. Moreover, a cycle has a non-zero probability of being sampled if and only if it is *inconsistent* (if C is perfectly consistent then $\theta_C = 0$ and $2 - 2\cos(\theta_C) = 0$).

Distribution $\mathcal{D}_\mathcal{M}$ is a rooted variant of the distribution considered in [22], and is in fact a DPP[4]. We can now state our main theorem.

Theorem 2.4. *Let ϕ be a MTSF sampled according to $\mathcal{D}_\mathcal{M}$. Denote by $a \xrightarrow{\phi} b$ the unique path from a to b in some tree of a MTSF ϕ and,* if *$v \in \mathcal{V}$ belongs to a rooted tree of ϕ, by $r_\phi(v)$ the root of this tree. Consider the estimator*

$$\tilde{f}(i, \phi, g) = \psi_{r_\phi(i) \xrightarrow{\phi} i}(g(r_\phi(i))) \tag{2.5}$$

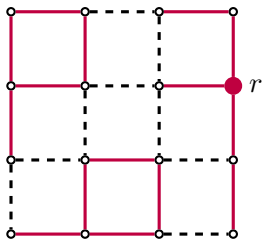
propagating the value from $r_\phi(i)$ to i if *$i \in \mathcal{V}$* belongs to a rooted tree, *and $\tilde{f}(i, \phi, g) = 0$* if *$i$* lies in a unicycle. *Then, we have:*

$$f_*(i) = \mathbf{E}_{\mathcal{D}_\mathcal{M}}(\tilde{f}(i, \phi, g)). \tag{2.6}$$

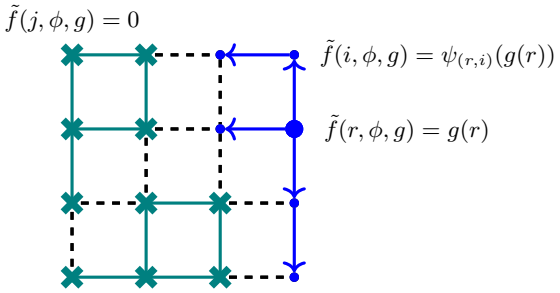
[4]So is the distribution of [22], but their non-combinatorial argument does not carry over to our rooted process. See the Section 9 of the Supplementary Material for a proof in this case

The proof of Theorem 2.4 relies on the reformulation of $\mathcal{D}_{\mathcal{M}}$ as a DPP, and an extension of arguments coming from both [35] and [51]. It is a special case of Theorem 9.1, proved in Section 9 of the Supplementary Material.

In a less technical phrasing, propagating for each tree in a MTSF ϕ the value of g at its root r to its other nodes i (along path $r \xrightarrow{\phi} i$) results in an unbiased estimator of f_*. The estimation on nodes belonging to unicycles is then 0. See Algorithm 2.2 and Figure 2.1.



(a) A MTSF ϕ in purple, with two unicycles (top left, bottom) and one tree (right) with its root r (large purple node).



(b) Estimation: we set $\tilde{f}(i,\phi,g) = \psi_{r \xrightarrow{\phi} i}(g_r)$ for nodes in the tree (blue arrows), and $\tilde{f}(j,\phi,g)=0$ on unicycles (teal crosses).

Figure 2.1: Theorem 2.4 illustrated on the 4×4 grid graph, on one MTSF.

Algorithm 2.2 MTSF-based estimator (Theorem 2.4).

$f_i \leftarrow 0 \ \forall i \in \mathcal{V}$
Repeat m times
  Sample $\phi \in \mathcal{M}(\mathcal{G})$ from $\mathcal{D}_{\mathcal{M}}$ ▷ Algorithm 2.3 can be used
  for $i \in \mathcal{V}$ **do**
    if i belongs to a rooted tree of ϕ **then**
      $f_i \leftarrow f_i + \psi_{r_\phi(i) \xrightarrow{\phi} i}(g(r_\phi(i)))$
    end if
  end for
Output $\frac{1}{m}(f_1, ..., f_{|\mathcal{V}|})$

Remark 2.5. *If the connection Ψ is trivial* (i.e., $\psi_e = \mathrm{id}_{\mathbf{C}_{s_e},\mathbf{C}_{t_e}}$ *for all edges), we recover the result from [51]. There are no inconsistent cycles in this case, and $\mathcal{D}_{\mathcal{M}}$ reduces to another determinantal distribution $\mathcal{D}_{\mathcal{F}}$ over* rooted spanning forests $\phi \in \mathcal{V} \cup \mathcal{E}$ *[3, 4].*

Unlike Algorithm 2.1, Algorithm 2.2 allows to update the estimates on *all the nodes* for

each sampled MTSF, in $\mathcal{O}(|\mathcal{V}|)$ time. On the other hand, it becomes necessary to sample a MTSF ϕ according to $\mathcal{D}_{\mathcal{M}}$. A first naive strategy is to use sampling algorithms designed for general DPPs, relying on the eigendecomposition of the DPP's kernel matrix (a Hermitian matrix of size $(|V|+|E|)\times(|V|+|E|)$), which is much too expensive. In the next section, we recall the much more efficient random-walk-based algorithm proposed in [22], and analyze its expected computational cost.

2.3. Sampling Multi-Type Spanning Forests. Efficient sampling from $\mathcal{D}_{\mathcal{M}}$ is possible thanks to *loop-erased random walks*, traditionally used in Wilson's algorithm for uniformly sampling spanning trees of a graph [69], under the following additional condition[5].

Condition 2.6 (Weak-Inconsistency). *For all cycles C in $\mathcal{G}$, $\cos(\theta_C)\geq 0$.*

We stress that this condition must hold for *all* the cycles of the graph, and not just for the cycles $\mathcal{C}(\phi)$ in some MTSF $\phi\in\mathcal{M}(G)$. When Condition 2.6 is satisfied, $1-\cos(\theta_C)\leq 1$ defines a probability measure over the oriented cycles in G, which can be leveraged to efficiently sample MTSF ϕ, as we will see in the following.

Remark 2.7. *Even though efficient sampling from $\mathcal{D}_{\mathcal{M}}$ can only be performed under Condition 2.6, Theorem 2.4 holds beyond this technical condition. Furthermore, in some applicative settings (*e.g. *directed graph signal processing [24], ranking [16, 71] or perceived luminance reconstruction [70]), this condition can be forced by tweaking the scale parameter γ.*

MTSF sampling algorithm. The algorithm of [22], recalled here as Algorithm 2.3, samples a MTSF ϕ according to $\mathcal{D}_{\mathcal{M}}$ by simulating multiple random walks on $\mathcal{G}^{\Gamma}$, and constructs ϕ iteratively from these random paths. The precise way in which a path is turned into a part of ϕ relies on a *loop-erasure* procedure. The full sampling scheme is detailed in Algorithm 2.3, performing random walks that can be stopped in a number of ways: by being interrupted (*i.e.* reaching Γ), by building a cycle θ_C (kept with probability $1-\cos(\theta_C)$), or by reaching a node already spanned by ϕ[6]. We suppose that the nodes of $\mathcal{G}$ are arbitrarily ordered in a queue, and say that a node i is spanned by ϕ if $i\in\phi$ or if i is an endpoint of some edge $e\in\phi$. We also use a function random_successor(u) that, at each call, randomly outputs either Γ with probability $\frac{q}{d_u+q}$, or some node v with probability $\frac{\mathsf{A}_{u,v}}{d_u+q}$.

One then proves the following by applying the arguments of [22] (which describes how to sample *unrooted* MTSFs), and keeping track of the roots.

Proposition 2.8. *Suppose that Condition 2.6 holds. Then, Algorithm 2.3 outputs a MTSF ϕ distributed according to $\mathcal{D}_{\mathcal{M}}$.*

Let us now discuss the cost of Algorithm 2.3.

[5] Condition 2.6 has already been identified as a technical sampling condition for Wilson-like algorithms (in [33] and [22]). It also appears in a different guise in [34], under the name of *trace-positivity*, where it allows to simplify some technical statements.

[6] Some implementation details of Algorithm 2.3 are not made completely explicit here, such as the precise way in which we detect cycles (instruction at line 11), or how to track the cycle-acceptance probabilities (*i.e.* θ_C). Please refer to Section 10 of the Supplementary Material and our Julia implementation for more details

Algorithm 2.3 MTSF sampling algorithm [22].

$\phi \leftarrow \emptyset$
while ϕ not spanning **do**
  Let $i \in \mathcal{V}$ be the first node in the queue not spanned by ϕ
  $u \leftarrow i$ ▷ u is the current node of the random walk
  $p \leftarrow \epsilon$ ▷ ϵ the empty path
  while ($u \neq \Gamma$) and (p does not intersect ϕ or contain a cycle) **do**
    $u' \leftarrow$ random_successor(u) ▷ Move to next node
    if $u' \neq \Gamma$ **then**
      $e \leftarrow (u, u')$, $p \leftarrow pe$ ▷ Add e to the path p
    end if
    if p contains a cycle C **then**
      Remove this cycle from p with probability $\cos(\theta_C)$
    end if
    $u_{\text{old}} \leftarrow u$, $u \leftarrow u'$ ▷ u_{old} the previous node
  end while
  if $u = \Gamma$ **then**
    $\phi \leftarrow \phi \cup p \cup u_{\text{old}}$ ▷ Add the sampled path p and the root u_{old} to ϕ[7]
  else
    $\phi \leftarrow \phi \cup p$
  end if
end while
Output ϕ

Proposition 2.9. *The expected number of random walk steps of Algorithm 2.3 is bounded from above by*

$$\operatorname{tr}\left((\mathsf{L} + q\mathsf{I})^{-1}(\mathsf{D} + q\mathsf{I})\right). \tag{2.7}$$

Furthermore, Algorithm 2.3 can be implemented with $\mathcal{O}\left(\frac{|\mathcal{E}|}{q}\right)$ expected time complexity.

See Section 10 for the proof. The bound is obtained by noting that Algorithm 2.3 necessitates *fewer* steps than the Wilson-like algorithm in [51], used to sample random spanning forests [3,4], with expected number of steps given by (2.7). Unlike the algorithm in [51] though, Algorithm 2.3 requires both tracking of the angular offsets accumulated along a cycle, and detecting said cycle, at additional computational cost. We discuss two possible implementations in Section 10 of the Supplementary Material, one of them resulting in the $\mathcal{O}\left(\frac{|\mathcal{E}|}{q}\right)$ expected *time complexity* mentioned in Proposition 2.9. This translates to $\mathcal{O}\left(\frac{|\mathcal{E}|}{q} + |\mathcal{V}|\right)$ expected runtime for Algorithm 2.2.

Remark 2.10. *In case Condition 2.6 is not satisfied, one can still estimate f_** up to a multiplicative constant *(af_* for some a), using the importance sampling strategy from [22].*

[7] We abuse notations and denote here by p the set of *non-oriented* edges in the path.

Specifically, one can threshold the incoherence $2 - 2\cos(\theta_C)$ *and sample from*

$$\mathbf{P}_{IS}(\phi) \propto q^{|\phi \cap \mathcal{V}|} \prod_{C \in \mathcal{C}(\phi)} \min(2, 2 - 2\cos(\theta_C)) \tag{2.8}$$

using a straightforward variant of Algorithm 2.3. The estimation then uses the importance weights

$$w(\phi) = \prod_{C \in \mathcal{C}(\phi)} \max(1, 1 - \cos(\theta_C)). \tag{2.9}$$

Both the Feynman-Kac-based Algorithm 2.1 and the MTSF-based Algorithm 2.2 can be roughly understood as performing a random walk on the graph before stopping at some root node, and then propagating the value from this root to the starting point of the random walk. One main difference is that MTSFs allow to update the estimated values on all nodes jointly.

2.4. Variance Reduction for the MTSF-based estimator. A paramount property of Monte-Carlo estimators is not only their unbiased behavior in expectation, but also their variance: they are significantly improved when used in conjunction with efficient variance reduction techniques. We propose two such improvements over $\tilde{f}$, based on the classical approaches of *Rao-Blackwellization* [7, 55] and *control variates* [36] respectively, generalizing the variance reduction techniques introduced in [51, 52].

Rao-Blackwellization. Rao-Blackwellization leverages the two laws of total expectation and variance, roughly stating that conditioning an estimator using another statistic extracted from the same sample still results in an unbiased estimator, with lower variance (see Section 11 of the Supplementary Material for more details in our case). Here, our technique consists in conditioning on the set of edges in ϕ. This yields the following estimator:

$$\overline{f}(i, \phi, g) = \psi_{r_\phi(i) \xrightarrow{\phi} i}(h_\phi(r_\phi(i), g)) \text{ if } i \text{ belongs to a rooted tree,} \tag{2.10}$$

where h_ϕ is defined as

$$h_\phi(r, g) = \frac{\sum_{j \in c_\phi(r)} \psi_{j \xrightarrow{\phi} r} g(j)}{|c_\phi(r)|}, \tag{2.11}$$

and $c_\phi(r)$ is the *set of nodes* spanned by the tree containing r. If i belongs to a unicycle, we once again set $\overline{f}(i, \phi, g) = 0$.
This results in Algorithm 2.4, which amounts to:

- Computing at the root r of each tree the average $h_\phi(r, g)$ of the values of g over the tree (obtained by propagating from each node i in the tree to r).
- Propagating this average back to the other nodes of the tree.

As compared with Algorithm 2.2, this procedure can be implemented at little additional cost[8].

[8] Note also the similarity to the Belief Propagation algorithm over trees [47].

Algorithm 2.4 Rao-Blackwellized MTSF-based estimator (Theorem 2.4).

$f_i = 0 \ \forall i \in \mathcal{V}$, $h_r = 0 \ \forall r \in \mathcal{V}$
Repeat m times
  Sample $\phi \in \mathcal{M}(\mathcal{G})$ from $\mathcal{D}_{\mathcal{M}}$ ▷ Via Algorithm 2.3
  for $r \in \phi \cap \mathcal{V}$ **do**
    for $j \in c_\phi(r)$ **do**
      $h_r = h_r + \psi_{j \xrightarrow{\phi} r} g(j)$ ▷ Propagate and average at the root
    end for
  end for
  for $i \in \mathcal{V}$ belonging to a rooted tree of ϕ **do**
    $f_i = f_i + \psi_{r_\phi(i) \xrightarrow{\phi} i} \left(\frac{h_{r_\phi(i)}}{|c_\phi(r_\phi(i))|} \right)$ ▷ Propagate back
  end for
Output $\frac{1}{m}(f_1, ..., f_{|\mathcal{V}|})$

Control variates. Second is the introduction of control variates: an addition of another random quantity to $\overline{f}$, designed to have zero mean (so that the expectation remains unchanged), but resulting in an estimator with lower variance when designed properly. We propose to use a single *gradient-descent* step with parametrized step-size α:

$$\hat{f}(\phi, g) = \overline{f}(\phi, g) - \alpha \mathsf{P}(q^{-1}(\mathsf{L}_\theta + q\mathsf{I})\overline{f}(\phi, g) - g), \tag{2.12}$$

where $\mathsf{P} = (q^{-1}\mathsf{D} + \mathsf{I})^{-1}$ is a diagonal preconditioner for the system $q^{-1}(\mathsf{L}_\theta + q\mathsf{I})f = g$, and $\alpha \in \mathbf{R}_+^*$. A good choice of step-size α is crucial in order to obtain a significant reduction of variance. We simply take $\alpha = 1$ in the following (see Section 11 of the Supplementary Material for empirical results backing this choice).

We prove in Section 11 of the Supplementary Material that generalizations of both $\overline{f}$ and $\hat{f}$ are unbiased estimators of f_*.

Proposition 2.11. $\mathbf{E}_{\mathcal{D}_{\mathcal{M}}}(\overline{f}(i, \phi, g)) = \mathbf{E}_{\mathcal{D}_{\mathcal{M}}}(\hat{f}(i, \phi, g)) = f_*(i)$

Both variance-reduction techniques are easy to implement, and do not incur large additional computational costs: The Rao-Blackwellization $\overline{f}$ has $\mathcal{O}(|\mathcal{V}|)$ additional cost, and the gradient-descent step in $\hat{f}$ entails *a single* matrix-vector multiplication, in $\mathcal{O}(|\mathcal{E}|)$ time.

3. Numerical Results under Weak-Inconsistency. We now analyze the behavior of the estimator proposed in Theorem 2.4, along with the improved versions discussed in Section 2.4, and compare their performance with (deterministic) conjugate-gradient-based solvers [56], on graphs with different topologies[9]. We perform experiments on the following connection model.

Connection Model. For a given graph $\mathcal{G}$ with n nodes, we associate to each of its nodes v an angle ω_v chosen uniformly in $[0, 2\pi)$, and we set

$$\theta_e = \omega_{t_e} - \omega_{s_e} + \eta\varepsilon_e \tag{3.1}$$

[9] In this Section and in Section 4, we perform all our measurements using *a single thread* on a laptop with an intel i7-1185G7 processor.

for all edges e, with ε_e a perturbation uniformly distributed in $[-1, 1]$, and $\eta \in \mathbf{R}_+^*$ a scaling constant. We set $\eta = \frac{\pi}{2n}$ to ensure that Condition 2.6 is satisfied (so that Algorithm 2.3 in fact samples a MTSF according to $\mathcal{D}_{\mathcal{M}}$).

3.1. A First Runtime Experiment on Erdös-Rényi Graphs. We first analyze the behavior of our estimator $\tilde{f}$ with respect to two parameters: the choice of regularization parameter q and the mean degree $\overline{d}$ of the graph (controlling the graph's density). In our first experiment, we let $\overline{d}$ take values in $\{50, 100, 150, 200\}$, and q take values equal to $q'\overline{d}$, with $q' \in \{10^{-3}, 10^{-1}, 1\}$. The complexity bound of Equation (2.7) becomes *linear in* $|\mathcal{V}|$ when using such a parametrization.

Setup. We generate Erdös-Rényi random graphs $\mathcal{G} \sim \mathrm{ER}(n, p)$ of size $n = 10000$ for varying $p \in [0, 1]$, so that each edge e independently appears with probability p. To control the density, we set $p = \frac{\overline{d}}{n-1}$ so that the expected mean degree of these random graphs is $\overline{d}$. For each such graph, we generate a random signal $f \in \mathbf{C}^{\mathcal{V}}$ with independent complex Gaussian entries $f_v \sim \mathcal{N}_{\mathbf{C}}(0, 1)$.

We then measure the running time of sampling *one* MTSF ϕ and applying the estimator $\tilde{f}$. As a reference, we also measure in the same manner the runtime of the matrix-vector multiplication $\mathsf{L}_\theta f$, where L_θ is implemented as a sparse Hermitian matrix in CSC format. Note that this operation is the most expensive part of an iteration of the Conjugate-Gradient algorithm, and serves as a simple baseline to compare computation costs.

The results, depicted in Figure 3.1, are the average over 10000 time measurements, themselves averaged over 10 realizations of the graph G.

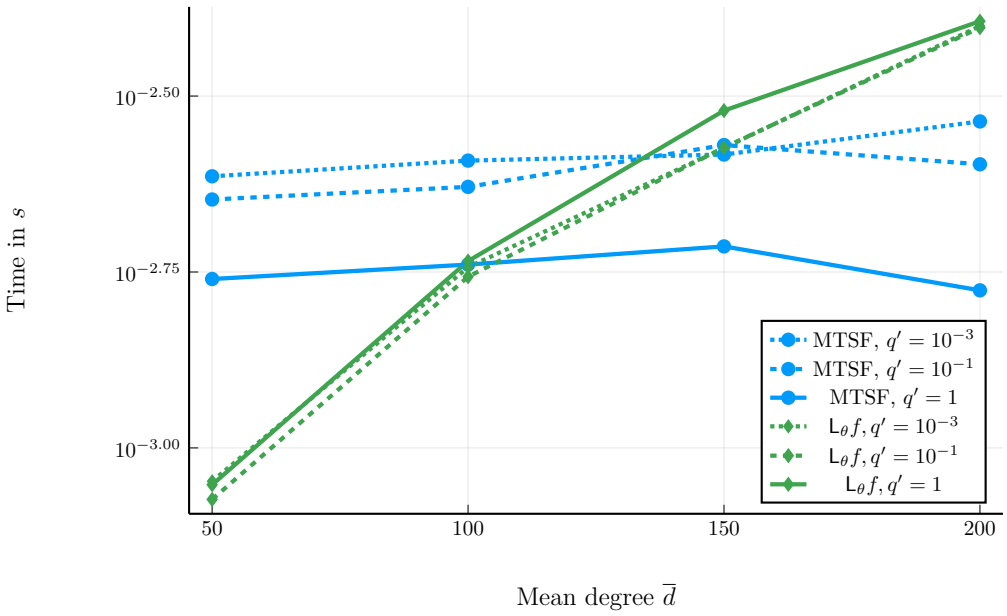


Figure 3.1: Runtime when varying $\overline{d}$, for different values of q.

Analysis. Two trends emerge in Figure 3.1. First, as q' (hence q) increases, computing $\tilde{f}$ becomes less expensive, since the probability of the random walk stopping at some root node increases. Second, in contrast to the matrix-vector product $\mathsf{L}_\theta f$, our estimator is less sensitive to the density of the graph: computing $\tilde{f}$ is more expensive than computing $\mathsf{L}_\theta f$ for $\overline{d} = 50$, and systematically faster for $\overline{d} = 200$. This reflects the $\mathcal{O}(|\mathcal{V}|)$ expected complexity for $\tilde{f}$ (obtained with the parametrization $q = q'\overline{d}$).

Remark 3.1. *Results are similar when varying the size n of the graph (not shown).*

3.2. Runtime-Precision Trade-offs in the context of Complex Graph Signal Denoising. We compare the performance of our improved estimators $\overline{f}$ and $\hat{f}$ with conjugate-gradient methods [56]. The objective consider is to recover a signal $f_\top \in \mathbf{C}^\mathcal{V}$ given a noisy degradation $g = f_\top + \varepsilon$.

In the specific instance where $f_\top \in \mathbf{R}^\mathcal{V}$ and the connection on $\mathcal{G}$ is trivial ($\psi_e = \mathrm{id}_{\mathbf{C}_{s_e},\mathbf{C}_{t_e}}$ for all $e \in \mathcal{E}$), a common assumption in the graph signal processing literature is that $f_\top$ is a *smooth* signal, that is, a linear combination of the first (low-frequency) eigenvectors of L (associated to the smallest eigenvalues) [43, 53].

We here similarly assume that $f_\top \in \mathbf{C}^\mathcal{V}$ is B-bandlimited (*i.e.* $f_\top \in \mathrm{ran}(u_1, ..., u_B)$[10], with u_i the i-th eigenvector of L_θ), so that solving the Tikhonov smoothing Problem (2.1):

$$\operatorname*{argmin}_{f \in \mathbf{C}^\mathcal{V}} \; q\|f - g\|_2^2 + \langle f, \mathsf{L}_\theta f \rangle, \tag{3.2}$$

should allow to faithfully recover $f_\top$ from g, by penalizing high-frequency components in the optimal solution $f_* = q(\mathsf{L}_\theta + q\mathsf{I})^{-1}g$. We refer the reader to Section 12 of the Supplementary Material for additional supporting arguments in this connection-aware setting.

Recall that this matrix inversion can be computed using a Cholesky decomposition (for an exact solution), or a conjugate-gradient-based iteration (for a high quality approximation). We compare our estimators with these methods, and consider two cost functions:

- The reconstruction error $e_r(f) = \|f - f_\top\|_2/n$, measuring the quality of the denoising.
- The approximation error $e_a(f) = \|f - f_*\|_2/n$, measuring the quality of the approximation of f_*.

Our estimators and the conjugate-gradient algorithms are respectively parametrized by the number of MTSFs used and the number of gradient steps, that we both denote by m.

Setup. For each graph, we generate a B-bandlimited signal $f_\top$, each u_i being weighted by a random complex Gaussian value $a_i \sim \mathcal{N}_\mathbf{C}(0, \sigma^2)$, with σ^2 such that the SNR is equal to 2. We then degrade $f_\top$ with some additive Gaussian noise $\varepsilon \sim \mathcal{N}_\mathbf{C}(0, 1)$ (independently on each entry), and determine the optimal parameter q_* for which $e_r(f_*) = e_r(q_*(\mathsf{L}_\theta + q_*\mathsf{I})^{-1}g)$ is minimized[11], before measuring the errors and running time associated to different values of m (taken amongst 10 logarithmically-spaced values from 1 to 100). The iterative algorithms are initialized at g.

We compare the following estimation strategies: the estimators $\overline{f}$ and $\hat{f}$, the conjugate-gradient descent with no preconditioner, with a simple diagonal preconditioner $\mathsf{P} = (q^{-1}\mathsf{D} +$

[10] The linear subspace generated by the vectors u_1,...,u_B.

[11] We perform our search in $(0, 30)$.

$\mathsf{I})^{-1}$, and with a CROUT ILU preconditioner[12]. Computing this high-quality ILU preconditioner is too expensive to be competitive with the other methods, and we only include these results on one type of graph, for illustration purposes.
Each runtime measurement is averaged over 100 runs. The results are averaged over 5 realizations of the noise ε and 10 samples for each random graph model. We plot in Figure 3.2 the mean results.

Graphs used. We use the following graphs:

- An ε-graph obtained by sampling $|\mathcal{V}| = 10000$ i.i.d. points x_i's in $[0,1]^3$, with an edge between x_i and x_j whenever $\|x_i - x_j\|_2 < 0.1$.
- A graph generated from a *Stochastic Block Model* (SBM), with $|\mathcal{V}| = 10000$ nodes, each belonging to one of two communities C_1 and C_2 of size 5000 each. In this model, an edge is drawn randomly between two nodes i, j with probability equal to $\frac{c_{k,l}}{n}$, where k, l denote respectively the community label of i and j. Here, we take $c_{1,1} = c_{2,2} = 36$ and $c_{1,2} = c_{2,1} = 4$, resulting in average degree $\overline{d} = 40$.
- A graph generated from a related *Degree-Corrected Stochastic Block Model* (DC-SBM 1), with two communities of size 5000 and an edge between nodes $i \in C_k$ and $j \in C_l$ present with probability proportional to $p_i p_j \frac{c_{k,l}}{n}$. p_i is a randomly sampled positive real value, representing the intrinsic *connectivity* of node i, with $\mathbf{E}(p_i) = 1$ and finite second moment [32]. Here, we take $c_{1,1} = c_{2,2} = 36$, $c_{1,2} = 4$, and the p_i's are drawn from a normalized mixture of Gaussian distributions[13], resulting in a graph with mean degree $\overline{d}$ close to 40. The objective of adding this model is to illustrate, at constant density, how the degree distribution affects the results.
- Another DC-SBM-graph (DC-SBM 2), with higher density (typically with an average degree more than 10 times that of the previous DC-SBM model). We take two communities of size 5000, $c_{1,1} = c_{2,2} = 480$, $c_{1,2} = 20$, and the p_i's are drawn from another mixture of Gaussian distributions[14]. The objective of this second DC-SBM model is to illustrate how density affects the results.
- We also illustrate the results on a real-world graph: a relationship graph for internet Autonomous Systems (AS), recorded by the Center for Applied Internet Data Analysis (CAIDA) and provided in the SNAP datasets [41][15].

In the event of a randomly generated graph containing isolated nodes, we remove them so that the graph is connected (to make the interpretations simpler). We summarize in Table 3.1 the information concerning the graphs generated, and the associated optimal q_*'s (averaged over all realizations of the noise ε). All those graphs are endowed with a synthetic connection as specified in Equation (3.1). The parameters used for the (DC) SBM models ensures that the resulting graphs have a strong community structure.

[12] We use the implementations from the libraries: https://github.com/JuliaLinearAlgebra/IterativeSolvers.jl, https://github.com/JuliaLinearAlgebra/Preconditioners.jl and https://github.com/haampie/IncompleteLU.jl. For the ILU preconditioner, we fix the drop threshold to 0.1.

[13] Specifically, the connectivity parameters are independently sampled from a mixture of $\mathcal{N}(50, 20)$, $\mathcal{N}(500, 100)$ and $\mathcal{N}(10000, 100)$, with weights of 0.59, 0.4 and 0.01 respectively, and then normalized.

[14] Here from the mixture of $\mathcal{N}(50, 20)$, $\mathcal{N}(1000, 50)$, $\mathcal{N}(5000, 100)$ and $\mathcal{N}(10000, 100)$, with weights of 0.45, 0.1, 0.44 and 0.01.

[15] We use the library available at https://github.com/JuliaGraphs/SNAPDatasets.jl.

Graph	$\lvert\mathcal{V}\rvert$	$\overline{d}$	$d_{\min}$	$d_{\max}$	q_*
ε-graph	10000	37.3	5.2	67.5	6.518
SBM	10000	39.87	17.7	66.6	21.04
DC-SBM 1	9833.7	33.4	1	906.1	2.634
DC-SBM 2	9932.8	418.8	1	1513.7	4.31
AS CAIDA	26475	4.032	1	2628	0.4808

Table 3.1: Experimental parameters associated to each graph. Values are averaged over all 10 realizations for random graph models.

For the ε-graph and the AS-graph, we arbitrarily set $B = 5$ when generating the bandlimited signal $f_\top$. For (DC) SBMs, we take $B = 2$. It is known that the eigenvectors of the *combinatorial* Laplacian L encode the community structure of these graphs (here,, and taking $B = 2$ here results in signals coherent with the community structure (see Section 12 of the Supplementary Material for more on the first few eigenvectors of L_θ).

Discussion. Let us first comment on the approximation error for the ε-graph (Figure 3.2a, left). Both of our estimators converge linearly in log-log-scale with a mild slope, as expected for Monte-Carlo estimators. Any of the other three conjugate-gradient-based algorithms achieves a better approximation error within 3 steps than our estimators in 100 steps (as expected for Monte-Carlo convergence rates), and our methods are not competitive in this setting. However, in this denoising setting, the quality of the denoising is reflected in the reconstruction error (depicted in all the plots of Figure 3.2). Overall, we observe that:

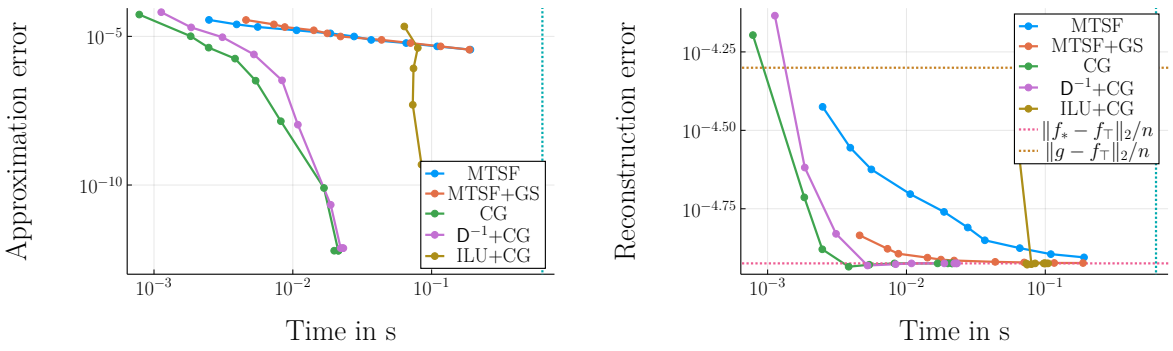
(1) $\hat{f}$ consistently performs better than $\overline{f}$, but how much better depends on the graph (see *e.g.* SBM and DC-SBM 1), and improvements are smaller for graphs with more heterogeneous degree distributions (DC-SBM graphs).
(2) Compared with MTSF-based estimators, the CG algorithms perform *worse* on graphs with heterogeneous degree distributions.
(3) CG solvers perform worse than our methods when the density increases.

CG's somewhat poor performance on DC-SBM 1 (bullet point (2)) is partly explained by the conditioning of the system considered: a bound on the condition number κ is given by

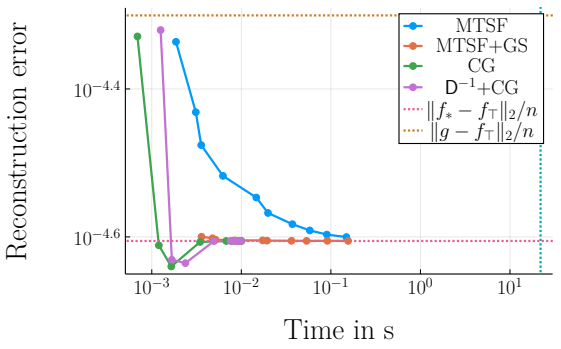
$$\kappa = \frac{q_* + \lambda_n}{q_* + \lambda_1} \leq \frac{q_* + 2\max_{v\in V} d_v}{q_* + \lambda_1}, \tag{3.3}$$

where $\lambda_1 \leq \ldots \leq \lambda_n$ are the eigenvalues of L_θ, and where λ_1 is a measure of the quality of the optimal angular assignment (the lower λ_1 is, the better the quality of the synchronization), which is small for our synthetic connection (good synchronization due to low incoherence). CG iterations are slower for high-density graphs ($\mathcal{O}(|\mathcal{E}|)$ time *per iteration*), and $\overline{f}$ and $\hat{f}$ reach near-optimal reconstruction around 10 *times faster* than (preconditionned) CG on DC-SBM 2 (with expected complexities in $\mathcal{O}\left(\frac{|\mathcal{E}|}{q} + |\mathcal{V}|\right)$ and $\mathcal{O}\left(\left(1 + \frac{1}{q}\right)|\mathcal{E}| + |\mathcal{V}|\right)$ respectively).

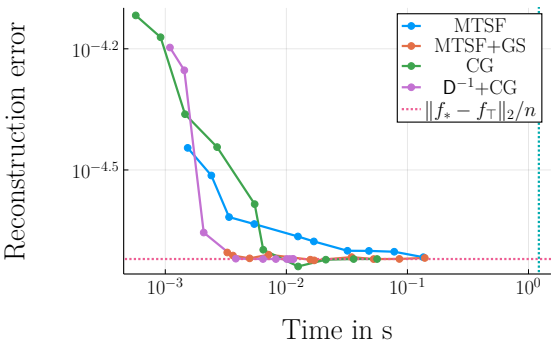
Our observations so far show that our estimators are competitive with standard methods,



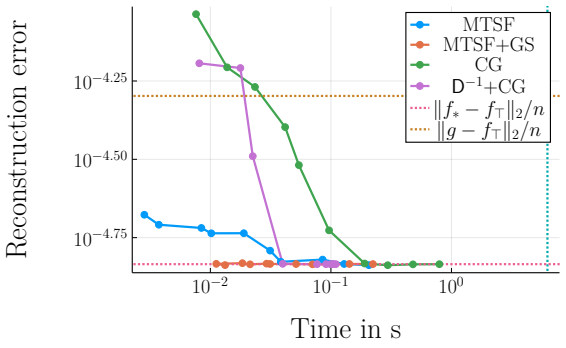
(a) ε-graph. Left: Approximation error. Right: Reconstruction error.



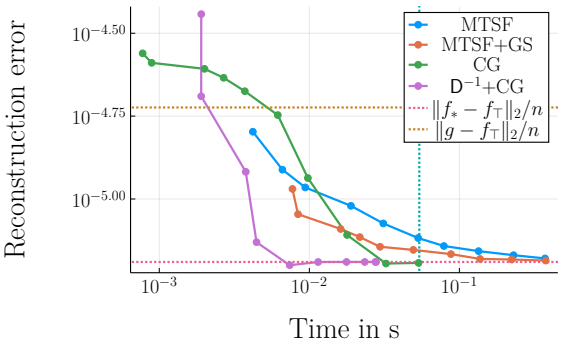
(b) SBM Reconstruction error.



(c) DC-SBM 1 Reconstruction error.



(d) DC-SBM 2 Reconstruction error.



(e) AS CAIDA Reconstruction error.

Figure 3.2: Runtime-precision trade-offs for Tikhonov smoothing with optimal q. Each datapoint corresponds to a value of m. The vertical blue line records the average runtime of the Cholesky-based solver, the horizontal red line the average reconstruction error of the exact solution to the Tikhonov problem, and the horizontal yellow line the average reconstruction error of the noisy signal. Vertical bars represent standard deviations.

especially for low-accuracy solutions, and in the following situations:

- On graphs resulting in poorly conditioned systems (with high-degree nodes and for low values of q_*).
- On graphs with higher density (to which MTSF-based methods less sensitive).

Further profiling also shows that the most expensive operations during the computation of $\overline{f}$ (and $\hat{f}$) are the instantiation of the data structures we use, and the successive memory accesses to the different θ_e. These issues will likely benefit from further democratization of randomized computational schemes, with more efficient compiler optimizations and architectures specialized for these types of computations. Note also that the connection model used here is only representative of a subset of applications (*e.g.* [71], [16] or [24], see also Remark 1.1).

4. Randomized Angular Synchronization. We describe in this Section how our novel estimators can be successfully applied to angular synchronization. Recall from Section 1.2 that, to perform angular synchronization, we aim to minimize the incoherence

$$\underset{f \in U(\mathbf{C})^n}{\operatorname{argmin}} \ \langle f, \mathsf{L}_\theta f \rangle, \tag{4.1}$$

which is NP-hard in general [72]. We go over existing approaches before introducing in Section 4.1 a randomized schemed based on a spectral relaxation of Problem (4.1).

Spectral relaxation and other existing approaches. Problem (4.1) is often relaxed to the following form [63, 71]:

$$\underset{\|f\|_2^2 = n}{\operatorname{argmin}} \ \langle f, \mathsf{L}_\theta f \rangle, \tag{4.2}$$

the solution of which is given by the eigenvector associated to the *smallest* eigenvalue of L_θ. This solution differs from the exact solution of Problem (4.1) (even though guarantees on its quality exist, *e.g.* [23]), and can be computed using either an inverse power method, a Rayleigh quotient iteration, or a Lanczos iteration [57]. Most theoretical studies focus on Erdös-Rényi graphs, and suggest that spectral relaxations work best in small noise regimes (as compared with other algorithms, regarding the resulting error) [10, 40, 50].

Other existing methods include (see [18] for a similar discussion):

- Semi-definite relaxations [63], providing flexible theoretical tools and performance similar to spectral relaxations, but impractical past mid-sized instances ($n \simeq 10^4$).
- The *generalized power method* proposed in [10], provably reaching the optimal solution of Problem (4.1) in the presence of (low) Gaussian noise, on complete graphs.
- Graph Neural Networks, with state-of-the-art performance for high noise [27, 30].
- Message-passing algorithms. For Gaussian noise, Approximate Message Passing has been conjectured to be statistically optimal among polynomial time algorithms [50], even at higher noise levels, but is limited to very dense graphs. Cycle-Edge Message-Passing allows exact recovery under a theoretical corruption model (with linear rate), but is more computationally-expensive than spectral relaxations [40].
- Descent techniques such as [46], with exact recovery under a (different) corruption model (see also [42]).

Note that message-passing algorithms and descent techniques (along with others we did not mention as well) can be applied to synchronization problems over more general groups, but are often not competitive on benchmarks for *angular* synchronization [27].

We will now discuss how to use our randomized estimators for eigenvector-computation.

4.1. Proposed Approach. One way to solve the spectral relaxation of Equation (4.2) is to perform an inverse power iteration, by setting $f_0 \in \mathbf{C}^{\mathcal{V}}$ and iterating:

$$f_{r+1} = \frac{\mathsf{L}_\theta^{-1} f_r}{\|\mathsf{L}_\theta^{-1} f_r\|_2} \tag{4.3}$$

until convergence. Note that for this iteration to be well-defined, L_θ needs to be invertible, *i.e.*, the angular synchronization problems needs to be non-trivial.

Our approach stems from the observation that L_θ and $q^{-1}(\mathsf{L}_\theta + q\mathsf{I})$ share the same eigenvectors. The power method applied to this regularized matrix consists in computing iterations of the form:

$$f_{r+1} = \frac{q(\mathsf{L}_\theta + q\mathsf{I})^{-1} f_r}{\|q(\mathsf{L}_\theta + q\mathsf{I})^{-1} f_r\|_2}, \tag{4.4}$$

where an estimation of $q(\mathsf{L}_\theta + q\mathsf{I})^{-1} f_r$ can be obtained by our estimators $\overline{f}$ and $\hat{f}$ (or other deterministic algorithms). The convergence of this iteration is geometric with ratio

$$\frac{\mu_1}{\mu_2} = \frac{\lambda_2 + q}{\lambda_1 + q}, \tag{4.5}$$

where $\mu_1 \leq \mu_2$ and $\lambda_1 \leq \lambda_2$ denote respectively the two smallest eigenvectors of $q(\mathsf{L}_\theta + q\mathsf{I})^{-1}$ and L_θ [57]. The choice of q is then a trade-off between:

- Low values of q that induce smaller $\frac{\lambda_2+q}{\lambda_1+q}$ ratio, favoring faster convergence of the power iteration.
- Large values of q that enable a faster sampling time of MTSFs (in $\mathcal{O}\left(\frac{|\mathcal{E}|}{q}\right)$).

Remark 4.1. *One could instead maximize $\langle f, \mathsf{A}_\theta f \rangle$ in Equation (4.1). This formulation is common in theoretical works, that mostly focus on (often dense) Erdös-Rényi graphs, and applying the power method to A_θ is efficient on these graphs. More involved estimators are necessary in case the ratio α_1/α_2 of the top eigenvalues of A_θ is large. For trivial connections, such cases include: regular grids (with α_1/α_2 growing worse with the dimension), graphs with homogeneous spatial correlations* (e.g., *ε-graphs, nearest-neighbors-graphs), or graphs with homogeneous degree distributions* and *bottlenecks (poor expansion) such as in SBMs (for regular graphs, this is the Cheeger inequality). Our experiments (not shown) suggest that the same behavior arises for non-trivial connections. Further, Cramér-Rao bounds for angular synchronization suggest that angular synchronization is difficult on these graphs [11].*
It is also possible to consider a normalized *Laplacian $\widetilde{\mathsf{L}_\theta} = \mathsf{D}^{-\frac{1}{2}} \mathsf{L}_\theta \mathsf{D}^{-\frac{1}{2}}$ in the spectral relaxation (4.2), which was proposed in [17, 71] and comes with similar guarantees [5]. Our approach is flexible enough to be used in this case. Indeed, our MTSF-based estimators extend*

to quantities such as $f_\circ = (\mathsf{L}_\theta + q\mathsf{D})^{-1}(qD)g'$ *(see Section 9 of the Supplementary Material), which allows to estimate* $q(\widetilde{\mathsf{L}_\theta} + q\mathsf{I})^{-1}g$ *by setting* $g' = \mathsf{D}^{-\frac{1}{2}}g$ *and leveraging the decomposition*

$$q(\widetilde{\mathsf{L}_\theta} + q\mathsf{I})^{-1}g = \mathsf{D}^{\frac{1}{2}}((\mathsf{L}_\theta + q\mathsf{D})^{-1}(q\mathsf{D}))g'. \tag{4.6}$$

4.2. Illustration: Intra-Class Denoising. Before delving into precise computation time comparisons with state-of-the-art methods, we first illustrate our proposed method on a toy denoising problem, inspired from an application in cryo-EM [59, 74].

We consider n copies $(I_i)_{i=1}^n$ of some image I_* that have been rotated and degraded:

$$I_i = r_i(I_*) + \varepsilon, \tag{4.7}$$

with r_i a rotation and $\varepsilon \sim \mathcal{N}(0, \sigma^2\mathsf{I})$ some Gaussian noise. The rotations r_i are unknown, and the goal is to recover image I_*. This is a simplified version of the intra-class denoising problem in cryo-EM, where each (2D) image corresponds to a noisy projection of a (3D) molecule observed in an unknown orientation.

In the absence of rotations r_i, a solution consists in averaging the I_i's, trivially recovering I_* as $n \to \infty$. This strategy fails in our setting (due to the rotations), and we need to estimate the r_i's before denoising. To do that, we first estimate angles $\theta_{i,j}$ between a subset of pair of images $\mathcal{E}$ (representing the edges of a graph) using image moments [26][16], and perform angular synchronization to estimate the rotations r_i's. We then rotate and average the images I_i's.

We take I_* the 256×256 Shepp-Logan phantom [61], $n = 1000$, $q = 10^{-3}$, $k = 20$ iterations of the power method, and randomly choose the underlying graph $\mathcal{G} \sim ER(n, p)$ with $p = \frac{5}{n}$. We uniformly sample rotations r_i with angles in $(-\frac{\pi}{2}, \frac{\pi}{2})$[17]. We display the recovered images in Figure 4.1 (solving the spectral relaxation (4.2) using both our method, with $m = 10$ MTSFs for the estimator $\overline{f}$ at each step, and an exact solver), for different noise levels $\sigma^2 \in \{1, 3, 5, 10\}$. We also plot the image reconstruction error $e_I(x) = \|I_* - x\|_2$ for $\sigma^2 = 5$ with varying values of m. Here, we have no guarantee that the connection resulting from the image-moment-based estimation is weakly-incoherent, and do not perform importance sampling. Instead, we always accept a cycle in Algorithm 2.3 if $\cos(\theta_C) \leq 0$, and apply estimator $\overline{f}$ on the resulting MTSF.

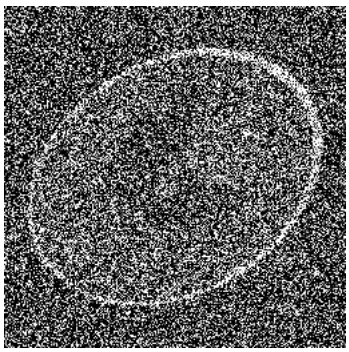
Even though the connection may not be weakly-incoherent, the MTSF-based synchronization allows to obtain results of a quality very similar to exact synchronization on this problem, even at higher noise levels.

We will now investigate trade-offs between the quality of the synchronization and the runtime of our methods depending on the solver used to compute the power-method iteration in Equation 4.4, and the topology of the graph.

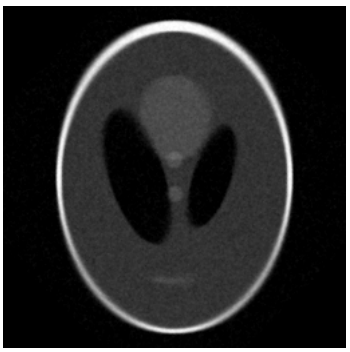
4.3. Numerical Evaluation. We compare the performance of the method from Section 4.1 on synthetic graph data using four different iterative solvers: $\overline{f}$, $\hat{f}$ and a conjugate-gradient descent with and without diagonal preconditioning.

[16] Specifically, we estimate the orientation of each of the two images using the eigenvectors of the matrix of its covariant moments, and take $\theta_{i,j}$ the angle between these two sets of eigenvectors

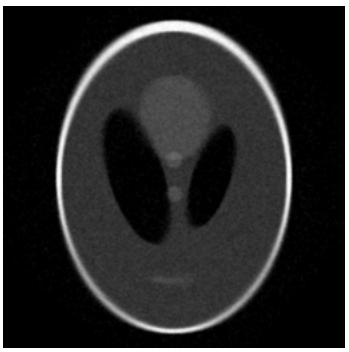
[17] We restrict the possible rotations so that we can use the simple image-moment-based registration, other methods could be used instead.



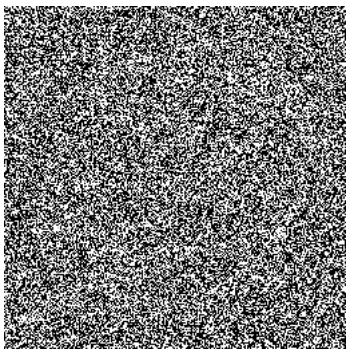
(a) $r(I_*) + \varepsilon, \sigma^2 = 1$



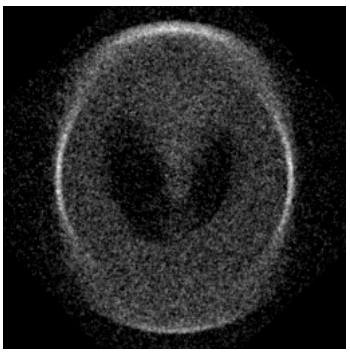
(b) $\sigma^2 = 1$, exact



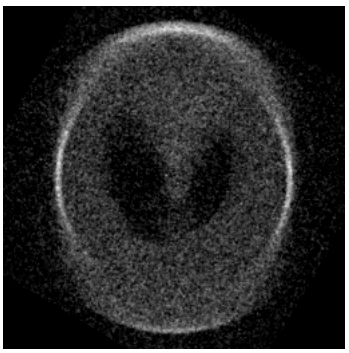
(c) $\sigma^2 = 1$, MTSF



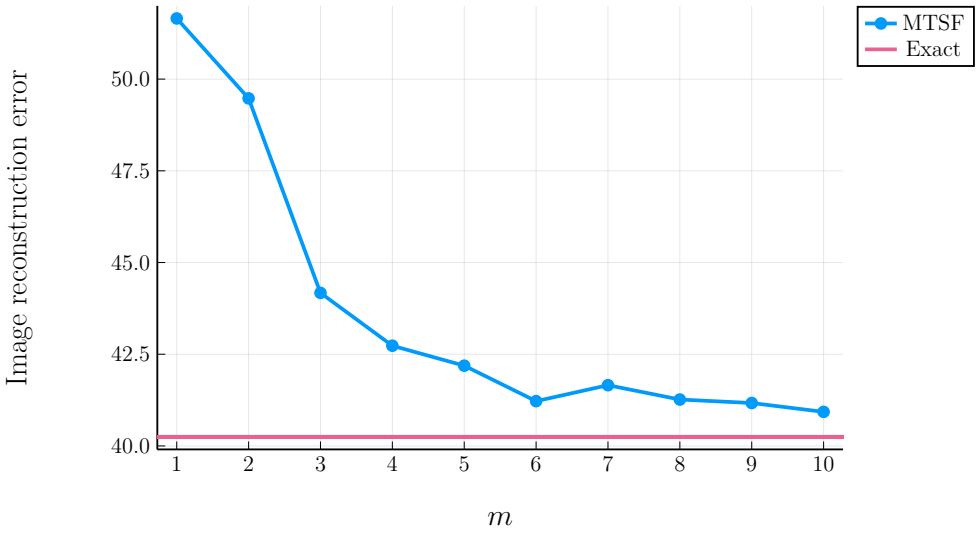
(d) $r(I_*) + \varepsilon, \sigma^2 = 5$



(e) $\sigma^2 = 5$, exact



(f) $\sigma^2 = 5$, MTSF



(g) Error for $\sigma^2 = 5$, $m \in \{1, ..., 10\}$,

Figure 4.1: Examples of recovery from noisy images, with different noise levels, using both the exact solution to the spectral relaxation (4.2) and the approach from Section 4.1.

Setup. We work with the SBM and DC-SBM graph models discussed in Section 3.2. We endow each graph G with a random connection generated according to the model described in Section 3, for both coherent ($\eta = \frac{\pi}{2n}$) and incoherent connections ($\eta = \frac{\pi}{10}$), and aim to recover the vector x with $x_i = e^{\iota\omega_i}$. Starting from f_0 taken uniformly in $U(\mathbf{C})^n$, we perform k iterations of the power iteration of Equation (4.4), estimating the solution with m MTSFs (resp. conjugate-gradient iterations) for each of our methods. We use $m = 3$ for coherent connections, and $m = 10$ for incoherent ones. For incoherent connections, we *do* use the importance sampling strategy from Remark 2.10[18]. We average the synchronization errors

$$e_s(f) = \min_{r \in U(\mathbf{C})} \frac{\|f - rx\|_2}{n} \tag{4.8}$$

over 20 executions, and measure the mean runtimes over 100 runs. We set $q = 10^{-2} \times \overline{d}$ and take measurements for different values of k (10 logarithmically-spaced values in $\{1, ..., 100\}$). The results are averaged over 10 realizations of the random graphs. See Figure 4.2.
We also take measurements for a Lanczos-iteration-based computation[19] (for the matrix L_θ), and for a naive synchronization algorithm going as follows. First, fix a root node $v \in V$, then:

- sample a spanning tree of G uniformly (a UST, using Wilson's algorithm [69]),
- propagate the value from v to the other nodes (taking into account the offsets).

Note that this procedure does *not* depend on k or m. Finally, we also evaluate a similar strategy propagating along a *maximum spanning tree* (MST), with respect to the edge-weights $w_{i,j} = |\cos(\theta_{i,j})|$ (a strategy inspired from surface reconstruction techniques such as [28]).

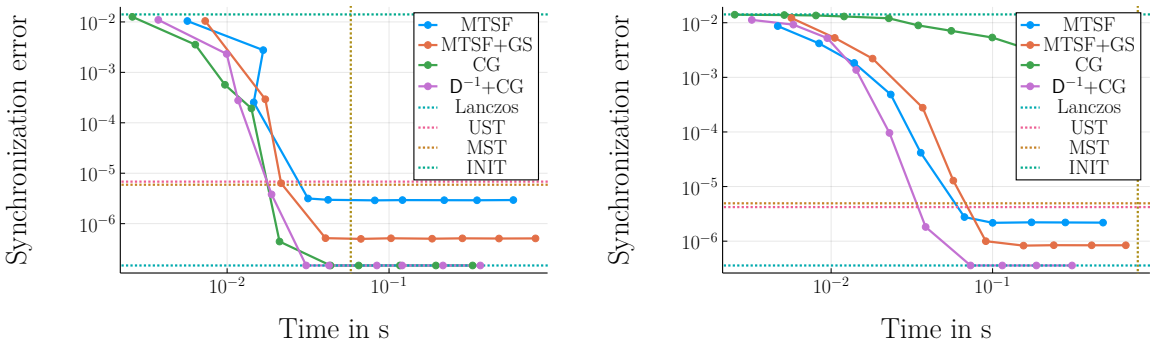
Comments. Results vary greatly depending on the graph and connection. For coherent connections ($\eta = \frac{\pi}{2n}$), we observe the following:

(1) Krylov-subspaces-based methods are sensitive to : all methods perform similarly on the SBM, but the CG-based power-methods and Lanczos iteration converge more slowly on the (less well-conditioned) DC-SBM 1 graph.
(2) Higher density results in significant slow-downs for Krylov-subspaces methods, and MTSF-based iterations are roughly 10 times faster than the preconditioned-CG iteration for equivalent precision on the DC-SBM 1.

For these coherent connections, the tree-based methods also achieve good synchronization quality, and are cheap to compute. This is no longer the case for incoherent connections (DC-SBM 1 with $\eta = \frac{\pi}{10}$): in this case, our estimators require more MTSFs (resp. CG iterations) to converge ($m = 10$), and achieve a synchronization error around 10 times smaller than tree-based methods.

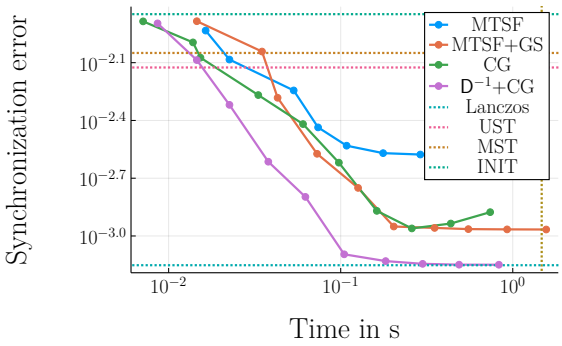
Remark 4.2. *We also experimented with two other estimation strategies (not shown): the spectral relaxation using the normalized Laplacian $\widetilde{\mathsf{L}_\theta}$, and a generalized-power-method-like algorithm, replacing the global normalization in Equation (4.4) by a component-wise normalization, so that each f_k belongs to $U(\mathbf{C})^n$ (like in [10]). In our experiments, we did not observe any qualitative differences with the method described in Section 4.1.*

[18] Here, there is no issue with estimating af_* instead of f_* due to the re-normalization in Equation (4.4).
[19] We use the variant implemented in https://jutho.github.io/KrylovKit.jl/stable/man/eig/#KrylovKit.eigsolve.

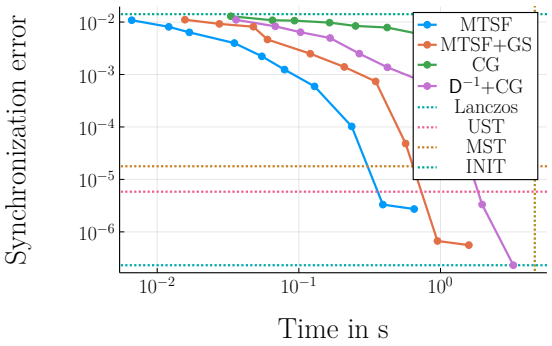


(a) SBM, $\eta = \frac{\pi}{2n}$, $m = 3$.

(b) DC-SBM 1, $\eta = \frac{\pi}{2n}$, $m = 3$.



(c) DC-SBM 1, $\eta = \frac{\pi}{10}$, $m = 10$.



(d) DC-SBM 2, $\eta = \frac{\pi}{2n}$, $m = 3$.

Figure 4.2: Runtime-precision trade-offs for angular synchronization for $m = 10$. Each data point corresponds to a value of k. The vertical yellow line records the average runtime of the Lanczos-based computation, the horizontal blue line the synchronization error for the Lanczos-based computation. The horizontal red (resp. orange) line is the mean synchronization error for the UST-based (resp. MST-based) synchronization, and the horizontal teal line the error for the random initialization.

For spectral relaxations, our results suggest that MTSF-based solutions perform no-worse than deterministic solutions, and get comparatively better as the density of the graph increases. These techniques are mostly suitable for approximations, especially since previous studies (*e.g.* [50]) suggest that spectral-relaxation-based techniques mainly offer good-quality solutions at *low* noise-level, but are otherwise *sub-optimal*. One drawback of our strategy is that it requires to set a value for q: further refinements may include adapting our methodology to Rayleigh-quotient iterations [66], such that q no longer needs to be fixed, and that better convergence rates may be obtained.

5. Conclusion and Perspectives. We proposed MTSF-based estimators for the problems of smoothing (in presence of a connection) and angular synchronization, reaching competitive performance on both problems, especially in low-precision regimes, and even *without* imple-

menting parallelization. Our estimators perform propagation along the branches of MTSFs and, as compared to standard deterministic methods, have another noteworthy advantage: they are insensitive to both the density of the graph *and* the conditioning of the system considered (which is typically bad for graphs with broad degree distributions). Such techniques can apply to a variety of other problems: one result we did not include is the extension of the smoothing estimator to the interpolation problem, which may find applications to *e.g.* synchronization in presence of anchor nodes.

This opens a number of research directions. From the theoretical side: could we lift the weak-inconsistency condition, following the development in Section 13 of the Supplementary Material? Could similar estimators be developed for $O(\mathbf{R}^d)$ synchronization? Instead of using a fixed number m of MTSFs for the estimation (or CG iterations) of each of k iterations of the power method, how should those km MTSFs be used the to achieve maximal precision?

From a more applied perspective, many applications of angular synchronization only consider the *exact* solution of the spectral relaxation, but how much loss in precision (and gain in speed) is actually acceptable? Our estimators can be seamlessly implemented on distributed systems (with communication complexity of the order of the number of steps in the sampling algorithm): could this be leveraged in practical applications? Finally, can random graph decompositions such as MTSFs be useful in other iterative inference algorithms, for instance message-passing algorithms [40, 47]?

REFERENCES

[1] B. Alexeev, A. S. Bandeira, M. Fickus, and D. G. Mixon, *Phase Retrieval with Polarization*, SIAM J. Imaging Sci., 7 (2014), pp. 35–66.

[2] M. Arie-Nachimson, S. Z. Kovalsky, I. Kemelmacher-Shlizerman, A. Singer, and R. Basri, *Global motion estimation from point matches*, in 2012 Second international conference on 3D imaging, modeling, processing, visualization & transmission, IEEE, 2012, pp. 81–88.

[3] L. Avena and A. Gaudillière, *Random spanning forests, Markov matrix spectra and well distributed points*, arXiv preprint arXiv:1310.1723, (2013).

[4] ———, *Two applications of random spanning forests*, Journal of Theoretical Probability, 31 (2018), pp. 1975–2004.

[5] A. S. Bandeira, A. Singer, and D. A. Spielman, *A Cheeger inequality for the graph connection Laplacian*, SIAM Journal on Matrix Analysis and Applications, 34 (2013), pp. 1611–1630.

[6] S. Barthelme, N. Tremblay, A. Gaudilliere, L. Avena, and P.-O. Amblard, *Estimating the inverse trace using random forests on graphs*, in GRETSI 2019 - XXVIIème Colloque francophone de traitement du signal et des images, Lille, France, Aug. 2019.

[7] D. Blackwell, *Conditional expectation and unbiased sequential estimation*, The Annals of Mathematical Statistics, (1947), pp. 105–110.

[8] B. Bollobás, *Modern Graph Theory*, vol. 184, Springer Science & Business Media, 1998.

[9] P. Bouboulis, *Wirtinger's calculus in general Hilbert spaces*, arXiv preprint arXiv:1005.5170, (2010).

[10] N. Boumal, *Nonconvex Phase Synchronization*, SIAM J. Optim., 26 (2016), pp. 2355–2377.

[11] N. Boumal, A. Singer, P.-A. Absil, and V. D. Blondel, *Cramér–Rao bounds for synchronization of rotations*, Information and Inference: A Journal of the IMA, 3 (2014), pp. 1–39.

[12] D. Brydges, J. Fröhlich, and E. Seiler, *On the construction of quantized gauge fields. I. General results*, Annals of Physics, 121 (1979), pp. 227–284.

[13] É. Cartan, *Les groupes d'holonomie des espaces généralisés*, Acta mathematica, 48 (1926), pp. 1–42.

[14] S. Chen, G. Huang, G. Piccioli, and L. Zdeborová, *Planted X Y model: Thermodynamics and inference*, Physical Review E, 106 (2022), p. 054115.

[15] F. R. Chung, *Spectral graph theory*, vol. 92, American Mathematical Soc., 1997.

[16] M. Cucuringu, *Sync-Rank: Robust Ranking, Constrained Ranking and Rank Aggregation via Eigenvector and SDP Synchronization*, IEEE Trans. Netw. Sci. Eng., 3 (2016), pp. 58–79.
[17] M. Cucuringu, Y. Lipman, and A. Singer, *Sensor network localization by eigenvector synchronization over the Euclidean group*, ACM Transactions on Sensor Networks (TOSN), 8 (2012), pp. 1–42.
[18] M. Cucuringu and H. Tyagi, *An extension of the angular synchronization problem to the heterogeneous setting*, arXiv preprint arXiv:2012.14932, (2020).
[19] M. Derezinski and M. W. Mahoney, *Determinantal point processes in randomized numerical linear algebra*, Notices of the American Mathematical Society, 68 (2021), pp. 34–45.
[20] M. Derezinski and M. K. Warmuth, *Reverse Iterative Volume Sampling for Linear Regression*, J. Mach. Learn. Res., 19 (2018), pp. 23:1–23:39.
[21] P. Drineas and M. W. Mahoney, *RandNLA: randomized numerical linear algebra*, Commun. ACM, 59 (2016), pp. 80–90.
[22] M. Fanuel and R. Bardenet, *Sparsification of the regularized magnetic Laplacian with multi-type spanning forests*, arXiv preprint arXiv:2208.14797, (2022).
[23] F. Filbir, F. Krahmer, and O. Melnyk, *On recovery guarantees for angular synchronization*, Journal of Fourier Analysis and Applications, 27 (2021), p. 31.
[24] S. Furutani, T. Shibahara, M. Akiyama, K. Hato, and M. Aida, *Graph Signal Processing for Directed Graphs Based on the Hermitian Laplacian*, in Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2019, Würzburg, Germany, September 16-20, 2019, Proceedings, Part I, U. Brefeld, É. Fromont, A. Hotho, A. J. Knobbe, M. H. Maathuis, and C. Robardet, eds., vol. 11906 of Lecture Notes in Computer Science, Springer, 2019, pp. 447–463.
[25] A. Giridhar and P. R. Kumar, *Distributed Clock Synchronization over Wireless Networks: Algorithms and Analysis*, in 45th IEEE Conference on Decision and Control, CDC 2006, San Diego, CA, USA, December 13-15, 2006, IEEE, 2006, pp. 4915–4920.
[26] R. C. Gonzalez, *Digital image processing*, Pearson education india, 2009.
[27] Y. He, G. Reinert, D. Wipf, and M. Cucuringu, *Robust Angular Synchronization via Directed Graph Neural Networks*, arXiv preprint arXiv:2310.05842, (2023).
[28] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, *Surface Reconstruction from Unorganized Points*, in Proceedings of the 19th annual conference on computer graphics and interactive techniques, 1992, pp. 71–78.
[29] J. B. Hough, M. Krishnapur, Y. Peres, and B. Virág, *Determinantal Processes and Independence*, Probability Surveys, 3 (2006), pp. 206 – 229.
[30] N. Janco and T. Bendory, *Unrolled algorithms for group synchronization*, IEEE Open Journal of Signal Processing, (2023).
[31] H. Jaquard, M. Fanuel, P.-O. Amblard, R. Bardenet, S. Barthelmé, and N. Tremblay, *Smoothing Complex-Valued Signals on Graphs with Monte-Carlo*, in ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2023, pp. 1–5.
[32] B. Karrer and M. E. Newman, *Stochastic blockmodels and community structure in networks*, Physical review E, 83 (2011), p. 016107.
[33] A. Kassel and R. Kenyon, *Random curves on surfaces induced from the Laplacian determinant*, The Annals of Probability, 45 (2017), pp. 932 – 964.
[34] A. Kassel and T. Lévy, *Covariant symanzik identities*, Probability and Mathematical Physics, 2 (2021), pp. 419–475.
[35] R. Kenyon, *SPANNING FORESTS AND THE VECTOR BUNDLE LAPLACIAN*, The Annals of Probability, 39 (2011), pp. 1983–2017.
[36] D. P. Kroese, T. Taimre, and Z. I. Botev, *Handbook of monte carlo methods*, John Wiley & Sons, 2013.
[37] A. B. Kuijlaars, *Convergence analysis of Krylov subspace iterations with methods from potential theory*, SIAM review, 48 (2006), pp. 3–40.
[38] A. Kulesza and B. Taskar, *Determinantal Point Processes for Machine Learning*, Found. Trends Mach. Learn., 5 (2012), pp. 123–286.
[39] R. Kyng and Z. Song, *A matrix chernoff bound for strongly rayleigh distributions and spectral sparsifiers from a few random spanning trees*, in 2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS), IEEE, 2018, pp. 373–384.

[40] G. Lerman and Y. Shi, *Robust Group Synchronization via Cycle-Edge Message Passing*, Foundations of Computational Mathematics, 22 (2022), pp. 1665–1741.
[41] J. Leskovec and A. Krevl, *SNAP Datasets: Stanford Large Network Dataset Collection*. http://snap.stanford.edu/data, June 2014.
[42] H. Liu, X. Li, and A. M.-C. So, *ReSync: Riemannian Subgradient-based Robust Rotation Synchronization*, arXiv preprint arXiv:2305.15136, (2023).
[43] P. Lorenzo, S. Barbarossa, and P. Banelli, *Sampling and recovery of graph signals*, in Cooperative and Graph Signal Processing, Elsevier, 2018, pp. 261–282.
[44] O. Macchi, *The coincidence approach to stochastic point processes*, Advances in Applied Probability, 7 (1975), pp. 83–122.
[45] P. Martinsson and J. A. Tropp, *Randomized numerical linear algebra: Foundations and algorithms*, Acta Numer., 29 (2020), pp. 403–572.
[46] T. Maunu and G. Lerman, *Depth descent synchronization in SO (d)*, International journal of computer vision, 131 (2023), pp. 968–986.
[47] M. Mezard and A. Montanari, *Information, physics, and computation*, Oxford University Press, 2009.
[48] L. A. S. Moreira, A. L. L. Ramos, M. L. R. de Campos, J. A. Apolinário, and F. G. Serrenho, *A Graph Signal Processing Approach to Direction of Arrival Estimation*, in 27th European Signal Processing Conference, EUSIPCO 2019, A Coruña, Spain, September 2-6, 2019, IEEE, 2019, pp. 1–5.
[49] M. E. Muller, *Some continuous Monte Carlo methods for the Dirichlet problem*, The Annals of Mathematical Statistics, (1956), pp. 569–589.
[50] A. Perry, A. S. Wein, A. S. Bandeira, and A. Moitra, *Message-Passing Algorithms for Synchronization Problems over Compact Groups*, Communications on Pure and Applied Mathematics, 71 (2018), pp. 2275–2322.
[51] Y. Y. Pilavci, P. Amblard, S. Barthelmé, and N. Tremblay, *Graph Tikhonov Regularization and Interpolation Via Random Spanning Forests*, IEEE Trans. Signal Inf. Process. over Networks, 7 (2021), pp. 359–374.
[52] Y. Y. Pilavci, P.-O. Amblard, S. Barthelmé, and N. Tremblay, *Variance Reduction in Stochastic Methods for Large-Scale Regularized Least-Squares Problems*, in 2022 30th European Signal Processing Conference (EUSIPCO), IEEE, 2022, pp. 1771–1775.
[53] G. Puy, N. Tremblay, R. Gribonval, and P. Vandergheynst, *Random sampling of bandlimited signals on graphs*, Applied and Computational Harmonic Analysis, 44 (2018), pp. 446–475.
[54] T. C. Raia, M. G. P. Thomas, F. G. Serrenho, and J. A. Apolinário Jr, *Gsp-based doa estimation for a multimission radar*, Proceedings of the XXXVIII Simpósio Brasileiro de Telecomunicações e Processamento de Sinais (SBrT2020), Florianopolis, Brazil, (2020), pp. 22–25.
[55] C. R. Rao, *Information and the accuracy attainable in the estimation of statistical parameters*, in Breakthroughs in Statistics: Foundations and basic theory, Springer, 1992, pp. 235–247.
[56] Y. Saad, *Iterative methods for sparse linear systems*, SIAM, 2003.
[57] ———, *Numerical methods for large eigenvalue problems: revised edition*, SIAM, 2011.
[58] R. Sawhney and K. Crane, *Monte Carlo geometry processing: A grid-free approach to PDE-based methods on volumetric domains*, ACM Transactions on Graphics, 39 (2020).
[59] S. H. Scheres, *RELION: implementation of a Bayesian approach to cryo-EM structure determination*, Journal of structural biology, 180 (2012), pp. 519–530.
[60] Sharp, Nicholas and Soliman, Yousuf and Crane, Keenan, *The vector heat method*, ACM Transactions on Graphics (TOG), 38 (2019), pp. 1–19.
[61] L. A. Shepp and B. F. Logan, *The Fourier reconstruction of a head section*, IEEE Transactions on nuclear science, 21 (1974), pp. 21–43.
[62] 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*, IEEE signal processing magazine, 30 (2013), pp. 83–98.
[63] A. Singer, *Angular synchronization by eigenvectors and semidefinite programming*, Applied and computational harmonic analysis, 30 (2011), pp. 20–36.
[64] A. Singer and H.-T. Wu, *Vector diffusion maps and the connection Laplacian*, Communications on pure and applied mathematics, 65 (2012), pp. 1067–1144.
[65] D. A. Spielman and N. Srivastava, *Graph sparsification by effective resistances*, in Proceedings of the

fortieth annual ACM symposium on Theory of computing, 2008, pp. 563–568.
[66] L. N. Trefethen and D. Bau, *Numerical linear algebra*, SIAM, 2022.
[67] N. Tremblay, P. Gonçalves, and P. Borgnat, *Design of graph filters and filterbanks*, in Cooperative and Graph Signal Processing, Elsevier, 2018, pp. 299–324.
[68] U. von Luxburg, *A tutorial on spectral clustering*, Stat. Comput., 17 (2007), pp. 395–416.
[69] D. B. Wilson, *Generating Random Spanning Trees More Quickly than the Cover Time*, in Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996, G. L. Miller, ed., ACM, 1996, pp. 296–303.
[70] S. X. Yu, *Angular embedding: From jarring intensity differences to perceived luminance*, in 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA, IEEE Computer Society, 2009, pp. 2302–2309.
[71] ———, *Angular Embedding: A Robust Quadratic Criterion*, IEEE Trans. Pattern Anal. Mach. Intell., 34 (2012), pp. 158–173.
[72] S. Zhang and Y. Huang, *Complex quadratic optimization and semidefinite programming*, SIAM Journal on Optimization, 16 (2006), pp. 871–890.
[73] X. Zhang, Y. He, N. Brugnone, M. Perlmutter, and M. J. Hirn, *MagNet: A Neural Network for Directed Graphs*, in Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual, M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan, eds., 2021, pp. 27003–27015.
[74] Z. Zhao and A. Singer, *Rotationally invariant image representation for viewing direction classification in cryo-EM*, Journal of structural biology, 186 (2014), pp. 153–166.

Supplementary Material

We work with a weighted graph $\mathcal{G}$ in all the following proofs, with $w_e \in \mathbf{R}_+$ the weight of edge e. We still denote by L_θ the resulting (weighted) connection Laplacian, with off-diagonal entries $(\mathsf{L}_\theta)_{i,j} = -w_{i,j}e^{\imath\theta_{(j,i)}}$ if $\{i,j\} \in \mathcal{E}$ ($(\mathsf{L}_\theta)_{i,j} = w_{(i,j)} = 0$ otherwise), and diagonal $(\mathsf{L}_\theta)_{i,i} = d_i = \sum_j w_{i,j}$ the weighted degree of node i.

6. Proof of Proposition 2.1. Denote by $C : \mathbf{C}^{\mathcal{V}} \to \mathbf{R}$ the cost function

$$C(f) = q\|f-g\|_2^2 + \langle f, \mathsf{L}_\theta f\rangle. \tag{6.1}$$

We use a standard argument from **CR**-calculus, and seek the zeros of the Fréchet Wirtinger derivatives of C (see *e.g.* [9]). Let us first compute these derivatives: for $f, h \in \mathbf{C}^{\mathcal{V}}$, we have

$$C(f+h) = q\langle (f+h)-g, (f+h)-g\rangle + \langle (f+h), \mathsf{L}_\theta(f+h)\rangle, \tag{6.2}$$

$$= C(f) + D_f^W C(h) + D_f^{W^*} C(h) + \left(q\langle h,h\rangle + \langle h, \mathsf{L}_\theta h\rangle\right), \tag{6.3}$$

where $(h \mapsto q\langle h,h\rangle + \langle h, \mathsf{L}_\theta h\rangle) \in o(h)$, and the associated (conjugate) Fréchet Wirtinger derivatives of C (at f) $D_f^W C$ and $D_f^{W^*} C : \mathbf{C}^{\mathcal{V}} \to \mathbf{C}$ are given by:

$$D_f^W C(h) = \langle q(f-g) + \mathsf{L}_\theta, h\rangle \tag{6.4}$$

$$D_f^{W^*} C(h) = \langle h, q(f-g) + \mathsf{L}_\theta\rangle. \tag{6.5}$$

The result follows from the fact that $D_f^W C = 0$ (resp. $D_f^{W^*} C = 0$) identically if and only if $f = q(\mathsf{L}_\theta + q\mathsf{I})^{-1} g$.

7. A Generalization of Property (P2). Define $\Delta_\theta = \mathsf{D}^{-1}\mathsf{L}_\theta$ the weighted connection-aware random walk Laplacian Δ_θ. Proposition 7.1 generalizes property (P2).

Proposition 7.1. *For all $l \geq 1$, we have*

$$(\mathsf{I} - \Delta_\theta)^l_{i,j} = \sum_{\substack{p \in P_i^j \\ l(p)=l}} \left(\prod_{0 \leq k < l} \frac{w_{(u_k,u_{k+1})}}{d_{u_k}} \right) \psi_{p^*}, \tag{7.1}$$

with $l(p)$ the length of a path $p = ((u_0,u_1),(u_1,u_2),...,(u_{l(p)-1},u_{l(p)}))$.

For each path p, the product in the r.h.s. of Equation (7.1) is the probability of observing p when performing a random walk from i to j on G. For the trivial connection this expression is equal to $(\mathsf{D}^{-1}\mathsf{A})^l_{i,j}$, the probability of a path going from i to j in l steps, but Proposition 7.1 more broadly captures *random propagations* along the corresponding sampled paths.
Observations similar to Proposition 7.1 have been laid out in a few works, such as [34] for continuous time random walk, and in a number of other situations, where the connection often stems from theoretical physics considerations [12].

Proof. The result is obvious for $l = 1$. We proceed by induction and assume the result true for some $l \in \mathbf{N}$. Then, recalling that $(\mathsf{I} - \Delta_\theta)_{v,j} = \frac{w_{v,j}}{d_v}$, we have:

$$(\mathsf{I} - \Delta_\theta)^{l+1}_{i,j} = \Big((\mathsf{I} - \Delta_\theta)^l (\mathsf{I} - \Delta_\theta)\Big)_{i,j} \tag{7.2}$$

$$= \sum_{v \in \mathcal{V}} \sum_{\substack{p \in P_i^v \\ l(p)=l}} \left(\prod_{0 \le k < l} \frac{w_{(u_k,u_{k+1})}}{d_{u_k}} \right) \frac{w_{v,j}}{d_v} \psi_{p^*} \psi_{(v,j)^*} \tag{7.3}$$

$$= \sum_{\substack{p \in P_i^j \\ l(p)=l+1}} \left(\prod_{0 \le k < l+1} \frac{w_{(u_k,u_{k+1})}}{d_{u_k}} \right) \psi_{p^*}. \tag{7.4}$$

■

Remark 7.2. *For $l = 0$, we have $(\mathsf{I} - \Delta_\theta)^l = \mathsf{I}$.*

8. Proof of Proposition 2.2. We will show an extension of Proposition 2.2 to heterogeneous values of q. That is, we associate to each node i some non-negative weight $q_i \in \mathbf{R}$ (at least one of them needs to be *positive*), that we record in the diagonal matrix Q with $\mathsf{Q}_{i,i} = q_i$. These q_i's and can be interpreted as edge weights in the extended graph $\mathcal{G}^\Gamma$, and we extend the definition of the measure ν_i over P_i^Γ to account for these weights, so that

$$\mathbf{P}_{\nu_i}(p) = \mathbf{1}_{\{i\}}(u_0) \mathbf{1}_{\{\Gamma\}}(u_l) \prod_{0 \le k < l} \left(\mathbf{1}_{\mathcal{V} \setminus \{\Gamma\}}(u_k) \frac{w_{(u_k,u_{k+1})}}{d_{u_k} + q_{u_k}} \right) \tag{8.1}$$

for a path $p = ((u_0, u_1), ..., (u_{l-1}, u_l))$ of length l, and $\mathbf{1}_S(x)$ the indicator that $x \in S$. We show the following Feynman-Kac formula, on weighted graphs and for heterogeneous q_i's.

Proposition 8.1. *For j the last node reached before absorption of a path p, we have*

$$\left((\mathsf{L}_\theta + \mathsf{Q})^{-1} \mathsf{Q} g \right)(i) = \mathbf{E}_{p \sim \nu_i} \left(\psi_{p^*_\Gamma(g_j)} \right). \tag{8.2}$$

Note that the l.h.s. is a more general smoothing operation than the solution to the Tikhonov problem, that can be leveraged in the presence of heteroscedastic noise (*e.g.* for the problem considered in Section 3.2 with different values of σ^2 for each node).

Proof. Consider the connection-aware random-walk Laplacian $\Delta_\theta^{\mathsf{Q}}$ associated to the extended graph $\mathcal{G}^\Gamma$. Its restriction to the rows and columns indexed by $\mathcal{V}$ reads $(\Delta_\theta^q)_\mathcal{V} = (\mathsf{D} + \mathsf{Q})^{-1}(\mathsf{L}_\theta + \mathsf{Q})$. It is clearly invertible, and all its eigenvalues lie in $(0, 2)$ (for instance by the Gershgorin circle theorem), hence the eigenvalues of $(\mathsf{I} - (\Delta_\theta^{\mathsf{Q}})_\mathcal{V})$ are in $(-1, 1)$, ensuring the convergence of the right-hand-side expression in:

$$\left((\Delta_\theta^q)_\mathcal{V}^{-1} \right)_{i,j} = \left[\sum_{l \ge 0} \left(\mathsf{I} - (\Delta_\theta^q)_\mathcal{V} \right)^l \right] \tag{8.3}$$

$$= \sum_{p \in P_i^j} \left(\prod_{0 \le k < l(p)} \frac{w_{(u_k,u_{k+1})}}{d_{u_k} + q_{u_k}} \right) \psi_{p^*}, \tag{8.4}$$

where Equation 8.4 follows from applying Proposition 7.1 to $\mathcal{G}^\Gamma$.
The result follows by right multiplication with $(\mathsf{D}+\mathsf{Q})^{-1}\mathsf{Q}$, where $\left((\mathsf{D}+\mathsf{Q})^{-1}\mathsf{Q}\right)_{j,j} = \frac{q_j}{d_{u_{l-1}}+q_j}$ accounts for the probability of the last transition from $u_{l-1}=j$ to Γ. ∎

9. Proof of Theorem 2.4. Before presenting the proof of our main Theorem, we state it for weighted graphs with heterogeneous q_i's.

Statement in the presence of weights. Consider the distribution $\mathcal{D}_\mathcal{M}$ over $\mathcal{M}(\mathcal{G})$ such that

$$\mathbf{P}_{\mathcal{D}_\mathcal{M}}(\phi) \propto \prod_{r\in\phi\cap\mathcal{V}} q_r \prod_{e\in\phi\cap\mathcal{E}} w_e \prod_{C\in\mathcal{C}(\phi)} \left((2-2\cos(\theta_C)\right), \tag{9.1}$$

and the estimator $\tilde{f}(i,\phi,g) = \psi_{r_\phi(i)\xrightarrow{\phi}i}(g(r_\phi(i)))$ propagating the value from $r_\phi(i)$ to i if i belongs to a rooted tree, and $\tilde{f}(i,\phi,g)=0$ if i lies in a unicycle. Then, we have:

Theorem 9.1.

$$\left((\mathsf{L}_\theta+\mathsf{Q})^{-1}\mathsf{Q}g\right)(i) = \mathbf{E}_{\mathcal{D}_\mathcal{M}}(\tilde{f}(i,\phi,g)). \tag{9.2}$$

We now proceed with the proof.

Determinantal Point Processes. The first step in our proof consists in re-stating the definition of $\mathcal{D}_\mathcal{M}$ as a Determinantal Point Process (DPP), which will provide us with powerful tools for reasoning about MTSFs. A (discrete) DPP over a finite set $\mathcal{X}$ associates a probability to each subset $X\subseteq\mathcal{X}$, and is defined by its marginal probabilities. It is parametrized by an Hermitian matrix $K\in M_{|\mathcal{X}|}(\mathbf{C})$, whose eigenvalues must all lie in $[0,1]$, known as the *marginal kernel* of the DPP, denoted $\mathrm{DPP}(K)$. Precisely, we say that $X\sim\mathrm{DPP}(K)$ if

$$\mathbf{P}_{\mathrm{DPP}(K)}(A\subseteq X) = \det(K)_{A,A} \tag{9.3}$$

for all $A\subseteq\mathcal{X}$, where $\det(K)_{R,C}$ is the minor of K restricted to the rows and columns indexed respectively by R and C. We write $\det(K)_{:,C}$ (resp. $\det(K)_{R,:}$) in case $R=\mathcal{X}$ (resp. $C=\mathcal{X}$).

Marginal kernel of $\mathcal{D}_\mathcal{M}$. Let us now describe the marginal kernel K that we will associate to distribution $\mathcal{D}_\mathcal{M}$. To this end, we consider a *twisted discrete differential* operator $\nabla:\mathbf{C}^\mathcal{V}\to\mathbf{C}^\mathcal{E}$, mapping complex values defined on the complex planes $\mathbf{C}_v$ (associated to the nodes of $\mathcal{G}$) to copies of the complex planes $\mathbf{C}_e$ associated to the *edges* of $\mathcal{G}$. Its expression relies on a splitting of the connection maps $\psi_e:\mathbf{C}_{s_e}\xrightarrow{\psi_{s_e,e}}\mathbf{C}_e\xrightarrow{\psi_{e,t_e}}\mathbf{C}_{t_e}$ such that $\psi_e=\psi_{e,t_e}\circ\psi_{s_e,e}$ (this is always possible, *e.g.* $\psi_{e,t_e}=\mathrm{id}_{\mathbf{C}_e,\mathbf{C}_{t_e}}$ and $\psi_{s_e,e}(z)=e^{\iota\theta_e}\cdot z$), and reads [35]:

$$(\nabla f)(e) = \sqrt{w_e}\psi_{t_e,e}(f(t_e)) - \sqrt{w_e}\psi_{s_e,e}(f(s_e)). \tag{9.4}$$

Expliciting the entries of the matrix of ∇, we have:

$$\nabla_{e,v} = \begin{cases} -\sqrt{w_e}\psi_{s_e,e} & \text{if } v=s_e \\ \sqrt{w_e}\psi_{e,t_e} & \text{if } v=t_e \\ 0 & \text{otherwise.} \end{cases} \tag{9.5}$$

We finally take $\mathcal{X} = \mathcal{V} \cup \mathcal{E}$ and define the marginal kernel K:

$$K = \nabla_{\mathsf{Q}} (\mathsf{L}_\theta + \mathsf{Q})^{-1} \nabla_Q^*, \text{ where } \nabla_{\mathsf{Q}} = \begin{bmatrix} \nabla \\ \sqrt{\mathsf{Q}} \end{bmatrix}. \tag{9.6}$$

Remark 9.2. *For trivial connections, ∇ is the edge-vertex incidence matrix of $\mathcal{G}$, and we have $\mathsf{L} = \nabla^t \nabla$. In a similar manner, one shows that $\mathsf{L}_\theta = \nabla^* \nabla$.*

We will show that $\mathcal{D}_{\mathcal{M}}$ is a DPP with kernel K. First, notice that

$$\mathsf{L}_\theta + \mathsf{Q} = \nabla_{\mathsf{Q}}^* \nabla_{\mathsf{Q}}, \tag{9.7}$$

so that K is a projection operator (*i.e.* $K^2 = K$), and all its eigenvalues are in $\{0, 1\}$. DPPs associated to such kernels are known as *projection DPPs*, and can be defined without resorting to marginal probabilities, with $\mathbf{P}_{\mathrm{DPP}(K)}(X) = \det(K)_{X,X}$ when $|X| = \mathrm{rk}(K)$ the rank of K ($\mathrm{rk}(K) = |\mathcal{V}|$ in our case), and $\mathbf{P}_{\mathrm{DPP}(K)}(X) = 0$ otherwise [29].
This will allow to show that the samples of DPP(K) are MTSFs, distributed according to $\mathcal{D}_{\mathcal{M}}$ (we generalize here an argument from [35]). Let $\phi \subseteq \mathcal{V} \cup \mathcal{E}$ with $|\phi| = |\mathcal{V}|$, and remark that:

$$\det(K)_{\phi,\phi} = \frac{\det(\nabla_{\mathsf{Q}}^* \nabla_{\mathsf{Q}})_{\phi,\phi}}{\det(\mathsf{L}_\theta + \mathsf{Q})}. \tag{9.8}$$

The next step then consists in expliciting $\det(\nabla_{\mathsf{Q}}^* \nabla_{\mathsf{Q}})_{\phi,\phi} = \det(\nabla_{\mathsf{Q}})_{\phi,:} \det(\nabla_{\mathsf{Q}}^*)_{:,\phi}$ depending on ϕ: we start by inspecting $\det(\nabla_{\mathsf{Q}}^*)_{:,\phi}$ (similar arguments apply to $\det(\nabla_{\mathsf{Q}})_{\phi,:}$). In this case, ϕ indexes the columns of ∇_{Q}, with two columns linearly independent if they do not belong to the same component c_ϕ. If c_ϕ spans m nodes, $\det(\nabla_{\mathsf{Q}}^*)_{:,c_\phi}$ can only be non-zero if $|c_\phi| = m$, so that c_ϕ must be either a unicycle or a rooted tree. We can then can compute $\det(\nabla_{\mathsf{Q}}^* \nabla_{\mathsf{Q}})_{c_\phi,c_\phi}$ from $\det(\nabla_{\mathsf{Q}}^*)_{:,\phi}$ and $\det(\nabla_{\mathsf{Q}})_{\phi,:}$ explicitly in those cases.

Lemma 9.3.

$$\det(\nabla_{\mathsf{Q}}^* \nabla_{\mathsf{Q}})_{c_\phi,c_\phi} = \begin{cases} (2 - 2\cos(\theta_C)) \prod_{e \in c_\phi} w_e & \text{if } c_\phi \text{ is a unicycle with cycle } C \\ q_r \prod_{e \in c_\phi \cap \mathcal{E}} w_e & \text{if } c_\phi \text{ is a tree rooted in } r \end{cases} \tag{9.9}$$

Proof. Suppose first that c_ϕ is a unicycle (this is the case treated in [35]). We fix an orientation of the edges of c_ϕ such that all edges are oriented *towards* the cycle C, and that the edges belonging to C form a directed cycle (there are only two such orientations, we choose one arbitrarily). The only non-zero permutations in the determinant

$$\det(\Delta_{\mathsf{Q}}^*)_{:,c_\phi} = \sum_{\sigma \in \mathcal{S}_m} \prod_{1 \leq i \leq m} \mathrm{sgn}(\sigma) (\Delta_{\mathsf{Q}}^*)_{i,\sigma(i)}, \tag{9.10}$$

where $\mathrm{sgn}(\sigma)$ denotes the signature of permutation (bijection from nodes to edges) σ, correspond to these orientations, and map vertices s_e to edges e ($\sigma(s_e) = e$). These two bijections,

denoted σ_C and σ_{C^*}, differ only on nodes adjacent to edges in C, and correspond to the two possible orientations of the cycle. We then obtain:

$$\det(\Delta^*_{\mathsf{Q}})_{:,c_\phi} = (-1)^m \mathrm{sgn}(\sigma_C) \prod_{e \in c_\phi \setminus C} \sqrt{w_e} \psi_{s_e,e} \left(\prod_{e' \in C} \sqrt{w_{e'}} \psi_{s_{e'},e'} - \prod_{e' \in C} \sqrt{w_{e'}} \psi_{t_{e'},e'} \right). \tag{9.11}$$

Performing a similar computation for $\det(\Delta_{\mathsf{Q}})_{c_\phi,:}$ and multiplying the two resulting expressions then allows to show that (this computation also appears in [35]):

$$\det(\nabla^*_{\mathsf{Q}} \nabla_{\mathsf{Q}})_{c_\phi,c_\phi} = (2 - 2\cos(\theta_C)) \prod_{e \in c_\phi} w_e, \tag{9.12}$$

where we used $\psi_e = \psi_{e,t_e} \circ \psi_{s_e,e}$ and $\psi_C + \psi^*_C = 2\cos(\theta_C)$.
Similarly, for c_ϕ a rooted tree with root $r \in c_\phi \cap \mathcal{V}$, we have

$$\det(\Delta^*_{\mathsf{Q}})_{:,c_\phi} = \sqrt{q_r} \times (-1)^m \mathrm{sgn}(\sigma_C) \prod_{e \in (c_\phi \cap]) \setminus C} \sqrt{w_e} \psi_{s_e,e}, \tag{9.13}$$

which results in

$$\det(\nabla_{\mathsf{Q}})_{c_\phi,c_\phi} = q_r \prod_{e \in c_\phi \cap \mathcal{E}} w_e. \tag{9.14}$$

■

As a corollary, we obtain

$$\mathbf{P}_{\mathrm{DPP}(K)}(\phi) = \frac{\prod_{r \in \phi \cap \mathcal{V}} q_r \prod_{e \in \phi \cap \mathcal{E}} w_e \prod_{C \in \mathcal{C}(\phi)} ((2 - 2\cos(\theta_C))}{\det(\mathsf{L}_\theta + \mathsf{Q})}, \tag{9.15}$$

which is exactly $\mathbf{P}_{\mathcal{D}_\mathcal{M}}(\phi)$.

Unbiased estimator. It remains to show that $\tilde{f}(i, \phi, g)$ is an unbiased estimator of the desired quantity $\left((\mathsf{L}_\theta + \mathsf{Q})^{-1} \mathsf{Q} g\right)(i)$. Our argument is inspired from that of [51], and we will require another determinantal tool.

Proposition 9.4 (Cauchy-Binet formula). *For two matrices $A \in M_{m,n}(\mathbf{C})$ and $B \in M_{n,m}(\mathbf{C})$ with $m < n$, we have:*

$$\det(AB) = \sum_T \det(A)_{:,T} \det(B)_{T,:}, \tag{9.16}$$

where T ranges over all subsets of $\{1, ..., n\}$ of size m.

We begin by rewriting $(\mathsf{L}_\theta + \mathsf{Q})^{-1}_{i,j}$ as an expectation, starting from Cramer's rule

$$(\mathsf{L}_\theta + \mathsf{Q})^{-1}_{i,j} = (-1)^{i+j} \frac{\det(\mathsf{L}_\theta + \mathsf{Q})_{\mathcal{V} \setminus \{j\}, \mathcal{V} \setminus \{i\}}}{\det(\mathsf{L}_\theta + \mathsf{Q})}. \tag{9.17}$$

We can rewrite the numerator using the Cauchy-Binet formula:

$$\det(\mathsf{L}_\theta + \mathsf{Q})_{\mathcal{V} \setminus \{j\}, \mathcal{V} \setminus \{i\}} = \sum_{\substack{\phi \subseteq \mathcal{V} \cup \mathcal{E} \\ |\phi| = |\mathcal{V}| - 1}} (-1)^{i+j} \det\left(\left[(\nabla^*_{\mathsf{Q}})_{:,\phi} \ \delta_j \right] \right) \det\left(\begin{bmatrix} (\nabla_{\mathsf{Q}})_{\phi,:} \\ \delta_i \end{bmatrix} \right), \tag{9.18}$$

where δ_i is the i-th vector in the usual basis of $\mathbf{R}^{\mathcal{V}}$. An argument analogous to Lemma 9.3 then shows that the product of determinants

$$\det\left(\left[(\nabla_{\mathsf{Q}}^*)_{:,\phi}\ \delta_j\right]\right)\det\left(\begin{bmatrix}(\nabla_{\mathsf{Q}})_{\phi,:}\\ \delta_i\end{bmatrix}\right) \tag{9.19}$$

can only be non-zero if ϕ contains a tree $T_i^j \subseteq \mathcal{E}$ spanning both i and j (with no root), with contribution $\psi_{j\xrightarrow{\phi}i}\prod_{e\in T_i^j} w_e$ to the product, and if *all* the other components are rooted trees or unicycles, with associated contributions described in 9.3. We thus obtain

$$(\mathsf{L}_\theta+\mathsf{Q})^{-1}_{i,j} = \frac{1}{q_j}\sum_{\phi\in\mathcal{M}(\mathcal{G})}\mathbf{P}_{\phi\sim\mathcal{D}_{\mathcal{M}}}(\phi)\mathbf{1}_{c_\phi(j)}(i)\psi_{j\xrightarrow{\phi}i} \tag{9.20}$$

$$= \frac{1}{q_j}\mathbf{E}_{\phi\sim\mathcal{D}_{\mathcal{M}}}\left(\mathbf{1}_{c_\phi(j)}(i)\psi_{j\xrightarrow{\phi}i}\right), \tag{9.21}$$

where $\mathbf{1}_{c_\phi(j)}(i)$ is the indicator that i belongs to the set of nodes $c_\phi(j)$ spanned by the tree rooted in r. Finally, we have

$$\left((\mathsf{L}_\theta+\mathsf{Q})^{-1}\mathsf{Q}g\right)(i) = \langle\delta_i,(\mathsf{L}_\theta+\mathsf{Q})^{-1}\mathsf{Q}g\rangle \tag{9.22}$$

$$= \sum_{j\in\mathcal{V}} q_j(\mathsf{L}_\theta+\mathsf{Q})^{-1}_{i,j}g(j) \tag{9.23}$$

$$= \sum_{j\in\mathcal{V}}\mathbf{E}_{\phi\sim\mathcal{D}_{\mathcal{M}}}\left(\psi_{j\xrightarrow{\phi}i}(g(j))\,\mathbf{1}_{c_\phi(j)}(i)\right) \tag{9.24}$$

$$= \mathbf{E}_{\phi\sim\mathcal{D}_{\mathcal{M}}}(\tilde{f}(i,\phi,g)). \tag{9.25}$$

■

10. Complexity Analysis. We will describe the complexity of different implementations of the MTSF-sampling algorithm. We recall the MTSF-sampling procedure in Algorithm 10.1. We consider here the generalization to weighted graphs with heterogeneous q_i's, and use a generalized random_successor(u) that outputs either Γ with probability $\frac{q_u}{d_u+q_u}$, or some node v with probability $\frac{w_{u,v}}{d_u+q_u}$. This generalized algorithms provably samples MTSFs from the distribution $\mathcal{D}_{\mathcal{M}}$, with the argument from [22] still applying.

10.1. Number of Steps. Let us re-state the upper bound on the expected number of steps of the random walk in the sampling Algorithm 2.3. Denote by T_ϕ the number of random neighbors sampled to build $\phi\in\mathcal{M}(\mathcal{G})$ during an execution of Algorithm 10.1. Then, we have:

Proposition 10.1.

$$\mathbf{E}_{\phi\sim\mathcal{D}_{\mathcal{M}}}(T_\phi) \leq \mathrm{tr}\left((\mathsf{L}+\mathsf{Q})^{-1}(\mathsf{D}+\mathsf{Q})\right), \tag{10.1}$$

Algorithm 10.1 MTSF sampling algorithm [22].

1: $\phi \leftarrow \emptyset$
2: **while** ϕ not spanning **do**
3: Let $i \in \mathcal{V}$ be the first node in the queue not spanned by ϕ
4: $u \leftarrow i$ ▷ u is the current node of the random walk
5: $p \leftarrow \epsilon$ ▷ ϵ the empty path
6: **while** $(u \neq \Gamma)$ and (p does not intersect ϕ or contain a cycle) **do**
7: $u' \leftarrow$ random_successor(u) ▷ Move to next node
8: **if** $u' \neq \Gamma$ **then**
9: $e \leftarrow (u, u')$, $p \leftarrow pe$ ▷ Add e to the path p
10: **end if**
11: **if** p contains a cycle C **then**
12: Remove this cycle from p with probability $\cos(\theta_C)$
13: **end if**
14: $u_{\text{old}} \leftarrow u$, $u \leftarrow u'$ ▷ u_{old} the previous node
15: **end while**
16: **if** $u = \Gamma$ **then**
17: $\phi \leftarrow \phi \cup p \cup u_{\text{old}}$ ▷ Add the sampled path p and the root u_{old} to ϕ
18: **else**
19: $\phi \leftarrow \phi \cup p$
20: **end if**
21: **end while**
22: **Output** ϕ

where we abuse notation and write $\phi \sim \mathcal{D}_{\mathcal{M}}$ for ϕ sampled using Algorithm 10.1. This bound is easily obtained by considering the special case of the trivial connection with $\psi_e = \mathrm{id}_{\mathbf{C}_{s_e}, \mathbf{C}_{t_e}}$ for all $e \in \overrightarrow{\mathcal{E}}$, in which case Algorithm 10.1 samples spanning forests of $\mathcal{G}$ from the distribution mentioned in Remark 2.5, and the expected number of steps is known to be [51]:

$$\operatorname{tr}\left((\mathsf{L}+\mathsf{Q})^{-1}(\mathsf{D}+\mathsf{Q})\right). \tag{10.2}$$

This is also the slowest scenario: in this case, $\theta_C = 0$ for all cycles, and p can never contain a cycle in step 6 of Algorithm 10.1. Hence, the only way to exit the **while** loop is to reach Γ or to intersect ϕ, and we obtain the bound of Proposition 10.1 as a consequence.

Let us briefly comment on the value of the trace in Equation 10.1. We have:

$$\operatorname{tr}\left((\mathsf{L}+\mathsf{Q})^{-1}(\mathsf{D}+\mathsf{Q})\right) = \sum_{i \in \mathcal{V}} (\mathsf{L}+\mathsf{Q})^{-1}_{i,i}(q_i + d_i), \tag{10.3}$$

where it is known that $(\mathsf{L}+\mathsf{Q})^{-1}_{i,i}$ equals the probability that $i \in \mathsf{f} \cap \mathcal{V}$ for $\mathsf{f} \sim \mathcal{D}_{\mathcal{F}}$, and hence lies in $[0,1]$. It follows that $\mathbf{E}_{\phi \sim \mathcal{D}_{\mathcal{M}}}(T_\phi)$ is in $\mathcal{O}\left(\frac{|\mathcal{E}|}{q_{\min}}\right)$.

Remark 10.2. *Algorithm 10.1 can be used in conjunction with Equation 9.2 to estimate* $q(\widetilde{\mathsf{L}_\theta} + q\mathsf{I})^{-1}g$ *(see Remark 4.1). In this context, we take* $\mathsf{Q} = q\mathsf{D}$*, and obtain the bound* $\mathbf{E}_{\phi \sim \mathcal{D}_{\mathcal{M}}}(T_\phi) \in \mathcal{O}\left(\left(1 + \frac{1}{q}\right)|\mathcal{V}|\right)$.

10.2. Implementation-specific Complexity. The time complexity of Algorithm 10.1 does not only depend on the number of steps taken by the random walk, but also on the cost of the cycle-detection (step 10). We propose two counter-based solutions.

One-counter detection. The first strategy consists in dynamically assigning a numerical value $\mathrm{c}_v \in \mathbf{N}$ to each node $v \in \mathcal{V}$ encountered during the random walk (initialized to $\mathrm{c}_v = 0$), based on a global counter $\mathrm{c} \in \mathbf{N}$. This counter is incremented each time a new random walk begins (instruction at line 3), and we set $\mathrm{c}_{u'} = \mathrm{c}$ whenever the random walk reaches u' on the instruction at line 7 (if it not already spanned by MTSF). If a cycle C is created in node u' and discarded, we reset $\mathrm{c}_v = 0$ for all nodes v spanned by C *except* u'. A cycle can then be detected in $\mathcal{O}(1)$ time by checking the value of $\mathrm{c}_{u'}$ at each step of the random walk: there is a cycle if $\mathrm{c}_{u'} = \mathrm{c}$, and no cycle if $\mathrm{c}_{u'} < \mathrm{c}$.
In a run of Algorithm 10.1, resetting the values c_v when cycles are discarded requires going through at most T_ϕ nodes, which results in $\mathcal{O}\left(\frac{|\mathcal{E}|}{q_{\min}}\right)$ overall expected *time complexity.*

In practice, we found it faster to use an implementation of the following strategy.

Multiple-counters detection. The c_v's need to be reset in the one-counter cycle-detection strategy because, whenever a cycle is discarded, nodes spanned by this cycle should no longer be remembered as spanned by the path p. This expensive resetting step can be bypassed by considering *multiple* counters, in addition to the c's (indexed by a global c). We will keep track of couples of values $(\mathrm{id}_v, \mathrm{val}_v) \in \mathbf{N}^2$ for each v, initialized to $(0, 0)$) and with updates based on two global counters $\mathrm{id}, \mathrm{val} \in \mathbf{N}^2$, and of values $\mathrm{cap}(\mathrm{id})$ associated to each id. id_v should be thought of as the ID of a counter at v (amongst multiple others), and val_v as its value (we only need to store the value associated to the largest id_v). These counters are reset to 0 whenever c is incremented (and a new random walk is initiated), and are otherwise updated by applying the following rules for all nodes u' reached by the random walk (if not already spanned by the MTSF).

- If no cycle is created and discarded at u', set $\mathrm{id}_{u'} = \mathrm{id}$, $\mathrm{val}_{u'} = \mathrm{val}$, and increment val.
- If a cycle is created at node u' and discarded, store the value $\mathrm{cap}(\mathrm{id}) = \mathrm{val}$ (the maximum value for the id^{th} counter), set $\mathrm{cap}(k) = 0$ for all $k > \mathrm{id}_{u'}$, $\mathrm{val}_{u'} = \mathrm{val}$, and increment id.

Using these counters, a cycle is detected at u' if $\mathrm{val}_{u'} \leq \mathrm{cap}(\mathrm{id}_{u'})$.
The number id of counters used is unbounded, and we did not derive an expected time complexity for this multiple-counters strategy, but consistently obtained better performance in our measurements when using it.

Remark 10.3. *The $\mathcal{O}\left(\frac{|\mathcal{E}|}{q_{\min}}\right)$ complexity can seem daunting for very small values of q_i's, but one should remember that this is only an upper bound, not accounting for the presence of unicycles in the sample. As the $q_i \to 0$, $\mathrm{DPP}(K)$ approaches a distribution over spanning forests of unicycles, with a Wilson-like sampling algorithm described in [33]. Our implementation strategy can also be applied in this case and translates to the bounds on the number of steps of the random walks they derive (see also [22]).*

Finally, we point out another implementation detail.

Remark 10.4. *The propagation maps used in Algorithm 2.2 can be computed during the*

sampling process and, one actually only needs knowledge of: the root of each tree, whether or not a node i belongs to a rooted tree and, if it does, the propagation map from the root $r_\phi(i)$ to i. The actual MTSF ϕ is never used in the estimation.

11. Variance Reduction. We show that the estimators in Equations 2.10 and 2.12 are unbiased (Proposition 2.11).

Rao-Blackwell estimator. Let us first re-define $\overline{f}$ for heterogeneous values of q. The difference resides in the aggregation function h_ϕ:

$$h_\phi(r, g) = \frac{\sum_{j \in c_\phi(r)} \psi_{j \xrightarrow{\phi} r} g(j)}{\sum_{r' \in c_\phi(r)} q_{r'}}, \tag{11.1}$$

where we recall that $c_\phi(r)$ denotes the set of nodes spanned by the tree containing r. We still take $\overline{f}(i, \phi, g) = \psi_{r_\phi(i) \xrightarrow{\psi} i}(h_\phi(r_\phi(i), g))$, and show that:

Proposition 11.1. $\mathbf{E}_{\phi \sim \mathcal{D}_\mathcal{M}}(\overline{f}(i, \phi, g)) = f_*(i)$.

Proof. We will express $\overline{f}$ as a conditional expectation, the result will follow from the law of total expectation. Here, we choose a connected subset of edges $\pi \subseteq \mathcal{E}$, and condition on $\phi \cap \mathcal{E}$ containing π as one of its (maximal) components, which we denote by $\pi \sqsubseteq_\mathcal{E} \phi$. For i belonging to the connected component spanned by π, we have:

$$\mathbf{E}_{\phi \sim \mathcal{D}_\mathcal{M}}(\tilde{f}(i, \phi, g) \mid \pi \sqsubseteq_\mathcal{E} \phi) = \sum_{\phi \in \mathcal{M}(\mathcal{G})} \mathbf{P}_{\mathcal{D}_\mathcal{M}}(\phi \mid \pi \sqsubseteq_\mathcal{E} \phi)\ \tilde{f}(i, \phi, g) \tag{11.2}$$

$$= \sum_{\substack{\phi \in \mathcal{M}(\mathcal{G}) \\ \pi \sqsubseteq_\mathcal{E} \phi}} \frac{q_r \mathbf{1}_{\phi \cap c_\phi(i)}(r)}{\sum_{r' \in c_\phi(i)} q_{r'}}\ \tilde{f}(i, \phi, g) \tag{11.3}$$

$$= \overline{f}(i, \phi, g). \tag{11.4}$$

■

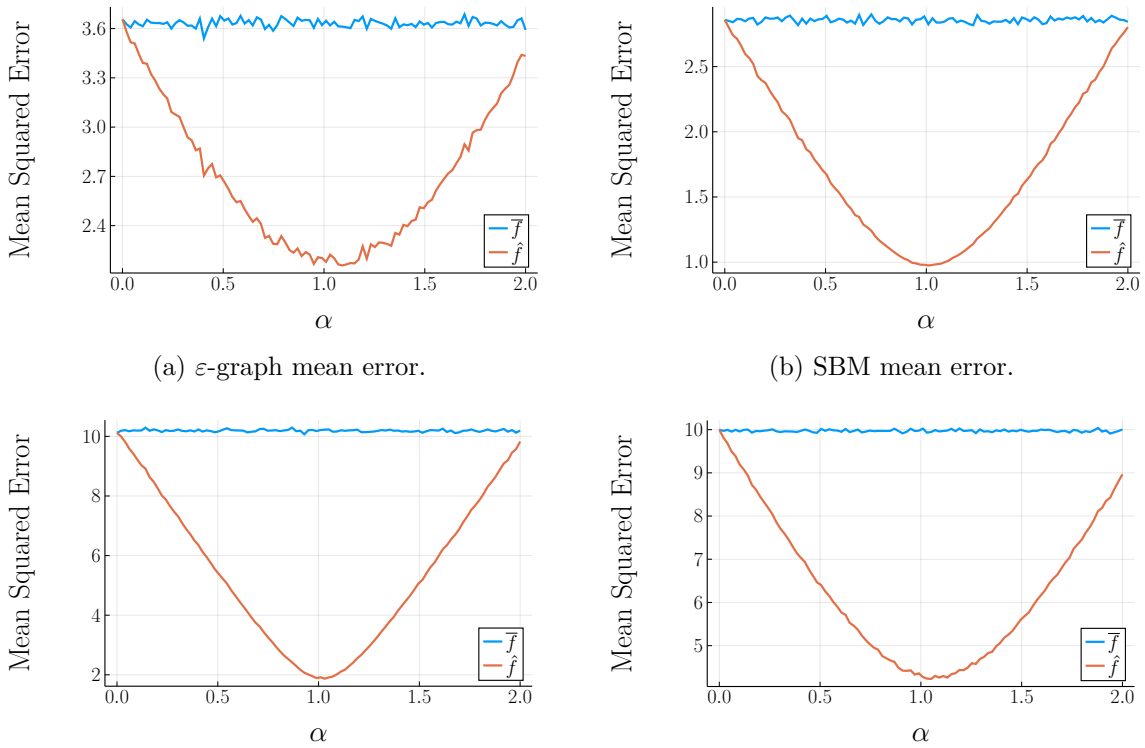
Gradient step as control variates. We show that $\mathbf{E}_{\phi \sim \mathcal{D}_\mathcal{M}}(\hat{f}(\phi, g)) = f_*$. First recall that $\mathbf{E}_{\mathcal{D}_\mathcal{M}}(\overline{f}) = f_*$ (and $f_* = q(\mathsf{L}_\theta + q\mathsf{I})^{-1} g$) so that, by linearity of the expectation:

$$\mathbf{E}_{\phi \sim \mathcal{D}_\mathcal{M}}(\hat{f}(\phi, g)) = f_* - \alpha \mathsf{P}(g - g) = f_* \tag{11.5}$$

where, for heterogeneous q_i's, we set $\mathsf{P} = (\mathsf{Q}^{-1}\mathsf{D} + \mathsf{I})^{-1}$.

This shows that $\hat{f}$ is unbiased. Let us now discuss the choice of α. We plot in Figure 11.1 the mean errors $\|\overline{f} - f_*\|$ and $\|\hat{f} - f_*\|$ over 5 trials, for different values values of α. We always take $m = 20$ MTSFs, and set $q = 10^{-2} \times \overline{d}$. We use the random graph models from Section 3.

The extent of the error reduction depends on the graph, but the optimal choice of α seems to always be close to 1. This behavior can be observed across order-of-magnitude variations of q (not shown), and setting $\alpha = 1$ resulted in meaningful variance reduction for our other experiments (*e.g.* Section 3). This contrasts with the (similar) variance reduction technique proposed in [52], developed for connection-free graphs, for which a good choice of step-size was less straightforward.



(a) ε-graph mean error.

(b) SBM mean error.

(c) DC-SBM 1 mean error.

(d) AS CAIDA mean error.

Figure 11.1: Mean error as a function of α.

12. Supporting Arguments for Tikhonov Smoothing. The goal of this Section is to argue that, when the connection is sufficiently consistent, eigenvectors of L and L_θ can be used similarly in graph-signal-processing applications, such as the smoothing experiment in Section 3.

Coherent connection. Let us first consider a perfectly coherent connection, such that $\theta_{i,j} = \omega_j - \omega_i$ for all edges $\{i, j\}$. Consider the eigendecomposition $\mathsf{L} = \mathsf{U}\Lambda\mathsf{U}^*$ of L, and denote respectively by x the vector with entries $x_i = e^{-\imath\omega_i}$, and D_x the diagonal matrix with $(\mathsf{D}_x)_{i,i} = x_i$. In this situation, we have

$$\mathsf{L}_\theta = \mathsf{D}_x \mathsf{L} \mathsf{D}_x^* = (\mathsf{D}_x \mathsf{U}) \Lambda (\mathsf{D}_x \mathsf{U})^*, \tag{12.1}$$

and the eigenvectors v_i of L_θ are given by $\mathsf{D}_x u_i$, with u_i the eigenvectors of L.

Noisy connection. Equation 12.1 no longer holds for incoherent connections. We provide illustrations of the resulting eigenvectors of L_θ for connections of the form:

$$\theta_{i,j} = \omega_j - \omega_i + \eta\varepsilon_{i,j}, \tag{12.2}$$

for different values of η, and $\varepsilon_{i,j}$ uniformly distributed in $[-1,1]$ (like in Section 3). Specifically, we represent u_i, v_i, and $\mathsf{D}_x v_i$ (which should approximate u_i). We also measure the difference between eigenvectors u_i and v_i as a function of η using the normalized error

$$\min_{r \in U(\mathbf{C})} \frac{\|u_i - r v_i\|_2}{n}, \tag{12.3}$$

and use the following graphs:

- An ε-graph built from $n = 100$ points uniformly sampled in $[0,1]^3$, with $\varepsilon = 0.3$ (Figure 12.1).
- A SBM with two communities on size 50 ($n = 100$), with $c_{1,1} = c_{2,2} = 19$ and $c_{1,2} = 1$ (Figure 12.2).

We only use a single realization of these graphs, equipped with a single realization of the connection. The normalized errors are computed across a linear range of values of η in $[0,1]$. For each graph, we plot the real and imaginary parts of the entries of the corresponding eigenvectors, for $\eta_1 \simeq 0.11$ and $\eta_2 \simeq 0.75$. The corresponding errors are also highlighted, along with the value $\eta_0 = \frac{\pi}{2n}$ (for which we know that the connection always satisfies the weak-inconsistency Condition 2.6). Recall that the eigenvectors of L_θ are only defined up to a global rotation, and this is apparent in our illustrations.

Finally, we provide similar illustrations for a cyclic graph on size $n = 100$, in a more controlled setup: the values of $\varepsilon_{i,j}$ are no longer randomly sampled, and we instead set $\varepsilon_e = 1$ for all e along a *coherent* orientation. This way, η_0 corresponds exactly to the weak-inconsistency threshold in Condition 2.6. We plot the corresponding normalized errors in Figure 12.3.

Discussion. We observe in Figure 12.1 that $\mathsf{D}_x^* v_i$ provides a good approximation of u_i for low levels of noise ($\eta = \eta_1$), but this is no longer the case at higher noise-levels ($\eta = \eta_2$): $\mathsf{D}_x^* v_2$ somewhat recovers the shape of u_2, but $\mathsf{D}_x^* v_3$ appears completely unrelated to u_3. Results for the SBM (Figure 12.2) suggest that major structural properties of the graph (here, the community structure) are still captured even for high noise: for both $\eta = \eta_1$ and $\eta = \eta_2$, $\mathsf{D}_x^* v_2$ clearly partition the graph into the same communities as u_i.

These behaviors are reflected in the normalized-error plots (Figure 12.1), with greater error associated to higher values of η, and smooth decay. Results on the cycle-graph (Figure 12.3) are very different, with three different behaviors (similar for both eigenvectors), occurring at sharp thresholds: perfect correspondence with u_i at $\eta = 0$, two small, low-error, *plateau* (one of them occurring as soon as $\eta > 0$, and containing $\eta = \eta_0$), and a higher, essentially constant, error otherwise. For the SBM (not shown), the error decayed linearly with η.

These illustrations provide weak evidence that eigenvectors of L and L_θ behave similarly in low-noise regimes. In this spirit, simple instances (*e.g.* $\omega_v = 0$ for all v but non-trivial noise) may be amenable to perturbative analyses, with L_θ understood as an analytic perturbation of L (this is similar to the setting mentioned in Remark 1.1). The situation is much more nuanced for strongly-incoherent connections, and likely requires more involved mathematical descriptions.

13. Towards Extensions: an Alternative Construction. We exhibit how the loop-erasure procedure in Algorithm 2.3 can be understood as stemming from the Feynman-Kac formula in Proposition 2.2. The aim is twofold: first, to better understand the relation between the

Feynman-Kac formula in Proposition 2.2 and the MTSF-based estimaton of Theorem 2.4 and, second, to propose a possible proof outline for generalizations of our work. We are not aware of a similar observation appearing in the literature and, as such, the argument may also be of independent interest. For notational brevity, we consider here the simple case of paths sampled from ν_i containing a single cycle, the reasoning is straightforwardly extended to paths with multiple cycles at the price of heavier notations.

Loop-erasure from Feynman-Kac. Define the equivalence relation $p \simeq q$ on P_i^Γ for paths in the extended graph $\mathcal{G}^\Gamma$ that differ by the orientation of at most *one* cycle (denoted $P_i^\Gamma(1)$): that is, $p \simeq q$ if $p = aCb$ and $q = aC^*b$, with C a (possibly empty) cycle. For convenience, we will denote the two elements of each of the induced equivalence classes as $p(C)$ and $p(C^*)$, with $\overline{p(C)}$ denoting the equivalence class itself. For node $i \in \mathcal{V}$, we then have:

$$\sum_{p \in P_i^\Gamma(1)} \mathbf{P}_{\nu_i}(p)\psi_{p_\Gamma^*} = \sum_{\overline{p(C)} \in P_i^\Gamma(1)/\simeq} \left(\mathbf{P}_{\nu_i}(p(C))\psi_{p(C)_\Gamma^*} + \mathbf{P}_{\nu_i}(p(C^*))\psi_{p(C^*)_\Gamma^*}\right) \tag{13.1}$$

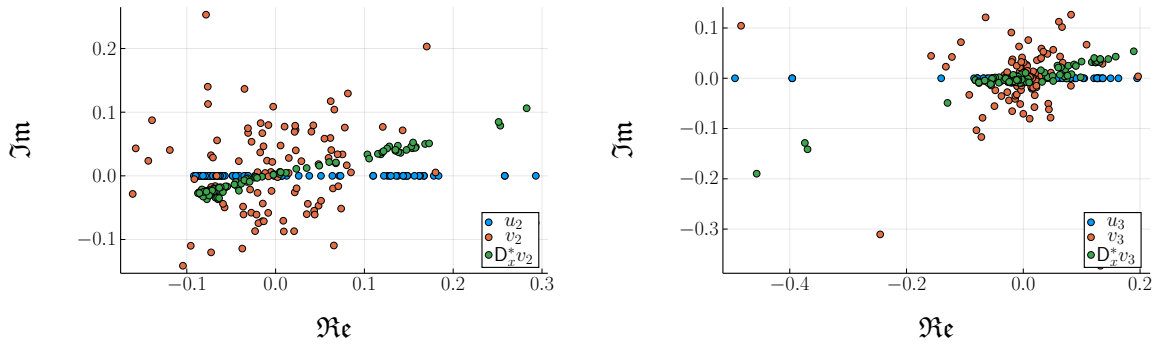
$$= \sum_{\overline{p(C)} \in P_i^\Gamma(1)/\simeq} \mathbf{P}_{\nu_i}(\overline{p(C)})\left(\psi_{a^*} \circ (\psi_C + \psi_{C^*}) \circ \psi_{b^*}\right) \tag{13.2}$$

$$= \sum_{\overline{p(C)} \in P_i^\Gamma(1)/\simeq} 2\mathbf{P}_{\nu_i}(\overline{p_C})\left(\left(\psi_{a^*} \circ \left(\frac{\psi_C + \psi_{C^*}}{2}\right)\right) \circ \psi_{b^*}\right), \tag{13.3}$$

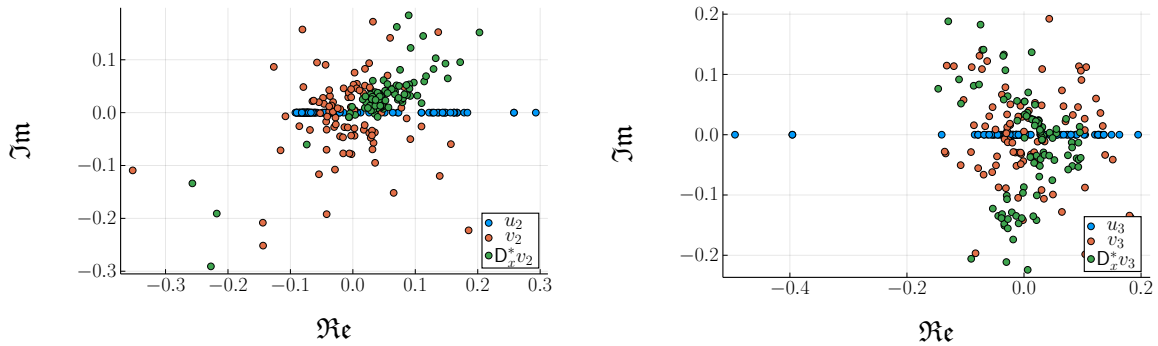
where $p_C = aCb$, and $\mathbf{P}_{\nu_i}(\overline{p_C}) = \mathbf{P}_{\nu_i}(p_C)$ in Equation 13.2 (we use $\mathbf{P}_{\nu_i}(p_C) = \mathbf{P}_{\nu_i}(p_{C^*})$). Let us rephrase this observation: whereas propagation along a cycle C in Proposition 2.2 depends on its orientation, *in expectation* this dependency disappears when propagating not by ψ_C but by $\frac{\psi_C+\psi_{C^*}}{2}$, the factor 2 in Equation 13.3 in turn accounting for the possibility of sampling either p_C or p_{C^*}. Loop-erasures appear when probabilistically interpreting the maps $\frac{\psi_C+\psi_{C^*}}{2}$, acting by multiplication with $\cos(\theta_C)$, which, under Condition 2.6, defines a Bernoulli trial for cycle acceptance.

Extending this argument to general paths (with more than one cycle) provides an alternative proof that propagating along the first branch sampled in Algorithm 2.3 (starting from node i) results in an unbiased estimator of $f_*(i)$. As a corollary of the same analysis, if $\cos(\theta_C) < 0$ for some cycle C encountered in path p, one can perform a Bernoulli trial with success probability $-\cos(\theta_C) = (-1) \times \cos(\theta_C)$, and multiply the resulting estimate -1 each time such a cycle is constructed in path p, extending the estimation to *any* connection.

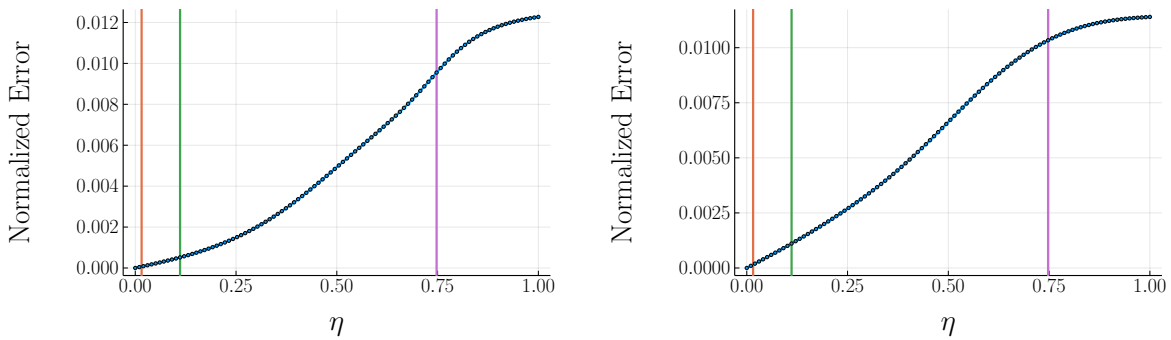
Now, the question is: can such a construction be extended to *all* nodes in $\mathcal{G}$, resulting in unbiased MTSF-based estimators free from the weak-inconsistency Condition 2.3? Also, could similar observations be leveraged to generalize these estimators to higher-dimensional connections (see Remark 2.3)?



(a) $\eta_1 \simeq 0.11$



(b) $\eta_2 \simeq 0.75$



(c) Normalized error as a function of η. The colored vertical lines represent the values of η_0 (orange), η_1 (green) and η_2 (purple).

Figure 12.1: Results for the ε-graph. Left: second eigenvector. Right: third eigenvector.

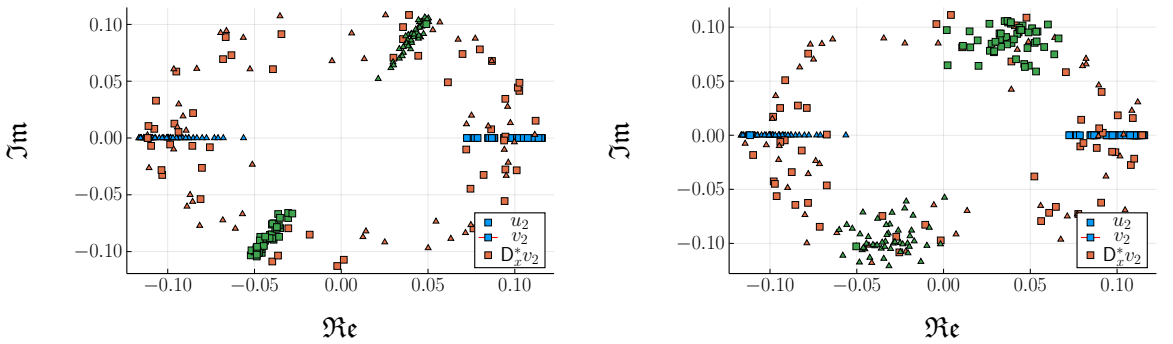


Figure 12.2: Second eigenvector for the SBM-graph. The two communities are depicted as triangles and squares respectively. Left: $\eta_1 \simeq 0.11$. Right: $\eta_2 \simeq 0.75$.

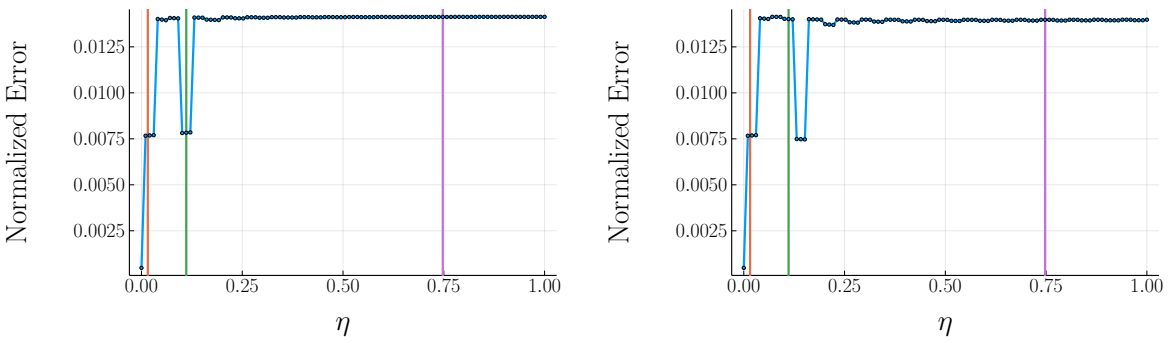


Figure 12.3: Normalized errors for the cycle-graph. Left: second eigenvector. Right: third eigenvector. Vertical lines represent η_0 (orange), η_1 (green) and η_2 respectively (purple).