Variance Reduction for Inverse Trace Estimation via Random Spanning Forests

Yusuf Yiğit PİLAVCI, Pierre-Olivier Amblard, Simon Barthelmé and Nicolas Tremblay*

Univ. Grenoble Alpes, CNRS, Grenoble INP, GIPSA-Lab

Abstract. The trace $\mathrm{tr}(q(\mathsf{L}+q\mathsf{I})^{-1})$, where L is a symmetric diagonally dominant matrix, is the quantity of interest in some machine learning problems. However, its direct computation is impractical if the matrix size is large. State-of-the-art methods include Hutchinson's estimator combined with iterative solvers, as well as the estimator based on random spanning forests (a random process on graphs). In this work, we show two ways of improving the forest-based estimator via well-known variance reduction techniques, namely control variates and stratified sampling. Implementing these techniques is easy, and provides substantial variance reduction, yielding comparable or better performance relative to state-of-the-art algorithms.

Keywords: trace estimation, graph signal processing, sampling, random spanning forests

1 Introduction

Randomized methods are useful to approximate the trace of a matrix if the matrix is not explicitly known. These methods come into play in various problems [21] in which $\mathsf{A} \in \mathbb{R}^{n\times n}$ is typically a large matrix (*e.g.* $n \geq 10^6$) and $\mathrm{tr}(f(\mathsf{A}))$ is the quantity of interest. In this work, we focus on calculating the trace of $f(\mathsf{L}) = q(\mathsf{L}+q\mathsf{I})^{-1}$ without taking the matrix inverse when L is a symmetric diagonally dominant (SDD) matrix *i.e.* $\forall i, |\mathsf{L}_{i,i}| \geq \sum_{j\neq i} |\mathsf{L}_{i,j}|$. A natural use case of $\mathrm{tr}(q(\mathsf{L}+q\mathsf{I})^{-1})$ arises in graph Tikhonov regularization problem [17] where L is the graph Laplacian. In this problem, we are given a noisy signal over n vertices $\mathbf{y} = [y_1, y_2, \ldots, y_n]^\top$ and we aim to recover the original signal $\mathbf{x}$ by solving the following problem:

$$\hat{\mathbf{x}} = \underset{\mathbf{z}\in\mathbb{R}^n}{\operatorname{argmin}}\, q||\mathbf{y}-\mathbf{z}||_2^2 + \mathbf{z}^\top \mathsf{L}\mathbf{z}, \quad q>0 \tag{1}$$

* This work was partly funded by the French National Research Agency in the framework of the "Investissements d'avenir" program (ANR-15-IDEX-02), the LabEx PERSYVAL (ANR-11-LABX-0025-01), the ANR GraVa (ANR-18-CE40-0005), the ANR GRANOLA (ANR-21-CE48-0009), the MIAI@Grenoble Alpes chairs "LargeDATA at UGA" and "Pollutants" (ANR-19-P3IA-0003).

where the hyper-parameter $q > 0$ controls the regularization. The explicit solution $\hat{\mathbf{x}}$ reads $\mathsf{K}\mathbf{y}$ where $\mathsf{K} = q(\mathsf{L} + q\mathsf{I})^{-1}$. Notice that the recovery error, *i.e.* $||\mathbf{x} - \hat{\mathbf{x}}||_2^2$, highly depends on q and there are several methods to automatically choose the value of q such that the solution $\hat{\mathbf{x}}$ approaches to $\mathbf{x}$. Many of them, such as generalized cross-validation (GCV), Akaike or Bayesian information criteria, use the measurements $\mathbf{y}$ and $\mathrm{tr}(\mathsf{K})$ as a measure of the degrees of freedom of the linear smoother K [7]. For example, GCV computes the following score for evaluating different choices of q:

$$GCV(q) = \frac{1}{N}\left(\sum_{i=1}^{n} \frac{y_i - \hat{x}_i}{1 - \left(\mathrm{tr}(\mathsf{K})/n\right)}\right).$$

As in this example, along with the solution $\hat{\mathbf{x}}$, one needs to access the diagonal entries of K for computing the trace.

State-of-the-art. The standard estimator for $\mathrm{tr}(\mathsf{K})$ is due to Hutchinson [9]. Given N samples of a Bernoulli random vector $\mathbf{a} \in \{1, -1\}^n$ with $\forall i$, $\mathbb{P}(a_i = \pm 1) = 1/2$, Hutchinson's estimator is defined as $h \coloneqq \frac{1}{N}\sum_{i=1}^{N} {\mathbf{a}^{(i)}}^\top \mathsf{K}\mathbf{a}^{(i)}$. where $\mathbf{a}^{(i)}$s are samples of the random vector $\mathbf{a}$. The estimator h is an unbiased estimator of $\mathrm{tr}(\mathsf{K})$. Note that one can change the law of $\mathbf{a}$ to any distribution satisfying $\mathbb{E}[\mathbf{a}] = \mathbf{0}$ and $\mathrm{Var}(\mathbf{a}) = \mathsf{I}$ (*e.g.* Girard's estimator [6] arises when $\mathbf{a}$ is Gaussian with zero mean and unit variance).

Computing $\mathsf{K}\mathbf{a}$ is expensive due the matrix inverse. Even leveraging the sparsity by using Cholesky decomposition has a time complexity $\mathcal{O}(n^3)$ in the worst case. For large n, this cost becomes prohibitive. The state-of-the-art that avoids this cubic cost consists of (preconditioned) conjugate gradient [16], algebraic multigrid [15], polynomial approximations. Specific to inverting SDD matrices, there are graph-based preconditioners [18] that speeds up the convergence of iterative algorithms, yielding a nearly linear time algorithm with m (See [19] for a detailed survey). They compute $\mathsf{K}\mathbf{a}$ with very small error, often much less than the Monte Carlo error induced by Hutchinson's estimator, and they scale linearly with the number of edges m.

RSF estimator. In [2], we proposed an alternative method to estimate $\mathrm{tr}(\mathsf{K})$ when L is a SDD matrix. This method is based on random spanning forests (RSF) [1], a random process on graphs. We showed that the number of roots is an unbiased estimator for $\mathrm{tr}(\mathsf{K})$.

Our contributions. In this work, we improve the efficiency of the RSF-based estimator by well-known variance reduction (VR) techniques from the Monte Carlo literature. The main results of this paper are listed as follows:

- We show two novel ways of applying VR techniques to the RSF-based estimator,
- The additional computations remain in the time complexity $\mathcal{O}(m)$ and come with practical implementations,
- Empirical evidence on various graphs shows that the proposed methods perform at least as well as Hutchinson's estimator, while outperforming it in many settings.

2 Background

In this section, we introduce our notation and revisit some theoretical properties of RSFs.

Graph theory. Consider an undirected, weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$ with $|\mathcal{V}| = n$ nodes and $|\mathcal{E}| = m$ edges. The weight function $w : \mathcal{V} \times \mathcal{V} \to \mathbb{R}_{\geq 0}$ maps $\mathcal{E}$ to positive weights and for $(i, j) \notin \mathcal{E}$, $w(i, j)$ equals to 0. The (weighted) adjacency matrix of a graph is the matrix $\mathsf{A} = [w(i, j)]_{i,j} \in \mathbb{R}^{n \times n}$. Degree of a node i is $d_i = \sum_{j\mathcal{N}(i)} w(i, j)$ where $\mathcal{N}(i)$ is the neighborhood of i. We form the degree matrix as $\mathsf{D} = \mathrm{diag}(\mathbf{d})$. Finally, the graph Laplacian $\mathsf{L} = \mathsf{D} - \mathsf{W}$ is a useful object with many applications in graph combinatorics, machine learning and graph signal processing [17].

Random spanning forests. A tree is a cycle-free subgraph of $\mathcal{G}$. It is a *spanning* tree if it reaches all vertices of $\mathcal{G}$. A rooted tree is a directed tree whose edges are oriented towards a special node called a root. A rooted spanning forest, denoted by ϕ, is a set of disjoint rooted trees on $\mathcal{G}$ whose union reaches all vertices. Let us denote the set of all spanning forests by $\mathcal{F}$. We define an RSF Φ_q as a random object that is defined over $\mathcal{F}$ and has the following distribution:

$$\mathbb{P}(\Phi_q = \phi) \propto q^{|\rho(\phi)|} \prod_{(i,j)\in\phi} w(i, j), \quad q > 0. \tag{2}$$

where $\rho(\phi)$ is the root set of ϕ. Although $|\mathcal{F}|$ can be very large, a modified version of Wilson's algorithm [20] can be used to sample a forest [1]. The algorithm is based on loop-erased random walks on $\mathcal{G}$. Thus, the time complexity of the algorithm is reported as the expected number of steps until it terminates which is equal to $\mathrm{tr}(\mathsf{K}(\mathsf{I} + \frac{1}{q}\mathsf{D})) \leq n + \frac{2m}{q}$ [11] [1].

The random object Φ_q has fascinating theoretical properties that connect various concepts [1]. An important example for this paper is $\mathbb{E}[|\rho(\Phi_q)|]$, which equals $\mathrm{tr}(\mathsf{K})$. Previously, we deployed $|\rho(\Phi_q)|$ as an unbiased estimate of $\mathrm{tr}(\mathsf{K})$. According to experiments performed on various graphs, this estimator is competitive and outperforms in some cases Girard's estimator in terms of the required time for reaching a certain precision. In this work, we improve the expected error of the RSF estimator by VR techniques for Monte Carlo estimators.

3 Proposed Methods

Two VR methods are applicable to the RSF estimator: The control variate (CV) technique and stratified sampling.[2] Moreover, generalizing these methods to SDD matrices is straightforward [2]. Both methods use some additional information (*e.g.* a statistic with a known mean) on the estimator to reduce variance. The

[1] $\mathrm{tr}(\mathsf{K}) + \frac{1}{q}\mathrm{tr}(\mathsf{KD}) \leq n + \frac{1}{q}\mathrm{tr}(\mathsf{KD}) \leq n + \frac{1}{q}\mathrm{tr}(\mathsf{D}) = n + \frac{2m}{q}$.

[2] We omit the main motivations behind these methods due to space limitations but we refer the reader to [10] for more details.

difficulty in applying such methods is to find which additional statistic will be both fast to estimate and provide a substantial decrease in the variance. This paper shows practical ways to adapt these techniques for the RSF estimator.

3.1 Control Variates

We give two RSF based unbiased estimators for K in [14]. Both relies on the root relation $r_\phi : \mathcal{V} \to \rho(\Phi_q)$ which maps every node to its root in ϕ. The first estimator is $\tilde{\mathsf{S}} \coloneqq [\mathbb{I}(r_{\Phi_q}(i) = j)]_{i,j}$ and verifies $\mathbb{E}[\tilde{\mathsf{S}}] = \mathsf{K}$ since $\mathbb{P}(r_{\Phi_q}(i) = j) = \mathsf{K}_{i,j}$. An improved version of this estimator with the CV method is [13]:

$$\tilde{\mathsf{Z}} = \tilde{\mathsf{S}} - \alpha(\mathsf{K}^{-1}\tilde{\mathsf{S}} - \mathsf{I}) \tag{3}$$

Since $\mathbb{E}[\tilde{\mathsf{Z}}] = \mathsf{K}$, we find a unbiased trace estimator:

$$\tilde{s} \coloneqq \mathrm{tr}(\mathsf{Z}) = |\rho(\Phi_q)| - \alpha\tilde{c}, \tag{4}$$

where

$$\tilde{c} = \left(n - |\rho(\Phi_q)| - \frac{1}{q} \sum_{\substack{i \in \rho(\Phi_q) \\ j \in \mathcal{N}(i)}} w(i,j)\mathbb{I}(r_{\Phi_q}(j) \neq i) \right).$$

The random variable $\tilde{c}$ is called the "control variate", and its mean is n. To calculate $\tilde{c}$, one only needs to count the neighbors of each root i that are not rooted in i. For $|\rho(\Phi_q)| \ll n$, the computational cost remains negligible, whereas, in the worst case, it might require traversing every edge of the graph. One can also adapt these calculations for the second estimator in [14]. To do so, let us recall this estimator in matrix form; the trees of Φ_q depict a random partition $\mathcal{P} = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_{|\rho(\Phi_q)|}\}$ over $\mathcal{V} = \cup_{i=1}^{|\rho(\Phi_q)|}\mathcal{V}_i$. We enumerate these components from 1 to $|\rho(\Phi_q)|$ and consider a mapping t from each vertex i to the number of the component that i belongs to. Then, the second estimator takes the form: $\bar{\mathsf{S}} = \left[\frac{\mathbb{I}(i \in \mathcal{V}_{t(j)})}{|\mathcal{V}_{t(j)}|}\right]_{i,j}$. So, one has:

$$\bar{s} \coloneqq |\rho(\Phi_q)| - \alpha\bar{c}, \tag{5}$$

where $\bar{c} = n - |\rho(\Phi_q)| - \frac{1}{q}\sum_{\substack{i \in \mathcal{V} \\ j \in \mathcal{N}(i)}} \bar{\mathsf{S}}_{i,i} w(i,j)\mathbb{I}(r_{\Phi_q}(j) \neq i)$. In this case, the control variate requires keeping track of partition sizes and neighbors at partition boundaries. While the former can be done in $\mathcal{O}(n)$, the latter requires traversing all edges. However, it provides more variance reduction than the previous option(See Prop. 1 and 2 in [14]).

How to choose α. As can be deduced from Prop. 2 in [13], a safe value of α, *i.e.* a value that guarantees variance reduction, is $\frac{2q}{q+d_{max}}$ where d_{max} is the maximum degree in $\mathcal{G}$. We also observe that $\frac{q}{q+d_{avg}}$ is usually a good estimate of $\alpha^\star$ where d_{avg} is the average degree in $\mathcal{G}$.

3.2 Stratified Sampling

Stratification reduces the Monte Carlo error by dividing the sample space into sub-parts, each called a stratum, based on another random variable. Stratified sampling can substantially decrease approximation error when applicable. In the following, we give a way of applying stratification to the RSF-based trace estimator.

Stratification for the RSF estimator. Consider the root set that are sampled at the first visit of random walks in Wilson's algorithm. Let us denote them by $\rho'(\Phi_q)$ and define a random variable $R_i \coloneqq \mathbb{I}(i \in \rho'(\Phi_q))$ where $\mathbb{I}$ is the indicator function. Notice that each R_i is an independent Bernoulli variable with $\mathbb{P}(R_i = 1) = \frac{q}{q+d_i}$. Building on this, we propose to use the cardinality $|\rho'(\Phi_q)| = \sum_{i\in\mathcal{V}} R_i \in \{0, 1, \ldots, n\}$ to apply stratification on the RSF estimator as follows; i/ take disjoint K-fold strata $C_1, \ldots, C_K$ verifying $\bigcup_{i=1}^{K} C_i = \{0, 1, \ldots, n\}$, ii/ get N_i samples of $\Phi_q \Big| |\rho'(\Phi_q)| \in C_i$ for each stratum C_i, iii/ compute the following weighted sum:

$$s_{st} \coloneqq \sum_{i=1}^{K} \frac{1}{N_i} \left(\sum_{\substack{j=1 \\ |\rho'(\phi^{(j)})|\in C_i}}^{N_i} |\rho(\phi^{(j)})| \right) \mathbb{P}(|\rho'(\Phi_q)| \in C_i). \tag{6}$$

For $N = \sum_{i=1}^{K} N_i$ samples, s_{st} gives an unbiased estimation of $\mathrm{tr}(\mathsf{K})$ due to the law of conditional expectation. Moreover, given a fixed N, certain settings of N_i's provide lower theoretical variance, *e.g.* $N_i = N\mathbb{P}(|\rho'(\Phi_q)| \in C_i)$ [10].

Implementation. We address two issues in implementing stratified sampling. The first one is the calculation of the probabilities $\mathbb{P}(|\rho'(\Phi_q)| \in C_i)$. We approximate the distribution of $|\rho'(\Phi_q)|$ by a normal distribution with a mean $\mu = \sum_{i\in\mathcal{V}} \frac{q}{q+d_i}$ and a variance $\sigma^2 = \sum_{i\in\mathcal{V}} \frac{qd_i}{(q+d_i)^2}$ to avoid expensive calculations of the exact methods [8]. The second is to sample the random variable $|\rho(\Phi_q)| \Big| |\rho'(\Phi_q)| \in C_i$. Given a set $\mathcal{X} \subseteq \mathcal{V}$ verifying $|\mathcal{X}| \in C_i$, we can easily adapt Wilson's algorithm for sampling $\Phi_q | \rho'(\Phi_q) = \mathcal{X}$ with two modifications; i/ we pass $\mathcal{X}$ as the initial root set of Φ_q, ii/ we prevent any node $i \notin \mathcal{X}$ being a root at the first visit of walks in Wilson's algorithm. For the generation of the fixed set $\mathcal{X}$, we use rejection sampling [3] which is fast if $\forall i$, $\mathbb{P}(|\rho'(\Phi_q)| \in C_i) \gg 0$.

4 Experiments

We empirically compare the proposed methods to Hutchinson's estimator over various graphs by following a similar procedure to [2]. Notice that all estimators here are Monte Carlo. Therefore, the asymptotic relation between the variance of a Monte Carlo estimator over a single sample σ_1, and N samples σ_N, *i.e.* $\sigma_N \approx \frac{\sigma_1}{N^{1/2}}$, applies to all the estimators in the comparison [3]. We leverage this fact to

[3] This holds for the stratified sampling with $N_i = N\mathbb{P}(|\rho'(\Phi_q)| \in C_i)$ for all i. However, it is not necessarily true for other choices of N_i's.

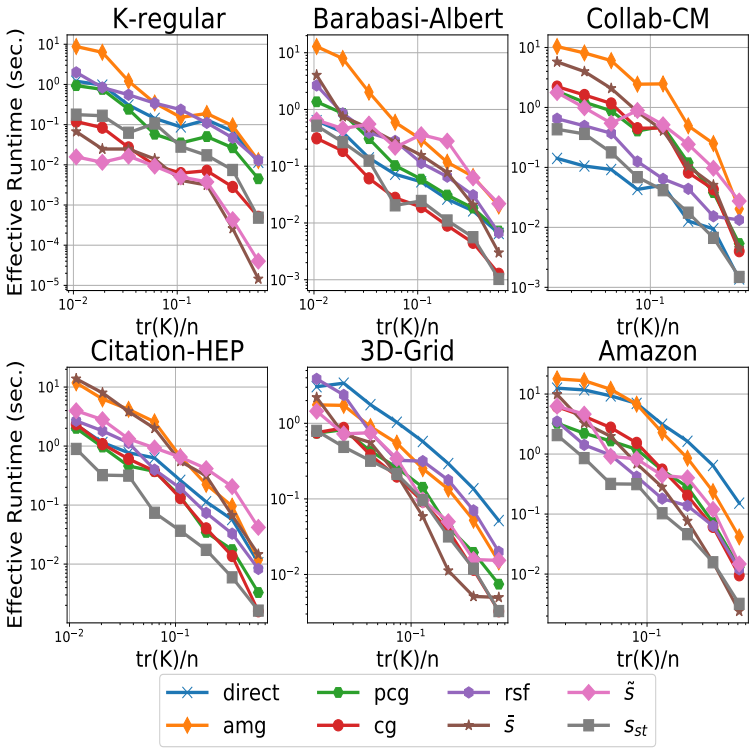

Fig. 1. Effective Runtime vs $\mathrm{tr}(\mathsf{K})/n$.

compare the effective runtimes of all methods *i.e.* the time needed to reach a fixed relative error ϵ. First, we run all methods with $N = 100$. This gives us the average runtime for the computation per sample and the sample variance $\hat{\sigma}_N^2$. Then, we approximate $\hat{\sigma}_1 = \sqrt{N}\hat{\sigma}_N$ for each method. By using this approximation, we solve $\epsilon = \frac{\hat{\sigma}_1}{\mathrm{tr}(K)\sqrt{k}}$ for $\epsilon = 0.002$ to calculate the number of iterations k needed for reaching ϵ error. Finally, we calculate the effective runtime per method by multiplying k by the average time for generating a single sample.

In Hutchinson's estimator, we compute Ka using; Algebraic Multigrid (AMG) [4], Conjugate Gradient (CG) [5], CG with AMG preconditioning, and finally sparse Cholesky decomposition using CHOLMOD [4]. Here, the CG methods benefit from block implementations [12]. We compare these with our proposed methods over various graphs. For $\tilde{s}$ and $\bar{s}$, we set $\alpha = \frac{q}{q+d_{avg}}$. In stratified sampling, we divide the sample space into 5 strata $C_1, \ldots, C_5$ verifying $\mathbb{P}(|\rho'(\Phi_q)| \in C_k) \approx 0.2$ for all $k = 1, \ldots, 5$. We set $N_k = N\mathbb{P}(|\rho'(\Phi_q)| \in C_k)$ per stratum k. The graphs that we use in these experiments are:

[4] `https://github.com/JuliaLinearAlgebra/AlgebraicMultigrid.jl`
[5] `https://docs.juliahub.com/KrylovMethods`

- **Barabasi-Albert**: A random graph generated by Barabasi-Albert model ($k = 10$) with $n = 10^4$ and $m = 99900$,
- **K-random regular**: A random regular graph with $n = 10^4$ and $m = 10^5$ ($k = 20$),
- **Collab-CM**: A collaboration network of $n = 21363$ authors in Arxiv on condense matter physics with $m = 91342$ links,
- **Citation-HEP**: A citation network of $n = 34401$ in Arxiv on high energy physics with $m = 420828$ links.
- **3D Grid**: 3-dimensional grid with $n = 50^3 = 125000$ nodes and $m = 375000$ edges.
- **Amazon**: A real-life network over $n = 262111$ products in Amazon with $m = 899792$. A link between two products indicates that the same client purchases these two products.[6]

We choose 8 logarithmically spaced values of q such that the ratio $\mathrm{tr}(\mathsf{K})/n$ takes values up to 65%. All experiments are implemented in Julia and run in a single thread of a laptop.

Fig. 1 summarizes the results. For relatively small and sparse graphs, such as Collab-CM, the direct method gives the best performance, closely followed by the RSF methods. However, the approximate ones beat the direct method when the graphs become larger or denser. In these cases, the proposed methods give either the best or a comparable performance with the other state-of-the-art methods. A comparison between the regular and highly irregular graphs, *e.g.* K-regular vs Barabasi-Albert, shows that the CV estimators $\tilde{s}$ and $\bar{s}$ gives small expected error in regular cases. This is an expected result since $\tilde{c}$ and $\bar{c}$ have lower variances on regular graphs as they are summations over the neighbors of the roots. In irregular graphs, the stratified sampling estimator often outperforms state-of-the-art.

5 Conclusion

The rich theoretical properties of RSFs give us several ways to improve the RSF trace estimator. In the future, we plan to develop estimators for other Laplacian based quantities, such as the elements of K, or the effective resistances. We also note that we use relatively naive implementations for the stratified sampling method, *e.g.* the normal approximation for the Poisson-Binomial distribution can be improved by using *e.g.* Cornish-Fisher or saddlepoint approximations [5].

References

1. L. Avena and A. Gaudillière. Two applications of random spanning forests. *Journal of Theoretical Probability*, 31(4):1975–2004, 2018.
2. 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.

[6] The real-life data sets can be found in `https://snap.stanford.edu/data/`

3. C. M. Bishop and N. M. Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.
4. Y. Chen, T. A. Davis, W. W. Hager, and S. Rajamanickam. Algorithm 887: Cholmod, supernodal sparse cholesky factorization and update/downdate. *ACM Transactions on Mathematical Software (TOMS)*, 35(3):1–14, 2008.
5. A. DasGupta. *Asymptotic theory of statistics and probability*, volume 180. Springer, 2008.
6. D. Girard. Un algorithme simple et rapide pour la validation croisée généralisée sur des problèmes de grande taille. Technical report, 1987.
7. T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.
8. Y. Hong. On computing the distribution function for the poisson binomial distribution. *Computational Statistics & Data Analysis*, 59:41–51, 2013.
9. M. F. Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076, 1989.
10. J. P. Kleijnen, A. Ridder, and R. Rubinstein. Variance reduction techniques in monte carlo methods. 2010.
11. P. Marchal. Loop-erased random walks, spanning trees and hamiltonian cycles. *Electronic Communications in Probability*, 5:39–50, 2000.
12. D. P. O'Leary. The block conjugate gradient algorithm and related methods. *Linear algebra and its applications*, 29:293–322, 1980.
13. Y. Pilavcı, P.-O. Amblard, S. Barthelmé, and N. Tremblay. Variance reduction in stochastic methods for large-scale regularised least-squares problems. *arXiv preprint arXiv:2110.07894*, 2021.
14. Y. Y. Pilavcı, P.-O. Amblard, S. Barthelme, and N. Tremblay. Graph tikhonov regularization and interpolation via random spanning forests. *IEEE transactions on Signal and Information Processing over Networks*, 7:359–374, 2021.
15. J. W. Ruge and K. Stüben. Algebraic multigrid. In *Multigrid methods*, pages 73–130. SIAM, 1987.
16. J. R. Shewchuk et al. An introduction to the conjugate gradient method without the agonizing pain, 1994.
17. 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(3):83–98, 2013.
18. D. A. Spielman and S.-H. Teng. Nearly linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems. *SIAM Journal on Matrix Analysis and Applications*, 35(3):835–885, 2014.
19. N. K. Vishnoi et al. Lx= b. *Foundations and Trends® in Theoretical Computer Science*, 8(1–2):1–141, 2013.
20. D. B. Wilson. Generating random spanning trees more quickly than the cover time. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 296–303, 1996.
21. L. Wu, J. Laeuchli, V. Kalantzis, A. Stathopoulos, and E. Gallopoulos. Estimating the trace of the matrix inverse by interpolating from the diagonal of an approximate inverse. *Journal of Computational Physics*, 326:828–844, 2016.