COMPUTING THE LOG-DETERMINANT OF SYMMETRIC, DIAGONALLY DOMINANT MATRICES IN NEAR-LINEAR TIME WORKING DRAFT

TIMOTHY HUNTER TJHUNTER@EECS.BERKELEY.EDU
AHMED EL ALAOUI ELALAOUI@EECS.BERKELEY.EDU
ALEXANDRE M. BAYEN BAYEN@BERKELEY.EDU

Abstract. We present new algorithms for computing the log-determinant of symmetric, diagonally dominant matrices. Existing algorithms run with cubic complexity with respect to the size of the matrix in the worst case. Our algorithm computes an approximation of the log-determinant in time near-linear with respect to the number of non-zero entries and with high probability. This algorithm builds upon the utra-sparsifiers introduced by Spielman and Teng for Laplacian matrices and ultimately uses their refined versions introduced by Koutis, Miller and Peng. We also present simpler algorithms that compute upper and lower bounds and that may be of more immediate practical interest.

Key words. Log-determinant, Determinant preconditioners, SDD matrices, Graph sparsification, Laplacian generalized stretch.

AMS subject classifications.

1. Introduction.

We consider the problem of computing the determinant of symmetric, diagonally dominant (SDD) matrices, i.e. real symmetric matrices A for which:

$$A_{ii} \geq \sum_{j \neq i} |A_{ij}|$$

The set of all such matrices of size $n \times n$ is denoted SDD_n, and the set of all symmetric real matrices is called $\mathcal{S}_n$. Call m the number of non-zero entries in A. We are interested in computing the determinant of sparse matrices, i.e. matrices for which $m \ll n^2$.

The best exact algorithm known for computing the determinant of general matrices, the Cholesky factorization, runs in a cubic complexity $O\left(n^3\right)$. Computing the factorization can be sped up for a few specific patterns such as trees, but no algorithm has been shown to work in a generic way for SDD_n, let alone general symmetric matrices. We present an algorithm that returns an approximation of the logarithm of the determinant in time quasi-linear with the number of non-zero entries of A. More specifically, we show that our algorithm, `UltraLogDet`, computes an ϵ-approximation of the logarithm of the determinant with high probability and in expected time[1]:

$$\tilde{O}\left(m\epsilon^{-2}\log^3 n \log^2\left(\frac{n\kappa_A}{\epsilon}\right)\right)$$

where κ_A is the condition number of A. This algorithm builds upon the work of Spielmann and Teng on *ultra-sparsifiers* [26], and it critically exploits the recent improvements from Koutis, Miller and Peng [13]. This is to our knowledge the first algorithm that presents a nearly linear complexity which depends neither on the condition number of A (except through a log-term) nor on a specific pattern for the non-zero coefficients of A.

[1] We use the notation $\tilde{O}$ to hide a factor at most $(\log\log n)^8$

The high sophistication of the algorithm transpires through the large exponent of $\log\log n$. However, our algorithm will directly benefit from any improvement on ultra-sparsifiers. Given the considerable practical importance of such preconditioners, we expect some fast improvements in this area. Also, the bulk of the work is performed in a Monte Carlo procedure that is straightforward to parallelize. Furthermore, we also present simpler, non-optimal algorithms that compute upper and lower bounds of the logarithm of the determinant, and that may be of more immediate practical interest.

1.1. Background. There are two approaches in numerical linear algebra to approximately compute a determinant (or the log of the determinant): by performing a (partial) Cholesky factorization of A, or by considering the trace of some power series.

As mentioned above, the Cholesky factorization performs a decomposition of the form: $A = PLDL^TP^T$ with P a permutation matrix, L a low-triangular matrix with 1 on the diagonal and D a diagonal matrix of non-negative coefficients. Then the log-determinant of A is simply[2]:

$$\log|A| = \sum_i \log D_{ii}$$

The complexity of dense Cholesky factorization for dense matrices is $O\left(n^3\right)$. Unfortunately, Cholesky factorization usually does not gain much from the knowledge of the sparsity pattern due to the *fill-in problem* (see [20], section 3.2). There is one case, though, for which Cholesky factorization is efficient: if the sparsity pattern of A is a tree, then performing Cholesky factorization takes $O(n)$ time, and the matrix L is a banded matrix [16]. If the sparsity pattern of A is not a tree, however, this advantageous decomposition does not hold anymore.

When the matrix A is close to the identity, more precisely when the spectral radius of $M = A - I$ is less than 1, one can use the remarkable Martin expansion of the log-determinant [18]:

$$\log|A| = \mathrm{Tr}\left(\log A\right) \tag{1.1}$$

where $\log A$ is the matrix logarithm defined by the series expansion:

$$\log A = \sum_{i=0}^{\infty} \frac{(-1)^i}{i+1} M^i \tag{1.2}$$

The determinant can then be computed by a sum of traces of the power of M, and the rate of convergence of this series is driven by the spectral radius M. This line of reasoning has led researchers to look for decompositions of A of the form $A = U + V$ with the determinant of U being easier to compute and $U^{-1}V + I$ having a small spectral radius. Then $\log|A| = \log|U| + \log\left|U^{-1}V + I\right|$. The most common decomposition U, V is in terms of block diagonal and off-diagonal terms, which can then use Hadamard inequalities on the determinant to bound the error [11]. Diagonal blocks also have the advantage of having determinants easy to compute. However, this approach requires some strong assumptions on the condition number of A, which may not hold in practice.

[2]We will use the $|\cdot|$ operator to denote the determinant, it will be clear from the context that it is different from the absolute value.

The trace approach is driven by *spectral properties* (the condition number) while the Cholesky approach is driven by *graphical* properties (the non-zero pattern). We propose to combine these two approaches by decomposing the problem with one component that is close to a tree (and is more amenable to Cholesky methods), and one component that has a bounded condition number. Our solution is to use a *spectral sparsifier* introduced by Spielman in [25].

1.2. Applications. The problem of estimating determinants has important applications in spatial data analysis, statistical physics and statistics. In spatial statistics, it is often convenient to interpolate measurements in a 2-, 3- or 4-dimensional volume using a sparse Gaussian process, a technique known in the geospatial community as *kriging* [30, 15]. Computing the optimal parameters of this Gaussian process involves repeated evaluations of the partition function, which is a log-determinant. In this context, a diagonally dominant matrix for the Gram matrix of the process corresponds to distant interactions between points of measure (which is verified in some contexts, see [21]). Determinants also play a crucial role in quantum physics and in theoretical physics. The wave function of a system of multiple fermion particles is an antisymmetric function which can be described as a determinant (Slatter determinant, [3, 17]). In the theory of quantum chromodynamics (QCD), the interaction between particles can be discretized on a lattice, and the energy level of particles is the determinant of some functional operators over this lattice [9]. It is itself a very complex problem because of the size of the matrices involved for any non-trivial problem, for which the number of variables is typically in the millions [6]. In this setting, the restriction to diagonally dominant matrices can be interpreted as an interaction between relatively massive particles [7], or as a bound on the propagation of interactions between sites in the lattice [6].

For these reasons, computing estimates of the log-determinant has been an active problem in physics and statistics. In particular, the Martin expansion presented in Equation (1.1) is extensively used in quantum physics [11], and it can be combined with sampling method to estimate the trace of a matrix series ([31],[19],[32]). Another different line of research has worked on bounds on the values of the determinant itself. This is deeply connected to simplifying statistical models using variational methods. Such a relaxation using a message-passing technique is presented in [28]. Our method is close in spirit to Reuksen's work [22] by the use of a preconditioner. However, Reuksen considers preconditioners based on a clever approximation of the Cholesky decomposition, and its interaction with the eigenvalues of the complete matrix is not well understood. Using simpler methods based on sampling, we are able to carefully control the spectrum of the remainder, which in turn leads to strong convergence guarantees.

1.3. A note on scaling. Unlike other common characteristics of linear operators, the determinant and the log-determinant are very sensitive to dimensionality. We will follow the approach of Reuksen [22] and consider the *regularized log-determinant* $f(A) = n^{-1} \log |A|$ instead of the log-determinant. The regularized determinant has appealing properties with respect to dimensionality. In particular, its sensitivity to perturbations does not increase with the dimensionality, but only depends on spectral properties of the operator A. For example, calling $\lambda_{\min}$ and $\lambda_{\max}$ the minimum and maximum eigenvalues of A, respectively:

$$\log \lambda_{\min} \leq f(A) \leq \log \lambda_{\max}$$

$$\left|f\left(A+\epsilon I\right)-f\left(A\right)\right|\leq\epsilon\left\Vert A^{-1}\right\Vert_{2}+O\left(\epsilon^{2}\right)$$

The last inequality in particular shows that any perturbation to $\log|A|$ will be in the order $O\left(n\right)$, and so that all the interesting log-determinants in practice will be dominated by some $O\left(n\right)$.

1.4. Main results. We first present some general results about the preconditioning of determinants. Consider $A\in SDD_{n}$ invertible, and some other matrix $B\in SDD_{n}$ that is close to A in the spectral sense. All the results of this article stem from observing that:

$$\begin{aligned}\log|A| &= \log|B|+\log\left|B^{-1}A\right|\\ &\log|B|+\mathrm{Tr}\left(\log\left(B^{-1}A\right)\right)\end{aligned}$$

The first section is concerned with estimating the remainder term $\mathrm{Tr}\left(\log\left(B^{-1}A\right)\right)$ using the Martin expansion. The exact inverse B^{-1} is usually not available, but we are given instead a linear operator C that is an $\epsilon-$approximation of B^{-1}, for example using a conjugate gradient method. We show in Section 2 that if the precision of this approximation is high enough, we can estimate the remainder with high probability and with a reasonable number of calls to the operator C (this sentence will be made precise in the rather technical Theorem 2.5). Using this general framework, the subsequent Section 3.1 shows that spectral sparsifiers make excellent preconditioners that are close enough to A and so that computing the Martin expansion is not too expansive. In particular, we build upon the recursive structure of Spielman-Teng ultra-sparsifiers to obtain our main result:

THEOREM 1.1. *On input $A\in SDD_{n}$ with m non-zeros, $\eta>0$, the algorithm* `UltraLogDet` *returns a scalar z so that:*

$$\mathbb{P}\left[\left|z-n^{-1}\log|A|\right|>\epsilon\right]\leq\eta$$

and this algorithm completes in expected time $\tilde{O}\left(m\epsilon^{-2}\log^{2}n\log^{2}\left(\frac{\kappa_{A}}{\epsilon}\right)\log\left(\eta^{-1}\right)\right)$. Moreover, if $\epsilon>\Omega(n^{-1})$, then the running time improves by a factor ϵ.

The rest of the article is structured as follows. In the next section, we present some results about estimating the log-determinant from a truncated expansion. These results will justify the use of *preconditioners* to compute the determinant of a matrix. The techniques developed by Spielman et al. work on the Laplacians of weighted graphs. Section 3 introduces some new concepts to expand the notion of determinants to Laplacian matrices, and presents a few straightforward results in the relations between graph Laplacians and SDD matrices. Section 3.2 will use these new concepts to introduce a first family of preconditioners based on low-stretch spanning trees. Finally, Section 3.3 contains the proof of our main result, an algorithm to compute determinants in near-linear time.

2. Preconditioned log-determinants.

We begin by a close inspection of a simple sampling algorithm to compute log-determinants, presented first in [5]. We will first present some error bounds on this algorithm that expand on bounds previously presented in [4] and [5]. This section considers general symmetric matrices and does not make assumptions about diagonal dominance.

Consider a real symmetric matrix $S\in\mathcal{S}_{n}^{+}$ such that its spectral radius is less than 1: $0\preceq S\preceq\left(1-\delta\right)I$ for some $\delta\in\left(0,1\right)$. Our goal is to compute $\log|I-S|$ up to precision ϵ and with high probability. From the Martin expansion:

$$\log |I - S| = -\mathrm{Tr}\left(\sum_{k=1}^{\infty} \frac{1}{k} S^k\right) \tag{2.1}$$

This series of traces can be estimated by Monte Carlo sampling, up to precision ϵ with high probability, by truncating the series and by replacing the exact trace evaluation by $x^T S^k x$ for some suitably chosen random variables x. In order to bound the errors, we will bound the large deviation errors using the following Bernstein inequality:

Lemma 2.1. *[Bernstein's inequality] Let $X_1 \cdots X_n$ be independent random variables with $\mathbb{E}[X_i] = 0$, $|X_i| < c$ almost surely. Call $\sigma^2 = \frac{1}{n}\sum_i \mathrm{Var}(X_i)$, then for all $\epsilon > 0$:*

$$\mathbb{P}\left[\frac{1}{n}\Big|\sum_i X_i\Big| \geq \epsilon\right] \leq 2\exp\left(-\frac{n\epsilon^2}{2\sigma^2 + 2c\epsilon/3}\right)$$

We can adapt some results from [5] to prove this bound on the deviation from the trace.

Lemma 2.2. *Consider $H \in \mathcal{S}_n$ with the assumption $\lambda_{\min} I_n \preceq H \preceq \lambda_{\max} I$. Consider p vectors sampled from the standard Normal distribution: $\mathbf{u}_i \sim \mathcal{N}(\mathbf{0}, I_n)$ for $i = 1 \cdots p$. Then for all $\epsilon > 0$:*

$$\mathbb{P}\left[\left|\frac{1}{p}\sum_{i=1}^{p} \frac{\mathbf{u}_i^T H \mathbf{u}_i}{\mathbf{u}_i^T \mathbf{u}_i} - \frac{1}{n}\mathrm{Tr}(H)\right| \geq \epsilon\right] \leq 2\exp\left(-\frac{p\epsilon^2}{4n^{-1}(\lambda_{\max} - \lambda_{\min})^2 + 2(\lambda_{\max} - \lambda_{\min})\epsilon/3}\right)$$

Proof. The distribution of $\mathbf{u}_i$ is invariant through a rotation, so we can consider H diagonal. We assume without loss of generality that $H = \mathrm{diag}(\lambda_1, \cdots, \lambda_n)$. Again without loss of generality, we assume that $\lambda'_{\max} = \lambda_{\max} - \lambda_{\min}$ and $\lambda'_{\min} = 0$ (by considering $H' = H - \lambda_{\min} I$). Call $V_i = \frac{\mathbf{u}_i^T H \mathbf{u}_i}{\mathbf{u}_i^T \mathbf{u}_i} - n^{-1}\mathrm{Tr}(H)$. Using results from [5], we have: $|V_i| \leq \lambda_{\max} - \lambda_{\min}$, $\mathbb{E}[V_i] = 0$ and

$$\mathrm{Var}(V_i) = \frac{2}{n(n+2)} \sum_{i=1}^{n} \left(\lambda_i - n^{-1}\mathrm{Tr}(H)\right)^2$$

Each of the variables V_i is independent, so invoking Lemma 2.1 gives:

$$\mathbb{P}\left[\frac{1}{p}\left|\sum_{i=1}^{p} V_i\right| \geq \epsilon\right] \leq 2\exp\left(-\frac{p\epsilon^2}{2\sigma^2 + 2(\lambda_{\max} - \lambda_{\min})\epsilon/3}\right)$$

with

$$\begin{aligned}\sigma^2 &= \frac{2}{n(n+2)} \sum_{i=1}^{n} \left(\lambda_i - n^{-1}\mathrm{Tr}(H)\right)^2 \\ &\leq \frac{2}{n^2} \sum_{i=1}^{n} (\lambda_{\max} - \lambda_{\min})^2 = \frac{2}{n}(\lambda_{\max} - \lambda_{\min})^2\end{aligned}$$

□

The previous lemma shows that if the eigenspectrum of a matrix is bounded, we can obtain a Bernstein bound on the error incurred by sampling the trace. Furthermore, the convergence of the series (2.1) is also determined by the extremal eigenvalues of S. If we truncate the series (2.1), we can bound the truncation error using the extremal eigenvalues. We formalize this intuition in the following theorem, which is adapted from the main theorem in [5]. While that main theorem in [5] only considered a confidence interval based on the covariance properties of Gaussian distribution, we generalize this result to a more general Bernstein bound.

THEOREM 2.3. *Consider $S \in \mathcal{S}_n^+$ with $0 \preceq S \preceq (1-\delta) I$ for some $\delta \in (0,1)$. Call $y = n^{-1} \log |I - S|$ the quantity to estimate, and consider $\mathbf{u}_i \sim \mathcal{N}(\mathbf{0}, I_n)$ for $i = 1 \cdots p$ all independent. Call $\hat{y}_{p,l}$ an estimator of the truncated series of l elements computed by sampling the trace using p samples:*

$$\hat{y}_{p,l} = -\frac{1}{p} \sum_{j=1}^{p} \sum_{k=1}^{l} \frac{1}{k} \frac{\mathbf{u}_j^T S^k \mathbf{u}_j}{\mathbf{u}_j^T \mathbf{u}_j}$$

Given $\epsilon > 0$ and $\eta \in (0,1)$, the $\hat{y}_{p,l}$ approximates y up to precision ϵ with probability at least $1 - \eta$ by choosing $p \geq 16\left(\frac{1}{\epsilon} + \frac{1}{n\epsilon^2}\right) \log(2/\eta) \log^2\left(\delta^{-1}\right)$ and $l \geq 2\delta^{-1} \log\left(\frac{n}{\delta\epsilon}\right)$:

$$\mathbb{P}\left[|y - \hat{y}_{p,l}| \geq \epsilon\right] \leq \eta$$

The proof of this result is detailed in Appendix A.

From this theorem we derive two results that justify the notion of preconditioners for determinants: one for exact preconditioners and one for approximate preconditioners. The corresponding algorithm, which we call `PreconditionedLogDetMonteCarlo`, is presented in Algorithm 1.

COROLLARY 2.4. *Let $A \in \mathcal{S}_n^+$ and $B \in \mathcal{S}_n^+$ be positive definite matrices so that B is a $\kappa-$approximation of A:*

$$A \preceq B \preceq \kappa A \tag{2.2}$$

Given $\epsilon > 0$ and $\eta \in (0,1)$, the algorithm `PreconditionedLogDetMonteCarlo` computes $\frac{1}{n} \log\left|B^{-1}A\right|$ up to precision ϵ with probability greater than $1 - \eta$, by performing $16\kappa\left(\frac{1}{\epsilon} + \frac{1}{n\epsilon^2}\right) \log\left(\frac{2\kappa}{\epsilon}\right) \log(2/\eta) \log^2(\kappa)$ vector inversions from B and vector multiplies from A.

The proof of this corollary is presented in Appendix A. Usually, computing the exact inverse by an SDD matrix is too expensive. We can instead extend the previous result to consider a black box procedure that approximatively computes $B^{-1}x$. If the error introduced by the approximate inversion is small enough, the result from the previous corollary still holds. This is what the following theorem establishes:

THEOREM 2.5. *Consider $A, B \in \mathcal{S}_n^+$ positive definite with B a $\kappa-$approximation of A with $\kappa \geq 2$. Furthermore, assume there exists a linear operator C so that for all $y \in \mathbb{R}^n$, C returns a $\nu-$approximation of $B^{-1}y$:*

$$\left\|C(y) - B^{-1}y\right\|_B \leq \nu \left\|B^{-1}y\right\|_B$$

Given $\eta \in (0,1)$ and $\epsilon > 0$, if $\nu \leq \min\left(\frac{\epsilon}{8\kappa^3\kappa(B)}, \frac{1}{2\kappa}\right)$, then the algorithm `PreconditionedLogDet-MonteCarlo` returns a scalar z so that:

$$\mathbb{P}\left[\left|z - n^{-1} \log\left|B^{-1}A\right|\right| \geq \epsilon\right] \leq \eta$$

by performing $64\kappa\left(\frac{1}{\epsilon}+\frac{1}{n\epsilon^2}\right)\log\left(\frac{2\kappa}{\epsilon}\right)\log\left(2/\eta\right)\log^2\left(\kappa\right)$ *vector calls to the operator* C *and vector multiplies from* A*.*

The proof of this result is detailed in Appendix A. While the overall bound looks the same, the constant (taken away by the $O(\cdot)$ notation) is four times as large as in Corollary 2.4.

This last theorem shows that we can compute a good approximation of the log-determinant if the preconditioner B: (a) is close to A in the spectral sense, and (b) can be approximately inverted and the error introduced by the approximate inversion can be controlled. This happens to be the case for symmetric, diagonally dominant matrices.

```
Algorithm **PreconditionedLogDetMonteCarlo**($B$,$A$,$\eta$,$p$,$l$):
$y \leftarrow 0$
for $j$ from 1 to $p$:
  Sample $\mathbf{u} \sim \mathcal{N}(\mathbf{0}, I)$
  $\mathbf{v} \leftarrow \mathbf{u}/\|\mathbf{u}\|$
  $z \leftarrow 0$
  for $k$ from 1 to $l$:
    $\mathbf{v} \leftarrow B^{-1}A\mathbf{v}$ up to precision $\eta$
    $z \leftarrow z + k^{-1}\mathbf{v}^T\mathbf{u}$
  $y \leftarrow y + p^{-1}z$
Return $y$
```

Algorithm 1: PreconditionedLogDetMonteCarlo

3. Ultra-sparsifiers as determinant preconditioners.

3.1. Reduction on a Laplacian.

From now on, we consider the computation of $\log A$, where $A \in SDD_n$. The techniques we will develop work on Laplacian matrices instead of SDD matrices. An SDD matrix is positive semi-definite while a Laplacian matrix is always singular, since its nullspace is spanned by $\mathbf{1}$. We generalize the definition of the determinant to handle this technicality.

Definition 3.1. Pseudo-log-determinant (PLD): *Let* $A \in \mathcal{S}^{n+}$ *be a non-null positive semi-definite matrix. The pseudo-log-determinant is defined by the sum of the logarithms of all the positive eigenvalues:*

$$\operatorname{ld}(A) = \sum_{\lambda_i > 0} \log(\lambda_i)$$

where λ_i *are the eigenvalues of* A*.*

The interest of the PLD lies in the connection between SDD matrices and some associated Laplacian. It is well-known that solving an SDD system in SDD_n can be reduced to solving a Laplacian system of size $2n+1$, using the reduction technique introduced Gremban in [10]. Recall that a Laplacian has all its non-diagonal terms non-positive, the sum of each row and each column being zero. The reduction has been simplified by Kelner et al. in [12], Appendix A. Using the Kelner et al. reduction, we can turn the computation of a the log-determinant of a SDD system into the computation of two PLDs of Laplacians, as shown in the next lemma.

Lemma 3.2. Kelner et al. reduction for log-determinants. *Given an invertible SDD matrix* A*, consider the Kelner decomposition* $A = D_1 + A_p + A_n + D_2$ *where:*

- A_p *is the matrix that contains all the positive off-diagonal terms of* A

- *A_n is the matrix that contains all the negative off-diagonal terms of A*
- *D_1 is a diagonal matrix that verifies $D_1(i,i) = \sum_{j \neq i} |A(i,j)|$*
- *D_2 is the excess diagonal matrix: $D_2 = A - A_p - A_n - D_1$*

Call $\hat{A} = D_1 + A_n - A_p$ and $\tilde{A} = \begin{pmatrix} D_1 + D_2/2 + A_n & -D_2/2 - A_p \\ -D_2/2 - A_p & D_1 + D_2/2 + A_n \end{pmatrix}$. Then $\hat{A}$ and $\tilde{A}$ are both Laplacian matrices and

$$\log|A| = \mathrm{ld}\left(\tilde{A}\right) - \mathrm{ld}\left(\hat{A}\right)$$

Proof. The matrices $\hat{A}$ and $\tilde{A}$ are Laplacian by constructions, and we show that the eigenvalues of $\tilde{A}$ are exactly the concatenation of the eigenvalues of $\hat{A}$ and A. Call λ_i an eigenvalue of A with x an associated eigenvector. Then the vector $\begin{pmatrix} x \\ -x \end{pmatrix}$ is an eigenvector of $\tilde{A}$ with associated eigenvalue λ. Similarly, call μ_i an eigenvalue of $\hat{A}$ with y an associated eigenvector. Then μ is an eigenvalue of $\tilde{A}$ with associated eigenvector $\begin{pmatrix} y \\ y \end{pmatrix}$. Since $\tilde{A}$ is exactly of size $2n$, the set of eigenvalues of $\tilde{A}$ is exactly the concatenation of the eigenvalues of $\hat{A}$ and A. By definition of the PLD: $\mathrm{ld}\left(\tilde{A}\right) = \sum_{i:\lambda_i>0} \log \lambda_i + \sum_{\mu_i>0} \log \mu_i$. Since A is invertible, $\lambda_i > 0$ for all i and $\sum_{i:\lambda_i>0} \log \lambda_i = \sum_i \log \lambda_i = \log|A|$. Finally, by definition of the PLD, we get $\sum_{\mu_i>0} \log \mu_i = \mathrm{ld}\left(\hat{A}\right)$.
□

To any Laplacian L we can associate a unique positive definite matrix F_L (up to a permutation), and this transform preserves eigenvalues and matrix inequalities. We call this process "floating" of the Laplacian, by analogy to the "grounding" in the electrical sense of the SDD matrix as a Laplacian introduced by Gremban (see [10], Chapter 4).

Definition 3.3. Floating a Laplacian. *Consider L a Laplacian matrix. Call F_L the matrix formed by removing the last row and the last column from L.*

The following lemma shows that the Laplacian matrix overdetermines a system, and that no information is lost by floating it.

Lemma 3.4. *Consider Z a (weighted) Laplacian matrix of a connected graph, then:*

1. *The eigenvalues of F_Z are the positive eigenvalues of Z, and the corresponding eigenvectors for F_Z are the same eigenvectors, truncated by the last coefficient.*
2. $\mathrm{ld}(Z) = \log|F_Z|$
3. *Given Z_1, Z_2 Laplacian matrices, we have $Z_1 \preceq Z_2 \Rightarrow F_{Z_1} \preceq F_{Z_2}$.*

The proof of this lemma is straightforward, and is contained in Appendix B.

A Laplacian matrix can be considered either for its graphical properties, or for its algebraic properties. Recent results have shown a deep connection between these two aspects, and they let us develop a general framework for computing determinants: consider a Laplacian L_G identified to its graph G. Using graphical properties of L_G, we can construct a subgraph H of G for which the PLD is easier to compute and that is a good approximation of G in the spectral sense. Then we can float the subgraph H and apply results of section 2 to approximate the remainder with high probability.

More precisely:

$$\begin{aligned}\operatorname{ld}(L_G) &= \log|F_{L_G}| \\ &= \operatorname{ld}(L_H) - \log|F_{L_H}| + \log|F_{L_G}| \\ &= \operatorname{ld}(L_H) + \log\left|F_{L_H}^{-1}F_{L_G}\right|\end{aligned}$$

The first term $\operatorname{ld}(L_H)$ is usually easier to compute by considering the graphical properties of L_H, while the remainder $\log\left|F_{L_H}^{-1}F_{L_G}\right|$ is approximated by sampling. Preconditioner graphs L_H are typically efficient to factorize using Cholesky factorization, and close enough to G so that the sampling procedure from the previous section can be applied to compute $\log\left|F_{L_H}^{-1}F_{L_G}\right|$. We will see how to adapt Spielman and Teng's remarkable work on *ultra-sparsifiers* to produce good preconditioners H for the determinant.

3.2. A first preconditioner. While the results in this section are not the main claims of this paper, we hope they will provide some intuition, and an easier path towards an implementation.

We present a first preconditioner that is not optimal, but that will motivate our results for stronger preconditioners: a tree that spans the graph G. Every graph has a low-stretch spanning tree, as discovered by Alon et al. [2]. The bound of Alon et al. was then improved by Abraham et al. [1]. We restate their main result.

LEMMA 3.5. *(Lemma 9.2 from [26]). Consider a weighted graph G. There exists a spanning tree T that is a subgraph of G so that:*

$$L_T \preceq L_G \preceq \kappa L_T$$

with $\kappa = \tilde{\mathcal{O}}(m \log n)$.

Proof.

This follows directly from [26]. T is a subgraph of G (with the same weights on the edges), so $L_T \preceq L_G$ (see [26] for example for a proof of this fact). Furthermore, we have $L_G \preceq \operatorname{st}_T(G)\, L_T$. This latter inequality is a result of Spielman et al. in [23] that we generalize in Lemma 3.13. Finally, a result by [1] shows that T can be chosen such that $\operatorname{st}_T(G) \leq \mathcal{O}(m \log n (\log\log n)^3)$.□

Trees enjoy a lot of convenient properties for Gaussian elimination. The Cholesky factorization of a tree can be computed in linear time, and furthermore this factorization has a linear number of non-zero elements [26]. This factorization can be expressed as:

$$L_T = PLDL^T P^T$$

where P is a permutation matrix, L is a lower-triangular matrix with the diagonal being all ones, and D a diagonal matrix in which all the elements but the last one are positive, the last element being 0. These well-known facts about trees are presented in [26]. Once the Cholesky factorization of the tree is performed, the log-determinant of the original graph is an immediate by-product:

$$\log|L_T| = \sum_{i=1}^{n-1} \log D_{ii}$$

Furthermore, computing $L_T^+ x$ also takes $O(n)$ computations by forward-backward substitution (see [8]). Combining Corollary 2.4 and Lemma (3.5) gives immediately the following result.

THEOREM 3.6. *Let G be a graph with n vertices and m edges. Its PLD can be computed up to a precision ϵ and with high probability in time:*

$$\tilde{O}\left(m^2 \log n \log^2(m)\left(\frac{1}{\epsilon}+\frac{1}{n\epsilon^2}\right)\log\left(\frac{2m}{\epsilon}\right)\log(2/\eta)\right)$$

Proof. Using Lemma (3.5), we compute a low-stretch tree L_T so that $L_T \preceq L_G \preceq \kappa L_T$ with $\kappa = \tilde{\mathcal{O}}(m \log n)$. Using Corollary (2.4), approximating the PLD with high precision requires

$$\begin{aligned}&\tilde{O}\left(\kappa\left(\frac{1}{\epsilon}+\frac{1}{n\epsilon^2}\right)\log\left(\frac{2\kappa}{\epsilon}\right)\log(2/\eta)\log^2(\kappa)\right)\\ &=\tilde{O}\left(m\log n\left(\frac{1}{\epsilon}+\frac{1}{n\epsilon^2}\right)\log\left(\frac{2m}{\epsilon}\right)\log(2/\eta)\log^2(m)\right)\end{aligned}$$

inversions by the tree T (done in $O(n)$) and vector products by the floated Laplacian F_{L_G}(done in $O(m)$). The overall cost is $\tilde{O}\left(m^2 \log n \log^2(m)\left(\frac{1}{\epsilon}+\frac{1}{n\epsilon^2}\right)\log\left(\frac{2m}{\epsilon}\right)\log(2/\eta)\right)$.
□

The previous result shows that the log-determinant can be compute in roughly $\mathcal{O}\left(m^2\right)$ (m being the number of non-zero entries). This result may be of independent interest since it requires relatively little machinery to compute, and it is a theoretical improvement already for graphs with small vertex degree ($m = \mathcal{O}\left(n^{1+o(1)}\right)$) over the Cholesky factorization of G (which has complexity $\mathcal{O}\left(n^3\right)$ in all generality). Also, note that the PLD of the tree constructed above provides an upper bound to the log-determinant of G since $L_G \preceq \kappa L_T$. We will see in Subsection 3.4 that we can compute a non-trivial lower bound as well.

3.3. Incremental sparsifiers. We can do better and achieve near-linear time by using ultra-sparsifiers. The main insight of our result is that the class preconditioners presented by Spielman and Teng are based on incomplete Cholesky factorization, and hence have a determinant that is relatively easy to compute, and furthermore that they are excellent spectral preconditioners, so the procedure `PreconditionedLogDet-MonteCarlo` is efficient to apply. We reintroduce some concepts presented in [13] to present a self-contained result. The following paragraphs are well-known facts about Spielman-Teng preconditioners and have been presented in [13, 26].

The central idea to the Spielman-Teng preconditioner is to sample $O(n)$ edges from the graph A, to form a subgraph B that is close to a tree (hence it is easy to compute some partial Cholesky factorization), yet it is close to the original A is the spectral sense ($A \preceq B \preceq \kappa A$), thanks to the additional edges. The partial Cholesky factorization is computed using the `GreedyElimination` algorithm presented in [13]. In order for this section to be self-contained, we include here the main results of Section 4 in [26].

Consider the Laplacian matrix L_B of the subgraph B. There exists an algorithm that computes the partial Cholesky factorization:

$$L_B = PLCL^TP^T$$

where:

- P is a permutation matrix

- L is a non-singular, low triangular matrix of the form

$$L = \begin{pmatrix} L_{1,1} & 0 \\ L_{2,1} & I_{n_1} \end{pmatrix}$$

with the diagonal of $L_{1,1}$ being all ones.
- C has the form

$$C = \begin{pmatrix} D_{n-n_1} & 0 \\ 0 & L_{A_1} \end{pmatrix}$$

and every row and column of L_{A_1} has at least 3 non-zero coefficients. Furthermore, L_{A_1} is itself Laplacian and:

$$\mathrm{ld}\left(L_G\right) = \sum_{1}^{n-n_1} \log D_{ii} + \mathrm{ld}\left(L_{A_1}\right)$$

The exact algorithm that achieves this factorization is called `GreedyElimination` and is presented in [13]. Using this factorization, the PLD of the original Laplacian L_A is:

$$\begin{aligned} \mathrm{ld}\left(L_A\right) &= \mathrm{ld}\left(L_B\right) + \mathrm{ld}\left(B^+A\right) \\ &= \sum_{1}^{n-n_1} \log D_{ii} + \mathrm{ld}\left(A_1\right) + \mathrm{ld}\left(B^+A\right) \end{aligned} \tag{3.1}$$

Thus, we are left with solving a smaller problem A_1, and we approximate the value of $\mathrm{ld}\left(B^+A\right)$ using the algorithm `SampleLogDet`. ST preconditioners are appealing for this task: they guarantee that A_1 is substantially smaller than A, so the recursion completes in $O\left(\log\log n\right)$ steps. Furthermore, computing the vector product B^+Ax is itself efficient (in can be done approximated in near-linear time), so we can apply Theorem 2.5. We formalize the notion of chain of preconditioners by reintroducing some material from [13].

```
Algorithm UltraLogDet($A$,$\epsilon$,$\eta$):
If $A$ is of a small size (<100), directly compute $\mathrm{ld}(A)$ with a dense Cholesky factorization.
Compute $B$ =IncrementalSparsify($A$)
Compute $D, A'$ =PartialCholesky($B$)
$\eta \leftarrow \min\left(\frac{\epsilon}{8\kappa^3\kappa(B)}, \frac{1}{2\kappa}\right)$
$p \leftarrow 8\left(\frac{1}{\epsilon} + \frac{1}{n\epsilon^2}\right)\log\left(\eta^{-1}\right)\log^2\left(\delta^{-1}\right)$
$l \leftarrow \delta^{-1}\log\left(\frac{2}{\epsilon\delta}\right)$
Compute $s$ =PreconditionedLogDetMonteCarlo($B, A, \eta, p, l$)
Return $s + \log|D|$ +UltraLogDet($A'$,$\epsilon$,$\eta$)
```
Algorithm 2: Sketch of the main algorithm

DEFINITION 3.7. *Definition 4.2 from [14]. Good preconditioning chain. Let $d \in \mathbb{N}^*$, $\mathcal{C} = \{A_1 = A, B_1, A_2, B_2, A_3 \dots B_{d-1}, A_d\}$ be a chain of graphs and $\mathcal{K} = (\kappa_1 \cdots \kappa_{d-1}) \in \mathbb{R}_+^{d-1}$. We say that $\{\mathcal{C}, \mathcal{K}\}$ is a good preconditioning chain for A if there exists $\mathcal{U} = (\mu_1 \cdots \mu_d) \in \mathbb{N}_+^d$ so that:*

1. $A_i \preceq B_i \preceq \kappa_i A_i$.

2. A_{i+1}= `GreedyElimination`(B_i) .

3. The number of edges of A_i is less than μ_i.

4. $\mu_1 = \mu_2 = m$ where m is the number of edges of A.

5. $\mu_i/\mu_{i+1} \geq c_r \left\lceil \sqrt{\kappa_i} \right\rceil$ for some constant c_r.

6. $\kappa_{i+1} \leq \kappa_i$.

7. μ_d is smaller than some fixed arbitrary constant.

Good chains exist, as found by Koutis, Miller and Peng:

LEMMA 3.8. *(Lemma 4.5 from [14]) Given a graph A, the algorithm* ***BuildChain***(A, p) *from [14] produces with probability $1-p$ a good preconditioning chain $\{\mathcal{C}, \mathcal{K}\}$ such that $\kappa_1 = \tilde{O}\left(\log^2 n\right)$ and $\kappa_i = \kappa_c$ for all $i \geq 2$ for some constant κ_c. The length of the chain is $d = \mathcal{O}(\log n)$ and the algorithm runs in expected time $\tilde{O}(m \log n)$.*

These chains furthermore can be used as good preconditioners for conjugate gradient and lead to near-linear algorithms for approximate inversion (Lemma 7.2 from [13]). This remarkable result has been significantly strenghtened in the previous years, so that SDD systems can be considered to be solved in (expected) linear time.

LEMMA 3.9. *(Theorem 4.6 from [14]). Given $A \in SDD_n$ with m non-zero entries, $b \in \mathbb{R}^n$ and $\nu > 0$, a vector x such that $\left\|x - A^+ b\right\|_A < \nu \left\|A^+ b\right\|_A$ can be computed in expected time $\tilde{O}(m \log n \log(1/\nu))$.*

It should now become clear how we can combine a good chain with the Algorithm `PreconditionedLogDetMonteCarlo`. We start by building a chain. The partial Cholesky factorizations at each step of the chain provide an upper bound to $\mathrm{ld}(A)$. We then refine this upper bound by running `PreconditionedLogDetMonteCarlo` at each state of the chain to approximate $\mathrm{ld}\left(B_i^+ A_i\right)$ with high probability. The complete algorithm is presented in Algorithm 2. We now have all the tools required to prove Theorem 1.1.

Proof of Theorem 1.1. First, recall that we can consider either an SDD or its grounded Laplacian thanks to the relation $\log A = \mathrm{ld} L_A$. Call $A_1 = L_A$ the first element of the chain. In this proof, all the matrices will be Laplacian from now on. Using Lemma 3.8, consider $\mathcal{C} = \{A_1 = A, B_1, A_2, \ldots A_d\}$ a good chain for A, with $d = \mathcal{O}(\log n)$. More precisely, since A_{i+1}= `GreedyElimination`(B_i), the Laplacian B_i can be factored as:

$$B_i = P_i L_i \begin{pmatrix} D^{(i)} & 0 \\ 0 & A_{i+1} \end{pmatrix} L_i^T P_i^T$$

with P_i a permutation matrix, L_i a lower triangular matrix with 1 one the diagonal and $D^{(i)}$a positive definite diagonal matrix. The matrix $D^{(i)}$ is an immediate by-product of running the algorithm `GreedyElimination` and can be obtained when forming the chain $\mathcal{C}$ at no additional cost.

From the discussion at the start of the section, it is clear that $\mathrm{ld} B_i = \sum_k \log D_k^{(i)} + \mathrm{ld} A_{i+1}$. From the discussion in Section 2, the log-determinant of A is:

$$\begin{aligned}
\log A &= \mathrm{ld} A_1 \\
&= \mathrm{ld} B_1 + \mathrm{ld}\left(B_1^+ A_1\right) \\
&= \sum_k \log D_k^{(1)} + \mathrm{ld} A_{i+1} + \mathrm{ld}\left(B_1^+ A_1\right) \\
&= \mathrm{ld} A_d + \sum_{i=1}^{d} \left(\sum_k \log D_k^{(i)} \right) + \sum_{i=1}^{d} \mathrm{ld}\left(B_i^+ A_i\right)
\end{aligned}$$

The term $\mathrm{ld} A_d$ can be estimated by dense Cholesky factorization at cost $\mathcal{O}(1)$, and the diagonal Cholesky terms $\sum_k \log D_k^{(i)}$ are already computed from the chain. We are left with estimating the d remainders $\mathrm{ld}\left(B_i^+ A_i\right)$. By construction, $A_i \preceq B_i \preceq \kappa_i A_i$ and by Lemma 3.9, there exists an operator C_i so that $\left\|C_i\left(b\right) - B_i^+ b\right\|_{B_i} < \nu \left\|B_i^+ b\right\|_{B_i}$ for all b with a choice of relative precision $\nu = \frac{\epsilon}{16\kappa_i^3 \kappa(B_i)}$.

This relative precision depends on the condition number $\kappa\left(B_i\right)$ of B_i. We can coarsely relate this condition number to the condition number of $A_1$by noting the following:

- Since $A_i \preceq B_i \preceq \kappa_i A_i$ by construction, $\kappa\left(B_i\right) \leq \kappa_i \kappa\left(A_i\right)$
- For diagonally dominant matrices or Laplacian matrices, the condition number of the partial Cholesky factor is bounded by the condition number of the original matrix. This can be seen by analyzing one update in the Cholesky factorization. Given a partially factorized matrix $\tilde{A} = \begin{pmatrix} I_p & 0 & 0 \\ 0 & a & b \\ 0 & b^T & S \end{pmatrix}$, after factorization, the next matrix is $\begin{pmatrix} I_{p+1} & 0 \\ 0 & S - a^{-1}bb^T \end{pmatrix}$. The spectrum of the Schur complement $S - a^{-1}bb^T$is bounded by the spectrum of $\begin{pmatrix} a & b \\ b^T & S \end{pmatrix}$ (see Corollary 2.3 in [29]) and thus its condition number is upper bounded by that of $\tilde{A}$.

As a consequence, we have for all i: $\kappa\left(A_{i+1}\right) \leq \kappa\left(B_i\right) \leq \kappa_i \kappa\left(A_i\right) \leq \prod_{j=1}^{i} \kappa_j \kappa\left(A_1\right) = \tilde{O}\left(\kappa_1 \kappa_c^{i-1} \kappa\left(A\right)\right)$ with κ_c the constant introduced in Lemma 3.8. This coarse analysis gives us the bound:

$$\kappa\left(B_i\right) \leq \tilde{O}\left(\kappa_c^{\log n} \log^2 n \; \kappa\left(A\right)\right) = \tilde{O}\left(n^{\log \kappa_c} \log^2 n \; \kappa\left(A\right)\right).$$

Consider the relative precision $\tilde{\nu} = \tilde{\mathcal{O}}\left(n^{-\log \kappa_c} \log^{-8} n \frac{\epsilon}{\kappa(A)}\right)$ so that $\tilde{\nu} \leq \nu_i$ for all i. Constructing the operator C_i is a byproduct of forming the chain $\mathcal{C}$. By Theorem 2.3, each reminder $\mathrm{ld}\left(B_i^+ A_i\right)$ can be approximated to precision ϵ with probability at least $1 - \eta$ using Algorithm 1. Furthermore, this algorithm works in expected time

$$\begin{aligned} &\tilde{O}\left(m \log n \log\left(1/\tilde{\nu}\right) \kappa_1 \left(\frac{1}{\epsilon} + \frac{1}{n\epsilon^2}\right) \log\left(\frac{n\kappa_1}{\tilde{\nu}}\right) \log^2\left(\kappa_1\right) \log\left(\eta^{-1}\right)\right) \\ &= \tilde{O}\left(m \log^3 n \left(\frac{1}{\epsilon} + \frac{1}{n\epsilon^2}\right) \log^2\left(\frac{n\kappa\left(A\right)}{\epsilon}\right) \log\left(\eta^{-1}\right)\right) \end{aligned}$$

By a union bound, the result also holds on the sum of all the $\log n$ approximations of the remainders. We can simplify this bound a little by assuming that $\epsilon \geq n^{-1}$, which then becomes $\tilde{O}\left(m\epsilon^{-1} \log^3 n \log^2\left(\frac{n\kappa(A)}{\epsilon}\right) \log\left(\eta^{-1}\right)\right)$.

3.4. Stretch bounds on preconditioners. How good is the estimate provided by the preconditioner? Intuitively, this depends on how well the preconditioner L_H approximates the graph L_G. This notion of quality of approximation can be formalized by the notion of *stretch*. This section presents a deterministic bound on the PLD of L_G based on the PLD of L_H and the stretch of G relative to H. This may be useful in practice as it gives a (tight) interval for the PLD before performing any Monte-Carlo estimation of the residual.

The stretch of a graph is usually defined with respect to a (spanning) tree. In our analysis, it is convenient and straightforward to generalize this definition to arbitrary graphs. To our knowledge, this straightforward extension is not considered in the litterature, so we feel compelled to properly introduce it.

Definition 3.10. Generalized stretch. *Consider $\mathcal{V}$ a set of vertices, $G = (\mathcal{V}, \mathcal{E}_G)$, $H = (\mathcal{V}, \mathcal{E}_H)$ connected graphs over the same set of vertices, and L_G, L_H their respective Laplacians. The stretch of G with respect to H is the sum of the effective resistances of each edge of graph G with repect to graph H,*

$$st_H(G) = \sum_{(u,v)\in\mathcal{E}_G} L_G(u,v)(\mathcal{X}_u - \mathcal{X}_v)^T L_H^+ (\mathcal{X}_u - \mathcal{X}_v)$$

with $\mathcal{X}_u \in \mathbb{R}^n$ the unit vector that is 1 at position u, and zero otherwise.

If the graph H is a tree, this is a standard definition of stretch, because the effective resistance $(\mathcal{X}_u - \mathcal{X}_v)^T L_H^+ (\mathcal{X}_u - \mathcal{X}_v)$ between vertices u and v is the sum of all resistances over the unique path between u and v (see Lemma 2.4 in [27]). Furthermore, the arguments to prove Theorem 2.1 in [27] carry over to our definition of stretch. For the sake of completeness, we include this result:

Lemma 3.11. *(Straightforward generalization of Theorem 2.1 in [27]) Let $G = (\mathcal{V}, \mathcal{E}_G)$, $H = (\mathcal{V}, \mathcal{E}_H)$ be connected graphs over the same set of vertices, and L_G, L_H their respective Laplacians. Then:*

$$st_H(G) = Tr\left(L_H^+ L_G\right)$$

with $L_H^+$the pseudo-inverse of L_H.

Proof.

We denote $E(u,v)$ the Laplacian unit matrix that is 1 in position u, v: $E(u,v) = (\mathcal{X}_u - \mathcal{X}_v)(\mathcal{X}_u - \mathcal{X}_v)^T$. This is the same arguments as the original proof:

$$\begin{aligned}
\mathrm{Tr}\left(L_H^+ L_G\right) &= \sum_{(u,v)\in\mathcal{E}_G} L_G(u,v)\,\mathrm{Tr}\left(E(u,v)\,L_H^+\right)\\
&= \sum_{(u,v)\in\mathcal{E}_G} L_G(u,v)\,\mathrm{Tr}\left((\mathcal{X}_u - \mathcal{X}_v)(\mathcal{X}_u - \mathcal{X}_v)^T L_H^+\right)\\
&= \sum_{(u,v)\in\mathcal{E}_G} L_G(u,v)(\mathcal{X}_u - \mathcal{X}_v)^T L_H^+ (\mathcal{X}_u - \mathcal{X}_v)\\
&= \mathrm{st}_H(G)
\end{aligned}$$

□

A consequence is $\mathrm{st}_H(G) \geq \mathrm{Card}(\mathcal{E}_G) \geq n-1$ for connected G and H with $L_G \succeq L_H$, and that for any connected graph G, $\mathrm{st}_G(G) = n-1$. Scaling and matrix inequalities carry over with the stretch as well. Given A, B, C connected graphs, and $\alpha, \beta > 0$:

$$\begin{aligned}
&\mathrm{st}_{\alpha A}(\beta B) = \alpha^{-1}\beta\,\mathrm{st}_A(B)\\
&L_A \preceq L_B \Rightarrow \mathrm{st}_A(C) \geq \mathrm{st}_B(C)\\
&L_A \preceq L_B \Rightarrow \mathrm{st}_C(A) \leq \mathrm{st}_C(B)
\end{aligned}$$

Lemma 3.12. *For any connected graph G, $st_G(G) = n-1$.*

Proof. Consider the diagonalization of L_G: $L_G = P\Delta P^T$ with $P \in \mathbb{R}^{n\times n-1}$ and $\Delta = \mathrm{diag}(\lambda_1, \cdots, \lambda_{n-1})$. Then

$$\mathrm{st}_G(G) = \mathrm{Tr}\left(P\Delta P^T P\Delta^{-1}P^T\right) = \mathrm{Tr}(I_{n-1}) = n-1$$

□

A number of properties of the stretch extend to general graphs using the generalized stretch. In particular, the stretch inequality (Lemma 8.2 in [26]) can be generalized to arbitrary graphs (instead of spanning trees).

Lemma 3.13. *Let $G = (\mathcal{V}, \mathcal{E}_G)$, $H = (\mathcal{V}, \mathcal{E}_H)$ be connected graphs over the same set of vertices, and L_G, L_H their respective Laplacians. Then:*

$$L_G \preceq st_H(G)\, L_H$$

Proof. The proof is very similar to that of Lemma 8.2 in [27], except that the invocation of Lemma 8.1 is replaced by invoking Lemma 3.23 in Appendix B. The Laplacian G can be written as a linear combination of edge Laplacian matrices:

$$L_G = \sum_{e\in\mathcal{E}_G} \omega_e L(e) = \sum_{(u,v)\in\mathcal{E}_G} \omega_{(u,v)} (\mathcal{X}_u - \mathcal{X}_v)(\mathcal{X}_u - \mathcal{X}_v)^T$$

and a positivity result on the Schur complement gives

$$(\mathcal{X}_u - \mathcal{X}_v)(\mathcal{X}_u - \mathcal{X}_v)^T \preceq \Big((\mathcal{X}_u - \mathcal{X}_v)^T L_H^+ (\mathcal{X}_u - \mathcal{X}_v) \Big) L_H$$

By summing all the edge inequalities, we get:

$$\begin{aligned} L_G &\preceq \sum_{(u,v)\in\mathcal{E}_G} \omega_{(u,v)} (\mathcal{X}_u - \mathcal{X}_v)^T L_H^+ (\mathcal{X}_u - \mathcal{X}_v) L_H \\ &\preceq \mathrm{st}_H(G)\, L_H \end{aligned}$$

□

This bound is remarkable as it relates any pair of (connected) graphs, as opposed to spanning trees or subgraphs. An approximation of the generalized stretch can be quickly computed using a construct detailled in [24], as we will see below. We now introduce the main result of this section: a bound on the PLD of L_G using the PLD of L_H and the stretch.

Theorem 3.14. *Let $G = (\mathcal{V}, \mathcal{E}_G)$, $H = (\mathcal{V}, \mathcal{E}_H)$ be connected graphs over the same set of vertices, and L_G, L_H their respective Laplacians. Assuming $L_H \preceq L_G$, then:*

$$ld(L_H) + \log(st_H(G) - n + 2) \le ld(L_G) \le ld(L_H) + (n-1)\log\left(\frac{st_H(G)}{n-1}\right) \quad (3.2)$$

This bound is tight.

Proof. This is an application of Jensen's inequality on $\mathrm{ld}\left(L_H^+ L_G\right)$. We have $\mathrm{ld}(L_G) = \mathrm{ld}(L_H) + \mathrm{ld}\left(L_H^+ G\right)$ and $\mathrm{ld}\left(L_H^+ G\right) = \mathrm{ld}\left(\sqrt{L_H}^+ L_G \sqrt{L_H}^+\right)$ with $\sqrt{T}$ the matrix square root of T. From Lemma 3.24, we have the following inequality:

$$\begin{aligned} \mathrm{ld}\left(\sqrt{L_H}^+ L_G \sqrt{L_H}^+\right) &\le (n-1)\log\left(\frac{\mathrm{Tr}\left(\sqrt{L_H}^+ L_G \sqrt{L_H}^+\right)}{n-1}\right) \\ &= (n-1)\log\left(\frac{\mathrm{Tr}\left(L_H^+ L_G\right)}{n-1}\right) \\ &= (n-1)\log\left(\frac{\mathrm{st}_H(G)}{n-1}\right) \end{aligned}$$

The latter equality is an application of Lemma 3.11.

The lower bound is slightly more involved. Call λ_i the positive eigenvalues of $\sqrt{L_H}^+ L_G \sqrt{L_H}^+$ and $\sigma = \mathrm{st}_H(G)$. We have $1 \le \lambda_i$ from the asumption $L_H \preceq L_G$. By definition: $\mathrm{ld}\left(L_H^+ L_G\right) = \sum_i \log \lambda_i$. Furthermore, we know from Lemma 3.11 that $\sum_i \lambda_i = \sigma$. The upper and lower bounds on λ_i give:

$$\mathrm{ld}\left(L_H^+ L_G\right) \ge \min \sum_i \log \lambda_i$$

$$\text{s.t.}\, \lambda_i \ge 1,\, \sum_i \lambda_i = \sigma$$

Since there are precisely $n-1$ positive eigenvalues λ_i, one can show that the minimization problem above has a unique minimum which is $\log(\sigma - n + 2)$.

To see that, consider the equivalent problem of minimizing $\sum_i \log(1+u_i)$ under the constraints $\sum_i u_i = \sigma - (n-1)$ and $u_i \ge 0$. Note that:

$$\sum_i \log(1+u_i) = \log\left(\prod_i [1+u_i]\right) = \log\left(1 + \sum_i u_i + \mathrm{Poly}(u)\right)$$

with $\mathrm{Poly}(u) \ge 0$ for all $u_i \ge 0$, so we get: $\sum_i \log(1+u_i) \ge \log(1 + \sum_i u_i)$ and this inequality is tight for $u_1 = \sigma - (n-1)$ and $u_{i\ge 2} = 0$. Thus the vector $\lambda^* = (\sigma - n + 2,\, 1 \cdots 1)^T$ is (a) a solution to the minimization problem above, and (b) the objective value of any feasible vector λ is higher or equal. Thus, this is the solution (unique up to a permutation). Hence we have $\mathrm{ld}\left(L_H^+ L_G\right) \ge \sum_i \log \lambda_i^* = \log(\sigma - n + 2)$.

Finally, note that if $H = G$, then $\mathrm{st}_H(G) = n - 1$, which gives an equality.□

Note that Lemma 3.13 gives us $L_H \preceq L_G \preceq \mathrm{st}_H(G)\, L_H$ which implies $\mathrm{ld}(L_H) \le \mathrm{ld}(L_G) \le \mathrm{ld}(L_H) + n \log \mathrm{st}_H(G)$. The inequalities in Theorem 3.14 are stronger. Interestingly, it does not make assumption on the topology of the graphs (such as L_H being a subset of L_G). Research on conditioners has focused so far on low-stretch approximations that are subgraphs of the original graph. It remains to be seen if some better preconditioners can be found with stretches in $\mathcal{O}(n)$ by considering more general graphs. In this case, the machinery developped in Section 3 would not be necessary.

From a practical perspective, the stretch can be calculated also with respect to the number of non-zero entries.

Lemma 3.15. *Let $G = (\mathcal{V}, \mathcal{E}_G)$, $H = (\mathcal{V}, \mathcal{E}_H)$ be connected graphs over the same set of vertices, and L_G, L_H their respective Laplacians. Call $r = \max_e L_H(e) / \min_e L_H(e)$. Given $\epsilon > 0$, there exists an algorithm that returns a scalar y so that:*

$$(1-\epsilon)\, st_H(G) \le y \le (1+\epsilon)\, st_H(G)$$

with high probability and in expected time $\tilde{\mathcal{O}}\left(m\epsilon^{-2}\log(rn)\right)$.

Proof. This is a straightforward consequence of Theorem 2 in [24]. Once the effective resistance of an edge can be approximated in time $O\left(\log n/\epsilon^2\right)$, we can sum it and weight it by the conductance in G for each edge.
□

3.5. Fast inexact estimates. The bound presented in Equation 3.2 has some interesting consequences if one is interested only in a rough estimate of the log-determinant: if $\epsilon = O(1)$, it is possible to approximate the log-determinant in expected time $\tilde{O}\left(m + n\log^3 n\right)$. We will make use of this sparsification result from Spielman and Srivastava [24]:

LEMMA 3.16. *(Theorem 12 in [24]). Given a Laplacian L_G with m edges, there is an expected $\tilde{O}\left(m/\epsilon^2\right)$ algorithm that produces a graph L_H with $O\left(n\log n/\epsilon^2\right)$ edges that satisfies $(1-\epsilon)\,L_G \preceq L_H \preceq (1+\epsilon)\,L_G$.*

An immediate consequence is that given any graph, we can find a graph with a near-optimal stretch (up to an ϵ factor) and $O\left(n\log n/\epsilon^2\right)$ edges.

LEMMA 3.17. *Given a Laplacian L_G with m edges, there is an expected $\tilde{O}\left(m/\epsilon^2\right)$ algorithm that produces a graph L_H with $O\left(n\log n/\epsilon^2\right)$ edges that satisfies $(n-1) \leq st_H(G) \preceq \frac{1+\epsilon}{1-\epsilon}(n-1)$.*

Proof. Consider a graph H produced by Lemma 3.16, which verifies $(1-\epsilon)\,L_G \preceq L_H \preceq (1+\epsilon)\,L_G$. Using the stretch over this matrix inequality, this implies:

$$\mathrm{st}_{(1+\epsilon)G}(G) \leq \mathrm{st}_H(G) \leq \mathrm{st}_{(1-\epsilon)G}(G)$$

which is equivalent to:

$$(1+\epsilon)^{-1}\,\mathrm{st}_G(G) \leq \mathrm{st}_H(G) \leq (1-\epsilon)^{-1}\,\mathrm{st}_G(G)$$

and the stretch of a connected graph with respect to itself is $n-1$. By rescaling H to $(1+\epsilon)^{-1} H$, we get:

$$n-1 \leq \mathrm{st}_H(G) \leq \frac{1+\epsilon}{1-\epsilon}(n-1)$$

□

Here is the main result of this section:

PROPOSITION 3.18. *There exists an algorithm that on input $A \in SDD_n$, returns an approximation $n^{-1}\log|A|$ with precision $1/2$ in expected time $\tilde{O}\left(m + n\log^3 n\log^2\kappa(A)\right)$ with $\kappa(A)$ the condition number of A.*

Proof. Given L_A, compute H from Lemma 3.17 using $\epsilon = 1/16$ so that $(n-1) \leq \mathrm{st}_H(G) \preceq (1+1/8)(n-1)$. Then, using Theorem 3.14, this leads to the bound:

$$\mathrm{ld}(H) \leq \log|A| \leq \mathrm{ld}(H) + \frac{n-1}{4}$$

since H has $O(n\log n)$ edges by construction, we can use Theorem 1.1 to compute a 1/4- approximation of $\mathrm{ld}(H)$ in expected time $\tilde{O}\left(n\log^3 n\log^2(\kappa(H))\right)$. By construction $\kappa(H) \leq \frac{1+1/16}{1-1/16}\kappa(A)$, hence the result.

□

It would be interesting to see if this technique could be developed to handle arbitrary precision as well.

Comments. Since the bulk of the computations are performed in estimating the residue PLD, it would be interesting to see if this could be bypassed using better bounds based on the stretch.

Also, even if this algorithm presents a linear bound, it requires a fairly advanced machinery (ST solvers) that may limit its practicality. Some heuristic implementation, for example based on algebraic multi-grid methods, could be a first step in this direction.

The authors are much indebted to Satish Rao and James Demmel for suggesting the original idea, and to Benjamin Recht for helpful comments on the draft of this article.

Appendix A: Proofs of Section 2.

3.6. Proof of Theorem 2.3. *Proof.* The proof of this theorem follows the proof of the Main Theorem in [5] with some slight modifications. Using triangular inequality:

$$|y - \hat{y}_{p,l}| \leq |\mathbb{E}[\hat{y}_{p,l}] - \hat{y}_{p,l}| + |y - \mathbb{E}[\hat{y}_{p,l}]|$$

Since S is upper-bounded by $(1-\delta) I$, we have for all $k \in \mathbb{N}$:

$$\left|\mathrm{Tr}\left(S^k\right)\right| \leq n(1-\delta)^k$$

We have $\mathbb{E}[\hat{y}_{p,l}] = -\sum_{i=1}^{l} i^{-1} S^i$ and $y = -\sum_{i=1}^{\infty} i^{-1} S^i$. Using again triangle inequality, we can bound the error with respect to the expected value:

$$\begin{aligned}
|y - \mathbb{E}[\hat{y}_{p,l}]| &= n^{-1} \left| \sum_{i=l+1}^{\infty} \frac{1}{i} \mathrm{Tr}\left(S^k\right) \right| \\
&\leq n^{-1} \sum_{i=l+1}^{\infty} \frac{1}{i} \left|\mathrm{Tr}\left(S^k\right)\right| \\
&\leq \frac{1}{n(l+1)} \sum_{i=l+1}^{\infty} \left|\mathrm{Tr}\left(S^k\right)\right| \\
&\leq \frac{1}{l+1} \sum_{i=l+1}^{\infty} (1-\delta)^k \\
&\leq \frac{1}{l+1} \frac{(1-\delta)^{l+1}}{\delta} \\
&\leq \frac{(1-\delta)^{l+1}}{\delta}
\end{aligned}$$

And since $\delta \leq -\log(1-\delta)$, for a choice of $l \geq \delta^{-1} \log\left(\frac{2}{\epsilon\delta}\right)$, the latter part is less than $\epsilon/2$. We now bound the first part using Lemma 2.2. Call H the truncated series:

$$H = -\sum_{i=1}^{m} \frac{1}{i} S^i$$

This truncated series is upper-bounded by 0 (H is negative, semi-definite). The lowest eigenvalue of the truncated series can be lower-bounded in terms of δ:

$$H = -\sum_{i=1}^{m} \frac{1}{i} S^i \succeq -\sum_{i=1}^{m} \frac{1}{i} (1-\delta)^i I \succeq -\sum_{i=1}^{+\infty} \frac{1}{i} (1-\delta)^i I = (\log \delta) I$$

We can now invoke Lemma 2.2 to conclude:

$$\mathbb{P}\left[\left|\frac{1}{p} \sum_{i=1}^{p} \left(\mathbf{u}_i^T \mathbf{u}_i\right)^{-1} \mathbf{u}_i^T H \mathbf{u}_i - n^{-1} \mathrm{Tr}(H)\right| \geq \frac{\epsilon}{2}\right] \leq 2 \exp\left(-\frac{p\epsilon^2}{16 n^{-1} (\log(1/\delta))^2 + 4 \log(1/\delta)\, \epsilon/3}\right)$$

Thus, any choice of

$$p \geq 16\left(\frac{1}{\epsilon} + \frac{1}{n\epsilon^2}\right) \log(2/\eta) \log^2\left(\delta^{-1}\right) \geq \log(2/\eta)\, \epsilon^{-2} \left(16 n^{-1} \log^2\left(\delta^{-1}\right) + \frac{4}{3}\epsilon \log\left(\delta^{-1}\right)\right)$$

satisfies the inequality:$2 \exp\left(-\frac{p\epsilon^2}{16 n^{-1} (\log(1/\delta))^2 + 4 \log(1/\delta)\epsilon/3}\right) \leq \eta$.

□

3.7. Proof of Corollary 2.4. *Proof.* We introduce some notations that will prove useful for the rest of the article:

$$H = I - B^{-1}A$$

$$S = I - B^{-1/2}AB^{-1/2}$$

with $B^{-1/2}$ the inverse of the square root[3] of the positive-definite matrix B. The inequality (2.2) is equivalent to $\kappa^{-1}B \preceq A \preceq B$, or also:

$$\left(1-\kappa^{-1}\right) I \succeq I - B^{-1/2}AB^{-1/2} \succeq 0$$

$$\left(1-\kappa^{-1}\right) I \succeq S \succeq 0 \tag{3.3}$$

The matrix S is a contraction, and its spectral radius is determined by κ. Furthermore, computing the determinant of $B^{-1}A$ is equivalent to computing the determinant of $I - S$:

$$\begin{aligned} \log |I-S| &= \log \left|B^{-1/2}AB^{-1/2}\right| \\ &= \log |A| - \log |B| \\ &= \log \left|B^{-1}A\right| \\ &= \log |I-H| \end{aligned}$$

and invoking Theorem 2.3 gives us bounds on the number of calls to matrix-vector multiplies with respect to S. It would seem at this point that computing the inverse square root of B is required, undermining our effort. However, we can reorganize the terms in the series expansion to yield only full inverses of B. Indeed, given $l \in \mathbb{N}^*$,

[3]Given a real PSD matrix X, which can be diagonalized: $X = Q\Delta Q^T$ with Δ diagonal, and $\Delta_{ii} \geq 0$. Call $Y = Q\sqrt{\Delta}Q^T$ the square root of X, then $Y^2 = X$.

consider the truncated series:

$$\begin{aligned}
y_l &= -\mathrm{Tr}\left(\sum_{i=1}^{l}\frac{1}{i}S^i\right)\\
&= -\sum_{i=1}^{l}\frac{1}{i}\mathrm{Tr}\left(S^i\right)\\
&= -\sum_{i=1}^{l}\frac{1}{i}\mathrm{Tr}\left(\sum_j\left(\begin{array}{c} j\\ i-j\end{array}\right)(-1)^j\left(B^{-1/2}AB^{-1/2}\right)^j\right)\\
&= -\sum_{i=1}^{l}\frac{1}{i}\sum_j\left(\begin{array}{c} j\\ i-j\end{array}\right)(-1)^j\,\mathrm{Tr}\left(\left(B^{-1/2}AB^{-1/2}\right)^j\right)\\
&= -\sum_{i=1}^{l}\frac{1}{i}\sum_j\left(\begin{array}{c} j\\ i-j\end{array}\right)(-1)^j\,\mathrm{Tr}\left(\left(B^{-1}A\right)^j\right)\\
&= -\sum_{i=1}^{l}\frac{1}{i}\mathrm{Tr}\left(\sum_j\left(\begin{array}{c} j\\ i-j\end{array}\right)(-1)^j\left(B^{-1}A\right)^j\right)\\
&= -\sum_{i=1}^{l}\frac{1}{i}\mathrm{Tr}\left(H^i\right)
\end{aligned}$$

Hence, the practical computation of the latter sum can be done on $A^{-1}B$. To conclude, if we compute $p = 16\left(\frac{1}{\epsilon}+\frac{1}{n\epsilon^2}\right)\log\left(2/\eta\right)\log^2\left(\kappa\right)$ truncated chains of length $l = \kappa\log\left(\frac{2\kappa}{\epsilon}\right)$, we get our result. This requires lp multiplications by A and inversions by B. □

3.8. Proof of Theorem 2.5. We prove here the main result of Section 2. In the following, A and B are positive-definite matrices in $\mathcal{S}_n$, and B is a $\kappa-$approximation of A ($A \preceq B \preceq \kappa A$). The following notations will prove useful:

$$S = I - B^{-1/2}AB^{-1/2} \tag{3.4}$$

$$R = I - B^{-1}A \tag{3.5}$$

$$\varphi = \kappa^{-1}$$

Recall the definition of the matrix norm. Given $M \in \mathcal{S}_n^+$, $\|M\|_B = \max_{x\neq 0}\sqrt{\frac{x^TMx}{x^TBx}}$

Lemma 3.19. *S and R are contractions for the Euclidian and $B-$norms:*

$$\begin{aligned}
\|S\| &\leq 1-\varphi\\
\|R\| &\leq 1-\varphi\\
\|R\|_B &\leq (1-\varphi)^2
\end{aligned}$$

Proof. Recall the definition of the matrix norm: $\|S\| = \max_{x^T x \leq 1} \sqrt{x^T S x}$. Since we know from Equation (3.3) that $S \preceq (1-\varphi) I$, we get the first inequality.

The second inequality is a consequence of Proposition 3.3 from [26]: A and B have the same nullspace and we have the linear matrix inequality $A \preceq B \preceq \kappa A$, which implies that the eigenvalues of $B^{-1}A$ lie between $\kappa^{-1} = \varphi$ and 1. This implies that the eigenvalues of $I - B^{-1}A$ are between 0 and $1-\varphi$.

Recall the definition of the matrix norm induced by the B-norm over $\mathbb{R}^n$:

$$\begin{aligned}
\|R\|_B &= \max_{x \neq 0} \frac{\|Rx\|_B}{\|x\|_B} \\
&= \max_{\|x\|_B^2 \leq 1} \sqrt{x^T R^T B R x} \\
&= \max_{x^T B x \leq 1} \sqrt{x^T R^T B R x} \\
&= \max_{y^T y \leq 1} \sqrt{y^T B^{-1/2} R^T B R B^{-1/2} y}
\end{aligned}$$

and the latter expression simplifies:

$$\begin{aligned}
B^{-1/2} R^T B R B^{-1/2} &= B^{-1/2} \left(I - AB^{-1}\right) B \left(I - B^{-1}A\right) B^{-1/2} \\
&= \left(I - B^{-1/2} A B^{-1/2}\right) \left(I - B^{-1/2} A B^{-1/2}\right) \\
&= S^2
\end{aligned}$$

so we get:

$$\|R\|_B = \left\|S^2\right\| \leq \|S\|^2 \leq (1-\varphi)^2$$

□

The approximation of the log-determinant is performed by computing sequences of power series $\left(R^k x\right)_k$. These chains are computed approximately by repeated applications of the R operator on the previous element of the chain, starting from a random variable x_0. We formalize the notion of an approximate chain.

Definition 3.20. Approximate power sequence. *Given a linear operator H, a start point $x^{(0)} \in \mathbb{R}^n$, and a positive-definite matrix D, we define an $\epsilon-$approximate power sequence as a sequence that does not deviate too much from the power sequence:*

$$\left\|x^{(k+1)} - Hx^{(k)}\right\|_D \leq \epsilon \left\|Hx^{(k)}\right\|_D$$

We now prove the following result that is quite intuitive: if the operator H is a contraction and if the relative error ϵ is not too great, the sum of all the errors on the chain is bounded.

Lemma 3.21. *Let H be a linear operator and D a norm over the space of that linear operator. Assume that the operator H is a contraction under this norm ($\|H\|_D < 1$) and consider $\rho \in (0,1)$ so that $\|H\|_D \leq (1-\rho)^2$. Consider $\left(x^{(k)}\right)_k$ a $\nu-$approximate power sequence for the operator H and the norm D. If $\rho \leq 1/2$ and $\nu \leq \rho/2$, the total error is bounded:*

$$\sum_{k=0}^{\infty} \left\|x^{(k)} - H^k x^{(0)}\right\|_D \leq 4\rho^{-2}\nu \left\|x^{(0)}\right\|_D$$

Proof. Call $\omega_k = \left\|x^{(k)} - H^k x^{(0)}\right\|_D$ and $\theta_k = \left\|Hx^{(k)}\right\|_D$. We are going to bound the rate of convergence of these two series. We have first using triangular inequality on the D norm and then the definition of the induced matrix norm.

$$\begin{aligned}\theta_k &\leq \left\|Hx^{(k)} - H^k x^{(0)}\right\|_D + \left\|H^k x^{(0)}\right\|_D \\ &= \omega_k + \left\|H^k x^{(0)}\right\|_D \\ &\leq \omega_k + \|H\|_D^k \left\|x^{(0)}\right\|_D\end{aligned}$$

We now bound the error on the ω_k sequence:

$$\begin{aligned}\omega_{k+1} &= \left\|x^{(k+1)} - Hx^{(k)} + Hx^{(k)} - H^{k+1}x^{(0)}\right\|_D \\ &\leq \left\|Hx^{(k)} - H^{k+1}x^{(0)}\right\|_D + \left\|x^{(k+1)} - Hx^{(k)}\right\|_D \\ &\leq \|H\|_D \left\|x^{(k)} - H^k x^{(0)}\right\|_D + \nu \left\|Hx^{(k)}\right\|_D \\ &= \|H\|_D \omega_k + \nu\theta_k \\ &\leq \|H\|_D \omega_k + \nu \left(\omega_k + \|H\|_D^k \left\|x^{(0)}\right\|_D\right) \\ &\leq \left[\|H\|_D + \nu\right] \omega_k + \nu \|H\|_D^k \left\|x^{(0)}\right\|_D\end{aligned}$$

The assumption $\rho \leq 1 - \sqrt{\|H\|_D}$ is equivalent to $\|H\|_D \leq (1-\rho)^2$, so the last inequality implies:

$$\omega_{k+1} \leq \left[(1-\rho)^2 + \nu\right] \omega_k + \nu (1-\rho)^{2k} \left\|x^{(0)}\right\|_D$$

Note that the inequality $(1-\rho)^2 + \nu \leq 1 - \rho$ is equivalent to $\nu \leq \rho - \rho^2$. Using the hypothesis, this implies:

$$\omega_{k+1} \leq (1-\rho)\,\omega_k + \nu (1-\rho)^{2k} \left\|x^{(0)}\right\|_D \tag{3.6}$$

We show by induction that:

$$\forall k, \omega_k \leq \frac{\nu \left\|x^{(0)}\right\|_D}{1 - \sqrt{1-\rho}} \left(\sqrt{1-\rho}\right)^{k-1}$$

Note first that

$$\begin{aligned}\omega_1 &= \left\|x^{(1)} - Hx^{(0)}\right\|_D \\ &\leq \nu \left\|Hx^{(0)}\right\|_D \\ &\leq \nu \|H\|_D \left\|x^{(0)}\right\|_D \\ &\leq \nu (1-\rho)^2 \left\|x^{(0)}\right\|_D \\ &\leq \nu \left\|x^{(0)}\right\|_D\end{aligned}$$

So this relation is verified for $k = 1$. Now, assuming it is true for k, we use Equation (3.6) to see that:

$$\begin{aligned}
\omega_k &\le (1-\rho)\,\omega_k + \nu\,(1-\rho)^{2k}\left\|x^{(0)}\right\|_D \\
&\le (1-\rho)\,\omega_k + \nu\left(\sqrt{1-\rho}\right)^k\left\|x^{(0)}\right\|_D \\
&\le \nu\left\|x^{(0)}\right\|_D\left[\frac{(1-\rho)}{1-\sqrt{1-\rho}}\left(\sqrt{1-\rho}\right)^{k-1} + \left(\sqrt{1-\rho}\right)^k\right] \\
&= \nu\left\|x^{(0)}\right\|_D\left(\sqrt{1-\rho}\right)^k\left[\frac{\sqrt{1-\rho}}{1-\sqrt{1-\rho}} + 1\right] \\
&= \frac{\nu\left\|x^{(0)}\right\|_D}{1-\sqrt{1-\rho}}\left(\sqrt{1-\rho}\right)^k
\end{aligned}$$

which is the the property for $k+1$. Using this property, we can sum all the errors by a geometric series (note that $\omega_0 = 0$).

$$\sum_{k=1}^{\infty}\omega_k \le \frac{\nu\left\|x^{(0)}\right\|_D}{1-\sqrt{1-\rho}}\sum_{k=0}^{\infty}\left(\sqrt{1-\rho}\right)^k = \frac{\nu\left\|x^{(0)}\right\|_D}{\left(1-\sqrt{1-\rho}\right)^2}$$

Finally, note that for $\rho \in (0, 1/2)$, the inequality $\nu \le \rho/2$ implies $\nu \le \rho - \rho^2$. Furthermore, by concavity of the square root function, we have $\sqrt{1-\rho} \le 1 - \rho/2$ for $\rho \le 1$. Thus, $\left(1-\sqrt{1-\rho}\right)^2 \ge \rho^2/4$ and we get our result.
□

We can use the bound on the norm of A to compute bound the error with a preconditioner:

Lemma 3.22. *Consider A, B with the same hypothesis as above, $x_0 \in \mathbb{R}^n$, and the additional hypothesis $\nu \in \left(0, \frac{1}{2\kappa}\right)$ and $\kappa \ge 2$, and $(x_u)_u$ an $\nu-$approximate power sequence for the operator R with start vector x_0. Then:*

$$\left|\sum_{i=1}^{l}\frac{1}{i}x_0^T R^i x_0 - \sum_{i=1}^{l}\frac{1}{i}x_0^T x_i\right| \le 4\nu\kappa^2\sqrt{\kappa(B)}\,\|x_0\|^2$$

where $\kappa(B)$ is the condition number of B.

Proof.

Call $\hat{z}$ the truncated sequence:

$$\hat{z} = \sum_{i=1}^{l}\frac{1}{i}x_0^T x_i$$

This sequence is an approximation of the exact sequence z:

$$z = \sum_{i=1}^{l}\frac{1}{i}x_0^T R^i x_0$$

We now bound the error between the two sequences:

$$|\hat{z} - z| \le \sum_{i=1}^{l}\frac{1}{i}\left|x_0^T\left(R^i x_0 - x_i\right)\right| \le \sum_{i=1}^{l}\left|x_0^T\left(R^i x_0 - x_i\right)\right| \le \sum_{i=1}^{l}\left|\left(B^{-1}x_0\right)^T B\left(R^i x_0 - x_i\right)\right| \tag{3.7}$$

Using the Cauchy-Schwartz inequality, we obtain:

$$\left|\left(B^{-1}x_0\right)^T B\left(R^i x_0 - x_i\right)\right| = \left|\left\langle B^{-1}x_0, R^i x_0 - x_i\right\rangle_B\right| \leq \left\|B^{-1}x_0\right\|_B \left\|R^i x_0 - x_i\right\|_B \tag{3.8}$$

From Lemma (3.19), we have $\|R\|_B \leq (1-\varphi)^2$, and from the hypothesis, we have $\nu \in (0, \varphi/2)$ and $\varphi \leq 1/2$, so we can bound the deviation using the bound from Lemma 3.21:

$$\sum_{i=1}^{l} \left\|R^i x_0 - x_i\right\|_B \leq \sum_{i=1}^{\infty} \left\|R^i x_0 - x_i\right\|_B \leq 4\varphi^{-2}\nu \|x_0\|_B = 4\kappa^2\nu \|x_0\|_B \tag{3.9}$$

Combining Equations (3.7), (3.8) and (3.9), we get:

$$|\hat{z} - z| \leq \left\|B^{-1}x_0\right\|_B \sum_{i=1}^{l} \left\|R^i x_0 - x_i\right\|_B \leq 4\nu\kappa^2 \left\|B^{-1}x_0\right\|_B \|x_0\|_B$$

Finally, it is more convenient to consider the Euclidian norm for the norm of x_0. Call $\lambda_{\max}$ and $\lambda_{\min}$ the extremal eigenvalues of the positive semidefinite matrix B. By definition of the matrix norm: $\|x_0\|_B = \sqrt{x_0^T B x_0} \leq \sqrt{\lambda_{\max}} \|x_0\|$ and $\left\|B^{-1}x_0\right\|_B = \sqrt{x_0^T B^{-1} x_0} \leq \sqrt{\lambda_{\min}^{-1}} \|x_0\|$ so we get:

$$|\hat{z} - z| \leq 4\nu\kappa^2 \sqrt{\kappa(B)} \|x_0\|^2$$

where $\kappa(B)$ is the condition number of B. □

We now have all the elements required for the proof of Theorem 2.5.

Proof.

Consider $\mathbf{u}_j \sim \mathcal{N}(0, I_n)$ for $j = 1 \cdots p$, and $x_{i,j} = \begin{cases} \mathbf{u}_j / \|\mathbf{u}_j\| & i = 0 \\ x_{i-1,j} - C\left(Ax_{i-1,j}\right) & i > 0 \end{cases}$

Call

$$z_{p,l} = \frac{1}{p} \sum_{j=1}^{p} \sum_{i=1}^{l} \frac{1}{i} \left(x_{0,j}\right)^T x_{i,j}$$

$$\hat{y}_{p,l} = \frac{1}{p} \sum_{j=1}^{p} \sum_{k=1}^{l} \frac{1}{k} \left(x_{0,j}\right)^T S^k x_{0,j}$$

By construction, $\left(x_{i,j}\right)_i$ is an $\nu-$approximate chain for the operator R. Applying Lemma 3.22 to the operator R under the norm B, we get:

$$|z_{p,l} - \hat{y}_{p,l}| \leq 4\nu\kappa^2 \sqrt{\kappa(B)} \left[\frac{1}{p} \sum_{j=1}^{p} \left\|x_{0,j}\right\|^2\right] = 4\nu\kappa^2 \sqrt{\kappa(B)}$$

since $\left\|x_{0,j}\right\|^2 = 1$, which gives us a deterministic bound. Consider $\nu \leq \min\left(\frac{\epsilon}{8\kappa^2\sqrt{\kappa(B)}}, \frac{1}{2\kappa}\right)$. Then $|z_{p,l} - \hat{y}_{p,l}| \leq \epsilon/2$. Furthermore:

$$|z_{p,l} - y| \leq |z_{p,l} - \hat{y}_{p,l}| + |y - \hat{y}_{p,l}|$$

and $\mathbb{P}\left[|y - \hat{y}_{p,l}| \geq \epsilon/2\right] \leq \eta$ for a choice of $p \geq 16\left(\frac{1}{\epsilon} + \frac{1}{n\epsilon^2}\right)\log\left(2/\eta\right)\log^2\left(\delta^{-1}\right)$ and $l \geq 4\kappa \log\left(\frac{n}{\delta\epsilon}\right)$. Hence, we get our bound result of

$$pl = 64\kappa\left(\frac{1}{\epsilon} + \frac{1}{n\epsilon^2}\right)\log\left(2/\eta\right)\log^2\left(\delta^{-1}\right)\log\left(\frac{n}{\delta\epsilon}\right)$$

□

Appendix B: Proofs of Section 3.1. We put here the proofs that pertain to Section 3.1.

3.9. Properties of the generalized Laplacian. Proof of Lemma 3.4.

Proof. The first statement is obvious from the construction of the grounded Laplacian.

Statment (2) is a direct consequence of the fact that $F_Z = PZP^T$ with $P = (I_n\, 0)$.

Then the third statement is a simple consequence of statement 2, as $\mathrm{ld}\left(Z\right) = \sum_i \log \lambda_i$ with $\left(\lambda_i\right)_i$ the $n-1$ positive eigenvalues of Z.

Statement (4) is straightforward after observing that the floating procedure is a linear transform from $\mathcal{S}_n$ to $\mathcal{S}_{n-1}$, so it preserves the matrix inequalities.

□

3.10. Technical lemmas for Theorem 1.1. This lemma generalizes Lemma 8.1 in [26].

Lemma 3.23. *Consider $A \in \mathcal{S}_n$ positive semi-definite, and $x \in \mathbb{R}^n$. Then $xx^T \preceq \left(x^T A^+ x\right) A$*

Proof. Without loss of generality, consider $x^T x = 1$. Consider the eigenvalue decomposition of A: $A = \sum_i \lambda_i u_i u_i^T$. Since $\left(u_i\right)_i$ is an orthonormal basis of $\mathbb{R}^n$, we only need to establish that $\left(u_i^T x\right)^2 \leq \left(x^T A^+ x\right) u_i^T A u_i$ for all i. The latter term can be simplified:

$$\begin{aligned}\left(x^T A^+ x\right) u_i^T A u_i &= \left(x^T \left[\sum_j \lambda_j^{-1} u_j u_j^T\right] x\right)\lambda_i \\ &= \lambda_i \sum_j \lambda_j^{-1}\left(u_j^T x\right)^2 \\ &\geq \left(u_i^T x\right)^2\end{aligned}$$

which is the inequality we wanted.

□

Lemma 3.24. *Jensen inequality for the matrix logarithm. Let $A \in \mathcal{S}_n$ be a positive semi-definite matrix with p positive eigenvalues. Then*

$$ld\left(A\right) \leq p \log\left(\frac{Tr\left(A\right)}{p}\right)$$

Proof. This is a direct application of Jensen's inequality. Call $\left(\lambda_i\right)_i$ the positive eigenvalues of A. Then $\mathrm{ld}\left(A\right) = \sum_i \log \lambda_i$. By concavity of the logarithm:

$$\sum_i \log \lambda_i \leq p \log\left(\frac{\sum \lambda_i}{p}\right) = p \log\left(\frac{\mathrm{Tr}\left(A\right)}{p}\right)$$

□

REFERENCES

[1] Ittai Abraham, Yair Bartal, and Ofer Neiman, *Nearly tight low stretch spanning trees*, Foundations of Computer ..., (2008), pp. 781–790.

[2] Noga Alon, RM Karp, D Peleg, and Douglas West, *A graph-theoretic game and its application to the k-server problem*, SIAM Journal on Computing, 24 (1995), pp. 78–100.

[3] Peter W Atkins and Ronald S Friedman, *Molecular quantum mechanics*, Oxford university press, 2011.

[4] Zhaojun Bai, Mark Fahey, and Gene H. Golub, *Some large-scale matrix computation problems*, Journal of Computational and Applied ..., 74 (1996), pp. 71–89.

[5] Ronald Paul Barry and R. Kelley Pace, *Monte Carlo estimates of the log determinant of large sparse matrices*, Linear Algebra and its Applications, (1999).

[6] Shannon Bernardson, Paul McCarty, and Chris Thron, *Monte carlo methods for estimating linear combinations of inverse matrix entries in lattice* $\{QCD\}$, Computer Physics Communications, 78 (1994), pp. 256 – 264.

[7] Philippe de Forcrand and Rajan Gupta, *Multigrid techniques for quark propagator*, Nuclear Physics B - Proceedings Supplements, 9 (1989), pp. 516 – 520.

[8] Iain S Duff, Albert Maurice Erisman, and John Ker Reid, *Direct methods for sparse matrices*, Clarendon Press Oxford, 1986.

[9] A Duncan, E Eichten, and H Thacker, *Efficient algorithm for qcd with light dynamical quarks*, Physical Review D, 59 (1998), p. 014505.

[10] Keith D. Gremban, *Combinatorial Preconditioners for Sparse, Symmetric, Diagonally Dominant Linear Systems*, PhD thesis, Carnegie Mellon University, 1996.

[11] Ilse C F Ipsen and Dean J Lee, *Determinant approximations*, Numerical Linear Algebra with Applications (under ..., (2006).

[12] Jonathan A Kelner, Lorenzo Orecchia, Aaron Sidford, and Zeyuan Allen Zhu, *A simple, combinatorial algorithm for solving sdd systems in nearly-linear time*, in Proceedings of the 45th annual ACM symposium on Symposium on theory of computing, ACM, 2013, pp. 911–920.

[13] Ioannis Koutis, Gary L Miller, and Richard Peng, *Approaching optimality for solving SDD linear systems*, (2010), pp. 1–16.

[14] ———, *A nearly-m log n time solver for SDD linear systems*, (2011), pp. 1–16.

[15] Runze Li and Agus Sudjianto, *Analysis of computer experiments using penalized likelihood in gaussian kriging models*, Technometrics, 47 (2005).

[16] J Liu, *The Role of Elimination Trees in Sparse Factorization*, SIAM Journal on Matrix Analysis and Applications, 11 (1990), pp. 134–172.

[17] Per-Olov Löwdin, *Quantum theory of many-particle systems. iii. extension of the hartree-fock scheme to include degenerate systems and correlation effects*, Physical review, 97 (1955), p. 1509.

[18] R J Martin, *Approximations to the determinant term in Gaussian maximum likelihood estimation of some spatial models*, Communications in Statistics-Theory and Methods, 22 (1992), pp. 189–205.

[19] M McCourt, *A Stochastic Simulation for Approximating the log-Determinant of a Symmetric Positive Definite Matrix*, compare, 2 (2008), pp. 1–10.

[20] Gérard A Meurant, *Computer Solution of Large Linear Systems*, North-Holland: Amsterdam, 1999.

[21] R. Kelley Pace and Ronald Barry, *Sparse spatial autoregressions*, Statistics and Probability Letters, 33 (1997), pp. 291 – 297.

[22] Arnold Reusken, *Approximation of the Determinant of Large Sparse Symmetric Positive Definite Matrices*, SIAM Journal on Matrix Analysis and Applications, 23 (2002), p. 799.

[23] Daniel A Spielman, *Algorithms , Graph Theory , and Linear Equations in Laplacian Matrices*, tech. report, Proceedings of the International Congress of Mathematicians, Hyderabad, India, 2010.

[24] Daniel A Spielman and Nikhil Srivastava, *Graph Sparsification by Effective Resistances*, SIAM Journal on Computing, 40 (2011), pp. 1913–1926.

[25] Daniel A Spielman and Shang-Hua Teng, *A Local Clustering Algorithm for Massive Graphs and its Application to Nearly-Linear Time Graph Partitioning*, (2008).

[26] ———, *Nearly-Linear Time Algorithms for Preconditioning and Solving Symmetric , Diagonally Dominant Linear Systems*, (2009), pp. 1–48.

[27] Daniel A Spielman and Jaeoh Woo, *A Note on Preconditioning by Low-Stretch Spanning Trees*, (2009), pp. 1–4.

[28] M.J. Wainwright and M.I. Jordan, *Log-determinant relaxation for approximate inference*

in discrete Markov random fields, IEEE Transactions on Signal Processing, 54 (2006), pp. 2099–2109.

[29] Fuzhen Zhang, *The Schur complement and its applications*, 2005.

[30] Hao Zhang and Yong Wang, *Kriging and cross-validation for massive spatial data*, Environmetrics, 21 (2010), pp. 290–304.

[31] Yunong Zhang, W.E. Leithead, D.J. Leith, and L. Walshe, *Log-det approximation based on uniformly distributed seeds and its application to Gaussian process regression*, Journal of Computational and Applied Mathematics, 220 (2008), pp. 198–214.

[32] Y. Zhang and W. E. Leithead, *Approximate implementation of the logarithm of the matrix determinant in Gaussian process regression*, Journal of Statistical Computation and Simulation, 77 (2007), pp. 329–348.