Journal of Machine Learning Research 1 (2000) 1-48 Submitted 4/00; Published 10/00

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
Department of Electrical Engineering and Computer Sciences
University of California
Berkeley, CA 94720-1776, USA

Editor: Editor

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 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 in expected time[1]:

$$\tilde{\mathcal{O}}\left(m \left(\log n\right)^7 \epsilon^{-2} \log\left(\frac{n\kappa_A}{\epsilon}\right)\right)$$

where κ_A is the condition number of A. This algorithm builds upon the work of Spielmann and Teng on *ultra-sparsifiers* Spielman and Teng (2009), and it critically exploits the recent improvements from Koutis, Miller and Peng Koutis et al. (2010). 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.

The high complexity of the algorithm transpires through the large exponent of $\log n$. 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. We use the notation $\tilde{O}$ to hide a factor at most $(\log\log n)^8$

Background There are two approaches in numerical linear algebra to compute approximately 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 Meurant (1999), 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 Liu (1990). 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 Martin (1992):

$$\log|A| = \mathrm{Tr}\left(\log A\right) \tag{1}$$

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

$$\log A = \sum_{i=0}^{\infty} \frac{(-1)^i}{i+1} M^i \tag{2}$$

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 lead to looking 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 Ipsen and Lee (2006). 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.

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 Spielman and Teng (2008).

The problem of estimating determinants has important applications in spatial data analysis, statistical physics and statistics. For example, a number of quantum system properties can be inferred by computing determinants of large graphs (determinant quantum Monte Carlo). This is why computing estimates of the log-determinant has been an active problem in physics and statistics. In particular, the Martin expansion presented in Equation 1 can be combined with sampling method to estimate the trace of a matrix series (Zhang et al. (2008),McCourt (2008),Zhang and Leithead (2007)). 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 Wainwright and Jordan (2006). Our method is close in spirit to Reuksen's work Reusken (2002) by the use of a preconditioner. However,

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.

Reuksen considers preconditioners based on a clever approximation of the Cholesky decomposition, and its interaction with the eigenvalues of the complete matrix. Using simpler methods based on sampling, we are able to carefully control the spectrum of the reminder, which in turn leads to strong convergence guarantees.

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 4 will use these new concepts to introduce a first family of preconditioners based on low-stretch spanning trees. Finally, Section 5 contains the proof of our main result, an algorithm to compute determinants in near-linear time.

Algorithm **UltraLogDet**(A,ϵ):
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)
Compute s =**PreconditionedLogDetMonteCarlo**(B, A, ϵ)
Return $s + \log|D|$ +**UltraLogDet**(A',ϵ)

Algorithm 1: The main algorithm

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

Algorithm 2: PreconditionedLogDetMonteCarlo

2. Results for log-determinants

We begin by considering a simple sampling algorithm to compute log-determinants, presented first in Barry and Pace (1999). We will first present some error bounds on this algorithm that expand on bounds previously presented in Bai et al. (1996) and Barry and Pace (1999).

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

$$\log|I - S| = -\mathrm{Tr}\left(\sum_{k=1}^{\infty} \frac{1}{k} S^k\right) \tag{3}$$

This series of traces can be estimated by Monte Carlo sampling, up to precision ϵ with high probability. In order to bound the errors, we will bound the large deviation errors using Hoeffding inequality. We use a modified version of Proposition 4.2 from Bai et al. (1996):

Lemma 1 *Consider $H \in \mathcal{S}_n$ lower- and upper-bounded by λ_{min} and λ_{max}: $\lambda_{\min} I \preceq H \preceq \lambda_{\max} I$. 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 $\nu > 0$:*

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

Proof This is a simple adaptation from Bai et al. (1996). ■

The previous lemma shows that if the eigenspectrum of a matrix is bounded, we can obtain a Hoeffding bound on the error done by sampling the trace. Furthermore, the convergence of the series 3 is also determned by the extremal eigenvalues of S. If we truncate the series (3), 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 Barry and Pace (1999). While that main Theorem in Barry and Pace (1999) only considered a confidence interval based on the covariance properties of Gaussian distribution, we generalize this result to more general Hoeffding bounds.

Theorem 2 *Consider $S \in \mathcal{S}_n^+$ with $0 \preceq S \preceq (1-\delta) I$ for some $\delta \in (0,1)$. Call y the quantity to estimate:*

$$y = \log|I - S|$$

and consider $\mathbf{u}_i \sim \mathcal{N}(\mathbf{0}, I_n)$ for $i = 1 \cdots p$. Call $\hat{y}_{p,l}$ an estimator of the truncated series of l elements computed by sampling the trace using p samples:

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

By choosing $p \geq 4\epsilon^{-2}(\log\delta)^2 \log n$ and $l \geq 2\delta^{-1}\log\left(\frac{n}{\delta\epsilon}\right)$, the error probability is small:

$$\mathbb{P}\left[|y - \hat{y}_{p,l}| \geq \epsilon\right] \leq \frac{1}{n}$$

Proof The proof of this theorem follows the proof of the Main Theorem in Barry and Pace (1999) 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, we have for all $k \in \mathbb{N}$:

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

Using again triangle inequality, we can bound the error with respect to the expected value:

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

$$\begin{aligned} &\leq \frac{n}{l+1}\frac{(1-\delta)^{l+1}}{\delta} \\ &\leq n\frac{(1-\delta)^{l+1}}{\delta} \end{aligned}$$

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

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

The lowest eigenvalue of the truncated series can be lower-bounded in terms of δ:

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

We can now invoke Lemma 1 to conclude. Indeed, call $\nu = 2^{-1}\epsilon p$, then $\mathbb{P}\left[\left|\frac{1}{p}\sum_{i=1}^{p}\mathbf{u}_i^T H\mathbf{u}_i^T - \mathrm{Tr}\left(H\right)\right| \geq \frac{\epsilon}{2}\right] \leq 2\exp\left(-\frac{p\epsilon^2}{2(\log\delta)^2}\right)$ and $2\exp\left(-\frac{p\epsilon^2}{2(\log\delta)^2}\right) \leq n^{-1}$ is equivalent to $-\frac{p\epsilon^2}{2(\log\delta)^2} \leq -\log 2 - \log n \leq -2\log n$, and this latter inequality is verified for any $p \geq 4\epsilon^{-2}\left(\log\delta\right)^2\log n$. ■

We immediately get the following result that justifies the notion of preconditioners for determinants. The corresponding algorithm, which we call `PreconditionedLogDetMonteCarlo`, is presented in Algorithm 2.

Corollary 3 *If A and B are positive definite matrices so that B is a $\kappa-$approximation of A:*

$$A \preceq B \preceq \kappa A \tag{4}$$

then the algorithm `PreconditionedLogDetMonteCarlo` computes an ϵ-approximation of $\log\left|B^{-1}A\right|$ with high probability, by performing $O\left(\kappa\epsilon^{-2}\left(\log\kappa\right)^2\log n\left(\log n + \log\kappa + \log\left(\epsilon^{-1}\right)\right)\right)$ vector inversions from B and vector multiplies from A.

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 of the positive-definite matrix B[3]. The inequality 4 is equivalent to $\kappa^{-1}B \preceq A \preceq B$, or also:

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

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

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 $I - S$:

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

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 $Y^2 = X$.

and invoking Theorem 2 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 $l \in \mathbb{N}^*$, consider the truncated series:

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

Hence, the practical computation of the latter sum can be done on $A^{-1}B$. To conclude, if we compute $p = 4\epsilon^{-2} \left(\log \kappa\right)^2 \log n$ truncated chains of length $l = \kappa \left(2 \log n + \log \kappa + \log\left(\epsilon^{-1}\right)\right)$, we get our result. This requires lp multiplies by A and inversions by B. ■

Intuitively, this scheme is useful if B is easy to inverse and has good spectral properties (κ is small). We are going to present some algorithms for a class of matrices (symmetric, diagonally dominant matrices) that enjoy such good properties.

3. Making the problem laplacian

From now on, we consider all matrices to be in SDD_n. Most techniques we will be using 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 4 Pseudo-log-determinant (PLD): *Let $A \in \mathcal{S}^{n+}$ be a positive semi-definite matrix of which the nullspace is of rank at most 1 (i.e., it has at most one zero eigenvalue). The pseudo-log-determinant is defined by the sum over all the eigenvalues except the last one:*

$$ld(A) = \sum_{i=2}^{n} \log\left(\lambda_i\right)$$

where λ_i are the eigenvalues of A sorted in increasing order.

The matrix logarithm does not converge for matrices that are not invertible, so we need to extend its definition.

Definition 5 Pseudo-matrix logarithm. *Consider $A \in \mathcal{S}_n^+$ so that $0 \preceq A \prec 2$. The pseudo-logarithm of a matrix is defined by the projection of the matrix logarithm onto the span of A:*

$$\log^+ A = \sum_k \frac{1}{k} P \left(I - A\right)^k P^T$$

with P an orthogonal projector onto span(A).

The interest of the PLD lies in the connection between SDD matrices and their Laplacian. Recall that a Laplacian matrix can be constructed from any SDD matrix by adding a "ground" node (see Gremban's thesis Gremban (1996), Chapter 4).

Definition 6 Augmented Laplacian of an SDD matrix: *Let $A \in SDD_n$. Let L, D be the decomposition of A into L a Laplacian matrix and D a (non-negative) diagonal matrix such that $A = L + D$. The augmented matrix of A is:*

$$L_A = \begin{pmatrix} A & -\mathbf{d} \\ -\mathbf{d}^T & h \end{pmatrix}$$

with $\mathbf{d} \in \mathbb{R}^n$: $\mathbf{d}_i = D_{ii}$ and $h = \sum_i D_{ii}$.

From now on, given any $A \in SDD_n$, we will implicitly define a Laplacian L_A associated to it, and call the graph of A the weighted undirected graph characterized by L_A. The PLD enjoys some of the usual properties of the log-determinant. The following proposition shows the interest of the PLD in the study of the Laplacian of a graph.

Proposition 7 Let $A \in SDD_n$, and L_A its augmented Laplacian. Let $P = (I_n \, \mathbf{0})$ be a projection matrix over the first n columns of L_A. Then:

- If λ is an eigenvalue of A with associated eigenvector x, it is also an eigenvalue of PL_AP^T with the same eigenvector
- $\mathrm{ld}\left(L_A\right) = \log|A|$
- Let $A, B \in SDD_n$. $\mathrm{ld}\left(L_A L_B\right) = \mathrm{ld}\left(L_A\right) + \mathrm{ld}\left(L_B\right)$ and $\mathrm{ld}\left(L_B^+ L_A\right) = \mathrm{ld}\left(L_A\right) - \mathrm{ld}\left(L_B\right) = -\mathrm{ld}\left(L_A^+ L_B\right)$ with $A^+$the pseudo inverse of A.
- Let $A \in SDD_n$ with $\|I - A\| < 1$ then $\mathrm{ld}\left(L_A\right) = -\mathrm{Tr}\left(\log^+ L_A\right)$ with $\log^+ L_A$ the pseudo-logarithm of L_A

Note in particular that the PLD is well defined over connected graphs. From now on, we will consider connected graphs only. If graphs have several unconnected components, the log-determinant of the corresponding SDD is the sum of the PLDs of the Laplacians of each component.

Lemma 8 *The pseudo-log-determinant of a connected graph is well-defined.*

Proof A connected graph G has non-zero conductance Φ_G and using Cheeger's inequality (see Theorem 4.1 in Spielman (2008)) $\lambda_2 \geq \Phi_G^2 > 0$. Hence $\lambda_i \geq \lambda_2 > 0$ for all $i \geq 2$ and $\mathrm{ld}\left(G\right) \in \mathbb{R}$. ■

From now on, we will focus on computing the PLD of Laplacians. Note that the kernel of all Laplacian matrices of connected graphs is precisely $\mathbb{R}\mathbf{1}_n$, so we can use the PLD with pseudo inverses as if we were using the log determinant with SDD matrices: Consider a graph $G = (V, E, w)$, identified with its Laplacian matrix L_G. Consider a weighted subgraph $H = \left(V, \tilde{E}, \tilde{w}\right)$ of G, for which it is easier to compute the determinant, and that closely approximates G in the spectral sense. From Proposition 7:

$$\mathrm{ld}\left(G\right) = \mathrm{ld}\left(H\right) + \mathrm{ld}\left(H^+ G\right)$$

We will see how to adapt Spielman and Teng's remarkable work on *ultra-sparsifiers* to produce good preconditioners H for the determinant. In particular, once a good preconditioners is available for G, we present an algorithm that computes an upper bound and a lower bound of the PLD *for free*. We will improve this results to produce an algorithm that computes an ϵ-approximation with high probability.

4. A first preconditioner

We now 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. Alon et al. (1995). The bound of Alon et al. was then improved by Abraham et al. Abraham et al. (2008). We restate their main result.

Lemma 9 *Consider G a weighted graph. There exists a tree T that is a subgraph of G so that:*

$$G \preceq T \preceq \kappa G$$

with $\kappa = Cm \log n \log\log n \left(\log\log\log n\right)^3$ for some constant $C > 0$.

Proof This follows directly from Abraham et al. (2008). T is a subgraph of G (with the same weights on the edges), so $T \preceq G$. Furthermore, we have $T \preceq \mathrm{st}_G(T)\, G$ using a result of Boman and Hendrickson Boman et al. (2004) cited by Spielman in Spielman (2010). ■

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 Spielman and Teng (2009). This factorization can be expressed as:

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

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 Spielman and Teng (2009). Once the Cholesky factorization of the tree is performed, the log-determinant of the original graph is an immediate by-product:

$$\mathrm{ld}(T) = \sum_{i=1}^{n-1} \log D_{ii}$$

Furthermore, computing $T^+ x$ also takes $O(n)$ computations by forward-backward substitution. Applying Corollary 3 gives immediately:

Theorem 10 *Let G be a graph with n vertices and m edges. Its PLD can be computed up to a precision of ϵ and with high probability in time $\tilde{O}\left(mn \left(\log m\right)^3 \left(\log n\right)^2 \frac{\log \epsilon^{-1}}{\epsilon^2}\right)$.*

Proof We assume G is connected, hence $m \geq n$ and $\log \kappa = \tilde{\mathcal{O}}(\log m)$. ■

This bound may be of independent interest since it requires relatively little machinery to compute, and it is an improvement already for graphs with small vertex degree ($m = \mathcal{O}\left(n^{1+o(1)}\right)$) over the Cholesky factorization of G.

How good is the estimate provided by the tree T? Intuitively, this depends on how well the tree T approximates the graph G. This notion of quality of approximation can be formalized by the notion of *stretch*.

Definition 11 *Stretch of a graph. Consider G a weighted graph, and T a spanning tree of G. The stretch of each edge e from G is defined as $st_T(e) = \omega_e \left(\sum_{f \in P} \frac{1}{\omega_f}\right)$ where ω_e is the weight of edge*

e in G, and P is the unique path between the endpoints of e in the tree T. The stretch of the graph is the sum of the stretches of each edge:

$$st_T(G) = \sum_{e \in G} st_T(e)$$

The stretch can be obtained as a by-product of the computation of low-stretch spanning trees, of which the construction can be done in $O\left(m \log n + n \log^2 n\right)$ Abraham et al. (2008). Knowing the stretch gives upper and lower bounds on the value of the log-determinant:

$$\mathrm{ld}(T) + (n-1)\log\left(\frac{\mathrm{st}_G(T)}{n-1}\right) \geq \mathrm{ld}(G) \geq \mathrm{ld}(T) + \log\left(\mathrm{st}_T(G) - n + 2\right) \tag{5}$$

Proof This is an application of Jensen's inequality on $\mathrm{ld}\left(T^+ G\right)$. We have $\mathrm{ld}G = \mathrm{ld}T + \mathrm{ld}\left(T^+ G\right)$ and $\mathrm{ld}\left(T^+ G\right) = \mathrm{ld}\left(\sqrt{T}^+ G \sqrt{T}^+\right) = \mathrm{Tr}\left(\log^+\left(\sqrt{T}^+ G \sqrt{T}^+\right)\right)$ with $\sqrt{T}$ the matrix square root of T. From Lemma 20, we have

$$\begin{aligned}
\mathrm{Tr}\left(\log^+\left(\sqrt{T}^+ G \sqrt{T}^+\right)\right) &\leq (n-1)\log\left(\frac{\mathrm{Tr}\left(\sqrt{T}^+ G \sqrt{T}^+\right)}{n-1}\right) \\
&= (n-1)\log\left(\frac{\mathrm{Tr}\left(T^+ G\right)}{n-1}\right) \\
&= (n-1)\log\left(\frac{\mathrm{st}_G(T)}{n-1}\right)
\end{aligned}$$

The lower bound is slightly more involved. Call λ_i the positive eigenvalues of $\sqrt{T^+} G \sqrt{T^+}$ and $\sigma = \mathrm{st}_T(G)$. We have $1 \leq \lambda_i \leq \sigma$ because of the inequality$T \preceq G \preceq \mathrm{st}_T(G)\, T$. There are precisely $n-1$ positive eigenvalues λ_i, and we have: $\mathrm{ld}\left(T^+ G\right) = \sum_i \log \lambda_i$. One can show that $\sum_i \log \lambda_i \geq \log(\sigma - n + 2)$ by considering the problem of minimizing $\sum_i \log x_i$ under the constraints $\sum_i x_i = \sigma$ and $x_i \geq 1$. This bound is tight. ■

Intuitively since the stretch of a tree is bounded my $O(m \log n)$, it means that a low-stretch spanning tree provides an approximation of a log-determinant up to a factor of $O(n \log n)$. This could be of interest for nearly-cut graphs, with large condition numbers.

5. 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 that furthermore they are excellent spectral preconditioners, so the procedure `PreconditionedLogDetMonteCarlo` is efficient to apply. We reintroduce some concepts presented in Koutis et al. (2010) to present a self-contained result.

The central idea 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 Koutis et al. (2010). This section is based on Section 4 of Spielman and Teng (2009).

Consider a Laplacian matrix G. There exists an algorithm that computes the partial Cholesky factorization:

$$B = PLCL^T P^T$$

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

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

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

$$\mathrm{ld}\,(B) = \sum_{1}^{n-n_1} \log D_{ii} + \mathrm{ld}\,(A_1)$$

Thus, the PLD of the original Laplacian A is:

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

We are left with solving a smaller problem A_1, and approximate the value of $\mathrm{ld}\,(B^+ A)$ 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 $O(\log\log n)$ steps. Furthermore, computing the vector product $B^+ Ax$ is itself efficient (in can be done in near-linear time), so the sampling algorithm can be run is reasonable time. We formalize the notion of chain of preconditioners by reintroducing some material from Koutis et al. (2010).

Definition 12 *Definition 7.1 from Koutis et al. (2010). $\kappa(n)$-good chain. $\mathcal{C} = \{A_1 = A, B_1, A_2, \dots A_d\}$ a chain of graphs with n_i and m_i the number of vertices and edges of A_i. $\mathcal{C}$ is $\kappa(n)$-good for A if:*

1. $A_i \preceq B_i \preceq \kappa(n_i) A_i$
2. A_{i+1}= GreedyElimination(B_i)
3. $m_i/m_{i+1} \geq c_r \sqrt{\kappa(n_i)}$ for some constant c_r.

Good chains exist (Lemma 7.3 from Koutis et al. (2010)):

Lemma 13 *Given a graph A, `BuildChain`(A, p) from Koutis et al. (2010) produces a $\tilde{O}\left(\log^4 n\right)$-good chain for A with probability $1-p$. The algorithm runs in time*

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

These chains furthermore can be used a good preconditioners for conjugate gradient and lead to near-linear algorithms for approximate inversion (Lemma 7.2 from Koutis et al. (2010))

Lemma 14 *Given a $\kappa(n)$-good chain for A, a vector x such that $\left\|x - L_A^+ b\right\|_A < \epsilon \left\|L_A^+ b\right\|_A$ can be computed in $O\left(m_d^3 m_1 \sqrt{\kappa(n_1)} \log(1/\epsilon)\right)$*

It should now become clear how we can combine a $\kappa(n)$-good chain with the Algorithm `PreconditionedLogDetMonteCarl` We start by building a chain. The partial Cholesky factorizations 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 1.

We have three sources of errors introduced in our estimate:

- The truncation of the series of $\log(B^+ A)$
- The sampling error induced by having a finite number of samples when approximating the trace of $\log(B^+ A)$
- The approximate inversion by B

The following lemmas show that we can reasonably control the inversion error by running the ST-solver at a higher precision $O\left(\kappa(n_1)^{-3}\kappa(A_1)^{-1}\epsilon\right)$. In particular, the dependency on the condition number of A_1 stems from the fact that the convergence guarantees offered by the ST-solver are relative to the matrix norm $\|\|_A$. The ST-solvers present some convergence guarantees using the matrix norm induced by A instead of the regular Euclidian norm. The following lemmas deal with this technicality.

Proofs of convergence

We need to prove that the error introduced by approximating the solution of the preconditioner does not have too great an impact on the accuracy of the solution. We use the same notations as above with B a $\kappa-$approximation of A, and we call:

$$A \preceq B \preceq \kappa A$$

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

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

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

Lemma 15 *The matrix norm of the elements is bounded and less than* 1*:*

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

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

The second inequality is a consequence of Proposition 3.3 from Spielman and Teng (2009): A and B have the same nullspace and we have the LMI $A \preceq B \preceq \kappa A$, 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 $1-\varphi$ and 0.

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

$$\|R\|_B = \max_{x \neq 0} \frac{\|Rx\|_B}{\|x\|_B}$$

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

and the latter expression simplifies:

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

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 $\left(R^k x\right)_k$. These chains are computed approximately by repeated applications of the R operator on the previous element of the chain, starting from a Gaussian white noise element $x_0 \sim \mathcal{N}(0, \mathbf{I})$. We formalize the notion of an approximate chain.

Definition 16 Approximate power sequence. *Given a linear operator R, a start point $x^{(0)} \in \mathbb{R}^n$, and a positive-definite matrix B, we define an $\epsilon-$approximate power sequence as a sequence that does not deviate too much from the power sequence:*

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

We now prove the following result that is quite intuitive: if the operator R 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 17 *Given the previous hypothesis, and assuming furthermore that $\|R\|_B \leq 1 - \eta$ and that $2\epsilon \leq \eta \leq 1/2$, the total error is bounded by $\mathcal{O}\left(\eta^{-3}\epsilon\right)$:*

$$
\sum_{k=0}^{\infty} \left\|x^{(k)} - R^k x^{(0)}\right\|_B \leq 4\epsilon\eta^{-3} \left\|x^{(0)}\right\|_B
$$

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

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

We now bound the error on the ω_k sequence:

$$
\begin{aligned}
\omega_{k+1} &= \left\|x^{(k+1)} - R x^{(k)} + R x^{(k)} - R^{k+1} x^{(0)}\right\|_B \\
&\leq \left\|R x^{(k)} - R^{k+1} x^{(0)}\right\|_B + \left\|x^{(k+1)} - R x^{(k)}\right\|_B
\end{aligned}
$$

$$\begin{aligned}
&\leq \|R\|_B \left\|x^{(k)} - R^k x^{(0)}\right\|_B + \epsilon \left\|Rx^{(k)}\right\|_B \\
&= \|R\|_B \omega_k + \epsilon\theta_k \\
&\leq \|R\|_B \omega_k + \epsilon \left(\omega_k + \|R\|_B^k \left\|x^{(0)}\right\|_B\right) \\
&\leq \left[(1-\eta)^2 + \epsilon\right] \omega_k + \epsilon (1-\eta)^{2k} \left\|x^{(0)}\right\|_B
\end{aligned}$$

Note that the condition $(1-\eta)^2 + \epsilon \leq 1-\eta$ is equivalent to $\epsilon \leq \eta - \eta^2$. We can assume without loss of generality that $0 \leq \eta \leq 1/2$. Under this condition, $\eta/2 \leq \eta - \eta^2$. So the condition $2\epsilon \leq \eta$ implies $(1-\eta)^2 + \epsilon \leq 1-\eta$ which leads to:

$$\omega_{k+1} \leq (1-\eta)\omega_k + \epsilon (1-\eta)^{2k} \left\|x^{(0)}\right\|_B$$

By induction, one obtains:

$$\omega_k \leq \omega_1 (1-\eta)^k + \frac{\epsilon \left\|x^{(0)}\right\|_B}{\eta} (1-\eta)^{2k-1}$$

and since $\omega_1 = \left\|x^{(1)} - Rx^{(0)}\right\|_B \leq \epsilon \left\|Rx^{(0)}\right\|_B \leq \epsilon \|R\|_B \left\|x^{(0)}\right\|_B \leq \epsilon (1-\eta)^2 \left\|x^{(0)}\right\|_B$, we have a final bound that depends on ϵ:

$$\omega_k \leq \epsilon (1-\eta)^2 \left\|x^{(0)}\right\|_B (1-\eta)^k + \frac{\epsilon \left\|x^{(0)}\right\|_B}{\eta} (1-\eta)^{2k-1}$$

This is the sum of two geometric series, which give the bound:

$$\sum_k \omega_k \leq \epsilon\eta^{-1} \left[1 + \frac{1}{\eta^2 (1-\eta)^2}\right] \left\|x^{(0)}\right\|_B$$

Since $\eta \leq 1/2$, it implies $(1-\eta)^{-2} \leq 4$ and $1 \leq \eta^{-2}$, so we can further simplify:

$$\sum_k \omega_k \leq 4\epsilon\eta^{-3} \left\|x^{(0)}\right\|_B$$

■

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

Proposition 18 *Given a $\kappa(n)$-good chain for A, a start vector x, one can compute an $\epsilon-$estimate of the truncated series:*

$$Tr\left(\sum_{i=1}^{l} \frac{1}{i} \left(I - B_1^{-1}A\right)^i xx^T\right)$$

in time $\mathcal{O}\left(lm\sqrt{\kappa(n_1)}\left(\log \epsilon^{-1} + \log \kappa(n_1) + \log \kappa(B)\right)\right)$.

Proof Consider an $\epsilon-$approximate power sequence $\left(x^{(k)}\right)_k$ with respect to the operator $I - B^{-1}A$, and $\hat{z}$ the truncated sequence:

$$\hat{z} = \sum_{k=1}^{l} \frac{1}{k} \left(x^{(0)}\right)^T x^{(k)}$$

This sequence is an approximation of the exact sequence z:

$$z = \mathrm{Tr}\left(\sum_{i=1}^{l} \frac{1}{i} \left(I - B^{-1}A\right)^i x^{(0)} \left(x^{(0)}\right)^T\right)$$

We now bound the error between the two sequences:

$$|\hat{z}-z| \leq \sum_{k=1}^{l} \frac{1}{i} \left|x_0^T \left(R^k x_0 - x_k\right)\right| \leq \sum_{k=1}^{l} \left|x_0^T \left(R^k x_0 - x_k\right)\right| \leq \sum_{k=1}^{l} \left|\left(B^{-1}x_0\right)^T B \left(R^k x_0 - x_k\right)\right|$$

Using the Cauchy-Schwartz inequality, we obtain:

$$\left|\left(B^{-1}x_0\right)^T B \left(R^k x_0 - x_k\right)\right| = \left|\left\langle B^{-1}x_0, R^k x_0 - x_k\right\rangle_B\right| \leq \left\|B^{-1}x_0\right\|_B \left\|R^k x_0 - x_k\right\|_B$$

so that we can bound the deviation:

$$|\hat{z}-z| \leq \left\|B^{-1}x_0\right\|_B \sum_{k=1}^{l} \left\|R^k x_0 - x_k\right\|_B \leq 4\epsilon\kappa^3 \left\|B^{-1}x_0\right\|_B \|x_0\|_B \leq 4\epsilon\kappa^3\kappa\left(B\right)\|x_0\|^2$$

where $\kappa(B)$ is the condition number of B. Now, we can call the ST-solver routine with $\tilde{\epsilon} = \kappa^{-3}\kappa(B)^{-1}\epsilon$, the cost of each call to the operator R being then $\mathcal{O}\left(m\sqrt{\kappa}\left(\log\epsilon^{-1} + \log\kappa + \log\kappa(B)\right)\right)$. ■

We now have all the elements required to prove our main theorem. Now we can use the previous proposition to bound the error done during the estimation part. We have already found some bounds for the estimates of the log-determinant of the preconditioners, and some probability estimates for the error bounds on the trace series.

Theorem 19 *Consider A a symmetric, diagonally dominant matrix of size $n \times n$ with m non-zero entries, and $0 < \epsilon < \log^{-5} n$. One can compute $y \in \mathbb{R}$ so that $\mathbb{P}\left(|y - \log|A|| > \epsilon\right) \leq \frac{1}{n}$ in expected time $\tilde{O}\left(m\epsilon^{-2}\log^7 n\left(\log\epsilon^{-1} + \log n + \log\kappa(A)\right)\right)$.*

Proof Consider a $\tilde{O}\left(\log^4 n\right)$-chain 13 for the Laplacian of A. The cost of constructing such a chain is negligible against the sampling step. From Corollary 3, for each level of the chain, we compute $p = \tilde{O}\left(\epsilon^{-2}\log n\right)$ approximate truncated chains of length $l = \tilde{O}\left(\log^4 n\left(\log n + \log\left(\epsilon^{-1}\right)\right)\right)$. From Proposition 18, the cost of running each chain is $\tilde{O}\left(lm\log^2 n\left(\log\epsilon^{-1} + \log n + \log\kappa(A)\right)\right)$. Thus the time to approximate the residue PLD at level i is bounded by $\tilde{O}\left(m\epsilon^{-2}\log^7 n\left(\log\epsilon^{-1} + \log n + \log\kappa(A)\right)\right)$. Finally, since m_i decreases faster than geometrically, the number d of steps in the chain is $\tilde{O}(1)$. ■

Comments

A first approximation upper-bound of the log-determinant follows immediately from the computation of the $\kappa-$good chain. We presented in Section 4 a first analysis to bound the value of the residue PLD. This analysis was done using trees as preconditioners, it can be carried on on more general preconditioners by introducing a generalization of the stretch over a subgraph (see the Appendix, section 5.1). 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.

When looking at each step of the analysis, one can see that the ϵ^{-2} factor comes from the approximation of the trace of a matrix by sampling. This seems to be a fundamental limitation of this method and it is shared with other algorithms that rely on random projections. This result is absolute. It would be interesting to see if the analysis could be tightened to present a relative bound that does not depend on the condition number of A. 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 Jim Demmel for suggesting the original idea and their helpful comments on the draft of this article.

Appendix

Lemma 20 *Jensen inequality for the matrix logarithm. Given $A \in \mathcal{S}_n^+$, $0 \prec A \prec 2$, the following inequalities hold:*

$$\log\left(\frac{Tr(A)}{n}\right) \geq \frac{1}{n} Tr(\log A)$$

$$ld\left(\frac{Tr(A)}{n-1}\right) \geq \frac{1}{n-1} Tr(\log A)$$

with $\log A = \sum_{k\geq 1} \frac{1}{k} (I-A)^k$

Proof Consider the diagonalization of A: $A = P\Delta P^T$ with Δ a diagonal (positive) matrix, and P an orthogonal matrix. Then

$$\log A = \sum_{k\geq 1} \frac{1}{k} \left(PP^T - P\Delta P^T\right)^k = \sum_{k\geq 1} \frac{1}{k} \left[P(I-\Delta)P^T\right]^k = P\left[\sum_{k\geq 1} \frac{1}{k}(I-\Delta)^k\right] P^T = P\Gamma P^T$$

with Γ a diagonal matrix that verifies $\Gamma_{ii} = \log \Delta_{ii}$. We can then conclude using the concavity of the logarithm over the reals:

$$\log\left(\frac{\mathrm{Tr}(A)}{n}\right) = \log\left(\frac{\sum_i \Delta_{ii}}{n}\right) \geq \frac{1}{n}\sum_i \log \Delta_{ii} = \frac{1}{n}\mathrm{Tr}(\Gamma) = \frac{1}{n}\mathrm{Tr}(\log A)$$

The same reasoning holds for the pseudo-log-determinant while considering all but one eigenvalue ■

Lemma 21 *Power relation for the matrix log: Given $A \in \mathcal{S}_n^+$, $0 \prec A \prec 2$, and $k \in \mathbb{N}$, then* $\log\left(A^k\right) = k \log A$

5.1 Stretch of a graph

We introduce here a refinement on an upper bound that is a byproduct of using the algorithm XXX.

Definition 22 *We define the **stretch of a graph** $G = (V, E, \omega)$ with respect to a subgraph $H = \left(V, \tilde{E}, \tilde{\omega}\right) \subset G$ by the weighted sum of effective resistances of edges $e \in E$ with respect to the graph H:*

$$st_H(G) = \sum_{e\in E} \omega_e eff_H(e)$$

This is a generalization of the notion of stretch defined by Alon, Karp, Peleg and West in .... We can use this definition to generalize theorem 2.1 in Spielman and Woo (2009)

Theorem 23 *Let $G = (V, E, \omega)$ be a connected graph and let $H = \left(V, \tilde{E}, \tilde{\omega}\right)$ be a connected subgraph of G. Let L_G and L_H be the Laplacian matrices of G and H respectively. Then:*

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

where L_H^+ is the pseudo inverse of L_H.

Proof The proof is nearly identical to that of Spielman and Woo (2009), except for the last line:

$$\begin{aligned}
\mathrm{Tr}\left(L_H^+ L_G\right) &= \sum_{(u,v)\in E} \omega\left(u,v\right) \mathrm{Tr}\left(L_{(u,v)} L_H^+\right) \\
&= \sum_{(u,v)\in E} \omega\left(u,v\right) \mathrm{Tr}\left(\left(\psi_u - \psi_v\right)\left(\psi_u - \psi_v\right)^T L_H^+\right) \\
&= \sum_{(u,v)\in E} \omega\left(u,v\right) \left(\psi_u - \psi_v\right)^T L_H^+ \left(\psi_u - \psi_v\right)
\end{aligned}$$

and the latter term is the effective resistance between u and v in the graph H:

$$\begin{aligned}
&= \sum_{(u,v)\in E} \omega\left(u,v\right) \mathrm{eff}_H\left(u,v\right) \\
&= \mathrm{st}_H\left(G\right)
\end{aligned}$$

■

From a practical perspective, the graph stretch can be computed in $\tilde{O}\left(m \log n/\epsilon^2\right)$

Proposition 24 *There exists an algorithm that computes* an ϵ-approximation of $\mathrm{st}_H\left(G\right)$ in $\tilde{O}\left(m \log n/\epsilon^2\right)$

References

Ittai Abraham, Yair Bartal, and Ofer Neiman. Nearly tight low stretch spanning trees. *Foundations of Computer ...*, pages 781–790, 2008. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4691010.

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(1):78–100, 1995. doi: 10.1137/S0097539792224474. URL http://epubs.siam.org/doi/abs/10.1137/S0097539792224474 http://epubs.siam.org/doi/pdf/10.1137/S0097539792224474.

Zhaojun Bai, Mark Fahey, and Gene H. Golub. Some large-scale matrix computation problems. *Journal of Computational and Applied ...*, 74(1):71–89, 1996. doi: 10.1.1.56.8150. URL http://www.sciencedirect.com/science/article/pii/0377042796000180.

Ronald Paul Barry and R. Kelley Pace. Monte Carlo estimates of the log determinant of large sparse matrices. *Linear Algebra and its Applications*, 1999. URL http://www.sciencedirect.com/science/article/pii/S002437959710009X.

Erik G. Boman, Doron Chen, Bruce Hendrickson, and Sivan Toledo. Maximum-weight-basis preconditioners. *Numerical Linear Algebra with Applications*, 11(89):695–721, October 2004. ISSN 1070-5325. doi: 10.1002/nla.343. URL http://doi.wiley.com/10.1002/nla.343.

Keith D. Gremban. *Combinatorial Preconditioners for Sparse, Symmetric, Diagonally Dominant Linear Systems.* PhD thesis, Carnegie Mellon University, 1996. URL www.cs.cmu.edu/ glmiller/Publications/GrembanPHD.ps.gz.

Ilse C F Ipsen and Dean J Lee. Determinant approximations. *Numerical Linear Algebra with Applications (under ...*, (X), 2006. URL http://www.ncsu.edu/crsc/reports/ftp/pdf/crsc-tr03-30.pdf.

Ioannis Koutis, Gary L Miller, and Richard Peng. Approaching optimality for solving SDD linear systems. pages 1–16, 2010.

J Liu. The Role of Elimination Trees in Sparse Factorization. *SIAM Journal on Matrix Analysis and Applications*, 11(1):134–172, 1990. doi: 10.1137/0611010. URL http://epubs.siam.org/doi/abs/10.1137/0611010.

R J Martin. Approximations to the determinant term in Gaussian maximum likelihood estimation of some spatial models. *Communications in Statistics-Theory and Methods*, 22(1):189–205, 1992.

M McCourt. A Stochastic Simulation for Approximating the log-Determinant of a Symmetric Positive Definite Matrix. *compare*, 2:1–10, 2008. URL http://www.thefutureofmath.com/mathed/logdet.pdf.

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

Arnold Reusken. Approximation of the Determinant of Large Sparse Symmetric Positive Definite Matrices. *SIAM Journal on Matrix Analysis and Applications*, 23(3):799, 2002. ISSN 08954798. doi: 10.1137/S089547980036869X. URL http://link.aip.org/link/SJMAEL/v23/i3/p799/s1&Agg=doi.

Daniel A Spielman. Spectral sparsification of graphs. *Arxiv preprint arXiv:0808.4134*, 2008. URL http://arxiv.org/abs/0808.4134.

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

Daniel A Spielman and Shang-Hua Teng. A Local Clustering Algorithm for Massive Graphs and its Application to Nearly-Linear Time Graph Partitioning. 2008.

Daniel A Spielman and Shang-Hua Teng. Nearly-Linear Time Algorithms for Preconditioning and Solving Symmetric , Diagonally Dominant Linear Systems. pages 1–48, 2009.

Daniel A Spielman and Jaeoh Woo. A Note on Preconditioning by Low-Stretch Spanning Trees. pages 1–4, 2009.

M.J. Wainwright and M.I. Jordan. Log-determinant relaxation for approximate inference in discrete Markov random fields. *IEEE Transactions on Signal Processing*, 54 (6):2099–2109, June 2006. ISSN 1053-587X. doi: 10.1109/TSP.2006.874409. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1634807.

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(4):329–348, April 2007. ISSN 0094-9655. doi: 10.1080/10629360600569279. URL http://www.tandfonline.com/doi/abs/10.1080/10629360600569279.

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(1-2): 198–214, October 2008. ISSN 03770427. doi: 10.1016/j.cam.2007.08.012. URL http://linkinghub.elsevier.com/retrieve/pii/S0377042707004360.