CSC 2420 : Paper Presentation[Kyng16] Presentation : December 22, 2022

Approximate Gaussian Elimination for Laplacians - Fast, Sparse and Simple

Instructor: Sushant Sachdeva *Present/scribe: Zhiyang Chen, Mohan Zhao*

1 Problem Statement

The problem of solving systems of linear equations $Lx = b$, where L is an SDD matrix (and often a Laplacian) is a fundamental primitive and arises in different applications.

A natural approach to solving systems of linear equations is Gaussian elimination, or its variant for symmetric matrices, Cholesky factorization, which produces a factorization $L = \mathcal{L}\mathcal{D}\mathcal{L}^\top$ where $\mathcal{L}$ is a lower-triangular system and $\mathcal{D}$ is a diagonal matrix. However, $\mathcal{L}$ can be a dense matrix even when the original matrix L is sparse. This phenomenon is called *fill-in*.

For Laplacian systems, sequentially eliminating variables often produces a sequence of increasingly-dense systems, resulting in an $O(n^3)$ worst-case time even for sparse L.

This paper[**Kyng16**] presents the first nearly linear time algorithm that generate a **sparse** approximate Cholesky decomposition for (general!) Laplacian matrices, with provable approximation guarantees. It does not use any graph theoretic constructions, only uses random sampling.

2 Notation

In the rest of the note, we will use the following notation:

- $S^{(k)}$: Schur complement at iteration k.
- Cl_k: Laplacian of the clique added at iteration k of Cholesky Factorization.
- $(L)_v$: Laplacian of a graph with all edges incident on v.
- $L_{(u,v)}$: Laplacian of a graph with only one edge between u and v. $L_{(u,v)} = b_{u,v}b_{u,v}^\top$
- For a matrix M, $\overline{M} = L^{-\frac{1}{2}}ML^{-\frac{1}{2}}$, where L is the Laplacian of the original graph.

3 Preliminaries

Definition 3.1. *A symmetric matrix L is called Symmetric and Diagonally Dominant (SDD) if for all i, $L(i,i) \geq \sum_{j\neq i} |L(i,j)|$.*

Definition 3.2. *An SDD matrix L is a Laplacian if $L(i,j) \leq 0$ for $i \neq j$, and for all i, $\sum_j L(i,j) = 0$. A Laplacian matrix is naturally associated with a graph on its vertices, where i, j are adjacent if $L(i,j) \neq 0$.*

Definition 3.3. *Consider a connected undirected multi-graph $G = (V,E)$, let e_i denote the i^{th} standard basis vector. Given an edge $e = (u,v)$, we define pair vector $b_e = b_{u,v} = e_v - e_u$. Also, we define $L_{(u,v)} = b_{u,v}b_{u,v}^\top$*

Definition 3.4. *A graph in which multiple edges may connect the same pair of vertices is called a* multi-graph. *Since there can be multiple edges between the same pair of vertices, the multiplicity of edge tells the number of edges between two vertices.*

4 Cholesky factorization

4.1 Algorithm

We now formally introduce Cholesky factorization. Choleskey factorization is an algorithm that eventually produces a factorization $L = \mathcal{L}\mathcal{D}\mathcal{L}^\top$, where $\mathcal{L}$ is a lower-triangular matrix, and $\mathcal{D}$ is a diagonal matrix. Such a factorization allows us to solve a system $Lx = b$ by computing $x = L^{-1}b = (\mathcal{L}^{-1})^\top \mathcal{D}^{-1}\mathcal{L}^{-1}b$.

Given a Laplacian matrix L and denote $L(:, i)$ as the ith column of L. In the first iteration, we eliminate the first variable x_1, and produces a new linear system $S^{(1)}x' = b'$.

$$S^{(1)} \stackrel{\text{def}}{=} L - \frac{1}{L(1,1)} L(:,1)L(:,1)^\top$$

Note $S^{(1)}$ is called the Schur complement of L with respect to 1. Note the first row and first column of $S^{(1)}$ are all zeros, and $S^{(1)}$ and its submatrix $S^{(1)}_{2:n,2:n}$ are both Laplacians. We can write $L = S^{(1)} + \alpha_1 c_1 c_1^\top$.

At the second iteration, similarly, we can perform the same factorization to $S^{(1)}$, and we get $S^{(1)} = S^{(2)} + \alpha_2 c_2 c_2^\top$

At the kth iteration, we have $S^{(k)} = S^{(k-1)} + \alpha_k c_k c_k^\top$, where

$$\alpha_k = S^{(k-1)}(v_k, v_k)$$

,

$$\boldsymbol{c}_k = \frac{1}{\alpha_k} S^{(k-1)}(:, v_k)$$

, and

$$S^{(k)} = S^{(k-1)} - \alpha_k \boldsymbol{c}_k \boldsymbol{c}_k^T$$

At each step of the algorithm, we substract a rank one matrix from the current matrix, obtaining its Schur complement. The iteration stops when $\alpha_k = 0$.

In the end, we have $L = \sum_{k=1}^{n} \alpha_k c_k c_k^\top = \mathcal{C}\mathcal{D}\mathcal{C}^\top$. Define the permutation matrix P by $P\boldsymbol{e}_i = \boldsymbol{e}_{v_i}$. Letting $\mathcal{L} = P^\top \mathcal{C}$, we have $L = P\mathcal{L}\mathcal{D}\mathcal{L}^\top P^\top$. This decomposition is known as Cholesky factorization. Note, $\mathcal{L}$ is lower triangular and $\mathcal{D}$ is diagonal.

4.2 Graph interpretation

Cholesky factorization in fact works for all positive semi-definite matrices, however, when it is applied to Laplacians, we have a graph interpretation of schur complements at each iteration – the clique structure of the Schur complement.

Let's consider the first step of Cholesky factorization, we have $S^{(1)} = L - \frac{1}{L(1,1)} L(:,1)L(:,1)^\top$. The first row and first column of $S^{(1)}$ are all zeros, and $S^{(1)}$ and its submatrix $S^{(1)}_{2:n,2:n}$ are both Laplacians. For $i \neq j$ and $i, j \neq 1$,

$$S^{(1)}(i,j) = L(i,j) - \frac{1}{L(1,1)} L(1,i)L(1,j) = \underline{L(i,j) - (L)_1} + \underline{(L)_1 - 1/L(1,1)L(1,i)L(1,j)}$$

.

In the corresponding graph G, the first iteration can be regarded as removing vertex 1 and its incident edges (i.e. the first underline $L(i,j) - (L)_1$), and adding a clique to the neighbors of vertex 1 (i.e. the second underline $+(L)_1 - 1/L(1,1)L(1,i)L(1,j)$). The second iteration can be regraded as removing vertex 2 and its incident edges from $S^{(1)}$, and adding a clique to the neighbors of vertex 2. The rest of the iterations can be interpreted in the same way, as shown in algorithm 1.

The Laplacian of the clique added can be expressed as

$$(L)_v - \frac{1}{L(v,v)} L(:,v)L(:,v)^\top = \frac{1}{2} \sum_{u_1} \sum_{u_2} \frac{w(v,u_1)w(v,u_2)}{w(v)} L_{(u_1,u_2)} = \sum_{u_1 < u_2} \frac{w(v,u_1)w(v,u_2)}{w(v)} L_{(u_1,u_2)}$$

Algorithm 1 Cholesky Factorization

Inputs: a Laplacian L

Outputs: $S^{(0)}, S^{(1)}, S^{(2)}, ...$ that can be further used to find the lower-triangular matrix $\mathcal{L}$ and diagonal matrix D

```
1: $S^{(0)} = L$
2: **for** $k \leftarrow 1, 2, ..., n$ **do**
3:     $S^{(k)}$ = Eliminate vertex $k$ in $S^{(k-1)}$
4:     $S^{(k)}$ = Add a clique $Cl_k$ on neighbors of vertex $k$ in $S^{(k)}$.
5: **end for**
```

4.3 Time complexity

In the algorithm 1, at each iteration k, it first removs vertex k and $deg(k)$ edges, then add a clique of $deg(k)^2$ edges. So the overall time complexity is

$$O(\sum_v deg(v)^2) = O(n^3)$$

.

During this procedure, considering the updated graph G_k at each iteration as a multi-graph, we can see the number of edges in G_k is increasing at each iteration. This results in a time complexity of $O(n^3)$. Even if the original Laplacian L is sparse, Choleskey factorization can still produce a dense lower-triangular matrix $\mathcal{L}$.

5 Approximate Cholesky Factorization

Key idea: Randomize over the choice of the vertex to eliminate, and draw independent sample edges to approximate and replace the clique added in Choleskey Factorization.

Algorithm 2 shows the procedure. At each iteration, instead of adding a clique as in Algorithm 1, we sample a vertex uniformly at random from the remaining vertices, and eliminate it from the current matrix. Then we sample a clique on the neighbors of the sampled vertex.

Algorithm 2 Approximate Cholesky Factorization

Inputs: an $\rho > 0$ and a Laplacian L

Outputs: $S^{(0)}, S^{(1)}, S^{(2)}, ...$ that can be further used to find the lower-triangular matrix $\mathcal{L}$ and diagonal matrix D

```
  Replace each edge e by $\rho$ parallel edges each with weight $\frac{1}{\rho} w_e$        ▷ Preprocessing
  $\widehat{S}^{(0)} = L$ with edges replaced
  for $k \leftarrow 1, 2, ..., \rho n$ do
      Sample a vertex $\pi(k)$ uniformly random from remaining vertices
      $\widehat{S}^{(k)}$ = Eliminate vertex $k$ in $\widehat{S}^{(k-1)}$
      $\widehat{Cl}_v$ = CliqueSample $(S^{(k-1)}, v)$
      $\widehat{S}^{(k)}$ = Add $\widehat{Cl}_v$ to $\widehat{S}^{(k)}$
  end for
```

5.1 Time complexity

We will introduce the details of CliqueSample algorithm and other lemmas regarding Algorithm 2 later. Here will give a high level analysis of the time complexity of Algorithm 2.

Assume CliqueSample algorithm runs in $O(deg_{S^{(k-1)}}(v))$, and returns a clique with $\leq deg_{S^{(k-1)}}(v)$ edges. Then at each iteration, it removes $deg_{S^{(k-1)}}(v)$ edges and add $\leq deg_{S^{(k-1)}}(v)$ edges.

We can conclude that, the number edges of schur complement $S^{(k)}$ is smaller than or equal to the number edges of schur complement $S^{(k-1)}$. Thus the number of edges in $S^{(k)}$ is at most $|E|$, which means Approximate Cholesky Factorization(Algorithm 2) is much faster than Cholesky Factorization(Algorithm 1).

6 Clique Sampling

Algorithm 3 gives the pseudo-code for our CliqueSample algorithm.

From high level perspective, the algorithm CliqueSample produces a sparse Laplacian matrix which approximates the clique $Cl_v(S)$ in the ExactCholesky algorithm. The elimination step in SparseCholesky algorithm removes $deg_{S^{(k)}}(v)$ edges incident to vertex v and add a sparse Laplacian matrix given by CliqueSample (S,v). Since CliqueSample (S,v) only adds at most $deg_S(v)$ edges to the graph, the total number of edges does not increase with each elimination step.

7 Analysis of Clique Sampling

As mentioned before, algorithm CliqueSample samples a sparse Laplacian matrix that approximates the clique $C_v(S)$ in the ExactCholesky algorithm. In this section, our goal is to show

Algorithm 3 CliqueSample (S,v)

Inputs: a multi-graph S, a vertex v to be eliminated

Outputs: a sparse Laplacian matrix consisting of several i.i.d samples of edges.

```
1:  for $i \leftarrow 1$ to $deg_S(v)$ do
2:      Sample one edge $(v, u_1)$ with probability $\frac{1}{deg_S(v)}$.
3:      Sample another edge $(v, u_2)$ with probability $\frac{w(v,u_2)}{w(v)}$
4:      if $u_1 \neq u_2$ then
5:          $Y_i \leftarrow \frac{w(v,u_1)w(v,u_2)}{w(v,u_1)+w(v,u_2)} L_{(u_1,u_2)}$
6:      else
7:          $Y_i = 0$
8:      end if
9:  end for
10: return $\widehat{Cl}_v = \sum_i Y_i$
```

the samples in expectation behave like the clique $C_v(S)$ and the algorithm CliqueSample runs in linear time.

Lemma 7.1. *$\widehat{Cl}_v$ is a laplacian matrix.*

Proof. To prove $\widehat{Cl}_v$ is a laplacian matrix, since sum of laplacian matrices is a laplacian matrix, it suffices to show Y_i is either 0 or a laplacian matrix for each i. Since $L_{(u_1,u_2)}$ is a laplacian matrix of an edge of weight 1, Y_i is a laplacian matrix if $u_1 \neq u_2$. If $u_1 = u_2$, then $Y_i = 0$. $\square$

Lemma 7.2. $\mathbb{E}(\widehat{Cl_v}) = \mathbb{E}(\sum_i Y_i) = C_v(S)$. *That is, the expectation of the sampled matrix $\widehat{C}$ is the clique $C_v(S)$ in the* ExactCholesky *algorithm.*

Proof. The expectation of sampled matrix is equal to the sum of the expectation of each samples of edges generated after eliminating vertex v from the multi-graph S.

Note for each sampled edge (u_1, u_2), there are two possibilities that they are sampled. If (v, u_1) is sampled in line 2, and (v, u_2) is sampled in line 3, then the expectation of that sample is $\frac{w(v,u_2)}{w_v} \cdot \frac{w(v,u_1)w(v,u_2)}{w(v,u_1)+w(v,u_2)}$. Another case, if (v, u_2) is sampled in line 2, and (v, u_1) is sampled in line 3, then the expectation of that sample is $\frac{w(v,u_1)}{w_v} \cdot \frac{w(v,u_1)w(v,u_2)}{w(v,u_1)+w(v,u_2)}$. Therefore, for each sample edge (u_1, u_2), its expectation the sum of both, which is $\frac{w(v,u_1)w(v,u_2)}{w(v,u_1)+w(v,u_2)}$.

$$
\begin{aligned}
\mathbb{E}[\widehat{Cl}] = \mathbb{E}\left[\sum_i Y_i\right] &= \sum_i \mathbb{E}[Y_i] \\
&= \sum_{u_1}\sum_{u_2} \frac{w(v,u_2)}{w(v)} \frac{w(v,u_1)w(v,u_2)}{w(v,u_1)+w(v,u_2)} L_{(u_1,u_2)} \\
&= \sum_{u_1<u_2} \frac{w(v,u_1)w(v,u_2)}{w(v)} \frac{w(v,u_1)+w(v,u_2)}{w(v,u_1)+w(v,u_2)} L_{(u_1,u_2)} \\
&= \sum_{u_1<u_2} \frac{w(v,u_1)w(v,u_2)}{w(v)} L_{(u_1,u_2)}
\end{aligned}
$$

$$= Cl_v(S)$$

□

Definition 7.3. *In rest of the notes, we will need frequently refer to the matrices normalized by L. We will use the following notation: Given a symmetric matrix S, s.t.* $\ker(L) \subseteq \ker(S)$, $\overline{S} \overset{\text{def}}{=} (L^+)^{1/2} S (L^+)^{1/2}$.

Definition 7.4. *We say a multi-edge e is* $1/\rho$*-bounded if* $||w(e)\overline{b_e b_e^\top}|| \leq 1/\rho$. *Given a Laplacian S that corresponds to a multi-graph* G_S*, we say a multi-graph S is* $1/\rho$*-bounded if all its multi-edges of S are* $1/\rho$*-bounded.*

Lemma 7.5. $||\overline{Y_i}|| \leq 1/\rho$*, i.e. the norm of the sample edge in the* CliqueSample *algorithm(line 5) is* $1/\rho$*-bounded w.r.t. L. It holds for every iteration in the* SparseCholesky *algorithm.*

To prove the above lemma, we need to first introduce the following lemma – *effective resistance* in Laplacians is a distance[**Klein93**].

Lemma 7.6. *Given a connected weighted multi-graph G with associated Laplacian matrix L, consider three distinct vertices* $u, v, z \in V$.

$$||\overline{L_{(u,v)}}|| \leq ||\overline{L_{(u,z)}}|| + ||\overline{L_{(z,v)}}||$$

Proof.

$$\begin{aligned}
||\overline{L_{(u,v)}}|| &= ||\overline{b_{(u,v)} b_{(u,v)}^\top}|| = ||L^{-1/2} b_{u,v} b_{u,v}^\top L^{-1/2}|| \\
&= ||L^{-1/2}(b_{u,z} + b_{z,v})(b_{u,z} + b_{z,v})^\top L^{-1/2}|| \\
&= ||L^{-1/2}(b_{u,z}b_{u,z}^\top + b_{u,z}b_{z,v}^\top + b_{z,v}b_{u,z}^\top + b_{z,v}b_{z,v}^\top)L^{-1/2}|| \\
&= ||L^{-1/2}(b_{u,z}b_{u,z}^\top)L^{-1/2} + L^{-1/2}(b_{z,v}b_{z,v}^\top)L^{-1/2} + L^{-1/2}(b_{u,z}b_{z,v}^\top)L^{-1/2} + L^{-1/2}(b_{z,v}b_{u,z}^\top)L^{-1/2}|| \\
&\leq ||\overline{L_{(u,z)}}|| + ||\overline{L_{(z,v)}}|| + ||L^{-1/2}(b_{u,z}b_{z,v}^\top + b_{z,v}b_{u,z}^\top)L^{-1/2}|| \\
&\leq ||\overline{L_{(u,z)}}|| + ||\overline{L_{(z,v)}}||
\end{aligned}$$

□

Proof of Lemma 7.5. We use induction to prove lemma 7.5 is true for every iteration in the SparseCholesky algorithm.

Base step:

For $\widehat{S}^{(0)} = L$ with edges replaced, clearly every edge in $\widehat{S}^{(0)}$ is $1/\rho$-bounded. For any splitted edge e' with weight $\frac{1}{\rho}w_e$, we have $||w(e')\overline{b_e b_e^\top}|| = \frac{1}{\rho}||w(e)\overline{b_e b_e^\top}|| = \leq 1/\rho$

Inductive step: Suppose all edges in $\widehat{S}^{(k)}$ are $1/\rho$-bounded. We need to show that all edges in $\widehat{S}^{(k+1)}$ are $1/\rho$-bounded.

If $u_1 = u_2$ and $Y_i = 0$, clearly the lemma holds.

If $u_1 \neq u_2$, then $Y_i = \frac{w(v,u_1)w(v,u_2)}{w(v,u_1)+w(v,u_2)} L_{(u_1,u_2)}$. By Lemma 7.6, we have $||\overline{L_{(u_1,u_2)}}|| \leq ||\overline{L_{(v,u_1)}} + \overline{L_{(v,u_2)}}||$.

$$\frac{w(v,u_1)w(v,u_2)}{w(v,u_1)+w(v,u_2)}||\overline{L_{u_1,u_2}}|| \leq \frac{w(v,u_1)w(v,u_2)}{w(v,u_1)+w(v,u_2)}(||\overline{L_{v,u_1}}|| + ||\overline{L_{v,u_2}}||)$$

$$
\begin{aligned}
&= \frac{w(v,u_2)}{w(v,u_1)+w(v,u_2)}(w(v,u_1)||\overline{L_{v,u_1}}||) \\
&+ \frac{w(v,u_1)}{w(v,u_1)+w(v,u_2})(w(v,u_2)||\overline{L_{v,u_2}}||) \\
&\leq \frac{w(v,u_2)}{w(v,u_1)+w(v,u_2)}\frac{1}{\rho} + \frac{w(v,u_1)}{w(v,u_1)+w(v,u_2)}\frac{1}{\rho} \\
&= \frac{1}{\rho},
\end{aligned}
$$

All edges added in CliqueSample algorithm are $1/\rho$-bounded; thus, all edges in $\widehat{S}^{(k+1)}$ are $1/\rho$-bounded.

According to **Base step** and **Inductive step**, we have proved that every edge in $\widehat{S}^{(i)}$ is $1/\rho$-bounded, and it holds for every iteration in the SparseCholesky algorithm.

□

8 Analysis of Approximate Cholesky Factorization

In this section, we will show the approximation guarantee of the SparseCholesky algorithm.

Theorem 8.1. *Given a connected undirected multi-graph $G = (V, E)$, with positive edges weights $w : E \to \mathbb{R}_+$, and associated Laplacian L, and scalars $\delta > 1$, $0 < \epsilon \leq 1/2$, the algorithm* SparseCholesky (L, ϵ, δ) *returns an approximate sparse Cholesky decomposition $(P, \mathcal{L}, \mathcal{D})$ s.t. with probability at least $1 - 2/n^{\delta}$,*

$$(1-\epsilon)L \preceq P\mathcal{L}\mathcal{D}\mathcal{L}^{\top}P^{\top} \preceq (1+\epsilon)L. \qquad (1)$$

The expected number of non-zero entries in $\mathcal{L}$ is $O(\frac{\delta^2}{\epsilon^2}m\log^3 n)$. The algorithm runs in expected time $O(\frac{\delta^2}{\epsilon^2}m\log^3 n)$.

In order to prove this theorem, we will use the following results.

Definition 8.2. *A matrix martingale $\{Y_k \in \mathcal{R}^{n\times n}\}_{k\geq 0}$ is a sequence of matrix such that $Y_0 = 0$. $(E|_{k-1}[Y_k] = Y_{k-1})$.*

Lemma 8.3. *$\{\widehat{L}^{(k)}\}$ is a matrix martingale, and $L^{(n)} - L_0 = L^{(n)} - L = \sum_{i=1}^{n-1}\sum_e X_e^{(i)}$.*

Proof.

$$
\begin{aligned}
\widehat{L}^{(k)} - \widehat{L}^{(k-1)} &= \widehat{S}^{(k)} - \widehat{S}^{(k-1)} + c_k c_k^T \\
&= -(\widehat{S}^{(k-1)})_{\pi(k)} + \widehat{Cl_k} + c_k c_k^T \\
&= -c_k x_k^T - Cl_k + \widehat{Cl_k} + c_k c_k^T \\
&= -Cl_k + \widehat{Cl}_k,
\end{aligned}
$$

Then, according to lemma 7.2, we have $\{\widehat{L}^{(k)}\}$ is a matrix martingale. Since the expected value of $L^{(k)}$ is $L^{(k-1)}$, defining $X_e^{(k)} = Y_e^{(k)} - Y_e^{(k-1)}$, where e denotes the eth sample of $X_e^{(k)}$, we can rewrite $L^{(n)} - L$ as

$$L^{(n)} - L_0 = L^{(n)} - L = \sum_{i=1}^{n-1}\sum_e X_e^{(i)}.$$

□

In practice, we normalize the martingale by L, thus we consider $\overline{L^{(n)}} - \overline{L_0} = \sum_{i=1}^{n-1}\sum_e \overline{X_e^{(i)}}$.

Lemma 8.4. *Suppose that we have $Z = \sum_{i=1}^{n-1}\sum_e X_e^{(i)}$, a martingale of $d \times d$ matrices satisfying*

- *The l2 norm of every sample $Z_e^{(i)}$ is bounded by σ_1.*
- *For every i, we have $\left\|\sum_e E[(Z_e^{(i)})^2]\right\| \le \sigma_2^2$*
- *$\forall\ \epsilon > 0$, we have $Pr[Z \npreceq \epsilon I] \le exp(-\epsilon^2/4\sigma^2)$, where $\sigma^2 = \max\{\sigma_3^2, \frac{\epsilon}{2}\sigma_1, \frac{4\epsilon}{5}\sigma_2\}$.*

The proof of this lemma is introduced in [**Kyng16**].

In order to use the above lemma, we require that $\left\|\overline{L^{(i)}}\right\|$ is bounded, or that the probability of $\left\|\overline{L^{(i)}}\right\|$ grows large is small. We introduce the ϵ-truncated martingale, that is, considering the event $A_h = \left[\forall 1 \le j < h. \sum_{i=1}^{j}\sum_e Z_e^{(i)} \preceq \epsilon I\right]$, where $h \in \{1, \ldots, k+1\}$, we define

$$\tilde{Z} = \sum_{i=1}^{k}(\mathbb{1}_{A_h}\sum_e Z_e^{(i)}).$$

It can be shown that the associated ϵ-truncated martingale $\tilde{Z}$ is also a bags-of-dice martingale, and

$$Pr[\epsilon I \npreceq Z \text{ or } Z \npreceq \epsilon I] \le Pr[\epsilon I \npreceq \tilde{Z} \text{ or } \tilde{Z} \npreceq \epsilon I].$$

Thus, we need to prove the concentration of the truncated martingale.

Lemma 8.5. *Given an integer $1 \le k \le n-1$, conditional on the choices of the SparseCholesky algorithm until step $k-1$, the following statements hold.*

- *conditional on $\pi(k)$, $E_{Y_e}[\mathbb{1}_{A_h}\overline{X_e}] = 0$*
- *$\left\|\mathbb{1}_{A_h}\overline{X_e}\right\| \le 1/\rho$ holds always.*
- *Conditional on $\pi(k)$, $\sum_e E_{Y_e}[\mathbb{1}_{A_h}\overline{X_e}] \preceq \mathbb{1}_{A_h}\frac{1}{\rho}\overline{C_v(S)}$.*
- *$\left\|\mathbb{1}_{A_h}\overline{C_{\pi(k)}(\widehat{S}^{(k-1)})}\right\| \le 1+\epsilon$.*
- *$E[\mathbb{1}_{A_h}\overline{C_{\pi(k)}(\widehat{S}^{(k-1)})}] \preceq \frac{2(1+\epsilon)}{n+1-k}I$.*

We prove the theorem 8.1 as follows:

Proof.

$$\begin{aligned}
&Pr[(1-\epsilon)L \preceq L^{(n)} \preceq (1+\epsilon)L]\\
=&1 - Pr[(1-\epsilon)L \npreceq L^{(n)} \text{ or } L^{(n)} \npreceq (1+\epsilon)L]\\
=&1 - Pr[-Z \npreceq \epsilon I \text{ or } Z \npreceq \epsilon I]
\end{aligned}$$

$$\begin{aligned}&\geq 1 - Pr[-\tilde{Z} \npreceq \epsilon I \text{ or } \tilde{Z} \npreceq \epsilon I]\\ &\geq 1 - Pr[-\tilde{Z} \npreceq \epsilon I] - Pr[\tilde{Z} \npreceq \epsilon I]\end{aligned}$$

To bound this probability, we need to use lemma 8.4. Noted that we have $E_{Y_e}[\mathbb{1}_{A_k}\overline{X_e^{(k)}}] = 0$ and $\left\|\mathbb{1}_{A_k}\overline{X_e^{(k)}}\right\| \leq 1/\rho$. If we set $\sigma_1 = 1/\rho$ and $\sigma_2 = \sqrt{3/(2\rho)}$ in lemma 8.4, we obtain

$$\begin{aligned}&E[\sum_e E(\left\|\mathbb{1}_{A_k}\overline{X_e^{(k)}}\right\|)^2]\\ &\preceq E[\frac{1}{\rho}\left\|\mathbb{1}_{A_h}\frac{1}{\rho}\overline{C_{\pi(k)}(S)}\right\|]\\ &\preceq \frac{2(1+\epsilon)}{\rho(n+1-k)}I\\ &\preceq \frac{3}{\rho(n+1-k)}I.\end{aligned}$$

By letting $\sigma_3^2 = \frac{3\ln n}{\rho} \geq \sum_{k=1}^{n-1}\frac{3}{\rho(n+1-k)}$ and using lemma 8.4, we have $\sigma^2 = \max\{\sigma_3^2, \frac{\epsilon}{2}\sigma_1, \frac{4\epsilon}{5}\sigma_2\}$. Then $\sigma^2 <= \frac{\epsilon^2}{4(1+\delta)\ln n}$ can be achieved by choosing some ρ.

Finally,

$$\begin{aligned}Pr[-\tilde{Z} \npreceq \epsilon I] + Pr[\tilde{Z} \npreceq \epsilon I] &\leq 2n\exp\left(-\epsilon^2/4\sigma^2\right)\\ &= 2n\exp\left(-(1+\delta)\ln n\right)\\ &= 2n^{-\delta}.\end{aligned}$$

This completes the proof. ☐

9 Conclusion

In this report, we presented the paper [**Kyng16**] which designs the first nearly linear time solver for Laplacian system, and does not use any graph theoretic constructions. We described the algorithm and showed the guarantees for the approximate Cholesky Factorization by introducing the concentration bounds for a class of matrix martingales, as developed in the paper.

References

[1] Klein, Douglas J., and Milan Randić. "Resistance distance." Journal of mathematical chemistry 12.1 (1993): 81-95.

[2] Kyng, Rasmus, and Sushant Sachdeva. "Approximate gaussian elimination for laplacians-fast, sparse, and simple." 2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS). IEEE, 2016.