Approximate Gaussian Elimination for Laplacian Systems

MA4291 Presentation Report 2

Ang Yan Sheng
A0144836Y

Motivation A *Laplacian matrix* is a symmetric matrix $\mathbf{L}$ such that all entries off the diagonal are nonpositive, and the entries in each row sum to 0. Note that every Laplacian matrix $\mathbf{L}$ corresponds to a weighted graph, since for some suitable weights $w_{ij} \geqslant 0$, we can write

$$\mathbf{L} = \sum_{1 \leqslant i < j \leqslant n} w_{ij} (\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^T.$$

The main motivation for studying Laplacians comes from the following natural quadratic form on weighted graphs:

$$\mathbf{x}^T \mathbf{L} \mathbf{x} = \sum_{1 \leqslant i < j \leqslant n} w_{ij} (x_i - x_j)^2.$$

This form arises in various contexts, with interpretations such as energy (system of springs, graph drawing), power dissipated (resistor networks), or an error function (function learning on graphs), usually as a quantity to be minimised. Moreover, the Poisson equation $\Delta u = f$, ubiquitous in digital geometry processing, can also be discretised on a mesh and reduced to minimising this form.

By calculus, this problem can be reduced to solving a linear system of the form $\mathbf{L}\mathbf{x} = \mathbf{b}$. We can solve this in $O(n^3)$ time by Gaussian elimination (ie. LU factorisation, or Cholesky factorisation for symmetric matrices), but in applications for sparse graphs with millions or even billions of vertices, this is not fast enough. Moreover, the worst case running time does not improve even when $\mathbf{L}$ is sparse, since the Cholesky factorisation of $\mathbf{L}$ might not be sparse. Hence we arrive at the following problem statement:

Problem. *Given a Laplacian matrix* $\mathbf{L} \in \mathbb{R}^{n \times n}$ *with* m *nonzero entries, and* $\mathbf{b} \in \mathbb{R}^n$, *efficiently solve* $\mathbf{L}\mathbf{x} = \mathbf{b}$ *to accuracy* ε.

The first major theoretical breakthrough for this problem was Spielman and Teng's (2004) nearly linear time algorithm, which runs in $O(m \ln^{50} n \ln(1/\varepsilon))$ time.

Since then the field has progressed steadily, and the current state of the art algorithm (Cohen et al. 2014) runs in $O(m \ln^{1/2} n \ln(1/\varepsilon))$ time—in other words, solving a sparse Laplacian system is even faster than sorting the nonzero entries of the matrix!

Most of these near-linear time algorithms rely on difficult graph-theoretic constructions such as low-stretch trees and expander graphs. We will present a new approach, based on Kyng-Sachdeva (2016) and Kyng's thesis (2017), which only uses random sampling.

Approximate Cholesky factorisation A popular method to solve the system $\mathbf{Lx} = \mathbf{b}$ numerically is by using an iterative scheme such as $\mathbf{x}_{k+1} = \mathbf{M}\mathbf{x}_k + \mathbf{N}^{-1}\mathbf{b}$. It is easy to show that this scheme converges to a solution of the original equation for all initial $\mathbf{x}_0$ if and only if $\mathbf{M} = \mathbf{I} - \mathbf{N}^{-1}\mathbf{L}$ and $\|\mathbf{M}\| < 1$.

Hence if we can find a *sparse approximate Cholesky factorisation* to $\mathbf{L}$, namely some $\mathbf{N} = \mathbf{U}^\mathsf{T}\mathbf{U}$ with $\mathbf{U}$ sparse upper-triangular, and $\mathbf{N} \approx_{1/2} \mathbf{L}$ (meaning $\|\mathbf{N}^{-1}\mathbf{L} - \mathbf{I}\| < 1/2$), then we get an ε-approximation to the solution after $O(\ln(1/\varepsilon))$ iterations.

Since we are working with Laplacian matrices, we should interpret Cholesky factorisation in graph theoretic terms. As such, we recast each step of the Cholesky factorisation algorithm as subtracting a rank 1 matrix which agrees with the first row and column of the given matrix. For example, we may write

$$
\begin{aligned}
\mathbf{L} = \begin{pmatrix} 16 & -4 & -8 & -4 \\ -4 & 5 & 0 & -1 \\ -8 & 0 & 9 & -1 \\ -4 & -1 & -1 & 6 \end{pmatrix}
&= \begin{pmatrix} 16 & -4 & -8 & -4 \\ -4 & 1 & 2 & 1 \\ -8 & 2 & 4 & 2 \\ -4 & 1 & 2 & 1 \end{pmatrix} + \begin{pmatrix} & & & \\ & 4 & -2 & -2 \\ & -2 & 5 & -3 \\ & -2 & -3 & 5 \end{pmatrix} \\
&= \begin{pmatrix} 16 & -4 & -8 & -4 \\ -4 & 1 & 2 & 1 \\ -8 & 2 & 4 & 2 \\ -4 & 1 & 2 & 1 \end{pmatrix} + \begin{pmatrix} & & & \\ & 4 & -2 & -2 \\ & -2 & 1 & 1 \\ & -2 & 1 & 1 \end{pmatrix} + \begin{pmatrix} & & & \\ & & & \\ & & 4 & -4 \\ & & -4 & 4 \end{pmatrix} \\
&= \mathbf{c}_1\mathbf{c}_1^\mathsf{T} + \mathbf{c}_2\mathbf{c}_2^\mathsf{T} + \mathbf{c}_3\mathbf{c}_3^\mathsf{T},
\end{aligned}
$$

where

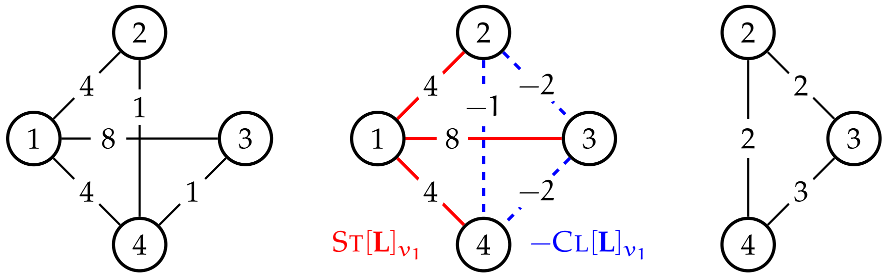
$$
\mathbf{c}_1 = (4, -1, -2, -1)^\mathsf{T}, \quad \mathbf{c}_2 = (0, 2, -1, -1)^\mathsf{T} \quad \mathbf{c}_3 = (0, 0, 2, -2)^\mathsf{T}.
$$

We can then read off the Cholesky factorisation as $\mathbf{L} = \mathbf{U}^\mathsf{T}\mathbf{U}$, with

$$
\mathbf{U} = \begin{pmatrix} \mathbf{c}_1 & \mathbf{c}_2 & \mathbf{c}_3 & \mathbf{0} \end{pmatrix}^\mathsf{T} = \begin{pmatrix} 4 & -1 & -2 & -1 \\ & 2 & -1 & -1 \\ & & 2 & -1 \\ & & & 0 \end{pmatrix}.
$$

Note that rank 1 matrices are also Laplacians, so we can look at the graphs corresponding to the matrices in each step (see figure). Hence subtracting a rank 1 matrix from a Laplacian removes the set of all edges (or *star*) $\textsc{St}[\mathbf{L}]_v$ from a vertex, while adding a *clique* $\textsc{Cl}[\mathbf{L}]_v$ to its neighbours.

$$\begin{pmatrix} 16 & -4 & -8 & -4 \\ -4 & 5 & 0 & -1 \\ -8 & 0 & 9 & -1 \\ -4 & -1 & -1 & 6 \end{pmatrix} = \begin{pmatrix} 16 & -4 & -8 & -4 \\ -4 & 1 & 2 & 1 \\ -8 & 2 & 4 & 2 \\ -4 & 1 & 2 & 1 \end{pmatrix} + \begin{pmatrix} & & & \\ & 4 & -2 & -2 \\ & -2 & 5 & -3 \\ & -2 & -3 & 5 \end{pmatrix}$$


$\mathrm{ST}[\mathbf{L}]_{v_1}$
$-\mathrm{CL}[\mathbf{L}]_{v_1}$

Thus at each step of Cholesky factorisation, we remove $\deg(v)$ edges but add in $\binom{\deg(v)}{2}$ new ones; this results in a *fill-in* phenomenon, which is why sparse Laplacians might not have sparse Cholesky factorisations.

The main idea of Kyng-Sachdeva (2016) and Kyng (2017) is to add only $\deg(v)$ edges randomly sampled from $\mathrm{CL}[\mathbf{L}]_v$ at each step (line 6 of Algorithm 1), which ensures that the graph stays sparse.

Algorithm 1 $\mathrm{CHOLAPX}'(\mathbf{L}, \delta)$

```
1: Split each edge into $\rho = 12\ln^2(1/\delta)$ copies
2: Pick random permutation of vertices $v_1, \dots, v_n$
3: $\mathbf{S}_0 \leftarrow \mathbf{L}$
4: for $i = 1, \dots, n$ do
5:     $\mathbf{c}_i \leftarrow \begin{cases} \frac{1}{\sqrt{\mathbf{S}_{i-1}(v_i,v_i)}}\mathbf{S}_{i-1}(v_i,:) & \mathbf{S}_{i-1}(v_i,v_i) \neq 0 \\ \mathbf{0} & \text{else} \end{cases}$
6:     $\mathbf{C}_i \leftarrow \mathrm{CLIQUESAMPLE}(\mathbf{S}_{i-1}, v_i)$
7:     $\mathbf{S}_i \leftarrow \mathbf{S}_{i-1} - \mathrm{ST}[\mathbf{S}_{i-1}]_{v_i} + \mathbf{C}_i$
8: end for
9: $\mathbf{U} \leftarrow \begin{pmatrix} \mathbf{c}_1 & \cdots & \mathbf{c}_n \end{pmatrix}^T$
10: return $\mathbf{U}$
```

Theorem (Kyng 2017). *Let* $\delta < n^{-100}$. *Then with probability* $1 - O(\delta)$, $\mathrm{CHOLAPX}'(\mathbf{L}, \delta)$ *returns* $\mathbf{U}$ *with* $O(m \ln^2(1/\delta) \ln n)$ *nonzero entries such that* $\mathbf{U}^T\mathbf{U} \approx_{1/2} \mathbf{L}$. *Moreover, for* $t > 1$, $\mathrm{CHOLAPX}'(\mathbf{L}, \delta)$ *runs in* $O(tm \ln^2(1/\delta) \ln n)$ *time with probability* $1 - n^{-t}$.

To prove the above result, the key observation is that the partial Cholesky factorisations produced by this algorithm, namely

$$\mathbf{L}_i = \mathbf{S}_i + \sum_{j=1}^{i} \mathbf{c}_j \mathbf{c}_j^T,$$

satisfies $\mathbf{L}_0 = \mathbf{L}$, $\mathbf{L}_n = \mathbf{U}^T\mathbf{U}$, and $\mathbf{L}_i - \mathbf{L}_{i-1} = \mathbf{C}_i - \mathrm{CL}[\mathbf{S}_{i-1}]_{v_i}$, which has expected value zero (conditional on $\mathbf{L}_0, \dots, \mathbf{L}_{i-1}$). Hence $(\mathbf{L}_i)_{i=0}^n$ is a matrix martingale.

Algorithm 2 $\mathrm{CLIQUESAMPLE}(\mathbf{S}, v)$

```
1: for $e = 1, \dots, \deg_{\mathbf{S}}(v)$ do
2:     Sample $(v, u_1)$ with probability $\frac{w(v,u_1)}{w_{\mathbf{S}}(v)}$
3:     Sample $(v, u_2)$ uniformly
4:     $\mathbf{Y}_e \leftarrow \frac{w(v,u_1)w(v,u_2)}{w(v,u_1)+w(v,u_2)}(\mathbf{e}_{u_1} - \mathbf{e}_{u_2})(\mathbf{e}_{u_1} - \mathbf{e}_{u_2})^T$
5: end for
6: return $\sum_e \mathbf{Y}_e$          ▷ expected value = $\mathrm{CL}[\mathbf{S}]_v$
```

We want to show $\mathbb{P}\left(\|\mathbf{L}^{-1}\mathbf{L}_n - \mathbf{I}\| \geqslant \frac{1}{2}\right) = O(\delta)$; by the above discussion, this is a matrix martingale concentration bound. The main tool that Kyng uses here is a matrix analogue of the Freedman inequality (1975):

Theorem (Tropp 2011). *Let* $(\mathbf{A}_k)_{k\geqslant 0}$ *be a symmetric* $d \times d$*-matrix martingale with* $\lambda_{max}(\mathbf{B}_k) \leqslant R$*, where* $\mathbf{B}_k = \mathbf{A}_k - \mathbf{A}_{k-1}$*. Let* $\mathbf{W}_k = \sum_{j=1}^{k} \mathbb{E}_{<j}(\mathbf{B}_j^2)$*. Then for all* $t \geqslant 0$ *and* $\sigma^2 > 0$*,*

$$\mathbb{P}\left(\exists k : \lambda_{max}(\mathbf{A}_k) \geqslant t \text{ and } \|\mathbf{W}_k\| \leqslant \sigma^2\right) \leqslant 2d \exp\left(\frac{-t^2/2}{\sigma^2 + Rt/3}\right).$$

After normalising with respect to $\mathbf{L}$ and adding a stopping condition to the martingale, Kyng splits the failure probability into two terms, according to the variation measure $\mathbf{W}_k$. The matrix Freedman inequality is used directly to bound the probability when $\mathbf{W}_k$ is small. Surprisingly, by another application of the matrix Freedman inequality on $\mathbf{W}_k$ itself (!), we can also bound the probability that $\mathbf{W}_k$ grows large, which completes the proof.

Extensions Note that the running time of CHOLAPX$'$ is probabilistic. Intuitively, the algorithm takes longer to run if it happens to pick many vertices with high degree. This suggests a tweak where we only pick vertices with degree at most twice the average degree, and Kyng also showed that this modified algorithm CHOLAPX has running time uniformly bounded by $O(m \ln^2(1/\delta) \ln n)$.

Inspired by this algorithm, Spielman has proposed a variation where instead of removing, sampling and replacing vertices, we perform analogous operations on edges; this corresponds to Gaussian elimination on individual entries instead of on rows and columns.[*] In practice, this heuristic performs remarkably consistently across different families of graphs. However, the matrix martingale techniques used by Kyng break down in this context, and a theoretical explanation of why this algorithm works is still an open problem.

References

R. Kyng. *Approximate Gaussian Elimination*. Yale University, PhD dissertation, 2017.
D. Spielman. "The Laplacian Matrices of Graphs." IEEE International Symposium on Information Theory. Barcelona, 16 Jul 2016. Plenary Lecture.
J. Tropp. "Freedman's Inequality for Matrix Martingales." *Elect. Comm. in Probab.*, **16**, 2011, 262–270.

[*]This is implemented as the `approxCholLap` method in the Laplacians.jl package, a large collection of Laplacian-related graph algorithms. For solving Laplacian systems, this heuristic is recommended over other algorithms, including an implementation of CHOLAPX.