On Approximate Elimination

Kyng, Spielman

March 9, 2020

Contents

Overview

I'm going to write up notes on the version of Approximate Cholesky Elimination that is implemented in Laplacians.jl. It is in turn based on the observation that elimination can proceed edgewise as in the nearly linear-time directed Laplacian solver of mine, which Dan and I wanted to find a way to use on undirected Laplacians. And of course, it's very related to the original Approximate Gaussian Elimination paper of Sushant and I. Note that the code in Laplacians.jl is based on Section 2.1, but it might be better to used the alternative form given in Section 2.3.

1 Preliminaries

1.1 Linear Algebra

All of this dissertation concerns questions in theoretical computer science that draw heavily on linear algebra. Below we introduce some basic notation and definitions that will be used in later chapters.

Upper and Lower Triangular Matrices. We say a square matrix $\boldsymbol{\mathcal{U}}$ is upper triangular if it has non-zero entries $\boldsymbol{\mathcal{U}}(i,j) \neq 0$ only for $i \leq j$ (i.e. above the diagonal). Similarly, we say a square matrix $\boldsymbol{\mathcal{L}}$ is lower triangular if it has non-zero entries $\boldsymbol{\mathcal{U}}(i,j) \neq 0$ only for $i \geq j$ (i.e. below the diagonal). Often, we will work with matrices that are not upper or lower triangular, but which for we know a permutation matrix $\boldsymbol{P}$ s.t. $\boldsymbol{P\mathcal{U}P}^\top$ is upper (respectively lower) triangular. For computational purposes, this is essentially equivalent to having a upper or lower triangular matrix, and *we will refer to such matrices as upper (or lower) triangular.* The algorithms we develop for factorization will always compute the necessary permutation.

Moore-Penrose Pseudo-inverse. We use $\boldsymbol{B}^\dagger$ to denote the Moore-Penrose pseudo-inverse of a matrix $\boldsymbol{B}$.

Positive Definite Matrices and Positive Semi-Definite Matrices. We say a square matrix $\boldsymbol{M} \in \mathbb{R}^{n \times n}$ is positive definite if for all $\boldsymbol{x} \in \mathbb{R}^n$ where $\boldsymbol{x} \neq \boldsymbol{0}$, we have $\boldsymbol{x}^\top \boldsymbol{M}\boldsymbol{x} > 0$. If for all $\boldsymbol{x} \in \mathbb{R}^n$ where $\boldsymbol{x} \neq \boldsymbol{0}$, we have $\boldsymbol{x}^\top \boldsymbol{M}\boldsymbol{x} \geq 0$, then $\boldsymbol{M}$ is positive semi-definite (PSD).

Loewner Order. We use $\preceq$ to denote a partial order on symmetric matrices, where $\boldsymbol{A} \preceq \boldsymbol{B}$ if and only $\boldsymbol{B} - \boldsymbol{A}$ is PSD.

Restriction. Given an $m \times n$ matrix $\boldsymbol{B}$, and index sets $F \subseteq [m]$, $C \subseteq [n]$, we use $(\boldsymbol{B})_{FC}$ to denote the $|F| \times |C|$ matrix obtained by restricting $\boldsymbol{B}$ to the rows F and the columns C. When it does not cause ambiguity, we will sometimes omit the brackets and write $\boldsymbol{B}_{FC}$ to denote the matrix $(\boldsymbol{B})_{FC}$.

Projection Matrix Given matrix $\boldsymbol{B} \in \mathbb{R}^{m \times n}$, let $\boldsymbol{\Pi}_{\boldsymbol{B}} \stackrel{\text{def}}{=} \boldsymbol{B}(\boldsymbol{B}\boldsymbol{B}^\top)^\dagger \boldsymbol{B}^\top$, i.e. the orthogonal projection onto the image of $\boldsymbol{B}$.Note that $\boldsymbol{\Pi}_{\boldsymbol{B}} = \boldsymbol{\Pi}_{\boldsymbol{B}}^\top$ and $\boldsymbol{\Pi}_{\boldsymbol{B}} = \boldsymbol{\Pi}_{\boldsymbol{B}}^2$.

Norms from Quadratic Forms. For any PSD matrix $\boldsymbol{M}$ and any vector $\boldsymbol{v}$, define the semi-norm $\|\boldsymbol{v}\|_{\boldsymbol{M}} \stackrel{\text{def}}{=} \sqrt{\boldsymbol{v}^\top \boldsymbol{M}\boldsymbol{v}}$.

Kernel and Cokernel. Given a matrix $\boldsymbol{M}$, we use $\ker(\boldsymbol{M})$ to denote the kernel of $\boldsymbol{M}$, i.e. the subspace of vectors $\boldsymbol{x}$ s.t. $\boldsymbol{M}\boldsymbol{x} = \boldsymbol{0}$. The term cokernel refers to the kernel of $\boldsymbol{M}^\top$.

Let $\boldsymbol{1} \in \mathbb{R}^n$ denote the all ones vector, with dimension n that will always be made clear in the context of its use. Similarly, we let $\boldsymbol{0}$ denote the all zero vector or matrix, depending on context.

The following fact is useful, since we often need to apply the pseodo-inverse of a matrix.

Fact 1.1 (Pseudo-inverse of a product)**.** *Suppose $\boldsymbol{M} = \boldsymbol{A}\boldsymbol{B}\boldsymbol{C}$ is square real matrix, where $\boldsymbol{A}$ and $\boldsymbol{C}$ are non-singular. Then*

$$\boldsymbol{M}^\dagger = \boldsymbol{\Pi}_{\boldsymbol{M}} \boldsymbol{C}^{-1} \boldsymbol{B}^\dagger \boldsymbol{A}^{-1} \boldsymbol{\Pi}_{\boldsymbol{M}^\top}.$$

Definition 1.2 (PSD Spectral Approximation)**.** Given two PSD matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ and a scalar $\epsilon \geq 0$, we say $\boldsymbol{A} \approx_\epsilon \boldsymbol{B}$ if and only if

$$\exp(-\epsilon)\boldsymbol{A} \preceq \boldsymbol{B} \preceq \exp(\epsilon)\boldsymbol{A}.$$

We say $\boldsymbol{A}$ is an ϵ-approximation of $\boldsymbol{B}$.

1.2 Matrix Classes

In this section, we introduce several families of matrices. Each is significant, because we are able to construct fast linear system solvers for matrices in these classes.

Symmetric Diagonally Dominant (SDD) Matrices. A matrix $\boldsymbol{M} \in \mathbb{R}^{n \times n}$ is said to be Symmetric Diagonally Dominant (SDD), if it is symmetric and for each row i,

$$\boldsymbol{M}(i,i) \geq \sum_{j \neq i} |\boldsymbol{M}(i,j)| \,. \tag{1}$$

It can be shown that every SDD matrix is positive semi-definite.

Undirected Laplacians, a.k.a. Laplacians. We consider a connected undirected multi-graph $G = (V, E)$, with positive edges weights $w : E \to \mathbb{R}_+$. Let $n = |V|$ and $m = |E|$. We label vertices 1 through n, s.t. $V = \{1, \ldots, n\}$. Let $\boldsymbol{\chi}_i$ denote the i^{th} standard basis vector. Given an ordered pair of vertices (u, v), we define the pair-vector $b_{u,v} \in \mathbb{R}^n$ as $\boldsymbol{b}_{u,v} = \boldsymbol{\chi}_v - \boldsymbol{\chi}_u$. For a multi-edge e, with endpoints u, v (arbitrarily ordered), we define $\boldsymbol{b}_e = \boldsymbol{b}_{u,v}$.

By assigning an arbitrary direction to each multi-edge of G we define the Laplacian of G as $\boldsymbol{L} = \sum_{e \in E} w(e)\boldsymbol{b}_e\boldsymbol{b}_e^\top$. Note that the Laplacian does not depend on the choice of direction for each edge. Given a single multi-edge e, we refer to $w(e)\boldsymbol{b}_e\boldsymbol{b}_e^\top$ as the Laplacian of e.

A weighted multi-graph G is not uniquely defined by its Laplacian, since the Laplacian only depends on the sum of the weights of the multi-edges on each edge. We want to establish a one-to-one correspondence between a weighted multi-graph G and its Laplacian $\boldsymbol{L}$, so from now on, we will consider every Laplacian to be maintained explicitly as a sum of Laplacians of multi-edges, and we will maintain this multi-edge decomposition as part of our algorithms.

Fact 1.3. *If G is connected, then the kernel of the corresponding Laplacian $\boldsymbol{L}$ is the span of the vector* $\mathbf{1}$.

1.3 Solutions to Linear Equations

For Laplacian solvers, the approximation error of an approximate solution $\boldsymbol{x}$ to a system $\boldsymbol{L}\boldsymbol{x} = \boldsymbol{b}$ is measured by the ϵ s.t.

$$\left\| \tilde{\boldsymbol{x}} - \boldsymbol{L}^\dagger \boldsymbol{b} \right\|_{\boldsymbol{L}} \leq \epsilon \left\| \boldsymbol{L}^\dagger \boldsymbol{b} \right\|_{\boldsymbol{L}}.$$

A key tool in iterative methods is *preconditioners*. Given a linear equation $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ in a square matrix $\boldsymbol{A}$, a preconditioner for $\boldsymbol{A}$ is a matrix that in some sense approximates $\boldsymbol{A}$, but is easier to

invert. In iterative methods, if we can compute a preconditioner for the matrix, we usually replace the condition number dependence in the running time of the iterative method with a dependence on the approximation quality between $\boldsymbol{A}$ and the preconditioner, at the cost of having to apply the inverse of the preconditioner.

1.3.1 Preconditioned Iterative Refinement for Positive Semi-Definite Matrices

We briefly introduce Preconditioned Iterative Refinement [Hig02, Chapter 12] to solve the system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, where $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{b} \in \mathbb{R}^n$. Suppose $\boldsymbol{Z}$ is a preconditioner for $\boldsymbol{A}$. Preconditioned Iterative Refinement refers to the procedure which computes the iterates below.

$$\boldsymbol{x}^{(0)} = \boldsymbol{0}, \qquad \boldsymbol{x}^{(i+1)} = \boldsymbol{x}^{(i)} - \boldsymbol{Z}^{\dagger}(\boldsymbol{A}\boldsymbol{x}^{(i)} - \boldsymbol{b}),$$

We use $\textsc{PreconIterRefinement}(\boldsymbol{A}, \boldsymbol{Z}, \epsilon, \boldsymbol{b})$ to denote the routine which performs preconditioned iteration refinement as described above, to compute and return $\boldsymbol{x}^{(t)}$, with $t = 3\log\frac{1}{\epsilon}$.

Theorem 1.4. *Consider a positive semi-definite matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and a vector $\boldsymbol{b} \in \mathbb{R}^n$. Suppose $\boldsymbol{Z}$ is a linear operator s.t. $\boldsymbol{Z} \approx_{1/2} \boldsymbol{A}$. Then for all $0 < \epsilon \leq 1/2$, $\textsc{PreconIterRefinement}(\boldsymbol{A}, \boldsymbol{Z}, \epsilon, \boldsymbol{b})$ returns $\boldsymbol{x}^{(t)}$, where $t = \lceil 3\log\frac{1}{\epsilon}\rceil$, s.t. $\left\|\boldsymbol{x}^{(t)} - \boldsymbol{A}^{\dagger}\boldsymbol{b}\right\|_{\boldsymbol{A}} \leq \epsilon \left\|\boldsymbol{A}^{\dagger}\boldsymbol{b}\right\|_{\boldsymbol{A}}$.*

1.3.2 Schur Complements

Gaussian Elimination is a classical algorithm for solving systems of linear equations. It is closely related to the notion of Schur Complements, which we introduce in this section. Suppose $\boldsymbol{M} \in \mathbb{R}^{n\times n}$ is a square matrix and $F, C \subset [n]$ is a partition of $[n]$ into two sets. W.l.o.g. taking F to be the first $|F|$ indices of $[n]$, we can write

$$\boldsymbol{M} = \begin{pmatrix} \boldsymbol{M}_{F,F} & \boldsymbol{M}_{F,C} \\ \boldsymbol{M}_{C,F} & \boldsymbol{M}_{C,C} \end{pmatrix}.$$

When $\boldsymbol{M}_{F,F}$ is invertible, we define the Schur complement of $\boldsymbol{M}$ onto C as

$$\textsc{Sc}[\boldsymbol{M}]_C \stackrel{\text{def}}{=} \boldsymbol{M}_{C,C} - \boldsymbol{M}_{C,F}(\boldsymbol{M}_{F,F})^{-1}\boldsymbol{M}_{F,C}.$$

The Schur complement is related to inverses and the LDU decompositions of a matrix. One way to see this is through the blockwise LDU decomposition of a matrix

$$\boldsymbol{M} = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{M}_{C,F}\boldsymbol{M}_{F,F}^{-1} & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{M}_{F,F} & \boldsymbol{0} \\ \boldsymbol{0} & \textsc{Sc}[\boldsymbol{M}]_C \end{pmatrix} \begin{pmatrix} \boldsymbol{I} & \boldsymbol{M}_{F,F}^{-1}\boldsymbol{M}_{F,C} \\ \boldsymbol{0} & \boldsymbol{I} \end{pmatrix}$$

This also implies that when $\boldsymbol{M}$ is invertible,

$$\begin{aligned} \boldsymbol{M}^{-1} &= \begin{pmatrix} \boldsymbol{I} & -\boldsymbol{M}_{F,F}^{-1}\boldsymbol{M}_{F,C} \\ \boldsymbol{0} & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{M}_{F,F}^{-1} & \boldsymbol{0} \\ \boldsymbol{0} & \textsc{Sc}[\boldsymbol{M}]_C^{-1} \end{pmatrix} \begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \\ -\boldsymbol{M}_{C,F}\boldsymbol{M}_{F,F}^{-1} & \boldsymbol{I} \end{pmatrix} \\ &= \begin{pmatrix} \boldsymbol{M}_{F,F}^{-1} + \boldsymbol{M}_{F,F}^{-1}\boldsymbol{M}_{F,C}\textsc{Sc}[\boldsymbol{M}]_C^{-1}\boldsymbol{M}_{C,F}\boldsymbol{M}_{F,F}^{-1} & -\boldsymbol{M}_{F,F}^{-1}\boldsymbol{M}_{F,C}\textsc{Sc}[\boldsymbol{M}]_C^{-1} \\ -\textsc{Sc}[\boldsymbol{M}]_C^{-1}\boldsymbol{M}_{C,F}\boldsymbol{M}_{F,F}^{-1} & \textsc{Sc}[\boldsymbol{M}]_C^{-1} \end{pmatrix}. \end{aligned}$$

This in turn implies immediately that

$$(\boldsymbol{M}^{-1})_{C,C} = \textsc{Sc}[\boldsymbol{M}]_C^{-1}\,.$$

Suppose that $C_2 \subseteq C_1 \subseteq [n]$. Then

$$\textsc{Sc}[\boldsymbol{M}]_{C_2} = \textsc{Sc}\big[\textsc{Sc}[\boldsymbol{M}]_{C_1}\big]_{C_2}\,.$$

This follows from noting that $(\boldsymbol{M}^{-1})_{C_2,C_2} = ((\boldsymbol{M}^{-1})_{C_1,C_1})_{C_2,C_2}$ and then inverting both matrices.

This in turn tells us that we can compute Schur Complements in stepwise manner, first Schur complementing onto $n-1$ variables (eliminating only one variable), then Schur complementing onto $n-2$ of the remaining variables etc. This also holds when $\boldsymbol{M}$ is not invertible, but we omit the proof. Our proof still requires that the matrix on indices $F = [n] \setminus C_2$, denoted by $\boldsymbol{M}_{F,F}$, is invertible.

The blockwise LDU decomposition can also be written as a sum of the zero-padded Schur complement and a term that agrees with $\boldsymbol{M}$ on all except the $\boldsymbol{M}_{C,C}$ block.

$$\boldsymbol{M} = \begin{pmatrix} \boldsymbol{M}_{F,F} & \boldsymbol{M}_{F,C} \\ \boldsymbol{M}_{C,F} & \boldsymbol{M}_{C,F}\boldsymbol{M}_{F,F}^{-1}\boldsymbol{M}_{F,C} \end{pmatrix} + \begin{pmatrix} \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \textsc{Sc}[\boldsymbol{M}]_C \end{pmatrix}$$

Fact 1.5 (Schur Complements and Pseudo-Inverses)**.** *Suppose $\boldsymbol{L}$ is an undirected Laplacian. Let $\boldsymbol{S} = \textsc{Sc}[\boldsymbol{L}]_C^{\dagger}$. Then $\boldsymbol{\Pi}_{\boldsymbol{S}}(\boldsymbol{L}^{\dagger})_{C,C}\boldsymbol{\Pi}_{\boldsymbol{S}} = \textsc{Sc}[\boldsymbol{L}]_C^{\dagger}$.*

Claim 1.6. *Consider $n \times n$ Laplacians $\boldsymbol{A}, \boldsymbol{B} \succeq \boldsymbol{0}$, s.t. for some $0 < \epsilon < 1$ we have $\boldsymbol{A} \approx_\epsilon \boldsymbol{B}$. Let $F \subseteq [n]$. Then*

$$\textsc{Sc}[\boldsymbol{A}]_F \approx_\epsilon \textsc{Sc}[\boldsymbol{B}]_F\,.$$

1.3.3 Gaussian Elimination, Cholesky Factorization, and LU-decomposition

An LU-decomposition of a square matrix $\boldsymbol{M} \in \mathbb{R}^{n\times n}$ is a factorization $\boldsymbol{M} = \boldsymbol{\mathcal{L}}\boldsymbol{\mathcal{U}}$, where $\boldsymbol{\mathcal{L}}$ is a lower-triangular matrix and $\boldsymbol{\mathcal{U}}$ is an upper-triangular matrix, both up to a permutation (see Section 1.1).

When $\boldsymbol{M}$ is non-singular, linear equations $\boldsymbol{\mathcal{L}}\boldsymbol{y} = \boldsymbol{b}$ and $\boldsymbol{\mathcal{U}}\boldsymbol{x} = \boldsymbol{y}$ and be solved by forward and backward substitution algorithms respectively (e.g. see [TBI97]), which run in time $O(\text{nnz}(\boldsymbol{\mathcal{L}}))$ and $O(\text{nnz}(\boldsymbol{\mathcal{U}}))$. I.e. $\boldsymbol{\mathcal{L}}^{-1}$ and $\boldsymbol{\mathcal{U}}^{-1}$ can be applied in time proportional to the number of non-zeros in $\boldsymbol{\mathcal{L}}$ and $\boldsymbol{\mathcal{U}}$ respectively. This means that if a decomposition $\boldsymbol{\mathcal{L}}, \boldsymbol{\mathcal{U}}$ of $\boldsymbol{M}$ is known, then linear systems in $\boldsymbol{M}$ be solved in time $O(\text{nnz}(\boldsymbol{\mathcal{L}}) + \text{nnz}(\boldsymbol{\mathcal{U}}))$, since $\boldsymbol{M}\boldsymbol{x} = \boldsymbol{b}$ implies $\boldsymbol{x} = \boldsymbol{\mathcal{U}}^{-1}\boldsymbol{\mathcal{L}}^{-1}\boldsymbol{b}$.

When $\boldsymbol{M}$ is singular the same forward and backward substitution algorithms can be used to compute the pseudo-inverse, in some situations. We focus on a special case where we can instead factor $\boldsymbol{M}$ as $\boldsymbol{M} = \boldsymbol{\mathcal{L}}'\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{U}}'$, where $\boldsymbol{\mathcal{D}}$ is singular and $\boldsymbol{\mathcal{L}}', \boldsymbol{\mathcal{U}}'$ are non-singular.

For the case of interest to us, Undirected Laplacians of connected graphs, this slightly modified factorization can be trivially obtained from an LU-decomposition, by $\boldsymbol{\mathcal{L}}'$ equal to $\boldsymbol{\mathcal{L}}$ except $\boldsymbol{\mathcal{L}}(n,n) = 1$ and $\boldsymbol{\mathcal{U}}'$ equal to $\boldsymbol{\mathcal{U}}$ except $\boldsymbol{\mathcal{U}}(n,n) = 1$ and taking $\boldsymbol{\mathcal{D}}$ to be the identity matrix, except $\boldsymbol{\mathcal{D}}(n,n) = 0$.

Now, by Fact 1.1,

$$\boldsymbol{M}^{\dagger} = \boldsymbol{\Pi}_{\boldsymbol{M}}\boldsymbol{\mathcal{L}}'^{-1}\boldsymbol{\mathcal{D}}^{\dagger}\boldsymbol{\mathcal{U}}'^{-1}\boldsymbol{\Pi}_{\boldsymbol{M}^{\top}}.$$

For connected Laplacians, the kernel and co-kernel are always the span of the $\mathbf{1}$ vector, so we can apply $\mathbf{\Pi}_{\boldsymbol{M}}$ and $\mathbf{\Pi}_{\boldsymbol{M}^\top}$ efficiently.

Gaussian Elimination is an algorithm that can be used to compute an LU-decomposition of some matrices. Gaussian Elimination proceeds by writing $\boldsymbol{M}$ as the sum of a rank 1 term that agrees with $\boldsymbol{M}$ on the first row and column and a (zero-padded) Schur complement:

As a convenient notational convention, we define $\boldsymbol{S}^{(0)} \stackrel{\text{def}}{=} \boldsymbol{M}$, the "0th" Schur complement. We then define

$$\begin{aligned}
\boldsymbol{l}_i &= (\boldsymbol{S}^{(i-1)}(i,i)^{1/2})^{-1}\boldsymbol{S}^{(i-1)}(:,i) \\
\boldsymbol{u}_i^\top &= (\boldsymbol{S}^{(i-1)}(i,i)^{1/2})^{-1}\boldsymbol{S}^{(i-1)}(i,:) \\
\boldsymbol{S}^{(i)} &= \boldsymbol{S}^{(i-1)} - \boldsymbol{l}_i\boldsymbol{u}_i^\top,
\end{aligned} \tag{2}$$

unless $\boldsymbol{S}^{(i-1)}(i,i) = 0$, in which case we define $\boldsymbol{S}^{(i)} = \boldsymbol{S}^{(i-1)}$, if $\boldsymbol{S}^{(i-1)}(:,i) = \mathbf{0}$ and $\boldsymbol{S}^{(i-1)}(i,:) = \mathbf{0}$. We do not define the Schur complement when the diagonal is zero but off-diagonals are not. By setting

$$\boldsymbol{\mathcal{L}} = \begin{pmatrix} \boldsymbol{l}_1 & \boldsymbol{l}_2 & \cdots & \boldsymbol{l}_n \end{pmatrix}$$

and

$$\boldsymbol{\mathcal{U}}^\top = \begin{pmatrix} \boldsymbol{u}_1 & \boldsymbol{u}_2 & \cdots & \boldsymbol{u}_n \end{pmatrix}$$

we get an LU-decomposition $\boldsymbol{M} = \boldsymbol{\mathcal{L}}\boldsymbol{\mathcal{U}}$.

It can be shown that Gaussian Elimination as described above always produces an LU-decomposition when applied an Undirected Laplacian. For some other classes of matrices, it does not always succeed (e.g. if the first column has a zero diagonal but other non-zero entries).

When $\boldsymbol{M}$ is a positive semi-definite matrix, it can be decomposed as $\boldsymbol{M} = \boldsymbol{\mathcal{L}}\boldsymbol{\mathcal{L}}^\top$, i.e. an LU decomposition where the upper triangular matrix is the transpose of the lower triangular matrix. This special case is known as a *Cholesky factorization*. When the Gaussian Elimination algorithm described above is applied to a Laplacian matrix it computes the Cholesky factorization.

Definition 1.7 (Partial LU-decomposition and Partial Cholesky Factorization)**.** If Gaussian Elimination is terminated after k steps (see Equation (2)), we get a decomposition where

$$\boldsymbol{\mathcal{L}} = \begin{pmatrix} \boldsymbol{l}_1 & \boldsymbol{l}_2 & \cdots & \boldsymbol{l}_k \end{pmatrix}$$

and

$$\boldsymbol{\mathcal{U}}^\top = \begin{pmatrix} \boldsymbol{u}_1 & \boldsymbol{u}_2 & \cdots & \boldsymbol{u}_k \end{pmatrix}$$

s.t. $\boldsymbol{M} = \boldsymbol{S}^{(k)} + \boldsymbol{\mathcal{L}}\boldsymbol{\mathcal{U}}$, where $\boldsymbol{S}^{(k)}$ is zero except on indices in $C \times C$ where $C = \{k+1, \ldots, n\}$ and $(\boldsymbol{S}^{(k)})_{C,C} = \textsc{Sc}[\boldsymbol{M}]_C$. Letting $F = [n] \setminus C$, this decomposition can also be written as

$$\boldsymbol{M} = \begin{pmatrix} \boldsymbol{\mathcal{L}}_{C,F} & \mathbf{0} \\ \boldsymbol{\mathcal{L}}_{C,F} & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{I} & \mathbf{0} \\ \mathbf{0} & (\boldsymbol{S}^{(k)})_{C,C} \end{pmatrix} \begin{pmatrix} \boldsymbol{I} & \boldsymbol{\mathcal{U}}_{F,C} \\ \mathbf{0} & \boldsymbol{\mathcal{U}}_{C,C} \end{pmatrix}$$

We refer to $\boldsymbol{\mathcal{L}}, \boldsymbol{\mathcal{U}}, \boldsymbol{S}$ as a *partial LU-decomposition* to set C, and when $\boldsymbol{M}$ is positive semi-definite, so $\boldsymbol{\mathcal{L}} = \boldsymbol{\mathcal{U}}^\top$, we refer to $\boldsymbol{\mathcal{L}}, \boldsymbol{S}$ as a *partial Cholesky factorization* to set C.

Definition 1.8 (Approximate Cholesky Factorization and Partial Cholesky Factorization)**.** We refer to $\boldsymbol{\mathcal{L}}$ as an *ϵ-approximate Cholesky factorization* of a matrix $\boldsymbol{M}$ if $\boldsymbol{\mathcal{L}}$ is a Cholesky factorization of a matrix $\tilde{\boldsymbol{M}}$ s.t. $\tilde{\boldsymbol{M}} \approx_\epsilon \boldsymbol{M}$.

We refer to $\boldsymbol{\mathcal{L}}, \boldsymbol{S}$ as an *ϵ-approximate partial Cholesky factorization* to a set C of a matrix $\boldsymbol{M}$ if $\boldsymbol{\mathcal{L}}, \boldsymbol{S}$ is a partial Cholesky factorization to a set C of a matrix $\tilde{\boldsymbol{M}}$ s.t. $\tilde{\boldsymbol{M}} \approx_\epsilon \boldsymbol{M}$.

1.3.4 Schur Complements and Closure

Some classes of matrices have the property that if $\boldsymbol{M}$ is a matrix in the class, then for any subset C of the indices of $\boldsymbol{M}$, the Schur complement $\textsc{Sc}[\boldsymbol{M}]_C$ is also in that class.

This observation, combined with the fact that some classes of matrices can be well-approximated with sparse matrices of the same class, is at the core of all the fast algorithms for solving systems of linear equations that we develop in this dissertation.

Positive definite and positive semi-definite matrices both have the property of being closed under Schur complement, i.e. the Schur complement of a positive definite matrix is a positive definite matrix and the Schur complement of a positive semi-definite matrix is positive semi-definite.

Below, we state the closure property for the matrix classes that we use in this dissertation.

Fact 1.9. *For each of the following classes, it holds that if $\boldsymbol{M}$ is a matrix of this class with index set $[n]$, then for all $C \subseteq [n]$, $\textsc{Sc}[\boldsymbol{L}]_C$ is a matrix of the same class.*

- *Positive Definite Matrices.*
- *Positive Semi-Definite Matrices.*
- *Undirected Laplacians.*

1.3.5 The Clique Structure of Schur Complements

In the previous section, we introduced Schur complement closure properties for several classes of matrices. In this section, we show how to prove the closure property for Schur complements of Laplacians, while also observing that the Schur complement of a Laplacian onto $n-1$ indices has additional structure that will help us develop algorithms for approximating these Schur complements.

Given a Laplacian $\boldsymbol{L}$, let $\textsc{St}[\boldsymbol{L}]_v \in \mathbb{R}^{n \times n}$ denote the Laplacian corresponding to the edges incident on vertex v (the *star* on v), i.e.

$$\textsc{St}[\boldsymbol{L}]_v \stackrel{\text{def}}{=} \sum_{e \in E : e \ni v} w(e) \boldsymbol{b}_e \boldsymbol{b}_e^\top. \tag{3}$$

For example, we denote the first column of $\boldsymbol{L}$ by $\begin{pmatrix} d \\ -\boldsymbol{a} \end{pmatrix}$, then $\textsc{St}[\boldsymbol{L}]_1 = \begin{bmatrix} d & -\boldsymbol{a}^\top \\ -\boldsymbol{a} & \operatorname{diag}(\boldsymbol{a}) \end{bmatrix}$. We can write the Schur complement $\textsc{Sc}[\boldsymbol{L}]_{[n]\setminus\{v\}}$ as

$$\textsc{Sc}[\boldsymbol{L}]_{[n]\setminus\{v\}} = \boldsymbol{L} - \textsc{St}[\boldsymbol{L}]_v + \textsc{St}[\boldsymbol{L}]_v - \frac{1}{\boldsymbol{L}(v,v)} \boldsymbol{L}(:,v)\boldsymbol{L}(v,:).$$

It is immediate that $\boldsymbol{L} - \textsc{St}[\boldsymbol{L}]_v$ is a Laplacian matrix, since $\boldsymbol{L} - \textsc{St}[\boldsymbol{L}]_v = \sum_{e \in E : e \not\ni v} w(e) \boldsymbol{b}_e \boldsymbol{b}_e^\top$. A more surprising (but well-known) fact is that

$$\textsc{Cl}[\boldsymbol{L}]_v \stackrel{\text{def}}{=} \textsc{St}[\boldsymbol{L}]_v - \frac{1}{\boldsymbol{L}(v,v)} \boldsymbol{L}(:,v)\boldsymbol{L}(v,:) \tag{4}$$

is also a Laplacian, and its edges form a clique on the neighbors of v. It suffices to show it for $v = 1$. We write $i \sim j$ to denote $(i,j) \in E$. Then

$$\textsc{Cl}[\boldsymbol{L}]_1 = \begin{bmatrix} \mathbf{0} & \mathbf{0}^\top \\ \mathbf{0} & \operatorname{diag}(\boldsymbol{a}) - \frac{\boldsymbol{a}\boldsymbol{a}^\top}{d} \end{bmatrix} = \sum_{i \sim 1} \sum_{j \sim 1} \frac{w(1,i)w(1,j)}{d} \boldsymbol{b}_{(i,j)} \boldsymbol{b}_{(i,j)}^\top.$$

Thus $\textsc{Sc}[\boldsymbol{L}]_{[n]\setminus\{v\}}$ is a Laplacian since it is a sum of two Laplacians. By induction, for all $C \subseteq [n]$, $\textsc{Sc}[\boldsymbol{L}]_C$ is a Laplacian.

2 Edgewise Elimination

In this section, we describe an alternative approach to factorizing a Laplacian $\boldsymbol{L}$ in terms of a product of upper and lower triangular matrices. Ultimately, the goal is to produce an approximate factorization $\boldsymbol{\mathcal{L}}\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{L}}^\top$ of a Laplacian $\boldsymbol{L}$, where $\boldsymbol{\mathcal{D}}$ is diagonal and $\boldsymbol{\mathcal{L}}$ is lower-triangular, $\boldsymbol{L} \approx \boldsymbol{\mathcal{L}}\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{L}}^\top$.

We can then get a linear equation solver for $\boldsymbol{L}\boldsymbol{x} = \boldsymbol{b}$, by using the preconditioned iterative refinement described in Theorem 1.4 with $\boldsymbol{A} \leftarrow \boldsymbol{L}$ and $\boldsymbol{Z} \leftarrow \boldsymbol{\mathcal{L}}\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{L}}^\top$. Note that in practice, the iterative algorithm we use is Preconditioned Conjugate Gradient.

We assume we are dealing with a connected graph.

We let

$$\boldsymbol{L} = \begin{pmatrix} d & -w & -\boldsymbol{a}^\top \\ -w & \begin{pmatrix} w & \mathbf{0}^\top \\ \mathbf{0} & \operatorname{diag}(\boldsymbol{a}) \end{pmatrix} & \\ -\boldsymbol{a} & & \end{pmatrix} + \boldsymbol{L}_{-1}$$

Degree-1 elimination. When eliminating a vertex of degree 1, we use

$$\begin{pmatrix} w & 0 & \mathbf{0}^\top \\ 0 & & \\ \mathbf{0} & \boldsymbol{L}_{-1} & \end{pmatrix} = \begin{pmatrix} 1 & 0 & \mathbf{0}^\top \\ 1 & & \\ \mathbf{0} & \boldsymbol{I} & \end{pmatrix} \begin{pmatrix} w & -w & \mathbf{0}^\top \\ -w & \begin{pmatrix} w & \mathbf{0}^\top \\ \mathbf{0} & \mathbf{0} \end{pmatrix} & \\ \mathbf{0} & & \end{pmatrix} + \boldsymbol{L}_{-1} \begin{pmatrix} 1 & 1 & \mathbf{0}^\top \\ 0 & & \\ \mathbf{0} & \boldsymbol{I} & \end{pmatrix}$$

And hence, by applying inverses of the outer matrices in the factorization,

$$\begin{pmatrix} w & -w & \mathbf{0}^\top \\ -w & \begin{pmatrix} w & \mathbf{0}^\top \\ \mathbf{0} & \mathbf{0} \end{pmatrix} & \\ \mathbf{0} & & \end{pmatrix} + \boldsymbol{L}_{-1} = \begin{pmatrix} 1 & 0 & \mathbf{0}^\top \\ -1 & & \\ \mathbf{0} & \boldsymbol{I} & \end{pmatrix} \begin{pmatrix} w & 0 & \mathbf{0}^\top \\ 0 & & \\ \mathbf{0} & \boldsymbol{L}_{-1} & \end{pmatrix} \begin{pmatrix} 1 & -1 & \mathbf{0}^\top \\ 0 & & \\ \mathbf{0} & \boldsymbol{I} & \end{pmatrix}$$

Edge elimination. When the vertex we're in the process of eliminating has degree more than 1, we instead apply the following factorization, where $d = \mathbf{1}^\top \boldsymbol{a} + w$ and $\theta = w/d$.

$$\begin{pmatrix} d(1-\theta)^2 & 0 & -(1-\theta)\boldsymbol{a}^\top \\ 0 & \begin{pmatrix} w - w^2/d & -\frac{w}{d}\boldsymbol{a}^\top \\ -\frac{w}{d}\boldsymbol{a} & \operatorname{diag}(\boldsymbol{a}) \end{pmatrix} & \\ -(1-\theta)\boldsymbol{a} & & \end{pmatrix} + \boldsymbol{L}_{-1} \tag{5}$$

$$= \begin{pmatrix} 1-\theta & 0 & \mathbf{0}^\top \\ \theta & & \\ \mathbf{0} & & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} d & -w & -\boldsymbol{a}^\top \\ -w & \begin{pmatrix} w & \mathbf{0}^\top \\ \mathbf{0} & \mathrm{diag}(\boldsymbol{a}) \end{pmatrix} & \\ -\boldsymbol{a} & & \end{pmatrix} + \boldsymbol{L}_{-1} \begin{pmatrix} 1-\theta & \theta & \mathbf{0}^\top \\ 0 & & \\ \mathbf{0} & & \boldsymbol{I} \end{pmatrix}$$

Note that the matrix on the LHS is a Laplacian because $d(1-\theta)^2 = (1-\theta)\mathbf{1}^\top \boldsymbol{a}$. And again, by applying inverses of the outer matrices in the factorization, we have

$$\begin{pmatrix} d & -w & -\boldsymbol{a}^\top \\ -w & \begin{pmatrix} w & \mathbf{0}^\top \\ \mathbf{0} & \mathrm{diag}(\boldsymbol{a}) \end{pmatrix} & \\ -\boldsymbol{a} & & \end{pmatrix} + \boldsymbol{L}_{-1}$$
$$= \begin{pmatrix} \frac{1}{1-\theta} & 0 & \mathbf{0}^\top \\ \frac{-\theta}{1-\theta} & & \\ \mathbf{0} & & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} d(1-\theta)^2 & 0 & -(1-\theta)\boldsymbol{a}^\top \\ 0 & \begin{pmatrix} w - w^2/d & -\frac{w}{d}\boldsymbol{a}^\top \\ -\frac{w}{d}\boldsymbol{a} & \mathrm{diag}(\boldsymbol{a}) \end{pmatrix} & \\ -(1-\theta)\boldsymbol{a} & & \end{pmatrix} + \boldsymbol{L}_{-1} \begin{pmatrix} \frac{1}{1-\theta} & \frac{-\theta}{1-\theta} & \mathbf{0}^\top \\ 0 & & \\ \mathbf{0} & & \boldsymbol{I} \end{pmatrix}$$

Schur complement invariance. We can also see that the Schur complement onto the remaining vertices is

$$\boldsymbol{S} = \mathrm{Sc}\left[\begin{pmatrix} d & -w & -\boldsymbol{a}^\top \\ -w & \begin{pmatrix} w & \mathbf{0}^\top \\ \mathbf{0} & \mathrm{diag}(\boldsymbol{a}) \end{pmatrix} & \\ -\boldsymbol{a} & & \end{pmatrix} + \boldsymbol{L}_{-1} \right]_{[n]\setminus\{1\}} = \begin{pmatrix} w - w^2/d & -\frac{w}{d}\boldsymbol{a}^\top \\ -\frac{w}{d}\boldsymbol{a} & \mathrm{diag}(\boldsymbol{a}) - \frac{1}{d}\boldsymbol{a}\boldsymbol{a}^\top \end{pmatrix} + \boldsymbol{L}_{-1}$$
$$= \mathrm{Sc}\left[\begin{pmatrix} d(1-\theta)^2 & 0 & -(1-\theta)\boldsymbol{a}^\top \\ 0 & \begin{pmatrix} w - w^2/d & -\frac{w}{d}\boldsymbol{a}^\top \\ -\frac{w}{d}\boldsymbol{a} & \mathrm{diag}(\boldsymbol{a}) \end{pmatrix} & \\ -(1-\theta)\boldsymbol{a} & & \end{pmatrix} + \boldsymbol{L}_{-1} \right]_{[n]\setminus\{1\}}$$

2.1 Eliminating a vertex, one edge at a time

Let us summarize these observations into a statement about how to write a factorization of $\boldsymbol{L}$ that eliminates the first vertex, *which we will now denote by vertex 0.* Assume vertex 0 has degree k, and that its neighbors are vertices $1, 2, \ldots, k$.

$$\boldsymbol{L} = \begin{pmatrix} d & -\boldsymbol{a}^\top \\ -\boldsymbol{a} & \mathrm{diag}(\boldsymbol{a}) + \boldsymbol{L}_{-1} \end{pmatrix} \tag{6}$$

We denote the weight on the edges from vertex 0 to its neighbors by $\boldsymbol{a}(1), \boldsymbol{a}(2), \ldots, \boldsymbol{a}(k)$. We then write

$$\boldsymbol{L} = \boldsymbol{\mathcal{L}}_1 \boldsymbol{\mathcal{L}}_2 \ldots \boldsymbol{\mathcal{L}}_k \begin{pmatrix} \phi & \mathbf{0}^\top \\ \mathbf{0} & \boldsymbol{S} \end{pmatrix} \boldsymbol{\mathcal{L}}_k^\top \ldots \boldsymbol{\mathcal{L}}_2^\top \boldsymbol{\mathcal{L}}_1^\top \tag{7}$$

where $\boldsymbol{S} = \mathrm{Sc}[\boldsymbol{L}]_{[n]\setminus\{1\}}$, and where for $i < k$, we have

$$\boldsymbol{\mathcal{L}}_i = \begin{pmatrix} \frac{1}{1-\theta_i} & \mathbf{0}^\top \\ \frac{-\theta_i}{1-\theta_i}\boldsymbol{e}_i & \boldsymbol{I} \end{pmatrix}$$

where $\boldsymbol{e}_i$ is the i basis vector in dimension $n-1$, and $\theta_i = \frac{\boldsymbol{a}(i)\Pi_{j<i}(1-\theta_i)}{d\cdot\Pi_{j<i}(1-\theta)^2} = \frac{\boldsymbol{a}(i)}{d\cdot\Pi_{j<i}(1-\theta_i)}$. We can simplify this, using the observation that $\Pi_{j<i}(1-\theta_j) = \frac{d-\sum_{j<i}\boldsymbol{a}(j)}{d}$. Hence

$$\theta_i = \frac{\boldsymbol{a}(i)}{d - \sum_{j<i}\boldsymbol{a}(j)}.$$

The last factor is given by,

$$\boldsymbol{\mathcal{L}}_k = \begin{pmatrix} 1 & \mathbf{0}^\top \\ -\boldsymbol{e}_k & \boldsymbol{I} \end{pmatrix}$$

and $\phi = \boldsymbol{a}(k)\Pi_{j<k}(1-\theta_j) = \frac{\boldsymbol{a}(k)^2}{d}$.

2.2 Sampled Edgewise Elimination

We can think of the elimination procedure described in the previous section is proceeding through a sequence of edge eliminations as we apply partial factorization to the Laplacian. We adopt the same notation as in Section 2.1, and define $\boldsymbol{L}$ as in Equation (6). Here $\boldsymbol{L} \in \mathbb{R}^{n\times n}$, and $\boldsymbol{a} \in \mathbb{R}^{n-1}$. We also define $\boldsymbol{a}_i \in \mathbb{R}^{n-1}$ given by

$$\boldsymbol{a}_i(l) = \begin{cases} \boldsymbol{a}(l) & \text{for } l > i \\ 0 & \text{for } l \leq i \end{cases}$$

These partial factorizations can be used to define an intermediate resulting fromt partial factorization for each $i < k$.

$$\boldsymbol{L} = \boldsymbol{\mathcal{L}}_1\boldsymbol{\mathcal{L}}_2\cdots\boldsymbol{\mathcal{L}}_i\boldsymbol{L}_i\boldsymbol{\mathcal{L}}_i^\top\cdots\boldsymbol{\mathcal{L}}_2^\top\boldsymbol{\mathcal{L}}_1^\top$$

We used Equation (5) to observe that $\boldsymbol{L}_1$ is a Laplacian, and from this we can prove by induction that each $\boldsymbol{L}_i$ is a Laplacian.

In the notation of Equation (5), the contribution of the ith edge elimination (where $i < k$) to $\boldsymbol{S}$ is the Laplacian matrix

$$\boldsymbol{S}_i = \left(\operatorname{diag}\left(\tfrac{\boldsymbol{a}(i)\mathbf{1}^\top\boldsymbol{a}_i}{d}\boldsymbol{e}_i + \tfrac{\boldsymbol{a}(i)}{d}\boldsymbol{a}_i\right) - \tfrac{\boldsymbol{a}(i)}{d}\left(\boldsymbol{e}_i\boldsymbol{a}_i^\top + \boldsymbol{a}_i\boldsymbol{e}_i^\top\right)\right) \tag{8}$$

In other words, we can write

$$\boldsymbol{S} = \boldsymbol{L}_{-1} + \sum_{i=1}^{k-1}\boldsymbol{S}_i$$

We will replace this with the Laplacian of a single randomly chosen reweighted edge, s.t. in expectation, this edge Laplacian yields $\boldsymbol{S}_i$. More generally, we choose the ith random index γ_i with according to a probability distribution given by

$$\Pr[\gamma_i = l] = p_i(l) \text{ where } p_i(l) = \frac{\boldsymbol{a}_i(l)}{\mathbf{1}^\top\boldsymbol{a}_i}.$$

We can also write this as

$$p_i(l) = \begin{cases} \frac{\boldsymbol{a}(l)}{\sum_{j>i}\boldsymbol{a}(j)} & \text{for } l > i \\ 0 & \text{for } l \leq i \end{cases}.$$

To make the expectations work out, if the outcome is $\gamma_i = l$, we then choose a weight of

$$\tilde{w}(i,l) = \frac{1}{p_i(l)}\frac{1}{d}\boldsymbol{a}(l)\boldsymbol{a}(i) = \frac{\boldsymbol{a}(i)\mathbf{1}^\top\boldsymbol{a}_i}{d}.$$

and we output the single edge Laplacian

$$\tilde{\boldsymbol{S}}_i = (\boldsymbol{e}_i - \boldsymbol{e}_{\gamma_i})(\boldsymbol{e}_i - \boldsymbol{e}_{\gamma_i})^\top$$

We can show that $\mathbf{E}_{\gamma_i}\left[\tilde{\boldsymbol{S}}_i\right] = \boldsymbol{S}_i$. Now, our output will be the approximate decomposition

$$\tilde{\boldsymbol{L}} = \boldsymbol{\mathcal{L}}_1\boldsymbol{\mathcal{L}}_2\cdots\boldsymbol{\mathcal{L}}_k \begin{pmatrix} \phi & \mathbf{0}^\top \\ \mathbf{0} & \boldsymbol{L}_{-1} + \sum_{i=1}^{k-1} \tilde{\boldsymbol{S}}_i \end{pmatrix} \boldsymbol{\mathcal{L}}_k^\top \cdots \boldsymbol{\mathcal{L}}_2^\top \boldsymbol{\mathcal{L}}_1^\top \tag{9}$$

which satisfies $\mathbf{E}\left[\tilde{\boldsymbol{L}}\right] = \boldsymbol{L}$.

This only describes elimination of the first vertex, so now we have to eliminate the rest.

2.3 Another View: Single Lower-triangular Factor

We can prove the following:

$$\begin{pmatrix} 0 & \mathbf{0}^\top \\ \mathbf{0} & \boldsymbol{S} \end{pmatrix} = \boldsymbol{\mathcal{L}}_1\boldsymbol{\mathcal{L}}_2\cdots\boldsymbol{\mathcal{L}}_k \begin{pmatrix} 0 & \mathbf{0}^\top \\ \mathbf{0} & \boldsymbol{S} \end{pmatrix} \boldsymbol{\mathcal{L}}_k^\top \cdots \boldsymbol{\mathcal{L}}_2^\top \boldsymbol{\mathcal{L}}_1^\top \tag{10}$$

and

$$\frac{1}{d}\begin{pmatrix} d \\ -\boldsymbol{a} \end{pmatrix}\begin{pmatrix} d \\ -\boldsymbol{a} \end{pmatrix}^\top = \boldsymbol{\mathcal{L}}_1\boldsymbol{\mathcal{L}}_2\cdots\boldsymbol{\mathcal{L}}_k \begin{pmatrix} \phi & \mathbf{0}^\top \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \boldsymbol{\mathcal{L}}_k^\top \cdots \boldsymbol{\mathcal{L}}_2^\top \boldsymbol{\mathcal{L}}_1^\top \tag{11}$$

To establish Equation (10), we observe that given any vector $\boldsymbol{b} \in R^n$ and any matrix $\boldsymbol{C} \in R^{(n-1)\times d}$, with any number of columns d, we have

$$\begin{pmatrix} \boldsymbol{b} & \mathbf{0}^\top \\ | & \boldsymbol{I}_{(n-1)\times(n-1)} \end{pmatrix}\begin{pmatrix} \mathbf{0}^\top \\ \boldsymbol{C} \end{pmatrix} = \begin{pmatrix} \mathbf{0}^\top \\ \boldsymbol{C} \end{pmatrix}.$$

Repeatedly applying this observation the right hand side of Equation 10 lets us conclude that it equals the left hand side, proving the equation holds.

From the preliminaries, we know that it is possible to write

$$\boldsymbol{L} = \begin{pmatrix} 0 & \mathbf{0}^\top \\ \mathbf{0} & \boldsymbol{S} \end{pmatrix} + \frac{1}{d}\begin{pmatrix} d \\ -\boldsymbol{a} \end{pmatrix}\begin{pmatrix} d \\ -\boldsymbol{a} \end{pmatrix}^\top$$

Equating this with Equation (7), simplifying by Equation (10), can subtracting $\begin{pmatrix} 0 & \mathbf{0}^\top \\ \mathbf{0} & \boldsymbol{S} \end{pmatrix}$ on both sides, we can then establish Equation (11).

By similar observations, we can express the approximate factorization in Equation (9) as

$$\tilde{\boldsymbol{L}} = \begin{pmatrix} d & \mathbf{0}^\top \\ -\boldsymbol{a} & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} 1/d & \mathbf{0}^\top \\ \mathbf{0} & \boldsymbol{L}_{-1} + \sum_{i=1}^{k-1} \tilde{\boldsymbol{S}}_i \end{pmatrix} \begin{pmatrix} d & -\boldsymbol{a}^\top \\ \mathbf{0} & \boldsymbol{I} \end{pmatrix}. \tag{12}$$

2.4 Computing a Full Approximate Factorization

Sections 2.1 and 2.3 described two different approaches to computing an approximate factorization that eliminates the first variable of $\boldsymbol{L}$, summarized in Equations (9) and (12) respectively. The factorizations are essentially equivalent: they correspond to the same linear operator, but they suggest two different ways to apply that operator.

In this section, we briefly remark how to repeatedly eliminate vertices to obtain a full factorization.

Factorization with one column per vertex. Let us see how Equation (12) can be repeatedly applied: Let $\boldsymbol{l}_1 = \begin{pmatrix} d \\ -\boldsymbol{a} \end{pmatrix}$ denote the first column in the lower triangular factor, and introduce a diagonal matrix $\boldsymbol{\mathcal{D}}$ where $\boldsymbol{\mathcal{D}}(1,1) = 1/d$ records the first diagonal of the middle factor. If we recursively compute a factorization in the same way of the remaining matrix $\boldsymbol{L}_{-1} + \sum_{i=1}^{k-1} \tilde{\boldsymbol{S}}_i \in \mathbb{R}^{(n-1)\times(n-1)}$, this yields a sequence of columns $\boldsymbol{l}_2 \in \mathbb{R}^{(n-1)}, \boldsymbol{l}_3 \in \mathbb{R}^{(n-2)}, \ldots, \boldsymbol{l}_n \in \mathbb{R}^1$ and diagonals of which we may record in $\boldsymbol{\mathcal{D}}$ s.t. if we let

$$\boldsymbol{\mathcal{L}} = \begin{pmatrix} \begin{pmatrix} \boldsymbol{l}_1 \\ | \end{pmatrix} & \begin{pmatrix} 0 \\ \boldsymbol{l}_2 \\ | \end{pmatrix} & \begin{pmatrix} 0 \\ 0 \\ \boldsymbol{l}_3 \end{pmatrix} & \cdots & \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \boldsymbol{l}_n \end{pmatrix} \end{pmatrix} \tag{13}$$

Then $\boldsymbol{L} \approx \boldsymbol{\mathcal{L}}\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{L}}^\top$.

Factorization with one factor per edge elimination. Now let us see how Equation (9) can be repeatedly applied to obtain a full factorization. Let us slightly modify the notation for the first elimination to write

$$\tilde{\boldsymbol{L}} = \boldsymbol{\mathcal{L}}_1^{(1)}\boldsymbol{\mathcal{L}}_2^{(1)}\cdots\boldsymbol{\mathcal{L}}_{k_1}^{(1)} \begin{pmatrix} \phi_1 & \boldsymbol{0}^\top \\ \boldsymbol{0} & \boldsymbol{L}_{-1} + \sum_{i=1}^{k-1} \tilde{\boldsymbol{S}}_i \end{pmatrix} (\boldsymbol{\mathcal{L}}_{k_1}^{(1)})^\top \cdots (\boldsymbol{\mathcal{L}}_2^{(1)})^\top (\boldsymbol{\mathcal{L}}_1^{(1)})^\top,$$

where k_1 denotes the degree of the vertex we eliminated.

Now if we recursively factor the remaining matrix $\boldsymbol{L}_{-1} + \sum_{i=1}^{k-1} \tilde{\boldsymbol{S}}_i$, we can eventually write

$$\boldsymbol{L} \approx \left(\Pi_{i=1}^{n}\Pi_{j=1}^{k_i} \begin{pmatrix} \boldsymbol{I}_{i-1} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\mathcal{L}}_j^{(i)} \end{pmatrix}\right) \boldsymbol{\Phi} \left(\Pi_{i=1}^{n}\Pi_{j=1}^{k_i} \begin{pmatrix} \boldsymbol{I}_{i-1} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\mathcal{L}}_j^{(i)} \end{pmatrix}\right)^\top \tag{14}$$

where $\boldsymbol{\Phi}$ is a diagonal matrix with $\boldsymbol{\Phi}(i,i) = \phi_i$.

Note that the many subblocks with identity matrices never lead to computations that we have to perform. When the operators or their inverses are applied, these blocks correspond leave vector entries unchanged.

Note also that k_i is the degree of the ith vertex being eliminated, in the approximate Schur complement that it is being eliminated from. I.e. it is not the original degree or the degree in the exact Schur complement of the original Laplacian onto the remaining vertices.

References

[Hig02] Nicholas J Higham. *Accuracy and stability of numerical algorithms*. Siam, 2002.

[TBI97] Lloyd N Trefethen and David Bau III. *Numerical linear algebra*, volume 50. Siam, 1997.