Abstract  Laplacian matrices provide a rich, linear algebraic approach to studying the properties of graphs, and have significant applications in electrical flows, random walks, graph partitioning, and even machine learning. In many of these contexts, it is necessary to solve the linear system $\mathcal{L}x = b$ induced by the Laplacian, for which there are a variety of direct and iterative methods. However, when the condition number of a matrix — the ratio between its largest and smallest (in this case, non-zero) singular values — is too high, the system becomes sensitive to small perturbations of the input, and traditional methods fail. A 2016 paper by Cohen et al. tackles the problem of solving ill-conditioned Laplacian systems for directed graphs, presenting a crude iterative algorithm that solves an ill-conditioned Laplacian by reducing to a series of well-conditioned systems. In this project, we rigorously extend the analysis presented for the case of undirected graphs, and are able to achieve improved error bounds, generalized to take into account additional parameters. Although the error bounds proved in Cohen et al. required unfeasible parameters, we implement the algorithm in Julia and show that even with reasonable parameter values, it successfully solves to high accuracy a variety of extremely ill-conditioned systems on which existing solvers fail.

1 Introduction

Systems of equations of the form $\mathcal{L}x = b$, where $\mathcal{L} = D - A$ is the Laplacian of a graph, have received significant attention in the research community over the last two decades. Solving such a system has applications to diverse areas and problems, from graph partitioning and random walks to machine learning and medical imaging [1]. Certain applications of Laplacian solvers involve undirected graphs whose edge weights vary over many orders of magnitude, resulting in a high ratio between the Laplacian’s largest and smallest eigenvalues. In such an ill-conditioned system, small perturbations in the input can drastically reduce the accuracy of the numerically computed system; very high precision data types are one way to combat the issue. However, a recent paper by Cohen et al. [2] presents a reduction for approximately solving ill-conditioned, directed Laplacian systems. To do so, it makes $O(\log(n\kappa))$ calls to a solver of well-conditioned Laplacians, where $n$ is the number of nodes in the graph and $\kappa$ is an upper bound on the condition number of $\mathcal{L}$. Thus, the algorithm has only logarithmic dependence on condition number, and circumvents the challenges of the original ill-conditioned system. However, the error guarantee of [2] requires extreme parameter values, and achieves only the crude error bound

$$||x - \mathcal{L}^+b||_\mathcal{L} \leq \frac{1}{2}||\mathcal{L}^+b||_\mathcal{L}$$

For this project, we expand the theoretical development of the algorithm in the specific case of undirected graphs, implement it in Julia, and explore its performance on a variety of graphs.
1.1 Condition Number

1.1.1 General Case

The condition number of a symmetric, real-valued non-singular matrix $A$ with eigenvalues $|\lambda_1| < |\lambda_2| < \ldots < |\lambda_n|$ is traditionally defined as

\[
\frac{|\lambda_n|}{|\lambda_1|}
\]

Given a system $A\vec{x} = \vec{b}$, it roughly measures how stable $\vec{x}$ is to perturbations of $\vec{b}$. Matrices with high condition number are referred to as ill-conditioned, and it is generally much more difficult to solve for $A^{-1}\vec{b}$ in these cases.

1.1.2 Laplacians

Recall that the eigenvalues of $L$ are $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$, so $L$ is singular and the previous definition of condition number is undefined. However, we instead measure the finite condition number $\lambda_n/\lambda_2$.

We can bound the condition number of $L$ using the minimum edge weight $w_{\text{min}}$ and maximum degree $d_{\text{max}}$ of a connected graph. We first bound from below the smallest non-zero eigenvalue in terms of $w_{\text{min}}$:

\[
\lambda_2 \geq \frac{\pi^2 w_{\text{min}}}{n^2}
\]

This follows by noting that the path graph $P$ is a subgraph of a connected graph, so for an undirected connected graph’s Laplacian $U$ and Laplacian of the path graph $L_P$, $U \geq L_P$. Letting $L$ be the Laplacian of a weighted version of the previous graph, of minimum weight $w_{\text{min}}$, we then have

\[
x^T = \sum_{e=(i,j)} w_{ij} (x_i - x_j)^2 \geq w \sum_{e=(i,j)} (x_i - x_j)^2 = w_{\text{min}} (x^T U x)
\]

So $\lambda_2(L) \geq w_{\text{min}} \lambda_2(L_P)$. It is known that $\lambda_2(P) = 2(1 - \cos (\pi/n)) = 4 \sin \left(\frac{\pi}{2n}\right)^2 \approx \frac{\pi^2}{n^2}$ for large $n$; from this, the stated bound follows.

We also bound from above the largest non-zero eigenvalue in terms of $w_{\text{max}}$:

\[
\lambda_n \leq 2d_{\text{max}}
\]

This follows from the Greshgorin circle theorem, which states that for every eigenvalue $\lambda_i$ of a square matrix $A$

\[
\lambda_i \in B(a_{ii}, \sum_{j \neq i} |a_{ij}|)
\]

Then $L_{ii} \leq d_{\text{max}}$ and $\sum_{j \neq i} |a_{ij}| \leq d_{\text{max}}$ together imply $\lambda_n \leq 2d_{\text{max}}$.

Then for a connected graph of minimum edge weight $w_{\text{min}}$ and maximum weighted degree $d_{\text{max}}$, we then have

\[
\frac{\lambda_n}{\lambda_2} \leq \frac{2d_{\text{max}}n^2}{\pi^2 w_{\text{min}}}
\]

In applications such as maximum flow problems, where edge weights vary over many orders of magnitude (e.g. from 1 to $10^{50}$), this upper bound grows significantly and existing solvers fail.

1.2 Existing Algorithm

The algorithm proposed in [2] solves a potentially ill-conditioned system by reducing to a series of polynomially well-conditioned systems. To do so, it treats $\lambda_n$ and $\lambda_2$ separately. First, high-weight
edges are “contracted” to produce a coarse-grained approximation of the original graph, where the linear contraction operator $C$ is defined with respect to a partition of the vertices $S_1, ..., S_k$ as follows:

$$C(\vec{e}_i) = k \text{ such that } i \in S_k$$

In matrix form and assuming the vertices are labeled in order of membership, $C \in \mathbb{N}^{k \times n}$ looks like

$$
\begin{bmatrix}
1 & \cdots & 0 & \cdots & 0 \\
0 & \cdots & 0 & \cdots & 0 \\
\vdots & \ddots & \ddots & \ddots & \vdots \\
0 & \cdots & 0 & \cdots & 1
\end{bmatrix}
$$

As an operation on vectors, $C : \mathbb{R}^n \to \mathbb{R}^k$ and $C^T : \mathbb{R}^k \to \mathbb{R}^n$. $C(\vec{v})$ sums the entries of $\vec{v}$ according to the partition $S_1, ..., S_k$, and $C^T(\vec{v})$ “expands” $\vec{v}$ by copying $v_i$ to all nodes in $S_i$.

Intuitively, contraction makes the most sense when applied to Laplacian matrices: if $L$ corresponds to weighted graph on $n$ nodes, $CLC^T$ is the Laplacian of a contracted graph. In other words, the nodes of $S_i$ are clumped together into one single group-node, with edge weights between group-nodes equal to the total edge weight between their original member nodes. Observe also that although $C(A)$ has self-loops, $L = D - A$ ensures that the graph corresponding to $C(L)$ is indistinguishable from that of a graph with self-loops removed.

We show a simple example of the contraction operator below, for $S_1 = \{1, 2, 3\}$, $S_2 = \{4\}$, and $S_3 = \{5, 6\}$. The original graph is on the left, while the contracted version is on the right.

Necessary for the algorithm is also the projection operator, again defined with respect to a partition $S_1, ..., S_k$, such that $\text{Proj} : \mathbb{R}^n \to \mathbb{R}^n$ and $\text{Proj}(\vec{x})$ is the orthogonal projection of $\vec{x}$ onto the kernel of $C$. Explicitly:

$$\text{Proj}(\vec{x})_j = \vec{x}_j - \frac{1}{|S_k|} \sum_{v \in S_k} \vec{x}_v$$

It is easily shown that this formula is the desired orthogonal projection:

$$\text{Proj}(\vec{x}) \in \ker(C)$$

$$\sum_{i \in S_k} \text{Proj}(\vec{x})_i = \sum_{i \in S_k} \vec{x}_i - \sum_{i \in S_k} \vec{x}_i = 0$$

$\vec{x} - \text{Proj}(\vec{x}) \perp \ker(C)$ Let $\vec{v} \in \ker(C)$, and observe that $\vec{x} - \text{Proj}(\vec{x})$ is constant over each set $S_i$. Then $\langle \vec{v}, \vec{x} - \text{Proj}(\vec{x}) \rangle = 0$

Contracting the input graph reduces the maximum weight value and thus the largest eigenvalue to $\lambda'_n$; however, if $\lambda_2 << \lambda'_n$, the matrix may still be ill-conditioned. As such, [2] proposes adding a small multiple of the identity matrix to the contracted Laplacian. Doing so shifts up the entire spectrum of $L$, not just $\lambda_2$; however, assuming $\alpha > 0$ and $\lambda'_n > \lambda_2$, $\frac{\lambda'_n}{\lambda_2} > \frac{\lambda'_n + \alpha}{\lambda_2 + \alpha}$.

The complete algorithm proposed by [2] is shown below. Note that extra parameters have been added, for more flexible analysis in subsequent parts. In addition, we generalize to any class of
positive semidefinite matrices $M$, and functions $r = g(n)$. The exact algorithm of [2] is obtained by setting $r = (1000n^{10})^2$, $u = 2$, $h = 1$, and $M = I$.

**CrudeSolveIllConditioned($\mathcal{L}, \vec{b}, g(n), u, h$)**

**Input:** Undirected Laplacian $\mathcal{L} = D - A$, $\vec{b} \perp 1$, parameters $u, h$ and function $g(n)$

**Output:** An approximate solution $\vec{x}$ to $\mathcal{L}\vec{x} = \vec{b}$, with bound on $||\vec{x} - \mathcal{L}^+\vec{b}||_\mathcal{L}$ dependent on input parameters

1. Let $r = g(n)$
2. Let $w^0$ be the smallest edge in a maximal spanning tree of $\mathcal{L}$
3. let $\vec{x}^0 = 0$, $\vec{b}^0 = b$, $i = 0$
4. Loop:
   a. Let $C^i$ and $\text{Proj}^i$ be the contraction and projection operators, defined with respect to the connected components in the graph of $\mathcal{L}$ with weight $\geq w^i$
   b. Let $\vec{z}^i = C^{iT}(C^i\mathcal{L}C^{iT} + \frac{hw^i}{r}M_i)^{-1}C^i\vec{b}^i$ (a $\frac{1}{2}$ solver can be substituted for the inverse)
   c. Let $\vec{x}^{i+1} = \vec{x}^i + \vec{z}^i$
   d. Let $\vec{b}^{i+1} = \text{Proj}^i(\vec{b}^i - \mathcal{L}\vec{z}^i)$
   e. Let $w^{i+1}$ be the smallest edge weight in $\mathcal{L}$ that is $\geq w^i$. If none, stop looping.
   f. Let $w^{i+1} = uw^i$
   g. $i = i + 1$
5. Return $\vec{x}^{i+1}$

2 Proof of Correctness for Undirected Case

2.1 Exact Solves, Without Addition of Clique

To build intuition about why the algorithm works, it is helpful to consider the short and simple proof of the case where (1) all solves are exact, and (2) there is no offset by $\alpha M$. In this case, it is clear that the algorithm strongly resembles iterative refinement. An “approximate” solution is attained at each iteration (in the sense that it is not a solution to the original system), and each subsequent system solves for the error of the last. We will see, however, that under the assumption that $C^i\mathcal{L}C^{iT}$ can be exactly inverted, the solution returned is still precisely $\mathcal{L}^+\vec{b}$.

**Lemma 2.1** $\vec{x}^0 + \mathcal{L}^+\vec{b} = \mathcal{L}^+\vec{b}$

**Proof:** We proceed by induction. For $i = 0$, $\vec{x}^0 = 0$ and $\vec{b}^0 = b$, so the statement follows. Suppose then that $\vec{x}^i + \mathcal{L}^+\vec{b} = \mathcal{L}^+\vec{b}$. According to the algorithm,

\[
\vec{x}^i = C^{iT}(C^i\mathcal{L}C^{iT})^{-1}C^i\vec{b}^i
\]

\[
\vec{x}^{i+1} = \vec{x}^i + \vec{z}^i
\]

\[
\vec{b}^{i+1} = \text{Proj}^i(\vec{b}^i - \mathcal{L}\vec{z}^i)
\]

However, in the idealized case where $\alpha = 0$ and solves are exact, $\vec{b}^i - \mathcal{L}\vec{z}^i$ is already in the kernel of $C^i$:

\[
C^i(\vec{b}^i - \mathcal{L}\vec{z}^i) = C^i\vec{b}^i - C^i\mathcal{L}(C^i\mathcal{L}C^{iT})^{-1}C^i\vec{b}^i
\]
Once again, this implies that such that
This implies that $\sum \vec{x}$
Then $\tilde{x}(i+1) + \mathcal{L}^+ \tilde{b}(i+1) = \tilde{x}^i + \tilde{z} + \mathcal{L}^+ (\tilde{b} - \mathcal{L} \tilde{z}) = \tilde{x}^i + \mathcal{L}^+ \tilde{b} = \mathcal{L}^+ \tilde{b} \tag{5}$

Next, notice that the weight threshold above which edges are contracted increases by some positive factor $u > 1$ with each iteration of the algorithm. Thus in the last iteration $i$, weights above some threshold $t$ are contracted; however, $t$ will be greater than the maximum edge weight of the graph. In this situation, no edges are contracted: $C^i = I$, so $\tilde{b} = \mathcal{L} \tilde{z}$ precisely and $\tilde{b}(i+1) = 0$ at termination. (In fact, this holds for $\tilde{b}(i+1)$ even for inexact solves, as the projection of any vector onto the kernel of $I$ is simply the 0 vector; this will be crucial for the proof of correctness in the more complicated case.) Then by the previous lemma, $\tilde{x}(i+1) + \mathcal{L}^+ \tilde{b}(i+1) = \tilde{x}^{i+1} = \mathcal{L}^+ \tilde{b}$

2.2 Full Algorithm

Here, we adapt the proof found in [2] for the case of undirected Laplacians. It follows exactly the structure of the existing proof, and simplifies several steps that were previously specific to directed Laplacians. It also leaves error bounds in terms of three parameters of practical importance: $r$, the updates to $w^i$, and the matrix used to perturb the contracted system, $M$. (In [2], $M = I$.)

In addition, we improve a few error bounds and fill in several missing steps, including a rigorous expansion of two lemmas whose proofs required analysis of flows on graphs. The proofs of lemmas which required no alteration, simplification, or further explanation, are not included.

The entire method of proof centers around the following analog of Lemma 2.1, which is proved by induction in [2]:

$$\tilde{x}^i + \mathcal{L}^+ \tilde{b}^i = \sum_{j<i} \tilde{e}^j + \mathcal{L}^+ \tilde{b}^j$$

where $\tilde{e}^j$ and $\tilde{b}^j$ capture the error induced by adding a small multiple of $M$ and of the blackbox solver itself. As such, the rest of the analysis is dedicated to rounding the $\mathcal{L}$-norm of these error terms.

The first part of the proof seeks to understand the error, in terms of the $\mathcal{L}$-norm, induced by solving the shifted system. To do, the following lemma will be necessary.

**Lemma 2.2** Let $\mathcal{L}$ be a connected Laplacian and $\tilde{x}, \tilde{b} \neq 0$ with $\mathcal{L} \tilde{x} = \tilde{b}$. Then the maximum and minimum value of $\tilde{x}$ are obtained on the support of $\tilde{b}$

**Proof:** Let the set $S$ be all indices of nodes such that $\tilde{x}$ attains its maximum value $v$. If $S = V,$ the statement follows. If not:

$$\sum_{s \in S} \tilde{b}_s = \sum_{s \in S} \left( \sum_{j \in V} \tilde{x}_j \mathcal{L}_{sj} \right) = \sum_{s \in S} \left( \tilde{x}_s \mathcal{L}_{ss} + \sum_{j \neq s} \tilde{x}_j \mathcal{L}_{sj} \right) = \sum_{s \in S} \left( \tilde{x}_s \sum_{k \neq s} w_{sk} - \sum_{j \neq s} \tilde{x}_j w_{sj} \right)$$

But for $a, b \in S$, the terms $x_{a}w_{ab}$ and $-x_{a}w_{ba}$ will cancel, as will $x_{b}w_{ba}$ and $-x_{b}w_{ab}$. So, consolidating terms:

$$\sum_{s \in S, k \notin S} x_{s}w_{sk} - \sum_{s \in S, k \notin S} x_{k}w_{sk} > \sum_{s \in S, k \notin S} v_{w_{sk}} - \sum_{s \in S, k \notin S} v_{w_{sk}} = 0$$

This implies that $\sum_{s \in S} \tilde{b}_s > 0$, so $\text{supp}(\tilde{b})$ overlaps $S$. Similarly, we can set $S$ to indices of nodes such that $\tilde{x}$ attains its minimum value $v$:

$$\sum_{s \in S, k \notin S} x_{s}w_{sk} - \sum_{s \in S, k \notin S} x_{k}w_{sk} < \sum_{s \in S, k \notin S} v_{w_{sk}} - \sum_{s \in S, k \notin S} v_{w_{sk}} = 0$$

Once again, this implies that $S$ must overlap $\text{supp}(\tilde{b})$.

In addition, reasoning about electrical flows on graphs is used in subsequent lemmas. We include the necessary definitions and derivations here.
2.2.1 Graph Flow Preliminaries

Suppose we have a weighted, undirected graph $G = (V, E, w)$ with $n$ nodes and $m$ edges, and $\vec{b} \in \mathbb{R}^n$, $\vec{b} \perp \perp$. $\vec{b}$ is often referred to as the demand vector, inducing $\vec{x}$ such that $\mathcal{L}\vec{x} = \vec{b}$. Let $B$ be the matrix that is nonzero except when $i < j$ and $(i, j)$ is the $l$th edge of $E$, in which case $B_{ii} = 1$ and $B_{ij} = -1$. For convenience, let $B_e$ be the row of $B$ corresponding to edge $e$. Finally, let $W \in \mathbb{R}^{m \times n}$ be the diagonal matrix such that $W_{ii} = w_i$. Then the flow $y = W\mathcal{L}^{+}\vec{b}$ and in fact $\mathcal{L} = B^TWB$. It is standard to define the energy of a flow $y$ by

$$\sum_e \frac{y_e^2}{w_e} = ||\vec{b}||^2_{\mathcal{L}^+}$$

The last equivalence can be derived as follows [1]:

$$f = WB\vec{x} = W\mathcal{L}^{+}\vec{b} \implies f_e = w_eB_e \mathcal{L}^{+}\vec{b}$$

$$\sum_e \frac{f_e^2}{w_e} = \sum_e \frac{(w_eB_e \mathcal{L}^{+}\vec{b})^2}{w_e} = \sum_e \frac{(w_eB_e \mathcal{L}^{+}\vec{b})^T(w_eB_e \mathcal{L}^{+}\vec{b})}{w_e}$$

$$= \sum_e w_e \vec{b}^T \mathcal{L}^{+T}B_e^T \mathcal{L}^{+} \vec{b} = \vec{b}^T \mathcal{L}^{+} \vec{b} = ||\vec{b}||_{\mathcal{L}^+}$$

The following lemmas about flows on weighted graphs will be necessary for later steps.

**Lemma 2.3** Let $f$ be the flow induced by demands $b$, such that $f$ is supported on edges of weight at most $w$. Then $||b||_1 \leq 2n\sqrt{w}||b||_{\mathcal{L}^+}$

**Proof:**

$$||b||_1 \leq 2||f||_1 \leq 2n||f||_2 = 2n \sqrt{\sum_e f_e^2} = 2n \sqrt{w \sum_e \frac{f_e^2}{w}} \leq 2n \sqrt{w \sum_e \frac{f_e^2}{w}} = 2n\sqrt{w}||b||_{\mathcal{L}^+}$$

Note that the first inequality is because each edge’s flow can contribute to the demands of at most 2 nodes, while the second is by Cauch-Schwarz with vector length $\leq n^2$. ■

**Lemma 2.4** Let $f$ be the flow induced by demands $b$, such that nodes in the support of $b$ are connected by edges of weight at least $w$. Then $||b||_{\mathcal{L}^+} \leq \sqrt{\frac{n}{w}}||b||_1$

**Proof:** First, consider vectors $\vec{b}_j = e_u - e_v$ that have one entry of 1, one entry of $-1$, and zeros everywhere else. We will call these elementary vectors. Also, assume that the graph has edges of weight at least $w$. Then $\vec{b}_j^T\mathcal{L}^{+}\vec{b}_j$ is the effective resistance between $u$ and $v$, which is at most $\frac{n}{w}$ as $u$ and $v$ are connected by edges of weight at least $w$. For general $b$, we can always write

$$b = \sum_j p_j\vec{b}_j$$

such that $p_j > 0$, all $b_j$ are elementary vectors and

$$||b||_1 = \sum_j p_j||b_j||_1$$

To prove the existence of this decomposition, proceed by induction on the number of nonzero entries of $b$. Since we assume the demands $b$ sum to 0 (and ignore the trivial case of all-zero demands), our base case is 2. In this case, $b = k(e_u - e_v)$ for some $k > 0$, so the result holds. For $b$ with more
nonzero values, let $u$ be the nonzero coordinate of $b$ of smallest absolute value, and let $v$ be any other nonzero coordinate of opposite sign. Setting $b_1 = b(u)(e_u - e_v)$ and $\hat{b} = b - b_1$, we then have
\[
b = b_1 + \hat{b}
\]
\[
||\hat{b}||_1 = ||b_1||_1 + ||\hat{b}||_1
\]
But since $\hat{b}$ has one less nonzero entry than $b$, by induction there are elementary vectors $b_j$ such that
\[
\hat{b} = \sum_j p_j b_j
\]
\[
||\hat{b}||_1 = \sum_j p_j ||b_j||_1
\]
Also, $b_1 = b(u)(e_u - e_v)$ is a positive sum of elementary vectors for $b(u) > 0$, and if $b(u) < 0$, we can write $b_1 = -(b(u))(e_v - e_u)$. The decomposition then holds for all zero-sum $b$. We now write
\[
b = \sum_j p_j b_j
\]
that satisfies the previous property of $\ell 1$-additivity. Also, note that for elementary vectors $b_k \neq b_j$, $b_k^T \mathcal{L} b_j \leq b_j^T \mathcal{L} b_k$. (This holds for Laplacian matrices because they are non-negative on the diagonal and non-positive off the diagonal.) Returning to the lemma’s assertion:
\[
b^T \mathcal{L} b = \left( \sum_k p_k b_k^T \right) \left( \sum_j p_j \mathcal{L} b_j \right) = \sum_k p_k \left( \sum_j p_j b_k^T \mathcal{L} b_j \right) \leq \sum_k p_k \left( \sum_j p_j \frac{n}{w_j} \right) = \frac{n}{w} \left( 2 \sum_j p_j \right)
\]
However, $||b||_1 = \sum_j p_j ||b_j||_1 = 2 \sum_j p_j$. So, $||b||_{\mathcal{L}^+} \leq \frac{\sqrt{n}}{\sqrt{w}} ||b||_1$ as stated.\footnote{This proof was proposed by Dan Spielman.}

Returning to the proof structure of [2], suppose that for $\alpha > 0$, $\mathcal{L}$ connected, and $\hat{b} \in \text{im}(\mathcal{L})$ such that any two nodes in $\text{supp}(\hat{b})$ are connected by a path of only edges of weight $\geq \beta$, we have
\[
|| (\mathcal{L} + \alpha M) b + \mathcal{L} \hat{b} ||_{\mathcal{L}^+} \leq f(n) \sqrt{\alpha \beta || \hat{b} ||_{\mathcal{L}^+}}
\]
Note that in [2], $f(n) = n$; however, this will vary depending on choice of $M$, so we leave bounds in terms of $f(n)$ until the end. Independent of $f(n)$, the next lemma achieves an improvement for the undirected case by a factor of $n$.

**Lemma 2.5** Let $\mathcal{L}$ be a connected Laplacian and $S_1, \ldots, S_k$ the connected components of edges of weight $\geq \beta$. For $\alpha > 0$ and $b$ such that $\sum_{i \in S_j} b_i = 0$ for all $i$. Then
\[
|| (\mathcal{L} + \alpha M) b + \mathcal{L} \hat{b} ||_{\mathcal{L}^+} \leq 2 f(n) \frac{\sqrt{n}}{\sqrt{\beta}} || \hat{b} ||_{\mathcal{L}^+}
\]

**Proof:** As in [2], begin by decomposing $\hat{b} = \sum_j \tilde{b}_j$, with $b_j$ supported only on $S_j$. To bound $|| \tilde{b}_j ||_{\mathcal{L}^+}$, consider the electrical flow $\tilde{\gamma}_j$ on the graph’s edges, induced by demands $\tilde{b}_j$ at the nodes. Let $\tilde{\gamma}_j$ be the restriction of the flow to the internal edges of $S_j$; in other words, zero out all entries of $\gamma$ except those corresponding to internal edges of $S_j$. Then $\tilde{\gamma}_j$ no longer satisfies the demands $\tilde{b}_j$ at each node; instead, it satisfies some other set of demands $\tilde{b}_j$. (In particular, these demands are 0 for nodes not in $S_j$.) Since $\sum_{e_{ij}} \frac{\tilde{\gamma}_j}{w_{ij}} \leq \sum_{e_{ij}} \frac{\gamma_j}{w_{ij}}$, relates the energies of the two flows, then their demands satisfy the relationship $|| \tilde{b}_j ||_{\mathcal{L}^+} \leq || b_j ||_{\mathcal{L}^+}$. Also, $b_j$ and $\tilde{b}_j$ differ only on nodes on the boundary of $S_j$ (i.e. those
with an edge leaving \( S_j \), which we denote by \( \partial S_j \). So, the absolute value of their difference on one of these nodes is bounded above by the \( \ell_1 \) norm of the flows of \( y \) on its edges leaving \( S_j \). As such,

\[
||b'_j - b_j||_1 \leq \sum_{e \in \partial S_j} |y_e| \leq n \sqrt{\sum_{e \in \partial S_j} y^2_e} = n \sqrt{\frac{\beta}{\beta}} \leq n \sqrt{\frac{\beta}{\beta}} \leq n \sqrt{\beta} ||\bar{b}||_{L^+}
\]

The second inequality follows by Cauchy-Schwarz, as the number of edges is bounded by \( n^2 \), and the last by including all edges and recalling the two formulations of a flow’s energy. Also, let \( c = b'_j - b_j \), which is supported only on \( S_j \) and induces a flow \( f \) supported on edges within \( S_j \), thus only on edges of weight at least \( \beta \). Then by the Lemma 2.4

\[
||c||_{L^+} \leq \sqrt{\frac{n}{\beta}} ||c||_1
\]

Putting together the last two steps,

\[
||\bar{b}_j - \bar{b}_j||_{L^+} \leq n^{3/2} ||\bar{b}||_{L^+}
\]

By the triangle inequality:

\[
||\bar{b}_j||_{L^+} \leq ||\bar{b}_j||_{L^+} + ||\bar{b}_j - \bar{b}_j||_{L^+} \leq (1 + n^{3/2}) ||\bar{b}||_{L^+} \leq 2n^{3/2} ||\bar{b}||_{L^+}
\]

By the previous lemma,

\[
||(\mathcal{L} + \alpha M)^+ \bar{b} - \mathcal{L}^+ \bar{b}_j||_{L^+} = f(n) \sqrt{\frac{\alpha}{\beta}} ||\mathcal{L}^+ \bar{b}_j||_{L^+} = f(n) \sqrt{\frac{\alpha}{\beta}} ||\bar{b}||_{L^+} \leq 2f(n)n^{3/2} \sqrt{\frac{\alpha}{\beta}} ||\bar{b}||_{L^+}
\]

Finally,

\[
||(\mathcal{L} + \alpha M)^+ \bar{b} - \mathcal{L}^+ \bar{b}_j||_{L^+} = ||\sum_j \left( (\mathcal{L} + \alpha M)^+ \bar{b}_j - \mathcal{L}^+ \bar{b}_j \right)||_{L^+}
\]

\[
\leq \sum_j ||(\mathcal{L} + \alpha M)^{-1} \bar{b}_j - \mathcal{L}^+ \bar{b}_j||_{L^+} \leq 2f(n)n^{3/2} \sqrt{\frac{\alpha}{\beta}} ||\bar{b}||_{L^+}
\]

We now include two intermediate lemmas that require no alteration, but are still necessary for the ultimate error bound. The lemmas that follow use various methods to bound the norm of the error terms in the analog of Lemma 2.1, as stated at the start.

**Lemma 2.6**

\[
\bar{\nu} = (I - \mathcal{L} \mathcal{C}^{(i-1)T} (\mathcal{C}^{(i-1)T} \mathcal{L} \mathcal{C}^{(i-1)T})^+ \mathcal{C}^{(i-1)T}) \bar{\nu}^i
\]

**Lemma 2.7** For any symmetric positive semidefinite matrix \( M \) and arbitrary matrix \( C \), and \( \bar{v} \in \text{im}(M) \),

\[
||C\bar{v}||_{(CMC^T)^+} \leq ||\bar{v}||_M
\]

Since \( \mathcal{L}^+ \mathcal{L}^+ \) is exactly \( \mathcal{L}^+ \), the next lemma has an improved bound for the undirected case. Note that an even tighter bound in the opposite direction will later be possible, depending on the input parameters.

**Lemma 2.8**

\[
||\bar{\nu}||_{L^+} \leq 2 ||\bar{\nu}^i||_{L^+}
\]

**Proof:** By the previous lemma,

\[
||C^{(i-1)\bar{\nu}^i}||_{(\mathcal{C}^{(i-1)\mathcal{L} \mathcal{C}^{(i-1)T})^+ + (\mathcal{C}^{(i-1)T})^+ \mathcal{C}^{(i-1)T})} \leq ||\bar{\nu}^i||_{L^+}
\]

However,

\[
||C^{(i-1)\bar{\nu}^i}||_{(\mathcal{C}^{(i-1)\mathcal{L} \mathcal{C}^{(i-1)T})^+ + (\mathcal{C}^{(i-1)T})^+ \mathcal{C}^{(i-1)T})} = (C^{(i-1)\bar{\nu}^i})^T (C^{(i-1)\mathcal{L} \mathcal{C}^{(i-1)T})^+ + (\mathcal{C}^{(i-1)T})^+ \mathcal{C}^{(i-1)T})
\]

8
\[
(\mathcal{L}C^{(i-1)^T}C^{(i-1)}\mathcal{L}^{(i-1)^T}C^{(i-1)}\hat{b} + \mathcal{L}C^{(i-1)^T}C^{(i-1)}\hat{b})\mathcal{L}^+ (\mathcal{L}C^{(i-1)^T}C^{(i-1)}\mathcal{L}^{(i-1)^T}C^{(i-1)}\hat{b} + \mathcal{L}C^{(i-1)^T}C^{(i-1)}\hat{b})^T \leq \|\mathcal{L}C^{(i-1)^T}C^{(i-1)}\mathcal{L}^{(i-1)^T}C^{(i-1)}\hat{b}\|_{\mathcal{L}^+}^2 + 2\|\mathcal{L}C^{(i-1)^T}C^{(i-1)}\hat{b}\|_{\mathcal{L}^+}
\]

By Lemma 2.6
\[
\hat{b} = (I - \varphi_{i-1})\hat{b}^i
\]
Then by the triangle inequality
\[
||\hat{b}^i||_{\mathcal{L}^+} \leq ||\hat{b}^*||_{\mathcal{L}^+} + ||\varphi_{i-1}\hat{b}^i||_{\mathcal{L}^+} \leq 2||\hat{b}^*||_{\mathcal{L}^+}
\]

**Lemma 2.9**
\[
||\hat{c}^*||_{\mathcal{L}} \leq \left(\frac{2 + 4f(n)n^{5/2}\sqrt{hr}}{r}\right)||\hat{b}^*||_{\mathcal{L}}
\]

**Proof:** By the previous two lemmas,
\[
||C^\dagger\hat{b}||_{(C^\dagger\mathcal{L}C^\dagger)^\top} \leq ||\hat{b}||_{\mathcal{L}^+} \leq 2||\hat{b}^*||_{\mathcal{L}^+}
\]
Define four new variables:
\[
q^i \triangleq C^\dagger (C^\dagger\mathcal{L}C^\dagger)^\top C^\dagger \hat{b}^i
\]
\[
\hat{c}^i \triangleq \hat{c}^i - \hat{q}^i, \text{ where } \hat{c}^i \text{ is as defined in the algorithm}
\]
\[
\hat{f}^i \triangleq \text{Proj}^i (\mathcal{L}\hat{c}^i)
\]
\[
\hat{b}^i \triangleq \sum_j f_j^i
\]
Recall from the earlier section that if we had \(\hat{c}^i = \hat{q}^i\) at each iteration of the algorithm, then the result returned would be exact and \(\hat{c}^i = 0\). So, \(\hat{c}^i\) and \(\hat{f}^i\) will be our error terms for the approximate case. Note that \(\hat{c}^i = C^\dagger (\hat{c}^i - \hat{q}^i)\) implies
\[
||\hat{c}^i||_{\mathcal{L}} = ||C^\dagger (\hat{c}^i - \hat{q}^i)||_{\mathcal{L}} = ||\hat{c}^i - \hat{q}^i||_{(C^\dagger\mathcal{L}C^\dagger)^\top} \leq ||\hat{c}^i - \hat{q}^i||_{(C^\dagger\mathcal{L}C^\dagger)^\top} + ||\hat{c}^i - \hat{q}^i||_{(C^\dagger\mathcal{L}C^\dagger)^\top}
\]
This breakdown is useful, because it captures the two separate sources of error that arise in the algorithm. The first term describes the error from using an approximate, rather than exact solver. Using that \(M \geq 0\) and the fact at the start of the proof, we have
\[
||\hat{c}^i - \hat{q}^i||_{(C^\dagger\mathcal{L}C^\dagger)^\top} \leq ||\hat{c}^i - \hat{q}^i||_{(C^\dagger\mathcal{L}C^\dagger)^\top + \frac{\alpha}{\beta} M}
\]
\[
\leq \frac{1}{r}||\hat{c}^i - \hat{q}^i||_{(C^\dagger\mathcal{L}C^\dagger)^\top + \frac{\alpha}{\beta} M} = \frac{1}{r}||C^\dagger \hat{b}^i||_{(C^\dagger\mathcal{L}C^\dagger)^\top + \frac{\alpha}{\beta} M}^2
\]
\[
\leq \frac{1}{r}||C^\dagger \hat{b}^i||_{(C^\dagger\mathcal{L}C^\dagger)^\top}^2 \leq 2 \frac{2}{r}||\hat{b}^*||_{\mathcal{L}^+}
\]
The second term describes the error from adding a multiple of \(M\) to the contracted system. Let \(\hat{L} = C^\dagger\mathcal{L}C^\dagger, \text{ and } \hat{b} = C^\dagger \hat{b}^i\). Since \(\hat{b}^i\) has been projected onto the kernel of \(C^{(i-1)}, \text{ its entries over any connected component of edges of weight } \geq \frac{w_i}{u} \text{ sum to 0. This property additionally holds in the contracted case: the entries of } C^\dagger \hat{b}^i \text{ over any connected component of } C^\dagger\mathcal{L}C^\dagger \text{ of edges with weight } \geq \frac{w_i}{u} \text{ still sum to 0. Then by applying Lemma 2.5 with } \alpha = \frac{\beta}{\frac{\beta}{\beta}} \text{ and } \beta = \frac{\alpha}{\alpha},
\]
\[
||(\hat{L} + \frac{\alpha}{\beta} M)^\top \hat{b} - \hat{L}^\top \hat{b}||_{\mathcal{L}} \leq 2f(n)n^{5/2} \sqrt{\frac{\alpha}{\beta}} \frac{\beta}{\beta} ||\hat{b}||_{\mathcal{L}^+} \leq 4f(n)n^{5/2} \sqrt{\frac{\alpha}{\beta} ||\hat{b}^*||_{\mathcal{L}^+}}
\]
The final bound is then
\[
||\hat{c}^i||_{\mathcal{L}} \leq \left(\frac{2}{r} + 4f(n)n^{5/2} \sqrt{\frac{\alpha}{\beta}} \frac{\beta}{\beta} \right)||\hat{b}^*||_{\mathcal{L}^+}
\]

\[
\]
Lemma 2.10 \(|\|f^\top||_{L^+} \leq 5n^{3/2}\|e^3\|_L^+|\)

**Proof:** First, note that, in the undirected case, \(|\|L^2||_{L^+} = \sqrt{(L^2)^T L^+ (L^2)} = \sqrt{e^2^T L e^2} = \|e^2\|_L^+|\)

Recall that \(f^\top = \text{Proj}^f(L^2)\). Let \(g^\top\) be the electrical flow induced by demands \(L^2\), and let \(\tilde{g}^\top\) be its restriction to edges of weight \(\geq w^i\), corresponding to modified demand vector \(\tilde{b}^i\). Since \(\text{Proj}^f\) projects onto the kernel of \(C^i\), this amounts to shifting the weights on each component by a constant such that they sum to 0, and is a linear operation. Thus \(\text{Proj}^f(L^2) = \text{Proj}^f(\tilde{b}^i) + \text{Proj}^f(L^2 - \tilde{b}^i)\). However, \(\tilde{b}^i\) is the set of demands for a flow \(g^\top\) only over edges of weight \(\geq w^i\), so its entries already sum to 0 over components of weight over \(w^i\) and thus is invariant under projection: \(\text{Proj}^f(\tilde{b}^i) = \tilde{b}^i\). As in the previous flow analysis, \(g^\top\) being a restriction of \(g^\top\) implies it has lower energy:

\[||\tilde{g}^\top||_{L^+} \leq ||L^2||_{L^+} = ||e^3||_L^+\]

Similarly, observe that \(L^2 - \tilde{b}^i\) is the residual of the flow \(g^\top - \tilde{g}^\top\), which is the restriction of \(g^\top\) to edges of weight \(< w^i\). This implies that once again its energy is less than the energy of flow \(\tilde{g}^\top\):

\[||L^2 - \tilde{b}^i||_{L^+} \leq ||L^2||_{L^+}\]

By Lemma 2.3, we have

\[||L^2 - \tilde{b}^i||_1 \leq 2n\sqrt{w^i}||L^2||_{L^+}\]

Next, observe that by definition of \(\text{Proj}^f\), \(\text{Proj}^f(L^2 - \tilde{b}^i)\) is nonzero only on nodes connected by edges of weight at least \(w^i\). Then by Lemma 2.4,

\[||\text{Proj}^f(L^2 - \tilde{b}^i)||_{L^+} \leq \sqrt{\frac{n}{w^i}}||\text{Proj}^f(L^2 - \tilde{b}^i)||_{L^+}\]

For node \(j\) in component \(C_k\), as defined by \(C^i\), we have

\[\text{Proj}^f(\bar{x})_j = \bar{x}_j - \frac{1}{|C_k|} \sum_{v \in C_k} \bar{x}_v\]

Since \(1 \leq |C_k| \leq n\), \(||\text{Proj}^f(\bar{x})_C||_1 \leq 2||\bar{x}_C||_1\) by the triangle inequality. Thus

\[||\text{Proj}^f(\bar{x})||_1 \leq 2||\bar{x}||_1 \rightarrow ||\text{Proj}^f(L^2 - \tilde{b}^i)||_1 \leq 2||L^2 - \tilde{b}^i||_1\]

Putting these steps together:

\[||\text{Proj}^f(L^2 - \tilde{b}^i)||_{L^+} \leq 2\sqrt{\frac{n}{w^i}}||L^2 - \tilde{b}^i||_1 \leq 4n^{3/2}\|e^3\|_L^+\]

\[||\tilde{f}^\top||_{L^+} = ||\text{Proj}^f(L^2 - \tilde{b}^i + \tilde{b}^i)||_{L^+} \leq ||\text{Proj}^f(L^2 - \tilde{b}^i)||_{L^+} + ||\tilde{b}^i||_{L^+} \leq 4n^{3/2}\|e^3\|_L + ||e^3||_L\]

This gives the final result:

\[||\tilde{f}^\top||_{L^+} \leq 5n^{3/2}\|e^3\|_L^+\]

\[\blacksquare\]

**Lemma 2.11** For certain families of functions \(f(n)\) and set of values for \(r\), a bound is possible of the form

\[||\tilde{b}^i||_{L^+} \leq K_{r,f}(n)||\tilde{b}||_{L^+},\text{ where }K_{r,f}(n) \geq 1\]

For \(r\) sufficiently large relative to \(n\), \(\lim_{n \to \infty} K_{r,f}(n) = 1\)
Proof: Note that by combining the previous two lemmas,

$$||\vec{f}||_\mathcal{L}^+ \leq \frac{10n^{3/2} + 20 f(n)n^4 \sqrt{\text{hur}}}{r} ||\vec{b}^i||_\mathcal{L}$$

Also, observe that the number of iterations is bounded by $n^2$, an upper bound on the edges of the graph. At this point, it is necessary to make assumptions about $r$ and $f(n)$ (and less crucially, parameters $h$ and $a$); different bounds are possible depending on these selections. Observe that for all settings of parameters, by applying the triangle inequality to the definition of $\vec{b}^i$

$$||\vec{b}^i||_\mathcal{L}^+ \leq ||b||_\mathcal{L}^+ + \sum_{j < i} ||\vec{f}^j||_\mathcal{L}^+ \leq ||b||_\mathcal{L}^+ + \sum_{j < i} ||\vec{f}^j||_\mathcal{L}^+$$

$$\leq ||b||_\mathcal{L}^+ + \sum_{j < i} \frac{10n^{3/2} + 20 f(n)n^4 \sqrt{\text{hur}}}{r} ||\vec{b}^j||_\mathcal{L} \leq ||b||_\mathcal{L}^+ + \frac{10n^{7/2} + 20 f(n)n^6 \sqrt{\text{hur}}}{r} ||\vec{b}^j||_\mathcal{L}$$

Setting the parameters to the values specified, we have

$$\frac{10n^{7/2} + 20 n^7 \sqrt{2(1000n^{10})^2}}{(1000n^{10})^2} \left( 1 + \frac{1}{n^2} \right) \leq \frac{1}{n^2}$$

for sufficiently large $n$; naturally, lower exponents than 2 hold for small $n$.

$$r = (1000n^{10})^2, \ f(n) = n, \ h = 1, \ u = 2 \rightarrow K_{r,f} = \left( 1 + \frac{1}{n^2} \right)$$

This is precisely the case used in [2]; however, we will see that such a severe value for $r$ is not necessary in later cases. For now, we proceed by induction. For $i = 0$, $\vec{b}^i = \vec{b}$, so the result holds. Now, assuming the result holds, $\vec{b}^i = b - \sum_{j < i} \vec{f}^j$ implies

$$||b||_\mathcal{L}^+ + \frac{10n^{7/2} + 20 f(n)n^6 \sqrt{\text{hur}}}{r} ||\vec{b}^j||_\mathcal{L} \leq ||b||_\mathcal{L}^+ + \frac{10n^{7/2} + 20 f(n)n^6 \sqrt{\text{hur}}}{r} \left( 1 + \frac{1}{n^2} \right) ||b||_\mathcal{L}^+$$

for sufficiently large $n$; naturally, lower exponents than 2 hold for small $n$.

$$r = n^{15}, \ f(n) = n, \ h = 1, \ u = 2 \rightarrow K_{r,f} = \left( 1 + \frac{4}{n^2} \right)$$

This bound holds by the same reasoning as the previous. Although this value for $r$ is still unfeasible, it does demonstrate that we can reduce the size of $r$ in the undirected case and still achieve an equally good error bound.

$$r = (1000n^{10})^2, \ f(n) = \frac{2}{\sqrt{n}} n^3 \rightarrow K_{r,f} = \left( 1 + \frac{1}{n^2} \right)$$

In the next section, we will see that this $f(n)$ corresponds to rounding low-weight edges. It arises from the same reasoning as above. ■

Theorem 2.12 The algorithm achieves the fully general error bound

$$||\vec{x} - \mathcal{L}^+ b||_\mathcal{L} \leq K_{r,f}(r) \left( 2n^2 + 10n^{7/2} + 4 f(n) \sqrt{\text{hur}}(n^{9/2} + 5n^6) \right) ||\mathcal{L}^+ \vec{b}||_\mathcal{L}$$

Proof: As stated previously and proved by induction in [2],

$$\vec{x}^i = \mathcal{L}^+ \vec{b} = \sum_{j < i} \vec{e}^j + \mathcal{L}^+ \vec{b}^i$$

As described in the section focusing on exact solves, ultimately $\vec{b} = 0$. For this $i$, rearranging yields

$$\vec{x} = \sum_{j < i} \vec{e}^j + \mathcal{L}^+ \left( b - \sum_{j < i} \vec{f}^j \right)$$

$$\vec{x} - \mathcal{L}^+ b = \sum_{j < i} \vec{e}^j - \mathcal{L}^+ \sum_{j < i} \vec{f}^j$$
\[ \| \mathbf{x} - \mathbf{L}^+ \mathbf{b} \|_\mathcal{L} \leq \sum_{j<i} \| \mathbf{e}_j \|_\mathcal{L} + \sum_{j<i} \| \mathbf{L}^+ \mathbf{f}_j \|_\mathcal{L} = \sum_{j<i} \| \mathbf{e}_j \|_\mathcal{L} + \sum_{j<i} \| \mathbf{f}_j \|_\mathcal{L}^+ \]

\[ \leq \sum_{j<i} \left( \frac{2 + 4f(n)n^{5/2}\sqrt{\text{hur}}}{r} \right) \| \mathbf{e}_j \|_\mathcal{L} + \sum_{j<i} \left( \frac{10n^{3/2} + 20f(n)n^4\sqrt{\text{hur}}}{r} \right) \| \mathbf{f}_j \|_\mathcal{L}^+ \]

\[ \leq n^2 \mathcal{K}_{r,f}(n) \left( \frac{2 + 10n^{3/2} + 4f(n)\sqrt{\text{hur}}(n^{5/2} + 5n^4)}{r} \right) \| \mathbf{b} \|_\mathcal{L}^+ \]

\[ = \mathcal{K}_{r,f}(n) \left( \frac{2n^2 + 10n^{7/2} + 4f(n)\sqrt{\text{hur}}(n^{9/2} + 5n^6)}{r} \right) \| \mathbf{L}^+ \mathbf{b} \|_\mathcal{L} \]

\[ \square \]

2.2.2 Selected Cases

Although these both use the same unfeasible value for \( r \) as the original paper, they achieve an error bound that improves for large \( n \), rather than remaining constant.

Original

Although this is a messier final error bound than that of [2], it is both more general, and tighter. For instance, setting \( r = (1000n^{10})^2, f(n) = n, h = 1, \) and \( u = 2 \) mimics the original algorithm of [2]. In this case, we derived \( \mathcal{K}_{r,f}(n) = (1 + \frac{1}{\sqrt{n}}) \), which achieves a final bound

\[ \| \mathbf{x} - \mathbf{L}^+ \mathbf{b} \|_\mathcal{L} \leq O(\frac{1}{n^{1/3}}) \| \mathbf{L}^+ \mathbf{b} \|_\mathcal{L} \]

Rounding

We previously had \( r = (1000n^{10})^2, f(n) = \frac{2}{\sqrt{n}} n^3 \rightarrow \mathcal{K}_{r,f} = (1 + \frac{1}{\sqrt{n}}) \), which results in a bound

\[ \| \mathbf{x} - \mathbf{L}^+ \mathbf{b} \|_\mathcal{L} \leq O(\frac{1}{n}) \| \mathbf{L}^+ \mathbf{b} \|_\mathcal{L} \]

In the next section, we will derive this value of \( f(n) \) for the case of rounding low-weight edges, motivating this case.

3 Modifications

This analysis framework suggests that other positive semidefinite matrices \( \mathbf{M} \) may work equally well, or even better, in practice. For this reason, we repeat the relevant portion of the analysis for a plausible alternative to \( \mathbf{M} = \mathbf{I} \): rounding up all low-weight edges, such that the resultant set of edges are do not vary too much.

3.1 Rounding Edges

Lemma 3.1 Let \( \mathcal{L} \) be a connected Laplacian, and \( S \) a subset of the vertices such that any two vertices in \( S \) are connected by a path containing edges of weight at least \( \beta \). Let \( \mathbf{b} \) be a vector in the image of \( \mathcal{L} \) which is supported on \( S \). Finally, let \( \alpha > 0 \) and let \( \mathcal{R}(\alpha(\mathcal{L})) \) denote the Laplacian of the same graph in which all edge weights below \( \alpha \) have been increased by \( \alpha \). Then

\[ \left\| \mathcal{R}_\alpha(\mathcal{L})^+ - \mathbf{L}^+ \mathbf{b} \right\|_\mathcal{L} \leq 2\sqrt{\frac{\alpha}{\pi \beta}} n^3 \| \mathbf{x} \|_\mathcal{L}^2 \]

Proof: Observe that \( \mathcal{R}_\alpha(\mathcal{L}) \) can be written \( \mathcal{L} + \alpha \mathbf{M} \), where \( \mathbf{M} \) is the Laplacian of a weighted subgraph of edges of weight \( \leq 1 \). As such, \( \mathbf{M} \geq 0 \). We follow the structure of the proof of Lemma F.5 in [2]. Letting \( \mathbf{x} = \mathbf{L}^+ \mathbf{b} \) and \( \mathbf{y} = (\mathcal{L} + \alpha \mathbf{M})^+ \mathbf{b} \), we have

\[ \mathbf{y} - \mathbf{x} = \alpha(\mathcal{L} + \alpha \mathbf{M})^+ (\mathbf{M} \mathbf{x}) \]
 Combining the last two bounds, we have
\[
\|\vec{y} - \vec{x}\|^2_2 = \alpha^2(Mx)^T(L + \alpha M)^+L(L + \alpha M)^+(Mx)
\]

\(M\) is positive semidefinite and \(\alpha > 0\), \(L \leq L + \alpha M\), so
\[
\|\vec{y} - \vec{x}\|^2_2 \leq \alpha^2(Mx)^T(L + \alpha M)^+(L + \alpha M)(L + \alpha M)^+(Mx) = \alpha^2(Mx)^T(L + \alpha M)^+(Mx)
\]

Since \(L + \alpha M\) is the Laplacian \(\tilde{L}\) of a connected graph of minimum edge weight \(\alpha\), we have the following lower bound for the algebraic connectivity (derivation shown in the introduction):
\[
\lambda_2 \geq \frac{\pi^2 \alpha}{n^2}
\]
\[
\|\tilde{L}^+\| \leq \frac{1}{\lambda_2(\tilde{L})} \leq \frac{n^2}{\pi^2 \alpha}
\]
\[
\tilde{L}^+ \leq \frac{n^2}{\pi^2 \alpha} \mathbf{I}
\]

We then have
\[
\|\vec{y} - \vec{x}\|^2_2 \leq \alpha^2\left( \frac{n^2}{\pi^2 \alpha} \right)(Mx)^T(Mx) = \frac{\alpha n^2}{\pi^2} (x^T M^T M x) = \frac{\alpha n^2}{\pi^2} ||x||^2_{MT,M}
\]

To bound the largest eigenvalue of \(A \triangleq M^T M\), recall the Gershgorin circle theorem, which states the following fact about every eigenvalue \(\lambda_i\) of a square matrix \(A\):
\[
\lambda_i \in B(a_{ii}, \sum_{j \neq i} |a_{ij}|)
\]

For \(i \neq j\), using that \(M\) is a Laplacian of weights \(\leq 1\), we have
\[
|A_{ii}| \leq (n-1) + (n-1)^2 = n(n-1)
\]
\[
|A_{ij}| \leq (n-2) + 2(n-1) = 3n - 4
\]

Thus
\[
|\lambda_i| \leq n(n-1) + (n-1)(3n - 4) = 4(n-1)^2 \leq 4n^2 \forall i
\]
\[
M^T M \leq 4n^2 I
\]

For convenience, let \(f(n) = 4n^2\). Then
\[
||x||^2_{MT,M} \leq f(n)||x||^2_2 = f(n)x^T x
\]
\[
||\vec{y} - \vec{x}\|^2_2 \leq \frac{\alpha n^2}{\pi^2} f(n)x^T x \leq \frac{\alpha n^3}{\pi^2} f(n)||x||^2_\infty
\]

By Lemma 2.2, both the maximum and minimum entries of \(\vec{x}\) occur within \(S\). Furthermore, by assumption they are connected by a path of at most \(n\) vertices, each of weight at least \(\beta\). Then
\[
||\vec{x}||^2_2 = \sum_{(i,j) \in E} w_{ij}(x_i - x_j)^2 \geq \sum_{(i,j) \in E} \beta(x_i - x_j)^2 \geq \frac{\beta}{n} ||\vec{x}||^2_\infty
\]

Denoting by \(P\) the set of edges of the path from the minimum to maximum entry of \(\vec{x}\), the last inequality follows by the Cauchy-Schwarz inequality:
\[
\sum_{(i,j) \in E} (x_i - x_j)^2 \geq \sum_{(i,j) \in P} (x_i - x_j)^2 \geq \frac{1}{n} \left( \sum_{(i,j) \in P} x_i - x_j \right)^2 \geq \frac{||\vec{x}||_\infty}{n}
\]

Combining the last two bounds, we have
\[
||\vec{y} - \vec{x}||^2_2 \leq \frac{\alpha}{\pi^2 \beta} n^4 f(n)||\vec{x}||^2_2 \leq \frac{4\alpha}{\pi^2 \beta} n^6 ||\vec{x}||^2_2
\]
\[
||\vec{y} - \vec{x}||_\infty \leq 2 \sqrt{\frac{\alpha}{\pi \beta} n^3 ||\vec{x}||^2_2}
\]

\[\blacksquare\]
Theorem 3.2 As proven earlier, setting $r = (1000n^{10})^2$, $h = 1$, $u = 2$, and $M$ such that it rounds up edges above $\frac{u^2}{h}$ at each iteration, achieves error bound

$$||\tilde{x} - \mathcal{L}^+ b||_\mathcal{L} \leq O\left(\frac{1}{n}\right)||\mathcal{L}^+ \tilde{b}||_\mathcal{L}$$

3.2 Dropping Edges

It is natural to also consider simply deleting less significant edges, as another way of lowering the condition number. For completeness, here we present a similar lemma for the case of dropping low-weight edges. Although the offset to $\mathcal{L}$ is no longer positive semi-definite, our original upper bound on the finite condition number suggests that it may still result in a better-conditioned system.

Lemma 3.3 Let $\mathcal{L}$ be a connected Laplacian, and $S$ a subset of the vertices such that any two vertices in $S$ are connected by a path containing edges of weight at least $\alpha$. Let $\tilde{b}$ be a vector in the image of $\mathcal{L}$ which is supported on $S$. Finally, let $\alpha > 0$ and let $S_\alpha(\mathcal{L})$ denote the Laplacian of the same graph containing only edges of weight below $\alpha$. Then

$$||\mathcal{L} - S_\alpha(\mathcal{L})^+ - \mathcal{L}^+ \tilde{b}||_\mathcal{L} \leq \frac{2n^{3/2} \sqrt{2n}}{\pi \beta \sqrt{\alpha}} ||\tilde{x}||_\mathcal{L}$$

Proof: Observe that $\mathcal{L} - S_\alpha(\mathcal{L})$ is the Laplacian of the original graph, with all edges below $\alpha$ removed. As such, $0 \leq S_\alpha(\mathcal{L}) \leq \mathcal{L}$.

We again follow the structure of Lemma F.5 in [2]. Letting $\tilde{x} = \mathcal{L}^+ \tilde{b}$ and $\tilde{y} = (\mathcal{L} - S_\alpha(\mathcal{L}))^+ \tilde{b}$, we have

$$\tilde{y} - \tilde{x} = (\mathcal{L} - S_\alpha(\mathcal{L}))^+(S_\alpha(\mathcal{L}) \tilde{x})$$

$$||\tilde{y} - \tilde{x}||_2^2 = ((\mathcal{L} - S_\alpha(\mathcal{L}))^+(S_\alpha(\mathcal{L}) \tilde{x}))^T \mathcal{L}((\mathcal{L} - S_\alpha(\mathcal{L}))^+(S_\alpha(\mathcal{L}) \tilde{x}))$$

Let $u = 2d_{\max}$ be an upper bound for the largest eigenvalue of $S_\alpha(\mathcal{L})$, and let $\mathcal{V} = S_\alpha(\mathcal{L}) \tilde{x}$. Then

$$\mathcal{L} \leq \mathcal{L} - S_\alpha(\mathcal{L}) + uI$$

which implies

$$||\tilde{y} - \tilde{x}||_2^2 = \mathcal{V}^T(\mathcal{L} - S_\alpha(\mathcal{L}))^+\mathcal{L}(\mathcal{L} - S_\alpha(\mathcal{L}))^+\mathcal{V}^T$$

$$\leq \mathcal{V}^T(\mathcal{L} - S_\alpha(\mathcal{L}))^+(\mathcal{L} - S_\alpha(\mathcal{L}) + uI)(\mathcal{L} - S_\alpha(\mathcal{L}))^+\mathcal{V}^T \mathcal{V}$$

$$= u\mathcal{V}^T(\mathcal{L} - S_\alpha(\mathcal{L}))^+(\mathcal{L} - S_\alpha(\mathcal{L}))^+\mathcal{V}$$

As in the previous proof, $\mathcal{L} - S_\alpha(\mathcal{L})$ is the (possibly disconnected) Laplacian of a graph with minimum edge weight $\alpha$. All of its eigenvalues are upper-bounded by the largest eigenvalue of the Laplacians for each connected component. Letting $n'$ be the number of nodes for a specific component, that component’s inverse Laplacian eigenvalues can be upper bounded by $\frac{n'^2}{\pi \alpha^2}$. The eigenvalues of $(\mathcal{L} - S_\alpha(\mathcal{L}))^+(\mathcal{L} - S_\alpha(\mathcal{L}))^+$ are then upper bounded by $B \equiv \left(\frac{n'^2}{\pi \alpha^2}\right)^2 = \frac{n^4}{\pi^2 \alpha^2}$. The eigenvalues of $(\mathcal{L} - S_\alpha(\mathcal{L}))^+(\mathcal{L} - S_\alpha(\mathcal{L}))^+$ are then upper bounded by $B \equiv \left(\frac{n'^2}{\pi \alpha^2}\right)^2 = \frac{n^4}{\pi^2 \alpha^2}$

$$||\tilde{y} - \tilde{x}||_2^2 \leq uB\mathcal{V}^T \mathcal{V} = uB(S_\alpha(\mathcal{L}) \tilde{x})^T (S_\alpha(\mathcal{L}) \tilde{x})$$

$$= uB\tilde{x}^T S_\alpha(\mathcal{L})^T S_\alpha(\mathcal{L}) \tilde{x}$$

Let $C$ be an upper bound for the eigenvalues of $S_\alpha(\mathcal{L})^T S_\alpha(\mathcal{L})$. Since, in the most dramatic case, every edge is of weight below $\alpha$, we have the same upper bound from the previous proof for $M^T M$: $C = 4n^2$. Then

$$||\tilde{y} - \tilde{x}||_2^2 \leq uBC\tilde{x}^T \tilde{x} \leq nuBC||\tilde{x}||_\infty^2$$

14
As in the previous lemma, \( ||\vec{x}||_\infty^2 \leq \frac{n^2}{\beta} ||\vec{x}||_L^2 \). Combining these results, we have

\[
||\vec{y} - \vec{x}||_L^2 \leq \frac{n^2 u B C}{\beta} ||\vec{x}||_\infty^2 \leq \frac{4n^8 u}{\pi^2 \alpha \beta^2} ||\vec{x}||_L^2 \leq \frac{4n^9 u}{\pi^2 \alpha \beta^2} ||\vec{x}||_L^2
\]

Clearly, the bound obtained by similar methods of analysis as the case of rounding is significantly worse for dropping edges (\( n^{3/2} \) instead of \( n^3 \)). This indicates that shifting will outperform it, which will match the practical result. However, it may be that a combination of rounding and dropping work better for certain cases, in which case we may follow the same analysis framework.

### 3.3 Challenges, in Theory

Based on this analysis, there are several immediate problems with implementing this algorithm directly.

**3.3.1 \( r \) value**

First, \( r = (1000n^{10})^{2} \) is not feasible for any applications; the amount by which the contracted system is offset by \( M \) will be negligible. In addition, the proofs shown assume that the blackbox solver of well-conditioned systems provides a \( 1/r \) approximation. For any value of \( r \) that is too high a power of \( n \), or for applications where \( n \) is decently sized, this is no longer a safe assumption.

**3.3.2 Condition Number**

The algorithm is designed to call the outside solver only on systems that are at most polynomially ill-conditioned. However, a polynomial in \( n \) still may be too high for an existing solver to handle. Parameter adjustment allows us to contract high-weight edges more aggressively, but contracted components can still have up to \( n^2 \) edges between them; as such, even the contracted Laplacian may have high edge weights. Along the same vein, for high values of \( r \), the contracted system may not be shifted enough by \( M \) to have tractable condition number. (On the other hand, too small an \( r \) will accumulate too much error to produce an accurate solution.)

**3.3.3 Iterations**

In addition, \( u \) (the update factor for the weight cutoff, \( w^t \)) must be higher than 2 for many distributions of edge weights; otherwise, the number of iterations will be too high, and since each iteration solves a smaller Laplacian system, the algorithm will be too slow to be practical. The magnitude of weight updates also will affect the final accuracy, both because error is accumulated at each iteration, and because higher values for \( w^t \) mean changing the actual contracted system at each iteration more dramatically.

### 4 Empirical Results

Despite the projected challenges above, it was important to see how the algorithm performed in practice; after all, condition numbers are often a purely practical consideration. As such, the algorithm was implemented in Julia, using the `Laplacians.jl` package for the blackbox solvers (`approxCholLap` and `approxCholSDDM`, as appropriate), to compute connected components, and to compute the maximum spanning tree. All computations were done using sparse matrices. In general, practical values for \( r \) and \( u \) work well for Laplacian systems with arbitrarily high condition number, with accuracy at least as good as that of the base solver. Given the crudeness of the error bounds, it is both surprising and encouraging that it works with high accuracy in all tested situations.
In several cases with high condition number, the existing solver worked surprisingly well. However, the contraction solver could also handle to high accuracy very poorly-conditioned cases on which the existing solver failed, demonstrating its independent value. As suggested in [2], it is also possible to further improve performance in these cases by combining with preconditioned Richardson iteration, although this does contribute significantly to the runtime. Further development is needed before it will be a fully automated, practical tool for large systems, due to the challenges of parameter selection, but these preliminary experimental results are promising for future applications.

4.1 Implementation Notes

4.1.1 Estimating the Condition Number

For graphs on many nodes, it is not feasible to compute the condition number directly. (This is not strictly necessary for the algorithm itself, but is useful in testing \( r \) values, and ensuring that the contracted systems are indeed well-conditioned). Instead, we can estimate \( \frac{\lambda_2}{\lambda_n} \) by

\[
\frac{d_{\text{max}}}{w}
\]

where \( w \) is the minimum edge weight in a minimum spanning tree and \( d_{\text{max}} \) is the maximum degree. When \( \alpha I \) is added to the graph’s Laplacian, the estimate becomes

\[
\frac{d_{\text{max}}}{w + \alpha}
\]

As discussed earlier, this is a reasonable approximation since \( \lambda_n \leq 2d_{\text{max}} \), and \( w \) approximates \( \lambda_2 \).

4.1.2 Error

Examples were generated by choosing \( \vec{x}^* \) and then setting \( \vec{b} = L\vec{x}^* \). As such, the error was measured as:

\[
\frac{||\vec{x} - \vec{x}^*||_L}{||\vec{x}^*||_L}
\]

4.1.3 Precision

When testing graphs on many nodes, or with very high edge-weights, it is necessary to test the result in high-precision. For instance, with edge weights varying from 1 to \( 10^{30} \) and about 1000 nodes, using the ordinary Float64 data type in Julia results in false results, e.g. \( \vec{f}^T L \vec{f} \neq 0 \). As such, Julia’s BigFloat data type was used to compute the final error.

4.2 Performance

We show the algorithm’s performance on different graph types, varying the values for \( r \) and \( u \). For each type of graph, a vector \( \vec{x}^* \) was chosen randomly, and \( L\vec{x}^* = \vec{b} \) input to the function. Note that in all figures, \( \log(x) = \log_{10}(x) \). In all cases, it can achieve error at least as good as the base solver (and in the ill-conditioned cases, significantly better). However, the optimal parameters vary widely. We measure optimality by the error, as well as by the maximum condition number of all Laplacians solved throughout the algorithm.

4.2.1 Weighted Erdos-Renyi Random Graph

For this case, the adjacency matrix was generated using ErdosRenyi, on 1500 nodes and with around 450,000 edges.

As in many of the following cases, we see the expected tradeoff for \( r \), in which higher values achieve better error but requires solving systems with higher condition number. Similarly, higher update values \( u \) achieve worse error, but better condition numbers.
4.2.2 Weighted Chimera Graph

For this case, the adjacency matrix was generated using `wtedChimera` on 1500 nodes.

4.2.3 Ill-Conditioned Graph

To generate extremely ill-conditioned graphs, we began with an unweighted base graph and assigned a weight $y10^x$ to each node, where $y \in [0, 1]$ and $x \in \mathbb{N}$ is chosen at random from $\{a, ..., b\}$. In the first scheme (“maximum”), each edge weight is set to the maximum of its end nodes’ weights. In the second scheme (“product”), each edge weight is set to the product of its end nodes’ weights.
Figure 4: Maximum condition number as function of $u$ for weighted Chimera graph, with $r = 1500$

**Erdos-Renyi: Maximum** In this case, the base adjacency graph was an Erdos-Renyi random graph, with the maximum weight scheme described above and $a = -2$, $b = 60$.

Figure 5: Varying Parameter $r$ for ill-conditioned (maximum) Erdos-Renyi graph, $u = 2$

**Erdos-Renyi: Product** In this case, the base adjacency graph was again an Erdos-Renyi random graph, with the product weight scheme described above and $a = -2$, $b = 30$.

Figure 6: Maximum condition number as function of $u$ for ill-conditioned (maximum) Erdos-Renyi graph, with $r = 1500$
Figure 7: Varying Parameter $r$ for ill-conditioned (product) Erdos-Renyi graph, $u = 10^{22}$

Figure 8: Maximum condition number as function of $u$ for ill-conditioned (product) Erdos-Renyi graph, with $r = 10^5$

**Grid Graph: Maximum** For this example, a 2-dimensional grid graph on 1600 nodes was used, with the maximum weight scheme described above and $a = -2$, $b = 30$.

Figure 9: Varying Parameter $r$ for ill-conditioned (product) 2D grid graph, $u = 10^{15}$
4.2.4 Preconditioned Richardson Iteration

Cohen et al. [2] achieve a constant error guarantee, but suggest using standard preconditioned Richardson iteration to improve the approximate guarantee. Although with appropriate choice of parameters the error achieved is far better than $\frac{1}{2}$, we implemented this method as fprecon and it does improve the solution returned by contraction, as expected. This may be useful, since instead of trying to determine the precise optimal parameters for a graph, we can simply improve an approximate solution from a reasonable choice of parameters.

4.2.5 Rounding and Dropping

In practice, rounding and dropping edges achieve comparable error to adding self-loops (i.e. adding $I$), but do not have the same effect of lowering the condition number. Going forward, it seems that $M = I$ is the best approach.

The following were tested using a weighted Erdos-Renyi random graph on 1500 nodes, again with around 450,000 edges. The weight update value was fixed at $u = 2$.

5 Future Work

5.1 Theory

The methods used to obtain a final error bound were still quite messy, even in the undirected case. A simplified proof, possibly with better error bounds, is desirable, and it would be interesting to see if a complete different approach could yield a simpler proof. Such a development might also make potential improvements or related algorithms more obvious. One such method for a proof might be
to expand the discussion of graph flows. For instance, one of the lemmas shows that the energy of
the electrical flow induced always decreases with contraction; in this sense, the algorithm is solving
a “high-energy” system by solving a series of “low-energy” systems. Although this does not provide
much insight about the algorithm’s correctness on its own, expanding this intuition to quantitatively
account for contracted specifically high-weight edges could be one fruitful avenue for exploration.

5.2 Practice

Alternative matrices for $M$ may be interesting to try, to achieve improved accuracy or runtime. For
example, one could set $M$ to be a mixture of the options discussed: dropping some extremely low-
weight edges, rounding slightly higher-weight edges, and adding a small multiple of $I$. (Of course,
this introduces additional parameters.) Interestingly, range of edge weights, and occasionally condi-
tion number, did not always seem to be the absolute best predictor of accuracy for the existing
solver: it sometimes had quite high error in a well-conditioned intermediate solve. Is this an issue of
precision, is it specific to the solver used, or are there other characteristics of Laplacians that make
them harder or easier to solve? Questions such as these will be important in allowing the algorithm
to handle different varieties of graphs.

However, the biggest challenge in practice seems to be parameter selection. As shown above, the
choice of parameters determines whether or not the algorithm succeeds, and (experimentally) optimal
parameters vary widely from case to case. It would be useful to have a more structured approach
to finding the best parameters for a given family of graphs, possibly based on weight or degree dis-
tribution. For instance, the two schemes for generating matrices with extreme edge weights above
worked very differently — the “maximum” and “product” worked with very different parameters,
even with approximately the same maximum and minimum edge weights and the same (unweighted)
base graph. The two graphs have very different weight distributions, however:

![Figure 12: Histogram of edge weights in two extreme weight generation schemes](image)

(a) Node weights set to $y10^x$, where $x$ was an integer chosen uniformly at random be-
tween $-2$ and $60$ and $y \in R [0, 1]$
(b) Node weights set to $y10^x$, where $x$ was an integer chosen uniformly at random be-
tween $-2$ and $30$ and $y \in R [0, 1]$

Figure 12: Histogram of edge weights in two extreme weight generation schemes

It may be that a dynamic sequence of weight updates (i.e. a $u$ value that changes at each
iteration), based on characteristics of the graph like degree and weight distribution, would work
better for the product generation scheme. Since the contraction algorithm requires making many
calls to the outside solver, with the exact number dependent on the weight update value, this is
also necessary to ensure it runs quickly. Developing a reasonable heuristic for selecting the best
parameters will be crucial to adapting the powerful contraction algorithm proposed by Cohen et al.
[2] for the rich variety of ill-conditioned Laplacian systems that arise in practice.
6 Acknowledgements

I’d like to thank Dan for his help throughout this project, from crystal-clear explanations to sharing his many intuitions about algorithms and graphs. I’m very grateful for the introduction to this rich and intriguing area of study.

7 Utilities

\( f \) — Given \( L, \vec{b}, r, u, \) and a choice of mode (shift, round, or drop to alter \( \lambda_2 \)), use the contraction algorithm to return an approximate value for \( L^+ \vec{b} \)

\( \text{fprecon} \) — In addition to the parameters above, input a step size and number of iterations \( T \), and return an approximate value for \( L^+ \vec{b} \) via preconditioned Richardson iteration, with preconditioner \( f \) above

7.1 For Use in Algorithm

\( \text{contractvec} \) — Given \( \vec{v} \), efficiently compute \( C(\vec{v}) \)

\( \text{expandvec} \) — Given \( \vec{v} \), efficiently compute \( C^T(\vec{v}) \)

\( \text{contractmat} \) — Given \( M \), efficiently compute \( C(M) \)

\( \text{proj} \) — Given \( \vec{v} \) and the component of each node, efficiently compute \( \text{Proj}(\vec{v}) \)

\( \text{roundupedges} \) — Given \( \alpha \) and \( L \) (respectively, \( A \)), return \( L' \) (\( A' \)) such that positive edge weights below \( \alpha \) are rounded up to \( \alpha \)

\( \text{dropedges} \) — Given \( \alpha \) and \( L \) (respectively, \( A \)), return \( L' \) (\( A' \)) such that positive edge weights below \( \alpha \) are dropped

Note that all of the above use Julia’s \texttt{sparse} function for efficiency.

7.2 For Testing

\( \text{err} \) — Given \( \vec{x}^*, \vec{x} \), and high-precision adjacency matrix \( A \) (with corresponding \( L \)), compute

\[
\frac{||\vec{x} - \vec{x}^*||_L}{||\vec{x}^*||_L}
\]

\( \text{illcondadj} \) — Given \( a, b \), and a base, undirected graph, generate an ill-conditioned adjacency matrix using the “maximum” or “product” schemes described earlier.

\( \text{condofsparse} \) — Estimate the condition number of a Laplacian, with optional positive constant added to the diagonal.

References
