Hardness Results for Laplacians of Simplicial Complexes via Sparse-Linear Equation Complete Gadgets

Ming Ding
ming.ding@inf.ethz.ch
Department of Computer Science
ETH Zurich

Rasmus Kyng*
kyng@inf.ethz.ch
Department of Computer Science
ETH Zurich

Maximilian Probst Gutenberg*
maximilian.probst@inf.ethz.ch
Department of Computer Science
ETH Zurich

Peng Zhang
pz149@rutgers.edu
Department of Computer Science
Rutgers University

Abstract

We study linear equations in combinatorial Laplacians of k-dimensional simplicial complexes (k-complexes), a natural generalization of graph Laplacians. Combinatorial Laplacians play a crucial role in homology and are a central tool in topology. Beyond this, they have various applications in data analysis and physical modeling problems. It is known that nearly-linear time solvers exist for graph Laplacians. However, nearly-linear time solvers for combinatorial Laplacians are only known for restricted classes of complexes.

This paper shows that linear equations in combinatorial Laplacians of 2-complexes are as hard to solve as general linear equations. More precisely, for any constant $c \geq 1$, if we can solve linear equations in combinatorial Laplacians of 2-complexes up to high accuracy in time $\tilde{O}((\#\text{ of nonzero coefficients})^c)$, then we can solve general linear equations with polynomially bounded integer coefficients and condition numbers up to high accuracy in time $\tilde{O}((\#\text{ of nonzero coefficients})^c)$. We prove this by a nearly-linear time reduction from general linear equations to combinatorial Laplacians of 2-complexes. Our reduction preserves the sparsity of the problem instances up to poly-logarithmic factors.

*The research leading to these results has received funding from grant no. 200021 204787 of the Swiss National Science Foundation.

Contents

1 Introduction

1.1 Simplicial complexes, homology, and combinatorial Laplacians

We study linear equations whose coefficient matrices are combinatorial Laplacians of k-dimensional abstract simplicial complexes (k-complexes), which generalize the well-studied graph Laplacians. An abstract simplicial complex $\mathcal{K}$ is a family of sets, known as simplices, closed under taking subsets, i.e., every subset of a set in $\mathcal{K}$ is also in $\mathcal{K}$. The dimension of $\mathcal{K}$ is the largest size of the simplices in $\mathcal{K}$ minus 1. A geometric notion of abstract simplicial complexes is simplicial complexes, under which a k-simplex is the convex hull of $k+1$ vertices (for example, 0,1,2-simplexes are vertices, edges, and triangles, respectively). In particular, complexes in 1 dimension are graphs; combinatorial Laplacians in 1-complexes are graph Laplacians.

Nearly-linear time solvers exist for linear equations in graph Laplacians [ST14; KMP10; KMP11; PS14; Coh+14b; KS16; JS21], and their natural generalizations including such as connection Laplacians [Kyn+16] and directed Laplacians [Coh+17; Coh+18]. However, nearly-linear time solvers for linear equations in combinatorial Laplacians are only known for very restricted classes of 2-complexes [Coh+14a; Bla+22]. We ask whether one can extend these nearly-linear solvers to general combinatorial Laplacians.

Combinatorial Laplacians are defined via boundary operators of the chain spaces of an oriented complex. Given an oriented simplicial complex $\mathcal{K}$, a k-chain is a (signed) weighted sum of the k-simplices in $\mathcal{K}$. The boundary operator ∂_k is a linear map from the k-chain space to the $(k-1)$-chain space; in particular, it maps a k-simplex to a signed sum of its boundary $(k-1)$-simplices, where the signs are determined by the orientations of the k-simplex and its boundary $(k-1)$-simplices. For example, ∂_1 is the oriented vertex-edge incidence matrix. The combinatorial Laplacian $\mathcal{L}_k$ is defined to be $\partial_{k+1}\partial_{k+1}^\top + \partial_k^\top \partial_k$. In particular, $\mathcal{L}_0 = \partial_1 \partial_1^\top$ is the graph Laplacian.

Combinatorial Laplacians play an important role in both pure mathematics and applied areas. These matrices originate in the study of discrete Hodge decomposition [Eck44]: The kernel of $\mathcal{L}_k$ is isomorphic to the kth homology space of $\mathcal{K}$. The properties of combinatorial Laplacians are studied in subsequent works [Fri98; DW02; DKM09; DKM15; MN21]. A central problem in homology theory is evaluating the Betti number of the kth homology space, which equals the rank of $\mathcal{L}_k$. In the case of homology over the reals, computing the rank of $\mathcal{L}_k$ can be reduced to solving a poly-logarithmic number of linear equations in $\mathcal{L}_k$ [BV21]. Computation of Betti numbers over the reals is a key step in numerous problems in applied topology, computational topology, and topological data analysis [Zom05; Ghr08; Car09; EH10; Cha+16]. In addition, combinatorial Laplacians have applications in statistical ranking [Jia+11; Xu+12], graphics and images [Ma+11; Ton+03], electromagnetism and fluids mechanics [DKT08], data representations [CMZ18], cryo-electron microscopy [YL17], biology [Sch+20]. We refer to the readers to [Lim20] for an accessible survey.

The reader may be puzzled that despite a vast literature on combinatorial Laplacians and their central role in topology, little is known about solving linear equations in these matrices except in very restricted cases [Coh+14a; Bla+22]. In this paper, we show that approximately solving linear equations in general combinatorial Laplacians is as hard as approximately solving general linear equations over the reals, which explains the lack of special-purpose solvers for this class of equations. More precisely, if one can solve linear equations in combinatorial Laplacians of general 2-complexes to high accuracy in time $\tilde{O}((\#\text{ of nonzero coefficients})^c)$[1] for some constant $c \geq 1$, then one can solve general linear equations with polynomially bounded integer coefficients and condition numbers up to high accuracy in time $\tilde{O}((\#\text{ of nonzero coefficients})^c)$.

[1] $\tilde{O}$ hides poly-logarithmic factors in condition numbers and inverse of the accuracy parameter.

A recent breakthrough shows that general linear equations can be solved up to high accuracy in time $\widetilde{O}((\#\text{ of nonzero coefficients})^{2.27159})$ [PV21; Nie21], which for sparse linear equations is asymptotically faster than the long-standing runtime barrier $\tilde{O}(n^{2.3728596})$ [AW21] by fast matrix multiplication. Understanding the optimal value of c is a major open problem in numerical linear algebra. Our result, viewed more positively, shows that one can reduce designing fast solvers for general linear equations to that for combinatorial Laplacians.

1.2 Hardness results based on linear equations

Kyng and Zhang [KZ20] initiated the study of hardness results for solving structured linear equations. They showed that solving linear equations in a slight generalization of graph Laplacians such as 2-commodity Laplacians, 2-dimensional truss stiffness matrices, and 2-total-variation matrices is as hard as solving general linear equations.

Slightly more formally, consider an invertible matrix $\boldsymbol{A}$ and a right-hand side vector $\boldsymbol{b}$, both with polynomially bounded integer entries and $\boldsymbol{A}$ having a polynomially bounded condition number. Suppose we want to approximately solve linear equation $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, e.g., finding $\tilde{\boldsymbol{x}}$ such that $\|\boldsymbol{A}\tilde{\boldsymbol{x}} - \boldsymbol{b}\|_2 \leq \epsilon \|\boldsymbol{b}\|_2$ for some polynomially small ϵ. Now consider a family of matrices $\mathcal{B}$, and suppose that for any linear equation instance $(\boldsymbol{A}, \boldsymbol{b}, \epsilon)$ as described above, we can produce matrix $\boldsymbol{B} \in \mathcal{B}$, vector $\boldsymbol{c}$, and accuracy parameter δ, such that if we can solve $\boldsymbol{B}\boldsymbol{y} = \boldsymbol{c}$ up to error δ, then we can produce $\tilde{\boldsymbol{x}}$ that solves $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ to the desired accuracy.

If, given $(\boldsymbol{A}, \boldsymbol{b}, \epsilon)$, we can compute $(\boldsymbol{B}, \boldsymbol{c}, \delta)$ in $\tilde{O}(\text{nnz}(\boldsymbol{A}))$ time with $\text{nnz}(\boldsymbol{B}) = \tilde{O}(\text{nnz}(\boldsymbol{A}))$ and $\boldsymbol{B}$ having polynomially bounded entries and condition number, then we say that the class $\mathcal{B}$ is *sparse-linear-equation complete* (see Section 2 for a formal definition that also extends to non-invertible matrices). The reason for this terminology is that if a solver with polynomially small error and runtime $\tilde{O}((\#\text{ of nonzero coefficients})^c)$ is known for the class $\mathcal{B}$ whenever the matrix has polynomially bounded condition number, then a solver with polynomially small error and runtime $\tilde{O}((\#\text{ of nonzero coefficients})^c)$ exists for general matrices with polynomially bounded integer entries and condition numbers. Such solvers are known for the classes of Laplacian Matrices, Directed Laplacian Matrices, Connection Laplacian Matrices, and several more classes, all with $c = 1$. Thus, if any of these classes were sparse-linear-equation complete, we would immediately get nearly-linear time solvers for general linear equations.

In this language, Kyng and Zhang [KZ20] showed that 2-commodity Laplacians, 2-dimensional truss stiffness matrices, and 2-total-variation matrices are all sparse-linear-equation complete. We note that [KWZ20] considered a larger family of hardness assumptions based on linear equations, which, among other things, can express weaker hardness statements based on weaker reductions.

1.3 Our contributions

In the terminology established above, our main result can be stated very succinctly: Linear equations in combinatorial Laplacians of 2-complexes are sparse-linear-equation complete.

In fact, we show this by showing an even simpler problem is sparse-linear-equation complete, namely linear equations in the boundary operator of a 2-complex. This result is formally stated in Theorem 3.1. Our proof establishes the sparse-linear-equation completeness of boundary operators of a 2-complex via a two-step reduction.

In the first reduction step, we show sparse-linear-equation completeness of a very simple class of linear equations which we call *difference-average equations*: these are equations where every row either restricts the difference of two variables: $\boldsymbol{x}(i) - \boldsymbol{x}(j) = b$, or it sets one variable to be the average of two others: $\boldsymbol{x}(i) + \boldsymbol{x}(j) = 2\boldsymbol{x}(k)$. This reduction was implicitly proved in [KZ20] as

an intermediate step. In this paper, we make the reduction explicit, which may be of independent interest, as this reduction class is likely to be a good starting point for many other hardness reductions. One can think of this step as analogous to showing that 3-SAT is NP-complete: It gives us a simple starting point for proving the hardness of other problems. The formal theorem statement appears in Theorem 2.9.

In our second reduction step, we reduce a given difference-average equation problem to a linear equation in the boundary operator of a 2-complex.

Both the two steps preserve the number of nonzero coefficients in the linear equations up to a logarithmic factor, and only blow up the coefficients and condition numbers polynomially. The reductions are also robust to error in the sense that to solve the original problem to high accuracy, it suffices to solve the reduced problem to accuracy at most polynomially higher. Finally, we can compose the two reductions to show that solving linear equations in 2-complex boundary operators to high accuracy is as hard as solving general linear equations with polynomially bounded integer coefficients and condition numbers to high accuracy.

We give more details on both reductions below, but first we describe how to show that solving linear equations in combinatorial Laplacians $\mathcal{L}_1$ is also sparse-linear-equation complete.

1.3.1 Reduction from boundary operators of complexes to combinatorial Laplacians

Our main result shows that the class of linear equations in the boundary operators of 2-complexes is sparse-linear-equation complete. This also implies that combinatorial Laplacians $\mathcal{L}_1$ are as hard to solve as general linear equations, via standard arguments that we only sketch here. In particular, suppose we have a high-accuracy solver for combinatorial Laplacians of 2-complexes, which essentially means we can apply the Moore-Penrose pseudo-inverse of $\mathcal{L}_1$ to obtain $\mathcal{L}_1^\dagger \boldsymbol{d}$ up to a polynomially small error. A central and basic fact in the study of simplicial homology is that $\mathrm{im}(\partial_1^\top) \cap \mathrm{im}(\partial_2) = \{\mathbf{0}\}$. Because of this, $\mathcal{L}_1^\dagger = (\partial_1^\top \partial_1)^\dagger + (\partial_2 \partial_2^\top)^\dagger$, and using a graph Laplacian linear equation solver, we can approximate $(\partial_1^\top \partial_1)^\dagger \boldsymbol{d}$ to high accuracy in nearly-linear time, which in turn allows us to compute a high accuracy estimate of

$$(\partial_2 \partial_2^\top)^\dagger \boldsymbol{d} = \mathcal{L}_1^\dagger \boldsymbol{d} - (\partial_1^\top \partial_1)^\dagger \boldsymbol{d}.$$

Finally, suppose we wish to solve $\partial_2 \boldsymbol{f}_2 = \boldsymbol{d}$, or, more generally, compute $\partial_2^\dagger \boldsymbol{d}$. Then we can leverage that $\partial_2^\top (\partial_2 \partial_2^\top)^\dagger \boldsymbol{d} = \partial_2^\dagger \boldsymbol{d}$. Thus, if we are able to apply $\mathcal{L}_1^\dagger$, we can solve linear equations in ∂_2. Finally, one should note that $\mathrm{nnz}(\mathcal{L}_1) = O(\mathrm{nnz}(\partial_2))$, and that using our definition of condition number (see Section 2), both have polynomially related condition number – this also means a high accuracy solve in one can be converted to a high accuracy solve in the other.

In the discussion above, we saw that applying $(\partial_2 \partial_2^\top)^\dagger$ is at least as hard as solving linear equations in ∂_2. The problem of applying $(\partial_2 \partial_2^\top)^\dagger$ also arises when using Interior Point Methods to solve a generalized max-flow problem in higher-dimensional simplicial complexes as defined in [MN21]. We sketch how this pseudo-inverse problem arises when using an Interior Point Method in Appendix B.

1.3.2 Sparse-linear-equation completeness of Difference-Average equations

Our first reduction transforms general linear equations with polynomially bounded integer entries and condition numbers into difference-average equations. We first transform a general linear equation instance to a linear equation instance such that the coefficient matrix has row sum zero and the sum of positive coefficients in each row is a power of 2, by introducing a constant number

of more variables and equations. Then, we transform each single equation to a set of difference-average equations by bit-wise pairing and replacing each pair of variables with a new variable via an average equation.

1.3.3 Sparse-linear-equation completeness of boundary operators of simplicial complexes

Our second reduction transforms difference-average linear equations into linear equations in the boundary operators of 2-complexes. Solving $\partial_2 \boldsymbol{f} = \boldsymbol{d}$ can be interpreted as computing a flow $\boldsymbol{f}$ in the triangle space of a 2-complex subject to pre-specified edge demands $\boldsymbol{f}$.

Our reduction is inspired by a reduction in [MN21] that proves NP-hardness of computing maximum *integral* flows in 2-complexes via a reduction from graph 3-coloring problem. However, the correctness of their reduction heavily relies on that the flow values in the 2-complex are 0-1 integers, which does not apply in our setting. In addition, it is unclear how to encode linear equations as a graph coloring problem even if fractional colors are allowed.

We employ some basic building blocks used in [MN21] including punctured spheres and tubes. However, we need to carefully arrange and orient the triangles in the 2-complex to encode both the positive and negative coefficients in difference-average equations, and we need to express the averaging relations not covered by the previous work.

An important aspect of our contribution is that we carefully control the number of non-zeros of the boundary operator matrix that we construct, and we bound the condition number of this matrix and how error propagates from an approximate solution to the boundary operator problem back to the original difference-average equations. In order to do so, we develop explicit triangulation algorithms that specify the exact number of triangles needed to triangulate each building block, and allows a detailed error and condition number analysis.

We remark that our constructed 2-complex does not admit an embedding into a sphere in 3 dimensions. Recent work [Bla+22] has shown that simplicial complexes with a known embedding into $\mathbb{R}^3$ have non-trivial linear equation solvers, but the full extent to which embeddability can lead to better solvers remains an open question.

Our construction works in the Real RAM model. But it can be transferred to the fixed point arithmetic with $(\log n)^{O(1)}$ bits per number, where n is the size of the problem instance.

1.4 Related works

Generalized flows and cuts. Recall that in a graph, flows or circulations are vectors in the kernel of the boundary operator ∂_1, and cuts are vectors in the image of ∂_1. One can generalize the notions of flows and cuts to higher-dimensional simplicial complexes [DKM15; MN21]. In a k-complex, flows are defined to be vectors in the kernel of ∂_k, and cuts the vectors in the image of ∂_k. Given a demand vector $\boldsymbol{d}$ in the image of ∂_k and a capacity vector $\boldsymbol{c}$ for the k-simplexes, the task of the generalized max-flow problem is to compute a k-chain flow $\boldsymbol{f}$ satisfying $\partial_k \boldsymbol{f} = \alpha \boldsymbol{d}$ and $\boldsymbol{0} \leq \boldsymbol{f} \leq \boldsymbol{c}$ to maximize the flow value α.

Solving linear equations. Linear equations are ubiquitous in computational mathematics, computer science, engineering, physics, biology, and economics. The currently best known algorithm for solving general dense linear equations in dimensions $n \times n$ runs in time $\tilde{O}(n^\omega)$, where $\omega < 2.3728596$ is the matrix multiplication constant [AW21]. For sparse linear equations with N nonzero coefficients and condition number κ, the best known approximate algorithms run in time

$\widetilde{O}(\min\{N^{2.27159}, N\kappa\})$, where the first runtime is from [PV21; Nie21] and the second is by the conjugate gradient [2] [HS+52].

In contract to general linear equations, linear equations in graph Laplacians and its generalizations can be solved asymptotically faster, as mentioned earlier. In addition, faster solvers are also known for restricted classes of total-variation matrices [KMT11], stiffness matrices from elliptic finite element systems [BHV08], and 2 or 3-dimensional truss stiffness matrices [DS07; Kyn+18]. An interesting question is whether one can generalize these faster solvers to more classes of matrices.

Reduction from sparse linear equations. [KWZ20] defines a parameterized family of hypotheses for runtime of solving sparse linear equations. Under these hypotheses, they prove hardness of approximately solving packing and covering linear programs. For example, if one can solve a packing linear program up to ϵ accuracy in time $\tilde{O}(\#\text{ of nonzero coefficients} \times \epsilon^{-0.165})$, then one can solve a system of linear equations in time asymptotically faster than $\tilde{O}(\#\text{ of nonzero coefficients} \times$ condition number of matrix), which is the runtime of conjugate gradient.

1.5 Organization of the remaining paper

In Section 2, we present some basic background knowledge related to simplicial homology and systems of linear equations. In Section 3, we state our main theorem, together with an overview of our proof. We first show a reduction for difference-average linear equations to 2-complex boundary operation linear equations under the assumption that the right-hand side vector is in the image of the coefficient matrix. We describe the reduction algorithm and analyze it for exact solvers in Section 4, and analyze it for approximate solvers in Section 5. Then, we slightly modify the reduction for approximate solvers in general case (that is, without the assumption) in Section 6. In addition, we reduce general linear equations to difference-average linear equations in Appendix A.

2 Preliminaries

2.1 Simplicial homology

We define the basic concepts of simplicial homology. We recommend the readers the books [Mun18] and [Hat00] for a more complete treatment.

Simplicial complexes. A *k-dimensional simplex* (k-simplex) $\sigma = \text{conv}\{v_0, \ldots, v_k\}$ is the convex hull of $k+1$ affinely independent points $v_0, \ldots, v_k$. For example, 0,1,2-simplexes are vertices, edges, and triangles, respectively. A *face* of σ is the convex hull of a non-empty subset of $\{v_0, v_1, \ldots, v_k\}$. An *orientation* of σ is given by an ordering Π of its vertices, written as $\sigma = [v_{\Pi(0)}, \ldots, v_{\Pi(k)}]$, such that two orderings define the same orientation if and only if they differ by an even permutation. If Π is even, then $[v_{\Pi(0)}, \ldots, v_{\Pi(k)}] = [v_0, \ldots, v_k]$; if Π is odd, then $[v_{\Pi(0)}, \ldots, v_{\Pi(k)}] = -[v_0, \ldots, v_k]$.

A *simplicial complex* $\mathcal{K}$ is a finite collection of simplexes such that (1) for every $\sigma \in \mathcal{K}$ if $\tau \subset \sigma$ then $\tau \in \mathcal{K}$ and (2) for every $\sigma_1, \sigma_2 \in \mathcal{K}$, $\sigma_1 \cap \sigma_2$ is either empty or a face of both σ_1, σ_2. The *dimensions* of $\mathcal{K}$ is the maximum dimension of any simplex in $\mathcal{K}$. We refer to a simplicial complex in k dimensions as a k-complex.

[2] If the coefficient matrix is symmetric positive semidefinite, the runtime is $\tilde{O}(N\sqrt{\kappa})$.

Boundary operators. A k-chain is a formal sum of the oriented k-simplices in $\mathcal{K}$ with the coefficients over $\mathbb{R}$. Let $C_k(\mathcal{K})$ denote the kth chain space. The *boundary operator* is a linear map $\partial_k : C_k(\mathcal{K}) \to C_{k-1}(\mathcal{K})$ such that for an oriented k-simplex $\sigma = [v_0, v_1, \ldots, v_k]$,

$$\partial_k(\sigma) = \sum_{i=0}^{k} (-1)^i [v_0, \ldots, \hat{v}_i, \ldots, v_k],$$

where $[v_0, \ldots, \hat{v}_i, \ldots, v_k]$ is the oriented $(k-1)$-simplex obtained by removing v_i from σ, and $(-1)^i$ is its *induced orientation*. The operator ∂_k can be written as a matrix in $n_{k-1} \times n_k$ dimensions, where n_d is the number of d-simplices in $\mathcal{K}$. The (i,j)th entry of ∂_k is ± 1 if the ith $(k-1)$-simplex is a face of the jth k-simplex where the sign is determined by the orientations, and 0 otherwise. See Figure 1 for an example.

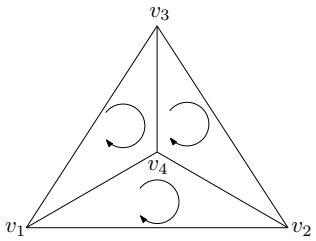


Figure 1: An example of boundary operator. We set a clockwise orientation for 2-simplices, and set the orientation for 1-simplices as the order of increased vertex indices.

$$\partial_2 = \begin{array}{c} \\ [v_1,v_2] \\ [v_2,v_3] \\ [v_1,v_3] \\ [v_1,v_4] \\ [v_2,v_4] \\ [v_3,v_4] \end{array} \begin{array}{c} \begin{array}{ccc} [v_1,v_4,v_2] & [v_2,v_4,v_3] & [v_1,v_3,v_4] \end{array} \\ \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} \end{array}.$$

An important property of the boundary operator is that applying the boundary operator twice results in the trivial operator, i.e.,

$$\partial_{k-1}\partial_k = \mathbf{0}. \tag{1}$$

This implies $\mathrm{im}(\partial_k) \subseteq \ker(\partial_{k-1})$. Thus, we can define the quotient space $H_k = \ker(\partial_k) \setminus \mathrm{im}(\partial_{k+1})$, referred to as the kth homology space of $\mathcal{K}$. The dimension of H_k is the kth Betti number of $\mathcal{K}$, which plays an important role in understanding the homology spaces.

Hodge theory and combinatorial Laplacians. Combinatorial Laplacians arise from the discrete Hodge decomposition.

Theorem 2.1 (Hodge decomposition [Lim20])**.** *Let $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{B} \in \mathbb{R}^{n \times p}$ be matrices satisfying $\boldsymbol{AB} = \mathbf{0}$. Then, there is an orthogonal direct sum decomposition*

$$\mathbb{R}^n = \mathrm{im}(\boldsymbol{A}^\top) \oplus \ker(\boldsymbol{A}^\top \boldsymbol{A} + \boldsymbol{B}\boldsymbol{B}^\top) \oplus \mathrm{im}(\boldsymbol{B}).$$

By Eq. (1), it is valid to set $\boldsymbol{A} = \partial_k$ and $\boldsymbol{B} = \partial_{k+1}$. The matrix we get in the middle term is the *combinatorial Laplacian*:

$$\mathcal{L}_k \stackrel{\text{def}}{=} \partial_k^\top \partial_k + \partial_{k+1}\partial_{k+1}^\top$$

In particular, $\mathcal{L}_0 = \partial_1 \partial_1^\top$ is the graph Laplacian. The kth homology space $H_k(\mathcal{K})$ is isomorphic to $\ker(\mathcal{L}_k)$, and thus the kth Betti number of $\mathcal{K}$ equals the rank of $\ker(\mathcal{L}_k)$.

Triangulation. A *triangulation* of a topological space $\mathcal{X}$ is a 2-simplicial complex $\mathcal{K}$ together with a homeomorphism between $\mathcal{X}$ and $\mathcal{K}$. An *oriented triangulation* of a space is a triangulation such that every edge is contained in exactly two triangles, together with an orientation for each triangle such that any two neighboring triangles induce opposite signs on their shared edge.

2.2 Notations for matrices and vectors

We use parentheses to denote entries of a matrix or a vector: Let $\boldsymbol{A}(i,j)$ bet the (i,j)th entry of a matrix $\boldsymbol{A}$, and let $\boldsymbol{x}(i)$ bet the ith entry of a vector $\boldsymbol{x}$. We use $\mathbf{1}_n, \mathbf{0}_n$ to denote n-dimensional all-one vector and all-zero vector, respectively. We define $\|\boldsymbol{x}\|_{\max} = \max_{i\in[n]} |\boldsymbol{x}(i)|$, $\|\boldsymbol{x}\|_1 = \sum_{i\in[n]} |\boldsymbol{x}(i)|$. Given a matrix $\boldsymbol{A} \in \mathbb{R}^{d\times n}$, we use $\boldsymbol{A}(i)$ to denote the ith row of $\boldsymbol{A}$ and $\mathrm{nnz}(\boldsymbol{A})$ the number of nonzero entries of $\boldsymbol{A}$. Without loss of generality, we assume that $\mathrm{nnz}(\boldsymbol{A}) \geq \max\{d, n\}$. We let $\|\boldsymbol{A}\|_{\max} = \max_{i,j} |\boldsymbol{A}(i,j)|$. We use $\mathrm{im}(\boldsymbol{A})$ to denote the image (i.e., the column space) of $\boldsymbol{A}$ and $\mathrm{null}(\boldsymbol{A})$ the null space of $\boldsymbol{A}$. We let $\boldsymbol{\Pi}_{\boldsymbol{A}} = \boldsymbol{A}(\boldsymbol{A}\boldsymbol{A}^\top)^\dagger \boldsymbol{A}^\top$ be the orthogonal projection onto $\mathrm{im}(\boldsymbol{A})$, where $\boldsymbol{M}^\dagger$ is the pseudo-inverse of $\boldsymbol{M}$. Let $\lambda_{\max}(\boldsymbol{A})$ be the maximum eigenvalue of $\boldsymbol{A}$ and $\lambda_{\min}(\boldsymbol{A})$ the minimum *nonzero* eigenvalue of $\boldsymbol{A}$. The condition number of $\boldsymbol{A}$, denoted by $\kappa(\boldsymbol{A})$, is the ratio of the maximum to the minimum *nonzero* singular value of $\boldsymbol{A}$.

We define a function U that takes a matrix $\boldsymbol{A}$ and a vector $\boldsymbol{b}$ as arguments and returns the maximum of $\|\cdot\|$ of all the arguments, that is,

$$U(\boldsymbol{A}, \boldsymbol{b}) = \max\{\|\boldsymbol{A}\|_{\max}, \|\boldsymbol{b}\|_{\max}\}.$$

2.3 Systems of linear equations

We define approximately solving linear equations in a general form, following [KZ20]. For more details, we refer the readers to Section 2.1 of [KZ20].

Definition 2.2 (Linear Equation Problem (LE))**.** Given a matrix $\boldsymbol{A} \in \mathbb{R}^{d\times n}$, a vector $\boldsymbol{b} \in \mathbb{R}^d$, we refer to the linear equation problem for the tuple $(\boldsymbol{A}, \boldsymbol{b})$, denoted by $\mathrm{LE}(\boldsymbol{A}, \boldsymbol{b})$, as the problem of finding an $\boldsymbol{x} \in \mathbb{R}^n$ such that

$$\boldsymbol{x} \in \arg\min_{\boldsymbol{x}\in\mathbb{R}^n} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2 .$$

Fact 2.3. *Let* $\boldsymbol{x}^* \in \arg\min_{\boldsymbol{x}\in\mathbb{R}^n} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2$*. Then,*

$$\boldsymbol{A}\boldsymbol{x}^* = \boldsymbol{A}(\boldsymbol{A}^\top\boldsymbol{A})^\dagger \boldsymbol{A}^\top \boldsymbol{b} = \boldsymbol{\Pi}_{\boldsymbol{A}} \boldsymbol{b}$$

and

$$\|\boldsymbol{A}\boldsymbol{x}^* - \boldsymbol{b}\|_2^2 = \|(\boldsymbol{I} - \boldsymbol{\Pi}_{\boldsymbol{A}})\boldsymbol{b}\|_2^2$$

By the above fact, solving $\mathrm{LE}(\boldsymbol{A}, \boldsymbol{b})$ is equivalent to finding an $\boldsymbol{x}$ such that $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{b}$. This equation is known as the normal equation, and it is always feasible. If $\boldsymbol{b} \in \mathrm{im}(\boldsymbol{A})$, then $\boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{b} = \boldsymbol{b}$.

In practice, we are more interested in *approximately* solving linear equations, since numerical errors are unavoidably in data collection and computation and approximate solvers may run faster.

Definition 2.4 (Linear Equation Approximation Problem (LEA))**.** Given a matrix $\boldsymbol{A} \in \mathbb{R}^{d\times n}$, vectors $\boldsymbol{b} \in \mathbb{R}^d$, and an error parameter $\epsilon \in (0, 1]$, we refer to linear equation approximate problem for the tuple $(\boldsymbol{A}, \boldsymbol{b}, \epsilon)$, denoted by $\text{LEA}(\boldsymbol{A}, \boldsymbol{b}, \epsilon)$, as the problem of finding an $\boldsymbol{x} \in \mathbb{R}^n$ such that

$$\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{b}\|_2 \leq \epsilon \|\boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{b}\|_2 .$$

Fact 2.5. *Let $\boldsymbol{x}$ be a solution to $\text{LEA}(\boldsymbol{A}, \boldsymbol{b}, \epsilon)$. Then,*

$$\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 \leq \|\boldsymbol{A}\boldsymbol{x}^* - \boldsymbol{b}\|_2^2 + \epsilon^2 \|\boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{b}\|_2^2 .$$

The definition of the approximate error in Definition 2.4 is equivalent to several error notions that are commonly used in solving linear equations. In particular,

$$\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{b}\|_2 = \left\|\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{x} - \boldsymbol{A}^\top\boldsymbol{b}\right\|_{(\boldsymbol{A}^\top\boldsymbol{A})^\dagger} = \|\boldsymbol{x} - \boldsymbol{x}^*\|_{\boldsymbol{A}^\top\boldsymbol{A}} .$$

2.3.1 Matrix classes

We are interested in linear equations whose coefficient matrices belonging to the following matrix classes.

1. $\mathcal{G}$ refers to the class of *General Matrices* that have polynomially-bounded integer entries and condition numbers and do not have all-0 rows and all-0 columns. We refer to linear equations whose coefficient matrix is in $\mathcal{G}$ as *general linear equations*.

2. $\mathcal{DA}$ refers to the class of *Difference-Average Matrices* whose rows fall into two categories:

   (a) A *difference row* which has exactly two nonzero entries 1 and -1;

   (b) An *average row* which has exactly three nonzero entries 1, 1, and -2.

   Multiplying a difference row vector to a column vector $\boldsymbol{x}$ gives $\boldsymbol{x}(i) - \boldsymbol{x}(j)$; multiplying an average row vector to $\boldsymbol{x}$ gives $\boldsymbol{x}(i)+\boldsymbol{x}(j)-2\boldsymbol{x}(k)$. We refer to linear equations whose coefficient matrix is in $\mathcal{DA}$ as *difference-average linear equations*.

3. $\mathcal{B}_2$ refers to the class of *Boundary Operator Matrices ∂_2 in 2-complexes*. We refer to linear equations whose coefficient matrix is in $\mathcal{B}_2$ as *2-complex boundary linear equations*.

2.3.2 Reduction between linear equations

We will again follow the definition of efficient reductions in [KZ20]. We say LEA over matrix class $\mathcal{M}_1$ is *nearly-linear time reducible* to LEA over matrix class $\mathcal{M}_2$, denoted by $\mathcal{M}_1 \leq_{nlt} \mathcal{M}_2$, if the following holds:

1. There is an algorithm that maps an arbitrary instance $\text{LEA}(\boldsymbol{M}_1, \boldsymbol{c}_1, \epsilon_1)$ where $\boldsymbol{M}_1 \in \mathcal{M}_1$ to an instance $\text{LEA}(\boldsymbol{M}_2, \boldsymbol{c}_2, \epsilon_2)$ where $\boldsymbol{M}_2 \in \mathcal{M}_2$ such that there is another algorithm that can map a solution to $\text{LEA}(\boldsymbol{M}_2, \boldsymbol{c}_2, \epsilon_2)$ to a solution to $\text{LEA}(\boldsymbol{M}_1, \boldsymbol{c}_1, \epsilon_1)$.

2. Both the two algorithms run in time $O(\text{nnz}(\boldsymbol{M}_1))$.

3. In addition, we can guarantee $\text{nnz}(\boldsymbol{M}_2) = \widetilde{O}(\text{nnz}(\boldsymbol{M}_1))$, and $\epsilon_2^{-1}, \kappa(\boldsymbol{M}_2), U(\boldsymbol{M}_2, \boldsymbol{b}_2) = \text{poly}(\text{nnz}(\boldsymbol{M}_1), \epsilon_1^{-1}, \kappa(\boldsymbol{M}_1), U(\boldsymbol{M}_1, \boldsymbol{b}_1))$.

We do not require a nearly-linear time reduction to preserve the dimensions of linear equations. The dimension of the new linear equation instance that we construct can be much larger than that of the original instance. On the other hand, a reduction that *only* preserves dimensions may construct a dense linear equation instance even if the original instance is sparse. A nearly-linear time reduction that preserves *both* the number of nonzeros and dimensions is stronger.

Fact 2.6. *If $\mathcal{M}_1 \leq_{nlt} \mathcal{M}_2$ and $\mathcal{M}_2 \leq_{nlt} \mathcal{M}_3$, then $\mathcal{M}_1 \leq_{nlt} \mathcal{M}_3$.*

Definition 2.7 (Sparse linear equation complete (SLE-complete))**.** We say LEA over a matrix class $\mathcal{M}$ is *sparse-linear-equation-complete* if $\mathcal{G} \leq_{nlt} \mathcal{M}$.

Fact 2.8. *Suppose* LEA *over $\mathcal{M}$ is* SLE*-complete. If one can solve all instances* $\text{LEA}(\boldsymbol{A}, \boldsymbol{b}, \epsilon)$ *with $\boldsymbol{A} \in \mathcal{M}$ in time $\tilde{O}(\text{nnz}(\boldsymbol{A})^c)$ where $c \geq 1$, then one can solve all instances* $\text{LEA}(\boldsymbol{A}', \boldsymbol{b}', \epsilon')$ *with $\boldsymbol{A}' \in \mathcal{G}$ in time $\tilde{O}(\text{nnz}(\boldsymbol{A}')^c)$.*

Under the above definitions, [KZ20] implicitly shows the following results. We provide an explicit and simplified proof in Appendix A.

Theorem 2.9 (Implicitly stated in [KZ20])**.** LEA *over $\mathcal{DA}$ is* SLE*-complete.*

3 Main results

Our main result is stated in the following theorem.

Theorem 3.1. LEA *over $\mathcal{B}_2$ is* SLE*-complete.*

Although our main theorem focuses on linear equation approximate problems, we construct nearly-linear time reductions for both linear equation problem LE and its approximate counterpart LEA. We first reduce LE and LEA $(\boldsymbol{A}, \boldsymbol{b})$ over difference-average matrices to those over 2-complex boundary operator matrices, *under the assumption* $\boldsymbol{b} \in \text{im}(\boldsymbol{A})$ (stated in Theorem 3.2 and 3.3). In this case, the constructed 2-complexes have unit edge weights. We then provide a slightly modified nearly-linear time reduction for LEA $(\boldsymbol{A}, \boldsymbol{b}, \epsilon)$ over difference-average matrices to LEA over 2-complex boundary operator matrices *without assuming* $\boldsymbol{b} \in \text{im}(\boldsymbol{A})$ (stated in Theorem 3.4). In this case, we introduce polynomially bounded edge weights for the constructed 2-complexes.

Theorem 3.2. *Given a linear equation instance* LE $(\boldsymbol{A}, \boldsymbol{b})$ *where $\boldsymbol{A} \in \mathcal{DA}$ and $\boldsymbol{b} \in \text{im}(\boldsymbol{A})$, we can reduce it to an instance* LE (∂_2, γ) *where $\partial_2 \in \mathcal{B}_2$, in time $O(\text{nnz}(\boldsymbol{A}))$, such that a solution to* LE (∂_2, γ) *can be mapped to a solution to* LE $(\boldsymbol{A}, \boldsymbol{b})$ *in time $O(\text{nnz}(\boldsymbol{A}))$.*

Theorem 3.3. *Given a linear equation instance* LEA $(\boldsymbol{A}, \boldsymbol{b}, \epsilon^{DA})$ *where $\boldsymbol{A} \in \mathcal{DA}$ and $\boldsymbol{b} \in \text{im}(\boldsymbol{A})$, we can reduce it to an instance* LEA $(\partial_2, \gamma, \epsilon^{B_2})$ *where $\partial_2 \in \mathcal{B}_2$ and $\epsilon^{B_2} \leq \frac{\epsilon^{DA}}{42\,\text{nnz}(\boldsymbol{A})}$, in time $O(\text{nnz}(\boldsymbol{A}))$, such that a solution to* LEA $(\partial_2, \gamma, \epsilon^{B_2})$ *can be mapped to a solution to* LEA $(\boldsymbol{A}, \boldsymbol{b}, \epsilon^{DA})$ *in time $O(\text{nnz}(\boldsymbol{A}))$.*

Theorem 3.4. *Given an instance* LEA $(\boldsymbol{A}, \boldsymbol{b}, \epsilon^{DA})$ *where $\boldsymbol{A} \in \mathcal{DA}$, we can reduce it to an instance* LEA $(\boldsymbol{W}^{1/2}\partial_2, \boldsymbol{W}^{1/2}\gamma, \epsilon^{B_2})$ *where $\partial_2 \in \mathcal{B}_2$ and $\boldsymbol{W}$ is a diagonal matrix with nonnegative diagonals, in time $O(\text{nnz}(\boldsymbol{A}))$. Let s, ϵ, K, U denote $\text{nnz}(\boldsymbol{A}), \epsilon^{DA}, \kappa(\boldsymbol{A}), U(\boldsymbol{A}, \boldsymbol{b})$, respectively. Then, we can guarantee that*

$$\text{nnz}(\partial_2) = O(s),\ U(\boldsymbol{W}^{1/2}\partial_2, \boldsymbol{W}^{1/2}\gamma) = O\left(sU\epsilon^{-1}\right),$$
$$\epsilon^{B_2} = \Omega(\epsilon U^{-1}s^{-1}),\ \kappa(\boldsymbol{W}^{1/2}\partial_2) = O\left(s^{15/2}K^2\epsilon^{-2}\right)$$

and a solution to LEA $(\boldsymbol{W}^{1/2}\partial_2, \boldsymbol{W}^{1/2}\gamma, \epsilon^{B_2})$ *can be mapped to a solution to* LEA $(\boldsymbol{A}, \boldsymbol{b}, \epsilon^{DA})$ *in time $O(\text{nnz}(\boldsymbol{A}))$.*

We will prove Theorem 3.2 in Section 4, Theorem 3.3 in Section 5, and Theorem 3.4 in Section 6.

3.1 Overview of our proof

Multiplying a 2-complex boundary operator $\partial_2 \in \mathbb{R}^{m\times t}$ to a vector $\boldsymbol{f} \in \mathbb{R}^t$ can be interpreted as transforming flows in the triangle space to flows in the edge space. Given $\boldsymbol{d} \in \mathbb{R}^d$, solving $\partial_2 \boldsymbol{f} = \boldsymbol{d}$ can be interpreted as finding flows in the triangle space subject to edge demands in $\boldsymbol{d}$. We will encode difference-average linear equations as a 2-complex flow network.

Encode a single equation. We observe a simple fact: If we glue two triangles Δ_1, Δ_2 with the same orientation, then the net flow $\partial_2 \boldsymbol{f}$ on the shared edge is $\boldsymbol{f}(\Delta_1) - \boldsymbol{f}(\Delta_2)$ (see Figure 2 (a)); if we glue two triangles Δ_1, Δ_2 with opposite orientations, then the net flow $\partial_2 \boldsymbol{f}$ on the shared edge is $\boldsymbol{f}(\Delta_1) + \boldsymbol{f}(\Delta_2)$ (see Figure 2 (b)). Given an equation $\boldsymbol{a}^\top \boldsymbol{x} = b$ with the nonzero coefficients being ± 1, we can encode it by gluing more triangles as above and setting the demand of the shared edge to be b. To handle the coefficient -2 in an average equation, say $\boldsymbol{x}(i) + \boldsymbol{x}(j) - 2\boldsymbol{x}(k)$, we implicitly interpret it as $\boldsymbol{x}(i) + \boldsymbol{x}(j) - \boldsymbol{x}(k_1) - \boldsymbol{x}(k_2)$ together with an additional difference equation $\boldsymbol{x}(k_1) = \boldsymbol{x}(k_2)$ (see Figure 2 (c)).

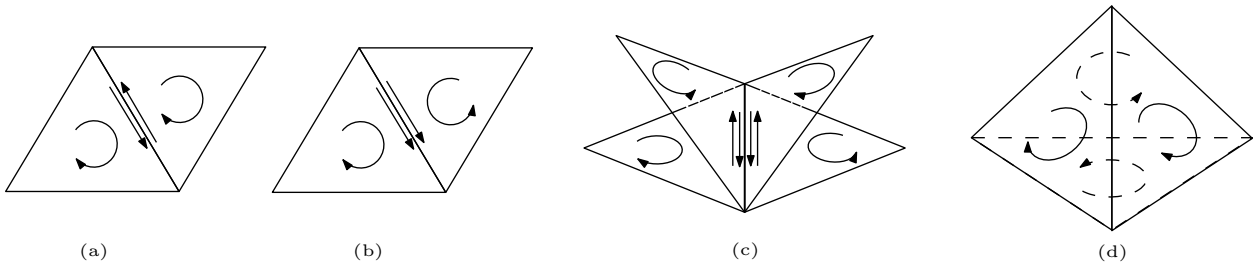


Figure 2: An illustration for encoding a single equation and encoding a variable.

Encode a variable. We use a sphere to encode a variable involved in many equations. We can obtain an oriented triangulation of the sphere and set all the edge demand to be 0 so that all the triangles on the sphere must have an equal flow value (see Figure 2 (d)).

Putting all together. For each variable $\boldsymbol{x}(i)$ and the sphere for $\boldsymbol{x}(i)$, we create a "hole" for each equation that involves $\boldsymbol{x}(i)$, and then attach a tube. We can have an oriented triangulation of the tubes so that the triangles on the tubes have equal value as the triangles on the sphere. We then connect these tubes properly to encode each given difference and average equation.

4 Reducing Exact Solvers for $\mathcal{DA}$ to $\mathcal{B}_2$ Assuming the Right-hand Side Vector in the Image of the Coefficient Matrix

In this section, we describe a nearly-linear time reduction from instances LE $(\boldsymbol{A}, \boldsymbol{b})$ over $\mathcal{DA}$ to instances LE over $\mathcal{B}_2$, under the assumption that $\boldsymbol{b} \in \mathrm{im}(\boldsymbol{A})$. In Section 5, we will show that the same reduction with a carefully chosen error parameter reduces linear equation approximate problem LEA over $\mathcal{DA}$ to LEA over $\mathcal{B}_2$, assuming $\boldsymbol{b} \in \mathrm{im}(\boldsymbol{A})$. In Section 6, we will slightly modify the reduction to drop the assuming $\boldsymbol{b} \in \mathrm{im}(\boldsymbol{A})$ for LEA.

Recall that an instance LE $(\boldsymbol{A}, \boldsymbol{b})$ over $\mathcal{DA}$ only consists of two types of linear equations:

1. *Difference equation*: $\boldsymbol{x}(i) - \boldsymbol{x}(j) = \boldsymbol{b}(q)$,

2. *Average equation*: $\boldsymbol{x}(i) + \boldsymbol{x}(j) - 2\boldsymbol{x}(k) = 0$.

Suppose LE $(\boldsymbol{A}, \boldsymbol{b})$ has d_1 difference equations and d_2 average equations. Without loss of generality, we reorder all the equations so that the first d_1 equations are difference equations and the rest are average equations.

4.1 Reduction algorithm

Given an instance LE $(\boldsymbol{A}, \boldsymbol{b})$ where $\boldsymbol{A}$ is a $d \times n$ matrix in $\mathcal{DA}$, the following algorithm constructs a 2-complex and a system of linear equations in its boundary operator.

1. For each $i \in [n]$ and variable $\boldsymbol{x}(i)$ in LE $(\boldsymbol{A}, \boldsymbol{b})$, we construct a sphere $\mathcal{S}_i$.

2. For each $q \in [d_1]$ and difference equation $\boldsymbol{x}(i) - \boldsymbol{x}(j) = \boldsymbol{b}(q)$, we add a *loop* α_q with a net flow demand $\boldsymbol{b}(q)$. Then,

   - we add a boundary component $\beta_{q,i}$ on $\mathcal{S}_i$, and a boundary component $\beta_{q,j}$ on $\mathcal{S}_j$;
   - we construct a tube $\mathcal{T}_{q,i}$ with boundary components $\{-\beta_{q,i}, \alpha_q\}$, and a tube $\mathcal{T}_{q,j}$ with boundary components $\{-\beta_{q,j}, -\alpha_q\}$.

   See Figure 3 for an illustration. [3]

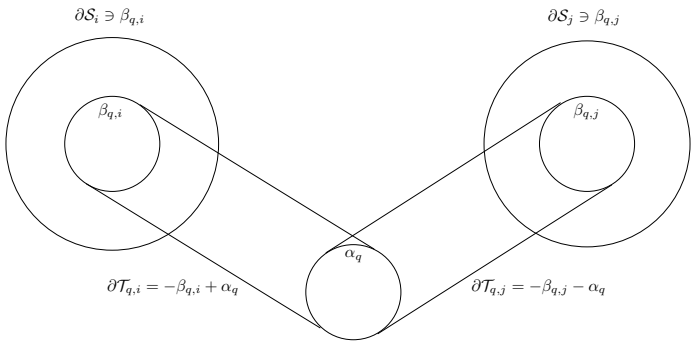


Figure 3: Construction for a difference equation $\boldsymbol{x}(i) - \boldsymbol{x}(j) = \boldsymbol{b}(q)$.

3. For each $q \in \{d_1 + 1, \ldots, d\}$ and average equation $\boldsymbol{x}(i) + \boldsymbol{x}(j) - 2\boldsymbol{x}(k) = 0$, we add a loop α_q with zero net flow demand. Then,

   - we add a boundary component $\beta_{q,i}$ on $\mathcal{S}_i$, a boundary component $\beta_{q,j}$ on $\mathcal{S}_j$, and two boundary components $\beta_{q,k,1}, \beta_{q,k,2}$ on $\mathcal{S}_k$;
   - we construct a tube $\mathcal{T}_{q,i}$ with boundary components $\{-\beta_{q,i}, \alpha_q\}$, a tube $\mathcal{T}_{q,j}$ with boundary components $\{-\beta_{q,j}, \alpha_q\}$, and two tubes $\mathcal{T}_{q,k,1}, \mathcal{T}_{q,k,2}$ with boundary components $\{-\beta_{q,k,1}, -\alpha_q\}$ and $\{-\beta_{q,k,2}, -\alpha_q\}$, respectively. [4]

   See Figure 4 for an illustration.

[3] Note that since the loop α_q has demand $\boldsymbol{b}(q)$, our construction is different from identifying the boundary component α_q of $\mathcal{T}_{q,i}$ and the boundary component $-\alpha_q$ of $\mathcal{T}_{q,j}$.

[4] In the rest of the paper, we introduce a third element $* \in \{1, 2\}$ in the subscript of $\beta_{q,k,*}$, which is activated only when $\boldsymbol{A}(q, k) = -2$.

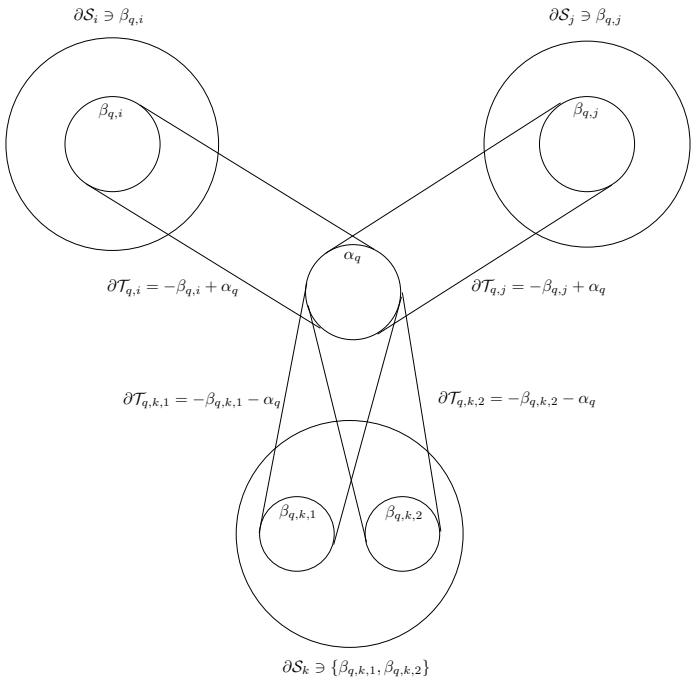


Figure 4: Construction for an average equation $\boldsymbol{x}(i) + \boldsymbol{x}(j) - 2\boldsymbol{x}(k) = 0$.

4. We triangulate the continuous topological space using $O(\text{nnz}(\boldsymbol{A}))$ triangles and get a 2-complex $\mathcal{K}$. Each edge on the loop α_q, referred to as a *boundary edge*, has net demand $\boldsymbol{b}(q)$; each other edge, referred to as an *interior edge*, has net demand 0. Let ∂_2 be the boundary operator of $\mathcal{K}$ and γ be the vector of net flow demands. We return LE (∂_2, γ).

By our construction, each boundary edge on α_q corresponds to the qth equation in $(\boldsymbol{A}, \boldsymbol{b})$. Each interior edge identifies the values of the two triangles that share this edge. In particular, if $\partial_2 \boldsymbol{f} = \gamma$, then $\boldsymbol{f}$ has equal value on the triangles belonging to the same $\mathcal{K}_i$.

To map a solution $\boldsymbol{f}$ to LE $(\boldsymbol{A}, \boldsymbol{b})$ to a solution $\boldsymbol{x}$ to LE $(\boldsymbol{A}, \boldsymbol{b})$, we choose an arbitrary triangle $\Delta_i \in \mathcal{S}_i$ as the *central triangle* for i and set $\boldsymbol{x}(i) = \boldsymbol{f}(\Delta_i)$.

Pseudo-codes are given in Algorithm 1 REDUCE$\mathcal{DA}$TO$\mathcal{B}_2(\boldsymbol{A}, \boldsymbol{b})$ [5] and Algorithm 2 MAPSOLN$\mathcal{B}_2$TO$\mathcal{DA}$.

[5] There is a slight presentation difference between the procedure description and the pseudo-code of Algorithm 1. In the procedure description, to better deliver the reduction intuition, we separate the construction of a continuous topological structure and the process of triangulating it into a 2-complex; when comes to implementation, the triangulation can be embedded into the process of construction.

Algorithm 1: ReduceDAToB2

Input: an instance LE $(\boldsymbol{A}, \boldsymbol{b})$ where $\boldsymbol{A} \in \mathcal{DA}$ is a $d \times n$ matrix and $\boldsymbol{b} \in \mathbb{R}^d$

Output: $(\partial_2, \gamma, \boldsymbol{\Delta}^c)$ where $\partial_2 \in \mathcal{B}_2$ is an $m \times t$ matrix, $\gamma \in \mathbb{R}^m$, and $\boldsymbol{\Delta}^c$ is a set of n triangles

Initialize $\boldsymbol{\Delta}^c = \emptyset$;
for *each* $1 \le i \le n$ **do**
  Initialize $\boldsymbol{s}_i = \emptyset$; `// the set of equation indices involving` $\boldsymbol{x}(i)$
  for *each* $1 \le q \le d$ **do**
    if $\boldsymbol{A}(q,i) \neq 0$ **then**
      $\boldsymbol{s}_i = \boldsymbol{s}_i \cup \{q\}$;
      $(\boldsymbol{T}_{q,i}, E_{q,i}, T_{q,i}) \leftarrow$ TubeTriangulation$(\boldsymbol{A}(q,i))$
    end
  end
  $(\boldsymbol{S}_i, \tilde{E}_i, \tilde{T}_i) \leftarrow$ SphereTriangulation$(\boldsymbol{A}, \boldsymbol{s}_i)$;
  $\boldsymbol{\Delta}^c = \boldsymbol{\Delta}^c \cup \{\Delta_i\}$, where Δ_i is an arbitrary triangle in $\tilde{T}_i$;
  $(\boldsymbol{K}_i, E_i, T_i) \leftarrow$ Merge$((\boldsymbol{S}_i, \tilde{E}_i, \tilde{T}_i), \{(\boldsymbol{T}_{s,i}, E_{s,i}, T_{s,i})\}_{s \in \boldsymbol{s}_i})$;
end
$(\partial_2, E, T) \leftarrow$Merge$(\{(\boldsymbol{K}_i, E_i, T_i)\}_{i \in [n]})$;
`/* construct` γ `*/`
Initialize $\gamma = \mathbf{0}_{|E|}$;
for $1 \le q \le d$ **do**
  $\gamma([v_q^1, v_q^2]) = \gamma([v_q^2, v_q^3]) = \gamma([v_q^3, v_q^1]) = \boldsymbol{b}(q)$
end
return $(\partial_2, \gamma, \boldsymbol{\Delta}^c)$

Algorithm 2: MapSolnB2toDA

Input: a tuple $(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{f}, \boldsymbol{\Delta}^c)$, where $\boldsymbol{A} \in \mathcal{DA}$ is a $d \times n$ matrix, $\boldsymbol{b} \in \mathbb{R}^d$, $\boldsymbol{f} \in \mathbb{R}^t$, $\boldsymbol{\Delta}^c$ is the set of n central triangles

Output: a vector $\boldsymbol{x} \in \mathbb{R}^n$

if $\boldsymbol{A}^\top \boldsymbol{b} = 0$ **then**
  return $\boldsymbol{x} = \mathbf{0}$
else
  for $1 \le i \le n$ **do**
    Set $\boldsymbol{x}(i) = \boldsymbol{f}(\Delta_i)$, where Δ_i is the ith central triangle in $\boldsymbol{\Delta}^c$
  end
  return $\boldsymbol{x}$
end

Oriented triangulation for punctured spheres. We denote the number of boundary components in sphere $\mathcal{S}_i$ by b_i, which equals to $\sum_{q=1}^{d} |\boldsymbol{A}(q,i)|$ by construction. Let $\tilde{t}_i$ and $\tilde{m}_i$ be the number of required triangles and edges on $\mathcal{S}_i$, respectively. We triangulate $\mathcal{S}_i$ as illustrated in Figure 5. Algorithm 3 gives the pseudocode for SphereTriangulation.

- If $b_i = 1$, then $\tilde{t}_i = 1, \tilde{m}_i = 3$ trivially, and we orient the triangle clockwise, which gives a clockwise induced orientation for edges on boundary.

- If $b_i = 2$, we subdivide the triangle in the case of $b_i = 1$ by adding 6 edges between vertices of the inner and the outer boundaries, i.e., a triangulated annulus. Then we get $\tilde{t}_i = 6, \tilde{m}_i = 12$, and we orient all triangles clockwise, which gives a clockwise induced orientation for edges on the outer boundary and anti-clockwise for edges on the inner boundary.
- If $b_i = 3$, we subdivide the rightmost triangle in the case of $b_i = 2$ with the same method. Then we get $\tilde{t}_i = 11, \tilde{m}_i = 21$, and we orient all triangles clockwise, which gives a clockwise induced orientation for edges on the outer boundary and anti-clockwise for edges on all the inner boundaries.[6]
- By induction, we have

$$\tilde{t}_i = 5b_i - 4, \qquad \tilde{m}_i = 9b_i - 6, \qquad \text{for } b_i \geq 1. \tag{2}$$

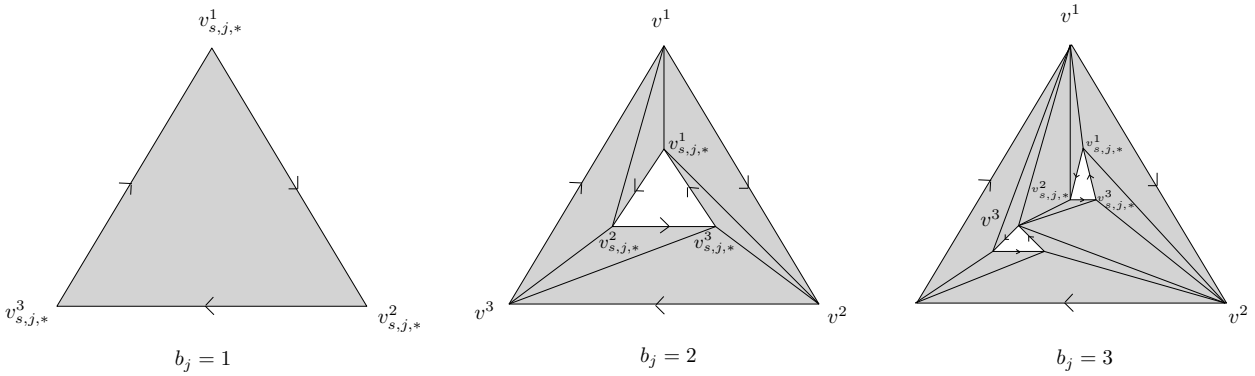


Figure 5: Oriented triangulation of punctured spheres.

Remark.

1. With this triangulation method, all boundary components are composed of 3 edges.
2. Clockwise or anti-clockwise induced orientation for edges on boundaries is only distinguished here for the purpose of picturization. There is no need to make such a distinction in implementation, and we can regard all boundary components oriented anti-clockwise.

Oriented triangulation for tubes. A tube is defined by two boundary components. By construction, for every tube connected to $\mathcal{S}_j$, one of the two boundary components is always $-\beta_{q,j,*}$, and the other one can be $\pm\alpha_q$, whose orientation is consistent with the sign of entry $\boldsymbol{A}(q, j)$. Therefore, there are two possibilities of boundary component combinations, and the corresponding oriented triangulations are shown in Figure 6. Algorithm 4 gives the pseudocode for TubeTriangulation.

- If $\boldsymbol{A}(q, j) > 0$, then the two boundary components have opposite orientations: $-\beta_{q,j,*} = [v^1_{q,j,*}, v^3_{q,j,*}, v^2_{q,j,*}]$ and $\alpha_q = [v^1_q, v^2_q, v^3_q]$. We triangulate by matching $v^1_{q,j,*}$ to v^1_q, $v^2_{q,j,*}$ to v^2_q, and $v^3_{q,j,*}$ to v^3_q.

[6]The orientation of the outer boundary is also anti-clockwise if we look at it from the other side. Simply speaking, by clockwise orienting all triangles, the induced orientation for edges on all boundary components are anti-clockwise if we look at each boundary component from an appropriate side.

[8]We simplify the pseudocode without iterating over the subscript $*$. If $\boldsymbol{A}(s, i) = -2$ such that the subscript $*$ is activated, we repeat the corresponding part twice.

[8]We abuse the notations to let $[v_i, v_j]$ also denote the indicator vector of edge $[v_i, v_j]$.

Algorithm 3: SphereTriangulation

Input: a tuple $(\boldsymbol{A}, \boldsymbol{s}_j)$, where $\boldsymbol{s}_j$ is the set of row indices such that the jth column of $\boldsymbol{A}$ has nonzero entries

Output: a tuple $(\boldsymbol{S}_j, \tilde{E}_j, \tilde{T}_j)$, where $\boldsymbol{S}_j$ is the boundary matrix of a triangulated punctured sphere, and $\tilde{E}_j, \tilde{T}_j$ are the sets of ordered labels of rows and columns of $\boldsymbol{S}_j$, respectively

$s = \boldsymbol{s}_j[1]$; `// the first element in` $\boldsymbol{s}_j$

Triangulate the boundary component: $\beta_{s,j,*} = [v^1_{s,j,*}, v^2_{s,j,*}, v^3_{s,j,*}]$;

Construct an edge set: $\tilde{E}_j = \left\{[v^1_{s,j,*}, v^2_{s,j,*}], [v^2_{s,j,*}, v^3_{s,j,*}], [v^3_{s,j,*}, v^1_{s,j,*}]\right\}$;

Construct a clockwise-oriented triangle set: $\tilde{T}_j = \left\{[v^1_{s,j,*}, v^2_{s,j,*}, v^3_{s,j,*}]\right\}$;

Let $v^1 = v^1_{s,j,*}, v^2 = v^2_{s,j,*}, v^3 = v^3_{s,j,*}$; `// the triangle for subdivision`

for $s \in \boldsymbol{s}_j[2:\ ]$[7] **do**

  `/* refer to Figure 5 */`

  Triangulate the boundary component: $\beta_{s,j,*} = [v^1_{s,j,*}, v^2_{s,j,*}, v^3_{s,j,*}]$;

  Append 6 edges to the edge set:
  $\tilde{E}_j = \tilde{E}_j \cup \left\{[v^1_{s,j,*}, v^2_{s,j,*}], [v^2_{s,j,*}, v^3_{s,j,*}], [v^3_{s,j,*}, v^1_{s,j,*}],\ \ [v^1, v^1_{s,j,*}], [v^1, v^2_{s,j,*}], \dots\right\}$; `// the orientation of edges on boundaries should be consistent with the boundary orientation; the orientation of other edges can be set arbitrarily.`

  Delete the triangle for subdivision from the triangle set, and append 6 new triangles:
  $\tilde{T}_i = \tilde{T}_i \backslash \left\{[v^1, v^2, v^3]\right\} \cup \left\{[v^1, v^2, v^1_{s,j,*}], [v^2, v^3_{s,j,*}, v^1_{s,j,*}], \dots\right\}$;

  Let $v^1 = v^1, v^2 = v^2, v^3 = v^2_{s,j,*}$; `// update the triangle for subdivision`

end

Initialize $\boldsymbol{S}_j = \boldsymbol{0}_{\tilde{m}_i \times \tilde{t}_i}$;

for $\Delta \in \tilde{T}_j$, *where we denote* $\Delta = [v_0, v_1, v_2]$ **do**

  $\boldsymbol{S}_j(:, \Delta) \leftarrow [v_1, v_2] - [v_0, v_2] + [v_0, v_1]$[8]

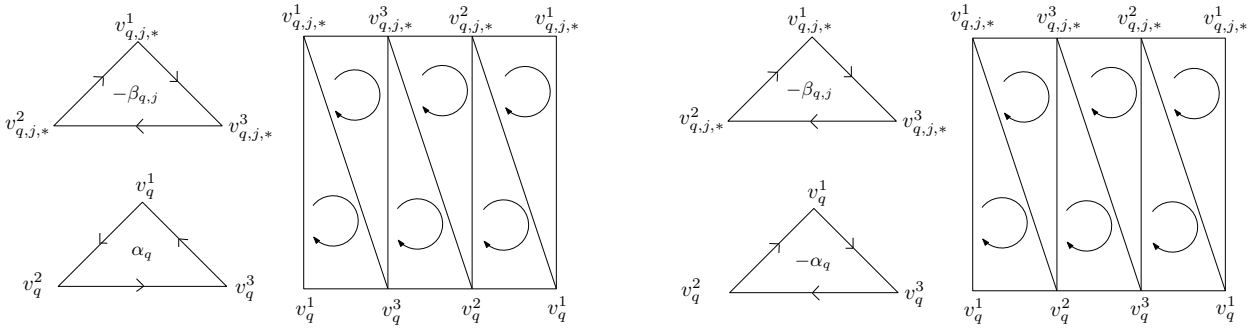
end

return $(\boldsymbol{S}_j, \tilde{E}_j, \tilde{T}_j)$

- If $\boldsymbol{A}(q, j) < 0$, then the two boundary components have identical orientations: $-\beta_{q,j,*} = [v^1_{q,j,*}, v^3_{q,j,*}, v^2_{q,j,*}]$ and $-\alpha_q = [v^1_q, v^3_q, v^2_q]$. We triangulate by matching $v^1_{q,j,*}$ to v^1_q, $v^3_{q,j,*}$ to v^2_q, and $v^2_{q,j,*}$ to v^3_q.

In either case, only 6 triangles and 12 edges are required for an oriented triangulation of any tube $\mathcal{T}_{q,j,*}$.

Remark. If multiple tubes are connected to a single boundary component, to avoid the intersection of tubes before attaching the boundary component, a higher-dimensional space is required.



(a) $\mathbf{A}(q,j) > 0$ (b) $\mathbf{A}(q,j) < 0$

Figure 6: Oriented triangulation of tubes with opposite or identical boundary orientations.

Algorithm 4: TubeTriangulation

Input: a nonzero entry $\boldsymbol{A}(q,j)$

Output: a tuple $(\boldsymbol{T}_{q,j}, E_{q,j}, T_{q,j})$, where $\boldsymbol{T}_{q,j}$ is the boundary matrix of triangulated tube(s), and $E_{q,j}, T_{q,j}$ are the sets of ordered labels of rows and columns of $\boldsymbol{T}_{q,j}$, respectively

if $\boldsymbol{A}(q,j) > 0$ **then**
  `/* refer to part (a) of Figure 6 */`
  Triangulate boundary components: $\beta_{q,j,*} = [v^1_{q,j,*}, v^2_{q,j,*}, v^3_{q,j,*}]$, $\alpha_q = [v^1_q, v^2_q, v^q_3]$;
  Construct an edge set:
  $E_{q,j} = \Big\{[v^1_{q,j,*}, v^2_{q,j,*}], [v^2_{q,j,*}, v^3_{q,j,*}], [v^3_{q,j,*}, v^1_{q,j,*}], \quad [v^1_q, v^2_q], [v^2_q, v^3_q], [v^3_q, v^1_q], \dots\Big\}$;
  Construct a clockwise-oriented triangle set: $T_{q,j} = \Big\{[v^1_{q,j,*}, v^3_q, v^1_q], \dots, [v^2_{q,j,*}, v^1_{q,j,*}, v^1_q]\Big\}$;
  Initialize $\boldsymbol{T}_{q,j} = \mathbf{0}_{12\times 6}$;
end
if $\boldsymbol{A}(q,j) < 0$ **then**
  `/* refer to part (b) of Figure 6 */`
  (similar to the previous case);
  if $\boldsymbol{A}(q,j) = -2$ **then**
    $E_{q,j} = E_{q,j,1} \cup E_{q,j,2}$, $T_{q,j} = T_{q,j,1} \cup T_{q,j,2}$;
    Initialize $\boldsymbol{T}_{q,j} = \mathbf{0}_{24\times 12}$;
  end
end
for $\Delta \in T_{q,j}$, *where we denote* $\Delta = [v_0, v_1, v_2]$ **do**
  $\boldsymbol{T}_{q,j}(:, \Delta) \leftarrow [v_1, v_2] - [v_0, v_2] + [v_0, v_1]$
end
return $(\boldsymbol{T}_{q,j}, E_{q,j}, T_{q,j})$

4.2 Notations

We introduce several notions and corresponding notations about the constructed 2-complex. These notions and notations will be used in the rest of the paper.

Algorithm 5: MERGE

Input: a set of matrices with ordered row and column labels $\{(\boldsymbol{M}_1, E_1, T_1), \ldots, (\boldsymbol{M}_k, E_k, T_k)\}$

Output: a tuple $(\boldsymbol{M}, E, T)$, where $\boldsymbol{M}$ is the merged matrix, E, T are the sets of ordered labels of rows and columns of $\boldsymbol{M}$, respectively

$\tilde{\boldsymbol{M}} = \mathrm{diag}(\boldsymbol{M}_i)_{i=1}^k$, $\tilde{E} = \mathrm{concatenate}(E_1, \cdots, E_k)$, $T = \mathrm{concatenate}(T_1, \cdots, T_k)$;

Detect duplicates of $\tilde{E}$, and add up corresponding rows in $\tilde{\boldsymbol{M}}$ into one row;

$\boldsymbol{M} \leftarrow \tilde{\boldsymbol{M}}$ after combination; $E \leftarrow \tilde{E}$ after removing duplicates;

return $(\boldsymbol{M}, E, T)$

For each $i \in [n]$, let $\mathcal{K}_i$ be the the union of the triangulated $\mathcal{S}_i$ and the triangulated tubes that are connected to $\mathcal{S}_i$, which we refer to as the ith *complex group*; let t_i be the number of triangles in $\mathcal{K}_i$; let m_i be the number of the interior edges in $\mathcal{K}_i$.

According to our triangulation, each loop α_q has three boundary edges, denoted by $\alpha_q^1 = [v_q^1, v_q^2]$, $\alpha_q^2 = [v_q^2, v_q^3]$, $\alpha_q^3 = [v_q^3, v_q^1]$. A triangle containing a boundary edge is called a *boundary triangle*. For each boundary edge α_q^r, where $q \in [d_1]$ and $r \in [3]$, corresponding to equation $\boldsymbol{x}(i) - \boldsymbol{x}(j) = \boldsymbol{b}(q)$, we denote the boundary triangles by $\Delta_{q,i,1}^r, \Delta_{q,j,1}^r$ where $\Delta_{q,i,1}^r \in \mathcal{T}_{q,i}$, $\Delta_{q,j,1}^r \in \mathcal{T}_{q,j}$; for each boundary edge α_q^r, where $q \in [d_1 + 1 : d_2]$ and $r \in [3]$, corresponding to equation $\boldsymbol{x}(i) + \boldsymbol{x}(j) = 2\boldsymbol{x}(k)$, we denote the boundary triangles by $\Delta_{q,i,1}^r, \Delta_{q,j,1}^r, \Delta_{q,k,1}^r, \Delta_{q,k,2}^r$ where $\Delta_{q,i,1}^r \in \mathcal{T}_{q,i}$, $\Delta_{q,j,1}^r \in \mathcal{T}_{q,j}$, $\Delta_{q,k,1}^r, \Delta_{q,k,2}^r \in \mathcal{T}_{q,k}$.

Given any two triangles $\Delta, \Delta' \in \mathcal{K}$, a *triangle path* from Δ to Δ' is an ordered collection of triangles $\mathcal{P} = [\Delta^{(0)} = \Delta, \Delta^{(1)}, \ldots, \Delta^{(l)} = \Delta']$ such that every neighboring triangles share an edge. The length of $\mathcal{P}$ is l. A triangle path can also be defined by an ordered collection of edges $\mathcal{P} = [e^{(1)}, \ldots, e^{(l)}]$, where $e^{(i)}$ denotes the edge shared by $\Delta^{(i-1)}$ and $\Delta^{(i)}$, for $i \in [l]$.

4.3 Algorithm runtime and problem size

In this section, we show that Algorithm 1 REDUCE$\mathcal{DA}$TO$\mathcal{B}_2$ and Algorithm 2 MAPSOLN$\mathcal{B}_2$TO$\mathcal{DA}$ both run in linear time, and Algorithm 1 constructs a 2-complex whose size is linear in the number of nonzeros in the input linear equations.

Lemma 4.1 (Runtime)**.** *Given a difference-average instance* LE $(\boldsymbol{A}, \boldsymbol{b})$ *where* $\boldsymbol{A} \in \mathbb{R}^{d \times n}$*, Algorithm* REDUCE$\mathcal{DA}TO\mathcal{B}_2(\boldsymbol{A}, \boldsymbol{b})$ *returns* $(\partial_2, \gamma, \boldsymbol{\Delta}^c)$ *in time* $O(\mathrm{nnz}(\boldsymbol{A}))$*. Given a solution* $\boldsymbol{f}$ *to* LE (∂_2, γ)*, Algorithm* MAPSOLN$\mathcal{B}_2$TO$\mathcal{DA}(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{f}, \boldsymbol{\Delta}^c)$ *returns* $\boldsymbol{x}$ *in time* $O(n)$*.*

Proof. For reduction, Algorithm REDUCE$\mathcal{DA}$TO$\mathcal{B}_2(\boldsymbol{A}, \boldsymbol{b})$ calls the subroutine TUBETRIANGULATION for $\|\boldsymbol{A}\|_1$ times, and the subroutine SPHERETRIANGULATION for n times. Algorithm TUBETRIANGULATION runs in time $O(1)$ since the there are a constant number of triangles in a tube; and Algorithm SPHERETRIANGULATION runs in time $O(\|\boldsymbol{A}(:, j)\|_1)$ for the jth call, $j \in [n]$. In addition, the total runtime of MERGE can be bounded by $O(d)$ since only those rows labeled by boundary edges need to be merged. Putting all together, the total runtime of REDUCE$\mathcal{DA}$TO$\mathcal{B}_2(\boldsymbol{A}, \boldsymbol{b})$ is $O\left(\|\boldsymbol{A}\|_1 + \sum_{j \in [n]} \|\boldsymbol{A}(:, j)\|_1 + d\right) \leq O(\mathrm{nnz}(\boldsymbol{A}))$, where we use the fact $\|\boldsymbol{A}\|_{\max} = 2$.

For solution mapping, the runtime of Algorithm MAPSOLN$\mathcal{B}_2$TO$\mathcal{DA}$ is obvious. □

Lemma 4.2 (Size of ∂_2)**.** *Given a difference-average instance* LE $(\boldsymbol{A}, \boldsymbol{b})$*, let* $(\partial_2, \gamma, \boldsymbol{\Delta}^c)$ *be returned by* REDUCE$\mathcal{DA}$TO$\mathcal{B}_2(\boldsymbol{A}, \boldsymbol{b})$*. Suppose* $\partial_2 \in \mathbb{R}^{m \times t}$*. Then,*

- $t \leq 22\,\mathrm{nnz}(\boldsymbol{A})$*;*

- $m \le 33\,\mathrm{nnz}(\boldsymbol{A})$;
- $\mathrm{nnz}(\partial_2) \le 66\,\mathrm{nnz}(\boldsymbol{A})$.

Proof. We first compute the total number of triangles in the constructed 2-complex $\mathcal{K}$. For sphere $\mathcal{S}_j$, we have $\tilde{t}_j = 5b_j - 4$ triangles by (2), where $b_j = \sum_{i\in[d]} |\boldsymbol{A}(i,j)|$. Therefore, the number of triangles of all spheres is

$$\sum_{j=1}^{n} \tilde{t}_j = \sum_{j=1}^{n}(5\sum_{i\in[d]} |\boldsymbol{A}(i,j)| - 4) = 5\,\|\boldsymbol{A}\|_1 - 4n.$$

Moreover, each boundary component on spheres corresponds to a tube, and each tube has 6 triangles. Hence, the number of triangles of all tubes is $6\,\|\boldsymbol{A}\|_1$. Putting spheres and tubes together, we get

$$t = 11\,\|\boldsymbol{A}\|_1 - 4n \le 22\,\mathrm{nnz}(\boldsymbol{A}),$$

where the last inequality is because entries of $\boldsymbol{A}$ are bounded by 2.

Next, we compute the total number of edges in $\mathcal{K}$. By construction, each triangle has 3 incident edges and each edge is shared by a constant number of triangles (2 for interior edges, and 4 for boundary edges). Thus, we have

$$m \le 1.5t \le 33\,\mathrm{nnz}(\boldsymbol{A}).$$

Since each column of ∂_2 has exactly 3 nonzero entries, we have

$$\mathrm{nnz}(\partial_2) = 3t \le 66\,\mathrm{nnz}(\boldsymbol{A}).$$

□

4.4 Relation between exact solutions

In this section, we show that Algorithm 1 ReduceDAToB2 and Algorithm 2 MapSolnB2toDA reduce instances LE $(\boldsymbol{A}, \boldsymbol{b})$ over $\mathcal{DA}$ to instances LE over $\mathcal{B}_2$, under the assumption that $\boldsymbol{b}$ is in the image of $\boldsymbol{A}$.

Given LE $(\boldsymbol{A}, \boldsymbol{b})$ where $\boldsymbol{A}$ is a $d \times n$ matrix in $\mathcal{DA}$, let $(\partial_2, \gamma, \boldsymbol{\Delta}^c)$ be returned by Algorithm ReduceDAToB2$(\boldsymbol{A}, \boldsymbol{b})$, and let $\mathcal{K}$ be the 2-complex constructed in Algorithm ReduceDAToB2$(\boldsymbol{A}, \boldsymbol{b})$ and the boundary operator of $\mathcal{K}$ is ∂_2. Let $\boldsymbol{f}$ be a solution to LE $(\boldsymbol{A}, \boldsymbol{b})$.

To simplify the analysis, we reorder the columns and rows of ∂_2. The columns $[1 : t_1]$ of ∂_2 correspond to the triangles in $\mathcal{K}_1$, the columns $[t_1 + 1 : t_2]$ correspond to the triangles in $\mathcal{K}_2$, and so on. Then, $\boldsymbol{f}$ can be written as

$$\boldsymbol{f} = \begin{bmatrix} \boldsymbol{f}_1 \\ \vdots \\ \boldsymbol{f}_n \end{bmatrix}, \qquad \text{where } \boldsymbol{f}_i \in \mathbb{R}^{t_i}, \forall i \in [n].$$

The rows of ∂_2 and the entries of γ are:

$$\partial_2 = \left[\begin{array}{ccc} & \boldsymbol{B}_1 & \\ & \vdots & \\ & \boldsymbol{B}_d & \\ \hline \boldsymbol{M}_1 & & \\ & \ddots & \\ & & \boldsymbol{M}_n \end{array}\right], \qquad \gamma = \left[\begin{array}{c} \boldsymbol{b}(1)\mathbf{1}_3 \\ \vdots \\ \boldsymbol{b}(d)\mathbf{1}_3 \\ \hline \mathbf{0}_{m_1} \\ \vdots \\ \mathbf{0}_{m_n} \end{array}\right], \tag{3}$$

Here, each submatrix $\boldsymbol{B}_q \in \{0, \pm 1\}^{3\times t}$ corresponds to the three boundary edges $\{\alpha_q^1, \alpha_q^2, \alpha_q^3\}$; each submatrix $\boldsymbol{M}_i \in \{0, \pm 1\}^{m_i \times t_i}$ corresponds to all the interior edges in $\mathcal{K}_i$. Interior edges in $\mathcal{K}_i$ and those in $\mathcal{K}_j$ do not share endpoints if $i \neq j$. Let $\boldsymbol{M} = \mathrm{diag}(\boldsymbol{M}_1, \boldsymbol{M}_2, \ldots, \boldsymbol{M}_n)$.

Claim 4.3. *For each $i \in [n]$, $\boldsymbol{f}_i = \alpha \mathbf{1}_{t_i}$ for some $\alpha \in \mathbb{R}$.*

Proof. For each $i \in [n]$, we have $\boldsymbol{M}_i \boldsymbol{f}_i = \mathbf{0}$. This means that for any two triangles Δ, Δ' in $\mathcal{K}_i$ sharing an interior edge, we have $\boldsymbol{f}_i(\Delta) = \boldsymbol{f}_i(\Delta')$. By our construction of $\mathcal{K}_i$, for any two triangles Δ, Δ' in $\mathcal{K}_i$, there exists a triangle path connecting Δ and Δ'. The values of $\boldsymbol{f}_i$ at the triangles in this triangle path are equal; in particular, $\boldsymbol{f}_i(\Delta) = \boldsymbol{f}_i(\Delta')$. Thus, the values of $\boldsymbol{f}_i$ at all the triangles in $\mathcal{K}_i$ are equal, that is, $\boldsymbol{f}_i = \alpha \mathbf{1}_{t_i}$ for some $\alpha \in \mathbb{R}$. □

Lemma 4.4 (Exact solvers for feasible linear equations)**.** *Given a difference-average instance* LE $(\boldsymbol{A}, \boldsymbol{b})$ *where* $\boldsymbol{b} \in \mathrm{im}(\boldsymbol{A})$*, let* $(\partial_2, \gamma, \boldsymbol{\Delta}^c)$ *be returned by* ReduceDATOB2$(\boldsymbol{A}, \boldsymbol{b})$*, and let* $\boldsymbol{f}$ *be a solution to* LE (∂_2, γ)*. Then,* $\boldsymbol{x} \leftarrow$MapSolnB2toDA$(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{f}, \boldsymbol{\Delta}^c)$ *is a solution to* LE $(\boldsymbol{A}, \boldsymbol{b})$*.*

Proof. By Claim 4.3, we can write $\boldsymbol{f}$ as

$$\boldsymbol{f} = \begin{bmatrix} \alpha_1 \mathbf{1}_{t_1} \\ \alpha_2 \mathbf{1}_{t_2} \\ \vdots \\ \alpha_n \mathbf{1}_{t_n} \end{bmatrix}, \quad \text{where } \alpha_1, \ldots, \alpha_n \in \mathbb{R}$$

According to Algorithm MapSolnB2toDA$(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{f}, \boldsymbol{\Delta}^c)$, for each $i \in [n]$, $\boldsymbol{x}(i) = \alpha_i$.

Our goal is to show $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$.

- For each difference equation in LE $(\boldsymbol{A}, \boldsymbol{b})$, say $\boldsymbol{x}(i) - \boldsymbol{x}(j) = \boldsymbol{b}(q)$, we look at the equations in LE (∂_2, γ) related to $\boldsymbol{B}_q$:

  $$\boldsymbol{f}_i(\Delta_{q,i}^r) - \boldsymbol{f}_j(\Delta_{q,j}^r) = \boldsymbol{b}(q), \ \forall r \in \{1, 2, 3\}.$$

  thus $\boldsymbol{x}(i) - \boldsymbol{x}(j) = \boldsymbol{b}(q)$ holds.

- For each average equation in LE $(\boldsymbol{A}, \boldsymbol{b})$, say $\boldsymbol{x}(i) + \boldsymbol{x}(j) - 2\boldsymbol{x}(k) = \boldsymbol{b}(q) = 0$, we look at the equations in LE (∂_2, γ) related to $\boldsymbol{B}_q$:

  $$\boldsymbol{f}_i(\Delta_{q,i}^r) + \boldsymbol{f}_j(\Delta_{q,j}^r) - \boldsymbol{f}_k(\Delta_{q,k,1}^r) - \boldsymbol{f}_k(\Delta_{q,k,2}^r) = 0, \ \forall r \in \{1, 2, 3\}.$$

  thus $\boldsymbol{x}(i) + \boldsymbol{x}(j) - 2\boldsymbol{x}(k) = 0$ holds.

□

5 Reducing Approximate Solvers for $\mathcal{DA}$ to $\mathcal{B}_2$ Assuming the Right-hand Side Vector in the Image of the Coefficient Matrix

In this section, we show that Algorithm 1 ReduceDAToB2 and Algorithm 2 MapSolnB2toDA reduce instances LEA $(\boldsymbol{A}, \boldsymbol{b}, \epsilon)$ over $\mathcal{DA}$ to instances LEA over $\mathcal{B}_2$, under the assumption that $\boldsymbol{b}$ is in the image of $\boldsymbol{A}$.

In Section 5.1, we do error analysis; in Section 5.2 we bound the condition number.

5.1 Relation between approximate solutions

Lemma 5.1 (Approximate solvers for feasible linear equations)**.** *Given a difference-average instance* LEA $(\boldsymbol{A}, \boldsymbol{b}, \epsilon^{DA})$ *where* $\boldsymbol{b} \in \mathrm{im}(\boldsymbol{A})$*, let* $(\partial_2, \gamma, \boldsymbol{\Delta}^c)$ *be returned by* REDUCE$\mathcal{DA}$TO$\mathcal{B}_2(\boldsymbol{A}, \boldsymbol{b})$*. Suppose* $\boldsymbol{f}$ *is a solution to* LEA $(\partial_2, \gamma, \epsilon^{B_2})$ *where*

$$\epsilon^{B_2} \leq \frac{\epsilon^{DA}}{42\,\mathrm{nnz}(\boldsymbol{A})},$$

and $\boldsymbol{x}$ *is returned by* MAPSOLN$\mathcal{B}_2$TO$\mathcal{DA}(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{f}, \boldsymbol{\Delta}^c)$*. Then,* $\boldsymbol{x}$ *is a solution to* LEA $(\boldsymbol{A}, \boldsymbol{b}, \epsilon^{DA})$*.*

Proof. Since $\boldsymbol{b} \in \mathrm{im}(\boldsymbol{A})$, we have $\|\partial_2 \boldsymbol{f} - \gamma\|_\infty \leq \|\partial_2 \boldsymbol{f} - \gamma\|_2 \leq \epsilon^{B_2} \|\gamma\|_2$.

We claim

$$\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_\infty \leq 24\,\mathrm{nnz}(\boldsymbol{A})^{1/2} \cdot \epsilon^{B_2} \|\gamma\|_2 . \tag{4}$$

Then,

$$\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2 \leq 24\,\mathrm{nnz}(\boldsymbol{A}) \cdot \epsilon^{B_2} \|\gamma\|_2 .$$

By Algorithm REDUCE$\mathcal{DA}$TO$\mathcal{B}_2(\boldsymbol{A}, \boldsymbol{b})$, $\|\gamma\|_2 \leq \sqrt{3}\,\|\boldsymbol{b}\|_2$. Thus,

$$\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2 \leq 42\,\mathrm{nnz}(\boldsymbol{A})^{1/2} \cdot \epsilon^{B_2} \|\boldsymbol{b}\|_2 \leq \epsilon^{DA} \|\boldsymbol{b}\|_2 ,$$

that is, $\boldsymbol{x}$ is a solution to LEA $(\boldsymbol{A}, \boldsymbol{b}, \epsilon^{DA})$.

To prove Eq. (4), consider an arbitrary equation in LEA $(\boldsymbol{A}, \boldsymbol{b}, \epsilon^{DA})$, say $a_i \boldsymbol{x}(i) + a_j \boldsymbol{x}(j) + a_k \boldsymbol{x}(k) = \boldsymbol{b}(q)$ where $a_i, a_j, a_k \in \{-2, -1, 0, 1\}$. According to Algorithm MAPSOLN$\mathcal{B}_2$TO$\mathcal{DA}(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{f}, \boldsymbol{\Delta}^c)$, for each $l \in \{i, j, k\}$, $\boldsymbol{x}(l) = \boldsymbol{f}(\Delta_l)$ where $\Delta_l \in \mathcal{K}_l$ is the lth central triangle in $\boldsymbol{\Delta}^c$. Then,

$$a_i \boldsymbol{x}(i) + a_j \boldsymbol{x}(j) + a_k \boldsymbol{x}(k) = a_i \boldsymbol{f}(\Delta_i) + a_j \boldsymbol{f}(\Delta_j) + a_k \boldsymbol{f}(\Delta_k).$$

Note that the equation in LEA $(\partial_2, \gamma, \epsilon^{B_2})$ related to the boundary edge α_q^1, shared by triangles $\Delta_{q,i,1}^1, \Delta_{q,j,1}^1$ and $\Delta_{q,k,1}^1, \Delta_{q,k,2}^1$ (if the equation is an average equation), satisfies

$$\left| a_i \boldsymbol{f}(\Delta_{q,i,1}^1) + a_j \boldsymbol{f}(\Delta_{q,j,1}^1) + \frac{1}{2} a_k \left(\boldsymbol{f}(\Delta_{q,k,1}^1) + \boldsymbol{f}(\Delta_{q,k,2}^1) \right) - \boldsymbol{b}(q) \right| \leq \epsilon^{B_2} \|\gamma\|_2 .$$

For each $\Delta \in \{\Delta_{q,i,1}^1, \Delta_{q,j,1}^1, \Delta_{q,k,1}^1, \Delta_{q,k,2}^1\}$, we will replace $\boldsymbol{f}(\Delta)$ with its corresponding central triangle Δ^c. We can find a triangle path connecting Δ and Δ^c, say $\mathcal{P} = [\Delta^{(0)} = \Delta, \ldots, \Delta^{(l_q)} = \Delta^c]$, such that two adjacent triangles $\Delta^{(l)}, \Delta^{(l+1)}$ share an interior edge $e^{(l)}$. Then,

$$\begin{aligned}
|\boldsymbol{f}(\Delta) - \boldsymbol{f}(\Delta^c)| &= \left| \sum_{l=1}^{l_q} \boldsymbol{f}(\Delta^{(l-1)}) - \boldsymbol{f}(\Delta^{(l)}) \right| \leq \sum_{l=1}^{l_q} \left| \boldsymbol{f}(\Delta^{(l-1)}) - \boldsymbol{f}(\Delta^{(l)}) \right| = \sum_{l=1}^{l_q} \left| [\partial_2 \boldsymbol{f}](e^{(l)}) \right| \\
&= \sum_{l=1}^{l_q} \left| [\partial_2 \boldsymbol{f} - \gamma](e^{(l)}) \right| && \text{since } \gamma(e^{(l)}) = 0 \text{ for interior edges} \\
&= \left\| [\partial_2 \boldsymbol{f} - \gamma](e^{(1)} : e^{(l_q)}) \right\|_1 && \text{where the subvector corresponds to } [e^{(1)}, \ldots, e^{(l_q)}] \\
&\leq \sqrt{t_i} \left\| [\partial_2 \boldsymbol{f} - \gamma](e^{(1)} : e^{(l_q)}) \right\|_2 && \text{since } l_q \leq t_i \\
&\leq \sqrt{t_i} \|\partial_2 \boldsymbol{f} - \gamma\|_2 \\
&\leq \sqrt{t_i} \cdot \epsilon^{B_2} \|\gamma\|_2
\end{aligned}$$

Thus,

$$\begin{aligned}
&|a_i\boldsymbol{x}(i) + a_j\boldsymbol{x}(j) + a_k\boldsymbol{x}(k) - \boldsymbol{b}(q)| \\
=&|a_i\boldsymbol{f}(\Delta_i) + a_j\boldsymbol{f}(\Delta_j) + a_k\boldsymbol{f}(\Delta_k) - \boldsymbol{b}(q)| \\
\leq&\underbrace{\left|a_i\boldsymbol{f}(\Delta^1_{q,i,1}) + a_j\boldsymbol{f}(\Delta^1_{q,j,1}) + \frac{1}{2}a_k\left(\boldsymbol{f}(\Delta^1_{q,k,1}) + \boldsymbol{f}(\Delta^1_{q,k,2})\right) - \boldsymbol{b}(q)\right|}_{\leq\epsilon^{B_2}\|\gamma\|_2} \\
&+\underbrace{\left|\boldsymbol{f}(\Delta^1_{q,i,1}) - \boldsymbol{f}(\Delta_i)\right|}_{\leq\sqrt{t_i}\epsilon^{B_2}\|\gamma\|_2} + \underbrace{\left|\boldsymbol{f}(\Delta^1_{q,j,1}) - \boldsymbol{f}(\Delta_j)\right|}_{\leq\sqrt{t_j}\epsilon^{B_2}\|\gamma\|_2} + \underbrace{\left|\boldsymbol{f}(\Delta^1_{q,k,1}) - \boldsymbol{f}(\Delta_k)\right|}_{\leq\sqrt{t_k}\epsilon^{B_2}\|\gamma\|_2} + \underbrace{\left|\boldsymbol{f}(\Delta^1_{q,k,2}) - \boldsymbol{f}(\Delta_k)\right|}_{\leq\sqrt{t_k}\epsilon^{B_2}\|\gamma\|_2} \\
\leq&5\sqrt{t}\cdot\epsilon^{B_2}\|\gamma\|_2 \\
\leq&24\,\mathrm{nnz}(\boldsymbol{A})^{1/2}\cdot\epsilon^{B_2}\|\gamma\|_2 \qquad \text{by Lemma 4.2}
\end{aligned}$$

That is, $\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_\infty \leq 24\,\mathrm{nnz}(\boldsymbol{A})^{1/2}\cdot\epsilon^{B_2}\|\gamma\|_2$. □

5.2 Bounding the condition number of the new matrix

In this section, we show that the condition number of ∂_2 is upper bounded by a polynomial of $\mathrm{nnz}(\boldsymbol{A}), \kappa(\boldsymbol{A})$.

Lemma 5.2 (The condition number of ∂_2)**.** *Given a difference-average instance* $\textsc{lea}(\boldsymbol{A}, \boldsymbol{b}, \epsilon^{DA})$ *where* $\boldsymbol{b} \in \mathrm{im}(\boldsymbol{A})$*, let* $(\partial_2, \gamma, \boldsymbol{\Delta}^c)$ *be returned by* $\textsc{ReduceDAToB}_2(\boldsymbol{A}, \boldsymbol{b})$*. Then,*

$$\kappa(\partial_2) \leq 10^9\,\mathrm{nnz}(\boldsymbol{A})^{9/2}\kappa(\boldsymbol{A})^2.$$

Note that

$$\kappa^2(\partial_2) = \kappa(\partial_2^\top\partial_2) = \frac{\lambda_{\max}(\partial_2^\top\partial_2)}{\lambda_{\min}(\partial_2^\top\partial_2)}.$$

We will upper bound $\lambda_{\max}(\partial_2^\top\partial_2)$ and lower bound $\lambda_{\min}(\partial_2^\top\partial_2)$. Our proof will heavily rely on the Courant-Fischer theorem.

Theorem 5.3 (The Courant-Fischer Theorem)**.** *Let* $\boldsymbol{M}$ *be a symmetric matrix in* $\mathbb{R}^{n\times n}$ *where* $\lambda_{\max}, \lambda_{\min}$ *are its maximum and minimum nonzero eigenvalue, respectively. Then*

$$\lambda_{\max} = \max_{\boldsymbol{x}\neq\boldsymbol{0}}\frac{\boldsymbol{x}^\top\boldsymbol{M}\boldsymbol{x}}{\boldsymbol{x}^\top\boldsymbol{x}}, \qquad \lambda_{\min} = \min_{\boldsymbol{x}\perp\mathrm{null}(\boldsymbol{M}),\boldsymbol{x}\neq\boldsymbol{0}}\frac{\boldsymbol{x}^\top\boldsymbol{M}\boldsymbol{x}}{\boldsymbol{x}^\top\boldsymbol{x}}.$$

5.2.1 The maximum eigenvalue

Lemma 5.4 (The maximum eigenvalue)**.** $\lambda_{\max}(\partial_2^\top\partial_2) \leq 12$.

Proof. By the Courant-Fischer theorem,

$$\lambda_{\max}(\partial_2^\top\partial_2) = \max_{\boldsymbol{f}:\|\boldsymbol{f}\|_2=1}\boldsymbol{f}^\top\partial_2^\top\partial_2\boldsymbol{f}.$$

Then for any $\boldsymbol{f}$ with $\|\boldsymbol{f}\|_2 = 1$,

$$\boldsymbol{f}^\top\partial_2^\top\partial_2\boldsymbol{f} = \sum_{i=1}^m\left(\sum_{\Delta:e_i\in\Delta}\partial_2(e_i,\Delta)\boldsymbol{f}(\Delta)\right)^2 \leq 4\sum_{i=1}^m\sum_{\Delta:e_i\in\Delta}\boldsymbol{f}^2(\Delta) = 12\,\|\boldsymbol{f}\|_2^2 = 12.$$

where the inequality is by the Cauchy-Schwarz inequality. □

5.2.2 The minimum nonzero eigenvalue

We start with proving a relation between the null space of $\boldsymbol{A}$ and that of ∂_2.

Lemma 5.5. *Let*

$$\boldsymbol{H} = \begin{bmatrix} \mathbf{1}_{t_1} & & \\ & \vdots & \\ & & \mathbf{1}_{t_n} \end{bmatrix} \in \mathbb{R}^{t \times n}.$$

Then, $\boldsymbol{H}$ is a bijection from $\mathrm{null}(\boldsymbol{A})$ *to* $\mathrm{null}(\partial_2)$.

Proof. Let $\boldsymbol{x} \in \mathrm{null}(\boldsymbol{A})$. By our construction of ∂_2, we have $\boldsymbol{H}\boldsymbol{x} \in \mathrm{null}(\partial_2)$. For any $\boldsymbol{f} \in \mathrm{null}(\partial_2)$, by the proof of Lemma 4.4, $\boldsymbol{f} = \boldsymbol{H}\boldsymbol{x}$ for some $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{A}\boldsymbol{x} = \mathbf{0}$. □

Lemma 5.6 (The minimium nonzero eigenvalue)**.**

$$\lambda_{\min}(\partial_2^\top \partial_2) \geq \frac{\min\{\lambda_{\min}(\boldsymbol{A}^\top \boldsymbol{A})^2, 1\}}{10^{16} d^7}. \tag{5}$$

Proof. By the Courant-Fischer theorem,

$$\lambda_{\min}(\partial_2^\top \partial_2) = \min_{\substack{\boldsymbol{f} \in \mathbb{R}^t : \boldsymbol{f} \perp \mathrm{null}(\partial_2) \\ \|\boldsymbol{f}\|_2 = 1}} \boldsymbol{f}^\top \partial_2^\top \partial_2 \boldsymbol{f}.$$

Let

$$C = \frac{\min\{\lambda_{\min}(\boldsymbol{A}^\top \boldsymbol{A}), 1\}}{10^6 d^{2.5}}.$$

We will exhaust all the vectors in $\{\boldsymbol{f} : \boldsymbol{f} \perp \mathrm{null}(\partial_2), \|\boldsymbol{f}\|_2 = 1\}$ by the following two cases.

Case 1. Suppose there exists $i \in [n]$ such that $\mathcal{K}_i$ contains two triangles Δ, Δ' satisfying $|\boldsymbol{f}(\Delta) - \boldsymbol{f}(\Delta')| \geq C$. Consider a triangle path in $\mathcal{K}_i$ connecting Δ and Δ', say $[\Delta^{(0)} = \Delta, \Delta^{(1)}, \ldots, \Delta^{(l)} = \Delta']$ where $l \leq 22d$. There must exists $i^* \in [l]$ such that

$$\left| \boldsymbol{f}(\Delta^{(i^*-1)}) - \boldsymbol{f}(\Delta^{(i^*)}) \right| \geq \frac{C}{l}.$$

Note that

$$\boldsymbol{f}^\top \partial_2^\top \partial_2 \boldsymbol{f} \geq \left(\boldsymbol{f}(\Delta^{(i^*-1)}) - \boldsymbol{f}(\Delta^{(i^*)}) \right)^2 \geq \left(\frac{C}{l} \right)^2 \geq \frac{\min\{\lambda_{\min}(\boldsymbol{A}^\top \boldsymbol{A})^2, 1\}}{10^{16} d^7}.$$

Case 2. Suppose for every $i \in [n]$ and every two $\Delta, \Delta' \in \mathcal{K}_i$, we have $|\boldsymbol{f}(\Delta) - \boldsymbol{f}(\Delta')| < C$. We write $\boldsymbol{f}$ as

$$\boldsymbol{f} = \tilde{\boldsymbol{f}} + \boldsymbol{\epsilon} = \begin{bmatrix} \alpha_1 \mathbf{1}_{t_1} \\ \alpha_2 \mathbf{1}_{t_2} \\ \vdots \\ \alpha_n \mathbf{1}_{t_n} \end{bmatrix} + \boldsymbol{\epsilon},$$

where α_i is the value of $\boldsymbol{f}$ at the central triangle of $\mathcal{K}_i$. Then,

$$\|\boldsymbol{\epsilon}\|_2 < \sqrt{t}C \le \sqrt{22d}C = o(1),$$
$$\left\|\tilde{\boldsymbol{f}}\right\|_2 = \|\boldsymbol{f} - \boldsymbol{\epsilon}\|_2 \in \left(\frac{1}{2}, 2\right).$$

We lower bound the quadratic value:

$$\begin{aligned}
\boldsymbol{f}^\top \partial_2^\top \partial_2 \boldsymbol{f} &= \tilde{\boldsymbol{f}}^\top \partial_2^\top \partial_2 \tilde{\boldsymbol{f}} + 2\tilde{\boldsymbol{f}}^\top \partial_2^\top \partial_2 \boldsymbol{\epsilon} + \boldsymbol{\epsilon}^\top \partial_2^\top \partial_2 \boldsymbol{\epsilon} \\
&\ge \tilde{\boldsymbol{f}}^\top \partial_2^\top \partial_2 \tilde{\boldsymbol{f}} - 2\left\|\partial_2^\top \partial_2 \tilde{\boldsymbol{f}}\right\|_2 \|\boldsymbol{\epsilon}\|_2 \\
&\ge \tilde{\boldsymbol{f}}^\top \partial_2^\top \partial_2 \tilde{\boldsymbol{f}} - 4\left\|\partial_2^\top \partial_2\right\|_2 \|\boldsymbol{\epsilon}\|_2 \\
&\ge \tilde{\boldsymbol{f}}^\top \partial_2^\top \partial_2 \tilde{\boldsymbol{f}} - 48\sqrt{22d}C && \text{by Lemma 5.4}
\end{aligned}$$

Note that

$$\tilde{\boldsymbol{f}}^\top \partial_2^\top \partial_2 \tilde{\boldsymbol{f}} = 3\|\boldsymbol{A}\boldsymbol{\alpha}\|_2^2,$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)^\top$. Write $\boldsymbol{\alpha} = \boldsymbol{\alpha}_\perp + \boldsymbol{\alpha}_0$, where $\boldsymbol{\alpha}_\perp$ is orthogonal to the null space of $\boldsymbol{A}$ and $\boldsymbol{\alpha}_0$ is in the null space of $\boldsymbol{A}$. Then,

$$\tilde{\boldsymbol{f}}^\top \partial_2^\top \partial_2 \tilde{\boldsymbol{f}} \ge 3\lambda_{\min}(\boldsymbol{A}^\top \boldsymbol{A})\|\boldsymbol{\alpha}_\perp\|_2^2. \tag{6}$$

It remains to lower bound $\|\boldsymbol{\alpha}_\perp\|_2$. We can write $\tilde{\boldsymbol{f}} = \boldsymbol{H}\boldsymbol{\alpha}_\perp + \boldsymbol{H}\boldsymbol{\alpha}_0$. By Lemma 5.5, $\boldsymbol{H}\boldsymbol{\alpha}_0 \in \mathrm{null}(\partial_2)$ and $\boldsymbol{H}\boldsymbol{\alpha}_\perp \perp \mathrm{null}(\partial_2)$. On the other hand, $\tilde{\boldsymbol{f}} = \boldsymbol{f} - \boldsymbol{\epsilon}$ and $\boldsymbol{f} \perp \mathrm{null}(\partial_2)$. We know

$$\|\boldsymbol{H}\boldsymbol{\alpha}_\perp\|_2 \ge \|\boldsymbol{f}\|_2 - \|\boldsymbol{\epsilon}\|_2,$$

and thus

$$\|\boldsymbol{\alpha}_\perp\|_2 \ge \frac{1}{\sqrt{t}}\|\boldsymbol{H}\boldsymbol{\alpha}_\perp\|_2 \ge \frac{1}{\sqrt{t}} - C > \frac{1}{\sqrt{22d}}.$$

Together with Eq. (6),

$$\tilde{\boldsymbol{f}}^\top \partial_2^\top \partial_2 \tilde{\boldsymbol{f}} \ge \frac{3\lambda_{\min}(\boldsymbol{A}^\top \boldsymbol{A})}{22d}.$$

This completes the proof. □

Proof of Lemma 5.2. The proof follows by combining Lemma 5.4 and 5.6. □

6 Reducing Approximate Solvers for $\mathcal{DA}$ to $\mathcal{B}_2$ in General Case

In this section, we drop the assumption of LEA $(\boldsymbol{A}, \boldsymbol{b}, \epsilon^{DA})$ over $\mathcal{DA}$ that $\boldsymbol{b} \in \mathrm{im}(\boldsymbol{A})$. In this more general case, LEA $(\boldsymbol{A}, \boldsymbol{b}, \epsilon^{DA})$ aims to compute an *approximate* solution to $\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2 = \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{b}\|_2$.

We remark that our reduction for this general case does not work for LE $(\boldsymbol{A}, \boldsymbol{b})$, which computes an *exact* solution to $\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2$. Approximate linear equation solvers, however, can be more interesting in practice, since numerical errors occur unavoidably during data collection and computation and approximate solutions may be computed much faster than exact solutions.

6.1 Reduction algorithm

Our reduction is almost the same as Algorithm 1 ReduceDAToB2, except that we assign edge weights for the constructed 2-complex.

Suppose we are given an instance LEA $(\boldsymbol{A}, \boldsymbol{b}, \epsilon^{DA})$ where $\boldsymbol{A}$ is a $d \times n$ matrix in $\mathcal{DA}$. Let $(\partial_2, \gamma, \boldsymbol{\Delta}^c) \leftarrow$ ReduceDAToB2$(\boldsymbol{A}, \boldsymbol{b})$. Let $\mathcal{K}$ be the 2-complex whose boundary operator is ∂_2. We compute the edge weights of $\mathcal{K}$ as follows.

1. For each boundary triangle $\Delta^1_{q,i,*}$ where $q \in [d], i \in [n]$ and $* \in \{1, 2\}$, we find a minimal triangle path $\mathcal{P}^1_{q,i,*}$ in $\mathcal{K}_i$ from the central triangle Δ_i to $\Delta^1_{q,i,*}$. Let $E^1_{q,j,*}$ be the set of the edges shared by neighboring triangles in $\mathcal{P}^1_{q,i,*}$ [9]. If a triangle is not in any triangle path $\mathcal{P}^1_{q,i,*}$, then we remove the column in ∂_2 corresponding to this triangle.

2. For each interior edge e and equation q, let $k_{q,e}$ be the number of triangle paths indexed by equation q that contain e. For each equation q, let l_q be the sum of the lengths of all the triangle paths indexed by equation q. Then, we set the weight for edge e to be

$$w_e = \begin{cases} 1, & \text{if } e \text{ is a boundary edge} \\ \alpha \sum_{q \in [d]} k_{q,e} l_q, & \text{if } e \text{ is an interior edge} \end{cases} \tag{7}$$

   where $\alpha = \frac{2}{(\epsilon^{DA})^2}$.

Let $\boldsymbol{W}$ be a diagonal matrix whose diagonals are the edge weights. We return $(\boldsymbol{W}, \partial_2, \gamma, \boldsymbol{\Delta}^c)$.

Note that some edges in $\mathcal{K}$ may have weight 0. If we want to make all the edge weights positive, we can impose polynomially small edge weights for these 0-weight edges, which only affects the error propagation and condition number up to polynomially factors.

A pseudo-code is in Algorithm 6 ReduceRegDAToB2. We use Algorithm MapSolnB2toDA to map a solution to LEA $(\boldsymbol{W}^{1/2}\partial_2, \boldsymbol{W}^{1/2}\gamma, \epsilon^{B_2})$ back to a solution to LEA $(\boldsymbol{A}, \boldsymbol{b}, \epsilon^{DA})$, where we choose

$$\epsilon^{B_2} \leq \frac{\epsilon^{DA}}{\sqrt{3\left(1 + \frac{1}{\alpha}\|\boldsymbol{b}\|_2^2 \operatorname{nnz}(\boldsymbol{A})\|\boldsymbol{A}\|_{\max}^2\right)}}.$$

Computing the edge weights in linear time. For each $i \in [n]$, consider a graph G_i whose vertices are the triangles in $\mathcal{K}_i$ and the two vertices are adjacent if and only if the two corresponding triangles share an edge in $\mathcal{K}_i$. We run the breadth-first-search (BFS) to construct a tree with root Δ_i, the ith central triangle, and with leaf nodes being the boundary triangles in $\mathcal{K}_i$, and with edges from G_i. For each boundary triangle $\Delta^1_{q,i,*}$, we choose the triangle path $\mathcal{P}^1_{q,i,*}$ to be the triangle path from the root to the node corresponding to $\Delta^1_{q,i,*}$ induced by the tree, whose length is the height of the node.

Our goal is to count $\sum_{q \in [d]} k_{q,e} l_q$ for each edge in the tree. We observe that an edge e appears in a triangle path from the root to a boundary triangle if and only if the boundary triangle is a descendant of an end-node of e with the larger height. First, we BFS traverse the tree to store the height of each boundary triangle node. Then, we traverse the tree to store l_q at each boundary triangle node indexed by q. Next, we traverse the tree from the leaf nodes to the root to count $\sum_{q \in [d]} k_{q,e} l_q$ for each edge e. The total runtime is linear in the number of triangles in $\mathcal{K}_i$.

[9] Note that any two neighboring triangles in $\mathcal{P}^1_{q,i,*}$ share exactly one edge. So, the length of $\mathcal{P}^1_{q,i,*}$ equals $\left|E^1_{q,i,*}\right|$.

Algorithm 6: ReduceReg$\mathcal{DA}$To$\mathcal{B}_2$

Input: a tuple $(\boldsymbol{A}, \boldsymbol{b}, \alpha)$, where $\boldsymbol{A} \in \mathcal{DA}$ is a $d \times n$ matrix, $\boldsymbol{b} \in \mathbb{R}^d$, and $\alpha > 0$

Output: a tuple $(\boldsymbol{W}, \partial_2, \gamma, \boldsymbol{\Delta}^c)$, where $\boldsymbol{W} \in \mathbb{R}_{>0}^{m \times m}$, $\partial_2 \in \mathcal{B}_2$ is an $m \times t$ matrix, $\gamma \in \mathbb{R}^m$, and $\boldsymbol{\Delta}^c$ is a set of n central triangles

$(\partial_2, \gamma, \boldsymbol{\Delta}^c) \leftarrow$ReduceReg$\mathcal{DA}To\mathcal{B}_2(\boldsymbol{A}, \boldsymbol{b})$;
Initiate $\boldsymbol{K} = \boldsymbol{0}_{d \times m}$; `// a count matrix where` $\boldsymbol{K}(q, e) = k_{q,e}$
Initialize $\boldsymbol{L} = \boldsymbol{0}_d$; `// a vector of sum of triangle path length where` $\boldsymbol{L}(q) = l_q$
for $1 \le q \le d$ **do**
  for $1 \le i \le n$ **do**
    $E_{q,i,*} \leftarrow$ the set of interior edges in $\mathcal{P}_{q,i,*}$, a triangle path from $\Delta_i \in \boldsymbol{\Delta}^c$ to $\Delta^1_{q,i,*}$;
    $\boldsymbol{L}(q) = \boldsymbol{L}(q) + |E_{q,i,*}|$;
    for $e \in E_{q,i,*}$ **do**
      $\boldsymbol{K}(q, e) = \boldsymbol{K}(q, e) + 1$;
    end
  end
end
Initialize $\boldsymbol{w} = \boldsymbol{0}_m$;
for $1 \le e \le m$ **do**
  $\boldsymbol{w}[e] = \alpha \sum_{q \in [d]} \boldsymbol{K}(q, e) \boldsymbol{L}(q)$;
  if $\boldsymbol{w}[e] = 0$ **then**
    $\boldsymbol{w}[e] = 1$; `// boundary edges do not appear in any triangle path`
  end
end
$\boldsymbol{W} = \mathrm{diag}(\boldsymbol{w})$;
return $(\boldsymbol{W}, \partial_2, \gamma, \boldsymbol{\Delta}^c)$

6.2 Relation between exact solutions

In this section, we show a relation between exact solutions to $(\boldsymbol{A}, \boldsymbol{b})$ and those to $(\boldsymbol{W}^{1/2}\partial_2, \boldsymbol{W}^{1/2}\gamma)$. This relation plays a key role in analyzing approximate solutions and condition numbers.

Claim 6.1. *For any* $\boldsymbol{f} \in \mathbb{R}^t$ *and* $\boldsymbol{x} \leftarrow$ MapSoln$\mathcal{B}_2$to$\mathcal{DA}(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{f}, \boldsymbol{\Delta}^c)$,

$$\frac{\alpha}{\alpha + 1} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 \le \left\| \boldsymbol{W}^{1/2}\partial_2 \boldsymbol{f} - \boldsymbol{W}^{1/2}\gamma \right\|_2^2.$$

Note that if $\boldsymbol{f}$ satisfies $\partial_2 \boldsymbol{f} = \gamma$, then $\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 = \|\partial_2 \boldsymbol{f} - \gamma\|_2^2$. Claim 6.1 states that by our choice of weights in $\boldsymbol{W}$, we can generalize this relation to more general $\boldsymbol{f}$.

Proof. For the convenience of analysis, we construct an auxiliary boundary matrix $\hat{\partial}_2$. For each interior edge e, let k_e be the number of all the triangle paths containing e, and we split the row $\partial_2(e)$ into k_e copies. For each copy related to equation q, we assign its weight to be αl_q. Let $\hat{\boldsymbol{W}}$ be the auxiliary weight matrix, and let $\hat{\gamma}$ be the auxiliary demand vector. We can check

$$\left\| \boldsymbol{W}^{1/2}\partial_2 \boldsymbol{f} - \boldsymbol{W}^{1/2}\gamma \right\|_2^2 = \left\| \hat{\boldsymbol{W}}^{1/2}\hat{\partial}_2 \boldsymbol{f} - \hat{\boldsymbol{W}}^{1/2}\hat{\gamma} \right\|_2^2. \tag{8}$$

We reorder rows of the matrices $\hat{\boldsymbol{W}}, \hat{\partial}_2$ and the vector $\hat{\gamma}$ in the following way. For each $q \in [d]$, let E_q be a (multi)set that is the union of the shared edges in the triangle paths indexed by q [10]. Then, we reorder rows of $\hat{\boldsymbol{W}}, \hat{\partial}_2, \hat{\gamma}$ by grouping those corresponding to the edges in $E_q \cup \{\alpha_q^1\}$:

$$\hat{\partial}_2 = \begin{bmatrix} \boldsymbol{G}_1 \\ \vdots \\ \boldsymbol{G}_d \end{bmatrix}, \qquad \text{where} \quad \boldsymbol{G}_q = \begin{bmatrix} \boldsymbol{B}_q \\ \boldsymbol{M}_q \end{bmatrix}. \tag{9}$$

where $\boldsymbol{B}_q$ corresponds to the boundary edge α_q^1 and $\boldsymbol{M}_q$ is the submatrix corresponding to all the interior edges in E_q. Correspondingly, we write

$$\hat{\boldsymbol{W}} = \begin{bmatrix} \hat{\boldsymbol{W}}_1 & & \\ & \ddots & \\ & & \hat{\boldsymbol{W}}_d \end{bmatrix}, \quad \text{and} \quad \hat{\gamma} = \begin{bmatrix} \hat{\gamma}_1 \\ \vdots \\ \hat{\gamma}_d \end{bmatrix}, \quad \text{where} \quad \hat{\gamma}_q = \begin{bmatrix} \boldsymbol{b}(q) \\ \boldsymbol{0}_{|E_q|} \end{bmatrix}. \tag{10}$$

Let

$$\hat{w}_{q,e} \stackrel{\text{def}}{=} \hat{\boldsymbol{W}}_q(e,e) = \begin{cases} 1, & \text{if } e \text{ is a boundary edge} \\ \alpha \cdot l_q, & \text{if } e \text{ is an interior edge} \end{cases}$$

For each $q \in [d], r \in [3]$, we define

$$\epsilon_q = \boldsymbol{A}(q)\boldsymbol{x} - \boldsymbol{b}(q),\ \xi_q = \boldsymbol{B}_q\boldsymbol{f} - \boldsymbol{b}(q),\ \delta_e = \boldsymbol{M}_q(e)\boldsymbol{f}.$$

We can check

$$\epsilon_q = \boldsymbol{1}^\top (\boldsymbol{G}_q\boldsymbol{f} - \hat{\gamma}_q) = \xi_q + \sum_{e \in E_q} \delta_e.$$

By the Cauchy-Schwarz inequality,

$$\epsilon_q^2 = \left(\xi_q + \sum_{e\in E_q} \delta_e\right)^2 \le \left(1 + \sum_{e\in E_q} \frac{1}{\hat{w}_{q,e}}\right)\left(\xi_q^2 + \sum_{e\in E_q} \hat{w}_{q,e}\delta_e^2\right) = \left(1+\frac{1}{\alpha}\right)\left(\xi_q^2 + \sum_{e\in E_q} \hat{w}_{q,e}\delta_e^2\right),$$

that is,

$$\|\boldsymbol{A}(q)\boldsymbol{x} - \boldsymbol{b}(q)\|_2^2 \le \left(1+\frac{1}{\alpha}\right)\left\|\hat{\boldsymbol{W}}_q^{1/2}\boldsymbol{G}_q\boldsymbol{f} - \hat{\boldsymbol{W}}_q^{1/2}\hat{\gamma}_q\right\|_2^2.$$

Summing over all d equations, we get

$$\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 \le \left(1+\frac{1}{\alpha}\right)\left\|\hat{\boldsymbol{W}}^{1/2}\hat{\partial}_2\boldsymbol{f} - \hat{\boldsymbol{W}}^{1/2}\hat{\gamma}\right\|_2^2.$$

By Eq. (8), we get the inequality in the statement. □

Claim 6.1 implies the following relation between the exact solutions to $(\boldsymbol{A}, \boldsymbol{b})$ and those to $(\boldsymbol{W}^{1/2}\partial_2, \boldsymbol{W}^{1/2}\gamma)$.

[10] If the qth equation is a difference equation, then every edge appears in E_q at most once; if the qth equation is an average equation, then some edges may appear twice.

Lemma 6.2. *Given any $d \times n$ matrix $\boldsymbol{A} \in \mathcal{DA}$ and vector $\boldsymbol{b} \in \mathbb{R}^d$, let $(\boldsymbol{W}, \partial_2, \gamma, \boldsymbol{\Delta}^c) \leftarrow$ ReduceReg$\mathcal{DA}$To$\mathcal{B}_2(\boldsymbol{A}, \boldsymbol{b}, \epsilon^{DA})$. Then,*

$$\frac{\alpha}{\alpha+1} \min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 \leq \min_{\boldsymbol{f}} \left\| \boldsymbol{W}^{1/2}\partial_2 \boldsymbol{f} - \boldsymbol{W}^{1/2}\gamma \right\|_2^2 \leq \min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 .$$

Remark that Lemma 6.2 can also be stated as

$$\frac{\alpha}{\alpha+1} \|(\boldsymbol{I} - \boldsymbol{\Pi}_{\boldsymbol{A}})\boldsymbol{b}\|_2^2 \leq \left\| (\boldsymbol{I} - \boldsymbol{\Pi}_{\boldsymbol{W}^{1/2}\partial_2}) \boldsymbol{W}^{1/2}\gamma \right\|_2^2 \leq \|(\boldsymbol{I} - \boldsymbol{\Pi}_{\boldsymbol{A}})\boldsymbol{b}\|_2^2 .$$

We do not prove equalities. But as $\alpha \to \infty$, the leftmost hand side and the rightmost side hand are equal.

Proof. Let $\boldsymbol{f}^* \in \arg\min_{\boldsymbol{f}} \left\| \boldsymbol{W}^{1/2}\partial_2 \boldsymbol{f} - \boldsymbol{W}^{1/2}\gamma \right\|_2$ and $\boldsymbol{x}^* \in \arg\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2$. Let $\boldsymbol{x} \leftarrow$ MapSoln$\mathcal{B}_2$to$\mathcal{DA}(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{W}^{1/2}\partial_2, \boldsymbol{f}^*)$. By Claim 6.1

$$\frac{\alpha}{\alpha+1} \|\boldsymbol{A}\boldsymbol{x}^* - \boldsymbol{b}\|_2^2 \leq \frac{\alpha}{\alpha+1} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 \leq \left\| \boldsymbol{W}^{1/2}\partial_2 \boldsymbol{f}^* - \boldsymbol{W}^{1/2}\gamma \right\|_2^2,$$

which is the first inequality in the lemma statement. Let

$$\boldsymbol{f} = \begin{bmatrix} \boldsymbol{x}^*(1)\mathbf{1}_{t_1} \\ \vdots \\ \boldsymbol{x}^*(n)\mathbf{1}_{t_n} \end{bmatrix}.$$

Then,

$$\left\| \boldsymbol{W}^{1/2}\partial_2 \boldsymbol{f}^* - \boldsymbol{W}^{1/2}\gamma \right\|_2^2 \leq \left\| \boldsymbol{W}^{1/2}\partial_2 \boldsymbol{f} - \boldsymbol{W}^{1/2}\gamma \right\|_2^2 = \|\boldsymbol{A}\boldsymbol{x}^* - \boldsymbol{b}\|_2^2 ,$$

which is the second inequality in the lemma statement. □

6.3 Relation between approximate solutions

Linear equation problem LE $(\boldsymbol{A}, \boldsymbol{b})$ aims to find a vector $\boldsymbol{x}$ such that $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{b}$. In our setting, both $\boldsymbol{A}$ and $\boldsymbol{b}$ have integer entries. We will need the following claim to lower bound $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{b}$.

Claim 6.3. *Let $\boldsymbol{A} \in \mathbb{R}^{d\times n}$ and $\boldsymbol{b} \in \mathbb{R}^d$ such that $\left\|\boldsymbol{A}^\top \boldsymbol{b}\right\|_2 \geq 1$. Then,*

$$\|\boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{b}\|_2^2 \geq \frac{1}{\lambda_{\max}\left(\boldsymbol{A}^\top \boldsymbol{A}\right)}.$$

Proof. Note that

$$\|\boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{b}\|_2^2 = \boldsymbol{b}^\top \boldsymbol{A}(\boldsymbol{A}^\top \boldsymbol{A})^\dagger \boldsymbol{A}^\top \boldsymbol{b} \geq \lambda_{\min}((\boldsymbol{A}^\top \boldsymbol{A})^\dagger) \left\|\boldsymbol{A}^\top \boldsymbol{b}\right\|_2^2 \geq \frac{1}{\lambda_{\max}\left(\boldsymbol{A}^\top \boldsymbol{A}\right)}.$$

□

Claim 6.3 and Lemma 6.2 enable us to relate $\|\boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{b}\|_2$ and $\left\|\boldsymbol{\Pi}_{\boldsymbol{W}^{1/2}\partial_2} \boldsymbol{W}^{1/2}\gamma\right\|_2$.

Claim 6.4.

$$\left\|\boldsymbol{\Pi}_{\boldsymbol{W}^{1/2}\partial_2} \boldsymbol{W}^{1/2}\gamma\right\|_2^2 \leq \left(1 + \frac{1}{\alpha}\lambda_{\max}(\boldsymbol{A}^\top \boldsymbol{A}) \|\boldsymbol{b}\|_2^2\right) \|\boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{b}\|_2^2 .$$

Proof. We apply Lemma 6.2.

$$\begin{aligned}\|\boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{b}\|_2^2 &= \|\boldsymbol{b}\|_2^2 - \|(\boldsymbol{I}-\boldsymbol{\Pi}_{\boldsymbol{A}})\boldsymbol{b}\|_2^2 \\ &\geq \left\|\boldsymbol{W}^{1/2}\gamma\right\|_2^2 - \frac{\alpha+1}{\alpha}\left\|(\boldsymbol{I}-\boldsymbol{\Pi}_{\boldsymbol{W}^{1/2}\partial_2})\,\boldsymbol{W}^{1/2}\gamma\right\|_2^2 \\ &= \left\|\boldsymbol{\Pi}_{\boldsymbol{W}^{1/2}\partial_2}\boldsymbol{W}^{1/2}\gamma\right\|_2^2 - \frac{1}{\alpha}\left\|(\boldsymbol{I}-\boldsymbol{\Pi}_{\boldsymbol{W}^{1/2}\partial_2})\,\boldsymbol{W}^{1/2}\gamma\right\|_2^2 \\ &\geq \left\|\boldsymbol{\Pi}_{\boldsymbol{W}^{1/2}\partial_2}\boldsymbol{W}^{1/2}\gamma\right\|_2^2 - \frac{1}{\alpha}\|(\boldsymbol{I}-\boldsymbol{\Pi}_{\boldsymbol{A}})\boldsymbol{b}\|_2^2\end{aligned}$$

Since $\boldsymbol{A}, \boldsymbol{b}$ have integer entries, by Claim 6.3,

$$\|(\boldsymbol{I}-\boldsymbol{\Pi}_{\boldsymbol{A}})\boldsymbol{b}\|_2^2 \leq \|\boldsymbol{b}\|_2^2 \leq \|\boldsymbol{b}\|_2^2 \cdot \lambda_{\max}(\boldsymbol{A}^\top\boldsymbol{A})\,\|\boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{b}\|_2^2\,.$$

Thus,

$$\left\|\boldsymbol{\Pi}_{\boldsymbol{W}^{1/2}\partial_2}\boldsymbol{W}^{1/2}\gamma\right\|_2^2 \leq \left(1+\frac{1}{\alpha}\lambda_{\max}(\boldsymbol{A}^\top\boldsymbol{A})\,\|\boldsymbol{b}\|_2^2\right)\|\boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{b}\|_2^2\,.$$

□

Now, we apply Claim 6.1 and Lemma 6.4 to prove a relation between approximate solutions.

Lemma 6.5. *Given a difference-average instance* LEA $(\boldsymbol{A}, \boldsymbol{b}, \epsilon^{DA})$*, let* $(\boldsymbol{W}, \partial_2, \gamma, \boldsymbol{\Delta}^c) \leftarrow$ REDUCEREG$\mathcal{DA}$TO$\mathcal{B}_2(\boldsymbol{A}, \boldsymbol{b}, \alpha)$*. Suppose* $\boldsymbol{f}$ *is a solution to* LEA $(\boldsymbol{W}^{1/2}\partial_2, \boldsymbol{W}^{1/2}\gamma, \epsilon^{B_2})$*, where* $\epsilon^{B_2} \leq \frac{\epsilon^{DA}}{10}$*. Let* $\boldsymbol{x} \leftarrow$ MAPSOLN$\mathcal{B}_2$TO$\mathcal{DA}(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{f}, \boldsymbol{\Delta}^c)$*. Then,* $\boldsymbol{x}$ *is a solution to* LEA $(\boldsymbol{A}, \boldsymbol{b}, \epsilon^{DA})$*.*

Proof. By Claim 6.1, we have

$$\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_2^2 \leq \frac{\alpha+1}{\alpha}\left\|\boldsymbol{W}^{1/2}\partial_2\boldsymbol{f} - \boldsymbol{W}^{1/2}\gamma\right\|_2^2. \tag{11}$$

Also note that

$$\begin{aligned}\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_2^2 &= \|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{b}\|_2^2 + \|(\boldsymbol{I}-\boldsymbol{\Pi}_{\boldsymbol{A}})\boldsymbol{b}\|_2^2\,, \\ \left\|\boldsymbol{W}^{1/2}\partial_2\boldsymbol{f} - \boldsymbol{W}^{1/2}\gamma\right\|_2^2 &= \left\|\boldsymbol{W}^{1/2}\partial_2\boldsymbol{f} - \boldsymbol{\Pi}_{\boldsymbol{W}^{1/2}\partial_2}\boldsymbol{W}^{1/2}\gamma\right\|_2^2 + \left\|(\boldsymbol{I}-\boldsymbol{\Pi}_{\boldsymbol{W}^{1/2}\partial_2})\,\boldsymbol{W}^{1/2}\gamma\right\|_2^2.\end{aligned}$$

Plugging these into Eq. (11) and apply Lemma 6.2,

$$\begin{aligned}\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{b}\|_2^2 &\leq \frac{\alpha+1}{\alpha}\left(\left\|\boldsymbol{W}^{1/2}\partial_2\boldsymbol{f} - \boldsymbol{\Pi}_{\boldsymbol{W}^{1/2}\partial_2}\boldsymbol{W}^{1/2}\gamma\right\|_2^2 + \left\|(\boldsymbol{I}-\boldsymbol{\Pi}_{\boldsymbol{W}^{1/2}\partial_2})\,\boldsymbol{W}^{1/2}\gamma\right\|_2^2\right) - \|(\boldsymbol{I}-\boldsymbol{\Pi}_{\boldsymbol{A}})\boldsymbol{b}\|_2^2 \\ &\leq \frac{\alpha+1}{\alpha}\left\|\boldsymbol{W}^{1/2}\partial_2\boldsymbol{f} - \boldsymbol{\Pi}_{\boldsymbol{W}^{1/2}\partial_2}\boldsymbol{W}^{1/2}\gamma\right\|_2^2 + \frac{1}{\alpha}\|(\boldsymbol{I}-\boldsymbol{\Pi}_{\boldsymbol{A}})\boldsymbol{b}\|_2^2\,.\end{aligned}$$

By the fact that $\boldsymbol{f}$ is a solution to LEA $(\boldsymbol{W}^{1/2}\partial_2, \boldsymbol{W}^{1/2}\gamma, \epsilon^{B_2})$ and by Claim 6.4,

$$\left\|\boldsymbol{W}^{1/2}\partial_2\boldsymbol{f} - \boldsymbol{\Pi}_{\boldsymbol{W}^{1/2}\partial_2}\boldsymbol{W}^{1/2}\gamma\right\|_2^2 \leq \left(\epsilon^{B_2}\right)^2\left\|\boldsymbol{\Pi}_{\boldsymbol{W}^{1/2}\partial_2}\boldsymbol{W}^{1/2}\gamma\right\|_2^2 \leq \frac{\left(\epsilon^{DA}\right)^2}{3}\|\boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{b}\|_2^2\,.$$

In addition,

$$\frac{1}{\alpha}\|(\boldsymbol{I}-\boldsymbol{\Pi}_{\boldsymbol{A}})\boldsymbol{b}\|_2^2 \leq \frac{1}{\alpha}\|\boldsymbol{b}\|_2^2 \leq \frac{\left(\epsilon^{DA}\right)^2}{2}\|\boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{b}\|_2^2\,.$$

Thus,

$$\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{b}\|_2^2 \leq \left(\epsilon^{DA}\right)^2\|\boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{b}\|_2^2\,,$$

that is, $\boldsymbol{x}$ is a solution to LEA $(\boldsymbol{A}, \boldsymbol{b}, \epsilon^{DA})$. □

6.4 Bounding the condition number of the new matrix

We will upper bound the maximum eigenvalue of $\partial_2^\top \boldsymbol{W} \partial_2$ and lower bound its minimum nonzero eigenvalue. The proofs are similar to those in Section 5.2, which bound the eigenvalues of $\partial_2^\top \partial_2$.

Lemma 6.6. $\kappa(\partial_2^\top \boldsymbol{W} \partial_2) = O\left((\epsilon^{DA})^{-2} \operatorname{nnz}(\boldsymbol{A})^{15/2} \kappa(\boldsymbol{A}^\top \boldsymbol{A})\right)$.

Proof. The proof follows the same proof line in Section 5.2 for $\boldsymbol{W} = \boldsymbol{I}$. Here, we lose a factor $w_{\max}$ when we upper bound $\lambda_{\max}(\partial_2^\top \boldsymbol{W} \partial_2)$, and we lose a factor $\frac{w_{\max}}{w_{\min}}$ when we lower bound $\lambda_{\max}(\partial_2^\top \boldsymbol{W} \partial_2)$, where $w_{\max}$ is the maximum diagonal in $\boldsymbol{W}$ and $w_{\min}$ is the minimum nonzero diagonal in $\boldsymbol{W}$. By our setting, $w_{\max} = O(\alpha \operatorname{nnz}(\boldsymbol{A})^2)$ and $w_{\min} = \alpha$, where $\alpha = 2(\epsilon^{DA})^{-2}$. □

References

[AW21] J. Alman and V. V. Williams. "A refined laser method and faster matrix multiplication". In: *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM. 2021, pp. 522–539 (cit. on pp. 2, 4).

[BHV08] E. G. Boman, B. Hendrickson, and S. Vavasis. "Solving elliptic finite element systems in near-linear time with support preconditioners". In: *SIAM Journal on Numerical Analysis* 46.6 (2008), pp. 3264–3284 (cit. on p. 5).

[Bla+22] M. Black, W. Maxwell, A. Nayyeri, and E. Winkel. "Computational Topology in a Collapsing Universe: Laplacians, Homology, Cohomology". In: *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM. 2022 (cit. on pp. 1, 4).

[BV21] M. Bafna and N. Vyas. "Optimal Fine-Grained Hardness of Approximation of Linear Equations". In: *48th International Colloquium on Automata, Languages, and Programming (ICALP 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik. 2021 (cit. on p. 1).

[Car09] G. Carlsson. "Topology and Data". In: *Bulletin of the American Mathematical Society* 46.2 (2009), pp. 255–308 (cit. on p. 1).

[Cha+16] F. Chazal, V. de Silva, M. Glisse, and S. Oudot. *The Structure and Stability of Persistence Modules*. Springer, 2016. ISBN: 978-3-319-42545-0 (cit. on p. 1).

[CMZ18] C. K. Chui, H. Mhaskar, and X. Zhuang. "Representation of functions on big data associated with directed graphs". In: *Applied and Computational Harmonic Analysis* 44.1 (2018), pp. 165–188 (cit. on p. 1).

[Coh+14a] M. B. Cohen, B. T. Fasy, G. L. Miller, A. Nayyeri, R. Peng, and N. Walkington. "Solving 1-laplacians in nearly linear time: Collapsing and expanding a topological ball". In: *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM. 2014, pp. 204–216 (cit. on p. 1).

[Coh+14b] M. B. Cohen, R. Kyng, G. L. Miller, J. W. Pachocki, R. Peng, A. B. Rao, and S. C. Xu. "Solving SDD linear systems in nearly m log 1/2 n time". In: *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*. ACM. 2014, pp. 343–352 (cit. on p. 1).

[Coh+17] M. B. Cohen, J. Kelner, J. Peebles, R. Peng, A. B. Rao, A. Sidford, and A. Vladu. "Almost-linear-time Algorithms for Markov Chains and New Spectral Primitives for Directed Graphs". In: *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*. STOC 2017. Montreal, Canada: ACM, 2017, pp. 410–419. ISBN: 978-1-4503-4528-6 (cit. on p. 1).

[Coh+18] M. B. Cohen, J. Kelner, R. Kyng, J. Peebles, R. Peng, A. B. Rao, and A. Sidford. "Solving directed laplacian systems in nearly-linear time through sparse LU factorizations". In: *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2018, pp. 898–909 (cit. on p. 1).

[DKM09] A. Duval, C. Klivans, and J. Martin. "Simplicial matrix-tree theorems". In: *Transactions of the American Mathematical Society* 361.11 (2009), pp. 6073–6114 (cit. on p. 1).

[DKM15] A. M. Duval, C. J. Klivans, and J. L. Martin. "Cuts and flows of cell complexes". In: *Journal of Algebraic Combinatorics* 41.4 (2015), pp. 969–999 (cit. on pp. 1, 4).

[DKT08] M. Desbrun, E. Kanso, and Y. Tong. "Discrete differential forms for computational modeling". In: *Discrete differential geometry*. Springer, 2008, pp. 287–324 (cit. on p. 1).

[DS07] S. I. Daitch and D. A. Spielman. "Support-graph preconditioners for 2-dimensional trusses". In: *arXiv preprint cs/0703119* (2007) (cit. on p. 5).

[DW02] X. Dong and M. L. Wachs. "Combinatorial Laplacian of the matching complex". In: *the electronic journal of combinatorics* (2002), R17–R17 (cit. on p. 1).

[Eck44] B. Eckmann. "Harmonische funktionen und randwertaufgaben in einem komplex". In: *Commentarii Mathematici Helvetici* 17.1 (1944), pp. 240–255 (cit. on p. 1).

[EH10] H. Edelsbrunner and J. Harer. *Computational Topology: An Introduction*. American Mathematical Soc., 2010 (cit. on p. 1).

[Fri98] J. Friedman. "Computing Betti numbers via combinatorial Laplacians". In: *Algorithmica* 21.4 (1998), pp. 331–346 (cit. on p. 1).

[Ghr08] R. Ghrist. "Barcodes: The Persistent Topology of Data". In: *Bulletin of the American Mathematical Society* 45.1 (2008), pp. 61–75 (cit. on p. 1).

[Hat00] A. Hatcher. *Algebraic topology*. Cambridge University Press, 2000 (cit. on p. 5).

[HS+52] M. R. Hestenes, E. Stiefel, et al. *Methods of conjugate gradients for solving linear systems*. Vol. 49. 1. NBS Washington, DC, 1952 (cit. on p. 5).

[Jia+11] X. Jiang, L.-H. Lim, Y. Yao, and Y. Ye. "Statistical ranking and combinatorial Hodge theory". In: *Mathematical Programming* 127.1 (2011), pp. 203–244 (cit. on p. 1).

[JS21] A. Jambulapati and A. Sidford. "Ultrasparse Ultrasparsifiers and Faster Laplacian System Solvers". In: *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 2021, pp. 540–559 (cit. on p. 1).

[KMP10] I. Koutis, G. L. Miller, and R. Peng. "Approaching Optimality for Solving SDD Linear Systems". In: *Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science*. FOCS '10. USA: IEEE Computer Society, 2010, pp. 235–244. ISBN: 978-0-7695-4244-7 (cit. on p. 1).

[KMP11] I. Koutis, G. L. Miller, and R. Peng. "A Nearly-m Log n Time Solver for Sdd Linear Systems". In: *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*. IEEE, 2011, pp. 590–598 (cit. on p. 1).

[KMT11] I. Koutis, G. L. Miller, and D. Tolliver. "Combinatorial preconditioners and multilevel solvers for problems in computer vision and image processing". In: *Computer Vision and Image Understanding* 115.12 (2011), pp. 1638–1646 (cit. on p. 5).

[KS16] R. Kyng and S. Sachdeva. "Approximate Gaussian Elimination for Laplacians-Fast, Sparse, and Simple". In: *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2016, pp. 573–582 (cit. on p. 1).

[KWZ20] R. Kyng, D. Wang, and P. Zhang. "Packing LPs Are Hard to Solve Accurately, Assuming Linear Equations Are Hard". In: *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2020, pp. 279–296 (cit. on pp. 2, 5).

[Kyn+16] R. Kyng, Y. T. Lee, R. Peng, S. Sachdeva, and D. A. Spielman. "Sparsified Cholesky and Multigrid Solvers for Connection Laplacians". In: *Proceedings of the Forty-eighth Annual ACM Symposium on Theory of Computing*. STOC '16. Cambridge, MA, USA: ACM, 2016, pp. 842–850. ISBN: 978-1-4503-4132-5 (cit. on p. 1).

[Kyn+18] R. Kyng, R. Peng, R. Schwieterman, and P. Zhang. "Incomplete nested dissection". In: *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*. 2018, pp. 404–417 (cit. on p. 5).

[KZ20] R. Kyng and P. Zhang. "Hardness results for structured linear systems". In: *SIAM Journal on Computing* 49.4 (2020), FOCS17–280 (cit. on pp. 2, 7, 8, 9, 32, 33, 34).

[Lim20] L.-H. Lim. "Hodge laplacians on graphs". In: *Siam Review* 62.3 (2020), pp. 685–715 (cit. on pp. 1, 6).

[Ma+11] W. Ma, J.-M. Morel, S. Osher, and A. Chien. "An L 1-based variational model for Retinex theory and its application to medical images". In: *CVPR 2011*. IEEE. 2011, pp. 153–160 (cit. on p. 1).

[Mad16] A. Madry. "Computing maximum flow with augmenting electrical flows". In: *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2016, pp. 593–602 (cit. on p. 41).

[MN21] W. Maxwell and A. Nayyeri. "Generalized max-flows and min-cuts in simplicial complexes". In: *arXiv preprint arXiv:2106.14116* (2021) (cit. on pp. 1, 3, 4).

[Mun18] J. R. Munkres. *Elements of algebraic topology*. CRC press, 2018 (cit. on p. 5).

[Nie21] Z. Nie. "Matrix anti-concentration inequalities with applications". In: *arXiv preprint arXiv:2111.05553* (2021) (cit. on pp. 2, 5).

[PS14] R. Peng and D. A. Spielman. "An Efficient Parallel Solver for SDD Linear Systems". In: *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing*. 2014, pp. 333–342 (cit. on p. 1).

[PV21] R. Peng and S. Vempala. "Solving sparse linear systems faster than matrix multiplication". In: *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM. 2021, pp. 504–521 (cit. on pp. 2, 5).

[Ren01] J. Renegar. *A mathematical view of interior-point methods in convex optimization*. SIAM, 2001 (cit. on p. 41).

[Sch+20] M. T. Schaub, A. R. Benson, P. Horn, G. Lippner, and A. Jadbabaie. "Random walks on simplicial complexes and the normalized hodge 1-laplacian". In: *SIAM Review* 62.2 (2020), pp. 353–391 (cit. on p. 1).

[ST14] D. A. Spielman and S.-H. Teng. "Nearly linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems". In: *SIAM Journal on Matrix Analysis and Applications* 35.3 (2014), pp. 835–885 (cit. on p. 1).

[Ton+03] Y. Tong, S. Lombeyda, A. N. Hirani, and M. Desbrun. "Discrete multiscale vector field decomposition". In: *ACM transactions on graphics (TOG)* 22.3 (2003), pp. 445–452 (cit. on p. 1).

[Xu+12] Q. Xu, Q. Huang, T. Jiang, B. Yan, W. Lin, and Y. Yao. "HodgeRank on random graphs for subjective video quality assessment". In: *IEEE Transactions on Multimedia* 14.3 (2012), pp. 844–857 (cit. on p. 1).

[YL17] K. Ye and L.-H. Lim. "Cohomology of cryo-electron microscopy". In: *SIAM Journal on Applied Algebra and Geometry* 1.1 (2017), pp. 507–535 (cit. on p. 1).

[Zom05] A. J. Zomorodian. *Topology for Computing*. Vol. 16. Cambridge university press, 2005 (cit. on p. 1).

A Reducing General Linear Equations to Difference-Average Linear Equations

In this section, we prove Theorem 2.9: LEA over $\mathcal{DA}$ is sparse-linear-equation complete.

We show a nearly-linear time reduction from linear equation problems over matrices in $\mathcal{G}$ to linear equation problems over matrices in $\mathcal{DA}$. This reduction is implicit in Section 4 of [KZ20], as an intermediate step to reduce linear equation problems over matrices in $\mathcal{G}$ to matrices in a slight generalization of Laplacians. We explicitly separate the reduction step and simplify the proofs in [KZ20], which might be of independent interest.

Recall that a matrix in $\mathcal{G}$ has polynomially bounded integer entries and polynomially bounded condition numbers, and they do not have all-0 rows or all-0 columns; a matrix $\boldsymbol{A} \in \mathcal{DA}$ only has two types of rows such that if we multiply $\boldsymbol{A}$ to a vector $\boldsymbol{x}$, then the entries of $\boldsymbol{Ax}$ are in the form of either $\boldsymbol{x}(i) - \boldsymbol{x}(j)$ or $\boldsymbol{x}(i) + \boldsymbol{x}(j) - 2\boldsymbol{x}(k)$.

The reduction in [KZ20] has three steps:

1. Reduce linear equation problems over matrices in $\mathcal{G}$ to matrices in $\mathcal{G}_z$, a subset of $\mathcal{G}$ containing matrices with row sum 0. Given an instance $(\boldsymbol{G}, \boldsymbol{b})$ where $\boldsymbol{G} \in \mathcal{G}$, we construct a new instance $(\boldsymbol{G}', \boldsymbol{b})$ where $\boldsymbol{G}' = \begin{bmatrix} \boldsymbol{G} & -\boldsymbol{G}\mathbf{1} \end{bmatrix} \in \mathcal{G}_z$.

2. Reduce linear equation problems over matrices in $\mathcal{G}_z$ to matrices in $\mathcal{G}_{z,2}$, a subset of $\mathcal{G}_z$ containing matrices such that the sum of positive entries in each row is a power of 2. Given an instance $(\boldsymbol{G}', \boldsymbol{b})$ where $\boldsymbol{G}' \in \mathcal{G}_z$, we construct a new instance $(\boldsymbol{G}'', \boldsymbol{b}'')$ where $\boldsymbol{G}'' = \begin{bmatrix} \boldsymbol{G}' & \boldsymbol{g} & -\boldsymbol{g} \\ \mathbf{0} & 1 & -1 \end{bmatrix} \in \mathcal{G}_{z,2}$ and $\boldsymbol{b}'' = \begin{bmatrix} \boldsymbol{b} \\ 0 \end{bmatrix}$.

3. Reduce linear equation problems over matrices in $\mathcal{G}_{z,2}$ to matrices in $\mathcal{DA}$.

The first two steps are proved in Section 7 of [KZ20], by standard tricks. We will focus on the third step.

In the rest part of this section, to be consistent with the notations used in [KZ20], we use subscripts to denote entries of a matrix or a vector. Let $\boldsymbol{A}_i$ denote the ith row of a matrix $\boldsymbol{A}$, and $\boldsymbol{A}_{ij}$ denote the (i,j)th entry of $\boldsymbol{A}$. Let $\boldsymbol{x}_i$ denote the ith entry of a vector $\boldsymbol{x}$, and $\boldsymbol{x}_{i:j}$ denote the subvector of entries $\boldsymbol{x}_i, \boldsymbol{x}_{i+1}, \ldots, \boldsymbol{x}_j$. Moreover, we let $\|\boldsymbol{A}\|_{\max}$ be the maximum magnitude of the entries of $\boldsymbol{A}$.

A.1 Reduction algorithm

Given an instance of linear equation problems over $\mathcal{G}_{z,2}$, our construction of an instance over $\mathcal{DA}$ does not depend on the error parameter. Therefore, we will describe our construction without the error parameters; then we will explain how to set the error parameter for the construction instance over $\mathcal{DA}$.

Let $(\boldsymbol{A}, \boldsymbol{c}^A)$ be an instance over $\mathcal{G}_{z,2}$. The idea is to write each equation in $(\boldsymbol{A}, \boldsymbol{c}^A)$ as a set of difference equations and average equations, via bitwise decomposition. Same as [KZ20], we first explain the idea by an example:

$$3\boldsymbol{x}_1 + 5\boldsymbol{x}_2 - \boldsymbol{x}_3 - 7\boldsymbol{x}_4 = 1. \tag{12}$$

We will manipulate the variables with positive coefficients and the variables with negative coefficients separately. Let us take the positive coefficients as an example. We pair all the variables with the odd positive coefficients and replace each pair with a new variable via an average equation. In this example, we pair $\boldsymbol{x}_1, \boldsymbol{x}_2$; we then create a new variable $\boldsymbol{x}_{12}$ and a new average equation

$$\boldsymbol{x}_1 + \boldsymbol{x}_2 - 2\boldsymbol{x}_{12} = 0.$$

Plugging this into Eq. (12), we get

$$2\boldsymbol{x}_1 + 4\boldsymbol{x}_2 + 2\boldsymbol{x}_{12} - \boldsymbol{x}_3 - 7\boldsymbol{x}_4 = 1.$$

We pull out a factor 2:

$$2(\boldsymbol{x}_1 + 2\boldsymbol{x}_2 + \boldsymbol{x}_{12}) - \boldsymbol{x}_3 - 7\boldsymbol{x}_4 = 1.$$

We repeat the pair-and-replace process to pair $\boldsymbol{x}_1, \boldsymbol{x}_{12}$ with a new variables $\boldsymbol{x}_{1,12}$ and a new average equation $\boldsymbol{x}_1 + \boldsymbol{x}_{12} - 2\boldsymbol{x}_{1,12} = 0$:

$$2(2\boldsymbol{x}_{1,12} + 2\boldsymbol{x}_2) - \boldsymbol{x}_3 - 7\boldsymbol{x}_4 = 1.$$

Pull out a factor 2:

$$4(\boldsymbol{x}_{1,12} + \boldsymbol{x}_2) - \boldsymbol{x}_3 - 7\boldsymbol{x}_4 = 1.$$

Repeat pair-and-replace with $\boldsymbol{x}_{1,12} + \boldsymbol{x}_2 - 2\boldsymbol{x}_{1,12,2} = 0$:

$$8\boldsymbol{x}_{1,12,2} - \boldsymbol{x}_3 - 7\boldsymbol{x}_4 = 1.$$

Similarly, we can use a sequence of average equations for the variables with odd coefficients, and the above equation becomes:

$$8\boldsymbol{x}_{1,12,2} - 8\boldsymbol{x}_{34} = 1.$$

where $\boldsymbol{x}_{34}$ is a new variable. The final equation is a difference equation.

The above reduction relies on that the sum of all the coefficients is 0 and the sum of all the positive coefficients are a power of 2.

Pseudo-code of the reduction from linear equation problems over $\mathcal{G}_{z,2}$ to $\mathcal{DA}$ is shown in Algorithm 7. Algorithm 8 shows how to map a solution to the instance over $\mathcal{DA}$ back to a solution to the original instance over $\mathcal{G}_{z,2}$.

[11] Note that given two singleton multi-sets each containing a single equation, e.g. $\{\boldsymbol{a}_1^\top \boldsymbol{x} = c_1\}$ and $\{\boldsymbol{a}_2^\top \boldsymbol{x} = c_2\}$ where $\boldsymbol{a}_1, \boldsymbol{a}_2$ are vectors, we define $\{\boldsymbol{a}_1^\top \boldsymbol{x} = c_1\} + \{\boldsymbol{a}_2^\top \boldsymbol{x} = c_2\} = \{\boldsymbol{a}_1^\top \boldsymbol{x} + \boldsymbol{a}_2^\top \boldsymbol{x} = c_1 + c_2\}$ and we define $\{\boldsymbol{a}_1^\top \boldsymbol{x} = c_1\} \cup \{\boldsymbol{a}_2^\top \boldsymbol{x} = c_2\} = \{\boldsymbol{a}_1^\top \boldsymbol{x} = c_1, \boldsymbol{a}_2^\top \boldsymbol{x} = c_2\}$.

Algorithm 7: REDUCE $\mathcal{G}_{\text{z},2}$ TO $\mathcal{DA}$

Input: $(\boldsymbol{A}, \boldsymbol{c}^{\text{A}})$ where $\boldsymbol{A} \in \mathcal{G}_{\text{z},2}$ is an $m \times n$ matrix, $\boldsymbol{c}^{\text{A}} \in \mathbb{R}^n$, and $\alpha > 0$.

Output: $(\boldsymbol{B}, \boldsymbol{c}^{\text{B}})$ where $\boldsymbol{B} \in \mathcal{DA}$ is an $m \times n$ matrix, and $\boldsymbol{c}^{\text{B}} \in \mathbb{R}^n$.

$\mathcal{X} \leftarrow \{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n\}$; `// DA variables and index of new variables`

Let $\boldsymbol{x}$ be the vector of variables corresponding to the set of variables $\mathcal{X}$

$t \leftarrow n + 1$

$\mathcal{A} \leftarrow \emptyset$, $\mathcal{B} \leftarrow \emptyset$; `// Multisets of main and DA auxiliary equations respectively`

for *each equation* $1 \le i \le m$ *in* $\boldsymbol{A}$ **do**

  if $\boldsymbol{A}_i$ *is already a difference equation or an average equation* **then**

    $\mathcal{A} \leftarrow \mathcal{A} \cup \{\boldsymbol{A}_{ij_+}\boldsymbol{u}_{j_+} - \boldsymbol{A}_{ij_+}\boldsymbol{u}_{j_-} = \boldsymbol{c}_i\}$

    $w_i \leftarrow \frac{\alpha}{\alpha+1}$

  end

  else

    Let $\mathcal{I}^{+1} \leftarrow \{j : \boldsymbol{A}_{ij} > 0\}, \mathcal{I}^{-1} \leftarrow \{j : \boldsymbol{A}_{ij} < 0\}$

    $\mathcal{A}_i \leftarrow \{\boldsymbol{A}_i\boldsymbol{u} = \boldsymbol{c}_i\}$

    $\mathcal{B}_i \leftarrow \emptyset$

    for $s = -1, +1$ **do**

      $r \leftarrow 0$

      while $\mathcal{A}_i$ *is neither a difference equation nor an average equation* **do**

        For each j, let $\widehat{\boldsymbol{A}}_{ij}$ be the coefficient of $\boldsymbol{u}_j$ in $\mathcal{A}_i$.

        $\mathcal{I}^s_{odd} \leftarrow \{j \in \mathcal{I}^s : \lfloor |\widehat{\boldsymbol{A}}_{ij}|/2^r \rfloor \text{ is odd}\}$

        Pair the indices of $\mathcal{I}^s_{odd}$ into disjoint pairs $(j_1, l_1), (j_2, l_2), \ldots$

        for *each pair of indices* (j_k, l_k) **do**

          $\mathcal{A}_i \leftarrow \mathcal{A}_i + s \cdot 2^r \{(2\boldsymbol{u}_t - (\boldsymbol{u}_{j_k} + \boldsymbol{u}_{l_k})) = 0\}$ [11]

          $\mathcal{B}_i \leftarrow \mathcal{B}_i \cup s \cdot 2^r \{(2\boldsymbol{u}_t - (\boldsymbol{u}_{j_k} + \boldsymbol{u}_{l_k})) = 0\}$

          $\mathcal{X} \leftarrow \mathcal{X} \cup \{\boldsymbol{u}_t\}$, update $\boldsymbol{x}$ accordingly

          $t \leftarrow t + 1$

          $r \leftarrow r + 1$

        end

      end

    end

    $w_i \leftarrow \alpha\,|\mathcal{B}_i|$

    $\mathcal{B} \leftarrow \mathcal{B} \cup w_i^{1/2} \cdot \mathcal{B}_i$

    $\mathcal{A} \leftarrow \mathcal{A} \cup \mathcal{A}_i$.

  end

end

return $\boldsymbol{B}, \boldsymbol{c}$ *s.t.* $\boldsymbol{B}\boldsymbol{x} = \boldsymbol{c}$ *corresponds to the equations in* $\mathcal{A} \cup \mathcal{B}$*, on the variable set* $\mathcal{X}$.

Notations. We will follow the notations in [KZ20]. The central object created by Algorithm 7 is the matrix $\boldsymbol{B}$, which contains both original and new equations and variables. We will superscript the variables with ^{A} to distinguish variables appear in the original equation $\boldsymbol{A}\boldsymbol{x}^{\text{A}} = \boldsymbol{c}^{\text{A}}$ from new variables. We will term the new variables as $\boldsymbol{x}^{\text{aux}}$, and write a vector solution to the new problem, $\boldsymbol{x}^{\text{B}}$, as:

$$\boldsymbol{x}^{\text{B}} = \begin{bmatrix} \boldsymbol{x}^{\text{A}} \\ \boldsymbol{x}^{\text{aux}} \end{bmatrix}. \tag{13}$$

Let n_A be the dimension of $\boldsymbol{x}^{\mathrm{A}}$, and n_B be the dimension of $\boldsymbol{x}^{\mathrm{B}}$, respectively.

Furthermore, we will distinguish the equations in $\boldsymbol{B}$ into ones formed from manipulating $\boldsymbol{A}$, i.e. the equations added to the set $\mathcal{A}$, from the auxiliary equations, i.e. the equations added to the set $\mathcal{B}$. We use $\boldsymbol{W}^{1/2} = \mathrm{diag}(w_i^{1/2})$ to refer to the diagonal matrix of weights w_i applied to the auxiliary equations $\mathcal{B}$ in Algorithm 7. In Algorithm 7, a real value $\alpha > 0$ is set initially and used when computing the weights $w_i^{1/2}$. For convenience, throughout most of this section, we will treat α as an arbitrary constant, and only eventually substitute in its value to complete our main proof. This leads to the following representation of $\boldsymbol{B}$ and $\boldsymbol{c}^{\mathrm{B}}$ which we will use throughout our analysis of the algorithm:

$$\boldsymbol{B} = \begin{bmatrix} \widehat{\boldsymbol{A}} \\ \boldsymbol{W}^{1/2}\widehat{\boldsymbol{B}} \end{bmatrix}. \tag{14}$$

Here the equations of $\widehat{\boldsymbol{A}}$ corresponds to $\mathcal{A}$ in Reduce $\mathcal{G}_{\mathrm{z},2}to\mathcal{DA}$, and $\widehat{\boldsymbol{B}}$ corresponds to the auxiliary constraints, i.e. equations of $\mathcal{B}$ in Reduce $\mathcal{G}_{\mathrm{z},2}to\mathcal{DA}$. Also, the vector $\boldsymbol{c}^{\mathrm{B}}$ created is simply an extension of $\boldsymbol{c}^{\mathrm{A}}$:

$$\boldsymbol{c}^{\mathrm{B}} = \begin{bmatrix} \boldsymbol{c}^{\mathrm{A}} \\ \boldsymbol{0} \end{bmatrix}. \tag{15}$$

Finally, as Algorithm 7 creates new equations for each row of $\boldsymbol{A}$ independently, we will use S_i to denote the subset of indices of the rows of $\widehat{\boldsymbol{B}}$ that is created from $\boldsymbol{A}_i$, and denote $m_i \stackrel{\text{def}}{=} |S_i|$, and use $\widehat{\boldsymbol{B}}_i$ to denote these part of $\widehat{\boldsymbol{B}}$ that corresponds to these rows.

We observe that the sets of auxiliary variables created for $\widehat{\boldsymbol{B}}_i$'s are disjoint.

Algorithm 8: MapSoln $\mathcal{DA}$to$\mathcal{G}_{\mathrm{z},2}$

Input: $m \times n$ matrix $\boldsymbol{A} \in \mathcal{G}_{\mathrm{z},2}$, $m' \times n'$ matrix $\boldsymbol{B} \in \mathcal{DA}$, vector $\boldsymbol{c}^{\mathrm{A}} \in \mathbb{R}^m$, vector $\boldsymbol{x}^{\mathrm{B}} \in \mathbb{R}^{n'}$.
Output: Vector $\boldsymbol{x}^{\mathrm{A}} \in \mathbb{R}^n$.

```
if A^⊤ c^A = 0 then
    return x^A ← 0
end
else
    return x^A ← x^B_{1:n}
end
```

Lemma A.1. *Let $\boldsymbol{A} \in \mathcal{G}_{z,2}$, and let $\boldsymbol{B}$ be returned by running Algorithm* Reduce $\mathcal{G}_{z,2}$to$\mathcal{DA}$ *on $\boldsymbol{A}$. Then,*

- $\mathrm{nnz}(\boldsymbol{B}) = O(\mathrm{nnz}(\boldsymbol{A}) \log \|\boldsymbol{A}\|_{\max})$;
- *both the dimensions of $\boldsymbol{B}$ are $O(\mathrm{nnz}(\boldsymbol{A}) \log \|\boldsymbol{A}\|_{\max})$.*

Proof sketch. The second statement is implied by the first one.

To prove the first statement, we focus on Algorithm Reduce $\mathcal{G}_{\mathrm{z},2}to\mathcal{DA}$ for a single equation in $\boldsymbol{A}_i$. The number of the while-iterations from line 16 to 27 is at most $\log \|\boldsymbol{A}\|_{\max}$. We observe that (1) in each iteration, the number of newly created variables and equations is at most the value of $\mathrm{nnz}(\mathcal{A}_i)$ at the beginning of the iteration; (2) once an auxiliary variable is added to $\mathcal{A}_i$, it must be replaced with a new auxiliary variable in the next iteration (if exists). Therefore, $\mathrm{nnz}(\mathcal{A}_i) = O(\mathrm{nnz}(\boldsymbol{A}_i))$ for every iteration. So, $\mathrm{nnz}(\boldsymbol{B}) = O(\mathrm{nnz}(\boldsymbol{A}) \log \|\boldsymbol{A}\|_{\max})$. □

A.2 Relation between exact solvers

The following lemma characterizes the relation between the error of a solution to $(\boldsymbol{A}, \boldsymbol{c}^{\mathrm{A}})$ to that to $(\boldsymbol{B}, \boldsymbol{c}^{\mathrm{B}})$. We will exploit it to derive relations between exactly and approximately solving $(\boldsymbol{A}, \boldsymbol{c}^{\mathrm{A}})$ and $(\boldsymbol{B}, \boldsymbol{c}^{\mathrm{B}})$, and to establish a bound for the condition number of the new constructed matrix $\boldsymbol{B}$.

Lemma A.2. *For any fixed $\boldsymbol{x}^A \in \mathbb{R}^n$,*

$$\left\|\boldsymbol{A}\boldsymbol{x}^A - \boldsymbol{c}^A\right\|_2^2 = \frac{\alpha+1}{\alpha} \min_{\boldsymbol{x}^{aux}} \left\| \boldsymbol{B} \begin{bmatrix} \boldsymbol{x}^A \\ \boldsymbol{x}^{aux} \end{bmatrix} - \boldsymbol{c}^B \right\|_2^2.$$

Lemma A.2 immediately implies the following corollaries.

Corollary A.3. *Given an* $\textsc{le}(\boldsymbol{A}, \boldsymbol{c}^A)$ *where* $\boldsymbol{A} \in \mathcal{G}_{z,2}$, *let* $\textsc{le}(\boldsymbol{B}, \boldsymbol{c}) \leftarrow \textsc{Reduce}\, \mathcal{G}_{z,2} \textsc{to} \mathcal{DA}(\boldsymbol{A}, \boldsymbol{c}^A, 0)$. *Then,* $\boldsymbol{B} \in \mathcal{DA}$, *and if* $\boldsymbol{x}^B = \begin{bmatrix} \boldsymbol{x}^A \\ \boldsymbol{x}^{aux} \end{bmatrix}$ *is a solution to* $\textsc{le}(\boldsymbol{B}, \boldsymbol{c}^B)$, *then* $\boldsymbol{x}^A$ *is a solution to* $\textsc{le}(\boldsymbol{A}, \boldsymbol{c}^A)$.

Corollary A.4.

$$\min_{\boldsymbol{x}^A} \left\|\boldsymbol{A}\boldsymbol{x}^A - \boldsymbol{c}^A\right\|_2^2 = \frac{\alpha+1}{\alpha} \min_{\boldsymbol{x}^B} \left\|\boldsymbol{B}\boldsymbol{x}^B - \boldsymbol{c}^B\right\|_2^2.$$

The optimal solutions to $\min_{\boldsymbol{x}^{\mathrm{B}}} \left\|\boldsymbol{B}\boldsymbol{x}^{\mathrm{B}} - \boldsymbol{c}^{\mathrm{B}}\right\|_2$ and $\min_{\boldsymbol{x}^{\mathrm{A}}} \left\|\boldsymbol{A}\boldsymbol{x}^{\mathrm{A}} - \boldsymbol{c}^{\mathrm{A}}\right\|_2$ have a one-to-one map. However, the optimal values are different; when $\alpha \to \infty$, the two optimal values approach the same value.

It remains to prove Lemma A.2.

Proof of Lemma A.2. Fix $\boldsymbol{x}^A$, and let $\boldsymbol{x}^{\mathrm{aux}}$ be arbitrary. For each $i \in [m]$ and $j \in S_i$, the set of the auxiliary equations created for $\boldsymbol{A}_i$, we denote

$$\epsilon_i \stackrel{\text{def}}{=} \boldsymbol{A}_i \boldsymbol{x}^{\mathrm{A}} - \boldsymbol{c}_i^{\mathrm{A}},\ \hat{\delta}_i \stackrel{\text{def}}{=} \widehat{\boldsymbol{A}}_i \boldsymbol{x}^{\mathrm{B}} - \boldsymbol{c}_i^{\mathrm{A}},\ \delta_j \stackrel{\text{def}}{=} \widehat{\boldsymbol{B}}_j \boldsymbol{x}^{\mathrm{B}}.$$

Then

$$\epsilon_i = \hat{\delta}_i + \sum_{j \in S_i} \delta_j$$

By the Cauchy-Schwarz inequality and $w_i = \alpha m_i$,

$$\epsilon_i^2 = \left(\hat{\delta}_i + \sum_{j \in S_i} \delta_j\right)^2 \le \left(\hat{\delta}_i^2 + w_i \sum_{j \in S_i} \delta_j^2\right)\left(1 + \frac{m_i}{w_i}\right) = \left(\hat{\delta}_i^2 + w_i \sum_{j \in S_i} \delta_j^2\right) \cdot \frac{\alpha+1}{\alpha}$$

that is,

$$(\boldsymbol{A}_i \boldsymbol{x}^A - \boldsymbol{c}_i^A)^2 \le \frac{\alpha+1}{\alpha} \left\| \begin{bmatrix} \widehat{\boldsymbol{A}}_i \\ \boldsymbol{W}_i^{1/2} \widehat{\boldsymbol{B}}_i \end{bmatrix} \boldsymbol{x}^{\mathrm{B}} - \begin{bmatrix} \boldsymbol{c}_i^{\mathrm{A}} \\ \boldsymbol{0} \end{bmatrix} \right\|_2^2.$$

The equality holds if and only if

$$\hat{\delta}_i = w_i \delta_j, \qquad \forall j \in S_i. \tag{16}$$

Sum over all the rows $i \in [n_A]$:

$$\left\|\boldsymbol{A}\boldsymbol{x}^{\mathrm{A}} - \boldsymbol{c}^{\mathrm{A}}\right\|_2^2 \le \frac{\alpha+1}{\alpha} \left\| \boldsymbol{B} \begin{bmatrix} \boldsymbol{x}^{\mathrm{A}} \\ \boldsymbol{x}^{\mathrm{aux}} \end{bmatrix} - \boldsymbol{c}^{\mathrm{B}} \right\|_2. \tag{17}$$

It remains to show when we take the minimum over $\boldsymbol{x}^{\mathrm{aux}}$, the right-hand side of Eq. (17) equals the left-hand side. That is, for any fixed $\boldsymbol{x}^{\mathrm{A}}$, there exists $\boldsymbol{x}^{\mathrm{aux}}$ such that Eq. (16) holds.

In particular, we will momentarily prove the following Claim.

Claim A.5. *For any fixed $\boldsymbol{x}^A$ and its associated error ϵ_i for each row i of $\boldsymbol{A}$ such that $\boldsymbol{A}_i$ is neither a difference equation nor an average equation, the following linear system*

$$\widehat{\boldsymbol{A}}_i \begin{bmatrix} \boldsymbol{x}^A \\ \boldsymbol{x}^{aux} \end{bmatrix} = \boldsymbol{c}_i^A + \frac{\alpha}{\alpha+1}\epsilon_i, \tag{18}$$

$$\widehat{\boldsymbol{B}}_j \begin{bmatrix} \boldsymbol{x}^A \\ \boldsymbol{x}^{aux} \end{bmatrix} = \frac{1}{(\alpha+1)m_i}\epsilon_i, \forall j \in S_i. \tag{19}$$

has a solution (which may not be unique).

Since every auxiliary variable is associated with only one row i of $\boldsymbol{A}$, Claim A.5 implies that we can choose $\boldsymbol{x}^{\text{aux}}$ s.t. all these linear subsystems are satisfied simultaneously. Given such a choice of $\boldsymbol{x}^{\text{aux}}$, Eq. (16) is satisfied and thus

$$\left\|\boldsymbol{A}\boldsymbol{x}^{\text{A}} - \boldsymbol{c}^{\text{A}}\right\|_2^2 = \frac{\alpha+1}{\alpha}\left\|\boldsymbol{B}\begin{bmatrix}\boldsymbol{x}^{\text{A}} \\ \boldsymbol{x}^{\text{aux}}\end{bmatrix} - \boldsymbol{c}^{\text{B}}\right\|_2^2.$$

Together with Eq. (17), this completes the proof of Lemma A.2. □

Proof of Claim A.5. We will construct an assignment to all the variables of $\boldsymbol{x}^{\text{aux}}$ such that Eq. (18) and (19) are satisfied. We start with an assignment $\boldsymbol{x}^{\text{A}}$ to the main variables, and we then assign values to auxiliary variables in the order that they are created by the algorithm REDUCE $\mathcal{G}_{\text{z},2}$TO$\mathcal{DA}$. Note that we will refer to variables j_k and l_k only in the context of a fixed value of t, which always ensures that they are unambiguously defined. When the algorithm processes pair $\boldsymbol{x}^{\text{B}}_{j_k}, \boldsymbol{x}^{\text{B}}_{l_k} = \boldsymbol{u}_{j_k}, \boldsymbol{u}_{l_k}$, the value of these variables will have been set already, but $\boldsymbol{x}^{\text{B}}_t = \boldsymbol{u}_t$ and the other newly created auxiliary variables have not. Suppose the new equation we added to $\mathcal{B}_i$ is $s{\cdot}2^r\{(2\boldsymbol{u}_t - (\boldsymbol{u}_{j_k} + \boldsymbol{u}_{l_k})) = 0\}$ in line 22. We simply assign $\boldsymbol{u}_t$ such that

$$s \cdot 2^r\left(2\boldsymbol{u}_t - (\boldsymbol{u}_{j_k} + \boldsymbol{u}_{l_k})\right) = \frac{1}{(\alpha+1)m_i}\epsilon_i.$$

Every auxiliary variable is associated with only one row, so we never get multiple assignments to a variable using this procedure.

By the above setting, Eq. (19) is satisfied. It remains to check Eq. (18). Sum up all the above equations over the auxiliary equations in $\mathcal{B}_i$ with minus sign and the original equation $\boldsymbol{A}_i\boldsymbol{x}^A = \boldsymbol{c}_i^A + \epsilon_i$:

$$\widehat{\boldsymbol{A}}_i\boldsymbol{x} = \boldsymbol{c}_i^A + \epsilon_i - \frac{1}{\alpha+1}\epsilon_i = \boldsymbol{c}_i^A + \frac{\alpha}{\alpha+1}\epsilon_i.$$

This completes the proof. □

A.2.1 Relation to Schur Complements

Note that we can write $\boldsymbol{B}$ as $(\boldsymbol{B}^A \;\; \boldsymbol{B}^{aux})$, where $\boldsymbol{B}^A$ corresponds to the original variables and $\boldsymbol{B}^{aux}$ the auxiliary variables. Then,

$$\boldsymbol{B}^\top\boldsymbol{B} = \begin{bmatrix} (\boldsymbol{B}^A)^\top\boldsymbol{B}^A & (\boldsymbol{B}^A)^\top\boldsymbol{B}^{aux} \\ (\boldsymbol{B}^{aux})^\top\boldsymbol{B}^A & (\boldsymbol{B}^{aux})^\top\boldsymbol{B}^{aux} \end{bmatrix}.$$

Lemma A.2 essentially states that $\frac{\alpha}{\alpha+1}\boldsymbol{A}^\top\boldsymbol{A}$ is the Schur complement of $(\boldsymbol{B}^{aux})^\top\boldsymbol{B}^{aux}$ of $\boldsymbol{B}\boldsymbol{B}^\top$. See the following definition and a fact of Schur complement.

Definition A.6 (Schur complement)**.** Let $\boldsymbol{C} \in \mathbb{R}^{n \times n}$ be a 2×2 block matrix: $\boldsymbol{C} = \begin{pmatrix} \boldsymbol{C}_{11} & \boldsymbol{C}_{12} \\ \boldsymbol{C}_{12}^\top & \boldsymbol{C}_{22} \end{pmatrix}$ The *Schur complement* of the block $\boldsymbol{C}_{22}$ of $\boldsymbol{C}$ is

$$\boldsymbol{C}/\boldsymbol{C}_{22} \stackrel{\text{def}}{=} \boldsymbol{C}_{11} - \boldsymbol{C}_{12}\boldsymbol{C}_{22}^{-1}\boldsymbol{C}_{12}^\top.$$

If $\boldsymbol{C}_{22}$ is not invertible, then we replace the inverse with the pseudo-inverse

Schur complement arises from block Gaussian elimination. Schur complement has the following important fact.

Fact A.7 (Schur complement and minimizer)**.** *For any fixed vector $\boldsymbol{x}$,*

$$\min_{\boldsymbol{y}} \begin{pmatrix} \boldsymbol{x}^\top & \boldsymbol{y}^\top \end{pmatrix} \boldsymbol{C} \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{pmatrix} = \boldsymbol{x}^\top (\boldsymbol{C}/\boldsymbol{C}_{22}) \boldsymbol{x}.$$

Proof. We expand the left hand side,

$$\begin{pmatrix} \boldsymbol{x}^\top & \boldsymbol{y}^\top \end{pmatrix} \boldsymbol{C} \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{pmatrix} = \boldsymbol{x}^\top \boldsymbol{C}_{11} \boldsymbol{x} + 2\boldsymbol{x}^\top \boldsymbol{C}_{12} \boldsymbol{y} + \boldsymbol{y}^\top \boldsymbol{C}_{22} \boldsymbol{y}. \tag{20}$$

Taking derivative w.r.t. $\boldsymbol{y}$ and setting it to be 0 give that

$$2\boldsymbol{C}_{22}\boldsymbol{y} + 2\boldsymbol{C}_{12}^\top \boldsymbol{x} = \boldsymbol{0}.$$

Plugging $\boldsymbol{y} = -\boldsymbol{C}_{22}^\dagger \boldsymbol{C}_{12}^\top \boldsymbol{x}$ into Eq. (20),

$$\min_{\boldsymbol{y}} \begin{pmatrix} \boldsymbol{x}^\top & \boldsymbol{y}^\top \end{pmatrix} \boldsymbol{C} \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{pmatrix} = \boldsymbol{x}^\top \boldsymbol{C}_{11} \boldsymbol{x} - \boldsymbol{x}^\top \boldsymbol{C}_{12} \boldsymbol{C}_{22}^\dagger \boldsymbol{C}_{12}^\top \boldsymbol{x} = \boldsymbol{x}^\top (\boldsymbol{C}/\boldsymbol{C}_{22}) \boldsymbol{x}.$$

This completes the proof. □

A.3 Relation between approximate solvers

We now show that approximate solvers for $\boldsymbol{B}$ also translate to approximate solvers for $\boldsymbol{A}$.

Lemma A.8. *Let* $\textsc{lea}(\boldsymbol{A}, \boldsymbol{c}^A, \epsilon^A)$ *be an instance over* $\mathcal{G}_{z,2}$ *where* $\epsilon^A \in (0,1)$*. Let* $(\boldsymbol{B}, \boldsymbol{c}^B) \leftarrow \textsc{Reduce}\, \mathcal{G}_{z,2} \textsc{to} \mathcal{DA}(\boldsymbol{A}, \boldsymbol{c}^A, \alpha = 1)$*, and*

$$\epsilon^{B_2} \leq \frac{\epsilon^A}{\sqrt{n_A m_A} \cdot \|\boldsymbol{A}\|_{\max} \|\boldsymbol{c}^A\|_2}.$$

Suppose $\boldsymbol{x}^B$ *is a solution to* $\textsc{lea}(\boldsymbol{B}, \boldsymbol{c}^B, \epsilon^B)$*, and* $\boldsymbol{x}^A \leftarrow \textsc{MapSoln}\, \mathcal{DA} \textsc{to} \mathcal{G}_{z,2}(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{c}^A, \boldsymbol{x}^B)$*. Then,* $\boldsymbol{x}^A$ *is a solution to* $\textsc{lea}(\boldsymbol{A}, \boldsymbol{c}^A, \epsilon^A)$*.*

Proof. Write $\boldsymbol{x}^{\mathrm{B}} = \begin{bmatrix} \boldsymbol{x}^{\mathrm{A}} \\ \boldsymbol{x}^{\mathrm{aux}} \end{bmatrix}$. By Lemma A.2,

$$\left\|\boldsymbol{A}\boldsymbol{x}^A - \boldsymbol{c}^{\mathrm{A}}\right\|_2^2 \leq \frac{\alpha}{\alpha+1} \left\|\boldsymbol{B}\boldsymbol{x}^{\mathrm{B}} - \boldsymbol{c}^{\mathrm{B}}\right\|_2^2$$
$$\left\|(\boldsymbol{I} - \boldsymbol{\Pi}_{\boldsymbol{A}})\boldsymbol{c}^A\right\|_2^2 = \frac{\alpha}{\alpha+1} \left\|(\boldsymbol{I} - \boldsymbol{\Pi}_{\boldsymbol{B}})\boldsymbol{c}^B\right\|_2^2$$

where $\alpha = 1$ in our setting. Then,

$$\begin{aligned}
\left\|\boldsymbol{A}\boldsymbol{x}^A - \boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{c}^A\right\|_2^2 &\le \frac{\alpha}{\alpha+1}\left\|\boldsymbol{B}\boldsymbol{x}^{\mathrm{B}} - \boldsymbol{\Pi}_{\boldsymbol{B}}\boldsymbol{c}^B\right\|_2^2 \\
&\le \frac{\alpha}{\alpha+1}(\epsilon^B)^2\left\|\boldsymbol{\Pi}_{\boldsymbol{B}}\boldsymbol{c}^{\mathrm{B}}\right\|_2^2 \\
&\le \frac{\alpha}{\alpha+1}(\epsilon^B)^2\left\|\boldsymbol{c}^{\mathrm{B}}\right\|_2^2\lambda_{\max}(\boldsymbol{A}^\top\boldsymbol{A})\left\|\boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{c}^{\mathrm{A}}\right\|_2^2 && \text{by Lemma 6.5} \\
&\le (\epsilon^{\mathrm{A}})^2\left\|\boldsymbol{\Pi}_{\boldsymbol{A}}\boldsymbol{c}^{\mathrm{A}}\right\|_2^2 && \text{since } \lambda_{\max}(\boldsymbol{A}^\top\boldsymbol{A}) \le n_A m_A\left\|\boldsymbol{A}\right\|_{\max}^2
\end{aligned}$$

□

A.4 The condition number of the new matrix

In this section, we show that the condition number of $\boldsymbol{B}$ is upper bounded by the condition number of $\boldsymbol{A}$ up to a poly(n) multiplicative factor.

Lemma A.9. *Let* $(\boldsymbol{B}, \boldsymbol{c}^B) \leftarrow$ Reduce $\mathcal{G}_{z,2}$to$\mathcal{DA}(\boldsymbol{A}, \boldsymbol{c}^A, \alpha = 1)$. *Then,* $\kappa(\boldsymbol{B}) = O\left(\max\left\{\frac{\mathrm{nnz}(\boldsymbol{A})^{7/2}\|\boldsymbol{A}\|_{\max}\log^2\|\boldsymbol{A}\|_{\max}}{\lambda_{\min}(\boldsymbol{A}^\top\boldsymbol{A})},\ \mathrm{nnz}(\boldsymbol{A})\right\}\right)$.

Note that $\lambda_{\min}(\boldsymbol{A}^\top\boldsymbol{A})$ is polynomially bounded, since both $\kappa(\boldsymbol{A}^\top\boldsymbol{A})$ and $\lambda_{\max}(\boldsymbol{A}^\top\boldsymbol{A})$ are polynomially bounded.

A.4.1 The maximum eigenvalue

We start with a simple observation.

Claim A.10. $\max_i \boldsymbol{W}_{i,i} = O(\mathrm{nnz}(\boldsymbol{A})\log\|\boldsymbol{A}\|_{\max})$.

We bound the maximum eigenvalue of the constructed matrix $\boldsymbol{B}$.

Lemma A.11. *Let* $\boldsymbol{B} \in \mathbb{R}^{m_B\times n_B}$ *be returned by* Reduce $\mathcal{G}_{z,2}$to$\mathcal{DA}(\boldsymbol{A}, \boldsymbol{c}^A, \alpha = 1)$. *Then,* $\lambda_{\max}(\boldsymbol{B}^\top\boldsymbol{B}) = O(\mathrm{nnz}(\boldsymbol{A})^2\|\boldsymbol{A}\|_{\max}\log\|\boldsymbol{A}\|_{\max})$.

Proof. Write $\boldsymbol{B} = \widetilde{\boldsymbol{W}}^{1/2}\widetilde{\boldsymbol{B}}$. By the Courant-Fischer theorem,

$$\lambda_{\max}(\boldsymbol{B}^\top\boldsymbol{B}) = \max_{\boldsymbol{x}:\|\boldsymbol{x}\|_2=1}\boldsymbol{x}^\top\widetilde{\boldsymbol{B}}^\top\widetilde{\boldsymbol{W}}\widetilde{\boldsymbol{B}}\boldsymbol{x} = \max_{\boldsymbol{x}:\|\boldsymbol{x}\|_2=1}\sum_{i=1}^{m_B}\widetilde{\boldsymbol{W}}_{i,i}(\widetilde{\boldsymbol{B}}_i\boldsymbol{x})^2.$$

Note that

$$\begin{aligned}
&\widetilde{\boldsymbol{W}}_{i,i} = O(\mathrm{nnz}(\boldsymbol{A})\log\|\boldsymbol{A}\|_{\max}), \\
&(\widetilde{\boldsymbol{B}}_i\boldsymbol{x})^2 \le 6\|\boldsymbol{A}\|_{\max}\sum_{j=1}^{n_B}\boldsymbol{x}_j^2\cdot\left|\{i\in[m_B] : \boldsymbol{B}_{ij}\neq 0\}\right|.
\end{aligned}$$

Thus,

$$\lambda_{\max}(\boldsymbol{B}^\top\boldsymbol{B}) = O(\mathrm{nnz}(\boldsymbol{A})^2\|\boldsymbol{A}\|_{\max}\log\|\boldsymbol{A}\|_{\max}).$$

□

A.4.2 The minimum nonzero eigenvalue

To apply the Courant-Fischer theorem for the minimum nonzero eigenvalue of $\boldsymbol{B}$, we need to first characterize the null space of $\boldsymbol{B}$. Given any $\boldsymbol{x}^{\mathrm{A}} \in \mathrm{null}(\boldsymbol{A})$ with dimensions n_A, we extend $\boldsymbol{x}^{\mathrm{A}}$ to a vector in dimension n_B:

$$\boldsymbol{p}(\boldsymbol{x}^{\mathrm{A}}) \stackrel{\mathrm{def}}{=} \begin{bmatrix} \boldsymbol{x}^{\mathrm{A}} \\ \boldsymbol{x}^{\mathrm{aux}} \end{bmatrix}. \tag{21}$$

where we assign the values of the auxiliary variables $\boldsymbol{x}^{\mathrm{aux}}$ in the order that they are created in Algorithm 7. In an auxiliary equation created in line 22, $\boldsymbol{u}_{j_k}$ and $\boldsymbol{u}_{l_k}$ have already been assigned, and we simply assign $\boldsymbol{u}_t$ to such that the new equation holds.

Lemma A.12. $\mathrm{null}(\boldsymbol{B}) = \mathrm{span}\{\boldsymbol{p}(\boldsymbol{x}^A) : \boldsymbol{x}^A \in \mathrm{null}(\boldsymbol{A})\}$.

Proof. Let $S = \mathrm{span}\{\boldsymbol{p}(\boldsymbol{x}^{\mathrm{A}}) : \boldsymbol{x}^{\mathrm{A}} \in \mathrm{null}(\boldsymbol{A})\}$. We can check that $\mathrm{null}(\boldsymbol{B}) \supseteq S$. Then, for any $\boldsymbol{x} \in \mathrm{null}(\boldsymbol{B})$, by Lemma A.2 with $\boldsymbol{c}^{\mathrm{A}} = \boldsymbol{0}$, we have $\boldsymbol{A}\boldsymbol{x}^{\mathrm{A}} = \boldsymbol{0}$, that is, $\boldsymbol{x}^{\mathrm{A}} \in \mathrm{null}(\boldsymbol{A})$. Thus, $\mathrm{null}(\boldsymbol{B}) \subseteq S$. □

Lemma A.13. $\lambda_{\min}(\boldsymbol{B}^\top \boldsymbol{B}) = \Omega\left(\min\left\{\frac{\lambda_{\min}(\boldsymbol{A}^\top \boldsymbol{A})}{\mathrm{nnz}(\boldsymbol{A})^{3/2}\log\|\boldsymbol{A}\|_{\max}},\ \mathrm{nnz}(\boldsymbol{A})\,\|\boldsymbol{A}\|_{\max}\log\|\boldsymbol{A}\|_{\max}\right\}\right)$.

Proof. Let $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}^{\mathrm{A}} \\ \boldsymbol{x}^{\mathrm{aux}} \end{bmatrix} \in \mathbb{R}^{n_B}$ be an arbitrary unit vector orthogonal to $\mathrm{null}(\boldsymbol{B})$, let $\boldsymbol{y} = \boldsymbol{x} - \boldsymbol{p}(\boldsymbol{x}^A)$, and let

$$\delta \stackrel{\mathrm{def}}{=} \min\left\{\frac{\lambda_{\min}(\boldsymbol{A}^\top \boldsymbol{A})}{C\,\mathrm{nnz}(\boldsymbol{A})^{5/2}\,\|\boldsymbol{A}\|_{\max}\log\|\boldsymbol{A}\|_{\max}},\ \frac{1}{5}\right\}.$$

where C is a constant such that $\lambda_{\max}(\boldsymbol{B}^\top \boldsymbol{B}) \le C\,\mathrm{nnz}(\boldsymbol{A})^2\,\|\boldsymbol{A}\|_{\max}\log\|\boldsymbol{A}\|_{\max}$ (by Lemma A.11, such C exists).

We will prove the statement by exhausting the cases of $\|\boldsymbol{y}\|_\infty$.

Suppose $\|\boldsymbol{y}\|_\infty > \delta$. Then, there must exist an auxiliary equation $2^r(2\boldsymbol{u}_t - (\boldsymbol{u}_{j_k} + \boldsymbol{u}_{l_k})) = 0$ such that $\left|2\boldsymbol{y}_t - (\boldsymbol{y}_{j_k} + \boldsymbol{y}_{l_k})\right| > \frac{\delta}{\log\|\boldsymbol{A}\|_{\max}}$, otherwise $\|\boldsymbol{y}\|_\infty < \delta$. Then,

$$\boldsymbol{x}^\top \boldsymbol{B}^\top \boldsymbol{B}\boldsymbol{x} \ge \mathrm{nnz}(\boldsymbol{A}) \cdot \|\boldsymbol{A}\|_{\max} \cdot \frac{\delta}{\log\|\boldsymbol{A}\|_{\max}}.$$

Suppose $\|\boldsymbol{y}\|_\infty \le \delta$. Then,

$$\begin{aligned} \left\|\boldsymbol{p}(\boldsymbol{x}^{\mathrm{A}})\right\|_2^2 &\ge \|\boldsymbol{x}\|_2^2 - 2\,\|\boldsymbol{x}\|_2\,\|\boldsymbol{y}\|_2 \ge 1 - 2\delta, \\ \left\|\boldsymbol{p}(\boldsymbol{x}^{\mathrm{A}})\right\|_2^2 &\le \|\boldsymbol{x}\|_2^2 + \|\boldsymbol{y}\|_2^2 \le 1 + \delta^2 n_B. \end{aligned}$$

Then,

$$\begin{aligned} \boldsymbol{x}^\top \boldsymbol{B}^\top \boldsymbol{B}\boldsymbol{x} &= (\boldsymbol{p}(\boldsymbol{x}^A) + \boldsymbol{y})^\top \boldsymbol{B}^\top \boldsymbol{B}(\boldsymbol{p}(\boldsymbol{x}^A) + \boldsymbol{y}) \\ &\ge \boldsymbol{p}(\boldsymbol{x}^A)^\top \boldsymbol{B}^\top \boldsymbol{B}\boldsymbol{p}(\boldsymbol{x}^A) - 2\left\|\boldsymbol{B}\boldsymbol{p}(\boldsymbol{x}^A)\right\|_2 \|\boldsymbol{B}\boldsymbol{y}\|_2 \\ &\ge \lambda_{\min}(\boldsymbol{A}^\top \boldsymbol{A})\left\|\boldsymbol{p}(\boldsymbol{x}^A)\right\|_2^2 - 2\lambda_{\max}(\boldsymbol{B}^\top \boldsymbol{B})\left\|\boldsymbol{p}(\boldsymbol{x}^A)\right\|_2 \|\boldsymbol{y}\|_2 \\ &\ge \lambda_{\min}(\boldsymbol{A}^\top \boldsymbol{A})(1 - 2\delta) - \lambda_{\max}(\boldsymbol{B}^\top \boldsymbol{B})\delta\sqrt{n_B}\sqrt{1 + \delta^2 n_B} \\ &\ge \frac{1}{2}\lambda_{\min}(\boldsymbol{A}^\top \boldsymbol{A}) \end{aligned}$$

where the last inequality is by the choice of δ. By the Courant-Fischer theorem,

$$\lambda_{\min}(\boldsymbol{B}\boldsymbol{B}^\top) \geq \min\left\{\frac{\lambda_{\min}(\boldsymbol{A}^\top\boldsymbol{A})}{C\,\mathrm{nnz}(\boldsymbol{A})^{3/2}\log\|\boldsymbol{A}\|_{\max}},\ \frac{1}{5}\,\mathrm{nnz}(\boldsymbol{A})\,\|\boldsymbol{A}\|_{\max}\log\|\boldsymbol{A}\|_{\max}\right\}.$$

□

B Connections with Interior Point Methods

We now show that in order to solve a generalized maxflow problem in a 2-complex flow network using an Interior Point Method (IPM), it suffices to be able to apply the pseudo-inverse of $\partial_2\boldsymbol{W}\partial_2^\top$ for diagonal positive weight matrices $\boldsymbol{W}$ (and this problem is essentially equivalent to applying the pseudo-inverse of the combinatorial Laplacian of the complex, c.f. Section 1.3.1). We sketch how the these pseudo-inverse problems arise when solving a generalized maxflow using IPM, which is motivated by [Mad16]. For the more curious readers, we recommend the book [Ren01] for a complete view of general IPM algorithms.

Given a 2-complex flow network $\mathcal{K}$ with m edges and t triangles, a non-negative capacity vector $\boldsymbol{c}\in\mathbb{R}^t_{\geq 0}$, and a demand vector $\gamma\in\mathbb{R}^m$ such that $\gamma\in\mathrm{im}(\partial_2)$. The γ-maxflow problem is formulated by the following linear programming:

$$\begin{aligned}\max_{F,\boldsymbol{f}}\ & F\\ \text{s.t.}\ & \partial_2\boldsymbol{f} = F\gamma\\ & -\boldsymbol{c}\leq\boldsymbol{f}\leq\boldsymbol{c}\end{aligned}\tag{22}$$

The γ-maxflow in 2-complex flow networks is a generalization of s-t maxflow in graphs. The first constraint encodes the conservation of flows for edges in $\mathcal{K}$. And the second constraint forces the flow on triangles to satisfy the capacity constraints.

We call F the flow value of $\boldsymbol{f}$ when $\partial_2\boldsymbol{f}=F\gamma$. We assume that the optimal flow value F^* is known by IPM algorithms, which can be estimated by the binary search.

The main idea of IPM is to get rid of inequality constraints by using barrier functions, and then apply Newton's method to a sequence of equality constrained problems. The most widely used barrier function is logarithmic barrier function, which in the γ-maxflow problem gives

$$V(\boldsymbol{f}) = \sum_{\Delta\in[t]} -\log(\boldsymbol{c}(\Delta)-\boldsymbol{f}(\Delta)) - \log(\boldsymbol{c}(\Delta)+\boldsymbol{f}(\Delta)).$$

Then for a given $0\leq\alpha<1$, we define the following Barrier Problem:

$$\begin{aligned}\min_{\boldsymbol{f}}\ & V(\boldsymbol{f})\\ \text{s.t.}\ & \partial_2\boldsymbol{f} = \alpha F^*\gamma\end{aligned}\tag{23}$$

We start with zero flow, i.e., $\alpha_0=0$, and then increase $\alpha_{i+1}=\alpha_i+\alpha'$ gradually in each iteration to make progress. Given a small enough α', each iteration is composed of a *progress step* and a *centering step*.

Progress step we first take a progress step by making a Newton step to Problem (23) at the current point $\boldsymbol{f}$, while increasing the flow value by α', which gives

$$\begin{aligned}\min_{\boldsymbol{\delta}}\ & \boldsymbol{g}^\top(\boldsymbol{f})\boldsymbol{\delta} + \frac{1}{2}\boldsymbol{\delta}^\top\boldsymbol{H}(\boldsymbol{f})\boldsymbol{\delta}\\ \text{s.t.}\ & \partial_2\boldsymbol{\delta} = \alpha' F^*\gamma\end{aligned}\tag{24}$$

where $\boldsymbol{g}(\boldsymbol{f})$ and $\boldsymbol{H}(\boldsymbol{f})$ are the gradient and Hessian of V at the current point $\boldsymbol{f}$, respectively.

Problem (24) has the Lagrangian

$$\mathcal{L}(\boldsymbol{\delta}, \boldsymbol{x}) = \boldsymbol{g}^\top(\boldsymbol{f})\boldsymbol{\delta} + \frac{1}{2}\boldsymbol{\delta}^\top \boldsymbol{H}(\boldsymbol{f})\boldsymbol{\delta} + \boldsymbol{x}^\top(\alpha' F^* \gamma - \partial_2 \boldsymbol{\delta}).$$

Using optimality condition, we have

$$\nabla_{\boldsymbol{\delta}} \mathcal{L}(\boldsymbol{\delta}, \boldsymbol{x}) = \boldsymbol{g}(\boldsymbol{f}) + \boldsymbol{H}(\boldsymbol{f})\boldsymbol{\delta} - \partial_2^\top \boldsymbol{x} = \boldsymbol{0},$$

which gives,

$$\boldsymbol{\delta} = \boldsymbol{H}^{-1}(\boldsymbol{f})(\partial_2^\top \boldsymbol{x} - \boldsymbol{g}(\boldsymbol{f}))$$

Multiplying ∂_2 in both sides and using the constraint $\partial_2 \boldsymbol{\delta} = \alpha' F^* \gamma$, we obtain

$$\partial_2 \boldsymbol{H}^{-1}(\boldsymbol{f})\partial_2^\top \boldsymbol{x} = \partial_2 \boldsymbol{H}^{-1}(\boldsymbol{f})\boldsymbol{g}(\boldsymbol{f}) + \alpha' F^* \gamma.$$

Thus, we have shown that it suffices to apply the pseudo-inverse of $\partial_2 \boldsymbol{H}^{-1}(\boldsymbol{f})\partial_2^\top$ to solve $\boldsymbol{x}$ and $\boldsymbol{\delta}$:

$$\boldsymbol{x} = \left(\partial_2 \boldsymbol{H}^{-1}(\boldsymbol{f})\partial_2^\top\right)^\dagger (\partial_2 \boldsymbol{H}^{-1}(\boldsymbol{f})\boldsymbol{g}(\boldsymbol{f}) + \alpha' F^* \gamma),$$

$$\boldsymbol{\delta} = \boldsymbol{H}^{-1}(\boldsymbol{f})\partial_2^\top \left(\partial_2 \boldsymbol{H}^{-1}(\boldsymbol{f})\partial_2^\top\right)^\dagger \left(\partial_2 \boldsymbol{H}^{-1}(\boldsymbol{f})\boldsymbol{g}(\boldsymbol{f}) + \alpha' F^* \gamma\right) - \boldsymbol{H}^{-1}(\boldsymbol{f})\boldsymbol{g}(\boldsymbol{f}).$$

Centering step We then take a centering step by making a Newton step to Problem (23) at the updated point of $\tilde{\boldsymbol{f}} \stackrel{\text{def}}{=} \boldsymbol{f} + \boldsymbol{\delta}$ without increasing the flow value, which gives

$$\begin{aligned} \min_{\tilde{\boldsymbol{\delta}}} \quad & \boldsymbol{g}^\top(\tilde{\boldsymbol{f}})\tilde{\boldsymbol{\delta}} + \frac{1}{2}\tilde{\boldsymbol{\delta}}^\top \boldsymbol{H}(\tilde{\boldsymbol{f}})\tilde{\boldsymbol{\delta}} \\ \text{s.t.} \quad & \partial_2 \tilde{\boldsymbol{\delta}} = \boldsymbol{0} \end{aligned} \tag{25}$$

Similar to the progress step, it suffices to apply the pseudo-inverse of $\partial_2 \boldsymbol{H}^{-1}(\tilde{\boldsymbol{f}})\partial_2^\top$ to solve $\tilde{\boldsymbol{\delta}}$:

$$\tilde{\boldsymbol{\delta}} = \left(\boldsymbol{H}^{-1}(\tilde{\boldsymbol{f}})\partial_2^\top \left(\partial_2 \boldsymbol{H}^{-1}(\tilde{\boldsymbol{f}})\partial_2^\top\right)^\dagger \partial_2 - \boldsymbol{I}\right) \boldsymbol{H}^{-1}(\tilde{\boldsymbol{f}})\boldsymbol{g}(\tilde{\boldsymbol{f}}).$$