Chordal Decomposition for Spectral Coarsening

HONGLIN CHEN, University of Toronto, Canada
HSUEH-TI DEREK LIU, University of Toronto, Canada
ALEC JACOBSON, University of Toronto, Canada
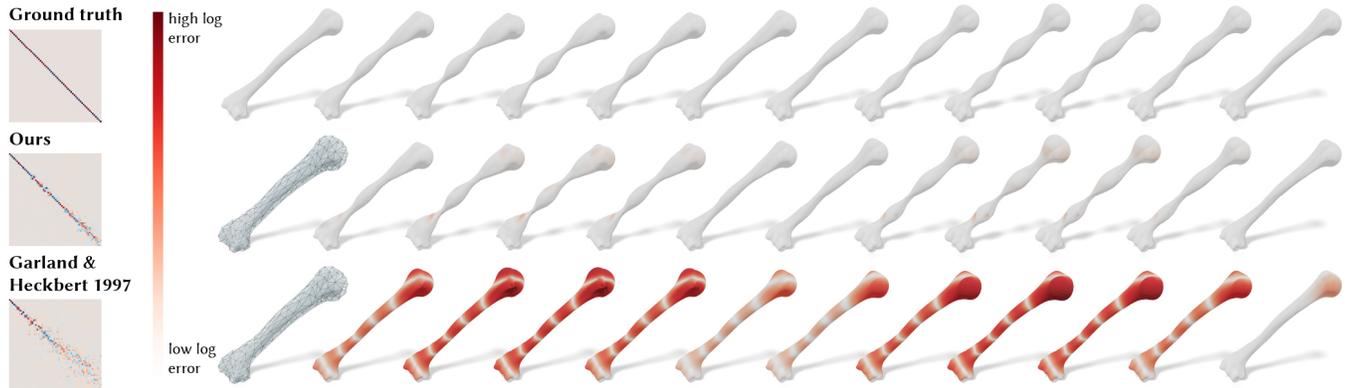DAVID I.W. LEVIN, University of Toronto, Canada

Fig. 1. We approximate the vibration modes of the cotangent Laplacian derived from the ground truth high-resolution mesh (top) using a coarse mesh with 250 vertices (the transparent cages on the left). A classical decimation method [Garland and Heckbert 1997] (bottom) preserves the appearance but fails in preserving the ground truth vibration modes. Our chordal spectral coarsening detaches the differential operator from the mesh, enabling one to optimize the operator independently to preserve the vibration modes (middle), without altering the coarse vertices. By visualizing the inner product matrices between vibration modes on the left, we show our approach leads to a result closer to the ground truth. Here we visualize the 9-th vibration mode with its frequency.

We introduce a novel solver to significantly reduce the size of a geometric operator while preserving its spectral properties at the lowest frequencies. We use chordal decomposition to formulate a convex optimization problem which allows the user to control the operator sparsity pattern. This allows for a trade-off between the spectral accuracy of the operator and the cost of its application. We efficiently minimize the energy with a change of variables and achieve state-of-the-art results on spectral coarsening. Our solver further enables novel applications including volume-to-surface approximation and detaching the operator from the mesh, *i.e.*, one can produce a mesh tailor-made for visualization and optimize an operator separately for computation.

CCS Concepts: • **Computing methodologies → Shape analysis**; • **Mathematics of computing → Semidefinite programming**.

Additional Key Words and Phrases: geometry processing, numerical coarsening, spectral geometry, chordal decomposition

Authors' addresses: Honglin Chen, University of Toronto, 40 St. George Street, Toronto, ON, Canada, M5S 2E4, chl9797@cs.toronto.edu; Hsueh-Ti Derek Liu, University of Toronto, 40 St. George Street, Toronto, ON, Canada, M5S 2E4, hsuehtil@cs.toronto.edu; Alec Jacobson, University of Toronto, 40 St. George Street, Toronto, ON, Canada, M5S 2E4, jacobson@cs.toronto.edu; David I.W. Levin, University of Toronto, 40 St. George Street, Toronto, ON, Canada, M5S 2E4, diwlevin@cs.toronto.edu.

1 INTRODUCTION

Discrete operators, such as the cotangent Laplacian, the Hessian of mesh energies, and the stiffness matrix in physics-based simulations, are ubiquitous in geometry processing. Many of these operators are represented by sparse positive semi-definite (PSD) matrices. These matrices are often constructed by looping over the elements of a discretized domain. When defined on a high-resolution domain, those matrices are computationally expensive to use, even if the final result only requires low frequency information.

Recent methods show that it is possible to simplify a discrete operator while preserving its spectral properties and matrix characteristics, such as positive semi-definiteness, avoiding the pitfalls of the naïve "decimate and reconstruct" approach. However, previous methods required the solution of a non-convex optimization problem, the solution to which sacrificed matrix sparsity.

In this paper, we overcome these challenges by applying the *chordal decomposition*. In contrast to the previous non-convex formulation, our method is now convex and can freely control the output sparsity, outperforming existing approaches for spectral coarsening and simplification. Our approach further enables novel applications on optimizing the operator independently to preserve some desired properties for computation without changing the mesh vertices.

Fig. 3. We use chordal decomposition to split a large sparse PSD constraint on X (left) into multiple small dense PSD constraints on Z_i (right), where we use $\cdot \geq 0$ to denote the PSD constraint. This enables us to be more efficient in handling optimization problems that involve sparse PSD matrix constraints.

In Fig. 1, we first decimate the model and optimize the operator independently to preserve the spectral properties of the cotangent Laplacian. Our approach achieves a higher quality approximation of the vibration modes of the high-resolution mesh compared to previous approaches.

By viewing the sparsity pattern of a matrix as a graph (see Fig. 2), one can utilize theories of chordal graphs to decompose a *sparse* matrix into a set of small *dense* matrices. This decomposition enables one to satisfy the sparse PSD constraint by projecting each small dense matrix to PSD in parallel (see Fig. 3). Such techniques have long been applied in the creation
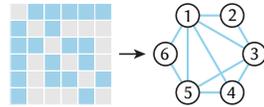

Fig. 2. Given a sparse matrix (left) where blue denotes nonzeros and gray denotes zeros, we can view the sparsity pattern as a graph (right) and then apply theorems of chordal graphs.

of efficient solvers for Semidefinite Programming (SDP). Here we generalize these notions to the spectral coarsening problem which leads to an accelerated solver that is faster, more accurate and with better sparsity control than the previous state-of-the-art. Our main contribution is an algorithm for projecting general sparse matrices to PSD ones using chordal decomposition in the context of spectral coarsening.

2 RELATED WORK

Spectral preservation is a widely studied topic in optimization and numerical methods. Below we outline the most salient related works from these areas, as well as recent developments in computer graphics and geometry processing.

2.1 Chordal Graphs in Sparse Matrix Optimization

Chordal graphs have been playing an important role in sparse matrix computation for decades [Blair and Peyton 1993; Vandenberghe and Andersen 2015]. Fukuda et al. [2001] and Nakata et al. [2003] introduce a generic framework to accelerate interior-point methods for solving large sparse SDPs. Their key idea is to exploit the sparsity of the matrix and the properties of chordal graphs [Grone et al. 1984] to decompose a large sparse matrix variable into multiple small dense ones. In the later literature, this is often called the *chordal decomposition*. Since then, this framework has been greatly improved by [Andersen et al. 2010; Burer 2003; Fujisawa et al. 2009; Srijuntongsiri and Vavasis 2004; Sun et al. 2014]. The idea of chordal decomposition has also been incorporated with other optimization methods. For instance, Sun and Vandenberghe [2015]

combine chordal decomposition with projected gradient and the Douglas–Rachford algorithms for sparse matrix nearness and completion problems. Zheng et al. [2017b, 2020] incorporate this idea to the alternating direction method of multipliers (ADMM) for solving SDPs. These chordal-based solvers have also been deployed to nonlinear matrix inequalities [Kim et al. 2011], the optimal power flow [Madani et al. 2015], controller synthesis [Zheng et al. 2018] and sum-of-squares problems [Zheng et al. 2017a, 2019].

Recently, Maron et al. [2016] formulate the point cloud registration problem into a SDP and use chordal decomposition to accelerate the computation. However, their method only supports matrices with a chordal sparsity pattern already, which is not applicable to our problem because most discrete operators are not chordal. In contrast, we utilize the ideas from [Sun and Vandenberghe 2015] to handle any sparsity pattern of choice, and the strategies in [Zheng et al. 2017b, 2020] to develop a chordal ADMM solver for the spectral coarsening energy [Liu et al. 2019]. We exploit the fact that many discrete operators are sparse and symmetric to perform a change of variables to significantly reduce the computational cost. We demonstrate that chordal decomposition is not only suitable for large scale SDPs, but also for problems in graphics that involve sparse PSD matrix variables.

2.2 Geometry Coarsening

Geometric coarsening has been extensively studied in computer graphics with the aims of preserving different geometric and physical properties. One class of methods focuses on preserving the appearance of a mesh for rendering purposes. Some prominent early examples include mesh optimization [Cohen-Steiner et al. 2004; Hoppe et al. 1993], mesh decimation [Garland and Heckbert 1997], progressive refinement [Hoppe 1996, 1997], and approaches based on parameterization [Cohen et al. 2003]. We refer readers to [Cignoni et al. 1998] for an overview and comparison of appearance-preserving simplification. Beyond preserving the appearance, these techniques have also been extended to preserve the texture information of a shape [Garland and Heckbert 1998; Lu et al. 2020]. Li et al. [2015] add modal displacement as part of the decimation metric to better preserve the acoustic transfer of a shape.

Numerical coarsening in simulation. Coarsening the geometry may alter the material properties and lead to *numerical stiffening* in simulations. Kharevych et al. [2009] propose a method to adjust the elasticity tensor of each element on a coarse mesh to approximate the dynamics of the original high-resolution mesh. In a similar spirit, Chen et al. [2015] use a data-driven lookup approach to reduce the error incurred by coarsening. To better capture vibration, Chen et al. [2017] address the numerical stiffening by simply rescaling the Young's modulus of the coarse model to match the lowest frequencies to its high-resolution counterpart. Chen et al. [2019b] extend this idea to re-fit the eigenvalues iteratively at each time step. Chen et al. [2018] propose to construct matrix-valued and discontinuous basis functions by solving a large amount of local quadratic constrained optimizations. Other recent approaches have included the wavelet approaches. Owhadi [2017] introduces a hierarchical construction of operator-adapted basis functions and their associated wavelets for scalar-valued PDE. The operator-adapted wavelets

have been extended to differential forms [Budninskiy et al. 2019] and to vector-valued equations [Chen et al. 2019a] which is then applied to fast simulation of heterogeneous materials with locally supported basis functions. Different from [Chen et al. 2018] and [Chen et al. 2019a] which increase the degrees of freedom (DOF) by using matrix-valued shape functions, our method can support more DOF by directly controlling the sparsity pattern of the scalar-valued matrix. Moreover, our method can also preserve the spectral properties using the same DOF and sparsity pattern.

Spectral graph coarsening in machine learning. Spectral-preserving graph reduction has been an active field in machine learning. Zhao et al. [2018] introduce a scalable spectral graph reduction method for scalable graph partitioning and data visualization based on node aggregation and graph sparsification. Jin et al. [2020] propose two methods for spectral graph coarsening based on iterative merging and k-means clustering, respectively. Various other approaches have also been recently adopted to coarsen a graph in a spectral-preserving way, including randomized edge contraction [Loukas and Vandergheynst 2018], local variation algorithm [Loukas 2019] and probabilistic algorithm [Bravo-Hermsdorff and Gunderson 2019]. In contrast to these combinatorial methods which focus more on optimizing the sparsity pattern, our algebraic approach enables one to further optimize over a specific sparsity pattern based on a convex formulation.

Spectral coarsening in geometry processing. Recently several approaches consider coarsening a geometry while preserving its spectral properties, namely eigenvalues and eigenvectors of the operators. Öztireli et al. [2010] compute samples on a manifold surface in order to preserve the spectrum of the Laplace operator. Nasikun et al. [2018] use a combination of Poisson disk sampling and local polynomial bases to efficiently solve an approximate Laplacian eigenvalue problem of a mesh. Beyond the Laplace operator, Liu et al. [2019] propose an algebraic approach to coarsen common geometric operators while preserving spectral properties. Lescoat et al. [2020] extend the formulation to achieve spectral-preserving mesh simplification. Our approach is purely algebraic. Our convex formulation leads us to have better spectral preservation compared to the similar algebraic approach [Liu et al. 2019] in spectral coarsening. Our flexibility in controlling the sparsity allows us to post-process the results of spectral simplification [Lescoat et al. 2020] and further improve its quality. In addition, we enable a novel application which independently optimizes the operator for computation purposes and the mesh vertices for preserving the appearance (see Fig. 1).

3 BACKGROUND

The description of our method depends on manipulating variables that represent sparse matrices. Throughout the paper, we use $\mathbf{P}$ to denote *selection matrices*, and use subscripts to differentiate between them. In practice, given a subset s, $\mathbf{P}_s$ is a sparse matrix defined as

$$(\mathbf{P}_s)_{jk} = \begin{cases} 1, & k = s(j), \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Let $\mathbf{x}$ be a vector and $\mathbf{z} = \mathbf{x}(s)$ be a sub-vector of $\mathbf{x}$. Selecting a subset from $\mathbf{x}$ can be achieved by a sparse matrix multiplication $\mathbf{z} = \mathbf{P}_s \mathbf{x}$.

Fig. 4. Chordal decomposition decomposes the matrix $\mathbf{X}$ into a set of maximal clique matrices $\mathbf{Z}_i$. We can extract each clique matrix via $\mathbf{Z}_i = \mathbf{P}_i \mathbf{X} \mathbf{P}_i^\top$.

Mapping the elements from $\mathbf{z}$ to a bigger vector $\mathbf{y}$ can be achieved with $\mathbf{y} = \mathbf{P}_s^\top \mathbf{z}$

$$\underbrace{\begin{bmatrix} c_1 \\ c_2 \\ c_4 \end{bmatrix}}_{\mathbf{z}} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{P}_s} \underbrace{\begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix}}_{\mathbf{x}}, \quad \underbrace{\begin{bmatrix} c_1 \\ c_2 \\ 0 \\ c_4 \end{bmatrix}}_{\mathbf{y}} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{P}_s^\top} \underbrace{\begin{bmatrix} c_1 \\ c_2 \\ c_4 \end{bmatrix}}_{\mathbf{z}}.$$

Let $\mathbf{X}$ be an n-by-n matrix, we use s to denote a subset of row and column indices into $\mathbf{X}$. $\mathbf{Z} = \mathbf{X}(s, s)$ creates a matrix $\mathbf{Z}$ that is of size $|s|$-by-$|s|$ and contains all values in $\mathbf{X}(s, s)$. We can compactly describe this operation using selection matrix $\mathbf{P}_s$ as $\mathbf{Z} = \mathbf{P}_s \mathbf{X} \mathbf{P}_s^\top$

$$\underbrace{\begin{bmatrix} c_{11} & c_{13} \\ c_{31} & c_{33} \end{bmatrix}}_{\mathbf{Z}} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{P}_s} \underbrace{\begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix}}_{\mathbf{X}} \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}}_{\mathbf{P}_s^\top}.$$

Similarly, we can map elements in $\mathbf{Z}$ back to $\mathbf{Y}$ via $\mathbf{Y} = \mathbf{P}_s^\top \mathbf{Z} \mathbf{P}_s$

$$\underbrace{\begin{bmatrix} c_{11} & 0 & c_{13} \\ 0 & 0 & 0 \\ c_{31} & 0 & c_{33} \end{bmatrix}}_{\mathbf{Y}} = \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}}_{\mathbf{P}_s^\top} \underbrace{\begin{bmatrix} c_{11} & c_{13} \\ c_{31} & c_{33} \end{bmatrix}}_{\mathbf{Z}} \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{P}_s}.$$

3.1 Chordal Decomposition

A *chordal graph* is an undirected graph in which for every cycle of length greater than three, there is an edge between nonconsecutive vertices in the cycle. Chordal graphs have drawn attention since the 1950s because a handful of NP-complete graph problems can be solved in polynomial time if the graph is chordal. Chordal graphs also received interests from the optimization community for solving sparse SDPs, combinatorial optimization, and Cholesky factorization. We refer readers to [Vandenberghe and Andersen 2015] for a survey of chordal graphs in optimization. We focus on its application to problems that involve sparse PSD matrices constraints, specifically arising from geometry processing.

An n-by-n symmetric matrix $\mathbf{X}$ has *chordal sparsity* pattern $C \in \{0, 1\}^{n \times n}$ if the graph induced by C is a chordal graph. The key theorem that supports our method is

THEOREM 1. *([Agler et al. 1988; Kakimura 2010]) Let $\mathbf{X}$ be a n-by-n symmetric matrix with chordal sparsity, and let $\{\mathbf{Z}_1, \mathbf{Z}_2, \cdots, \mathbf{Z}_p\}$ be a set of its p clique matrices. Then $\mathbf{X}$ is PSD if and only if it can be expressed as*

$$\mathbf{X} = \sum_{i=1}^{p} \mathbf{P}_i^\top \mathbf{Z}_i \mathbf{P}_i \tag{2}$$

with all $\mathbf{Z}_i$ being PSD.

where a *clique* is a subset of vertices such that every two distinct vertices in the clique are adjacent to each other, thus a *clique matrix* is a dense matrix of the size of a clique. We use $\mathbf{Z}_i$ to represent the ith clique matrix, $\mathbf{P}_i$ as the selection matrix to the ith clique set. This decomposition from $\mathbf{X}$ to a set of clique matrices is called the *chordal decomposition* (see Fig. 4), which has been applied to many recent SDP solvers.

Vectorization. In practice, the "sandwich" format $\mathbf{P}_i^\top \mathbf{Z}_i \mathbf{P}_i$ is not always easy to work with. It is often more desirable to *vectorize* a matrix by concatenating the columns of the matrix into a vector (see the inset). We can re-write the vectorized chordal decomposition as

$$\text{vec}(\mathbf{X}) = \sum_{i=1}^{p} \text{vec}(\mathbf{P}_i^\top \mathbf{Z}_i \mathbf{P}_i) = \sum_{i=1}^{p} \underbrace{(\mathbf{P}_i^\top \otimes \mathbf{P}_i^\top)}_{\mathbf{K}_i} \text{vec}(\mathbf{Z}_i), \qquad (3)$$

where we use $\text{vec}(\cdot)$ to denote the vectorization, with its inverse vec^{-1} (see the inset), and $\otimes$ to denote the Kronecker product. Intuitively, $\mathbf{K}_i$ acts like the transpose of a selection matrix, putting elements in $\text{vec}(\mathbf{Z}_i)$ back to $\text{vec}(\mathbf{X})$.
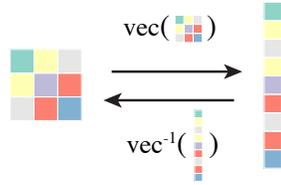
3.2 Chordal Extension

In practice, a majority of matrices we encounter in geometry processing do not naturally have chordal sparsity patterns, which makes Theorem 1 inapplicable. In response, we follow the idea in [Sun and Vandenberghe 2015] to first perform a *chordal extension* to transform the original non-chordal sparsity $\mathcal{E}$ to a chordal sparsity pattern C (see the inset). We maintain $\mathcal{E}$ by adding equality constraints to enforce new fill-in elements arising from the extension to be zeros

$$\mathbf{X} \in \mathbb{S}_\mathcal{E}^n \Rightarrow \begin{array}{l} \mathbf{X} \in \mathbb{S}_C^n, \\ \mathbf{X}_{jk} = 0, \quad \forall (j,k) \in C \backslash \mathcal{E}, \end{array} \qquad (4)$$

where $\mathbb{S}_\mathcal{E}^n$ and $\mathbb{S}_C^n$ denote n-by-n symmetric matrices with sparsity patterns $\mathcal{E}$ and C, respectively. We use $C \backslash \mathcal{E}$ to denote the entries that exist in C, but not in $\mathcal{E}$. Chordal extension adds degrees of freedom to our optimization problem. Our zero constraints enforce that, at a particular new fill-in entry, the sum of projected dense matrices must equal zero, not that each dense matrix must contribute a zero value to that entry. Computing the *minimum* chordal extension, where the number of fill-in edges is minimized, is NP-complete [Yannakakis 1981]. However, finding a *minimal* chordal extension can be solved in polynomial time [Heggernes 2006].
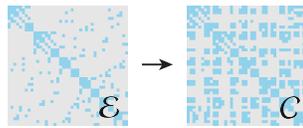
Notice that Theorem 1 also has a dual format $\mathbf{Y}_i = \mathbf{P}_i \mathbf{X} \mathbf{P}_i^\top$. If $\mathbf{X}$ has the chordal sparsity, this dual formulation can guarantee $\mathbf{X}$ to be PSD by ensuring all $\mathbf{Y}_i$ being PSD, proved by the Theorem 7 in [Grone et al. 1984]. However, this dual formulation cannot guarantee $\mathbf{X}$ to be PSD if the matrix $\mathbf{X}$ does not have chordal sparsity (see Sec. 3.2.2 in [Sun 2015]). Thus we build our algorithm surrounding Theorem 1.
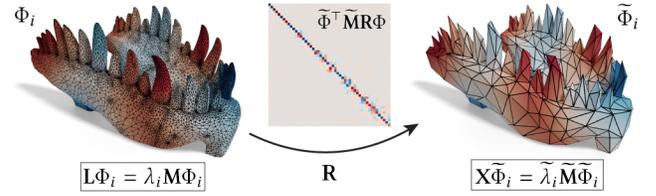
Fig. 5. We visualize the spectral preservation using the inner product matrix (middle) between the restricted eigenvectors $\mathbf{R}\Phi$ of the original operator $\mathbf{L}$ to the coarsened domain and the eigenvectors $\widetilde{\Phi}$ of the coarsened operator $\mathbf{X}$. Due to the orthonormality, the ground truth should be a diagonal matrix of 1 and -1 (denoted by red and blue, respectively). The closer the matrix to a diagonal matrix, the better the preservation of eigenvectors. We use $\mathbf{M}$ and $\widetilde{\mathbf{M}}$ to denote the mass matrices of the original and the coarsened meshes respectively.

4 METHOD

The goal of spectral coarsening is to reduce the size of a discrete operator, derived from a 3D shape, while preserving its spectral properties. Liu et al. [2019] show that it is possible to have a significant reduction without affecting the low-frequency eigenvectors and eigenvalues. They visualize the preservation of spectral properties with the inner product matrix between eigenvectors (see Fig. 5). This inner product matrix can be perceived as a *functional map* [Ovsjanikov et al. 2012], expressing how eigenfunctions on the original domain are mapped to the simplified domain (see Sec. 5.1).

Preserving the spectral properties of an operator can be cast as an optimization problem, minimizing the commutative energy [Liu et al. 2019]

$$f(\mathbf{X}) = \|\mathbf{R}\mathbf{M}^{-1}\mathbf{L}\Phi - \widetilde{\mathbf{M}}^{-1}\mathbf{X}\mathbf{R}\Phi\|_{\widetilde{\mathbf{M}}}^2, \qquad (5)$$

where $\mathbf{L}$ and $\mathbf{X}$ denote the original and the coarsened operators, $\mathbf{M}$ and $\widetilde{\mathbf{M}}$ are the original and the coarsened mass matrices, $\mathbf{R}$ is the restriction operator restricting functions from the original domain to the coarsened domain, and Φ are the functions (*e.g.*, eigenfunctions) used to measure the commutativity.

Intuitively, if the coarsened operator $\mathbf{X}$ preserves the spectral properties of the original operator $\mathbf{L}$, then given some functions Φ on the original domain, first applying the original operator $\mathbf{M}^{-1}\mathbf{L}$ and then restricting the functions to the coarsened domain via $\mathbf{R}$ should be the same as first restricting the functions via $\mathbf{R}$ and then applying the coarsened operator $\widetilde{\mathbf{M}}^{-1}\mathbf{X}$. In the Appendix C of [Liu et al. 2019] they show that, when Φ are eigenfunctions, minimizing the commutative energy also preserves eigenvalues.

Relationship to [Liu et al. 2019]. Many differential operators in geometry processing are sparse, symmetric, and positive semidefinite. Thus, the method of [Liu et al. 2019] adds constraints to Eq. 5 in order to preserve the three operator properties. They satisfy the constraints via change of variables from $\mathbf{X}$ to $\mathbf{G}$

$$\underset{\mathbf{X}}{\text{minimize}} \ f(\mathbf{X}) \Rightarrow \underset{\mathbf{G}}{\text{minimize}} \ f(\mathbf{G}^\top \mathbf{L}\mathbf{G}). \qquad (6)$$

where $\mathbf{G}$ has a predetermined sparsity pattern. However, this transforms the original convex formulation into a non-convex *quartic* one (see Eq.7 in [Liu et al. 2019]) and increases the sparsity of the

output operator to 3-rings. It also artificially limits the feasible region to a subset of PSD matrices determined by $\mathbf{G}$. In contrast, we will show how to directly optimize the commutative energy with respect to $\mathbf{X}$ while maintaining the convexity and enabling one to control over the output sparsity.

4.1 Chordal Spectral Coarsening

Spectral coarsening can be written as the following optimization

$$\underset{\mathbf{X}}{\text{minimize}} \quad f(\mathbf{X}) \tag{7}$$

$$\text{subject to} \quad \mathbf{X}\mathbf{v} = \mathbf{e} \tag{8}$$

$$\mathbf{X} \succeq 0 \tag{9}$$

$$\mathbf{X} \in \mathbb{S}_{\mathcal{E}}^n, \tag{10}$$

where f is the spectral coarsening energy in Eq. 5, $\mathbf{X} \succeq 0$ denotes the PSD constraint, and $\mathbb{S}_{\mathcal{E}}^n$ denotes the set of n-by-n sparse symmetric matrices with a user-defined (non-chordal) sparsity pattern $\mathcal{E}$. The equality $\mathbf{X}\mathbf{v} = \mathbf{e}$ represents the null-space constraint of a differential operator, in the case of Laplacian $\mathbf{v} = \mathbf{1}$ is a constant function and $\mathbf{e} = \mathbf{0}$ is a zero vector because every row or column of a Laplacian sums to zero. For the sake of simplicity, we describe the entire process without expanding the spectral coarsening energy f, and the complete formulation is detailed in App. E.

Applying the chordal extension (Sec. 3.2) and the chordal decomposition (Sec. 3.1) to Eq. 7 leads to

$$\underset{\mathbf{X}, \{\mathbf{Z}_i\}}{\text{minimize}} \quad f(\mathbf{X}) \tag{11}$$

$$\text{subject to} \quad \mathbf{X}\mathbf{v} = \mathbf{e} \tag{12}$$

$$\mathbf{X} \in \mathbb{S}_C^n \tag{13}$$

$$\mathbf{X}_{jk} = 0, \qquad \forall (j,k) \in C \backslash \mathcal{E} \tag{14}$$

$$\mathbf{X} = \sum_{i=1}^{p} \mathbf{P}_i^\top \mathbf{Z}_i \mathbf{P}_i \tag{15}$$

$$\mathbf{Z}_i \succeq 0, \qquad k = 1, \cdots, p, \tag{16}$$

where p is the number of maximal cliques. We convert the PSD constraint $\mathbf{X} \succeq 0$ in Eq. 9 to many small PSD constraints $\mathbf{Z}_i \succeq 0$ according to Theorem 1. Here we also perform chordal extension to switch the sparsity from non-chordal $\mathcal{E}$ to a chordal C with additional equality constraints $\mathbf{X}_{jk} = 0$ (see Eq. 4).

Ensuring the PSD property of the matrix requires a full (generalized) eigendecomposition followed by the removal of the negative eigenvalues. When the matrix is large, a full decomposition is intractable to compute. Using chordal decomposition to transform the big PSD constraint (Eq. 9) to a set of small ones (Eq. 16) allows us to efficiently project each $\mathbf{Z}_i$ to PSD in parallel.

4.2 Change of Variables

Utilizing the fact that $\mathbf{X}$ is symmetric with a sparsity pattern $\mathcal{E}$, we propose to accelerate the solver via change of variables from $\mathbf{X}$ to a compressed vector $\mathbf{x}_{\mathcal{E}}$ which consists of the non-zero elements of the lower triangular part defined by $\mathcal{E}$. This change of variables restricts the optimization to search only within the feasible sparsity $\mathcal{E}$. This is crucial to the performance of the solver because $\mathbf{X}$ is sparse thus $|\mathbf{x}_{\mathcal{E}}| \ll |\text{vec}(\mathbf{X})|$ significantly reduces the degrees of freedom. The relationship between $\mathbf{X}$ and $\mathbf{x}_{\mathcal{E}}$ is described by

$$\text{vec}(\mathbf{X}) = \mathbf{P}_{\mathcal{E}}^{-1} \mathbf{x}_{\mathcal{E}}, \qquad \mathbf{x}_{\mathcal{E}} = \mathbf{P}_{\mathcal{E}} \text{vec}(\mathbf{X}), \tag{17}$$

where $\mathbf{P}_{\mathcal{E}}$ is a selection matrix to the sub-vector $\mathbf{x}_{\mathcal{E}}$. $\mathbf{P}_{\mathcal{E}}^{-1}$ is the inverse of $\mathbf{P}_{\mathcal{E}}$ which is another matrix to re-index elements in $\mathbf{x}_{\mathcal{E}}$ back to $\text{vec}(\mathbf{X})$. Note that $\mathbf{P}_{\mathcal{E}}^{-1}$ is different from the $\mathbf{P}_{\mathcal{E}}^\top$ as each non-diagonal element in $\mathbf{x}_{\mathcal{E}}$ gets mapped to two entries in $\text{vec}(\mathbf{X})$, instead of one entry, and $\mathbf{P}_{\mathcal{E}}^{-1}$ can be assembled easily without the need of explicitly inverting the matrix. This change of variables incorporates both the chordal symmetric constraint $\mathbf{X} \in \mathbb{S}_C^n$ and the equality constraints $\mathbf{X}_{jk} = 0$ in Eq. 11. After some derivation in App. B, we have

$$\underset{\mathbf{x}_{\mathcal{E}}, \{\mathbf{z}_i\}}{\text{minimize}} \quad f(\mathbf{x}_{\mathcal{E}}) \tag{18}$$

$$\text{subject to} \quad \mathbf{G}\mathbf{x}_{\mathcal{E}} = \mathbf{e} \tag{19}$$

$$\mathbf{P}_{\mathcal{E}}^{-1} \mathbf{x}_{\mathcal{E}} = \sum_{i=1}^{p} \mathbf{K}_i \mathbf{z}_i \tag{20}$$

$$\text{vec}^{-1}(\mathbf{z}_i) \succeq 0, \quad i = 1, \cdots, p, \tag{21}$$

We define $\mathbf{z}_i := \text{vec}(\mathbf{Z}_i)$ to be the vectorized clique matrix. $\mathbf{G}\mathbf{x}_{\mathcal{E}} = \mathbf{e}$ is the vectorized version of the $\mathbf{X}\mathbf{v} = \mathbf{e}$ in Eq. 12. $\mathbf{P}_{\mathcal{E}}^{-1}\mathbf{x}_{\mathcal{E}} = \sum_{i=1}^{p} \mathbf{K}_i \mathbf{z}_i$ is the vectorized chordal decomposition Eq. 15. Here $\mathbf{K}_i$ denotes the index selection matrix for vectorized clique matrix $\mathbf{z}_i$.

We use another change of variables to further accelerate the algorithm by restricting the vectorized chordal decomposition in Eq. 20 to only the non-zeros in the chordal sparsity pattern C. That is because the summation of $\{\mathbf{z}_i\}$ in Eq. 20 only has non-zeros in the chordal sparsity pattern C. We introduce another index selection matrix $\mathbf{P}_C$ to change Eq. 20 into

$$\mathbf{P}_{\mathcal{E}}^{-1}\mathbf{x}_{\mathcal{E}} = \sum_{i=1}^{p} \mathbf{K}_i \mathbf{z}_i \implies \mathbf{P}_C \mathbf{P}_{\mathcal{E}}^{-1}\mathbf{x}_{\mathcal{E}} = \mathbf{P}_C \sum_{i=1}^{p} \mathbf{K}_i \mathbf{z}_i, \tag{22}$$

where $\mathbf{P}_C$ selects the lower triangular non-zeros in C from the original $\text{vec}(\mathbf{X})$. Here $\mathbf{P}_C$ is defined the same as the $\mathbf{P}_{\mathcal{E}}$ in Eq. 17 but with a different sparsity pattern C.

As $\mathbf{z}_i$ is the vectorization of a *symmetric* matrix $\mathbf{Z}_i$, another reduction is achieved by applying the same trick as Eq. 17 to restrict the degrees of freedom of $\mathbf{Z}_i$ to its lower triangular part $\widetilde{\mathbf{Z}}_i$ via an expansion matrix $\mathbf{Q}_i$ (see the inset).

$$\underbrace{\text{vec}(\mathbf{Z}_i)}_{\mathbf{z}_i} = \mathbf{Q}_i \underbrace{\text{vec}(\widetilde{\mathbf{Z}}_i)}_{\tilde{\mathbf{z}}_i}, \tag{23}$$

We use $\mathbf{z}_i, \tilde{\mathbf{z}}_i$ to denote the vectorized $\mathbf{Z}_i$ and the vectorized lower triangular part $\widetilde{\mathbf{Z}}_i$, respectively. We define $\mathbf{Q}_i$ as an inverse index selection matrix that expands the vector of the lower triangular element $\tilde{\mathbf{z}}_i$ to $\mathbf{z}_i$.

Combining the above results leads us to the reduced optimization problem

$$\underset{\mathbf{x}_{\mathcal{E}}, \tilde{\mathbf{z}}}{\text{minimize}} \quad f(\mathbf{x}_{\mathcal{E}}) \tag{24}$$

$$\text{subject to} \quad \mathbf{G}\mathbf{x}_{\mathcal{E}} = \mathbf{e} \tag{25}$$

$$\mathbf{P}_C \mathbf{P}_{\mathcal{E}}^{-1} \mathbf{x}_{\mathcal{E}} = \mathbf{P}_C \widetilde{\mathbf{K}} \tilde{\mathbf{z}} \tag{26}$$

$$\text{vec}^{-1}(\mathbf{Q}_i \tilde{\mathbf{z}}_i) \succeq 0, \quad i = 1, \cdots, p, \tag{27}$$

where

$$\widetilde{\mathbf{K}} = \begin{bmatrix} \mathbf{K}_1 \mathbf{Q}_1, & \cdots, & \mathbf{K}_p \mathbf{Q}_p \end{bmatrix}, \quad \tilde{\mathbf{z}} = \begin{bmatrix} \tilde{\mathbf{z}}_1 \\ \vdots \\ \tilde{\mathbf{z}}_p \end{bmatrix}. \tag{28}$$

This final reduced formulation is an optimization problem which involves only linear equalities and small dense PSD constraints. We solve this optimization using ADMM (see App. A), alternating between solving for $\mathbf{x}_{\mathcal{E}}$ and $\tilde{\mathbf{z}}$. Solving for $\mathbf{x}_{\mathcal{E}}$ when f is the spectral coarsening energy boils down to a single linear solve; solving for $\tilde{\mathbf{z}}$ leads to a subroutine of projecting each clique matrix to PSD by removing the negative eigenvalues. The update on $\tilde{\mathbf{z}}$ is efficient as each $\tilde{\mathbf{z}}_i$ is small and can be trivially parallelized. We provide details of the ADMM derivation in App. C.

4.3 Weighted Spectral Coarsening

Solving Eq. 24 results in a coarsened operator that preserves the spectral properties of the original one. One can freely control the sparsity pattern of the output by changing $\mathcal{E}$. In our experiments, we choose either 1-, 2-, or 3-ring sparsities. The more rings in use, the better the results because we have more degrees of freedom in minimizing the spectral coarsening energy Eq. 5.

When the degrees of freedom are limited, such as using only 1-ring, we notice that the solver would emphasize preserving relatively higher frequencies and lead to worse performance in preserving the lowest frequencies. In response, we weight the spectral coarsening energy Eq. 5 via the inverse of eigenvalues, which leads to this weighted version

$$f_w(\mathbf{X}) = \|\mathbf{R}\mathbf{M}^{-1}\mathbf{L}\Phi\Lambda^{-1} - \widetilde{\mathbf{M}}^{-1}\mathbf{X}\mathbf{R}\Phi\Lambda^{-1}\|_{\widetilde{\mathbf{M}}}^2, \tag{29}$$

where Λ is a diagonal matrix of the eigenvalues of the original operators $\mathbf{L}$. In Sec. 5, we show that the weighted version leads to a better spectral preservation in the low frequencies when using our solver. This weighted formulation also naturally captures the notion of "null-space reproduction" in Eq. 25, as we explicitly enforce the null-space corresponding to the eigenvalue 0 as a hard constraint, *i.e.*, with infinite weight.

5 RESULTS

We evaluate our solver by comparing against the existing state-of-the-art spectral coarsening [Liu et al. 2019] and simplification [Lescoat et al. 2020], using functional maps and the quantitative metrics $\|\cdot\|_L$ and $\|\cdot\|_D$ proposed in [Lescoat et al. 2020]. We further demonstrate the power of our solver in controlling the sparsity patterns, approximating volumetric behavior using only boundary surface vertices and detaching the differential operator from the mesh. We provide implementation details in App. F.
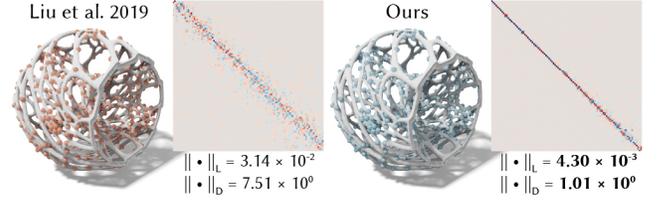
Fig. 6. Using the same 3-ring sparsity pattern, our convex formulation enables the ADMM solver to converge to a better result on shape (from 80,000 vertices to 600) where the gradient descent in [Liu et al. 2019] may struggle to converge.

5.1 Evaluation Metrics

Functional maps [Ovsjanikov et al. 2012] describe how to transport functions from one shape $\mathcal{M}$ to another shape $\mathcal{N}$. The idea of functional map has led to breakthroughs in computing shape correspondences [Ovsjanikov et al. 2017]. In the context of spectral coarsening, functional maps become a tool for evaluating how the eigenvectors of a discrete operator $\mathbf{L} \in \mathbb{R}^{n \times n}$ derived on a high-resolution mesh are maintained by a coarsened operator $\mathbf{X} \in \mathbb{R}^{m \times m}$. Following the notation in Fig. 5, let $\Phi \in \mathbb{R}^{n \times k}$ and $\widetilde{\Phi} \in \mathbb{R}^{m \times k}$ be two set of eigenvectors of $\mathbf{L}$ and $\mathbf{X}$, respectively, the functional map $\mathbf{C}$ can be computed as

$$\mathbf{C} = \widetilde{\Phi}^\top \widetilde{\mathbf{M}} \mathbf{R} \Phi. \tag{30}$$

Here $\widetilde{\mathbf{M}}$ is the mass matrix in the coarse domain and $\mathbf{R}$ is a restriction operator, encoding the correspondences information from the original mesh to its coarsened counterpart. The restriction operator is computed either during the decimation [Lescoat et al. 2020] or simply a subset selection matrix as in [Liu et al. 2019]. One can also perceive the matrix $\mathbf{C}$ as an inner product matrix between the eigenvectors $\widetilde{\Phi}$ on the coarsened domain and the restricted eigenvectors $\mathbf{R}\Phi$ to the coarsened domain. Due to the orthonormality between eigenvectors, the optimal functional map (or inner product matrix) $\mathbf{C}$ should be a diagonal matrix of values 1 and -1 (see inset).

Laplacian commutativity and Orthonormality norm. The functional map should be orthonormal and commute with the original Laplace operator in the reduced basis if and only if it preserves corresponding eigenfunctions and eigenvalues exactly, as shown in [Lescoat et al. 2020]. Thus the spectral preservation before and after coarsening and simplification can be quantified using two norms:

$$\text{Laplacian commutativity: } \|\cdot\|_L^2 = \frac{\|\mathbf{C}\Lambda - \widetilde{\Lambda}\mathbf{C}\|^2}{\|\mathbf{C}\|^2} \tag{31}$$

$$\text{Orthonormality: } \|\cdot\|_D^2 = \|\mathbf{C}^\top\mathbf{C} - \mathbf{I}\|^2. \tag{32}$$

In our experiments, we visualize the functional map $\mathbf{C}$ and report both norms to convey a complete picture of spectral preservation.

5.2 Spectral Coarsening

Comparing to the original non-convex formulation Eq. 6 [Liu et al. 2019], in Fig. 10 we show that our convex formulation consistently achieves lower objective values across different number of coarsened vertices (from 200 to 1200) and leads to better qualitative results (see
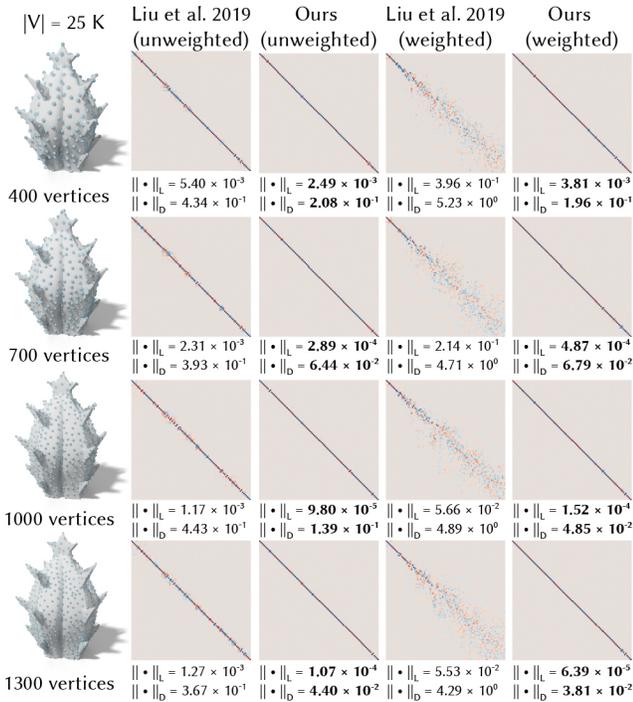
Fig. 7. Using the same 3-ring sparsity as [Liu et al. 2019], our method achieves better quality of resulting functional maps for both the weighted and unweighted versions, measured by the metrics proposed in [Lescoat et al. 2020].

Fig. 7, Fig. 6 and Fig. 9). For a fair comparison, we set the sparsity pattern of our approach to be 3 rings, the same as the method of [Liu et al. 2019]. We can further show that through weighting the energy with the inverse of the eigenvalues (see Eq. 29), we obtain an even better preservation of the low-frequencies, see Fig. 7 (right two) and Fig. 8. In general, the weighted version performs better in maintaining the lowest frequencies, while the unweighted version tends to preserve all the eigenmodes in a least-square sense.

With the reusable numerical factorization and separable PSD projection structures, our ADMM solver is able to solve the problem efficiently while the method of [Liu et al. 2019] takes longer to converge. In Fig. 11, we compare the runtime with [Liu et al. 2019], both using the optimal setups (our weighted version and [Liu et al. 2019] unweighted version). The $\arg\min_X$ step requires a linear solve of a KKT system and $\arg\min_Z$ are a set of PSD projections of the small clique matrices. For details about $\arg\min_X$ and $\arg\min_Z$ step, see App. A. Leveraging the fact that the KKT system matrix in $\arg\min_X$ remains the same until ρ is updated, we only perform numerical factorization when ρ is updated and reuse it until ρ changes again (usually after tens of iterations). As shown in Fig. 12, most of our runtime is spent on numerical factorization while the time spent on each $\arg\min_X$ and $\arg\min_Z$ step is relatively small. We report our detailed runtime in Fig. 23. For detailed runtime comparison within the weighted and unweighted versions, see Fig. 28.

We also compare the total runtime of our sparse ADMM solver with the MOSEK solver in CVX [Grant and Boyd 2008, 2014] in
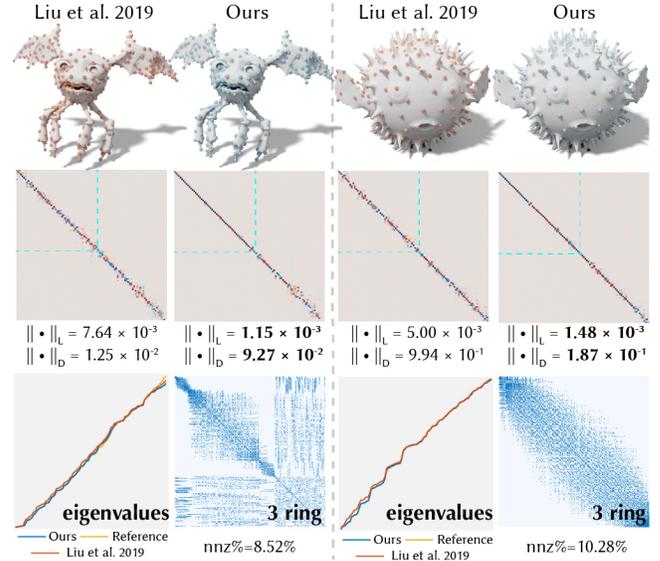

Fig. 8. For applications that desire to preserve low frequencies, our weighted formulation can focus on preserving the first few eigenvectors and eigenvalues (shown in increasing order). Our weighted formulation achieves better results comparing to [Liu et al. 2019] under the same sparsity pattern when coarsening the shapes from 8,000 (left) and 28,000 (right) vertices to 400, respectively. Here we show the Laplacian commutativity norm and Orthonormality norm based on the functional map of the first 50 eigenvectors (inside the dashed lines).
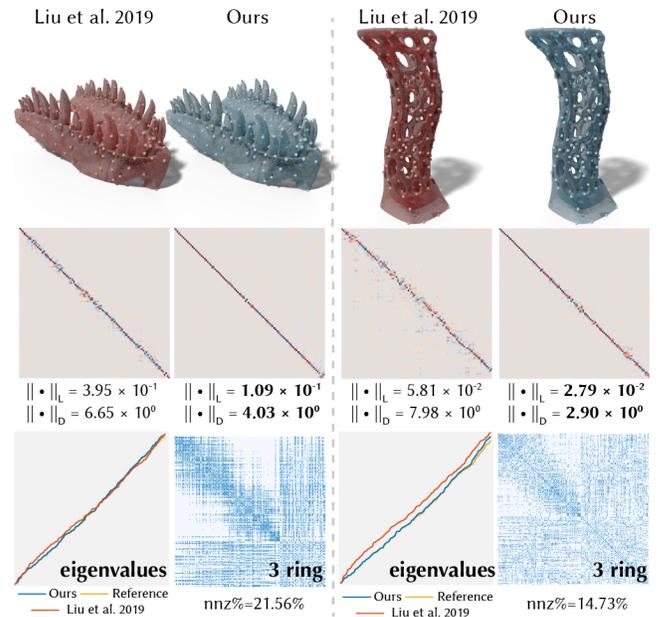

Fig. 9. As degrees of freedom increase for volumetric Laplacian, our method is still able to maintain the spectral properties of the tetrahedral meshes (from 32,000 and 27,000 vertices to 400 respectively) using the same sparsity as [Liu et al. 2019]. Here the eigenvalues are shown in increasing order.
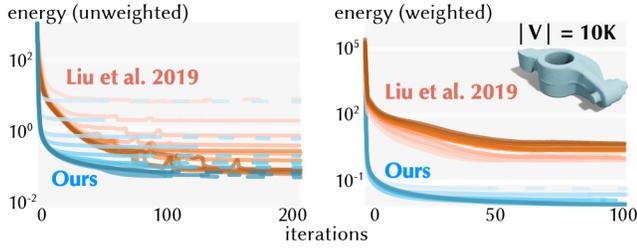
Fig. 10. Under the same 3-ring sparsity, our method consistently achieves better objective values comparing to the spectral coarsening method proposed by [Liu et al. 2019]. We evaluate both unweighted (top) and the weighted (bottom) versions across different numbers of coarse vertices ranging from 200 (light) to 1200 (dark). Note that the dashed lines denote that the optimization has already converged.
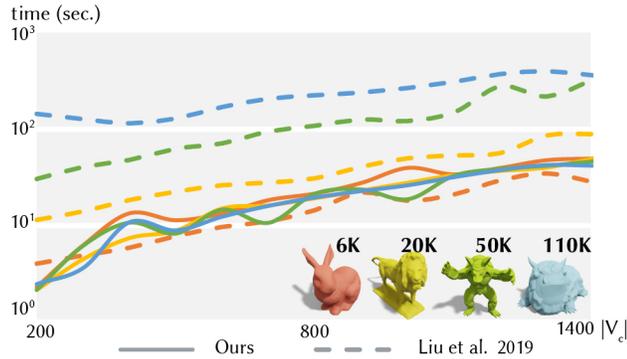

Fig. 11. We compare the runtime of our algorithm (weighted) with [Liu et al. 2019] (unweighted) using the same 3-ring sparsity pattern with respect to the number of coarsened vertices $|V_c|$, as our method performs better with the weighted version and [Liu et al. 2019] shows the opposite. Here we only consider the solve time, factoring out the precomputation for both our method and the method of [Liu et al. 2019]. As in our formulation the solve involved in ADMM is independent of the resolution of the original mesh, we are able to coarsen a high-resolution mesh without a significantly increased solve time compared to [Liu et al. 2019].
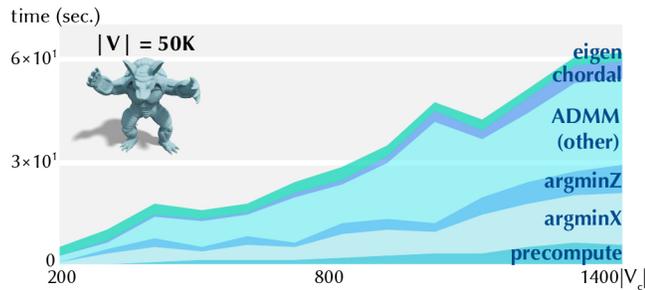

Fig. 12. We show the decomposition of the total runtime of our algorithm using the same 3 rings sparsity pattern as in Fig. 11. From bottom to top are the precomputation time, $\arg\min_X$ time, $\arg\min_Z$ time, other ADMM time (including numerical factorization), chordal decomposition time and eigendecomposition time, respectively. As shown in the figure, most of the runtime of our algorithm is spent on numerical factorization. By reusing numerical factorization until ρ changes, the time spent on each $\arg\min_X$ and $\arg\min_Z$ step is relatively small.
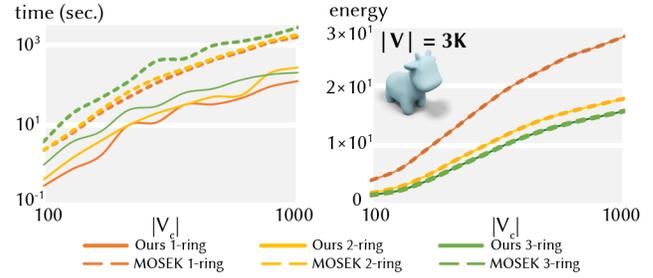
Fig. 13. We compare the total runtime of our solver and the MOSEK solver in CVX [Grant and Boyd 2008, 2014], which only supports dense SDP constraints and uses interior point method to solve the dense SDP problem using the 1-, 2- and 3-ring sparsity patterns of [Garland and Heckbert 1997]. As a dense SDP solver that is not designed to solve large sparse SDP problem, MOSEK takes a relatively long runtime when the matrix size is large or the rings of neighborhood increases. Here $|V_c|$ is number of the vertices in the coarse mesh.

Fig. 14. We visualize the biharmonic distance of our method and [Garland and Heckbert 1997] using the same 1-ring sparsity pattern. Our method can further postprocess and improve spectral preservation of the result from [Garland and Heckbert 1997] (from 110,000 vertices to 500).

Fig. 15. By visualizing the biharmonic distance, we show that our approach can also postprocess the result from [Lescoat et al. 2020] (from 10,000 vertices to 800) and achieve better spectral preservation while still maintaining the same 1-ring sparsity pattern.

Fig. 13, which uses the interior point method to solve the problem with dense PSD constraints. We show our solver can work on large problems in a more efficient way than MOSEK, while MOSEK, which only supports dense semi-definiteness constraints, is not designed for large sparse SDP problem and takes a relatively long time to converge when the matrix size is large or the rings of neighborhood increases. Here we use $0.8 \times |V_c|$ eigenvectors to ensure both methods converge.

5.3 Spectral Simplification

Our approach could further improve the results from the spectral simplification via post-processing. The method of [Lescoat et al. 2020] performs spectral simplification by greedily collapsing the edge with the minimum cost, thus it may result in suboptimal results. In Fig. 16 and Fig. 27 we post-process the cotangent Laplacian

$\|\bullet\|_L = 3.38 \times 10^{-1}$ $\|\bullet\|_L = \mathbf{1.38 \times 10^{-1}}$
$\|\bullet\|_D = 5.44 \times 10^{0}$ $\|\bullet\|_D = \mathbf{3.45 \times 10^{0}}$

300 vertices

$\|\bullet\|_L = 3.84 \times 10^{-2}$ $\|\bullet\|_L = \mathbf{1.32 \times 10^{-2}}$
$\|\bullet\|_D = 2.23 \times 10^{0}$ $\|\bullet\|_D = \mathbf{1.49 \times 10^{0}}$

600 vertices

$\|\bullet\|_L = 3.12 \times 10^{-3}$ $\|\bullet\|_L = \mathbf{7.99 \times 10^{-4}}$
$\|\bullet\|_D = 7.64 \times 10^{-1}$ $\|\bullet\|_D = \mathbf{3.08 \times 10^{-1}}$
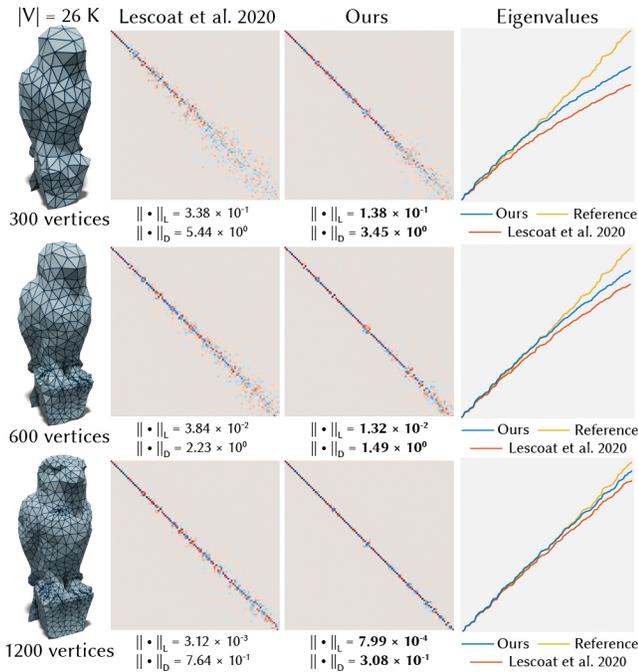
1200 vertices

Fig. 16. Using the same 1-ring sparsity pattern, our method can serve as a post-processing tool to further improve the resulting operator from the method of [Lescoat et al. 2020]. Given the original mesh with 26,000 vertices, our post-processed operators result in better functional maps (middle) compared to the output operators from [Lescoat et al. 2020] (left), as well as closer eigenvalues (right) to the reference.

$\|\bullet\|_L = 1.73 \times 10^{-2}$ $\|\bullet\|_L = \mathbf{4.30 \times 10^{-3}}$
$\|\bullet\|_D = 3.09 \times 10^{0}$ $\|\bullet\|_D = \mathbf{1.69 \times 10^{0}}$

50K → 0.8K

$\|\bullet\|_L = 1.04 \times 10^{-2}$ $\|\bullet\|_L = \mathbf{4.34 \times 10^{-3}}$
$\|\bullet\|_D = 3.37 \times 10^{0}$ $\|\bullet\|_D = \mathbf{2.74 \times 10^{0}}$
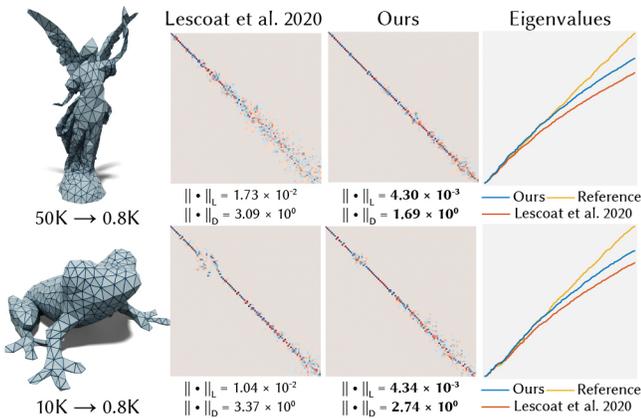
10K → 0.8K

Fig. 17. When the coarsening is aggressive, our method can still postprocess the results of [Lescoat et al. 2020] to improve the quality of spectral preservation.

from the results of [Lescoat et al. 2020] in a global manner to further improve the spectral preservation while keeping the sparsity pattern and the mesh vertices fixed. We further demonstrate the improvement of the spectral preservation by visualizing the biharmonic distance of our method and [Garland and Heckbert 1997] (Fig. 14) or [Lescoat et al. 2020] (Fig. 15) using the same 1-ring sparsity pattern. Our method can also recover the spectral properties when

$\|\bullet\|_L = 6.56 \times 10^{-2}$ $\|\bullet\|_L = \mathbf{1.72 \times 10^{-3}}$ $\|\bullet\|_L = 1.06 \times 10^{-2}$ $\|\bullet\|_L = \mathbf{7.80 \times 10^{-5}}$
$\|\bullet\|_D = 1.59 \times 10^{1}$ $\|\bullet\|_D = \mathbf{1.06 \times 10^{0}}$ $\|\bullet\|_D = 1.98 \times 10^{0}$ $\|\bullet\|_D = \mathbf{5.65 \times 10^{-2}}$

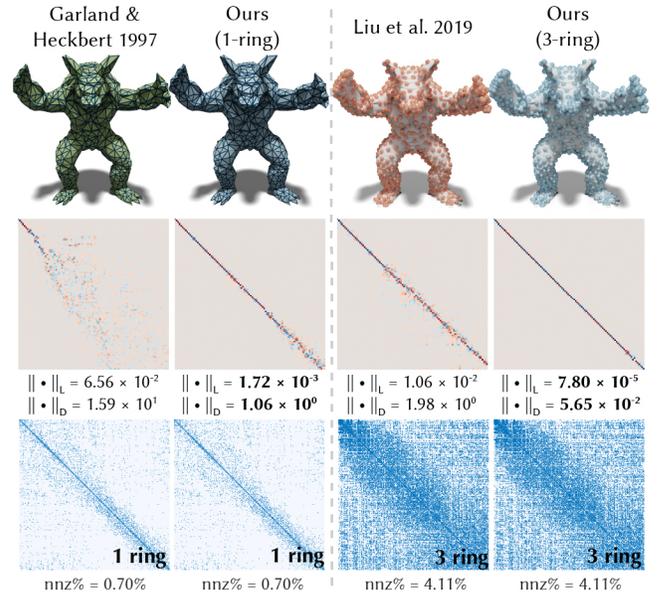nnz% = 0.70%   nnz% = 0.70%   nnz% = 4.11%   nnz% = 4.11%

Fig. 18. We simplify the anisotropic Laplacian (with parameter 20) from 50,000 vertices to 1,000 vertices using the same sparsity pattern as [Garland and Heckbert 1997] or [Liu et al. 2019]. Our method can handle anisotropic operators where [Garland and Heckbert 1997] may fail entirely due to the anisotropy. Our optimization scheme enables users to freely choose between 1-ring or 3-ring sparsity. In contrast, [Liu et al. 2019] has much less control on the sparsity pattern and only allows for 3-ring sparsity pattern, which introduces a significant amount of fill-ins.

$\|\bullet\|_L = 4.52 \times 10^{-3}$ $\|\bullet\|_L = 5.12 \times 10^{-4}$ $\|\bullet\|_L = 9.70 \times 10^{-5}$ $\|\bullet\|_L = \mathbf{7.41 \times 10^{-5}}$ $\|\bullet\|_L = 2.49 \times 10^{-4}$
$\|\bullet\|_D = 3.25 \times 10^{1}$ $\|\bullet\|_D = 6.76 \times 10^{0}$ $\|\bullet\|_D = 2.78 \times 10^{0}$ $\|\bullet\|_D = \mathbf{2.28 \times 10^{0}}$ $\|\bullet\|_D = 6.39 \times 10^{0}$

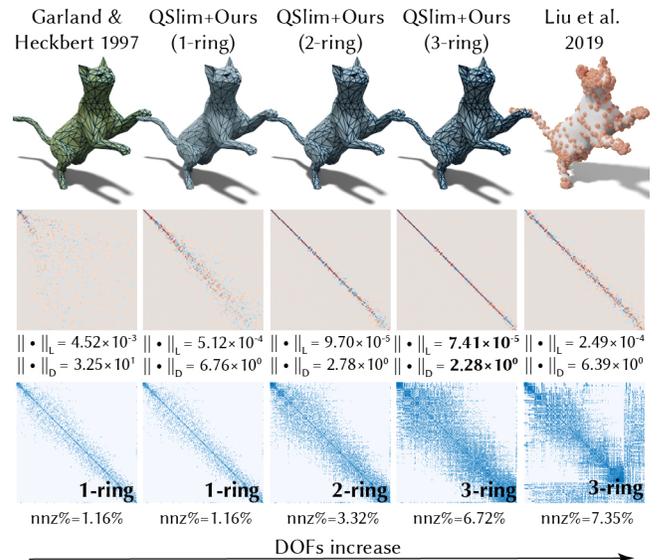nnz%=1.16%   nnz%=1.16%   nnz%=3.32%   nnz%=6.72%   nnz%=7.35%

DOFs increase

Fig. 19. Our optimization achieves better spectral preservation of the anisotropic Laplace operator (with parameter 60, from 5000 vertices to 400 vertices) when the rings of neighborhood increases. Increasing the nonzeros in the sparsity pattern will allow for more degrees of freedom, which enables our solver to converge to a better result.
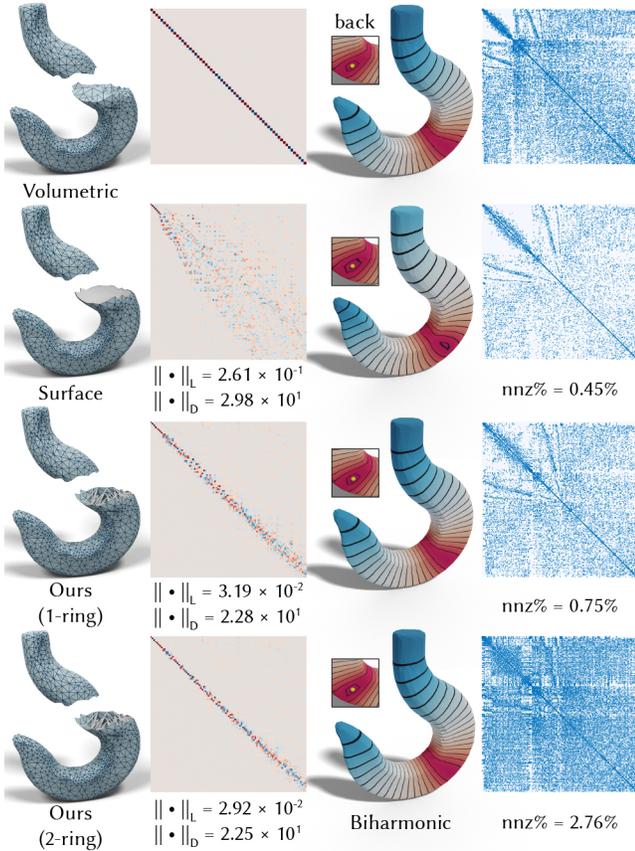
Fig. 20. Our method can encode the spectral behavior of a volumetric mesh only using its surface mesh with some added links. We approximate the volumetric behavior using a *sparse* matrix with a controllable sparsity pattern, while the corresponding matrix has to be *dense* in the traditional Boundary Element Method [James and Pai 1999]. Here the source vertex of the biharmonic distance is visualized as a green dot, and the added links are visualized as the gray lines (bottom two).

the coarsening is extreme for complicated shapes (see Fig. 17 and Fig. 26). In addition to the isotropic cotangent Laplacian, in Fig. 18 we demonstrate our capability in handling anisotropic operators without introducing any new fill-ins.

For downstream applications that accept changes in the sparsity pattern, our method enables one to freely control the sparsity patterns to achieve better results. As shown in Fig. 19, we can freely increase the sparsity pattern from 1 ring to 3 rings in order to allow more degrees of freedom and better results. But one should also consider the trade-off between the number of non-zero fill-ins and the quality of the results because more degrees of freedom implies a denser output operator with a longer runtime (see Fig. 13).

5.4 Volume to Surface

Surface-only representation is a more efficient alternative compared to its volumetric counterpart because three dimensional (volumetric) problem is reduced to two dimensions (surface). However, in computer animation and simulation, it is often more desirable to use
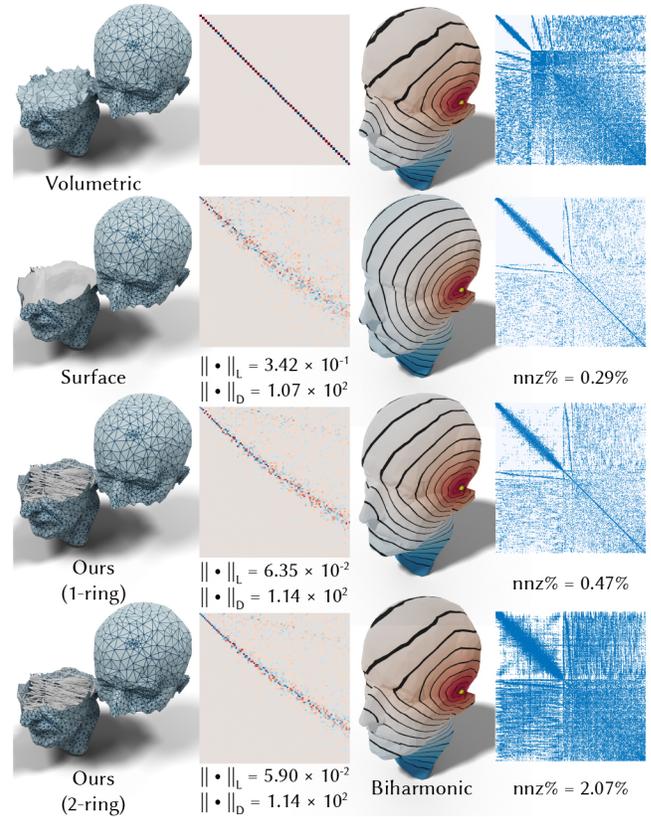

Fig. 21. Starting from the constrained Delaunay tetrahedralization, we can increase the number of rings of neighborhood to better approximate the volumetric Laplacian using a surface mesh with random links. Similar to the partial functional correspondence in [Rodolà et al. 2017], the diagonal of our functional map may be skewed because we may lose some internal eigenvectors during this partial matching.

a volumetric representations to simulate the volumetric behavior. We show that our approach can optimize the Laplacian of a surface-only mesh with random distant connections generated via TetGen [Si 2015] to approximate the spectral behavior of a volumetric mesh.

Taking the boundary surface mesh of a volumetric tetrahedral mesh as the input, we first add distant edges to the surface Laplacian to determine the sparsity pattern. We use the constrained Delaunay tetrahedralization in TetGen [Si

2015] to add the edges between "visible" but distant vertices (see inset), and use its pattern as the sparsity pattern of our modified surface Laplacian. Then we optimize the modified operator to preserve the spectral behavior of the volumetric Laplacian. Compared to the traditional discretization in Boundary Element Method [James and Pai 1999] where the boundary matrices are usually dense, in our method the surface-only Laplacian can still remain sparse and maintain a similar sparsity pattern as its surface cotangent Laplacian.

In Fig. 20 and Fig. 21, we visualize the functional map and biharmonic distance of our optimized surface Laplacian. We show that
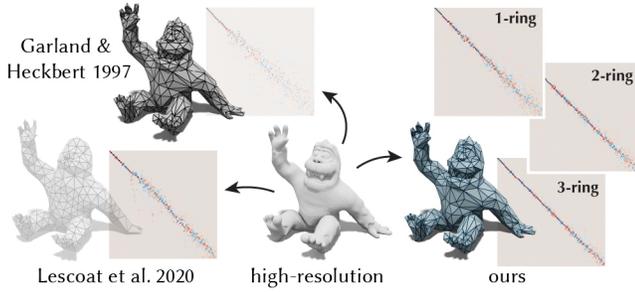
Fig. 22. When one ties the differential operator with the mesh, we can either preserves the appearance of the mesh [Garland and Heckbert 1997] or the spectral properties of the operator [Lescoat et al. 2020], but not both. Our approach enables one to detach the operator from the mesh (right) to achieve both simultaneously: using an appearance-preserving mesh for visualization and a spectral-preserving operator with user-desired sparsity patterns (*e.g.*, 1-ring, 2-ring, or 3-ring) for computation.

we can further capture the volumetric behavior by increasing the rings of neighborhood.

Similar to [Rodolí et al. 2017], our volume-to-surface mapping is also a partial functional mapping, which may lose some (internal) eigenvectors and result in a skewed functional map when the internal volume is large (see Fig. 21). Our method can also serve as a possible way to generate training data to find the best sparsity pattern without the presence of a volumetric mesh.

5.5 Operator Detachment

Sharp et al. [2019] propose to represent the same geometry using two discrete representations: one for visualization and one for computation. In a similar spirit to [Sharp et al. 2019], our approach enables one to have one mesh for visualization and one detached operator for computation.

Previous decimation methods either preserve the appearance but fail in preserving spectral properties or preserve the spectral properties but fail in preserving the appearance. This is partly due to the perspective of defining the operator directly on the discrete mesh, and partly due to the lack of tools to optimize the operator independently.

In order to simultaneously preserve the appearance and the spectral properties, in Fig. 22 we first obtain a coarsened mesh from an appearance-preserving decimation, then we optimize the operator separately using the sparsity pattern defined by the connectivity of the mesh. Intuitively, this optimization tries to retrieve the desired properties on the original mesh by manipulating the metric "seen" by the coarsened operator. At the end of this process, even though the "distorted" metric may not be embeddable, one can always use the embeddable appearance-preserving mesh to visualize the results of the computation. In Fig. 22, this detachment allows us to preserve both the appearance and the spectral properties, while the method of [Garland and Heckbert 1997] fails in preserving spectral properties and the method of [Lescoat et al. 2020] fails in preserving the appearance. In Fig. 1, we demonstrate the strength of this approach in approximating the vibration modes of a high-resolution mesh using a coarse mesh with a detached coarsened operator. Compared
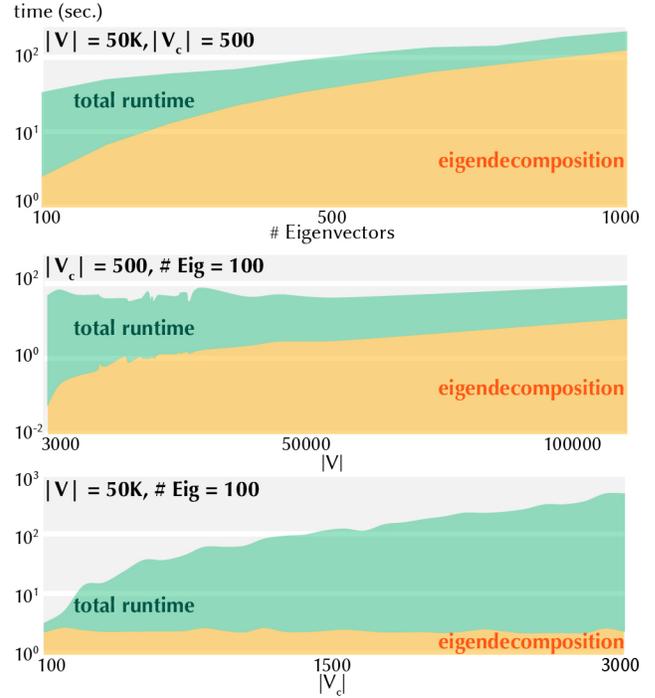

Fig. 23. Our runtime shows that our method is more suitable for aggressive coarsening (middle). When many eigenvectors are in use (top) or input meshes are large (middle), computing eigendecomposition can be the bottleneck.

to [Liu et al. 2019] which does not allow inputting an arbitrary sparsity pattern (instead it builds the output sparsity pattern by "squaring" an incidence matrix, see their Eq. 7), our method can take any sparsity pattern as input. This means one can geometrically simplify a mesh, then use that new mesh's sparsity pattern as input to our algorithm to optimize a compatible operator (see Fig. 1), which enables its use in applications that require an embedded mesh and an accurate coarse operator (*e.g.*, simulation with contact handling).

6 LIMITATIONS & FUTURE WORK

Further exploiting the limited degrees of freedom would enable an even better spectral preservation for 1-ring isotropic operator. Jointly optimizing the sparsity pattern and the operator entries may lead to even finer solutions, especially for volume-to-surface approximation. Exploring different regularizers and energy formulations would be desirable for solving the underdetermined system when degrees of freedom are too large compared to the number of eigenvectors in use. Avoid introducing additional low frequency eigenvectors during the optimization would benefit the downstream applications (see the inset). Reducing the memory consumption of the Kronecker product would further increase the scalability of our method (see App. **??**). Incorporat-

21K → 0.8K   1 ring

ing a fast eigen-approximation or removing the use of eigen decomposition would further accelerate the spectral coarsening (see
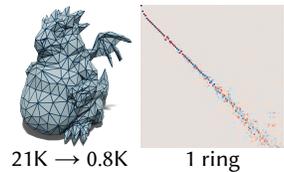
Fig. 23). Further analysis of the tradeoff between the convergence and the number of cliques could offer insight towards future applications of chordal decomposition. Extending our spectral coarsening of surface-based geometric operators to volumetric stiffness matrix could also provide an alternative way to deal with the numerical stiffening in simulation. As a first order method, ADMM is slow to obtain highly accurate solutions, but fast in getting moderately accurate solutions. Similar to other splitting methods, ADMM is sensitive to the conditioning of the problem data. Thus adding a preconditioner could make our solver more robust to the scaling problem and increase its performance. Finally, it would be also interesting to extend our method to many other applications beyond geometry processing and shape matching, such as physics-based simulation, topology optimization, algebraic multigrid and spectral graph reduction.

ACKNOWLEDGMENTS

REFERENCES

Jim Agler, William Helton, Scott McCullough, and Leiba Rodman. 1988. Positive semidefinite matrices with a given sparsity pattern. *Linear algebra and its applications* 107 (1988), 101–149.

Martin S. Andersen, Joachim Dahl, and Lieven Vandenberghe. 2010. Implementation of nonsymmetric interior-point methods for linear optimization over sparse matrix cones. *Math. Program. Comput.* 2, 3-4 (2010), 167–201.

Jean RS Blair and Barry Peyton. 1993. An introduction to chordal graphs and clique trees. In *Graph theory and sparse matrix computation*. 1–29.

Stephen P. Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. 2011. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. *Foundations and Trends in Machine Learning* 3, 1 (2011), 1–122.

Gecia Bravo-Hermsdorff and Lee Gunderson. 2019. A Unifying Framework for Spectrum-Preserving Graph Sparsification and Coarsening. In *Advances in Neural Information Processing Systems 32*. 7736–7747.

Max Budninskiy, Houman Owhadi, and Mathieu Desbrun. 2019. Operator-adapted wavelets for finite-element differential forms. *J. Comput. Phys.* 388 (2019), 144–177.

Samuel Burer. 2003. Semidefinite Programming in the Space of Partial Positive Semidefinite Matrices. *SIAM Journal on Optimization* 14, 1 (2003), 139–172.

Desai Chen, David I. W. Levin, Wojciech Matusik, and Danny M. Kaufman. 2017. Dynamics-aware numerical coarsening for fabrication design. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 84:1–84:15.

Desai Chen, David I. W. Levin, Shinjiro Sueda, and Wojciech Matusik. 2015. Data-driven finite elements for geometry and material design. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 74:1–74:10.

Jiong Chen, Hujun Bao, Tianyu Wang, Mathieu Desbrun, and Jin Huang. 2018. Numerical Coarsening Using Discontinuous Shape Functions. *ACM Transactions on Graphics (TOG)* 37, 4, Article 120 (July 2018), 12 pages.

Jiong Chen, Max Budninskiy, Houman Owhadi, Hujun Bao, Jin Huang, and Mathieu Desbrun. 2019a. Material-Adapted Refinable Basis Functions for Elasticity Simulation. *ACM Transactions on Graphics (TOG)* 38, 6, Article Article 161 (Nov. 2019), 15 pages.

Yu Ju Edwin Chen, David I. W. Levin, Danny Kaufmann, Uri M. Ascher, and Dinesh K. Pai. 2019b. EigenFit for consistent elastodynamic simulation across mesh resolution. In *Proceedings of the 18th annual ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA 2019*. 5:1–5:13.

Paolo Cignoni, Claudio Montani, and Roberto Scopigno. 1998. A comparison of mesh simplification algorithms. *Comput. Graph.* 22, 1 (1998), 37–54.

Jonathan D. Cohen, Dinesh Manocha, and Marc Olano. 2003. Successive Mappings: An Approach to Polygonal Mesh Simplification with Guaranteed Error Bounds. *Int. J. Comput. Geometry Appl.* 13, 1 (2003), 61.

David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. 2004. Variational shape approximation. *ACM Transactions on Graphics (TOG)* 23, 3 (2004), 905–914.

Katsuki Fujisawa, Sunyoung Kim, Masakazu Kojima, Yoshio Okamoto, and Makoto Yamashita. 2009. B-453 User's Manual for SparseCoLO: Conversion Methods for SPARSE COnic-form Linear Optimization Problems. (2009).

Mituhiro Fukuda, Masakazu Kojima, Kazuo Murota, and Kazuhide Nakata. 2001. Exploiting Sparsity in Semidefinite Programming via Matrix Completion I: General Framework. *SIAM Journal on Optimization* 11, 3 (2001), 647–674.

Michael Garland and Paul S. Heckbert. 1997. Surface simplification using quadric error metrics. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1997*. 209–216.

Michael Garland and Paul S. Heckbert. 1998. Simplifying surfaces with color and texture using quadric error metrics. In *Visualization '98, Proceedings*. 263–269.

Michael Grant and Stephen Boyd. 2008. Graph implementations for nonsmooth convex programs. In *Recent Advances in Learning and Control*. 95–110.

Michael Grant and Stephen Boyd. 2014. CVX: Matlab Software for Disciplined Convex Programming, version 2.1.

Robert Grone, Charles R Johnson, Eduardo M Sá, and Henry Wolkowicz. 1984. Positive definite completions of partial Hermitian matrices. *Linear algebra and its applications* 58 (1984), 109–124.

Pinar Heggernes. 2006. Minimal triangulations of graphs: A survey. *Discret. Math.* 306, 3 (2006), 297–317.

Hugues Hoppe. 1996. Progressive Meshes. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1996*. 99–108.

Hugues Hoppe. 1997. View-dependent refinement of progressive meshes. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1997*. 189–198.

Hugues Hoppe, Tony DeRose, Tom Duchamp, John Alan McDonald, and Werner Stuetzle. 1993. Mesh optimization. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1993*. 19–26.

Alec Jacobson et al. 2018. gptoolbox: Geometry Processing Toolbox. http://github.com/alecjacobson/gptoolbox.

Doug L. James and Dinesh K. Pai. 1999. ArtDefo: Accurate Real Time Deformable Objects. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '99)*. 65–72.

Yu Jin, Andreas Loukas, and Joseph JáJá. 2020. Graph Coarsening with Preserved Spectral Properties. In *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020*, Vol. 108. 4452–4462.

Naonori Kakimura. 2010. A direct proof for the matrix decomposition of chordal-structured positive semidefinite matrices. *Linear Algebra Appl.* 433, 4 (2010), 819–823.

Liliya Kharevych, Patrick Mullen, Houman Owhadi, and Mathieu Desbrun. 2009. Numerical coarsening of inhomogeneous elastic materials. *ACM Transactions on Graphics (TOG)* 28, 3 (2009), 51.

Sunyoung Kim, Masakazu Kojima, Martin Mevissen, and Makoto Yamashita. 2011. Exploiting sparsity in linear and nonlinear matrix inequalities via positive semidefinite matrix completion. *Math. Program.* 129, 1 (2011), 33–68.

Thibault Lescoat, Hsueh-Ti Derek Liu, Jean-Marc Thiery, Alec Jacobson, Tamy Boubekeur, and Maks Ovsjanikov. 2020. Spectral Mesh Simplification. *Computer Graphics Forum* 39, 2 (2020), 49–58.

Dingzeyu Li, Yun (Raymond) Fei, and Changxi Zheng. 2015. Interactive Acoustic Transfer Approximation for Modal Sound. *ACM Transactions on Graphics (TOG)* 35, 1 (2015), 2:1–2:16.

Hsueh-Ti Derek Liu, Alec Jacobson, and Maks Ovsjanikov. 2019. Spectral Coarsening of Geometric Operators. *ACM Transactions on Graphics (TOG)* 38, 4, Article Article 105 (July 2019), 13 pages.

Andreas Loukas. 2019. Graph Reduction with Spectral and Cut Guarantees. *J. Mach. Learn. Res.* 20 (2019), 116:1–116:42.

Andreas Loukas and Pierre Vandergheynst. 2018. Spectrally Approximating Large Graphs with Smaller Graphs. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, Vol. 80. 3243–3252.

Yunlong Lu, Qing Liu, Yi Wang, Peter Gardner, Wang He, Yi Chen, Jifu Huang, and Taijun Liu. 2020. Seamless Integration of Active Antenna With Improved Power

Efficiency. *IEEE Access* 8 (2020), 48399–48407.

Ramtin Madani, Abdulrahman Kalbat, and Javad Lavaei. 2015. ADMM for sparse semidefinite programming with applications to optimal power flow problem. In *54th IEEE Conference on Decision and Control, CDC 2015*. 5932–5939.

Haggai Maron, Nadav Dym, Itay Kezurer, Shahar Kovalsky, and Yaron Lipman. 2016. Point Registration via Efficient Convex Relaxation. *ACM Transactions on Graphics (TOG)* 35, 4, Article Article 73 (July 2016), 12 pages.

Kazuhide Nakata, Katsuki Fujisawa, and Mituhiro Fukuda. 2003. Exploiting sparsity in semidefinite programming via matrix completion II: implementation and numerical results. *Math. Program.* 95, 2 (2003), 303–327.

Ahmad Nasikun, Christopher Brandt, and Klaus Hildebrandt. 2018. Fast Approximation of Laplace-Beltrami Eigenproblems. *Comput. Graph. Forum* 37, 5 (2018), 121–134.

Maks Ovsjanikov, Mirela Ben-Chen, Justin Solomon, Adrian Butscher, and Leonidas J. Guibas. 2012. Functional maps: a flexible representation of maps between shapes. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 30:1–30:11.

Maks Ovsjanikov, Etienne Corman, Michael M. Bronstein, Emanuele Rodolà, Mirela Ben-Chen, Leonidas J. Guibas, Frédéric Chazal, and Alexander M. Bronstein. 2017. Computing and processing correspondences with functional maps. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference, SIGGRAPH '17 Courses*. 5:1–5:62.

Houman Owhadi. 2017. Multigrid with Rough Coefficients and Multiresolution Operator Decomposition from Hierarchical Information Games. *SIAM Rev.* 59, 1 (2017), 99–149.

A. Cengiz Öztireli, Marc Alexa, and Markus H. Gross. 2010. Spectral sampling of manifolds. *ACM Transactions on Graphics (TOG)* 29, 6 (2010), 168.

E. Rodolí, L. Cosmo, M. M. Bronstein, A. Torsello, and D. Cremers. 2017. Partial Functional Correspondence. *Comput. Graph. Forum* 36, 1 (Jan. 2017), 222–236.

Nicholas Sharp, Yousuf Soliman, and Keenan Crane. 2019. Navigating intrinsic triangulations. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 55.

Hang Si. 2015. TetGen, a Delaunay-Based Quality Tetrahedral Mesh Generator. *ACM Trans. Math. Softw.* 41, 2, Article 11 (Feb. 2015), 36 pages.

Gun Srijuntongsiri and Stephen A. Vavasis. 2004. A Fully Sparse Implementation of a Primal-Dual Interior-Point Potential Reduction Method for Semidefinite Programming. *CoRR* (2004).

Yifan Sun. 2015. *Decomposition methods for semidefinite optimization.* Ph.D. Dissertation. UCLA.

Yifan Sun, Martin S. Andersen, and Lieven Vandenberghe. 2014. Decomposition in Conic Optimization with Partially Separable Structure. *SIAM Journal on Optimization* 24, 2 (2014), 873–897.

Yifan Sun and Lieven Vandenberghe. 2015. Decomposition Methods for Sparse Matrix Nearness Problems. *SIAM J. Matrix Analysis Applications* 36, 4 (2015), 1691–1717.

Lieven Vandenberghe and Martin S. Andersen. 2015. Chordal Graphs and Semidefinite Optimization. *Foundations and Trends in Optimization* 1, 4 (2015), 241–433.

M. Yannakakis. 1981. Computing the Minimum Fill-In is NP-Complete. *SIAM Journal on Algebraic Discrete Methods* 2, 1 (1981), 77–79.

Zhiqiang Zhao, Yongyu Wang, and Zhuo Feng. 2018. Nearly-Linear Time Spectral Graph Reduction for Scalable Graph Partitioning and Data Visualization. *CoRR* (2018).

Yang Zheng, Giovanni Fantuzzi, and Antonis Papachristodoulou. 2017a. Exploiting sparsity in the coefficient matching conditions in sum-of-squares programming using ADMM. *IEEE control systems letters* 1, 1 (2017), 80–85.

Yang Zheng, Giovanni Fantuzzi, and Antonis Papachristodoulou. 2019. Fast ADMM for Sum-of-Squares Programs Using Partial Orthogonality. *IEEE Trans. Automat. Contr.* 64, 9 (2019), 3869–3876.

Yang Zheng, Giovanni Fantuzzi, Antonis Papachristodoulou, Paul Goulart, and Andrew Wynn. 2017b. Fast ADMM for semidefinite programs with chordal sparsity. In *2017 American Control Conference*. 3335–3340.

Yang Zheng, Giovanni Fantuzzi, Antonis Papachristodoulou, Paul Goulart, and Andrew Wynn. 2020. Chordal decomposition in operator-splitting methods for sparse semidefinite programs. *Math. Program.* 180, 1 (2020), 489–532.

Yang Zheng, Richard P. Mason, and Antonis Papachristodoulou. 2018. Scalable Design of Structured Controllers Using Chordal Decomposition. *IEEE Trans. Automat. Contr.* 63, 3 (2018), 752–767.

A ALTERNATING DIRECTION METHOD OF MULTIPLIERS

Alternating direction method of multipliers (ADMM) solves optimization problems in the following format

$$\min_{\mathbf{x},\mathbf{z}} f(\mathbf{x}) + g(\mathbf{z}) \qquad (33)$$

$$\text{s.t.} \quad \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} = \mathbf{c}. \qquad (34)$$

The (scaled) ADMM solves the problem by iteratively applying the following steps

$$\mathbf{x}^{t+1} := \arg\min_{\mathbf{x}} \left(f(\mathbf{x}) + \frac{\rho}{2} \|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z}^t - \mathbf{c} + \mathbf{u}^t\|_2^2 \right)$$

$$\mathbf{z}^{t+1} := \arg\min_{\mathbf{z}} \left(g(\mathbf{z}) + \frac{\rho}{2} \|\mathbf{A}\mathbf{x}^{t+1} + \mathbf{B}\mathbf{z} - \mathbf{c} + \mathbf{u}^t\|_2^2 \right) \qquad (35)$$

$$\tilde{\mathbf{u}}^{t+1} := \mathbf{u}^t + \mathbf{A}\mathbf{x}^{t+1} + \mathbf{B}\mathbf{z}^{t+1} - \mathbf{c}$$

$$\rho^{t+1}, \mathbf{u}^{t+1} := \text{update}(\rho^t),$$

where ρ is the *penalty parameter* and $\mathbf{u}$ is the *scaled dual variable*. In the last step, a common strategy is to update the penalty ρ as

$$\rho^{t+1} = \begin{cases} \tau^{\text{incr}} \rho^t & \text{if } \|\mathbf{r}^t\|_2 > \mu \|\mathbf{s}^t\|_2 \\ \rho^t / \tau^{\text{decr}} & \text{if } \|\mathbf{s}^t\|_2 > \mu \|\mathbf{r}^t\|_2 \\ \rho^t & \text{otherwise,} \end{cases} \qquad (36)$$

where $\tau^{\text{incr}} > 1$, $\tau^{\text{decr}} > 1$, $\mu > 1$ are parameters, $\mathbf{r}$ and $\mathbf{s}$ are the *primal residual* and the *dual residual*, respectively. We can compute them as

$$\mathbf{r}^{t+1} = \mathbf{A}\mathbf{x}^{t+1} + \mathbf{B}\mathbf{z}^{t+1} - \mathbf{c}, \quad \mathbf{s}^{k+1} = \rho \mathbf{A}^\top \mathbf{B}(\mathbf{z}^{t+1} - \mathbf{z}^t). \qquad (37)$$

After updating ρ we must also scale the dual variable $\mathbf{u}$ as

$$\mathbf{u}^{t+1} = \tilde{\mathbf{u}}^{t+1} \times \frac{\rho^t}{\rho^{t+1}}. \qquad (38)$$

A common stopping criteria is when both $\|\mathbf{r}^t\|_2 < \epsilon^{\text{pri}}$ and $\|\mathbf{s}^t\|_2 < \epsilon^{\text{dual}}$ are below the thresholds $\epsilon^{\text{pri}}, \epsilon^{\text{dual}}$. We only review basic concepts of ADMM here for self-containedness. We wholeheartedly refer the reader to a great survey [Boyd et al. 2011] for more information on ADMM.

B CHANGE OF VARIABLES

We describe the details on how to apply change of variables and vectorization for the constraints presented in Eq. 11.

Given the matrices $\mathbf{P}_{\mathcal{E}}$ and $\mathbf{P}_{\mathcal{E}}^{-1}$ in Eq. 17, which allow us to go back and forth between $\text{vec}(\mathbf{X})$ and $\mathbf{x}_{\mathcal{E}}$, we can vectorize the equality constraint in Eq. 11 as

$$\text{vec}(\mathbf{X}\mathbf{v}) = \text{vec}(\mathbf{e}) \implies (\mathbf{v}^\top \otimes \mathbf{I}) \text{vec}(\mathbf{X}) = \mathbf{e} \qquad (39)$$

$$\implies \underbrace{(\mathbf{v}^\top \otimes \mathbf{I})\mathbf{P}_{\mathcal{E}}^{-1}}_{\mathbf{G}} \mathbf{x}_{\mathcal{E}} = \mathbf{e} \qquad (40)$$

$$\implies \mathbf{G}\mathbf{x}_{\mathcal{E}} = \mathbf{e}, \qquad (41)$$

where $\mathbf{I}$ is the identity matrix. For the chordal decomposition Eq. 15, we can directly apply the vectorization strategy discussed in Sec. 3.1 as

$$\text{vec}(\mathbf{X}) = \sum_{i=1}^{p} \text{vec}(\mathbf{P}_i^\top \mathbf{Z}_i \mathbf{P}_i) \implies \text{vec}(\mathbf{X}) = \sum_{i=1}^{p} \mathbf{K}_i \underbrace{\text{vec}(\mathbf{Z}_i)}_{\mathbf{z}_i} \qquad (42)$$

$$\implies \mathbf{P}_{\mathcal{E}}^{-1}\mathbf{x}_{\mathcal{E}} = \sum_{i=1}^{p} \mathbf{K}_i \mathbf{z}_i, \qquad (43)$$

where we define $\mathbf{z}_i := \text{vec}(\mathbf{Z}_i)$. Therefore we can easily rewrite the PSD constraint on $\mathbf{Z}_i$ as

$$\mathbf{Z}_i = \text{vec}^{-1}(\mathbf{z}_i) \in \mathbb{S}_+^{n_i}. \qquad (44)$$

Combining all these results gives us the formulae in Eq. 18.

C DERIVATION OF ADMM STEPS

Here we describe how to derive the ADMM steps (see Eq. 35) to solve the optimization in Eq. 24. Our derivation follows a similar strategy described in Sec. 4.2 [Zheng et al. 2020].

We start by introducing an auxiliary variable $\mathbf{y}$ such that

$$\min_{\mathbf{x}_{\mathcal{E}},\mathbf{y},\tilde{\mathbf{z}}} f(\mathbf{x}_{\mathcal{E}}) \tag{45}$$

$$\text{s.t.} \quad \mathbf{G}\mathbf{x}_{\mathcal{E}} = \mathbf{e} \tag{46}$$

$$\mathbf{P}_C \mathbf{P}_{\mathcal{E}}^{-1}\mathbf{x}_{\mathcal{E}} = \mathbf{P}_C \widetilde{\mathbf{K}}\mathbf{y} \tag{47}$$

$$\text{vec}^{-1}(\mathbf{Q}_i \tilde{\mathbf{z}}_i) \succeq 0, \quad i = 1, \cdots, p \tag{48}$$

$$\mathbf{y} = \tilde{\mathbf{z}}, \tag{49}$$

Then we introduce the indicator function $\delta_{\mathcal{W}}$ as

$$\delta_{\mathcal{W}}(x) = \begin{cases} 0, & x \in \mathcal{W} \\ \infty, & \text{otherwise} \end{cases}. \tag{50}$$

This allows us to rewrite Eq. 45 as

$$\min_{\mathbf{x}_{\mathcal{E}},\mathbf{y},\mathbf{z}} \underbrace{f(\mathbf{x}_{\mathcal{E}}) + \delta_{\mathbf{e}}(\mathbf{G}\mathbf{x}_{\mathcal{E}}) + \delta_{\mathbf{0}}(\mathbf{P}_C \mathbf{P}_{\mathcal{E}}^{-1}\mathbf{x}_{\mathcal{E}} - \mathbf{P}_C \widetilde{\mathbf{K}}\mathbf{y})}_{\text{function of } \mathcal{X} = \{\mathbf{x}_{\mathcal{E}}, \mathbf{y}\}}$$

$$+ \underbrace{\sum_{i=1}^{p} \delta_{+}(\text{vec}^{-1}(\mathbf{Q}_i \tilde{\mathbf{z}}_i))}_{\text{function of } \mathcal{Z} = \{\tilde{\mathbf{z}}\}}$$

$$\text{s.t.} \quad \mathbf{y} - \tilde{\mathbf{z}} = 0, \tag{51}$$

where we use δ_{+} to denote the indicator function for the PSD constraint. This format of the optimization enables us directly apply the ADMM step Eq. 35. In particular the update of $\mathcal{X} = \{\mathbf{x}_{\mathcal{E}}, \mathbf{y}\}$ is as follows

$$\underset{\mathbf{x}_{\mathcal{E}},\mathbf{y}}{\arg\min} \ f(\mathbf{x}_{\mathcal{E}}) + \frac{\rho}{2}\|\mathbf{y} - \tilde{\mathbf{z}}^t + \mathbf{u}^t\|_2^2$$

$$\text{s.t.} \quad \mathbf{G}\mathbf{x}_{\mathcal{E}} = \mathbf{e} \tag{52}$$

$$\mathbf{P}_C \mathbf{P}_{\mathcal{E}}^{-1}\mathbf{x}_{\mathcal{E}} - \mathbf{P}_C \widetilde{\mathbf{K}}\mathbf{y} = 0,$$

where the solution depends on the energy function f in use. In the case of spectral coarsening energy, this boils down to a single linear solve of the KKT system (see App. E). The update of $\mathcal{Z} = \{\tilde{\mathbf{z}}\}$ is

$$\underset{\tilde{\mathbf{z}}}{\arg\min} \ \sum_{i=1}^{p} \|\mathbf{y}_i^{t+1} - \tilde{\mathbf{z}}_i + \mathbf{u}_i^t\|_2^2 \tag{53}$$

$$\text{s.t.} \quad \mathbf{Q}_i \tilde{\mathbf{z}}_i \succeq 0 \quad i = 1, \cdots, p. \tag{54}$$

This can be solved by projecting a set of small dense matrices $\text{vec}^{-1}(\mathbf{Q}_i(\mathbf{y}_i^{t+1} + \mathbf{u}_i^t))$ to PSD, which requires us to solve the eigendecomposition and remove the negative eigenvalues. Note that this process can be solved efficiently because each matrix to be projected is small and this process can be trivially parallelized.

D $\arg\min_{\mathcal{X}}$ FOR SPECTRAL COARSENING

Applying ADMM to solve the spectral coarsening problem requires us to derive the update on $\mathcal{X}$ (see Eq. 52). We start by vectorizing the spectral coarsening energy Eq. 5 as

$$f(\mathbf{X}) = \frac{1}{2}\|\mathbf{R}\mathbf{M}^{-1}\mathbf{L}\Phi - \widetilde{\mathbf{M}}^{-1}\mathbf{X}\mathbf{R}\Phi\|_{\widetilde{\mathbf{M}}}^2 \tag{55}$$

$$= \frac{1}{2}\|\underbrace{\widetilde{\mathbf{M}}^{1/2}\mathbf{R}\mathbf{M}^{-1}\mathbf{L}\Phi}_{\mathbf{W}} - \underbrace{\widetilde{\mathbf{M}}^{-1/2}}_{\mathbf{V}}\mathbf{X}\underbrace{\mathbf{R}\Phi}_{\mathbf{U}}\|_F^2 \tag{56}$$

$$= \frac{1}{2}\|\mathbf{W} - \mathbf{V}\mathbf{X}\mathbf{U}\|_F^2 \tag{57}$$

$$= \frac{1}{2}\|\text{vec}(\mathbf{W}) - \text{vec}(\mathbf{V}\mathbf{X}\mathbf{U})\|_2^2 \tag{58}$$

$$= \frac{1}{2}\|\text{vec}(\mathbf{W}) - (\mathbf{U}^\top \otimes \mathbf{V})\,\text{vec}(\mathbf{X})\|_2^2. \tag{59}$$

We then apply change of variables in Eq. 17 to modify the energy as follows

$$f(\mathbf{x}_{\mathcal{E}}) = \frac{1}{2}\|\underbrace{\text{vec}(\mathbf{W})}_{\mathbf{w}} - \underbrace{(\mathbf{U}^\top \otimes \mathbf{V})\mathbf{P}_{\mathcal{E}}^{-1}}_{\mathbf{E}}\mathbf{x}_{\mathcal{E}}\|_2^2 = \frac{1}{2}\|\mathbf{w} - \mathbf{E}\mathbf{x}_{\mathcal{E}}\|_2^2. \tag{60}$$

Updating $\mathcal{X} = \{\mathbf{x}_{\mathcal{E}}, \mathbf{y}\}$ in the ADMM (Eq. 52) amounts to obtaining the minimizer of the following problem

$$\min_{\mathbf{x}_{\mathcal{E}},\mathbf{y}} \frac{1}{2}\|\mathbf{w} - \mathbf{E}\mathbf{x}_{\mathcal{E}}\|_2^2 + \frac{\rho}{2}\|\mathbf{y} - \tilde{\mathbf{z}}^t + \mathbf{u}^t\|_2^2 \tag{61}$$

$$\text{s.t.} \quad \mathbf{G}\mathbf{x}_{\mathcal{E}} = \mathbf{e}, \tag{62}$$

$$\underbrace{\mathbf{P}_C \mathbf{P}_{\mathcal{E}}^{-1}}_{\mathbf{C}}\mathbf{x}_{\mathcal{E}} - \underbrace{\mathbf{P}_C \widetilde{\mathbf{K}}}_{\mathbf{D}}\mathbf{y} = 0. \tag{63}$$

We first derive the Lagrangian with multipliers μ_1, μ_2 as

$$\mathcal{L}(\mathbf{x}_{\mathcal{E}}, \mathbf{y}, \mu_1, \mu_2) = \frac{1}{2}\|\mathbf{w} - \mathbf{E}\mathbf{x}_{\mathcal{E}}\|_2^2 + \frac{\rho}{2}\|\mathbf{y} - \tilde{\mathbf{z}}^t + \mathbf{u}^t\|_2^2 \tag{64}$$

$$+ \mu_1^\top(\mathbf{C}\mathbf{x}_{\mathcal{E}} - \mathbf{D}\mathbf{y}) + \mu_2^\top(\mathbf{G}\mathbf{x}_{\mathcal{E}}). \tag{65}$$

Setting the derivatives to zeros gives us

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}_{\mathcal{E}}} = 0 \implies \mathbf{E}^\top \mathbf{E}\mathbf{x}_{\mathcal{E}} + \mathbf{C}^\top \mu_1 + \mathbf{G}^\top \mu_2 = \mathbf{E}^\top \mathbf{w} \tag{66}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{y}} = 0 \implies \mathbf{y} = \tilde{\mathbf{z}} - \mathbf{u} + \frac{1}{\rho}\mathbf{D}^\top \mu_1, \tag{67}$$

$$\frac{\partial \mathcal{L}}{\partial \mu_1} = 0 \implies \mathbf{C}\mathbf{x}_{\mathcal{E}} - \mathbf{D}\mathbf{y} = 0, \tag{68}$$

$$\frac{\partial \mathcal{L}}{\partial \mu_2} = 0 \implies \mathbf{G}\mathbf{x}_{\mathcal{E}} = 0. \tag{69}$$

We can substitute the expression of $\mathbf{y}$ from $\partial \mathcal{L}/\partial \mathbf{y} = 0$ to $\partial \mathcal{L}/\partial \mu_1 = 0$ and then obtain a set of equations

$$\mathbf{E}^\top \mathbf{E}\mathbf{x}_{\mathcal{E}} + \mathbf{C}^\top \mu_1 + \mathbf{G}^\top \mu_2 = \mathbf{E}^\top \mathbf{w} \tag{70}$$

$$\mathbf{C}\mathbf{x}_{\mathcal{E}} - \frac{1}{\rho}\mathbf{D}\mathbf{D}^\top \mu_1 = \mathbf{D}(\tilde{\mathbf{z}} - \mathbf{u}) \tag{71}$$

$$\mathbf{G}\mathbf{x}_{\mathcal{E}} = 0. \tag{72}$$

This enables us to obtain the optimal $\mathbf{x}_{\mathcal{E}}^{\star}, \mu_1^{\star}$ via solving a linear system

$$\begin{bmatrix} \mathbf{E}^{\top}\mathbf{E} & \mathbf{C}^{\top} & \mathbf{G}^{\top} \\ \mathbf{C} & -1/\rho\,\mathbf{D}\mathbf{D}^{\top} & \mathbf{0} \\ \mathbf{G} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{\mathcal{E}} \\ \mu_1 \\ \mu_2 \end{bmatrix} = \begin{bmatrix} \mathbf{E}^{\top}\mathbf{w} \\ \mathbf{D}(\tilde{\mathbf{z}} - \mathbf{u}) \\ \mathbf{0} \end{bmatrix}. \qquad (73)$$

Then we can recover the optimal $\mathbf{y}^{\star}$ as

$$\mathbf{y} = \tilde{\mathbf{z}} - \mathbf{u} + \frac{1}{\rho}\mathbf{D}^{\top}\mu_1^{\star}. \qquad (74)$$

E $\arg\min_{\chi}$ FOR SPECTRAL COARSENING

When the number of eigenvectors that are chosen to preserve is large, the size of $\mathbf{U}^{\top} \otimes \mathbf{V}$ in Eq. 60 can be large. However, we can avoid explicitly construct $\mathbf{U}^{\top} \otimes \mathbf{V}$ by leveraging the fact that only $\mathbf{E}^{\top}\mathbf{E}$ and $\mathbf{E}^{\top}\mathbf{w}$ are used in the linear solve Eq. 73. By using the properties $(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = (\mathbf{A}\mathbf{C}) \otimes (\mathbf{B}\mathbf{D})$ and $(\mathbf{B} \otimes \mathbf{A})\text{vec}(\mathbf{X}) = \text{vec}(\mathbf{A}\mathbf{X}\mathbf{B})$, we can instead compute $\mathbf{E}^{\top}\mathbf{E}$ and $\mathbf{E}^{\top}\mathbf{w}$ as

$$\mathbf{E}^{\top}\mathbf{E} = ((\mathbf{U}^{\top} \otimes \mathbf{V})\mathbf{P}_{\mathcal{E}}^{-1})^{\top}(\mathbf{U}^{\top} \otimes \mathbf{V})\mathbf{P}_{\mathcal{E}}^{-1} \qquad (75)$$

$$= (\mathbf{P}_{\mathcal{E}}^{-1})^{\top}(\mathbf{U}^{\top} \otimes \mathbf{V})^{\top}(\mathbf{U}^{\top} \otimes \mathbf{V})\mathbf{P}_{\mathcal{E}}^{-1} \qquad (76)$$

$$= (\mathbf{P}_{\mathcal{E}}^{-1})^{\top}(\mathbf{U} \otimes \mathbf{V}^{\top})(\mathbf{U}^{\top} \otimes \mathbf{V})\mathbf{P}_{\mathcal{E}}^{-1} \qquad (77)$$

$$= (\mathbf{P}_{\mathcal{E}}^{-1})^{\top}((\mathbf{U}\mathbf{U}^{\top}) \otimes (\mathbf{V}^{\top}\mathbf{V}))\mathbf{P}_{\mathcal{E}}^{-1}, \qquad (78)$$

$$\mathbf{E}^{\top}\mathbf{w} = ((\mathbf{U}^{\top} \otimes \mathbf{V})\mathbf{P}_{\mathcal{E}}^{-1})^{\top}\text{vec}(\mathbf{W}) \qquad (79)$$

$$= (\mathbf{P}_{\mathcal{E}}^{-1})^{\top}(\mathbf{U} \otimes \mathbf{V}^{\top})\text{vec}(\mathbf{W}) \qquad (80)$$

$$= (\mathbf{P}_{\mathcal{E}}^{-1})^{\top}\text{vec}(\mathbf{V}^{\top}\mathbf{W}\mathbf{U}^{\top}), \qquad (81)$$

where the size of $(\mathbf{U}\mathbf{U}^{\top}) \otimes (\mathbf{V}^{\top}\mathbf{V})$ and $\mathbf{V}^{\top}\mathbf{W}\mathbf{U}^{\top}$ are independent of the number of eigenvectors we choose to preserve.

F IMPLEMENTATION

Our solver is implemented in MATLAB using gptoolbox [Jacobson et al. 2018]. We adapt the MATLAB code from [Sun and Vandenberghe 2015] to compute the chordal decompoistion. Runtimes for all the examples were reported on a MacBook Pro with an Intel i5 2.3GHz processor, 16GB of RAM and an Intel Iris Plus Graphics 655 GPU. Experiments for volume to surface were tested on a Linux workstation with an Dual 14 Core 2.2Ghz processor, 383GB of RAM and 2 Titan RTX 24GB GPU. We did not use multi-threading, though the projection to PSD cones can be easily parallelized using MATLAB MEX file. Since the KKT system matrix in $\arg\min_{\chi}$ remains the same until ρ is updated, we only perform numerical factorization when ρ is updated and reuse it until ρ changes again (usually after tens of iterations).

For consistency, we choose to evaluate all the results on the first 100 eigenvectors across the experiments unless specified otherwise. We preserve the first 100 eigenvectors for surface Laplacian in spectral coarsening and simplification, and use an increased number of eigenvectors for volumetric Laplacian or when the system goes underdetermined. We also normalize all the eigenvectors to have unit length and scale the mesh to ensure each vertex has unit area. For a fair comparison, we compare the runtime of our MATLAB implementation with the MATLAB implementation of [Liu et al. 2019]. When comparing against [Lescoat et al. 2020], we use their
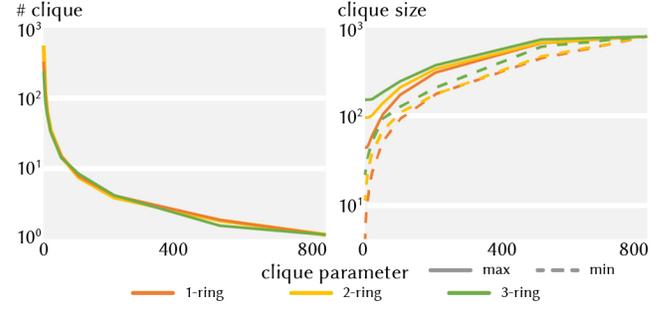

Fig. 24. We plot the change of the average number of cliques and the average maximal and minimal clique size with respect to clique parameters which we control in the chordal decomposition algorithm when coarsening various meshes to 800 vertices.
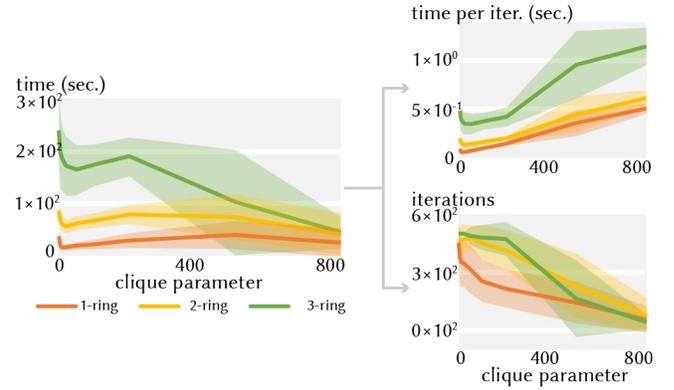

Fig. 25. We show the change of the total ADMM runtime, the ADMM runtime per iteration and the number of iterations with respect to clique parameters when coarsening a number of meshes to 800 vertices. Here the lines denote the average and the color regions denote the standard deviation.

decimation algorithm without edge flips and enable approximation of the minimizer on collapse edges.

In our implementation, the projecting of each clique matrix to PSD is relatively cheap because the size of clique matrix usually varies from tens to a few hundreds and can be controlled by the parameters during the clique merging stage of chordal decomposition. The size of the clique matrix after the chordal decomposition would be approximately around the clique merging parameters. In our experiments, we set the parameters for clique merging to be 200 so that the size of the clique matrix is in a few hundreds considering the tradeoff between eigendecomposition speed and convergence rate. As shown in Fig. 24 and Fig. 25, there is a non-monotonic relationship between the clique parameter and the ADMM runtime, and we experimentally determine that a parameter of 200 works best for all our examples. Optimal parameter determination is left for future work. We recommend setting the clique parameters to be larger than 100 when only preserving the first 100 eigenvectors to ensure our method converges. We also notice that increasing the number of eigenvectors preserved would lead to better convergence and avoid underdeterminism in the system. Let k be the number of the eigenvectors we choose to preserve and m be the number of
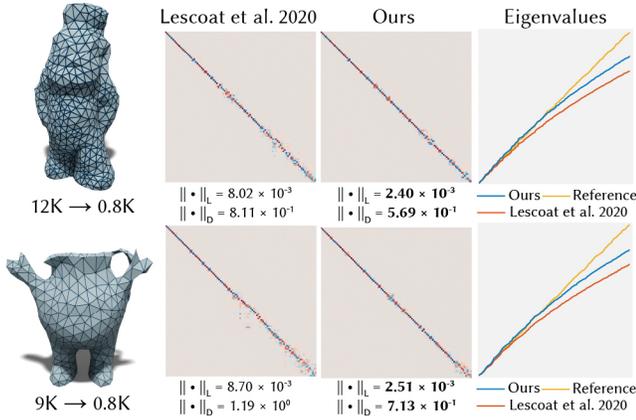
Fig. 26. Our algorithm can further improve the spectral properties of [Lescoat et al. 2020] as a post-processing step.
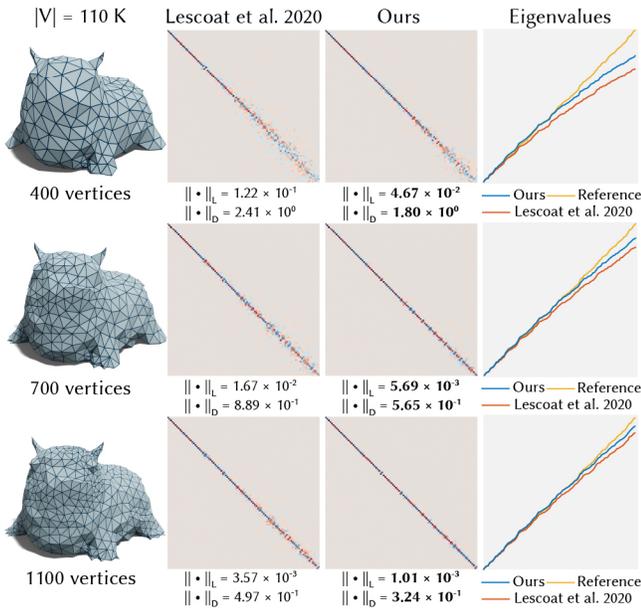

Fig. 27. Due to the freedom of choosing the output sparsity pattern, our method can serve as a post-processing tool to further improvze the resulting operator from the method of [Lescoat et al. 2020]. The results indicate that our post-processed operators result in better functional maps (middle) compared to the output operators from [Lescoat et al. 2020] (left), so as the eigenvalues (right).
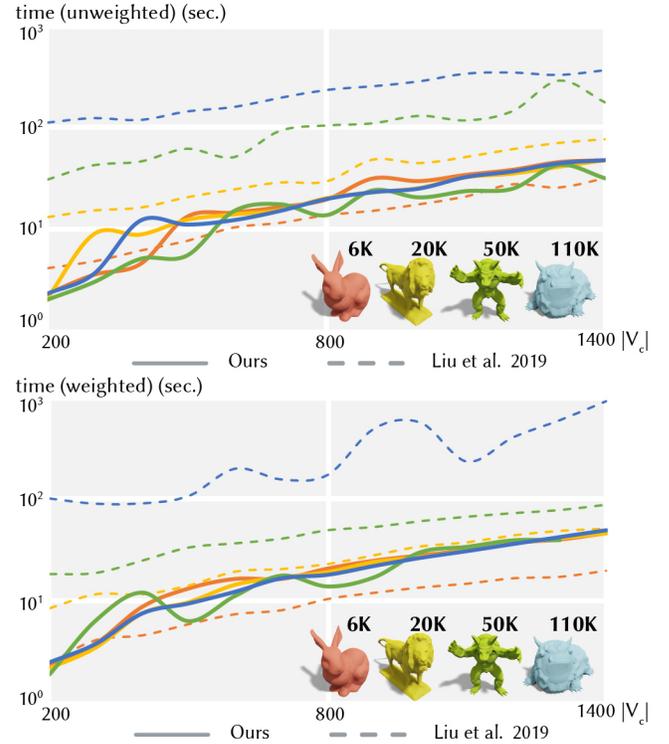

Fig. 28. We compare the runtime of our optimization algorithm of both the weighted and unweighted version with [Liu et al. 2019] using the same 3-ring sparsity pattern. Here we only consider the solve time, factoring out the precomputation for both our method and the method of [Liu et al. 2019].

G ADDITIONAL RESULTS

In addition to the results in Sec. 5, we report more results on the spectral simplification in Fig. 26 and Fig. 27 and an extended evaluation on runtime in Fig. 28 to complement the main text.

vertices in the coarsened domain. When the DOF defined by the sparsity pattern is large (*i.e.*, volumetric Laplacian, 3-ring surface Laplacian) , we recommend setting the number of the preserved eigenvectors to be $k > 0.5 * m$, and using the weighted energy to preserve the low-frequency modes. Experimentally, we observe that when the DOF is too large, the system may become underdetermined for volumetric mesh and 2- or 3-ring if $m > 2 \times k$.