GEOMETRIC DATA ANALYSIS ACROSS SCALES VIA LAPLACIAN EIGENVECTOR CASCADING*

Joshua L. Mike
Department of Computational Mathematics, Science and Engineering
Michigan State University
East Lansing, MI, USA.
mikejosh@msu.edu

Jose A. Perea
Department of Computational Mathematics, Science and Engineering
Michigan State University
East Lansing, MI, USA.
joperea@msu.edu

December 6, 2018

ABSTRACT

In this work, we develop a framework for constructing multiscale and geometrically meaningful functions on point cloud data. Specifically, a data set is represented as a family of graphs built at increasing scale, and we analyze the evolution of eigenspaces for the associated graph Laplacians. Eigenvectors of the graph Laplacian have been used extensively for nonlinear dimensionality reduction via spectral embeddings, performing diffusion in irregular spaces, identifying geometrically conspicuous regions in high-dimensional data, training in semi-supervised learning, and more. Leveraging ideas from computational topology, we build from a data set a sequence of graphs with varying coarseness levels, and relate them via simplicial maps. We cascade eigenvectors along these graph relations, not only to accelerate eigenvector computations, but also to yield consistent representations of the associated eigenspaces across scales. Theory is presented for eigenvector cascading along with general stability results and their application to our framework. For applications relying on several Laplacian eigenvectors, the LOBPCG (locally optimally block preconditioned conjugate gradient) method is employed for iterative eigenvector estimation. We present an application to multiscale TDA (topological data analysis) mapper, illustrating how our methodology identifies consistent features of mapper graphs at different resolutions.

Keywords Cover Tree · Multiscale graphs · Laplacian Eigenvectors · TDA Mapper · Persistent Homology.

1 Introduction

Many machine learning techniques purposefully view computational problems at multiple scales. Typically, the goal is to include important features at various resolutions [30, 41] or to increase computational accuracy and efficiency [39, 44]. While many state-of-the-art visualization and machine learning methods require dataset representations at a fixed scale [3, 47, 50, 35, 4, 11, 37], there is a lack of tools to validate a particular choice of resolution. One common representation for metric data is a graph, wherein nodes are data points and edges are typically drawn according to proximity predicates. With the addition of edge similarity weights, one can define graph Laplacian operators whose

*Support: NSF (DMS-1622301) and DARPA (HR0011-16-2-003).
eigenvectors and eigenvalues have been shown to encode the underlying geometry and topology of the data [40, 19]. Along these lines, we use a topological perspective — borrowing ideas from persistent homology and Hodge theory — to construct and relate graphs from a dataset viewed at multiple scales, by which we examine the progression of geometry from coarse to fine resolutions. The resulting graph relationships (simplicial maps) are used for two purposes. First, we show through several experiments that this multiscale approach can be used to speed-up Laplacian Eigenvector computations; and second, we describe a procedure to track weakly connected components in a multiscale graph in much the same fashion as 0-dimensional persistent homology — or hierarchical clustering for that matter — tracks connected components in a filtered space.

1.1 Prior Work

Graph Laplacian eigensystems have many applications including dimensionality reduction [47, 3], semi-supervised learning [4], consistent ranking [25], and direct quantification of geometry [7, 40, 12]. Still, computation with large, dense graphs often becomes prohibitive, since the data structure may grow quadratically in the number of vertices, and the associated graph Laplacian eigenvector problem grows in complexity even faster.

To alleviate computation, prior methods have viewed a graph from multiple scales to accelerate Laplacian eigenvector estimation. For example, [43] uses a divide-and-conquer approach, where a graph is divided into subgraphs and eigenvectors are computed on each cluster before tackling the full graph. In contrast, the works [49] and [28] both repeatedly collapse a graph by grouping vertices, compute eigenvectors on the coarsest graph, then use these to yield initial states in sequence for the finer graphs, until an eigensystem is estimated for the original graph. In [49] graphs are collapsed by combining vertices attached by heavily weighted edges, while in [28] a criterion called ‘relative connectivity’ is used to group vertices. The graph collapse used in the latter works is similar to the methodology used here, though the method and setting for collapse is markedly different. As part of our analysis, we extend the theory presented in [28] for our ambient method of graph collapse.

While the previous methods collapse or decompose a starting graph via edge predicates, here we build a series of graphs from a dataset in a metric space, and collapse according to cover set refinement. In addition, our primary goal is not simply to accelerate computation, but rather to capture the progression of Laplacian eigenvectors across multiple scales. Consequently, we perform a second form of eigenvector cascading which relates (clustered) eigenvectors between scales. When presented with a new dataset without a clear choice of scale, the persistence of features across multiple scales can be an important indicator of significance; indeed, a single object may exhibit important structures at various scales. In this fashion, one may also compare our methodology with the notion of persistent homology [14] which tracks the persistence of homological features as the scale changes. Since we are interested in all scales rather than only the finest one, the use of graph collapse is crucial to simplify computation at coarse scale, preventing Laplacian computation on a large and dense graph.

For large eigenvector problems, solving directly becomes intractable and so iterative methods are used instead. For example, the Lanczos method [29] is well-suited for large symmetric matrices, but each vector must be calculated in serial and the results suffer round-off errors as more eigenvectors are requested, though such round-off errors may be readjusted by a restart method [51]. Several methods for determining eigenvectors and eigenvalues can be jump-started via an initial state, such as Rayleigh quotient iteration [36] or conjugate gradient descent [42]. In particular, locally optimally block preconditioned conjugate gradient (LOBPCG) [27] is particularly well-suited for large-scale symmetric problems by utilizing parallel processing alongside a matrix’s sparse structure. Though LOBPCG has been shown to be very efficient in practice, there is still an incomplete theory to match its demonstrated efficacy [2]. Though more efficient than solving directly, iterative methods ‘wander’ within a spanning subspace if a group of eigenvalues are clustered or repeated, leading to slow and/or unstable results. Indeed, tracking the eigenspaces for repeated eigenvalues under any perturbation is highly nontrivial since they generically decompose [32].

1.2 Our Contribution

In this work we view a data set $X$ as a finite subset of a metric space $(\mathbb{X}, d)$. In particular, the 1-skeleton of the nerve complex $\mathcal{N}(\mathcal{U})$ (Def. 2.3) of a family $\mathcal{U} = \{U_j\}$ of open subsets of $\mathbb{X}$ with $X \subset \bigcup \mathcal{U}$ yields a graph representation for $X$. The Laplacian eigenvectors of this graph are viewed as functions on the vertices of $\mathcal{N}(\mathcal{U})$ and are then extended piecewise-linearly to the geometric realization of the nerve complex, $|\mathcal{N}(\mathcal{U})|$. Any partition of unity (Def. 2.4) subordinated to $\mathcal{U}$ yields a nerve map $\phi : \bigcup \mathcal{U} \to |\mathcal{N}(\mathcal{U})|$. Pre-composing the linearly extended eigenvectors with the nerve map defines an approximate eigenfunction on the data space. These approximate eigenfunctions can be viewed as geometric coordinates for the data set $X$, at the scale furnished by the covering $\mathcal{U}$. A brief vignette describing the nerve map and associated eigenfunctions is shown in Fig. 1.
Figure 1: The general pipeline for nerve mapping and defining approximate eigenfunctions is shown here. (A) A cover \( \mathcal{U} \) is provided for the dataset. (B) A partition of unity (PoU) subordinate to \( \mathcal{U} \) is defined. In practice, these are based on some geometrically determined bump functions. For example, the bumps shown here are Gaussian densities each fitted to an \( X \cap U_j \). (C) The PoU induces a map from \( \bigcup \mathcal{U} \) into \( |N(\mathcal{U})| \). (D) Edge weights induce a Laplacian operator and associated eigenvectors on the vertices of the nerve complex; extending linearly on the geometric nerve defines approximate eigenfunctions. (E) Composing with the nerve map defines the approximate eigenfunctions on the original dataset. Finally, Eigenfunction values act as geometric coordinates for the dataset.

The multiscale nature of our methodology stems from considering families of open covers \( \mathcal{U}^\alpha = \{ U_i^\alpha \} \) such that \( U_i \) refines \( U_i' \) whenever \( i \leq i' \). In topological terms, this series of cover refinements yields what we call a tower of covers. By factoring the cover refinements through the nerve construction, any function defined on the vertices of one graph (such as an eigenvector) canonically induces a function on the vertices of any finer graph. The details of the topological graph constructions are described in Section 2.

The cover tower (Def. 2.5) framework provides structure for two eigenvector cascade methods. First, eigenvectors from coarse scales are refined into initial states for calculation of eigenvectors at finer scales (Alg. 1). We show empirically (Ex. 3) that this first cascade method accelerates Laplacian eigenvector calculations when used with the LOBPCG algorithm [27]. While this is useful, it is not the primary goal of our work.

Instead, our goal is the correspondence of Laplacian eigenvectors as resolution progresses from coarse to fine. For high dimensional eigenspaces (or clustered eigenvalues) we orthogonally project the refined coarse eigenvectors onto the finer eigenspaces (Alg. 2). This second cascade of eigenvectors overcomes the non-uniqueness of eigenspace representation by choosing geometrically consistent basis vectors across scales. Double cascading is similar in spirit to the idea of persistent homology [14], where compatible homology bases are chosen consistently across a filtered space. We demonstrate with a few examples (Subsection 4.2) that double cascade can capture dataspace geometry continuously across scales despite the presence of tightly clustered or repeated eigenvalues. Provided theory (Cor. 1) supports the observed continuity, with the possibility for eigensystem jumps when eigenvalue groups cross or collide.

The methodology proposed here is instantiated by two sources of towers of covers: The cover tree construction (Def. 2.7), and TDA mapper [45]. We are able to provide a stability theory which is compatible with the cover tree construction (Subsection 3.4) and use this construction in the aforementioned cascade examples. Mapper uses a filter function and a cover of its range to generate overlapping regions in the data; the 1-skeleton of the resulting nerve complex is the output graph. The work [13] introduces a notion of multiscale mapper; their work explores the persistent homology of this object and proves its stability with respect to changes in the underlying dataset. By way of contrast, our focus is not only on the (co)homology of the mapper graph (captured by the kernel of the Laplacian), but also on weakly connected components (i.e. flares) and the other geometric features encoded in the eigenvectors of the Laplacian operator.

As an application of eigenvector cascade, we investigate the progression of Laplacian eigenvectors in multiscale mapper (subsection 4.3). Within this context, eigenvectors highlight weakly connected components in the mapper graph;
thus we track geometrically conspicuous regions of a dataset (with respect to the filter function) revealed by mapper and assess their stability to scale. In the limit of infinite data and infinitesimal covers, the mapper graph converges to the Reeb graph of the foliation of the dataspace [34]; our results suggest that the Laplacian eigenvectors converge in a fashion consistent with the geometry of this limiting Reeb graph.

2 Multiscale Graphs from Data: A topological perspective

2.1 Nerve Complexes and Nerve Maps

Here we describe the topological background for our graph constructions. First, we define the notion of a simplicial complex from both a combinatorial (i.e., abstract) and geometric point of view.

Definition 2.1. Combinatorially, a \( k \)-simplex \( \sigma \) is a collection of \( k + 1 \) distinct vertices \( \sigma = \{ v_0, \ldots, v_k \} \). Geometrically, \( \sigma \) is the convex hull of \( k + 1 \) affinely independent vertices \( v_0, \ldots, v_k \in \mathbb{R}^N \):

\[
\sigma = (v_0, \ldots, v_k) = \left\{ \sum_{j=0}^{k} \alpha_j v_j : \sum_{j=0}^{k} \alpha_j = 1 \text{ and } \alpha_j \geq 0 \forall j \right\}.
\]

The faces of a simplex consist of all simplices built from a subset of its vertices.

Low dimensional simplices \((k = 0, 1, 2, 3)\) are often called vertices, edges, triangles, and tetrahedra. Many simplices can be combined into a larger space called a simplicial complex; indeed, graphs are simplicial complexes containing only vertices and edges.

Definition 2.2. A combinatorial or abstract simplicial complex \( \mathcal{K} \) is a set of simplices satisfying the face relation: for any simplex \( \sigma \in \mathcal{K} \), \( \mathcal{K} \) contains all its faces. A geometric or realized simplicial complex \(|\mathcal{K}|\) has Euclidean coordinates for each vertex. Thus, any geometric simplicial complex is the union of its geometric simplices and additionally satisfies an embedding condition: the intersection of any two simplices is either empty or a common face.

A simplicial complex realized in \( \mathbb{R}^2 \) is shown in Fig. 2. Points within a geometric simplicial complex can be expressed on a large standard simplex via barycentric coordinates; such coordinates are expressed as the \( \alpha_i \) in Def. 2.2. For example, in Fig. 2, the point \([1 1 0 1 0 0 0 0 0] \) lies in the center of the triangle \( \{1, 2, 4\} \), while \([1 1 \cdots 1] \) is not a valid member of \( \mathcal{K} \).

![Figure 2: A simplicial complex with 9 numbered vertices. Triangles (2-simplices) are shown in grey.](image)

To build a simplicial complex from a dataset, consider a collection of open sets \( \mathcal{U} = \{ U_j \}_{j=1}^{N} \) within a topological space \( \mathcal{X} \), so that \( \bigcup \mathcal{U} \) contains the data. In practice, each open set \( U_j \) is a region of the dataspace such as a cluster or neighborhood. The nerve of this collection discretely describes its geometry via set intersections. An example is shown in Fig. 3.

Definition 2.3. For a collection of sets \( \mathcal{U} = \{ U_j \} \), the nerve \( \mathcal{N}(\mathcal{U}) \) is the abstract simplicial complex with simplices

\[
\{ j_0, \ldots, j_k \} \in \mathcal{N}(\mathcal{U}) \iff \bigcap_{\ell=0}^{k} U_{j_\ell} \neq \emptyset.
\]

Barycentric coordinates uniquely describe the position of points within the geometric realization of a simplicial complex. In order to relate the geometry of the data space \( \bigcup \mathcal{U} \) to that of \( |\mathcal{N}(\mathcal{U})| \), we employ the notion of a partition of unity.
Figure 3: Five colored neighborhoods overlaid with their nerve complex.

Definition 2.4. Given an open cover \( U = \{U_j\}_{j=1}^N \) of a topological space \( X \), a partition of unity (PoU) subordinate to \( U \) is a collection of continuous functions \( \phi_j : X \to [0, 1] \) such that \( \text{support}(\phi_j) \subset U_j \) for all \( j \), and \( \sum_{j=1}^N \phi_j(x) = 1 \) for all \( x \in X \).

Given a partition of unity \( \{\phi_j\}_{j=1}^N \) subordinate to \( U = \{U_j\}_{j=1}^N \) we define the nerve map

\[
\phi : \bigcup U \longrightarrow |N(U)|
\]

sending each point \( x \in \bigcup U \), to the point \( \phi(x) \in |N(U)| \) with barycentric coordinates \( \phi_1(x), \ldots, \phi_N(x) \). To make these constructions multiscale, we introduce the following notion.

Definition 2.5. Let \( X \) be a subset of a topological space \( X \). A cover tower of \( X \) is a collection \( \{U^i\}_{i \in I} \) where each \( U^i = \{U^i_j\}_{j \in J_i} \) is a family of open subsets of \( X \) with \( X \subset \bigcup U^i \), and such that each cover \( U^i \) is a refinement of all higher covers \( U^k \) with \( k > i \). In other words, if \( k > i \), then for any \( U^i_j \in U^i \), there is some \( U^k\ell \in U^k \) with \( U^i_j \subset U^k\ell \).

The relationships within a cover tower allow one to relate vertices between nerve complexes (and graphs) at different scales via the notion of a parent map.

Definition 2.6. Consider a cover tower \( \{U^i\}_{i \in I} \). A map \( p : J_{i_1} \to J_{i_2} \) with \( i_1 \leq i_2 \) is called a parent map between levels \( i_1 \) and \( i_2 \) if \( U^i_{j} \subset U^{i_2}_{p(j)} \) for any \( j \in J_{i_1} \).

The notion of parent map applies in particular to the vertices of the nerve complex. A parent map is used to define block structure for graph collapse in subsection 3.4.

2.2 Constructing Towers of Covers

Next we describe the procedures for building a series of cover sets and therefore nerve complexes and graphs, along with the edge weights and partitions of unity which are necessary to define the pertinent Laplacian operators, eigenvectors, and eigenfunctions. Precisely, we implement the cover tower (Def 2.5) via two different constructions.

Example 1 (Cover Tree Graphs). The first construction builds a cover tower via a cover tree [6, 24]. Cover trees were originally designed for efficient neighborhood query algorithms and thus enable tractable construction of large, sparse neighborhood graphs.

Definition 2.7. Consider a dataset \( X \) within a metric space \((X, d)\). A cover tree on \( X \) is a directed tree whose nodes are labeled with points in \( X \) and whose edges indicate adjacent parent-child relationships. Each point \( x \in X \) labels a string of parent-child related nodes in the cover tree. All nodes at fixed depth \( i \) determine a level set \( C_i \) of the tree. In particular, a dyadic cover tree [6] has the following properties:

- (nesting) \( C_i \subset C_{i-1} \).
- (cover) for each \( p \in C_i \) and each \( j > i \) there is a unique \( q \in C_j \) so that \( q \) is the grandparent of \( p \), satisfying \( d(p, q) < 2^{j+1} - 2^i = \sum_{k=i}^{j} 2^k \). Denote this relationship as \( q = GP^j_i(p) \).
- (separation) For distinct \( p, q \in C_j \) in the same level, \( d(p, q) > 2^j \).
Via the (cover) property, the collections $U^i = \{ B(x, R \times 2^{i+1}) : x \in C_i \}$ define a cover tower of $X$ for any ratio $R \geq 1$. The grandparent relationship explicitly states the cover refinement relationships of Def. 2.5.

In addition to the cover tower, we also require a partition of unity and edge weights. Define a radial kernel $k_h(x, y) = K(d(x, y)/h)$ with supp$(K) = [0, 1]$ so that supp$(k_h(x, \cdot)) = B(x, r)$ and fix radii $r_i = R \times 2^i$ from the cover tree construction. In practice, $K(x) = \max(1 - x, 0)$. For each scale $i$, the kernels $\{ k^i_r(x) = k_r(x, y) : x \in C_i \}$ define a partition of unity $\phi^i$ dominated by the cover $U^i$, namely

$$\phi^i_x(y) = \frac{k^i_r(x, y)}{\sum_{x \in C_i} k^i_r(x, y)}.$$

The kernel $k$ also defines edge weights $E_{kj}^i = k_{2sr_i}(x_k, x_j)$. To properly account for dataset density, each edge is reweighted according to the number of grandchildren from both indices; precisely, define

$$W^i_{kj} = |GP_j^{-1}(x_k)| |GP_j^{-1}(x_j)| k_{2sr_i}(x_k, x_j)$$

wherein domain($GP_j$) = $\bigcup_{i<j} C_i = X$ is the entire dataset.

**Example 2 (TDA Mapper Graphs).** The pipeline for mapper [45] is as follows. Consider a dataset $X$ and a filter function $f : X \to \mathbb{R}$. Next, the filter-space $[\min(f(X)), \max(f(X))]$ is split into a number of overlapping intervals $I_k$; the inverse image of each interval $f^{-1}(I_k)$ is then divided into clusters $X_i$. The collection of partial clusters act as a cover of $X$ and their nerve is the mapper graph.

The partial overlap of clusters yields edge weights equal to the intersection size $W_{ij} = |X_i \cap X_j|$. A Gaussian (or any other choice of local) density $g_i$ is fitted to each partial cluster $X_i$ and the collection of densities is renormalized to yield a partition of unity $\phi(x) = g(x)/\sum_j g_j(x)$

To obtain multiple scales, the number of intervals is changed while leaving all other parameters fixed. Dyadic splitting may yield a cover tower, but only if clusters split in a compatible fashion. Thus, the extended eigenfunctions are used to relate coarse and fine scales, averaging over each node to accomplish cascade. This approach is computationally reasonable because mapper graphs are by design and application smaller and sparser than neighborhood graphs.

The filter function, number of intervals, degree of overlap, and cluster method are all parameters of the mapper construction. Filter functions are typically geometric in nature for use in exploratory analysis, e.g., dataset eccentricity or a density estimate. The cluster method must be flexible with the number of clusters, such as thresholded hierarchical clustering. The clusters obtained from mapper emulate connected components which may have any shape, thus ideal clustering methods are robust against general geometry; such methods include single linkage, DBSCAN, and HDBSCAN [16, 8, 31]. Single linkage is included in the python mapper package [33] and is used in our implementation, though HDBSCAN may be a better choice since it is robust against variable density and noise.

## 3 Graph Laplacians and Cascading

### 3.1 Random Walk graph Laplacian

From a given similarity matrix $W_{kj}$, one can construct various graph Laplacians.

**Definition 3.1.** Let $G = (V, E)$ be a graph with edge-weight matrix $W$. The degree matrix $D$ is the diagonal matrix such that $D_{ii} = \sum_j W_{ij}$ is the degree of vertex $i$. The unnormalized graph Laplacian $L$, normalized symmetric graph Laplacian $L_{sym}$, and random walk graph Laplacian $L_{rw}$ are defined respectively as

$$L = D - W,$$

$$L_{sym} = D^{-1/2}(D - W)D^{-1/2} = I - D^{-1/2}WD^{-1/2},$$

$$L_{rw} = D^{-1}(D - W) = I - D^{-1}W.$$

The unnormalized graph Laplacian $L$ has poor spectral convergence properties [22], thus we consider the normalized symmetric and random walk graph Laplacians. We choose to implement eigenvector cascade with the random walk Laplacian due to the following properties: (i) The constant vector is an eigenvector (not true for $L_{sym}$); (ii) For data sampled from a manifold $M$ with probability $P$ and kernel-based similarities $W$, $L_{rw}$ converges to the weighted Laplacian $\Delta$ [5], whose eigenfunctions form a basis for $L^2(M, P)$ [20]; (iii) $I - L_{rw} = D^{-1}W$ is the Markov transition
kernel for the canonical random walk on the weighted graph; and (iv) $L_{rw}$ is a weighted Laplacian as described in [23], i.e. the composition of divergence and gradient operators weighted by $W$ and $D$ to provide compatible inner products on the respective dual spaces $E^*$ and $V^*$. Since $L_{sym}$ is Hermitian, the eigenvectors associated to the smallest eigenvalues can be computed efficiently by methods such as Lanczos [29] or LOBPCG [27]. However, $L_{sym}$ and $L_{rw}$ are similar matrices and therefore share the same spectrum; moreover, $v$ is an eigenvector of $L_{sym}$ if and only if $D^{-1/2}v$ is an eigenvector of $L_{rw}$ (with matching eigenvalue). Thus, we can cascade the small eigensystem of $L_{rw}$ efficiently.

3.2 Cover Tower Vector Refinement

**Definition 3.2.** Let $K$ be an abstract simplicial complex and $n \geq 0$ an integer. The space of real $n$-cochains in $K$, denoted $C^n(K)$, is the vector space of real-valued antisymmetric functions on the $n$-simplices of $K$. That is, given $\varphi \in C^n(K)$, an $n$-simplex $\{v_0, \ldots, v_n\} \in K$, and a permutation $\tau$ in the symmetric group on $\{0, \ldots, n\}$, then it is always the case that

$$\varphi(v_0, \ldots, v_n) = (-1)^{\text{sign}(\tau)} \varphi(v_{\tau(0)}, \ldots, v_{\tau(n)})$$

We say that a function $g : K \rightarrow K'$ between abstract simplicial complexes is a simplicial map, if $g(\{v_0, \ldots, v_n\}) = \{g(v_0), \ldots, g(v_n)\} \in K'$ for every $n$-simplex of $K$. It follows that a simplicial map $g : K \rightarrow K'$ induces linear maps $g^*_n : C^n(K') \rightarrow C^n(K)$, defined as

$$g^*_n(\varphi)(v_0, \ldots, v_n) = \varphi(g(v_0), \ldots, g(v_n))$$

Now, consider a cover tower $\{U^i\}$ for a data set $X$ as described in Def. 2.5. The refinement condition for $\{U^i\}_i$ implies the existence of well-defined (though not unique) functions $f_i : J_i \rightarrow J_{i+1}$, which we call parent maps (Def. 2.6), satisfying $U^i_j \subset U^{i+1}_{f_i(j)}$ for every $j \in J_i$. For each $i$ and every $\{j_0, \ldots, j_n\} \subset J_i$ we have $\emptyset \neq U^{j_0}_i \cap \cdots \cap U^{j_n}_i \subset U^{i+1}_{f_i(j_0)} \cap \cdots \cap U^{i+1}_{f_i(j_n)}$, so that $f_i$ extends to a (simplicial) map of nerve complexes:

$$f_i : N(U^i) \rightarrow N(U^{i+1}) \quad \{j_0, \ldots, j_n\} \mapsto \{f_i(j_0), \ldots, f_i(j_n)\}$$

and in particular these induce linear maps $f^*_i : C^n(N(U^{i+1})) \rightarrow C^n(N(U^i))$.

We now address the compatibility of Laplacian eigenvectors across scales. Indeed, the coboundary operator is the linear map $\delta_n : C^n(K) \rightarrow C^{n+1}(K)$ defined by the formula

$$\delta_n(\varphi)(v_0, \ldots, v_{n+1}) = \sum_{i=0}^{n+1} (-1)^i \varphi(v_0, \ldots, v_{i-1}, v_{i+1}, \ldots, v_{n+1})$$

and, as can be readily checked, it satisfies $\delta_n \circ \delta_{n+1} = 0$ for every $n \geq 0$, as well as $\delta^*_n \circ g^*_n = g^*_{n+1} \circ \delta_n$ for every simplicial map $g$. That is, linear transformations induced by simplicial maps commute with coboundary operators.

Since each $C^n(K)$ is a real vector space, it supports the definition of an inner product. From these choices, the coboundary maps have adjoints $\delta^*_n : C^{n+1}(K) \rightarrow C^n(K)$, and as is customary in the Hodge theory literature [23] one defines the Hodge Laplacian operator:

$$\Delta_n = \delta_{n-1} \circ \delta_n + \delta_n \circ \delta_{n+1}$$

In particular, when $n = 0$ we have that $\Delta_0 = \delta_1^* \circ \delta_0$ is the graph Laplacian for the 1-skeleton of $K$ (since $\delta_{-1} = 0$). Moreover, compatible choices of edge weights $W$ and vertex weights $D$ yield the random walk Laplacian of Def. 3.1 (see also [23]).

In contrast with the coboundary map $\delta_n$, the adjoints $\delta^*_n$ (and therefore the Laplacians $\Delta_n$) do not commute with the linear transformations $f^*_i$ induced by the ancestor maps. Thus, the methodologies used in persistent homology [14] for tracking multiscale topological properties, and in particular for the persistence of a self-map [15], do not apply directly to the Laplacian eigenvectors studied here.

3.3 Eigenvector Cascading

If $L_i$ denotes the matrix of the pertinent graph Laplacian for the 1-skeleton of $N(U^i)$, then the single cascade method shown in Alg. 1 uses the induced maps $f^*_i = f^*_i\mid_{U^i}$ to accelerate eigenvector determination for the collection $\{L_i\}$. Since the vertex set grows from $N(U^{i+1})$ to $N(U^i)$, only $|U^{i+1}|$ eigenvectors are determined at level $i$ for single cascade. The remaining initial state vectors can be chosen randomly followed by Gram-Schmidt; alternatively, Lemma 3.13 also provides eigenspaces $\text{Null}_A(f)$.
where the second cascade, shown in Alg. 2, comes in: Each eigenvector is cascaded down the cover tower by
As discussed in subsection 3.2, there is no set correspondence between the eigenvectors at different scales. That is
begin
Algorithm 1:
Data: Cover tower \( \{ U_i \}_{i=k}^N \), ancestor maps \( f_i \), Matrix operators \( L_i \), a method \( \text{EV}(\text{start}, \text{matrix}) \) to find
eigensystems of \( L_i \) from an initial state, a number of eigenvectors \( m \leq \# \{ U_i \} \), and an initial state \( \{ u_j^N \}_j \)
for the coarsest scale.
Result: A collection of eigenvectors \( \{ v_j^i \}_j \), eigenvalues \( \lambda_j^i \) and initial states \( \{ u_j^i \}_j \) for each scale \( i \).
begin
| \{ v_j^N \}_{j=N}^{m} , \{ \lambda_j^N \}_{j=1}^{m} | \leftarrow \text{EV} \left( \{ u_j^N \}_{j=1}^{m} , M_N \right)
for i from \( N-1 \) to 1 do
| \{ u_j^i \}_{j=1}^{m} \leftarrow f_i^* \left( \{ v_j^{i+1} \}_{j=1}^{m} \right)
| \{ v_j^i \}_{j=1}^{m} , \{ \lambda_j^i \}_{j=1}^{m} | \leftarrow \text{EV} \left( \{ u_j^i \}_{j=1}^{m} , L_i \right)
end
end

As discussed in subsection 3.2, there is no set correspondence between the eigenvectors at different scales. That is
where the second cascade, shown in Alg. 2, comes in: Each eigenvector is cascaded down the cover tower by \( f_i^* \) and projected onto the next eigenspace presented by clustered eigenvectors, yielding eigenspace basis vectors chosen with proper continuity from one scale to the next. More precisely, the eigenspaces \( \{ v_j^i \}_{j \in P_i[k]} \) and \( \{ v_j^{i+1} \}_{j \in P_i[k]} \) correspond while the new basis vectors eigenvectors, \( w_j^i \), correspond as \( i \) changes. While \( w_j^i \) are typically not eigenvectors, \( \text{Span} \left( \{ v_j^i \}_{j \in P_i[k]} \right) = \text{Span} \left( \{ w_j^i \}_{j \in P_i[k]} \right) \) and \( w_j^i \) are stable so long as the span is stable, unlike individual eigenvectors in a cluster (c.f. [26] or [10] theorem 2.1).

Similar to single cascade, second cascade can only relate a fraction of the eigenvectors in the fine graph to the eigenvectors in the coarse graph. Indeed, the input of Alg. 2 implicitly assumes that the induced maps \( f_i^* \) relate vectors spaces of dimension at least \( m \). In practice, if \( m \) eigenvectors are requested, all eigenvectors can be related by second cascade once the graph has more vertices than \( m \). In practice, the output of Alg. 1 is sufficient for Alg. 2 and are performed in tandem as double cascade for a collection of Laplacian operators \( \{ L_i \} \).

In the kernel-based construction of cover-trees (Ex. 1), the associated eigenfunctions have been shown to converge in the limit of infinite data and infinitesimal bandwidth [5]. Toward adaptive eigenvalue clustering, we group eigenvalues if they are within some fixed multiple of machine error or if their ratio is smaller than a fixed value, performed from smallest to largest eigenvalue. Some eigenvectors may be grouped erroneously (according to the limit) but not the individual basis vectors. Thus, this section primarily aims toward satisfying the basic hypothesis that

\[ \text{Span} \left( \{ v_j^i \}_{j \in P_i[k]} \right) = \text{Span} \left( \{ w_j^i \}_{j \in P_i[k]} \right) \]

and \( w_j^i \) are stable so long as the span is stable, unlike individual eigenvectors in a cluster (c.f. [26] or [10] theorem 2.1).

3.4 Multiscale Laplacian eigenvector stability

Here we aim to analyze the stability of the generalized eigenvalue problem \( (D - W)v = \lambda Dv \) associated with the random walk Laplacian. Classical results in eigenspace perturbation theory (c.f. [26] or [46] section 4.1) yield eigenspace stability which depends on eigenvalue separation; in short, the span of closely clustered or repeated eigenvalues is stable, but not the individual basis vectors. Thus, this section primarily aims toward satisfying the basic hypothesis that

\[ \left\| D_1^{-1}(D_1 - W_1) - D_2^{-1}(D_2 - W_2) \right\|_\infty \] (where \( \| M \| = \max_{i,j} |M_{i,j}| \)) is small under radius perturbation and graph collapse, and to what degree.

In order to obtain proper control, we restrict our analysis to the kernel-based scenario described in Ex. 1 wherein
\[ W_{ij} = m_i m_j k_r(x_i, x_j) \] for points \( x_i, x_j \in X \) in metric space \( (X, d) \) with local masses \( m_i \) and \( m_j \). We restrict \( k_r \) to a specific kind of radially symmetric, scalable kernel:

**Definition 3.3.** The kernel \( k_r(a, b) = K(d(a, b) / r) \) for metric \( d \) where \( K[0,1] : [0,1] \rightarrow [0,1] \) is a decreasing, Lipschitz surjection and \( K([1,\infty)) = \{0\} \) is called the cone-class kernel with base function \( K \) at resolution \( r \).

Next, we define the (debiased) random walk Laplacian in context:

**Definition 3.4.** Consider finite \( X \) in a metric space \( (X, d) \) with masses \( m : X \rightarrow \mathbb{R}^+ \) and edge weights \( k_r(a, b) \) defined by a cone-class kernel with base-function \( K \). We define the \( m_0 \)-debiased similarity matrix for \( X \) by

\[ W_r(a, b) = m(a)m(b)k_r(a, b) - \delta(a, b)m(a)m_0K(0), \]
Algorithm 2: Second Cascade

**Data:** A collection of $m$ eigenvectors $\{v_j^i\}_j$ and eigenvalues $\{\lambda_j^i\}_j$ (in increasing order) for each scale $i \in \{n, n+1, \ldots, N\}$, ancestor maps $f_i$ and small grouping thresholds $\delta$ and $\epsilon$.

**Result:** A collection of new basis vectors $\{w_j^i\}_j$ for each scale $i$, and a collection of proposed eigenspace groupings, $P_i$, which each partition $\{1, \ldots, m\}$.

```
begin
    \{w^N_j\} ← \{v^N_j\}
    for i from N−1 to n do
        P_i ← \{(1)\)
        N_i ← 1
        for j from 2 to m do
            if $|\lambda_j^i - \lambda_j^{i+1}| < \delta$ or $\lambda_j^{i+1}/\lambda_j^i - 1 < 1 + \epsilon$ then
                Append $j$ to $P_i[N_i]
            end
            else
                Append \(j\) to $P_i$
                N_i ← N_i + 1
            end
        end
        for k from 1 to $N_i$ do
            $V ← \text{span} \left( \left\{ v_j^i \right\}_{i ∈ P_i[k]} \right)$
            for $j$ in $P_i[k]$ do
                $w^i_j ← \text{Proj}_V \left( \mathcal{L}_i^*(w_j^{i+1}) \right)$
            end
        end
    end
end
```

where $\delta(a,b) = 1$ if $a = b$ and 0 otherwise, and its associated debiased degree matrix by $D_r(a,b) = \delta(a,b) \sum_{x ∈ X} W_r(a,x)$. The random walk operator is defined by the matrix $M_r = D_r^{-1}W_r$, and random walk Laplacian is $L_{rw} = I - M_r$.

**Remark 1.** When collapsing from a fine graph to a coarse graph, the number of points represented by a single vertex roughly measures sample density. Thus, if $m_0$ is chosen as the mass of one data point, then the value $m(a)m_0$ indicates the level of forced coincidence at the vertex $a$. Thus, the choice of $m_0$ in definition 3.4 reduces bias in the associated random walk Laplace operator since incidence of a point with itself should not affect the limiting random walk. In the case that $m(x) = m_0$ for all $x ∈ X$, we return to the usual notion of $W(a,a) = 0$.

We first aim to establish stability of the unnormalized and random-walk graph Laplacians with respect to $r$, then unambiguously relate the Laplacian eigenspaces at different levels in a cover tower within the general context of graph collapse; our approach is similar to that of [28].

We begin by establishing uniform stability for the kernel $k_r$ with respect to the resolution $r$:

**Lemma 3.5.** Let $(X,d)$ be a metric space and $k_r(a,b)$ cone-class kernel with base-function $K$. Then

$$|k_{r_1}(a,b) - k_{r_2}(a,b)| ≤ \text{Lip}(K) \frac{r_0 - r_1}{r_0}$$

for any $a, b ∈ X$ whenever $0 < r_1 < r_0$.

**Proof.** Case 1 ($r_1 < r_0 ≤ d(a,b)$):

$k_{r_0}(a,b) = k_{r_1}(a,b) = 0$.

Case 2 ($d(a,b) ≤ r_1 < r_0$):

$$|k_{r_0}(a,b) - k_{r_1}(a,b)| = K(d(a,b)/r_0) - K(d(a,b)/r_1) ≤ \text{Lip}(K) \frac{d(a,b)(r_0 - r_1)}{r_0 r_1} ≤ \text{Lip}(K) \frac{r_0 - r_1}{r_0}$$

Case 3 ($r_1 < d(a,b) < r_0$): Since $k_{r_0}(a,b) = 0 = K(1)$ and $K$ is decreasing,

$$|k_{r_0}(a,b) - k_{r_1}(a,b)| = K(d(a,b)/r_0) - K(1) ≤ K(r_1/r_0) - K(1) ≤ \text{Lip}(K)(1 - r_1/r_0).$$

□

9
Note that \( |1 - \min(r_1, r_0)|/\max(r_1, r_0) \) < \( \epsilon \) may be rephrased symmetrically as
\[
|\log(r_0) - \log(r_1)| < -\log(1 - \epsilon)
\]
Since the absolute values are unnecessary, for appropriately smooth base functions \( K \), the proof of Lemma 3.5 can be adjusted to prove that \( k_r(\cdot, \cdot) \) is equally smooth for \( r \in E \subset (0, \infty) \).

Lemma 3.5 immediately yields stability for the usual Laplacian \( L = D - W \) since
\[
[W_r(a, b) - W_r(a, b)] = m(a)m(b)[k_r(a, b) - k_r(a, b)]
\]
and the degree is a sum of such terms. Lastly, note that \( W(a, a) = m(a)(m(a) - m_0)K(0) \) is constant (and therefore uniformly Lipschitz) w.r.t. \( r \). Stability for the random walk Laplacian is slightly more involved. We begin by bounding below the degree of each vertex:

**Lemma 3.6.** Consider finite \( X \) in a metric space \( (\mathbb{X}, d) \) with masses \( m : X \rightarrow \mathbb{R}^+ \) and edge weights \( k_r(a, b) \) defined by a cone-class kernel with base-function \( K \). Assume \( m(x) \geq m_0 > 0 \) and define the \( m_0 \)-debiased similarity matrix \( W_r(a, b) \) and degree \( D_r(a, b) \) as in Def. 3.4. Then there is an \( R > 0 \) such that whenever \( r > R_0 > R \),
\[
D_r(a, a) \geq m(a)K(R/R_0) \geq m_0K(R/R_0) > 0
\]

**Proof.** Since \( X \subset (\mathbb{X}, d) \), there is an upper bound on the nearest neighbor distance:
\[
R = \min_{x \in X} d(x, x')
\]
In turn, if \( r \geq R_0 > R \), then
\[
\min_{x \in X} \max_{x' \neq X} k_r(x, x') \geq \frac{K(R/R_0)}{2}
\]
yielding the degree bound
\[
D_r(a, a) = \sum_{b \in X} m(a)m(b)k_r(a, b) - m_0m(a)K(0) \geq \sum_{b \neq a} m(a)m(b)k_r(a, b) \geq m(a)K(R/R_0) > 0.
\]

In practice the minimal degree is larger, but the proposition presents the bottleneck at which a discontinuity may occur. Finally, we prove stability for \( D^{-1}W \) and hence \( L_{rw} = I - D^{-1}W \) with respect to the resolution \( r \).

**Lemma 3.7.** Consider finite \( X \) in a metric space \( (\mathbb{X}, d) \) with masses \( m : X \rightarrow \mathbb{R}^+ \) and edge weights \( k_r(a, b) \) defined by a cone-class kernel with base-function \( K \). Assume \( m(x) \geq m_0 > 0 \) and define the \( m_0 \)-debiased similarity matrix \( W_r(a, b) \) and degree \( D_r(a, b) \) as in Def. 3.4. Further assume that \( \sum_{x \in X} m(x) = 1 \) describe probability masses. Let \( R, R_0 \) be the minimal radii from lemma 3.6. Then
\[
\|D^{-1}W_{r_0} - D^{-1}W_{r_1}\|_\infty \leq 2\left[\text{Lip}(K)/K(R/R_0)\right](r_0 - r_1)/r_0
\]
whenever \( r_0 > r_1 \geq R_0 \).

**Proof.** Suppose \( a \neq b \in X \). Cancel \( m(a) \) to obtain:
\[
[D^{-1}W_{r_0}(a, b) - D^{-1}W_{r_1}(a, b)](a, b) = \left[\sum_{x \in X} \frac{m(x)k_{r_0}(a, x) - m_0K(0)}{m(x)k_{r_1}(a, x) - m_0K(0)}\right] - \left[\sum_{x \in X} \frac{m(x)k_{r_1}(a, x) - m_0K(0)}{m(x)k_{r_0}(a, x) - m_0K(0)}\right] = \frac{A}{B}
\]
Abbreviate the denominator as \( B = [m(a)^{-1}D_{r_0}(a, a)] [m(a)^{-1}D_{r_1}(a, a)] \). Adding and subtracting the mixed term \( [m(x)k_{r_1}(a, b)k_{r_1}(a, x)] \) and rearranging expresses \( A \) as two major terms as follows
\[
A = m(b) \left[\sum_{x \in X} m(x)k_{r_0}(a, b)k_{r_1}(a, x) - m_0K(0)k_{r_0}(a, b)\right] - \left[\sum_{x \in X} m(x)k_{r_1}(a, b)k_{r_0}(a, x) - m_0K(0)k_{r_1}(a, b)\right] \leq m(b) \left[k_{r_0}(a, b) - k_{r_1}(a, b)\right] \left[\sum_{x \in X} m(x)k_{r_1}(a, x) - m_0K(0)\right] + m(b)k_{r_1}(a, b) \left[\sum_{x \in X} m(x)k_{r_1}(a, x) - k_{r_0}(a, x)\right].
\]
Lemma 3.5 and \( m(b)k_{r_1}(a,b) \leq [m(a)^{-1}D_{r_1}(a,a)] \) together yield:

\[
A \leq \left( 1 + \sum_{x \in X} m(x) \right) m(b)\text{Lip}(K) \left[ m(a)^{-1}D_{r_1}(a,a) \right] \left[ (r_0 - r_1)/r_0 \right]
\]

Altogether, canceling \( [m(a)^{-1}D_{r_1}(a,a)] \) with the matching factor in \( B \), bounding \( m(b) \) above by the maximum, and bounding the other factor \([m(a)^{-1}D_{r_0}(a,a)]\) in \( B \) below by \( K(R/R_0) \) finishes the proof.

Toward describing stability between cascade levels, we will work with weighted graphs.

**Definition 3.8.** A weighted graph \( G = (V,E,f,g) \) consists of vertices \( V \), edges \( E \subseteq V \times V \), vertex masses \( f : V \rightarrow \mathbb{R}^+ \), and edge function \( g : E \rightarrow \mathbb{R} \). We may also consider \( g : V \times V \rightarrow \mathbb{R} \) symmetric with \( g(a,b) = 0 \) whenever \((a,b) \notin E\).

We also utilize mass-aware extended edge weights \( h(a,b) = f(a)f(b)g(a,b) \), and define the degree

\[
d_j(a) = \sum_{b \in V} j(a,b)
\]

for any edge function \( j \). Borrowing from [28], we call a weighted graph an all-positive (AP) graph if \( g(a,b) > 0 \) whenever \((a,b) \in E\). Next, we introduce \( \epsilon \)-block structures and \( \epsilon \)-collapse to model the parent-child relationships within a cover tower.

**Definition 3.9.** Consider a weighted graph \( G = (V,E,f,g) \) and a partition of its vertices \( U = \{U_i\}_{i=1}^n \). The partition \( U \) is said to be an \( \epsilon \)-block structure for \( G \) if \((a,b) \in U_i \) implies \( |g(a,v) - g(b,v)| \leq \epsilon \) for all \( v \in V \).

**Definition 3.10.** Given an AP graph \( G = (V,E,f,g) \) with \( \epsilon \)-block structure \( U \), define the collapse vertex weight by \( f_U(A) = \sum_{a \in A} f(a) \) for any block \( A \in U \). Define a representative \( u_i \in U_i \) for each block \( U_i \in U \) and associated collapsed edge weight \( g_U(U_i,U_j) = g(u_i,u_j) \) so that \( |g_U(A,B) - g(a,b)| \leq 2\epsilon \) for all \( a \in A \in U \) and \( b \in B \in U \). Finally, define \( E_U = \{(A,B) \in U \times U : g_U(A,B) > 0\} \). We call the AP graph \( G_U = (U,E_U,f_U,g_U) \) the \( 2\epsilon \)-collapse of \( G \) along \( U \) via representatives \( \{u_i\} \).

**Remark 2.** Given an \( \epsilon \)-collapse \( G_U \) of \( G \), the re-expansion \( G' = (V',E',f',g') \) with \( g'(a,b) = g_U(A,B) \) for \( a \in A \), \( b \in B \) automatically satisfies \( |g'(a,b) - g(a,b)| \leq \epsilon \). Despite describing linear operators on different spaces, collapse and re-expansion allows us to compare \( g \) and \( g_U \). The re-expansion only depends on \( G_U \), \( f \), and the block structure \( U \). Thus, any other edge (or vertex) function \( \psi \) on \( G_U \) can be expanded to a \( U \) block-constant function \( \psi_U \) on \( G \) by replicating its outputs.

**Kernel-based cover towers provide the pertinent example of \( \epsilon \)-collapse.**

**Proposition 3.11.** Consider finite sets \( X' \subseteq X \) in a metric space \((X,d)\) such that \( X \subseteq \bigcup_{x \in X'} B(x,h) \) for some \( h > 0 \) with masses \( m : X \rightarrow \mathbb{R}^+ \) and cone-class kernel \( k_r(x,y) \) with base-function \( K \) at resolution \( r > 0 \). Then the collections

\[
U_A = \{B(a,r) : a \in A\}
\]

where \((A,r) = (X,r_1) \) or \((X',r_0)\) with \( r_0 \geq r_1 + h > 0 \) define a cover tower and parent function \( p : X \rightarrow X' \) (see Def. 2.6 and Ex. 1) with \( d(p(x),x) \leq h \) for all \( x \in X \) and \( p(x) = x \) for \( x \in X' \). Define the block structure \( U = \{U_x = p^{-1}\{x\} : x \in X'\} \). Then \( U \) is a \([2\text{Lip}(K)h/r]-\)block structure for \((X,E,m,k_r)\), where

\[
E = \{(x,y) \in X^2 : d(x,y) < r\} = \{(x,y) \in X^2 : k_r(x,y) > 0\}
\]

**Proof.** Fix \( x \in X' \) and let \( a,b \in U_x, v \in V \). Then \( x = p(a) = p(b) \) and

\[
|d(a,v) - d(b,v)| \leq d(a,b) \leq d(a,x) + d(b,x) \leq 2h
\]

and therefore

\[
|k_r(a,v) - k_r(b,v)| \leq 2\text{Lip}(K)h/r
\]

This kind of stability can be improved for the cover tree by (i) using a smaller factor than 2 between cover tree radii or (ii) increasing the size of the set relative to the cover tree radius (in our examples, \( \text{Lip}(K) = 1 \) and \( h/r = 1/3 \)). It remains to compare the collapsed graph to its re-expansion. Indeed, the original block structure \( U \) is a 0-block structure for the re-expansion \( G' \) and has (unique) 0-collapse \( G_U \) (see Rmk. 2). We hereafter refer to such graphs as having perfect block structure \( U \) and to \( G_U \) as the perfect collapse of \( G' \).
\textbf{Proposition 3.12.} Any weighted graph \( G = (V, E, f, g) \) with perfect block structure \( U \) has a unique perfect collapse \( G_U \) satisfying the following properties:

1. \( g_U(A, B) = g(a, b) \) for any \( a \in A \) \( U \) and \( b \in B \) \( U \);
2. \( h_U(A, B) = \sum_{a \in A, b \in B} h(a, b) \) for any \( A, B \) \( U \);
3. \( d_{g_Uf_U}(A) = \langle d_{g}f \rangle_{U}(A) \) for any \( A \) \( U \), and \( d_{g}f \) is block constant.

\textit{Proof.} Properties (1) and (2) follow trivially from the definition of a 0-collapse. Uniqueness of the collapse follows immediately from property (1). Property (3) is equivalent to \( d_{g_Uf_U}(A) = d_{g}f(a) \) whenever \( a \in A \) \( U \); indeed, if \( a \in \text{Ain} U \):

\[
d_{(g\text{f})_{U}}(A) = \sum_{B \in U} g_U(A, B)f_U(B) = \sum_{B \in U} \sum_{b \in B} g(a, b)f(b) = \sum_{b \in V} g(a, b)f(b) = d_{g}f(a).
\]

Property (3) of Prop 3.12 is crucial to establish that the collapse of a degree matrix is the degree matrix of the collapsed similarity matrix (ignoring \( m_0 \)-debiasing, which is handled separately). For the following, any vertex weight or edge weight induces a linear operator on \( V^* \) (the linear dual of \( V \)). In particular, for \( v \in V^* \), any edge weight \( g \) induces a linear operator via \([g^*(v)](a) = \sum_{b \in V} g(a, b)v(b)\) and any vertex weights induce a linear operator via \([f^*(v)](a) = f(a)v(a)\). Thus, we are concerned with eigenvectors in this vector space under the usual inner-product \( \langle v, w \rangle = \sum_{a \in V} v(a)w(a) \), so that the dual vertices form an orthonormal basis of \( V^* \). The concrete relationship between \( g \) and \( g_U \) in a perfect collapse extends to an explicit relationship between Laplacian eigenspaces in \( G \) and \( G_U \).

\textbf{Lemma 3.13.} Consider a weighted graph \( G = (V, E, f, g) \) and perfect collapse \( G_U = (U, E_U, f_U, g_U) \) along block structure \( U \), as well as extended edge weights \( h \) and \( h_U \). Fix a constant \( C \) \( U \) block-constant \( j : V \to \mathbb{R} \) and its collapse \( j_U(A) = j(a) \) for any \( A \) \( U \). Then,

1. The generalized eigenvectors of \( [h^* - Cf^*]v = \lambda[j^*f^*]v \) with \( \lambda \neq -C/j(a) \) for any \( a \in V \) are block-constant along \( U \).
2. The \( U \) expansion of each eigenvector of \( [h^*_U - Cf^*_U]v = \lambda[j^*_Uf^*_U]v, \ v_U, \) is an eigenvector of \( [h^* - Cf^*]v_U = \lambda[j^*f^*]v_U \) with identical eigenvalue.
3. The generalized eigenvalue problem \( [h^* - Cf^*]v = \lambda[j^*f^*]v \) decomposes into \( U \) block-constant eigenvectors from (2) and block-local eigenvectors in \( \text{Null}_U(f) = \{v \in V^* : \sum_{b \in B} f(b)v(b) = 0 \text{ for every } B \in \mathcal{U}\} \).

\textit{Proof.} (1) Let \( v \) solve \( [h^* - Cf^*]v = \lambda[j^*f^*]v \) with \( \lambda \neq -C/j(a) \) for any \( a \in V \). Fix \( a \in A \) \( U \); then,

\[
\lambda j_U(A)f(a)v(a) = \lambda j(a)f(a)v(a) = [\lambda j^*f^*](a) = [(h^* - Cf^*)v](a) = \sum_{b \in V} h(a, b)v(b) - Cf(a)v(a)
\]

\[
= \sum_{b \in V} f(a)f(b)g(a, b)v(b) - Cf(a)v(a) = f(a) \left( \sum_{B \in U} \sum_{b \in B} f(b)g_U(A, B) - Cv(a) \right).
\]

Solving shows that \( v(a) = (\sum_{B \in U} f_U(B)g_U(A, B)) / (C + \lambda j_U(A)) \) depends only on the block \( A \).

(2) Suppose \( v \) solves the generalized eigenvalue problem \( [h^*_U - Cf^*_U]v = \lambda[j^*_Uf^*_U]v \). Fixing a block \( A \in U \) and dividing both sides by \( f_U(A) \) yields

\[
\sum_{B \in U} f_U(B)g_U(A, B)v(B) = (\lambda j_U(A) + C)v(A).
\]
for all $A \in U$. Define $v^U \in V^*$ as the $U$-expansion of $v$, i.e., $v^U(a) = v(A)$ whenever $a \in A$. Collapsing and expanding and via Eq. (5) we have for $a \in A \in U$:

$$[h^* v^U](a) = \sum_{b \in V} h(a, b) v^U(b) = \sum_{B \in U} \sum_{b \in B} f(a) f(b) g_U(A, B) v(B) = f(a) \sum_{B \in U} g_U(A, B) f_U(B) v(B) = f(a) (\lambda_j v(A) + \lambda_j v(a)) = (\lambda_j + C) f^* v^U(a).$$

(3) Consider the spaces

$$\text{Null}_A(f) = \left\{ v \in V^* : \sum_{a \in A} f(a) v(a) = 0 \right\}$$

whose span are transverse to the $U$-block constant vectors of (2). Fix $A \in U$ and $v \in \text{Null}_A(f)$. Then, for any $a \in V$, $[h^* v](a) = f^*[g^* f^* v](a) = 0$ so that $\lambda_j f^* v(a) = (\lambda_j + C) f^* v(a)$. If $a \notin A$, then $-C f(a) v(a) = 0 = (-C/jv(A)) j(a) f(a) v(a)$; otherwise $a \in A$ so that $j(a) = jv(A)$ and $-C f(a) v(a) = (-C/jv(A)) j(a) f(a) v(a)$. Thus, $v$ is an eigenvector with eigenvalue $\lambda = -C/jv(A)$. Notably, if $C = 0$, then these are all trivial eigenvalues. Lastly, since $\text{dim}(\text{Null}_A(f)) = |A| - 1$, the dimension of the span is $|V| - |U|$ while the dimension of $U$-block constant vectors is $|U|$. Since these subspaces are transverse and have full dimension, they decompose $V^*$.

By taking $f(a) = m(a), g(a, b) = k_r(a, b)$, and $C = m_0 K(0)$, the debiased similarities $W_r$ of Def. 3.4 yield the linear operator $h - C f^*$. Moreover, Prop 3.12 (3) shows that the associated degree $D_r$ is associated to $f^* f^*$ for block-constant $j$. Thus, Lemma 3.13 applies in particular to any debiased random walk matrix $M_r = D_r^{-1} W_r$ and therefore also the random-walk Laplacian with the transform $\mu = 1 - \lambda$ for the eigenvalues. Since it is a decomposition by eigenspaces, the decomposition in part (3) yields an orthogonal decomposition for the eigenspaces of the (symmetric) normalized graph Laplacian.

**Remark 3.** A similar result holds for the eigenvalue problem $[j^* - g^* f^*] v = \lambda v$ with $A, j_U(A)$, which in particular applies to unnormalized weighted graph Laplacians of the form $D - W$ where $W_{ij} = k_r(x_i, x_j) f(x_i)$, so that $g = k_r$ and $D$ is the associated degree $d_q$. The identities of prop 3.12 establish that the collapsed Laplacian operator is in particular $f$ for block-constant $j$. Thus, Lemma 3.13 applies in particular to any debiased random walk matrix $M_r = D_r^{-1} W_r$ and therefore also the random-walk Laplacian with the transform $\mu = 1 - \lambda$ for the eigenvalues. Since it is a decomposition by eigenspaces, the decomposition in part (3) yields an orthogonal decomposition for the eigenspaces of the (symmetric) normalized graph Laplacian.

In practice, we change both the block structure and the kernel resolution as we cascade eigenvectors. Combining lemma 3.7, prop. 3.11, and lemma 3.13 in context we obtain the following Theorem, which in particular may yield useful stability for the projections in Alg. 2.

**Theorem 1.** Consider finite sets $X' \subset X$ in a metric space $(X, d)$ such that $X \subset \bigcup_{x \in X'} B(x, h)$ with $h < r_1 < r_0$ with masses $m : X \to \mathbb{R}^+$ and cone-like kernel $k_r(x, y)$ with base-function $K$ at resolutions $r = r_0$ or $r_1$. Then the collections $\mathcal{I}_A = \{B(a, r) : a \in A\}$ for $A, r = (X, r_1)$ or $(X', r_0)$ define a cover tower with parent function $p : X \to X'$ (Def. 2.6 and Ex. 1) such that $d(p(x), x) \leq h$ for all $x \in X$ and $p(x) = x$ for $x \in X'$, yielding block structure $U = \{U_x = p^{-1}(\{x\}) : x \in X'\}$. Further assume that $\sum_{x \in X} m(x) = 1$ describe probability masses and $m(x) \geq m_0 > 0$.

Define the $m_0$-debiased similarity matrices $W_r(a, b)$ and degrees $D_r(a, b)$ as in Def. 3.4 and expanded (block-constant) versions $W'_r$ and $D'_r$. Let $R, R_0$ be the minimal radii from lemma 3.6. Then,

$$\left\| (D'_r)^{-1} W'_r - D_{r_1}^{-1} W_{r_1} \right\|_\infty \leq 2 \frac{\text{Lip}(K)}{K(R/R_0)} \left[ \frac{r_0 - r_1}{r_0} + \frac{h}{r_0} \right].$$

**Remark 4.** The control in Theorem 1 relies upon the ratios $(r_0 - r_1)/r_0$ and $h/r_0$. The formal ratio results from changing resolution and can only be reduced by changing the resolution in smaller steps. To reduce the latter ratio $h/r_0$, we consider cover sets that are larger than necessary; i.e., if $X \subset \bigcup_{x \in X'} B(x, h)$, consider cover sets $\{B(x, th) : x \in X'\}$ for some fixed ratio $t$ to build the entries $W_{ij} = k_{(th)}(x_i, x_j)$. While both choices increase stability, they also require more computation by respectively yielding taller cover towers and denser graphs. This kind of computational trade-off is unavoidable; for example in [48], convergence of $L_{rw}$ (in the limit of infinite data) relies on growing neighborhood cardinality despite a shrinking kernel bandwidth (i.e., cover set radius).

The result of Thm. 1 can be paired with eigenspace stability results for tightly clustered eigenvalues, such as Theorem 2.1 from [10] shown below:
Lemma 3.14. Consider linear operators $L_0, L_1 : \mathbb{R}^n \to \mathbb{R}^n$. Assume the eigenvalues $\{\lambda_i\}$ of $L_0$ can be partitioned so that $\Lambda_j = \{\lambda_k \}_{k=1}^{i_j+1} \subset [\xi_j - \epsilon, \xi_j + \epsilon]$ and $|\xi_j - \xi_k| \geq 2\epsilon + \delta$ for some $\epsilon, \delta > 0$. Define $P_j$ be the spectral projector onto the span of eigenspaces associated to $\Lambda_j$. If $\|L_0 - L_1\|_\infty \leq \gamma < \delta/2$, then $\Lambda_j' \subset [\xi_j - (\epsilon + \gamma), \xi_j + (\epsilon + \gamma)]$ partitions the eigenvalues of $L_1$ with matching cardinalities ($|\Lambda_j'| = |\Lambda_j|$) and we may define $P_j'$ as the corresponding spectral projector $\Lambda_j'$. Moreover,

$$|(I - P_j')P_j| \leq (\epsilon + \gamma)/(\epsilon + \delta - \gamma)$$ (6)

for each $j$.

In particular, pairing Lemma 3.14 with the Laplacians $L_0 = I - (D_{r_0}')^{-1}W_{r_0}'$ and $L_1 = I - D_{r_1}^{-1}W_{r_1}$ from 1 yields the following:

Corollary 1. Under the hypotheses of Thm. 1, define $L_0 = I - (D_{r_0}')^{-1}W_{r_0}'$ and $L_1 = I - D_{r_1}^{-1}W_{r_1}$. Let $\Lambda_j$ partition the eigenvalues of $L_0$ with $\epsilon = \max_j (\text{diam}(\Lambda_j))/2$. Let $P_j$ be the spectral projection associated to $\Lambda_j$. If the eigenvalues of $L_1$ cluster so that $d(\Lambda_j, \Lambda_k) \geq \delta > 2 |\text{Lip}(K)(r_0 - r_1)/r_0 + \text{Lip}(K)h/r_0| / K(R/R_0) = 2\gamma$, then there is a partition $\Lambda_j'$ for the eigenvalues of $L_1$ and nearby spectral projectors $P_j'$ with

$$|(I - P_j')P_j| \leq (\epsilon + \gamma)/(\epsilon + \delta - \gamma)$$ (7)

for each $j$.

The spectral projectors $P_j'$ in Cor 1 are the same projections as Proj$_\psi$ used in Alg. 2. Since $P_j$ is a projection, the results of Cor. 1 show that $P_j'$ is nearly the identity on the subspace $\text{Im}(P_j)$. In particular, by Lemma 3.13, if $w^{i+1}_j$ span the corresponding eigenspace, then $f^*(w^{i+1}_j)$ of Alg. 2 lie in $\text{Im}(P_j)$, and so their distortion via $P_j' = \text{Proj}_\psi$ is controlled.

It is critical that the hypotheses of Lemma 3.14 will not hold in general. In particular, As the resolution progresses, eigenvalue groups may cross; indeed, this may occur by only changing the kernel radius $r$. Since the eigenvector features may change drastically, such crossing values describe changes in regime for how the data set geometry is represented. Analogous results also hold for the unnormalized and symmetric normalized graph Laplacians of Def. 3.1.

Besides the general stability presented here which applies to a fixed data set, the work [5] shows convergence of a kernel-based random-walk graph Laplacian in the limit of infinite data and infinitesimal bandwidth. In parallel with this convergence, in practice we often see stability of the Laplacian eigenvectors across a very broad range of scales.

4 Experimental Results

4.1 Speeding Up Laplacian Eigenvector Computations with Cascading

Example 3. Here we examine the computational benefit of eigenvector cascade (Alg. 1). We do not expect to establish state-of-the-art results in this subsection, but rather hope to re-affirm results such as [49] and [28] in a different context where the graphs and Laplacian operators are collapsed in a different fashion. We compare the run-time of LOBPCG via paired trials on several generated datasets both with and without cascading initial states. The trial results are given in Table 1. Cascading always yields a noticeable speedup at each scale and in most cases total cascade time undercuts naïve LOBPCG performed only at the finest cover level.

| dataset     | single cascade (seconds) | lobpcg final (sec) | lobpcg full (sec) |
|-------------|--------------------------|--------------------|-------------------|
| PIN_1       | 28.84                    | 55.36              | 86.55             |
| PIN_2       | 30.87                    | 67.49              | 74.5              |
| CANTOR      | 3.53                     | 4.59               | 12.9              |
| CARPET      | 117.06                   | 125.34             | 152.66            |
| SPHERE      | 40.02                    | 37.19              | 44.48             |
| BOXTREE     | 60.89                    | 56.08              | 68.03             |
| EDGES       | 305.09                   | 346.58             | 419.07            |

Table 1: Time in seconds to find 100 (smallest) Laplacian eigenvectors for generated datasets at multiple scales. The cover tree construction of Ex. 1 is applied to each dataset. Shortest times are shown in bold. Each dataset has roughly 15,000 points and intrinsic dimension 2, except EDGES contains 100,000 points and CANTOR and CARPET sample fractals with respective Hausdorff dimensions $\log_3(4)$ and $\log_3(8)$. 
Table 1 suggests that speedup ratio depends on eigenfunction convergence rate. Indeed, slow convergence is expected for the SPHERE dataset; since it consists of 6 tangent annuli, discrete approximations overestimate the degree of connection between the annuli, which is negligible (only tangent) in the limit.

In contrast, the CANTOR dataset attains the largest speedup. As a sample from the Cantor square, many connected components associate to $\text{Null}(L_{rw})$; thus, convergence is attained immediately once 100 connected components are resolved. The resulting speedup is recorded in Table 2.

| depth | non-cascade (sec) | single cascade (sec) |
|-------|-------------------|----------------------|
| 8     | 0.42              | 0.17                 |
| 9     | 0.79              | 0.20                 |
| 10    | 4.44              | 2.33                 |
| 11    | 2.66              | 0.60                 |
| 12    | 4.59              | 0.24                 |

Table 2: Time in seconds to perform LOBPCG with and without cascading initial states for the CANTOR dataset at various depths in the cover tree. There is a large drop in cascade time after depth 10, for which $\dim(\text{Null})(L_{rw}) \geq 100$.

### 4.2 Double Cascade of Laplacian Eigenspaces

Here we aim to utilize double cascade (via Algs. 1 and 2 in tandem) to obtain consistent eigenspace basis representation in practice. Due to spectral convergence [5], we expect repeated eigenvalues to cluster at fine enough scale. The following example datasets are given by a grid of points in order to enforce their symmetry and therefore accelerate convergence of repeated Laplacian eigenvalues.

**Example 4.** The PIN_1 dataset consists of points evenly dispersed within three triangles arranged to be invariant under 120 degree rotation. Since PIN_1 is connected, the first (null) eigenvector is ignored. Heat maps of the eigenfunctions derived from single and double cascade are shown in Fig. 4. Comparing these cases reveals two benefits to double cascade. First, the basis vectors converge alongside the eigenspaces with double cascade, and may not without it. Second, basis eigenvectors originate at coarse scale and therefore reflect coarse features in the dataset; in this case, for each pair, one basis eigenvector is constantly zero on one lobe of the pinwheel at all scales.

**Example 5.** The SPHERE dataset consists of six tangent geodesic annuli on a 2-sphere, and exhibits cubical symmetries. Since the dataspace is connected at any positive resolution, the first (null) eigenvector is ignored. Heat maps of the eigenfunctions derived from single and double cascade are shown in Fig. 5; in this example it is difficult to view eigenspace convergence without double cascade. With eigenspace projection, the chosen bases clearly converge and are dataspace-invariant rotations of each other.

**Example 6.** The CANTOR dataset consists of the seventh iteration of the Cantor square fractal. With a finite approximation, the Cantor square emulates a dataset with many connected components and thus all Laplacian eigenvalues approach zero, albeit at different rates. Heat maps of the eigenfunctions derived from single and double cascade are shown in Fig. 6. Double cascade orders the eigenvector bases by convergence rate; consequently, connected components are separated according to their distances similarly to single-linkage agglomerative clustering.
4.3 Spectral Persistence in Multiscale Mapper

Here we investigate the progression of Laplacian eigenfunctions of mapper graphs at ramping scale via double cascade for each of three real-world datasets. The graphs in question here are not very large, so that computation time of the Laplacian eigenvectors is negligible; rather, we are most interested in establishing correspondences between the eigenvectors at each resolution to analyze the geometry in a multi-scale fashion. Similar to Čech persistent homology [14], we expect that the true eigenspaces of the Reeb graph will be represented for a wide range of scales with unrelated features dominating extreme scales. Indeed, with either one interval or far too many, the mapper graph is wholly disconnected.

Mapper clusters with one point were removed to avoid spurious connected components in the graph and thus simplify presentation. Such singletons generally represent noise or the cut off tip of a flare. The following datasets have high intrinsic dimension and nonlinear shape; consequently, instead of comparing approximate eigenfunctions, the eigenvectors defined on the mapper graphs’ vertices will be compared directly via Fruchterman Reingold force-directed layout [18] in 2D.

Example 7. Consider the diabetes dataset from [38] which consists of diagnostic clinical measurements from patients with chemical or overt diabetes. This dataset was investigated in the initial mapper article [45]. As in the original analysis, eccentricity is chosen as the filter function. The resulting mapper graphs with eigenvector heat maps are plotted in Fig. 7. Sequential eigenfunctions show a great deal of similarity despite the changing underlying graphs; moreover, each eigenvector concentrates within a flare in the mapper graph indicating a region of potential importance.

Example 8. Next consider Fisher’s iris dataset [17, 1]. The filter function is a Gaussian density estimate, wherein the bandwidth is the average distance to the 10th nearest neighbor. The resulting mapper graphs with eigenvector heat maps are plotted in Fig. 8. The identification of eigenvectors is generally evident, helping to relate the various mapper
Figure 7: Eigenvectors for the mapper graphs obtained from the diabetes dataset [38]. Each row indicates an eigenvector index, while each column represents a particular scale (number of intervals).

graph embeddings. At the finest resolution (30 intervals), the mapper graph breaks into multiple components and the eigenvectors drift more rapidly.

Figure 8: Eigenvectors for the mapper graphs obtained from the iris dataset [17, 1]. Each row indicates an eigenvector index, while each column represents a particular scale (number of intervals).

Example 9. Consider the edges-in-images dataset of high-contrast 3x3 grayscale natural image patches, sampled randomly from the Van Hateren natural image dataset [21]. A related dataset is investigated in [9]; the same procedure is used to preprocess both datasets, but ours consists of just 5,000 patches instead of $4 \times 10^6$. The filter function is a Gaussian density estimate, wherein the bandwidth is the average distance to the $10^{th}$ nearest neighbor. The resulting mapper graphs and their eigenvectors are plotted in Fig. 9. The four flares represent connected components of the low density region, and support the 3-circle model presented in [9]. Though the graph layout permutes these 4 flares, the eigenfunctions relate very stably from 8 intervals to 14 intervals, evidenced by comparing the associated eigenfunctions. A different regime occurs at 16 and 18 intervals, with the last three eigenvectors shifting to some degree. Thus, double cascaded eigenvectors reliably track mapper features in this case. The eigenvector arrangements also imply the presence of some symmetry in the dataset.

5 Discussion and Conclusion

In this work we investigate the Laplacian eigenvectors obtained from viewing a dataset via a set of graphs built at multiple scales. A cover tower framework is proposed to construct and relate the graphs. Two methods were utilized to view a dataset as a graph: cover trees and mapper. With the cover tree construction, we utilized initial states obtained from coarser scales to aid in eigenvector determination at finer scales. First, jump-starting LOBPCG calculation of
the eigenvectors accelerates calculation. Second, by using eigenvalue clusters to emulate non-simple eigenspaces, we projected coarse-graph initial state vectors onto finer-graph eigenspaces. This second cascade process lends stability to the basis vectors and in some cases also leads to better features.

A framework of weighted graphs and graph collapse is presented. Theoretical results are given describing stability for linear operators on weighted graphs which are then related to kernel-based weighted graph Laplacian operators. While these results have a very general scope, the successful application of double cascade suggests that further stability can be obtained by utilizing the special case of samples from an underlying manifold. The notion of graph collapse is quite naturally linked to the notion of a simplicial map, inviting the more general notion of (weighted) simplicial collapse. In turn, simplicial collapse may be useful for cascading eigenvectors of weighted $n$-Laplacians which are defined in [23] (see also subsection 3.2).

Eigenvector cascade was also utilized to investigate spectral persistence in mapper graphs. Overall, our results show that the prominent mapper Laplacian eigenvectors reflect the geometry of the limiting Reeb graph. This story parallels that of persistent homology; specifically, important features of the dataset persist over a wide range of scales, while less important features are transient.

Our investigation leads us to consider a new notion of ‘persistent Laplacian eigenspaces,’ which is a continuous, geometric analogue to persistent homology. While persistent homology typically focuses on Betti numbers as opposed to homology classes, spectral persistence must focus on the eigenfunctions themselves rather than simply their presence or absence. Indeed, spaces generally have few homological cycles while the number of Laplacian eigenfunctions is typically countable. Thus, distinguishing between important and unimportant spectral features must involve both persistence and eigenvalue. Of particular note, when an eigenvalue associated to the $n$-Laplacian shrinks to 0, the eigenvector becomes a representative cocycle for a new homological generator; thus, persistent eigenvectors can be used to trace such cohomological features beyond their usual [birth,death] scope.

Moreover, while homological generators appear and die at fixed boundaries with clear identification between scales, eigenfunctions change continuously and may collide to become indistinguishable in a shared eigenspace. Simplicial maps can be used in both types of persistence, but such maps do not respect the eigenspaces in the same fashion as they do homological generators. Eigenfunction cascade can be seen as an attempt to do this approximately. Developing the concept of eigenvector persistence requires further analysis of eigenvector cascade, and in particular about the matching of cascaded guesses to determined eigenspaces.

**Conflict of Interest Statement**

On behalf of all authors, the corresponding author states that there is no conflict of interest.

**References**

[1] Anderson, E.: The irises of the gaspe peninsula. Bulletin of the American Iris society 59, 2–5 (1935)
[2] Argentati, M.E., Knyazev, A.V., Neymeyr, K., Ovtchinnikov, E.E., Zhou, M.: Convergence theory for preconditioned eigenvalue solvers in a nutshell. Foundations of Computational Mathematics 17(3), 713–727 (2017)

[3] Belkin, M., Niyogi, P.: Laplacian eigenmaps for dimensionality reduction and data representation. Neural computation 15(6), 1373–1396 (2003)

[4] Belkin, M., Niyogi, P.: Semi-supervised learning on riemannian manifolds. Machine learning 56(1-3), 209–239 (2004)

[5] Belkin, M., Niyogi, P.: Convergence of laplacian eigenmaps. In: Advances in Neural Information Processing Systems, pp. 129–136 (2007)

[6] Beygelzimer, A., Kakade, S., Langford, J.: Cover trees for nearest neighbor. In: Proceedings of the 23rd international conference on Machine learning, pp. 97–104. ACM (2006)

[7] Bronstein, A.M., Bronstein, M.M., Kimmel, R., Mahmoudi, M., Sapiro, G.: A gromov-hausdorff framework with diffusion geometry for topologically-robust non-rigid shape matching. International Journal of Computer Vision 89(2-3), 266–286 (2010)

[8] Campello, R.J., Moulavi, D., Sander, J.: Density-based clustering based on hierarchical density estimates. In: Pacific-Asia conference on knowledge discovery and data mining, pp. 160–172. Springer (2013)

[9] Carlsson, G., Ishkhanov, T., De Silva, V., Zomorodian, A.: On the local behavior of spaces of natural images. International journal of computer vision 76(1), 1–12 (2008)

[10] Davis, C.: The rotation of eigenvectors by a perturbation. Journal of Mathematical Analysis and Applications 6(2), 159–173 (1963)

[11] De Silva, V., Tenenbaum, J.B.: Sparse multidimensional scaling using landmark points. Tech. rep., Technical report, Stanford University (2004)

[12] De Witt, T., Lessig, C., Fiume, E.: Fluid simulation using laplacian eigenfunctions. ACM Transactions on Graphics (TOG) 31(1), 10 (2012)

[13] Dey, T.K., Mémoli, F., Wang, Y.: Multiscale mapper: Topological summarization via codomain covers. In: Proceedings of the twenty-seventh annual acm-siam symposium on discrete algorithms, pp. 997–1013. SIAM (2016)

[14] Edelsbrunner, H., Harer, J.: Persistent homology—a survey. Contemporary mathematics 453, 257–282 (2008)

[15] Edelsbrunner, H., Jabłoński, G., Mrozek, M.: The persistent homology of a self-map. Foundations of Computational Mathematics 15(5), 1213–1244 (2015)

[16] Ester, M., Kriegel, H.P., Sander, J., Xu, X., et al.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: KDD, vol. 96, pp. 226–231 (1996)

[17] Fisher, R., Marshall, M.: Iris data set. RA Fisher, UC Irvine Machine Learning Repository 440 (1936)

[18] Fruchterman, T.M., Reingold, E.M.: Graph drawing by force-directed placement. Software: Practice and experience 21(11), 1129–1164 (1991)

[19] Greub, W., Halperin, S., Vanstone, R.: Connections, Curvature, and Cohomology V1: De Rham cohomology of manifolds and vector bundles. Academic press (1972)

[20] Grigor’yan, A.: Heat kernels on weighted manifolds and applications. Cont. Math 398, 93–191 (2006)

[21] Hateren, J.H.v., Schaaf, A.v.d.: Independent component filters of natural images compared with simple cells in primary visual cortex. Proceedings: Biological Sciences 265(1394), 359–366 (1998)

[22] Hein, M., Audibert, J.Y., Luxburg, U.v.: Graph laplacians and their convergence on random neighborhood graphs. Journal of Machine Learning Research 8(Jun), 1325–1368 (2007)

[23] Horak, D., Jost, J.: Spectra of combinatorial laplace operators on simplicial complexes. Advances in Mathematics 244, 303–336 (2013)

[24] Izbicki, M., Shelton, C.: Faster cover trees. In: Proceedings of the 32nd International Conference on Machine Learning (ICML-15), pp. 1162–1170 (2015)

[25] Jiang, X., Lim, L.H., Yao, Y., Ye, Y.: Statistical ranking and combinatorial hodge theory. Mathematical Programming 127(1), 203–244 (2011)

[26] Kato, T.: Perturbation theory for linear operators, vol. 132. Springer Science & Business Media (2013)

[27] Knyazev, A.V.: Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method. SIAM journal on scientific computing 23(2), 517–541 (2001)
[28] Koren, Y., Carmel, L., Harel, D.: Ace: A fast multiscale eigenvectors computation for drawing huge graphs. In: Information Visualization, 2002. INFOVIS 2002. IEEE Symposium on, pp. 137–144. IEEE (2002)

[29] Lanczos, C.: An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. Journal of Research of the National Bureau of Standards 45(4) (1950)

[30] Liao, S., Zhu, X., Lei, Z., Zhang, L., Li, S.Z.: Learning multi-scale block local binary patterns for face recognition. In: International Conference on Biometrics, pp. 828–837. Springer (2007)

[31] McInnes, L., Healy, J.: Accelerated hierarchical density based clustering. In: Data Mining Workshops (ICDMW), 2017 IEEE International Conference on, pp. 33–42. IEEE (2017)

[32] Mills-Curran, W.C.: Calculation of eigenvector derivatives for structures with repeated eigenvalues. AIAA journal 26(7), 867–871 (1988)

[33] Müllner, D., Babu, A.: Python mapper: An open-source toolchain for data exploration, analysis and visualization (2013). URL http://danifold.net/mapper

[34] Munch, E., Wang, B.: Convergence between categorical representations of reeb space and mapper. In: 32nd International Symposium on Computational Geometry (SoCG 2016), pp. 53:1–53:16. Dagstuhl Publishing, Germany: Leibniz Center for Informatics (2016)

[35] Nadler, B., Lafon, S., Kevrekidis, I., Coifman, R.R.: Diffusion maps, spectral clustering and eigenfunctions of fokker-planck operators. In: Advances in neural information processing systems, pp. 955–962 (2006)

[36] Parlett, B.N.: The rayleigh quotient iteration and some generalizations for nonnormal matrices. Mathematics of Computation 28(127), 679–693 (1974)

[37] Platt, J.C.: Fastmap, metricmap, and landmark mds are all nyström algorithms. In: AISTATS 2005 Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics, pp. 261–268. The Society for Artificial Intelligence and Statistics (2005)

[38] Reaven, G., Miller, R.: An attempt to define the nature of chemical diabetes using a multidimensional analysis. Diabetologia 16(1), 17–24 (1979)

[39] Reininghaus, J., Huber, S., Bauer, U., Kwitt, R.: A stable multi-scale kernel for topological machine learning. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4741–4748. IEEE (2015)

[40] Rustamov, R.M.: Laplace-beltrami eigenfunctions for deformation invariant shape representation. In: Proceedings of the fifth Eurographics symposium on Geometry processing, pp. 225–233. Eurographics Association (2007)

[41] Sermanet, P., LeCun, Y.: Traffic sign recognition with multi-scale convolutional networks. In: Neural Networks (IJCNN), The 2011 International Joint Conference on, pp. 2809–2813. IEEE (2011)

[42] Shewchuk, J.R., et al.: An introduction to the conjugate gradient method without the agonizing pain (1994)

[43] Si, S., Shin, D., Dhillon, I.S., Parlett, B.N.: Multi-scale spectral decomposition of massive graphs. In: Advances in Neural Information Processing Systems, pp. 2798–2806 (2014)

[44] Simoncelli, E.P., Freeman, W.T.: The steerable pyramid: A flexible architecture for multi-scale derivative computation. In: Image Processing, 1995. Proceedings., International Conference on, vol. 3, pp. 444–447. IEEE (1995)

[45] Singh, G., Mémoli, F., Carlsson, G.E.: Topological methods for the analysis of high dimensional data sets and 3D object recognition. In: Eurographics Symposium on Point Based Graphics, pp. 91–100. European Association for Computer Graphics (2007). DOI 10.2312/SPBG/SPBG07/091-100

[46] Stewart, G.W.: Stochastic perturbation theory. SIAM review 32(4), 579–610 (1990)

[47] Tenenbaum, J.B., De Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. science 290(5500), 2319–2323 (2000)

[48] Ting, D., Huang, L., Jordan, M.: An analysis of the convergence of graph laplacians. arXiv preprint arXiv:1101.5435 (2011)

[49] Urschel, J.C., Hu, X., Xu, J., Zikatanov, L.T.: A cascadic multigrid algorithm for computing the fiedler vector of graph laplacians. arXiv preprint arXiv:1412.0565 (2014)

[50] Venna, J., Kaski, S.: Local multidimensional scaling. Neural Networks 19(6-7), 889–899 (2006)

[51] Wu, K., Simon, H.: Thick-restart lanczos method for large symmetric eigenvalue problems. SIAM Journal on Matrix Analysis and Applications 22(2), 602–616 (2000)