Topological Deep Learning with the Heat Kernel

Matt Piekenbrock
Khoury College of Computer Sciences
Northeastern University
Boston, Massachusetts
piekenbrock.m@northeastern.edu

Jose Perea
Khoury College of Computer Sciences,
College of Mathematics
Northeastern University
Boston, Massachusetts
j.pereabenitez @ northeastern.edu

***Abstract*—Relational strctures, such as graphs and hypergraphs, provide a versatile framework for modeling entity interactions across multiple domains. With the rise of geometric deep learning models, such graph convolutional networks (GCNs), these structures have become central to modern machine learning pipelines. Inspired by their success in pose-invariant shape analysis applications, we introduce a novel framework for encoding higher-order (hyperedge) interactions based on simplicial diffusion. Central to our approach is the matrix functional calculus, which we exploit to develop a highly scalable algorithm for extracting provably informative and multsicale features from hypergraphs. Confirming recent work on the topic, our empirical analysis suggests that from a learning perspective, a weighted simplicial complex conveys as much information as a hypergraph does without loss of information.**

***Index terms*—Geometric Deep Learning, Graph Neural Networks, Topological data analysis, Linear Algebra**

I. Introduction

Many real-world datasets involve complex, higher-order relationships that cannot be fully characterized by pairwise interactions alone. Hypergraphs are a natural and powerful way to represent such higher order structures; however, their lack of structure presents challenges for learning tasks, preventing the full utilization of techniques from **geometric deep learning** [1], which aims to generalize deep neural network models to non-Euclidean spaces (e.g graphs, manifolds). Indeed, recent work has shown certain simplicial relaxations of higher-order data can efficiently surpass existing graph-based learning tasks [2].

Building on these ideas, we propose a simple and efficient framework to extend the scope of hypergraphs to the more structured simplicial domain. Following the seminal work of [3], our method encodes hypergraphs as weighted simplicial complexes without loss of information. One notable application of this conversion is the exploitation of the heat kernel's properties, including the generation of multiscale, informative signatures that effectively capture higher-order interactions.

A. Overview

We briefly summarize our approach. Let $H = (V, \mathcal{E})$ denote undirected hypergraph with which one wishes to extract features for learning purposes. Our proposed framework for producing such as featurization is as follows:

1) Construct the d-dimensional simplicial closure S of H:
$$S(H) = \{\sigma \subseteq e : e \in \mathcal{E}\} \tag{1}$$
2) Define a map $w : S \to \mathbb{R}_+$ comprised of each simplex's **topological weight** w_σ and its **affinity weight** ω_σ:
$$w(\sigma) = \omega_\sigma + \sum_{\sigma' \in \operatorname{cofacet}(\sigma)} w_{\sigma'} \tag{2}$$
We discuss ways to compute these weights in Section IV.A.
3) To capture H's higher-order interactions, construct a p -Laplacian $\mathcal{L}_p$ from S, such as the *Hodge laplacian*:
$$\mathcal{L}_p = L_p^{\text{up}}(w) + L_p^{\text{dn}}(w) \tag{3}$$
4) The final featurization X on the p-simplices is defined by the diagonal of the a chosen matrix function:
$$X := \operatorname{diag}\big(f(\mathcal{L}_p)\big) = \operatorname{diag}(U f(\Lambda) U^T) \tag{4}$$
where $f : \mathbb{R}^n \to \mathbb{R}$ is an appropriate spectral function.

Our proposed featurization method (4) is succinct in that it is $O(n)$-sized for each function f, where $n = \left|S_p\right|$ is the number p-simplices in S. In Section IV.C we show that, for certain types of functions f (e.g. those deriving from the matrix exponential), the corresponding feature vectors are provably *informative*, *stable*, and *multiscale*.

II. Background work

TODO

III. Notation

In this work, a *hypergraph* (or family of sets) is a pair $H = (V, \mathcal{E})$ where V is a finite set of elements called *vertices* and $\mathcal{E} \subseteq \mathcal{P}(V)$ is any subset of the power set $\mathcal{P}(V)$ of V. A *simplicial complex* is a pair $S = (V, \Sigma)$ satisfying the properties that (a) if $v \in V$, then $\{v\} \in \Sigma$ and (b) if $\tau \subset \sigma$ for any $\sigma \in \Sigma$, then $\tau \in \Sigma$. Denote by S_p the set of p-simplices of S. An *oriented* simplex, denoted by $[\sigma]$, is a simplex $\sigma \in S$ with an ordering on its vertices. For simplicity, we will always assume to be in-

duced by an ordering on V. The p-th *chain group* $C_p(S, \mathbb{R})$ of S is the vector space over $\mathbb{R}$ with basis $\{[\sigma] : \sigma \in S_p\}$.

Let S be a simplicial complex with p-simplices S_p. A *weight function* of S is any positive function $w : S \to (0, \infty)$, i.e. any function that maps every simplex $\sigma \in S$ to a strictly positive weight. S is called *unweighted* is $w(\sigma) = 1$ for all $\sigma \in S$.

IV. Methodology

There are many ways to map higher-order interactions, such as hyperedges $\mathcal{E}$, to weighted simplices. Given a hypergraph H and its simplicial closure S, one such lossless encoding is given by the trivial identity map:

$$\begin{aligned} w : S &\to \mathbb{R}_+ \\ \sigma &\mapsto \mathbb{1}(\sigma \in \mathcal{E}) \end{aligned} \tag{5}$$

By construction, the closure S must contain a face $\sigma \in S$ for every hyperedge $e \in \mathcal{E}$, thus the map is well-defined.

The main issue with the map from (5)—aside from the potential size of S—is that its dynamics are less understood due to its lack of well-defined asymptotic behavior, as its not strictly a weight function in the sense above. Indeed, given any $p \in \mathbb{N}$, a weight function defines an inner product $\langle [\sigma], [\sigma'] \rangle_w$ on the chain group $C_{p(S,\mathbb{R})}$ as follows:

$$\langle [\sigma], [\sigma'] \rangle_w := \delta_{\sigma\sigma'} \cdot (w(\sigma))^{-1}, \forall \sigma, \sigma' \in S_p \tag{6}$$

Similarly, $\langle \cdot, \cdot \rangle_w$ induces an inner product on the cochain space $C^p(S, \mathbb{R})$ of S:

$$\|\langle f, g \rangle\|_w = \sum_{\sigma \in S_p} w(\sigma) \cdot f([\sigma]) g([\sigma])) \tag{7}$$

There is a one-to-one correspondence between weight functions and possible inner products on cochain groups $C^{p(S,\mathbb{R})}$, where elementary cochains are orthogonal [4]. Denote by $\partial_p^* : C_{p-1} \to C_p$ the adjoint of ∂_p under these inner products. The p-th *combinatorial Laplacian* $\mathcal{L}_p : C_p(S, \mathbb{R}) \to C_p(S, \mathbb{R})$ is defined as:

$$\Delta_p := \partial_{p+1} \circ \partial_{p+1}^* + \partial_p^* \circ \partial_p \tag{8}$$

Let W_p denote the diagonal representation of w applied to the p-simplices of S. The matrix representations of these operators is given by:

$$\mathcal{L}_p = \underbrace{\partial_{p+1} W_{p+1} \partial_{p+1}^T W_p^{-1}}_{L_p^{\text{up}}} + \underbrace{W_p \partial_p^T W_{p-1}^{-1} \partial_p}_{L_p^{\text{dn}}} \tag{9}$$

Clearly, the above expression is only well-defined if the corresponding weight function w takes strictly positive values, preventing the use of characteristic functions akin to (5).

A. Choosing the weights

In the absence of any 'natural' map from hyperedges to weighted simplices, Baccini et al. [3] proposed to weight each simplex $\sigma \in S$ in the closure of H via a combiniation of a *topological* and *affinity* weight. Specifically, for each hyperedge $e \in \mathcal{E}$ of dimension d, a d-dimensional simplex σ is constructed with each k-dimensional face $\tau \subseteq \sigma$ is assigned the following topological weight:

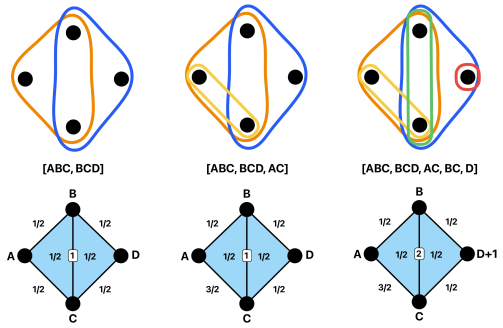

Fig. 1: Each simplexes *topological weight* sums of the weights of its codim-1 faces; each vertex weight matches the number of hyperedges it participates in.

$$w_\sigma(\tau) = (d - k)! \,/\, d! \tag{10}$$

Thus, both singletons and pairwise interactions (edges) are assigned unit weight, whereas higher-order simplices have successively lower weights. Some of the properties of this weight function are *strict posivity*, *monotonicity with respect to the face poset*, and *normalized scale*:

$$\begin{cases} w(\sigma) > 0 & \text{for every face } \sigma \in S \\ w(\sigma) \geq \sum_{\tau \supset \sigma} w(\tau) & \text{for every codim-1 } \tau \in S \\ w(v) = \sum_{e \in \mathcal{E}} \mathbb{1}(v \in \mathcal{E}) & \text{for all } v \in V \end{cases}$$

Moreover, this weight function is *additive* in the sense that the weight of any simplex $\sigma \in S$ in the simplicial closure of S of H is given by summing the weights disjoint union of the hyperedges:

$$w(\sigma) = \sum_{e \in \mathcal{E}} w_e(\sigma)$$

It may be readily verified that any choice of affinity weights ω_σ may be readily reconstructed from the final weight map $w : S \to \mathbb{R}_+$ using (2). To see this with an example, see Fig. 1.

B. Choosing the featurization

To featurize the weighted simplicial complexes for learning purposes, we exploit the functional calculus of matrices, which parameterizes linear operators via weighted projections onto their eigenspaces:

$$f(A) := \sum_{i=1}^{n} f(\lambda_i) P_i$$

where $P_i P_j = 0$ for $i \neq j$ and $\sum_{i=1}^n P_i = I$. Note each projector P_i is uniquely determined by the eigenspace U_{λ_i} generated by the pair (λ_i, u_i), i.e. the subspace of $\mathbb{R}^n$ wherein the action of A mimics scalar multiplication:

$$U_\lambda = \{\, u \in U : Au = \lambda u \,\}$$

In the literature, these linear maps are referred to as *Lowner* operators. One benefit of using the matrix functional calculus is that derivatives are available in closed-form.

Theorem 1 (Differentiability): if $f : \mathbb{R} \to \mathbb{R}$ is continuously differentiable, then the differential ∇F of F is available in closed-form:

$$\nabla F(A)[H] = U\big(f^{[1]}(\Lambda) \circ (U^T H U)\big)U^T$$

where $f^{[1]}$ denotes the first divided differences matrix:

$$\big(f^{[1]}(x)\big)_{ij} = \begin{cases} f'(x_i) \\ \frac{f(x_i)-f(x_j)}{x_i - x_j} \end{cases}$$

Different matrix functions inevitably produce different featurizations, some of which may be useful for learning purposes. A few exemplary parameterizations are given below:

$$f(A) = \mathbb{1}(\lambda \geq \lambda_k) \leftrightarrow \text{ Principle Component Analysis}$$
$$f(A) = \exp(-tA) \leftrightarrow \text{ Matrix exponential}$$

- $f(A) = \mathbb{1}(\lambda \geq \lambda_k) \leftrightarrow$ Principle Component Analysis
- $f(A) = e^{-tA} \leftrightarrow$ Matrix exponential
- $f(A) = (A + \lambda I)^{-1}$ Tikhonov Regularization
- $f(A) = (I - \alpha A)^{\{-1\}}$ PageRank

For more examples of common uses of these operators, see [5].

C. Diffusion-based signatures

Let $G = (V, E)$ denote a graph over $n = |V|$ vertices and $m = |E|$ edges. A *diffusion process* across G is modeled via a time-varying vector $v(t) \in \mathbb{R}^n$ where each component $v_{i(t)} \in \mathbb{R}$ represents the value of the diffusion at vertex v_i. If E comes equipped with edge weights w_{ij} representing *conductivity*, the *flow* from v_i to v_j for any pair $\big(v_i, v_j\big) \in V \times V$ is defined by the quantity $w_{ij}\big(v_i(t) - v_j(t)\big)$. Summing flows across neighbors yields the change in diffusion values:

$$v_{j'}(t) = \sum_{i \sim j} w_{ij}\big(v_i(t) - v_j(t)\big)$$

This change in diffusion is captured by the *graph Laplacian* L:

$$v'(t) = -Lv(t) \quad \leftrightarrow \quad L \cdot u(x,t) = -\frac{\partial u(x,t)}{\partial t}$$

With initial conditions $v(0) \in \mathbb{R}^n$ representing the amount of heart at each vertex at time $t = 0$, the solution to the partial differential equation above is given by the *Laplacian exponential diffusion kernel*:

$$v(t) = \exp(-tL)v(0)$$

Where the matrix operator $\exp(-tL)$, given in closed-form, is called the *heat kernel* at time $t \geq 0$:

$$\exp(-tL) = H_t = U \exp(-t\Lambda)U' = \sum_{i=1}^{n} e^{-t\lambda_i} u_i u_i^T$$

In other words, the value $v_i(t)$ describes the heat at vertex v_i at time t after a diffusion of heat given by $v(0)$ [6].

Due to its myriad of attractive properties and its intrinsic connection to diffusion, the heat kernel is a natural choice of matrix function for characterizing higher-order interactions. Indeed, it is an *isometric invariant* and *informative*, as shown by the following theorem.

Theorem 2 (Intrinsic & Informative [7]): Let M, N be compact Riemannian manifolds, and let $T : M \to N$ be a surjective map. If $H_t(T(x), T(y)) = H_t(x, y)$ for any $x, y \in M$ and $t > 0$, then T is an isometry. Moreover, if T is an isometry, then $H_t(x, y) = H_t(T(x), T(y))$ for any $x, y \in M$ and $t > 0$.

In other words, from a shape / geometric perspective, the heat kernel contains all of the information about the intrinsic geometry of the underlying shape and hence fully characterizes shapes up to isometry.

Our interest in the heat kernel stems from the fact that it is possible to extract *stable* features which capture higher-order interactions from it. One such featurization—the *Heat Kernel Signature* [7] (HKS)—is often used to distinguish shapes in pose-invariant 3D contexts due to its multiscale nature. Given a Laplace-Beltrami operator L with eigenpairs $(\lambda_i, \phi_i)_{i=1}^n$, it is defined as:

$$\text{hks}_t(x) := \sum_{i=1}^{n} e^{-t\lambda_i}\phi_i(x)\phi_j(x)$$

In other words, the HKS at a fixed point x is simply a restriction of the heat kernel to the temporal domain. Surprisingly, it can be shown that under relatively mild assumptions, the HKS characterizes all of the information of the Heat Kernel (see [7]). Moreover, for a fixed simplicial complex $S = (V, \Sigma)$ and its corresponding Laplacian $L_p = U\Lambda U^T$, it is straightforward to see from the definition above that it is in fact equivalent to the diagonal of the heat kernel.

$$\text{hks}_t(V) = \text{diag}(H_t) = \text{diag}(U \exp(-t\Lambda)U^T)$$

The stability and multiscale nature of the HKS originally motivated its use in multi-scale shape matching. As the diagonal of a particular type of matrix function, it fits squarely into our proposed framework via (4).

Example: Consider the set of all hypergraphs whose simplicial closures are given by the maximal simplices $S = \{(0,1,2),(1,2,3)\}$. As there are 9 non-maximal faces in the closure, there are 2^9 distinct hypergraphs with identical closures; for any conversion to be useful, we would like a lift capable of *distinguishing* between hypergraphs in this set.

Instead, in Fig. 2 we plot 5 hypergraphs and their weighted closures using (2). In the third column, we show a diffusion process starting with a unit amount of heat at the red vertex. Thoughout the diffusion, the red vertex (0) starts with a unit amount of heat, which it diffuses through the weighted edges (0,1) and (0,2). In the 3rd row, observe the blue vertex heats up faster than the green due to having higher conductivity, and the orange vertex (3) heats up the slowest due to being not adjacent to the heat source (0).

In the fourth column, we plot the HKS featurization of the bridge network at the same time points. Note this does not depend on an initial distribution of heat. Observe the HKS not only distinguishes distinct hypergraphs, but also symmetrically related graphs are handled naturally.

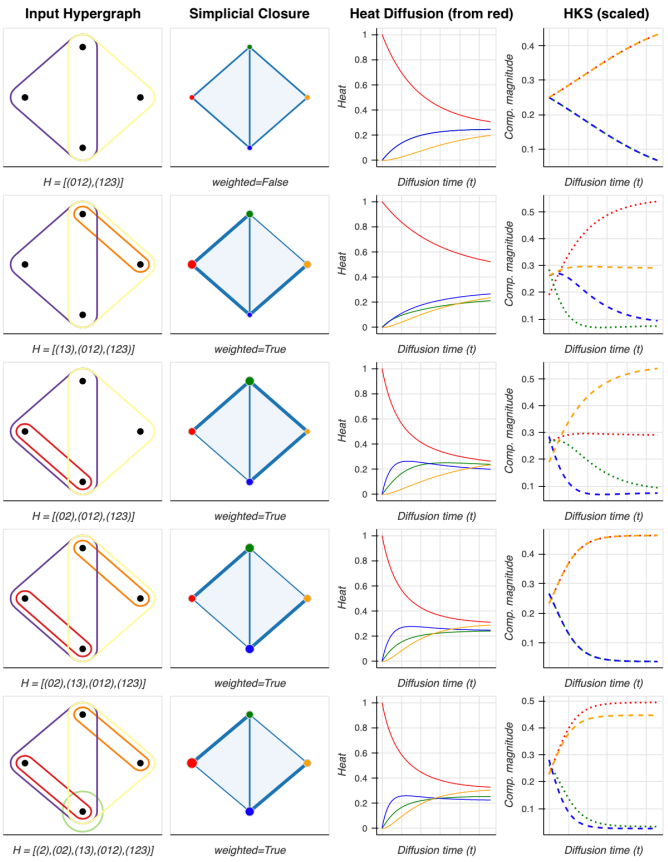

Fig. 2: Hypergraphs, their weighted closures, diffusion curves, and signatures. The 'width' of each edge is inversely proportional to its hypergraph-induced weight, changing both the flow rate across edges and the signatures.

V. Computation

Paramount to the interoperability of our proposed framework with modern machine learning method is its scalability with respect to large data sets. While computation of (4) can be accomplished by standard linear algebraic methods (e.g. Cuppens divide-and-conquer), for modern featurization methods to be competitive they typically must scale no faster than quadratic, and ideally they ought to work with sparse inputs.

Fortunately, as we show below, any featurization of the form $\mathrm{diag}(f(\mathcal{L}))$ where $\mathcal{L}$ is a Laplacian matrix can not only be computed exactly in $O(n^2)$ time and $O(n)$ space, but can also be iteratively $(1 \pm \varepsilon)$-approximated in linear time.

A. Exact computation

The heart of both the exact and approximate computations is the *Lanczos method*. Given a symmetric $A \in \mathbb{R}^{n \times n}$ with eigenvalues $\lambda_1 \geq \lambda_2 > \ldots \geq \lambda_r > 0$ and a vector $v \neq 0$, the Lanczos method generates the triple (K, Q, T):

$$K = [\, A^0 v \mid A^1 v \mid A^2 v \mid \ldots \mid A^{r-1} v \,]$$
$$Q = [\, q_1 \mid q_2 \mid \ldots \mid q_r \,] \longleftarrow \mathrm{qr}(K)$$
$$T = Q^T A Q$$

where Q is orthogonal, K is called the *Krylov matrix* with respect to (A, v), and T has a tridiagonal form. It can be shown that T is similar to A, and that its eigenvalues $\Lambda(T)$ may be obtained in $O(n^2)$ time (and $O(n)$ space), reducing the problem to obtaining T itself.

A classical result, due to Lanczos' *three-term recurrence*, shows that none of these matrices (K, Q, T) need be formed explicitly[8]; at most three vectors (q_{i-1}, q_i, q_{i+1}) need to be stored in memory at any given time. This is perhaps best demonstrated by the following result:

Theorem 3 (Lanczos complexity [9]): Given a symmetric rank-r matrix $A \in \mathbb{R}^{n \times n}$ whose matrix-vector operator $A \mapsto Ax$ requires $O(\eta)$ time and $O(\nu)$ space, the Lanczos iteration computes $\Lambda(A) = \{\lambda_1, \lambda_2, \ldots, \lambda_r\}$ in $O(\max\{\eta, n\} \cdot r)$ time and $O(\max\{\nu, n\})$ space, when executed in exact arithmetic.

Clearly, LemmaTheorem 3 implies a $O(nr)$ time and $O(n)$ space computation route so long as we can guarantee the action $v \mapsto Av$ is efficient enough (and we allow for exact arithmetic.) To address the former, we give the following result:

Corollary 3.1 *(Linear space complexity)*: Any featurization that can be written of the form given by (4), i.e. as the diagonal of a a matrix function of a combinatorial Laplacian operator, can be computed using $O(\max(n, m))$ space, where $n = |S_p|$ and $m = |S_{p+1}|$ are the number of p- and $(p+1)$-simplices in S, respectively.

Ideally, we would like to extend the analysis to a more realistic computation model (e.g. finite-precision WordRAM). However, the Lanczos method is known to break down from numerical issues that not only muddle the convergence criteria of the algorithm, but also pose barriers to analysis. Indeed, it's well known in practice that precise eigenvalue computation on the interior of the spectrum is infeasible to do with Lanczos in finite-precision unless (typically expensive) heuristics are added to the computation, such as partial reorthogonalization (cite) or deflation techniques (cite), however such techniques often increase computation time substantially and require $O(n^2)$ space. As a result, most uses of Lanczos in practice are isolated to use-cases that depend on the extremal parts of the spectrum. The Lanczos method is the primier method remains used in practice. (footnote)

Recently (in 2022), Musco and Musco established the first results regarding the bounds of the Lanczos method in exact arithmetic. IN particular, it was shown that the Paige's $O(n)$ -space A27 variant of Lanczos actually achieves within a constant factor the optimal finite-precision approximation of the matrix function. (cite)

Theorem 4 ():

This is, to our knowledge, the first finite precision results justifying the use of Lanczos on such problems (as compared to alternative algorithms, such as the Jacobi Davidson or Cheby-

chev approximations). Indeed, Chen et al. recommend the implementation of Cheychev approximations *via* Lanczos (cite). Combining this result with the prior two Lemmas, we have our first algorithm and corresponding theorem.

Theorem 5 (): Any featurization of the form $\mathrm{diag}(f(\mathcal{L}))$ can be computed in finite precision in $O(mr)$ time and and $O(\min(m,n))$ space.

The corresponding algorithm is given below.

B. $(1 \pm \varepsilon)$-*approximations*

While exact computation of the featurization is quite scalable, for some types of matrix functions it may be admissible to consider aproximations. One such approach is to exploit the the Girard-Hutchinson-type diagonal estimator, inspired by the trace estimator of the same kind:

+

Theorem 6: Under mild assumptions, if the function f is analytic in the domain $[\lambda_{\min}, \lambda_{\max}]$, any signature of the form (4) can be $(1 \pm \varepsilon)$-approximated using on the order of $O()$ matrix-function evaluations $v \mapsto f(A)v$.

Akin to the trace estimator, it's been shown that this is an unbiased estimator is that converges to the diagonal of a given operator A in $O()$. For some types of matrix functions, such as those coming from low-rank or fucntion thats depend on the magnitude of the larger eigenspaces, this crude Monte-carlo can be more than sufficient for featurization purposes.

However, it is possible to imrpovement. The famous Hutch++ estimator achieves not only achieves a $(1 \pm \varepsilon)$-approximation but also shows that it's possible to

More recent work has improved this yet again, see XTrace and Krylov aware block Lanczos.

C. Benchmarks

We implement the Hutchinson, Hutch++, divide-and-conquer, and Lanczos with reorthogonalization on real-world simplicial complex to compare the balance between approximation quality, computation time, and space usage.

VI. Conclusion

References

[1] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: going beyond euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.

[2] R. Yang, F. Sala, and P. Bogdan, "Efficient representation learning for higher-order data with simplicial complexes," in *Learning on Graphs Conference*, 2022, pp. 13–11.

[3] F. Baccini, F. Geraci, and G. Bianconi, "Weighted simplicial complexes and their representation power of higher-order network data and topology," *Physical Review E*, vol. 106, no. 3, p. 34319–34320, 2022.

[4] D. Horak and J. Jost, "Spectra of combinatorial Laplace operators on simplicial complexes," *Advances in Mathematics*, vol. 244, pp. 303–336, 2013.

[5] C. Musco, C. Musco, and A. Sidford, "Stability of the Lanczos method for matrix function approximation," in *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2018, pp. 1605–1624.

[6] J. Lafferty, G. Lebanon, and T. Jaakkola, "Diffusion kernels on statistical manifolds.," *Journal of Machine Learning Research*, vol. 6, no. 1, 2005.

[7] J. Sun, M. Ovsjanikov, and L. Guibas, "A concise and provably informative multi-scale signature based on heat diffusion," in *Computer graphics forum*, 2009, pp. 1383–1392.

[8] C. C. Paige, "Computational variants of the Lanczos method for the eigenproblem," *IMA Journal of Applied Mathematics*, vol. 10, no. 3, pp. 373–381, 1972.

[9] H. D. Simon, "Analysis of the symmetric Lanczos algorithm with reorthogonalization methods," *Linear algebra and its applications*, vol. 61, pp. 101–131, 1984.

Appendix

A. Proofs

Proof of Theorem 6: The proof of the above statement essentially follows from combining three recent works:

1) Ubaru's finite-precision results on the stability of the Lanczos method
2) The stochastic Hutchinson-style diagonal estimator (applied to matrix functions)
3) The above proof that the heat kernel reduces to a diagonal

□

Proof of Corollary 3.1: Proof: If the operation $v \to Av$ can be done in $O(\nu)$ space, it follows from the 3-term recurrence of the Lanczos method and the fact that combinatorial Laplacian require $O(n)$ space to perform matrix-vector actions that the above signature can be computed in space complexity $O(n)$. □

Proof : The proof of the finite-precision follows directly by combining Lemma's () and (), see for more details. To show that the space complexity cannot exceed $O(\min(m,n))$, we must show that both

$$\mathcal{L} = \partial_1 \partial_1^T + \partial_1^T \partial_1$$

i.e. that the up and down Laplacians both require at most $O(\min(m,n))$-space in their matrix-vector actions. To show this, we recall onnections between the up- and down- Laplacians:

+

Thus, the rank of either Laplacina is at most $\min(m,n)$. □