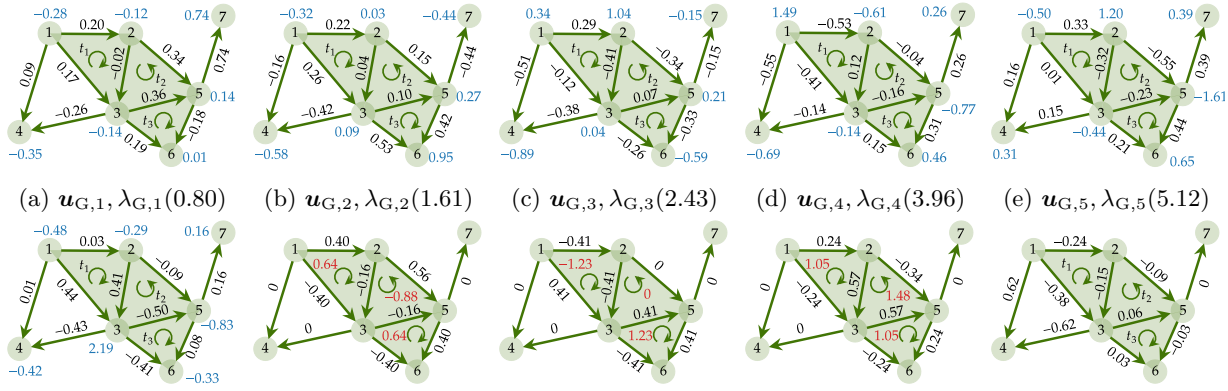
Contents

A Illustration for Background

Ths paper relies on the Hodge decomposition and the spectral simplicial theory. To ease the exposition, we illustrate them for the edge flow space. We refer to Barbarossa & Sardellitti (2020); Yang et al. (2021; 2022b) for more details.



(a) $\boldsymbol{u}_{\mathrm{G},1}, \lambda_{\mathrm{G},1}(0.80)$ (b) $\boldsymbol{u}_{\mathrm{G},2}, \lambda_{\mathrm{G},2}(1.61)$ (c) $\boldsymbol{u}_{\mathrm{G},3}, \lambda_{\mathrm{G},3}(2.43)$ (d) $\boldsymbol{u}_{\mathrm{G},4}, \lambda_{\mathrm{G},4}(3.96)$ (e) $\boldsymbol{u}_{\mathrm{G},5}, \lambda_{\mathrm{G},5}(5.12)$

(f) $\boldsymbol{u}_{\mathrm{G},6}, \lambda_{\mathrm{G},6}(6.08)$ (g) $\boldsymbol{u}_{\mathrm{C},1}, \lambda_{\mathrm{C},1}(1.59)$ (h) $\boldsymbol{u}_{\mathrm{C},2}, \lambda_{\mathrm{C},2}(3.00)$ (i) $\boldsymbol{u}_{\mathrm{C},3}, \lambda_{\mathrm{C},3}(4.41)$ (j) $\boldsymbol{u}_{\mathrm{H}}, \lambda_{\mathrm{H}}(0)$

Figure 9: (a)-(f) Six gradient frequencies and the corresponding Fourier basis. We also annotate their divergences, and we see that these eigenvectors with a small eigenvalue have a small magnitude of total divergence, i.e., the edge flow variation in terms of the nodes. Gradient frequencies reflect the nodal variations. (g)-(i) Three curl frequencies and the corresponding Fourier basis. We annotate their curls and we see that these eigenvectors with a small eigenvalue have a small magnitude of total curl, i.e., the edge flow variation in terms of the triangles. Curl frequencies reflect the rotational variations. (j) Harmonic basis with a zero frequency, which has a zero nodal and zero rotational variation.

A.1 Spectral simplicial theory

Here we show how the eigenvalues of $\boldsymbol{L}_k$ carry the notion of simplicial frequency Yang et al. (2022b). Specifically, we show for $k = 1$ an eigenvalue measures the total divergence or curl of the eigenvector.

- *Gradient Frequency:* the nonzero eigenvalues associated with the eigenvectors $\boldsymbol{U}_{1,\mathrm{G}}$ of $\boldsymbol{L}_{1,\mathrm{d}}$, which span the gradient space $\mathrm{im}(\boldsymbol{B}_1^\top)$, admit $\boldsymbol{L}_{1,\mathrm{d}}\boldsymbol{u}_{1,\mathrm{G}} = \lambda_{1,\mathrm{G}}\boldsymbol{u}_{1,\mathrm{G}}$ for any eigenpair $\boldsymbol{u}_{1,\mathrm{G}}$ and $\lambda_{1,\mathrm{G}}$. Thus, we have $\lambda_{1,\mathrm{G}} = \boldsymbol{u}_{1,\mathrm{G}}^\top \boldsymbol{L}_{1,\mathrm{d}}\boldsymbol{u}_{1,\mathrm{G}} = \boldsymbol{u}_{1,\mathrm{G}}^\top \boldsymbol{B}_1^\top \boldsymbol{B}_1 \boldsymbol{u}_{1,\mathrm{G}} = \|\boldsymbol{B}_1\boldsymbol{u}_{1,\mathrm{G}}\|_2^2$, which is an Euclidean norm of the divergence, i.e., the total nodal variation of $\boldsymbol{u}_{1,\mathrm{G}}$. If an eigenvector has a larger eigenvalue, it has a larger total divergence. For the SFT of an edge flow, if the gradient embedding $\tilde{\boldsymbol{x}}_{1,\mathrm{G}}$ has a large weight on such an eigenvector, it contains components with a large divergence, and we say it has a large gradient frequency. Thus, we call such eigenvalues associated with $\boldsymbol{U}_{1,\mathrm{G}}$ gradient frequencies.
- *Curl Frequency:* the nonzero eigenvalues associated with the eigenvectors $\boldsymbol{U}_{1,\mathrm{C}}$ of $\boldsymbol{L}_{1,\mathrm{u}}$, which span the curl space $\mathrm{im}(\boldsymbol{B}_2)$, admit $\boldsymbol{L}_{1,\mathrm{u}}\boldsymbol{u}_{1,\mathrm{C}} = \lambda_{1,\mathrm{C}}\boldsymbol{u}_{1,\mathrm{C}}$ for any eigenpair $\boldsymbol{u}_{1,\mathrm{C}}$ and $\lambda_{1,\mathrm{C}}$. Thus, we have $\lambda_{1,\mathrm{C}} = \boldsymbol{u}_{1,\mathrm{C}}^\top \boldsymbol{L}_{1,\mathrm{u}}\boldsymbol{u}_{1,\mathrm{C}} = \boldsymbol{u}_{1,\mathrm{C}}^\top \boldsymbol{B}_2\boldsymbol{B}_2^\top \boldsymbol{u}_{1,\mathrm{C}} = \|\boldsymbol{B}_2^\top \boldsymbol{u}_{1,\mathrm{C}}\|_2^2$, which is an Euclidean norm of the curl, i.e., the total rotational variation of $\boldsymbol{u}_{1,\mathrm{C}}$. If an eigenvector has a larger eigenvalue, it has a larger total curl. For the SFT of an edge flow, if the curl embedding $\tilde{\boldsymbol{x}}_{1,\mathrm{C}}$ has a large weight on such an eigenvector, it contains components with a large curl, and we say it has a large curl frequency. Thus, we call such eigenvalues associated with $\boldsymbol{U}_{1,\mathrm{C}}$ curl frequencies.
- *Harmonic Frequency:* the zero eigenvalues associated with the eigenvectors $\boldsymbol{U}_{1,\mathrm{H}}$, which span the harmonic space $\ker(\boldsymbol{L}_1)$, admit $\boldsymbol{L}_1\boldsymbol{u}_{1,\mathrm{H}} = \boldsymbol{0}$ for any eigenpair $\boldsymbol{u}_{1,\mathrm{H}}$ and $\lambda_{1,\mathrm{H}} = 0$. From the definition of $\boldsymbol{L}_1$, we have $\boldsymbol{B}_1\boldsymbol{u}_{1,\mathrm{H}} = \boldsymbol{B}_2^\top \boldsymbol{u}_{1,\mathrm{H}} = \boldsymbol{0}$. That is, the eigenvector $\boldsymbol{u}_{1,\mathrm{H}}$ is divergence- and curl-free. We also say such an eigenvector has zero signal variation in terms of the nodes and triangles. This resembles the constant graph signal in the node space. We call such zero eigenvalues as harmonic frequencies.

Fig. 9 shows the simplicial Fourier basis and the corresponding simplicial frequencies of the SC, from which we see how the eigenvalues of $\boldsymbol{L}_1$ can be interpreted as the simplicial frequencies.

For $k = 0$, the eigenvalues of $\boldsymbol{L}_0$ carry the notion of graph frequency, which measures the graph (node) signal smoothness w.r.t. the upper adjacent simplices, i.e., edges. Thus, the curl frequency of $k = 0$ coincides with the graph frequency and a constant graph signal has only harmonic frequency component. For a more general k, there exist these three types of simplicial frequencies, which measure the k-simplicial signal total variations in terms of faces and cofaces.

B Simplicial 2-Complex CNNs and Details on Properties

We give two examples where the first is a SCCNN on a SC of order two, and the second is the form of SCCNN with multi-features.

Example 25. For $k = 2$, a SCCNN layer reads as

$$\begin{aligned}\boldsymbol{x}_0^l &= \sigma(\boldsymbol{H}_0^l \boldsymbol{x}_0^{l-1} + \boldsymbol{H}_{0,\mathrm{u}}^l \boldsymbol{B}_1 \boldsymbol{x}_1^{l-1}),\\ \boldsymbol{x}_1^l &= \sigma(\boldsymbol{H}_{1,\mathrm{d}}^l \boldsymbol{B}_1^\top \boldsymbol{x}_0^{l-1} + \boldsymbol{H}_1^l \boldsymbol{x}_1^{l-1} + \boldsymbol{H}_{1,\mathrm{u}}^l \boldsymbol{B}_2 \boldsymbol{x}_2^{l-1}),\\ \boldsymbol{x}_2^l &= \sigma(\boldsymbol{H}_{2,\mathrm{d}}^l \boldsymbol{B}_2^\top \boldsymbol{x}_1^{l-1} + \boldsymbol{H}_2^l \boldsymbol{x}_2^{l-1}).\end{aligned} \tag{16}$$

Recursively, we see that a SCCNN layer takes as inputs $\{\boldsymbol{x}_0^{l-1}, \boldsymbol{x}_0^{l-2}, \boldsymbol{x}_1^{l-2}, \boldsymbol{x}_2^{l-2}\}$ to compute $\boldsymbol{x}_0^l$. One may find this familar as some type of skip connections in GNNs Xu et al. (2018b).

Example 26 (Multi-Feature SCCNN). A multi-feature SCCNN at layer l takes $\{\boldsymbol{X}_{k-1}^{l-1}, \boldsymbol{X}_k^{l-1}, \boldsymbol{X}_{k+1}^{l-1}\}$ as inputs, each of which has F_{l-1} features, and generates an output $\boldsymbol{X}_k^l$ with F_l features as

$$\boldsymbol{X}_k^l = \sigma\left(\sum_{t=0}^{T_\mathrm{d}} \boldsymbol{L}_{k,\mathrm{d}}^t \boldsymbol{B}_k^\top \boldsymbol{X}_{k-1}^{l-1} \boldsymbol{W}_{k,\mathrm{d},t}^{\prime l} + \sum_{t=0}^{T_\mathrm{d}} \boldsymbol{L}_{k,\mathrm{d}}^t \boldsymbol{X}_k^{l-1} \boldsymbol{W}_{k,\mathrm{d},t}^l + \sum_{t=0}^{T_\mathrm{u}} \boldsymbol{L}_{k,\mathrm{u}}^t \boldsymbol{X}_k^{l-1} \boldsymbol{W}_{k,\mathrm{u},t}^l + \sum_{t=0}^{T_\mathrm{u}} \boldsymbol{L}_{k,\mathrm{u}}^t \boldsymbol{B}_{k+1} \boldsymbol{X}_{k+1}^{l-1} \boldsymbol{W}_{k,\mathrm{u},t}^{\prime l}\right) \tag{17}$$

where $\boldsymbol{L}^t$ indicates the matrix t-power of $\boldsymbol{L}$, while superscript l indicates the layer index.

B.1 Simplicial locality in details

The construction of SCFs has an intra-simplicial locality. $\boldsymbol{H}_k \boldsymbol{x}_k$, which consists of basic operations $\boldsymbol{L}_{k,\mathrm{d}} \boldsymbol{x}_k$ and $\boldsymbol{L}_{k,\mathrm{u}} \boldsymbol{x}_k$. They are given, on simplex s_i^k, by

$$[\boldsymbol{L}_{k,\mathrm{d}} \boldsymbol{x}_k]_i = \textstyle\sum_{j \in \mathcal{N}_{i,\mathrm{d}}^k \cup \{i\}} [\boldsymbol{L}_{k,\mathrm{d}}]_{ij} [\boldsymbol{x}_k]_j, \quad [\boldsymbol{L}_{k,\mathrm{u}} \boldsymbol{x}_k]_i = \textstyle\sum_{j \in \mathcal{N}_{i,\mathrm{u}}^k \cup \{i\}} [\boldsymbol{L}_{k,\mathrm{u}}]_{ij} [\boldsymbol{x}_k]_j, \tag{18}$$

where s_i^k aggregates signals from its lower and upper neighbors, $\mathcal{N}_{i,\mathrm{d}}^k$ and $\mathcal{N}_{i,\mathrm{u}}^k$. We can compute the t-step shifting recursively as $\boldsymbol{L}_{k,\mathrm{d}}^t \boldsymbol{x}_k = \boldsymbol{L}_{k,\mathrm{d}} (\boldsymbol{L}_{k,\mathrm{d}}^{t-1} \boldsymbol{x}_k)$, a one-step shifting of the $(t-1)$-shift result; likewise for $\boldsymbol{L}_{k,\mathrm{u}}^t \boldsymbol{x}_k$. A SCF linearly combines such multi-step simplicial shiftings based on lower and upper adjacencies. Thus, the output $\boldsymbol{H}_k \boldsymbol{x}_k$ is localized in T_d-hop lower and T_u-hop upper k-simplicial neighborhoods (Yang et al., 2022b). SCCNNs preserve such *intra-simplicial locality* as the elementwise nonlinearity does not alter the information locality, shown in Figs. 2b and 2c.

A SCCNN takes the data on k- and $(k \pm 1)$-simplices at layer $l-1$ to compute $\boldsymbol{x}_k^l$, causing interactions between k-simplices and their (co)faces when all SCFs are identity. In turn, $\boldsymbol{x}_{k-1}^{l-1}$ contains information on $(k-2)$-simplices from layer $l-2$. Likewise for $\boldsymbol{x}_{k+1}^{l-1}$, thus, $\boldsymbol{x}_k^l$ also contains information up to $(k \pm 2)$-simplices if $L \geq 2$, because $\boldsymbol{B}_k \sigma(\boldsymbol{B}_{k+1}) \neq \boldsymbol{0}$. Accordingly, this *inter-simplicial locality* extends to the whole SC if $L \geq K$, unlike linear filters in a SC where the locality happens up to the adjacent simplices (Schaub et al., 2021; Isufi & Yang, 2022), which limits its expressive power. This locality is further coupled with the intra-locality through three SCFs such that a node not only interacts with the edges incident to it and direct triangles including it, but also edges and triangles further hops away which contribute to the neighboring nodes, as shown in Fig. 2d.

B.2 Complexity

In a SCCNN layer for computing $\boldsymbol{x}_k^l$, there are $2+T_\mathrm{d}+T_\mathrm{u}$ filter coefficients for the SCF $\boldsymbol{H}_k^l$, and $1+T_\mathrm{d}$ and $1+T_\mathrm{u}$ for $\boldsymbol{H}_{k,\mathrm{d}}^l$ and $\boldsymbol{H}_{k,\mathrm{u}}^l$, respectively, which gives the parameter complexity of order $\mathcal{O}(T_\mathrm{d}+T_\mathrm{u})$. This complexity will increase by $F_l F_{l-1}$ fold for the multi-feature case, and likewise for the computational complexity. Given the inputs $\{\boldsymbol{x}_{k-1}^{l-1}, \boldsymbol{x}_k^{l-1}, \boldsymbol{x}_{k+1}^{l-1}\}$, we discuss the computation complexity of $\boldsymbol{x}_k^l$ in Eq. (4).

First, consider the SCF operation $\boldsymbol{H}_k^l \boldsymbol{x}_k^{l-1}$. As discussed in the localities, it is a composition of T_d-step lower and T_u-step upper simplicial shiftings. Each simplicial shifting has a computational complexity of order $\mathcal{O}(n_k m_k)$ dependent on the number of neighbors m_k where n_k is the number of k-simplices. Thus, this operation has a complexity of order $\mathcal{O}(n_k m_k (T_\mathrm{d}+T_\mathrm{u}))$.

Second, consider the lower SCF operation $\boldsymbol{H}_{k,\mathrm{d}}^l \boldsymbol{B}_k^\top \boldsymbol{x}_{k-1}^{l-1}$. As incidence matrix $\boldsymbol{B}_k$ is sparse, it has $n_k(k+1)$ nonzero entries as each k-simplex has $k+1$ faces. This leads to a complexity of order $O(n_k k)$ for operation $\boldsymbol{B}_k^\top \boldsymbol{x}_{k-1}^{l-1}$. Followed by a lower SCF operation, i.e., a T_d-step lower simplicial shifting, thus, a complexity of order $\mathcal{O}(kn_k + n_k m_k T_\mathrm{d})$ is needed.

Third, consider the upper SCF operation $\boldsymbol{H}_{k,\mathrm{u}}^l \boldsymbol{B}_{k+1} \boldsymbol{x}_{k+1}^{l-1}$. Likewise, incidence matrix $\boldsymbol{B}_{k+1}$ has $n_{k+1}(k+2)$ nonzero entries. This leads to a complexity of order $\mathcal{O}(n_{k+1}k)$ for the projection operation $\boldsymbol{B}_{k+1}\boldsymbol{x}_{k+1}^{l-1}$. Followed by an upper SCF operation, i.e., a T_u-step upper simplicial shifting, thus, a complexity of order $\mathcal{O}(kn_{k+1} + n_k m_k T_\mathrm{u})$ is needed.

Finally, we have a computational complexity of order $\mathcal{O}(k(n_k+n_{k+1}) + N_k M_k (T_\mathrm{d}+T_\mathrm{u}))$ in total.

Remark 27. The lower SCF operation $\boldsymbol{H}_{k,\mathrm{d}}^l \boldsymbol{B}_k^\top \boldsymbol{x}_{k-1}^{l-1}$ can be further reduced if $n_{k-1} \ll n_k$. Note that we have

$$\boldsymbol{H}_{k,\mathrm{d}}^l \boldsymbol{B}_k^\top \boldsymbol{x}_{k-1}^{l-1} = \sum_{t=0}^{T_\mathrm{d}} w_{k,\mathrm{d},t}'^l \boldsymbol{L}_{k,\mathrm{d}}^t \boldsymbol{B}_k^\top \boldsymbol{x}_{k-1}^{l-1} = \boldsymbol{B}_k^\top \sum_{t=0}^{T_\mathrm{d}} w_{k,\mathrm{d},t}'^l \boldsymbol{L}_{k-1,\mathrm{u}}^t \boldsymbol{x}_{k-1}^{l-1}, \tag{19}$$

where the second equality comes from that $\boldsymbol{L}_{k,\mathrm{d}}\boldsymbol{B}_k^\top = \boldsymbol{B}_k^\top \boldsymbol{B}_k \boldsymbol{B}_k^\top = \boldsymbol{B}_k^\top \boldsymbol{L}_{k-1,\mathrm{u}}$, $\boldsymbol{L}_{k,\mathrm{d}}^2 \boldsymbol{B}_k^\top = (\boldsymbol{B}_k^\top \boldsymbol{B}_k)(\boldsymbol{B}_k^\top \boldsymbol{B}_k)\boldsymbol{B}_k^\top = \boldsymbol{B}_k^\top (\boldsymbol{B}_k \boldsymbol{B}_k^\top)(\boldsymbol{B}_k \boldsymbol{B}_k^\top) = \boldsymbol{B}_k^\top \boldsymbol{L}_{k-1,\mathrm{u}}$ and likewise for general t. Using the RHS of Eq. (19) where the simplicial shifting is performed in the $(k-1)$-simplicial space, we have a complexity of order $\mathcal{O}(kn_k + n_{k-1}m_{k-1}T_\mathrm{d})$. Similarly, we have

$$\boldsymbol{H}_{k,\mathrm{u}}^l \boldsymbol{B}_{k+1} \boldsymbol{x}_{k+1}^{l-1} = \sum_{t=0}^{T_\mathrm{u}} w_{k,\mathrm{u},t}'^l \boldsymbol{L}_{k,\mathrm{u}}^t \boldsymbol{B}_{k+1} \boldsymbol{x}_{k+1}^{l-1} = \boldsymbol{B}_{k+1} \sum_{t=0}^{T_\mathrm{u}} w_{k,\mathrm{u},t}'^l \boldsymbol{L}_{k+1,\mathrm{d}}^t \boldsymbol{x}_{k+1}^{l-1} \tag{20}$$

where the simplicial shifting is performed in the $(k+1)$-simplicial space. If it follows that $n_{k+1} \ll n_k$, we have a smaller complexity of $\mathcal{O}(kn_{k+1} + n_{k+1}m_{k+1}T_\mathrm{u})$ by using the RHS of Eq. (20).

B.3 Symmetries of SCs and simplicial data, Equivariance of SCCNNs

Permutation symmetry of SCs. There exists a permutation group P_{n_k} for each set $\mathcal{S}^k$ in a SC of order K. For $K=0$, this gives the graph permutation group. We can combine these groups for different simplex orders by a group product to form a larger permutation group $P = \times_k P_{n_k}$, which is a symmetry group of SCs and simplicial data, assuming vertices in each simplex are consistently ordered. That is, we have, for $p=(p_0,p_1,\ldots,p_K) \in P$, $[p\cdot \boldsymbol{L}_k]_{ij} = [\boldsymbol{L}_k]_{p_k^{-1}(i)p_k^{-1}(j)}$, $[p\cdot \boldsymbol{B}_k]_{ij} = [\boldsymbol{B}_k]_{p_{k-1}^{-1}(i)p_k^{-1}(j)}$, and $[p\cdot \boldsymbol{x}_k]_i = [\boldsymbol{x}_k]_{p_k^{-1}(i)}$. This permutation symmetry of SCs gives us the freedom to list simplices in any order.

Orientation symmetry of simplicial data. The orientation of a simplex is an equivalence class that two orientations are equivalent if they differ by an even permutation Lim (2020); Munkres (2018). Thus, for a simplex $s_i^k = \{i_0,\ldots,i_k\}$ with $k>0$, we have an *orientation symmetry* group $O_{k,i} = \{o_{k,i}^+, o_{k,i}^-\}$ by a group homomorphism which maps all the even permutations of $\{i_0,\ldots,i_k\}$ to the identity element $o_{k,i}^+$ and all the odd permutations to the reverse operation $o_{k,i}^-$.

We can further combine the orientation groups of all simplices in a SC as $O = \times_{i,k} O_{k,i}$ by using a group product. This however is not a symmetry group of an oriented SC because $o_{k,i}^- \cdot \boldsymbol{L}_k$ changes the signs of

$\boldsymbol{L}_k$ elements in ith column and row, and $o^-_{k,i} \cdot \boldsymbol{B}_k$ changes the ith row, resulting in a different SC topology. Instead, it is a symmetry group of the data space, due to its alternating nature w.r.t. simplices. For $o \in O$ we have $[o \cdot \boldsymbol{x}_k]_i = o_{k,i} \cdot f_k(s^k_i) = f_k(o^{-1}_{k,i} \cdot s^k_i)$, i.e., $[\boldsymbol{x}_k]_i$ remains unchanged w.r.t. the changed orientation of s^k_i. This gives us the freedom to choose reference orientations of simplices when working with simplicial data.

Theorem 28 (Permutation Equivariance)**.** *A SCCNN in Eq. (4) is P-equivariant. For all $p \in P$, we have* $p \cdot \mathrm{SCCNN}_k : \{p_{k-1} \cdot \boldsymbol{x}_{k-1}, p_k \cdot \boldsymbol{x}_k, p_{k+1} \cdot \boldsymbol{x}_{k+1}\} \to p_k \boldsymbol{x}_k$.

Theorem 29 (Orientation Equivariance)**.** *A SCCNN in Eq. (4) is O-equivariant if $\sigma(\cdot)$ is odd. For all $o \in O$, we have* $o \cdot \mathrm{SCCNN}_k : \{o_{k-1} \cdot \boldsymbol{x}_{k-1}, o_k \cdot \boldsymbol{x}_k, o_{k+1} \cdot \boldsymbol{x}_{k+1}\} \to o_k \cdot \boldsymbol{x}_k$.

Proof. (informal) Both the permutation group and orientation group have linear matrix representations. By following the same procedure in (Bodnar et al., 2021b, Appendix D) or Roddenberry et al. (2021), we can prove the equivariance. □

B.4 Diffusion process on SCs

Diffusion process on graphs can be generalized to SCs to characterize the evolution of simplicial data over the SC, in analogy to data diffusion on nodes Anand et al. (2022); Ziegler et al. (2022); Grady & Polimeni (2010). Here we provide an informal treatment of how discretizing diffusion equations on SCs can give resemblances of simplicial shifting layers. Consider diffusion equation and its Euler discretization with a unit time step

$$\dot{\boldsymbol{x}_k}(t) = -\boldsymbol{L}_k \boldsymbol{x}_k(t), \text{ Euler step: } \boldsymbol{x}_k(t+1) = \boldsymbol{x}_k(t) - \boldsymbol{L}_k \boldsymbol{x}_k(t) = (\boldsymbol{I} - \boldsymbol{L}_k)\boldsymbol{x}_k(t) \tag{21}$$

with an initial condition $\boldsymbol{x}_k(t) = \boldsymbol{x}^0_k$. The solution of this diffusion is $\boldsymbol{x}_k(t) = \exp(-\boldsymbol{L}_k t)\boldsymbol{x}^0_k$. As the time increases, the simplicial data reaches to a steady state $\dot{\boldsymbol{x}_k}(t) = \boldsymbol{0}$, which lies in the harmonic space $\ker(\boldsymbol{L}_k)$. The simplicial shifting layer resembles this Euler step with a weight and nonlinearity when viewing the time step as layer index. Thus, a NN composed of simplicial shifting layers can suffer from oversmoothing on SCs, giving outputs with decreasing Dirichlet energies as the number of layers increases.

Now let us consider the case where the two Laplacians have different coefficients

$$\dot{\boldsymbol{x}_k}(t) = -\boldsymbol{L}_{k,\mathrm{d}} \boldsymbol{x}_k(t) - \gamma \boldsymbol{L}_{k,\mathrm{u}} \boldsymbol{x}_k(t), \text{ Euler step: } \boldsymbol{x}_k(t) = (\boldsymbol{I} - \boldsymbol{L}_{k,\mathrm{d}} - \gamma \boldsymbol{L}_{k,\mathrm{u}})\boldsymbol{x}_k(t). \tag{22}$$

The steady state of this diffusion equation follows $(\boldsymbol{L}_{k,\mathrm{d}} + \gamma \boldsymbol{L}_{k,\mathrm{u}})\boldsymbol{x}_k(t) = \boldsymbol{0}$, where $\boldsymbol{x}_k(t)$ would be in the kernal space of $\boldsymbol{L}_k$ still. However, before reaching this state, when the time increases, $\boldsymbol{x}_k(t)$ would primarily approach to the kernel of $\boldsymbol{B}^\top_{k+1}$ if $\gamma \gg 1$, in which the lower part of the Dirichlet energy remains, i.e., the decrease of $D(\boldsymbol{x}(t))$ slows down.

When accounting for inter-simplicial couplings, consider there are nontrivial $\boldsymbol{x}_{k-1}$ and $\boldsymbol{x}_{k+1}$ and the diffusion equation becomes

$$\dot{\boldsymbol{x}_k}(t) = -\boldsymbol{L}_k \boldsymbol{x}_k(t) + \boldsymbol{B}^\top_k \boldsymbol{x}_{k-1} + \boldsymbol{B}_{k+1} \boldsymbol{x}_{k+1}, \tag{23}$$

which has source terms $\boldsymbol{B}^\top_k \boldsymbol{x}_{k-1} + \boldsymbol{B}_{k+1}\boldsymbol{x}_{k+1}$. Consider a steady state $\dot{\boldsymbol{x}_k} = 0$. We have $\boldsymbol{L}_k \boldsymbol{x}_k(t) = \boldsymbol{x}_{k,\mathrm{d}} + \boldsymbol{x}_{k,\mathrm{u}}$, where $\boldsymbol{x}_k$ is not in the kernel space of $\boldsymbol{L}_k$. The Euler discretization gives

$$\boldsymbol{x}_k(t+1) = (\boldsymbol{I} - \boldsymbol{L}_k)\boldsymbol{x}_k(t) + \boldsymbol{x}_{k,\mathrm{d}} + \boldsymbol{x}_{k,\mathrm{u}}. \tag{24}$$

The layer in Bunch et al. (2020) $\boldsymbol{x}^{l+1}_k = w_0(\boldsymbol{I} - \boldsymbol{L}_k)\boldsymbol{x}^l_k + w_1 \boldsymbol{x}_{k,\mathrm{d}} + w_2 \boldsymbol{x}_{k,\mathrm{u}}$ is a weighted variant of above step when viewing time steps as layers.

C Related works

We first compare SCCNN with other architectures on if they respect the three principles in Section 3 in Table 5. We then describe how the SCCNN in Eq. (17) generalize other NNs on graphs and SCs in Table 6. For simplicity, we use $\boldsymbol{Y}$ and $\boldsymbol{X}$ to denote the output and input, respectively, without the index l. Note that for GNNs, $\boldsymbol{L}_{0,\mathrm{d}}$ is not defined.

Table 5: Comparisons between SCCNN and other architectures on if they respect the three principles.

Methods	Scheme	P1	P2	P3
MPSN Bodnar et al. (2021b)	message-passing	yes	yes	no, only direct neighborhoods
Eq. (11) of MPSN, or Bunch et al. (2020)	convolutional	no	yes	no, only direct neighborhoods
Eq. (27) of MPSN	convolutional	yes	yes	no, only direct neighborhoods
SNN Ebli et al. (2020)	convolutional	no	no	yes
PSNN Roddenberry et al. (2021)	convolutional	yes	no	no, only direct neighborhoods
SCNN Yang et al. (2022a)	convolutional	yes	no	yes
SCCNN	convolutional	yes	yes	yes

Table 6: SCCNNs generalize other convolutional architectures on SCs.

Methods	Parameters (n.d. denotes "not defined")
Ebli et al. (2020)	$w^l_{k,\mathrm{d},t} = w^l_{k,\mathrm{u},t}, \boldsymbol{H}^l_{k,\mathrm{d}}, \boldsymbol{H}^l_{k,\mathrm{u}}$ n.d.
Roddenberry et al. (2021)	$T_\mathrm{d} = T_\mathrm{u} = 1, \boldsymbol{H}^l_{k,\mathrm{d}}, \boldsymbol{H}^l_{k,\mathrm{u}}$ n.d.
Yang et al. (2022a)	$\boldsymbol{H}^l_{k,\mathrm{d}}, \boldsymbol{H}^l_{k,\mathrm{u}}$ n.d.
Bunch et al. (2020)	$T_\mathrm{d} = T_\mathrm{u} = 1, \boldsymbol{H}^l_{k,\mathrm{d}} = \boldsymbol{H}^l_{k,\mathrm{u}} = \boldsymbol{I}$
Bodnar et al. (2021b)	$T_\mathrm{d} = T_\mathrm{u} = 1, \boldsymbol{H}^l_{k,\mathrm{d}} = \boldsymbol{H}^l_{k,\mathrm{u}} = \boldsymbol{I}$

Gama et al. (2020a) proposed to build a GNN layer with the form

$$\boldsymbol{Y}_0 = \sigma\left(\sum_{t=0}^{T_\mathrm{u}} \boldsymbol{L}_0^t \boldsymbol{X}_0 \boldsymbol{W}_{0,\mathrm{u},t}\right) \tag{25}$$

where the convolution step is performed via a graph filter (Sandryhaila & Moura, 2013; 2014; Gama et al., 2019a; 2020b). This GNN can be easily built as a special SCCNN without contributions from edges. Furthermore, Defferrard et al. (2016) considered a fast implementation of this GNN via a Chebyshev polynomial, while Wu et al. (2019) simplified this by setting $\boldsymbol{W}_{0,t,\mathrm{u}}$ as zeros for $t < T_\mathrm{u}$. Kipf & Welling (2017) further simplified this by setting $T_\mathrm{u} = 1$, namely, GCN.

Yang et al. (2022a) proposed a simplicial convolutional neural network (SCNN) to learn from k-simplicial signals

$$\boldsymbol{Y}_k = \sigma\left(\sum_{t=0}^{T_\mathrm{d}} \boldsymbol{L}_{k,\mathrm{d}}^t \boldsymbol{X}_k \boldsymbol{W}_{k,\mathrm{d},t} + \sum_{t=0}^{T_\mathrm{u}} \boldsymbol{L}_{k,\mathrm{u}}^t \boldsymbol{X}_k \boldsymbol{W}_{k,\mathrm{u},t}\right) \tag{26}$$

where the linear operation is also defined as a simplicial convolution filter in Yang et al. (2022b). This is a special SCCNN with a focus on one simplex level without taking into the lower and upper contributions consideration. The simplicial neural network (SNN) of Ebli et al. (2020) did not differentiate the lower and the upper convolutions with a form of $\boldsymbol{Y}_k = \sigma(\sum_{t=0}^{T} \boldsymbol{L}_k^t \boldsymbol{X}_k \boldsymbol{W}_{k,t})$, which leads to a joint processing in the gradient and curl subspaces as analyzed in Section 4.

While Roddenberry et al. (2021) proposed an architecture (referred to as PSNN)of a particular form of Eq. (26) with $T_\mathrm{d} = T_\mathrm{u} = 1$, performing only a one-step simplicial shifting Eq. (18). Keros et al. (2022) also performs a one-step simplicial shifting but with an inverted Hodge Laplacian to localize the homology group in an SC. An attention mechanism was added to both SCNNs and PSNNs by Giusti et al. (2022) and Goh et al. (2022), respectively. Battiloro et al. (2023) added the attention mechanism to SCCNNs.

To account for the information from adjacent simplices, Bunch et al. (2020) proposed a simplicial 2-complex CNN (S2CCNN)

$$\begin{aligned}
\boldsymbol{Y}_0 &= \sigma\big(\boldsymbol{L}_0 \boldsymbol{X}_0 \boldsymbol{W}_{0,\mathrm{u},1} + \boldsymbol{B}_1 \boldsymbol{X}_1 \boldsymbol{W}'_{0,\mathrm{u},0}\big) \\
\boldsymbol{Y}_1 &= \sigma\big(\boldsymbol{B}_1^\top \boldsymbol{X}_0 \boldsymbol{W}_{1,\mathrm{d},0} + \boldsymbol{L}_1 \boldsymbol{X}_1 \boldsymbol{W}_{1,1} + \boldsymbol{B}_2 \boldsymbol{X}_2 \boldsymbol{W}'_{1,\mathrm{u},0}\big) \\
\boldsymbol{Y}_2 &= \sigma\big(\boldsymbol{B}_2^\top \boldsymbol{X}_1 \boldsymbol{W}_{2,\mathrm{d},0} + \boldsymbol{L}_{2,\mathrm{u}} \boldsymbol{X}_2 \boldsymbol{W}_{2,\mathrm{u},1}\big)
\end{aligned} \tag{27}$$

which is limited to SCs of order two. Note that instead of Hodge Laplacians, simplicial adjacency matrices with self-loops are used in Bunch et al. (2020), which encode equivalent information as setting all filter orders in SCCNNs as one. It is a particular form of the SCCNN where the SCF is a one-step simplicial shifting operation without differentiating the lower and upper shifting, and the lower and upper contributions are simply added, not convolved or shifted by lower and upper SCFs. That is, Bunch et al. (2020) can be obtained from Eq. (4) by setting lower and upper SCFs as identity, $\boldsymbol{H}_{k,\mathrm{d}} = \boldsymbol{H}_{k,\mathrm{u}} = \boldsymbol{I}$, and setting $w_{k,\mathrm{d},t} = w_{k,\mathrm{u},t}$ and $T_{\mathrm{d}} = T_{\mathrm{u}} = 1$ for the SCF $\boldsymbol{H}_k$. The convolution in Yang et al. (2022c, Eq. 3) is the same as Bunch et al. (2020) though it was performed in a block matrix fashion.

The combination of graph shifting and edge shifting in Chen et al. (2022b) can be again seen as a special S2CCNN, where the implementation was performed in a block matrix fashion. Bodnar et al. (2021b) proposed a message passing scheme which collects information from one-hop simplicial neighbors and direct faces and cofaces as Bunch et al. (2020) and Yang et al. (2022c), but replacing the one-step shifting and projections from (co)faces by some learnable functions. The same message passing was applied for simplicial representation learning by Hajij et al. (2021).

Lastly, there are works on signal processing and NNs on cell complexes. For example, Sardellitti et al. (2021); Roddenberry et al. (2022) generalized the signal processing techniques from SCs to cell complexes, Bodnar et al. (2021a); Hajij et al. (2020) performed message passing on cell complexes as in SCs and Hajij et al. (2022) added the attention mechanism. Cell complexes are a more general model compared to SCs, where k-cells compared to k-simplices contain any shapes homeomorphic to a k-dimensional closed balls in Euclidean space, e.g., a filled polygon is a 2-cell while only triangles are 2-simplices. We refer to Hansen & Ghrist (2019) for a more formal definition of cell complexes. Despite cell complexes are more powerful to model real-world higher-order structures, SCCNNs can be easily generalized to cell complexes by considering any k-cells instead of only k-simplices in the algebraic representations, and the theoretical analysis in this paper can be adapted to cell complexes as well.

D Proofs for Section 3

D.1 Dirichlet energy minimization perspective

Hodge Laplacian smoothing. We can find the gradient of problem Eq. (7) as $\frac{\partial D}{\partial \boldsymbol{x}_k} = \boldsymbol{B}_k^\top \boldsymbol{B}_k \boldsymbol{x}_k + \gamma \boldsymbol{B}_{k+1} \boldsymbol{B}_{k+1}^\top \boldsymbol{x}_k$, thus, a gradient descent step follows as Eq. (7) with a step size η.

Proof of Proposition 5. Consider $\eta = 1$.

$$\begin{aligned} D(\boldsymbol{x}_k^{l+1}) &= w_0^2 \|\boldsymbol{B}_k(\boldsymbol{I} - \boldsymbol{L}_{k,\mathrm{d}} - \gamma \boldsymbol{L}_{k,\mathrm{u}})\boldsymbol{x}_k^l\|_2^2 + w_0^2 \|\boldsymbol{B}_{k+1}^\top(\boldsymbol{I} - \boldsymbol{L}_{k,\mathrm{d}} - \gamma \boldsymbol{L}_{k,\mathrm{u}})\boldsymbol{x}_k^l\|_2^2 \\ &= w_0^2 \|(\boldsymbol{I} - \boldsymbol{L}_{k-1,\mathrm{u}})\boldsymbol{B}_k \boldsymbol{x}_k^l\|_2^2 + w_0^2 \|(\boldsymbol{I} - \gamma \boldsymbol{L}_{k+1,\mathrm{d}})\boldsymbol{B}_{k+1}^\top \boldsymbol{x}_k^l\|_2^2 \\ &\leq w_0^2 \|(\boldsymbol{I} - \boldsymbol{L}_{k-1,\mathrm{u}})\|_2^2 \|\boldsymbol{B}_k \boldsymbol{x}_k^l\|_2^2 + w_0^2 \|(\boldsymbol{I} - \gamma \boldsymbol{L}_{k+1,\mathrm{d}})\|_2^2 \|\boldsymbol{B}_{k+1}^\top \boldsymbol{x}_k^l\|_2^2 \end{aligned} \tag{28}$$

which follows from triangle inequality. By definition, we have $\|\boldsymbol{I} - \boldsymbol{L}_{k-1,\mathrm{u}}\|_2^2 = \|\boldsymbol{I} - \boldsymbol{L}_{k,\mathrm{d}}\|_2^2$ and $\|\boldsymbol{I} - \boldsymbol{L}_{k,\mathrm{u}}\|_2^2 = \|\boldsymbol{I} - \boldsymbol{L}_{k+1,\mathrm{d}}\|_2^2$. Also, we have $\|\boldsymbol{I} - \boldsymbol{L}_k\|_2^2 = \max\{\|\boldsymbol{I} - \boldsymbol{L}_{k,\mathrm{d}}\|_2^2, \|\boldsymbol{I} - \boldsymbol{L}_{k,\mathrm{u}}\|_2^2\}$ Thus, we have $D(\boldsymbol{x}_k^{l+1}) \leq w_0^2 \|\boldsymbol{I} - \boldsymbol{L}_k\|_2^2 D(\boldsymbol{x}_k^l)$ when $\gamma = 1$. When $w_0^2 \|\boldsymbol{I} - \boldsymbol{L}_k\|_2^2 < 1$, Dirichlet energy $D(\boldsymbol{x}_k^{l+1})$ will exponentially decrease as l increases. □

When $\gamma \neq 1$, from Eq. (28), we have $D(\boldsymbol{x}_k^{l+1}) = D_{\mathrm{d}}(\boldsymbol{x}_k^{l+1}) + D_{\mathrm{u}}(\boldsymbol{x}_k^{l+1})$, which follows

$$D_{\mathrm{d}}(\boldsymbol{x}_k^{l+1}) \leq w_0^2 \|(\boldsymbol{I} - \boldsymbol{L}_{k,\mathrm{d}})\|_2^2 D_{\mathrm{d}}(\boldsymbol{x}_k^l) \text{ and } D_{\mathrm{u}}(\boldsymbol{x}_k^{l+1}) \leq w_0^2 \|(\boldsymbol{I} - \gamma \boldsymbol{L}_{k,\mathrm{u}})\|_2^2 D_{\mathrm{u}}(\boldsymbol{x}_k^l) \tag{29}$$

When $\gamma = 1$, the oversmoothing condition is $\|\boldsymbol{I} - \boldsymbol{L}_k\|_2^2 = \max\{\|\boldsymbol{I} - \boldsymbol{L}_{k,\mathrm{d}}\|_2^2, \|\boldsymbol{I} - \boldsymbol{L}_{k,\mathrm{u}}\|_2^2\} < \frac{1}{w_0^2}$. If $\|\boldsymbol{I} - \boldsymbol{L}_k\|_2^2 = \|\boldsymbol{I} - \boldsymbol{L}_{k,\mathrm{d}}\|_2^2$, under the oversmoothing condition, by not restricting γ to be 1, $w_0^2 \|(\boldsymbol{I} - \gamma \boldsymbol{L}_{k,\mathrm{u}})\|_2^2$ can be larger than 1 depending on the choice, which means $D_{\mathrm{u}}(\boldsymbol{x}_k^l)$ does not necessarily decrease, so does not $D(\boldsymbol{x}_k^l)$.

Hodge Laplacian smoothing with sources. The gradient of the objective in Eq. (8) is given by $\boldsymbol{L}_k\boldsymbol{x}_k^l - \boldsymbol{B}_k^\top \boldsymbol{x}_{k-1} - \boldsymbol{B}_{k+1}\boldsymbol{x}_{k+1}$, which gives the gradient descent update in Eq. (8) with a step size η.

Consider the layer in Bunch et al. (2020) $\boldsymbol{x}_k^{l+1} = w_0(\boldsymbol{I} - \boldsymbol{L}_k)\boldsymbol{x}_k^l + w_1\boldsymbol{x}_{k,\mathrm{d}} + w_2\boldsymbol{x}_{k,\mathrm{u}}$ with some weights. By triangle inequality, we have $D(\boldsymbol{x}_k^{l+1}) \leq w_0^2\|\boldsymbol{I} - \boldsymbol{L}_k\|_2^2 D(\boldsymbol{x}_k^l) + w_1^2\lambda_{\max}(\boldsymbol{L}_{k,\mathrm{d}})\|\boldsymbol{x}_{k,\mathrm{d}}\|_2^2 + w_2^2\lambda_{\max}(\boldsymbol{L}_{k,\mathrm{u}})\|\boldsymbol{x}_{k,\mathrm{u}}\|_2^2$. If the weight w_0 is small enough following the condition in Proposition 5, the contribution from the projections, controled by weights w_1 and w_2, can compromise the decrease by w_0, maintaining the Dirichlet energy.

E Proofs for Section 4

E.1 The SCF is Hodge-invariant in Proposition 12

Proof. We first give the following lemma.

Lemma 30. *Any finite set of eigenfunctions of a linear operator spans an invariant subspace.*

Then, the proof follows from Lemma 30 and Proposition 10. □

E.2 A derivation of the spectral frequency response in Eq. (11)

SFT of $\boldsymbol{x}_k$. First, the SFT of $\boldsymbol{x}_k$ is given by $\tilde{\boldsymbol{x}}_k = [\tilde{\boldsymbol{x}}_{k,\mathrm{H}}^\top, \tilde{\boldsymbol{x}}_{k,\mathrm{G}}^\top, \tilde{\boldsymbol{x}}_{k,\mathrm{C}}^\top]^\top$ with the *harmonic embedding* $\tilde{\boldsymbol{x}}_{k,\mathrm{H}} = \boldsymbol{U}_{k,\mathrm{H}}^\top\boldsymbol{x}_k = \boldsymbol{U}_{k,\mathrm{H}}^\top\boldsymbol{x}_{k,\mathrm{H}}$ in the zero frequencies, the *gradient embedding* $\tilde{\boldsymbol{x}}_{k,\mathrm{G}} = \boldsymbol{U}_{k,\mathrm{G}}^\top\boldsymbol{x}_k = \boldsymbol{U}_{k,\mathrm{G}}^\top\boldsymbol{x}_{k,\mathrm{G}}$ in the gradient frequencies, and the *curl embedding* $\tilde{\boldsymbol{x}}_{k,\mathrm{C}} = \boldsymbol{U}_{k,\mathrm{C}}^\top\boldsymbol{x}_k = \boldsymbol{U}_{k,\mathrm{C}}^\top\boldsymbol{x}_{k,\mathrm{C}}$ in the curl frequencies.

SFT of $\boldsymbol{H}_k\boldsymbol{x}_k$. By diagonalizing an SCF $\boldsymbol{H}_k$ with $\boldsymbol{U}_k$, we have

$$\boldsymbol{H}_k\boldsymbol{x}_k = \boldsymbol{U}_k\widetilde{\boldsymbol{H}}_k\boldsymbol{U}_k^\top\boldsymbol{x}_k = \boldsymbol{U}_k(\tilde{\boldsymbol{h}}_k \odot \tilde{\boldsymbol{x}}_k) \tag{30}$$

where $\widetilde{\boldsymbol{H}}_k = \mathrm{diag}(\tilde{\boldsymbol{h}}_k)$. Here, $\tilde{\boldsymbol{h}}_k = [\tilde{\boldsymbol{h}}_{k,\mathrm{H}}^\top, \tilde{\boldsymbol{h}}_{k,\mathrm{G}}^\top, \tilde{\boldsymbol{h}}_{k,\mathrm{C}}^\top]^\top$ is the *frequency response*, given by

$$\begin{cases} \textit{harmonic response}: \tilde{\boldsymbol{h}}_{k,\mathrm{H}} = (w_{k,\mathrm{d},0} + w_{k,\mathrm{u},0})\mathbf{1}, \\ \textit{gradient response}: \tilde{\boldsymbol{h}}_{k,\mathrm{G}} = \sum_{t=0}^{T_\mathrm{d}} w_{k,\mathrm{d},t}\boldsymbol{\lambda}_{k,\mathrm{G}}^{\odot t} + w_{k,\mathrm{u},0}\mathbf{1}, \\ \textit{curl response}: \tilde{\boldsymbol{h}}_{k,\mathrm{C}} = \sum_{t=0}^{T_\mathrm{u}} w_{k,\mathrm{u},t}\boldsymbol{\lambda}_{k,\mathrm{C}}^{\odot t} + w_{k,\mathrm{d},0}\mathbf{1}, \end{cases}$$

with $(\cdot)^{\odot t}$ the elementwise t-th power of a vector. Thus, we can express $\tilde{\boldsymbol{h}}_k \odot \tilde{\boldsymbol{x}}_k$ as

$$[(\tilde{\boldsymbol{h}}_{k,\mathrm{H}} \odot \tilde{\boldsymbol{x}}_{k,\mathrm{H}})^\top, (\tilde{\boldsymbol{h}}_{k,\mathrm{G}} \odot \tilde{\boldsymbol{x}}_{k,\mathrm{G}})^\top, (\tilde{\boldsymbol{h}}_{k,\mathrm{C}} \odot \tilde{\boldsymbol{x}}_{k,\mathrm{C}})^\top]^\top. \tag{31}$$

SFT of projections. Second, the lower projection $\boldsymbol{x}_{k,\mathrm{d}} \in \mathrm{im}(\boldsymbol{B}_k^\top)$ has only a nonzero gradient embedding $\tilde{\boldsymbol{x}}_{k,\mathrm{d}} = \boldsymbol{U}_{k,\mathrm{G}}^\top\boldsymbol{x}_{k,\mathrm{d}}$. Likewise, the upper projection $\boldsymbol{x}_{k,\mathrm{u}} \in \mathrm{im}(\boldsymbol{B}_{k+1})$ contains only a nonzero curl embedding $\tilde{\boldsymbol{x}}_{k,\mathrm{u}} = \boldsymbol{U}_{k,\mathrm{C}}^\top\boldsymbol{x}_{k,\mathrm{u}}$. The lower SCF $\boldsymbol{H}_{k,\mathrm{d}}$ has $\tilde{\boldsymbol{h}}_{k,\mathrm{d}} = \sum_{t=0}^{T_\mathrm{d}} w'_{k,\mathrm{d},t}\boldsymbol{\lambda}_{k,\mathrm{G}}^{\odot t}$ as the frequency response that modulates the gradient embedding of $\boldsymbol{x}_{k,\mathrm{d}}$ and the upper SCF $\boldsymbol{H}_{k,\mathrm{u}}$ has $\tilde{\boldsymbol{h}}_{k,\mathrm{u}} = \sum_{t=0}^{T_\mathrm{u}} w'_{k,\mathrm{u},t}\boldsymbol{\lambda}_{k,\mathrm{C}}^{\odot t}$ as the frequency response that modulates the curl embedding of $\boldsymbol{x}_{k,\mathrm{u}}$.

SFT of $\boldsymbol{y}_k$. For the output $\boldsymbol{y}_k = \boldsymbol{H}_{k,\mathrm{d}}\boldsymbol{x}_{k,\mathrm{d}} + \boldsymbol{H}_k\boldsymbol{x}_k + \boldsymbol{H}_{k,\mathrm{u}}\boldsymbol{x}_{k,\mathrm{u}}$, we have

$$\begin{cases} \tilde{\boldsymbol{y}}_{k,\mathrm{H}} = \tilde{\boldsymbol{h}}_{k,\mathrm{H}} \odot \tilde{\boldsymbol{x}}_{k,\mathrm{H}}, \\ \tilde{\boldsymbol{y}}_{k,\mathrm{G}} = \tilde{\boldsymbol{h}}_{k,\mathrm{d}} \odot \tilde{\boldsymbol{x}}_{k,\mathrm{d}} + \tilde{\boldsymbol{h}}_{k,\mathrm{G}} \odot \tilde{\boldsymbol{x}}_{k,\mathrm{G}}, \\ \tilde{\boldsymbol{y}}_{k,\mathrm{C}} = \tilde{\boldsymbol{h}}_{k,\mathrm{C}} \odot \tilde{\boldsymbol{x}}_{k,\mathrm{C}} + \tilde{\boldsymbol{h}}_{k,\mathrm{u}} \odot \tilde{\boldsymbol{x}}_{k,\mathrm{u}}. \end{cases} \tag{32}$$

E.3 Expressive power in Proposition 13

Proof. From the Cayley-Hamilton theorem Horn & Johnson (2012), we know that an analytical function $f(\boldsymbol{A})$ of a matrix $\boldsymbol{A}$ can be expressed as a matrix polynomial of degree at most its minimal polynomial degree, which equals to the number of distinct eigenvalues if $\boldsymbol{A}$ is positive semi-definite.

Consider an analytical function $\boldsymbol{G}_{k,\mathrm{d}}$ of $\boldsymbol{L}_{k,\mathrm{d}}$, defined on the spectrum of $\boldsymbol{L}_{k,\mathrm{d}}$ via analytical function $g_{k,\mathrm{G}}(\lambda)$ where λ is in the set of zero and the gradient frequencies. Then, $\boldsymbol{G}_{k,\mathrm{d}}$ can be implemented by a matrix polynomial of $\boldsymbol{L}_{k,\mathrm{d}}$ of order up to $n_{k,\mathrm{G}}$ where $n_{k,\mathrm{G}}$ is the number of nonzero eigenvalues of $\boldsymbol{L}_{k,\mathrm{d}}$, i.e., the number of distinct gradient frequencies. Likewise, any analytical function $\boldsymbol{G}_{k,\mathrm{u}}$ of $\boldsymbol{L}_{k,\mathrm{u}}$ can be implemented by a matrix polynomial of $\boldsymbol{L}_{k,\mathrm{u}}$ of order up to $n_{k,\mathrm{C}}$, which is the number of nonzero eigenvalues of $\boldsymbol{L}_{k,\mathrm{u}}$, i.e., the number of distinct curl frequencies.

Thus, as of the matrix polynomial definition of SCFs in a SCCNN, the expressive power of $\boldsymbol{H}_{k,\mathrm{d}}\boldsymbol{x}_{k,\mathrm{d}} + \boldsymbol{H}_k\boldsymbol{x}_k + \boldsymbol{H}_{k,\mathrm{u}}\boldsymbol{x}_{k,\mathrm{u}}$ is at most $\boldsymbol{G}'_{k,\mathrm{d}}\boldsymbol{x}_{k,\mathrm{d}} + (\boldsymbol{G}_{k,\mathrm{d}} + \boldsymbol{G}_{k,\mathrm{u}})\boldsymbol{x}_k + \boldsymbol{G}'_{k,\mathrm{u}}\boldsymbol{x}_{k,\mathrm{u}}$, when the matrix polynomial orders (convolution orders) follow $T_{k,\mathrm{d}} = T'_{k,\mathrm{d}} = n_{k,\mathrm{G}}$ and $T_{k,\mathrm{u}} = T'_{k,\mathrm{u}} = n_{k,\mathrm{C}}$. □

E.4 Hodge-aware of SCCNN in Theorem 14

Proof. Consider a linear mapping $T : V \to V$. An invariant subspace W of T has the property that all vectors $\boldsymbol{v} \in W$ are transformed by T into vectors also contained in W, i.e., $\boldsymbol{v} \in W \implies T(\boldsymbol{v}) \in W$. For an input $\boldsymbol{x} \in \mathrm{im}(\boldsymbol{B}_k^\top)$, the output $\boldsymbol{H}_k\boldsymbol{x}$ is in $\mathrm{im}(\boldsymbol{B}_k^\top)$ too, because of

$$\boldsymbol{H}_k\boldsymbol{x} = \sum_t \boldsymbol{L}_{k,\mathrm{d}}^t\boldsymbol{x} + \sum_t \boldsymbol{L}_{k,\mathrm{u}}^t\boldsymbol{x} = \sum_t \boldsymbol{L}_{k,\mathrm{d}}^t\boldsymbol{x} \in \mathrm{im}(\boldsymbol{B}_k^\top) \tag{33}$$

where the second equality comes from the orthogonality between $\mathrm{im}(\boldsymbol{B}_k^\top)$ and $\mathrm{im}(\boldsymbol{B}_{k+1})$. Similarly, we can show that for $\boldsymbol{x} \in \mathrm{im}(\boldsymbol{B}_{k+1})$, the output $\boldsymbol{H}_k\boldsymbol{x} \in \mathrm{im}(\boldsymbol{B}_{k+1})$; for $\boldsymbol{x} \in \ker(\boldsymbol{L}_k)$, the output $\boldsymbol{H}_k\boldsymbol{x} \in \ker(\boldsymbol{L}_k)$. This essentially says the three subspaces of the Hodge decomposition are invariant with respect to the SCF $\boldsymbol{H}_k$. Likewise, the gradient space is invariant with respect to the lower SCF $\boldsymbol{H}_{k,\mathrm{d}}$, which says any lower projection remains in the gradient space after passed by $\boldsymbol{H}_{k,\mathrm{d}}$; and the curl space is invariant with respect to the upper SCF $\boldsymbol{H}_{k,\mathrm{u}}$.

Lastly, through the spectral relation in Eq. (11), the learning operator $\boldsymbol{H}_k$ in the gradient space is controlled by the learnable weights $\{w_{k,\mathrm{d},t}\}$, which is independent of the learnable weights $\{w_{k,\mathrm{u},t}\}$, associated to the learning of $\boldsymbol{H}_k$ in the curl space. Likewise, the lower SCF learns in the gradient space as well but with another set of learnable weights $\{w'_{k,\mathrm{d},t}\}$, and the upper SCF learns in the curl space with learnable weights $\{w'_{k,\mathrm{u},t}\}$. From the spectral expressive power, we see that above four independent learning in the two subspaces can be as expressive as any analytical functions of the corresponding frequencies (spectrum). This concludes the independent and expressive learning in the gradient and curl spaces. □

F Proofs for Section 5

We first give the formulation of SCCNNs on weighted SCs, then we proceed the stability proof.

F.1 SCCNN on weighted SCs

A weighted SC can be defined through specifying the weights of simplices. We give the definition of a commonly used weighted SC with weighted Hodge Laplacians in Grady & Polimeni (2010); Horak & Jost (2013).

Definition 31 (Weighted SC and Hodge Laplacians)**.** In an oriented and weighted SC, we have diagonal weighting matrices $\boldsymbol{M}_k$ with $[\boldsymbol{M}]_{ii}$ measuring the weight of ith k-simplex. A weighted kth Hodge Laplacian is given by

$$\tilde{\boldsymbol{L}}_k = \tilde{\boldsymbol{L}}_{k,\mathrm{d}} + \tilde{\boldsymbol{L}}_{k,\mathrm{u}} = \boldsymbol{M}_k\boldsymbol{B}_k^\top\boldsymbol{M}_{k-1}^{-1}\boldsymbol{B}_k + \boldsymbol{B}_{k+1}\boldsymbol{M}_{k+1}\boldsymbol{B}_{k+1}^\top\boldsymbol{M}_k^{-1}. \tag{34}$$

where $\boldsymbol{L}_{k,\mathrm{d}}$ and $\boldsymbol{L}_{k,\mathrm{u}}$ are the weighted lower and upper Laplacians. A symmetric version follows $\boldsymbol{L}_k^s = \boldsymbol{M}_k^{-1/2}\boldsymbol{L}_k\boldsymbol{M}_k^{1/2}$, and likewise, we have $\boldsymbol{L}_{k,\mathrm{d}}^s = \boldsymbol{M}_k^{1/2}\boldsymbol{B}_k^\top\boldsymbol{M}_{k-1}^{-1}\boldsymbol{B}_k\boldsymbol{M}_k^{1/2}$ and $\boldsymbol{L}_{k,\mathrm{u}}^s = \boldsymbol{M}_k^{-1/2}\boldsymbol{B}_{k+1}\boldsymbol{M}_{k+1}\boldsymbol{B}_{k+1}^\top\boldsymbol{M}_k^{-1/2}$, with the weighted incidence matrix is $\boldsymbol{M}_{k-1}^{-1/2}\boldsymbol{B}_k\boldsymbol{M}_k^{1/2}$ (Horak & Jost, 2013; Guglielmi et al., 2023; Schaub et al., 2020).

SCCNNs in weighted SC. The SCCNN layer defined in a weighted SC is of form

$$\boldsymbol{x}_k^l = \sigma(\boldsymbol{H}_{k,\mathrm{d}}^l\boldsymbol{R}_{k,\mathrm{d}}\boldsymbol{x}_{k-1}^{l-1} + \boldsymbol{H}_k^l\boldsymbol{x}_k^{l-1} + \boldsymbol{H}_{k,\mathrm{u}}^l\boldsymbol{R}_{k,\mathrm{u}}\boldsymbol{x}_{k+1}^{l-1}) \tag{35}$$

where the three SCFs are defined based on the weighted Laplacians Eq. (34), and the lower and upper contributions $\boldsymbol{x}_{k,\mathrm{d}}^l$ and $\boldsymbol{x}_{k,\mathrm{u}}^l$ are obtained via projection matrices $\boldsymbol{R}_{k,\mathrm{d}} \in \mathbb{R}^{n_k \times n_{k-1}}$ and $\boldsymbol{R}_{k,\mathrm{u}} \in \mathbb{R}^{n_k \times n_{k+1}}$, instead of $\boldsymbol{B}_k^\top$ and $\boldsymbol{B}_{k+1}$. For example, Bunch et al. (2020) considered $\boldsymbol{R}_{1,\mathrm{d}} = \boldsymbol{M}_1 \boldsymbol{B}_1^\top \boldsymbol{M}_0^{-1}$ and $\boldsymbol{R}_{1,\mathrm{u}} = \boldsymbol{B}_2 \boldsymbol{M}_2$.

F.2 Proof of Stability of SCCNNs in Theorem 24

For a SCCNN in Eq. (35) in a weighted SC $\mathcal{S}$, we consider its perturbed version in a perturbed SC $\widehat{\mathcal{S}}$ at layer l, given by

$$\hat{\boldsymbol{x}}_k^l = \sigma(\widehat{\boldsymbol{H}}_{k,\mathrm{d}}^l \widehat{\boldsymbol{R}}_{k,\mathrm{d}} \hat{\boldsymbol{x}}_{k-1}^{l-1} + \widehat{\boldsymbol{H}}_k^l \hat{\boldsymbol{x}}_k^{l-1} + \widehat{\boldsymbol{H}}_{k,\mathrm{u}}^l \widehat{\boldsymbol{R}}_{k,\mathrm{u}} \hat{\boldsymbol{x}}_{k+1}^{l-1}) \tag{36}$$

which is defined based on perturbed Laplacians with the same set of filter coefficients, and the perturbed projection operators following relativ perturbation model.

Given the initial input $\boldsymbol{x}_k^0$ for $k = 0, 1, \ldots, K$, our goal is to upper bound the Euclidean distance between the outputs $\boldsymbol{x}_k^l$ and $\hat{\boldsymbol{x}}_k^l$ for $l = 1, \ldots, L$,

$$\begin{aligned} \|\hat{\boldsymbol{x}}_k^l - \boldsymbol{x}_k^l\|_2 = \|\sigma(&\widehat{\boldsymbol{H}}_{k,\mathrm{d}}^l \widehat{\boldsymbol{R}}_{k,\mathrm{d}} \hat{\boldsymbol{x}}_{k-1}^{l-1} - \boldsymbol{H}_{k,\mathrm{d}}^l \boldsymbol{R}_{k,\mathrm{d}} \boldsymbol{x}_{k-1}^{l-1} \\ &+ \widehat{\boldsymbol{H}}_k^l \hat{\boldsymbol{x}}_k^{l-1} - \boldsymbol{H}_k^l \boldsymbol{x}_k^{l-1} + \widehat{\boldsymbol{H}}_{k,\mathrm{u}}^l \widehat{\boldsymbol{R}}_{k,\mathrm{u}} \hat{\boldsymbol{x}}_{k+1}^{l-1} - \boldsymbol{H}_{k,\mathrm{u}}^l \boldsymbol{R}_{k,\mathrm{u}} \boldsymbol{x}_{k+1}^{l-1})\|_2. \end{aligned} \tag{37}$$

We proceed the proof in two steps: first, we analyze the operator norm $\|\widehat{\boldsymbol{H}}_k^l - \boldsymbol{H}_k^l\|_2$ of a SCF $\boldsymbol{H}_k^l$ and its perturbed version $\widehat{\boldsymbol{H}}_k^l$; then we look for the bound of the output distance for a general L-layer SCCNN. To ease notations, we omit the subscript such that $\|\boldsymbol{A}\| = \max_{\|\boldsymbol{x}\|_2=1} \|\boldsymbol{A}\boldsymbol{x}\|_2$ is the operator norm (spectral radius) of a matrix $\boldsymbol{A}$, and $\|\boldsymbol{x}\|$ is the Euclidean norm of a vector $\boldsymbol{x}$.

In the first step we omit the indices k and l for simplicity since they hold for general k and l. We first give a useful lemma.

Lemma 32. *Given the ith eigenvector $\boldsymbol{u}_i$ of $\boldsymbol{L} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\top$, for lower and upper perturbations $\boldsymbol{E}_\mathrm{d}$ and $\boldsymbol{E}_\mathrm{u}$, we have*

$$\boldsymbol{E}_\mathrm{d}\boldsymbol{u}_i = q_{\mathrm{d}i}\boldsymbol{u}_i + \boldsymbol{E}_1\boldsymbol{u}_i, \quad \boldsymbol{E}_\mathrm{u}\boldsymbol{u}_i = q_{\mathrm{u}i}\boldsymbol{u}_i + \boldsymbol{E}_2\boldsymbol{u}_i \tag{38}$$

with eigendecompositions $\boldsymbol{E}_\mathrm{d} = \boldsymbol{V}_\mathrm{d}\boldsymbol{Q}_\mathrm{d}\boldsymbol{V}_\mathrm{d}^\top$ and $\boldsymbol{E}_\mathrm{u} = \boldsymbol{V}_\mathrm{u}\boldsymbol{Q}_\mathrm{u}\boldsymbol{V}_\mathrm{u}^\top$ where $\boldsymbol{V}_\mathrm{d}$, $\boldsymbol{V}_\mathrm{u}$ collect the eigenvectors and $\boldsymbol{Q}_\mathrm{d}$, $\boldsymbol{Q}_\mathrm{u}$ the eigenvalues. It holds that $\|\boldsymbol{E}_1\| \leq \epsilon_\mathrm{d}\delta_\mathrm{d}$ and $\|\boldsymbol{E}_2\| \leq \epsilon_\mathrm{u}\delta_\mathrm{u}$, with $\delta_\mathrm{d} = (\|\boldsymbol{V}_\mathrm{d} - \boldsymbol{U}\|+1)^2 - 1$ and $\delta_\mathrm{u} = (\|\boldsymbol{V}_\mathrm{u} - \boldsymbol{U}\|+1)^2 - 1$ measuring the eigenvector misalignments.

Proof. We first prove that $\boldsymbol{E}_\mathrm{d}\boldsymbol{u}_i = q_{\mathrm{d}i}\boldsymbol{u}_i + \boldsymbol{E}_1\boldsymbol{u}_i$. The perturbation matrix on the lower Laplacian can be written as $\boldsymbol{E}_\mathrm{d} = \boldsymbol{E}_\mathrm{d}^{'} + \boldsymbol{E}_1$ with $\boldsymbol{E}_\mathrm{d}' = \boldsymbol{U}\boldsymbol{Q}_\mathrm{d}\boldsymbol{U}^\top$ and $\boldsymbol{E}_1 = (\boldsymbol{V}_\mathrm{d} - \boldsymbol{U})\boldsymbol{Q}_\mathrm{d}(\boldsymbol{V}_\mathrm{d} - \boldsymbol{U})^\top + \boldsymbol{U}\boldsymbol{Q}_\mathrm{d}(\boldsymbol{V}_\mathrm{d} - \boldsymbol{U})^\top + (\boldsymbol{V}_\mathrm{d} - \boldsymbol{U})\boldsymbol{Q}_\mathrm{d}\boldsymbol{U}^\top$. For the ith eigenvector $\boldsymbol{u}_i$, we have that

$$\boldsymbol{E}_\mathrm{d}\boldsymbol{u}_i = \boldsymbol{E}_\mathrm{d}^{'}\boldsymbol{u}_i + \boldsymbol{E}_1\boldsymbol{u}_i = q_{\mathrm{d}i}\boldsymbol{u}_i + \boldsymbol{E}_1\boldsymbol{u}_i \tag{39}$$

where the second equality follows from $\boldsymbol{E}_\mathrm{d}'\boldsymbol{u}_i = q_{\mathrm{d}i}\boldsymbol{u}_i$. Since $\|\boldsymbol{E}_\mathrm{d}\| \leq \epsilon_\mathrm{d}$, it follows that $\|\boldsymbol{Q}_\mathrm{d}\| \leq \epsilon_\mathrm{d}$. Then, applying the triangle inequality, we have that

$$\begin{aligned} \|\boldsymbol{E}_1\| \leq& \|(\boldsymbol{V}_\mathrm{d} - \boldsymbol{U})\boldsymbol{Q}_\mathrm{d}(\boldsymbol{V}_\mathrm{d} - \boldsymbol{U})^\top\| + \|\boldsymbol{U}\boldsymbol{Q}_\mathrm{d}(\boldsymbol{V}_\mathrm{d} - \boldsymbol{U})^\top\| + \|(\boldsymbol{V}_\mathrm{d} - \boldsymbol{U})\boldsymbol{Q}_\mathrm{d}\boldsymbol{U}\| \\ \leq& \|\boldsymbol{V}_\mathrm{d} - \boldsymbol{U}\|^2\|\boldsymbol{Q}_\mathrm{d}\| + 2\|\boldsymbol{V}_\mathrm{d} - \boldsymbol{U}\|\|\boldsymbol{Q}_\mathrm{d}\|\|\boldsymbol{U}\| \leq \epsilon_\mathrm{d}\|\boldsymbol{V}_\mathrm{d} - \boldsymbol{U}\|^2 + 2\epsilon_\mathrm{d}\|\boldsymbol{V}_\mathrm{d} - \boldsymbol{U}\| \\ =& \epsilon_\mathrm{d}((\|\boldsymbol{V}_\mathrm{d} - \boldsymbol{U}\|+1)^2 - 1) = \epsilon_\mathrm{d}\delta_\mathrm{d}, \end{aligned} \tag{40}$$

which completes the proof for the lower perturbation matrix. Likewise, we can prove for $\boldsymbol{E}_\mathrm{u}\boldsymbol{u}_i$. □

F.2.1 Step I: Stability of the SCF

Proof. **1. Low-order approximation of $\widehat{\boldsymbol{H}} - \boldsymbol{H}$.** Given a SCF $\boldsymbol{H} = \sum_{t=0}^{T_\mathrm{d}} w_{\mathrm{d},t}\boldsymbol{L}_\mathrm{d}^t + \sum_{t=0}^{T_\mathrm{u}} w_{\mathrm{u},t}\boldsymbol{L}_\mathrm{u}^t$, we denote its perturbed version by $\widehat{\boldsymbol{H}} = \sum_{t=0}^{T_\mathrm{d}} w_{\mathrm{d},t}\widehat{\boldsymbol{L}}_\mathrm{d}^t + \sum_{t=0}^{T_\mathrm{u}} w_{\mathrm{u},t}\widehat{\boldsymbol{L}}_\mathrm{u}^t$, where the filter coefficients are the same. The difference between $\boldsymbol{H}$ and $\widehat{\boldsymbol{H}}$ can be expressed as

$$\widehat{\boldsymbol{H}} - \boldsymbol{H} = \sum_{t=0}^{T_\mathrm{d}} w_{\mathrm{d},t}(\widehat{\boldsymbol{L}}_\mathrm{d}^t - \boldsymbol{L}_\mathrm{d}^t) + \sum_{t=0}^{T_\mathrm{u}} w_{\mathrm{u},t}(\widehat{\boldsymbol{L}}_\mathrm{u}^t - \boldsymbol{L}_\mathrm{u}^t), \tag{41}$$

in which we can compute the first-order Taylor expansion of $\widehat{\boldsymbol{L}}_{\mathrm{d}}^{t}$ as

$$\widehat{\boldsymbol{L}}_{\mathrm{d}}^{t} = (\boldsymbol{L}_{\mathrm{d}} + \boldsymbol{E}_{\mathrm{d}}\boldsymbol{L}_{\mathrm{d}} + \boldsymbol{L}_{\mathrm{d}}\boldsymbol{E}_{\mathrm{d}})^{t} = \boldsymbol{L}_{\mathrm{d}}^{t} + \boldsymbol{D}_{\mathrm{d},t} + \boldsymbol{C}_{\mathrm{d}} \tag{42}$$

with $\boldsymbol{D}_{\mathrm{d},t} := \sum_{r=0}^{t-1}(\boldsymbol{L}_{\mathrm{d}}^{r}\boldsymbol{E}_{\mathrm{d}}\boldsymbol{L}_{\mathrm{d}}^{t-r} + \boldsymbol{L}_{\mathrm{d}}^{r+1}\boldsymbol{E}_{\mathrm{d}}\boldsymbol{L}_{\mathrm{d}}^{t-r-1})$ parameterized by t and $\boldsymbol{C}_{\mathrm{d}}$ following $\|\boldsymbol{C}_{\mathrm{d}}\| \leq \sum_{r=2}^{t}\binom{t}{r}\|\boldsymbol{E}_{\mathrm{d}}\boldsymbol{L}_{\mathrm{d}} + \boldsymbol{L}_{\mathrm{d}}\boldsymbol{E}_{\mathrm{d}}\|^{r}\|\boldsymbol{L}_{\mathrm{d}}\|^{t-r}$. Likewise, we can expand $\widehat{\boldsymbol{L}}_{\mathrm{u}}^{t}$ as

$$\widehat{\boldsymbol{L}}_{\mathrm{u}}^{t} = (\boldsymbol{L}_{\mathrm{u}} + \boldsymbol{E}_{\mathrm{u}}\boldsymbol{L}_{\mathrm{d}} + \boldsymbol{L}_{\mathrm{d}}\boldsymbol{u})^{t} = \boldsymbol{L}_{\mathrm{u}}^{t} + \boldsymbol{D}_{\mathrm{u},t} + \boldsymbol{C}_{\mathrm{u}} \tag{43}$$

with $\boldsymbol{D}_{\mathrm{u},t} := \sum_{r=0}^{t-1}(\boldsymbol{L}_{\mathrm{u}}^{r}\boldsymbol{E}_{\mathrm{u}}\boldsymbol{L}_{\mathrm{u}}^{t-r} + \boldsymbol{L}_{\mathrm{u}}^{r+1}\boldsymbol{E}_{\mathrm{u}}\boldsymbol{L}_{\mathrm{u}}^{t-r-1})$ parameterized by t and $\boldsymbol{C}_{\mathrm{u}}$ following $\|\boldsymbol{C}_{\mathrm{u}}\| \leq \sum_{r=2}^{t}\binom{t}{r}\|\boldsymbol{E}_{\mathrm{u}}\boldsymbol{L}_{\mathrm{u}} + \boldsymbol{L}_{\mathrm{u}}\boldsymbol{E}_{\mathrm{u}}\|^{r}\|\boldsymbol{L}_{\mathrm{u}}\|^{t-r}$. Then, by substituting Eq. (42) and Eq. (43) into Eq. (41), we have

$$\widehat{\boldsymbol{H}} - \boldsymbol{H} = \sum_{t=0}^{T_{\mathrm{d}}} w_{\mathrm{d},t}\boldsymbol{D}_{\mathrm{d},t} + \sum_{t=0}^{T_{\mathrm{u}}} w_{\mathrm{u},t}\boldsymbol{D}_{\mathrm{u},t} + \boldsymbol{F}_{\mathrm{d}} + \boldsymbol{F}_{\mathrm{u}} \tag{44}$$

with negligible terms $\|\boldsymbol{F}_{\mathrm{d}}\| = \mathcal{O}(\|\boldsymbol{E}_{\mathrm{d}}\|^{2})$ and $\|\boldsymbol{F}_{\mathrm{u}}\| = \mathcal{O}(\|\boldsymbol{E}_{\mathrm{u}}\|^{2})$ because perturbations are small and the coefficients of higher-order power terms are the derivatives of analytic functions $\tilde{h}_{\mathrm{G}}(\lambda)$ and $\tilde{h}_{\mathrm{C}}(\lambda)$, which are bounded [cf. Definition 18].

2. Spectrum of $(\widehat{\boldsymbol{H}} - \boldsymbol{H})\boldsymbol{x}$. Consider a simplicial signal $\boldsymbol{x}$ with an SFT $\tilde{\boldsymbol{x}} = \boldsymbol{U}^{\top}\boldsymbol{x} = [\tilde{x}_1, \ldots, \tilde{x}_n]^{\top}$, thus, $\boldsymbol{x} = \sum_{i=1}^{n}\tilde{x}_i\boldsymbol{u}_i$. Then, we study the effect of the difference of the SCFs on a simplicial signal from the spectral perspective via

$$(\widehat{\boldsymbol{H}} - \boldsymbol{H})\boldsymbol{x} = \sum_{i=1}^{n}\tilde{x}_i\sum_{t=0}^{T_{\mathrm{d}}} w_{\mathrm{d},t}\boldsymbol{D}_{\mathrm{d},t}^{t}\boldsymbol{u}_i + \sum_{i=1}^{n}\tilde{x}_i\sum_{t=0}^{T_{\mathrm{d}}} w_{\mathrm{u},t}\boldsymbol{D}_{\mathrm{u},t}^{t}\boldsymbol{u}_i + \boldsymbol{F}_{\mathrm{d}}\boldsymbol{x} + \boldsymbol{F}_{\mathrm{u}}\boldsymbol{x} \tag{45}$$

where we have

$$\boldsymbol{D}_{\mathrm{d},t}^{t}\boldsymbol{u}_i = \sum_{r=0}^{t-1}(\boldsymbol{L}_{\mathrm{d}}^{r}\boldsymbol{E}_{\mathrm{d}}\boldsymbol{L}_{\mathrm{d}}^{t-r} + \boldsymbol{L}_{\mathrm{d}}^{r+1}\boldsymbol{E}_{\mathrm{d}}\boldsymbol{L}_{\mathrm{d}}^{t-r-1})\boldsymbol{u}_i, \text{ and } \boldsymbol{D}_{\mathrm{u},t}^{t}\boldsymbol{u}_i = \sum_{r=0}^{t-1}(\boldsymbol{L}_{\mathrm{u}}^{r}\boldsymbol{E}_{\mathrm{u}}\boldsymbol{L}_{\mathrm{u}}^{t-r} + \boldsymbol{L}_{\mathrm{u}}^{r+1}\boldsymbol{E}_{\mathrm{u}}\boldsymbol{L}_{\mathrm{u}}^{t-r-1})\boldsymbol{u}_i. \tag{46}$$

Since the lower and upper Laplacians admit the eigendecompositions for an eigenvector[2] $\boldsymbol{u}_i$

$$\boldsymbol{L}_{\mathrm{d}}\boldsymbol{u}_i = \lambda_{\mathrm{d}i}\boldsymbol{u}_i, \quad \boldsymbol{L}_{\mathrm{u}}\boldsymbol{u}_i = \lambda_{\mathrm{u}i}\boldsymbol{u}_i, \tag{47}$$

we can express the terms in Eq. (45) as

$$\boldsymbol{L}_{\mathrm{d}}^{r}\boldsymbol{E}_{\mathrm{d}}\boldsymbol{L}_{\mathrm{d}}^{t-r}\boldsymbol{u}_i = \boldsymbol{L}_{\mathrm{d}}^{r}\boldsymbol{E}_{\mathrm{d}}\lambda_{\mathrm{d}i}^{t-r}\boldsymbol{u}_i = \lambda_{\mathrm{d}i}^{t-r}\boldsymbol{L}_{\mathrm{d}}^{r}(q_{\mathrm{d}i}\boldsymbol{u}_i + \boldsymbol{E}_1\boldsymbol{u}_i) = q_{\mathrm{d}i}\lambda_{\mathrm{d}i}^{t}\boldsymbol{u}_i + \lambda_{\mathrm{d}i}^{t-r}\boldsymbol{L}_{\mathrm{d}}^{r}\boldsymbol{E}_1\boldsymbol{u}_i, \tag{48}$$

where the second equality holds from Lemma 32. Thus, we have

$$\boldsymbol{L}_{\mathrm{d}}^{r+1}\boldsymbol{E}_{\mathrm{d}}\boldsymbol{L}_{\mathrm{d}}^{t-r-1}\boldsymbol{u}_i = q_{\mathrm{d}i}\lambda_{\mathrm{d}i}^{t}\boldsymbol{u}_i + \lambda_{\mathrm{d}i}^{t-r-1}\boldsymbol{L}_{\mathrm{d}}^{r+1}\boldsymbol{E}_1\boldsymbol{u}_i. \tag{49}$$

With the results in Eq. (48) and Eq. (49), we can write the first term in Eq. (45) as

$$\begin{aligned}\sum_{i=1}^{n}\tilde{x}_i\sum_{t=0}^{T_{\mathrm{d}}} w_{\mathrm{d},t}\boldsymbol{D}_{\mathrm{d},t}^{t}\boldsymbol{u}_i = &\underbrace{\sum_{i=1}^{n}\tilde{x}_i\sum_{t=0}^{T_{\mathrm{d}}} w_{\mathrm{d},t}\sum_{r=0}^{t-1} 2q_{\mathrm{d}i}\lambda_{\mathrm{d}i}^{t}\boldsymbol{u}_i}_{\text{term 1}} \\ &+ \underbrace{\sum_{i=1}^{n}\tilde{x}_i\sum_{t=0}^{T_{\mathrm{d}}} w_{\mathrm{d},t}\sum_{r=0}^{t-1}(\lambda_{\mathrm{d}i}^{t-r}\boldsymbol{L}_{\mathrm{d}}^{r}\boldsymbol{E}_1\boldsymbol{u}_i + \lambda_{\mathrm{d}i}^{t-r-1}\boldsymbol{L}_{\mathrm{d}}^{r+1}\boldsymbol{E}_1\boldsymbol{u}_i)}_{\text{term 2}}.\end{aligned} \tag{50}$$

[2]Note that they can be jointly diagonalized.

Term 1 can be further expanded as

$$\text{term 1} = 2\sum_{i=1}^{n} \tilde{x}_i q_{\mathrm{d}i} \sum_{t=0}^{T_\mathrm{d}} t w_{\mathrm{d},t} \lambda_{\mathrm{d}i}^t \boldsymbol{u}_i = 2\sum_{i=1}^{n} \tilde{x}_i q_{\mathrm{d}i} \lambda_{\mathrm{d}i} \tilde{h}'_{\mathrm{G}}(\lambda_{\mathrm{d}i}) \boldsymbol{u}_i \tag{51}$$

where we used the fact that $\sum_{t=0}^{T_\mathrm{d}} t w_{\mathrm{d},t} \lambda_{\mathrm{d}i}^t = \lambda_{\mathrm{d}i} \tilde{h}'_{\mathrm{G}}(\lambda_{\mathrm{d}i})$. Using $\boldsymbol{L}_\mathrm{d} = \boldsymbol{U}\boldsymbol{\Lambda}_\mathrm{d}\boldsymbol{U}^\top$ we can write term 2 in Eq. (50) as

$$\text{term 2} = \sum_{i=1}^{n} \tilde{x}_i \boldsymbol{U} \mathrm{diag}(\boldsymbol{g}_{\mathrm{d}i}) \boldsymbol{U}^\top \boldsymbol{E}_1 \boldsymbol{u}_i \tag{52}$$

where $\boldsymbol{g}_{\mathrm{d}i} \in \mathbb{R}^n$ has the jth entry

$$[\boldsymbol{g}_{\mathrm{d}i}]_j = \sum_{t=0}^{T_\mathrm{d}} w_{\mathrm{d},t} \sum_{r=0}^{t-1} \left(\lambda_{\mathrm{d}i}^{t-r} [\boldsymbol{\Lambda}_\mathrm{d}]_j^r + \lambda_{\mathrm{d}i}^{t-r-1} [\boldsymbol{\Lambda}_\mathrm{d}]_j^{r+1} \right) = \begin{cases} 2\lambda_{\mathrm{d}i} \tilde{h}'_{\mathrm{G}}(\lambda_{\mathrm{d}i}) & \text{for } j = i, \\ \frac{\lambda_{\mathrm{d}i} + \lambda_{\mathrm{d}j}}{\lambda_{\mathrm{d}i} - \lambda_{\mathrm{d}j}} (\tilde{h}_{\mathrm{G}}(\lambda_{\mathrm{d}i}) - \tilde{h}_{\mathrm{G}}(\lambda_{\mathrm{d}j})) & \text{for } j \neq i. \end{cases} \tag{53}$$

Now, substituting Eq. (51) and Eq. (52) into Eq. (50), we have

$$\sum_{i=1}^{n} \tilde{x}_i \sum_{t=0}^{T_\mathrm{d}} w_{\mathrm{d},t} \boldsymbol{D}_{\mathrm{d},t}^t \boldsymbol{u}_i = 2\sum_{i=1}^{n} \tilde{x}_i q_{\mathrm{d}i} \lambda_{\mathrm{d}i} \tilde{h}'_{\mathrm{G}}(\lambda_{\mathrm{d}i}) \boldsymbol{u}_i + \sum_{i=1}^{n} \tilde{x}_i \boldsymbol{U} \mathrm{diag}(\boldsymbol{g}_{\mathrm{d}i}) \boldsymbol{U}^\top \boldsymbol{E}_1 \boldsymbol{u}_i. \tag{54}$$

By following the same steps as in Eq. (50)-Eq. (53), we can express also the second term in Eq. (45) as

$$\sum_{i=1}^{n} \tilde{x}_i \sum_{t=0}^{T_\mathrm{d}} w_{\mathrm{u},t} \boldsymbol{D}_{\mathrm{u},t}^t \boldsymbol{u}_i = 2\sum_{i=1}^{n} \tilde{x}_i q_{\mathrm{u}i} \lambda_{\mathrm{u}i} \tilde{h}'_{\mathrm{C}}(\lambda_{\mathrm{u}i}) \boldsymbol{u}_i + \sum_{i=1}^{n} \tilde{x}_i \boldsymbol{U} \mathrm{diag}(\boldsymbol{g}_{\mathrm{u}i}) \boldsymbol{U}^\top \boldsymbol{E}_2 \boldsymbol{u}_i \tag{55}$$

where $\boldsymbol{g}_{\mathrm{u}i} \in \mathbb{R}^n$ is defined as

$$[\boldsymbol{g}_{\mathrm{u}i}]_j = \sum_{t=0}^{T_\mathrm{d}} w_{\mathrm{u},t} \sum_{r=0}^{t-1} \left(\lambda_{\mathrm{u}i}^{t-r} [\boldsymbol{\Lambda}_\mathrm{u}]_j^r + \lambda_{\mathrm{u}i}^{t-r-1} [\boldsymbol{\Lambda}_\mathrm{u}]_j^{r+1} \right) = \begin{cases} 2\lambda_{\mathrm{u}i} \tilde{h}'_{\mathrm{C}}(\lambda_{\mathrm{u}i}) & \text{for } j = i, \\ \frac{\lambda_{\mathrm{u}i} + \lambda_{\mathrm{u}j}}{\lambda_{\mathrm{u}i} - \lambda_{\mathrm{u}j}} (\tilde{h}_{\mathrm{C}}(\lambda_{\mathrm{u}i}) - \tilde{h}_{\mathrm{C}}(\lambda_{\mathrm{u}j})) & \text{for } j \neq i. \end{cases} \tag{56}$$

3. Bound of $\|(\widehat{\boldsymbol{H}} - \boldsymbol{H})\boldsymbol{x}\|$. Now we are ready to bound $\|(\widehat{\boldsymbol{H}} - \boldsymbol{H})\boldsymbol{x}\|$ based on triangle inequality. First, given the small perturbations $\|\boldsymbol{E}_\mathrm{d}\| \leq \epsilon_\mathrm{d}$ and $\|\boldsymbol{E}_\mathrm{u}\| \leq \epsilon_\mathrm{u}$, we have for the last two terms in Eq. (45)

$$\|\boldsymbol{F}_\mathrm{d}\boldsymbol{x}\| \leq \mathcal{O}(\epsilon_\mathrm{d}^2)\|\boldsymbol{x}\|, \text{ and } \|\boldsymbol{F}_\mathrm{u}\boldsymbol{x}\| \leq \mathcal{O}(\epsilon_\mathrm{u}^2)\|\boldsymbol{x}\|. \tag{57}$$

Second, for the first term $\|\sum_{i=1}^n \tilde{x}_i \sum_{t=0}^{T_\mathrm{d}} w_{\mathrm{d},t} \boldsymbol{D}_\mathrm{d}^t \boldsymbol{u}_i\|$ in Eq. (45), we can bound its two terms in Eq. (51) and Eq. (52) as

$$\left\| \sum_{i=1}^{n} \tilde{x}_i \sum_{t=0}^{T_\mathrm{d}} w_{\mathrm{d},t} \boldsymbol{D}_{\mathrm{d},t}^t \boldsymbol{u}_i \right\| \leq \left\| 2\sum_{i=1}^{n} \tilde{x}_i q_{\mathrm{d}i} \lambda_{\mathrm{d}i} \tilde{h}'_{\mathrm{G}}(\lambda_{\mathrm{d}i}) \boldsymbol{u}_i \right\| + \left\| \sum_{i=1}^{n} \tilde{x}_i \boldsymbol{U} \mathrm{diag}(\boldsymbol{g}_{\mathrm{d}i}) \boldsymbol{U}^\top \boldsymbol{E}_1 \boldsymbol{u}_i \right\|. \tag{58}$$

For the first term on the RHS of Eq. (58), we can write

$$\left\| 2\sum_{i=1}^{n} \tilde{x}_i q_{\mathrm{d}i} \lambda_{\mathrm{d}i} \tilde{h}'_{\mathrm{G}}(\lambda_{\mathrm{d}i}) \boldsymbol{u}_i \right\|^2 \leq 4\sum_{i=1}^{n} |\tilde{x}_i|^2 |q_{\mathrm{d}i}|^2 |\lambda_{\mathrm{d}i} \tilde{h}'_{\mathrm{G}}(\lambda_{\mathrm{d}i})|^2 \leq 4\epsilon_d^2 c_\mathrm{d}^2 \|\boldsymbol{x}\|^2, \tag{59}$$

which results from, first, $|q_{\mathrm{d}i}| \leq \epsilon_\mathrm{d} = \|\boldsymbol{E}_\mathrm{d}\|$ since $q_{\mathrm{d}i}$ is an eigenvalue of $\boldsymbol{E}_\mathrm{d}$; second, the integral Lipschitz property of the SCF $|\lambda \tilde{h}'_{\mathrm{G}}(\lambda)| \leq c_\mathrm{d}$; and lastly, the fact that $\sum_{i=1}^n |\tilde{x}_i|^2 = \|\tilde{\boldsymbol{x}}\|^2 = \|\boldsymbol{x}\|^2$ and $\|\boldsymbol{u}_i\|^2 = 1$. We then have

$$\left\| 2\sum_{i=1}^{n} \tilde{x}_i q_{\mathrm{d}i} \lambda_{\mathrm{d}i} \tilde{h}'_{\mathrm{G}}(\lambda_{\mathrm{d}i}) \boldsymbol{u}_i \right\| \leq 2\epsilon_\mathrm{d} c_\mathrm{d} \|\boldsymbol{x}\|. \tag{60}$$

For the second term in RHS of Eq. (58), we have

$$\left\| \sum_{i=1}^{n} \tilde{x}_i \boldsymbol{U} \mathrm{diag}(\boldsymbol{g}_{\mathrm{d}i}) \boldsymbol{U}^\top \boldsymbol{E}_1 \boldsymbol{u}_i \right\| \leq \sum_{i=1}^{n} |\tilde{x}_i| \|\boldsymbol{U} \mathrm{diag}(\boldsymbol{g}_{\mathrm{d}i}) \boldsymbol{U}^\top\| \|\boldsymbol{E}_1\| \|\boldsymbol{u}_i\|, \tag{61}$$

which stems from the triangle inequality. We further have $\|\boldsymbol{U} \mathrm{diag}(\boldsymbol{g}_{\mathrm{d}i}) \boldsymbol{U}^\top\| = \|\mathrm{diag}(\boldsymbol{g}_{\mathrm{d}i})\| \leq 2C_{\mathrm{d}}$ resulting from $\|\boldsymbol{U}\| = 1$ and the c_{d}-integral Lipschitz of $\tilde{h}_{\mathrm{G}}(\lambda)$ [cf. Definition 18]. Moreover, it follows that $\|\boldsymbol{E}_1\| \leq \epsilon_{\mathrm{d}} \delta_{\mathrm{d}}$ from Lemma 32, which results in

$$\left\| \sum_{i=1}^{n} \tilde{x}_i \boldsymbol{U} \mathrm{diag}(\boldsymbol{g}_{\mathrm{d}i}) \boldsymbol{U}^\top \boldsymbol{E}_1 \boldsymbol{u}_i \right\| \leq 2C_{\mathrm{d}} \epsilon_{\mathrm{d}} \delta_{\mathrm{d}} \sqrt{n} \|\boldsymbol{x}\| \tag{62}$$

where we use that $\sum_{i=1}^{n} |\tilde{x}_i| = \|\tilde{\boldsymbol{x}}\|_1 \leq \sqrt{n} \|\tilde{\boldsymbol{x}}\| = \sqrt{n} \|\boldsymbol{x}\|$. By combining Eq. (59) and Eq. (62), we have

$$\left\| \sum_{i=1}^{n} \tilde{x}_i \sum_{t=0}^{T_{\mathrm{d}}} w_{\mathrm{d},t} \boldsymbol{D}_{\mathrm{d},t}^{t} \boldsymbol{u}_i \right\| \leq 2\epsilon_{\mathrm{d}} c_{\mathrm{d}} \|\boldsymbol{x}\| + 2C_{\mathrm{d}} \epsilon_{\mathrm{d}} \delta_{\mathrm{d}} \sqrt{n} \|\boldsymbol{x}\|. \tag{63}$$

Analogously, we can show that

$$\left\| \sum_{i=1}^{n} \tilde{x}_i \sum_{t=0}^{T_{\mathrm{d}}} w_{\mathrm{u},t} \boldsymbol{D}_{\mathrm{u},t}^{t} \boldsymbol{u}_i \right\| \leq 2\epsilon_{\mathrm{u}} c_{\mathrm{u}} \|\boldsymbol{x}\| + 2C_{\mathrm{u}} \epsilon_{\mathrm{u}} \delta_{\mathrm{u}} \sqrt{n} \|\boldsymbol{x}\|. \tag{64}$$

Now by combining Eq. (57), Eq. (63) and Eq. (64), we can bound $\|(\widehat{\boldsymbol{H}} - \boldsymbol{H})\boldsymbol{x}\|$ as

$$\begin{aligned} \|(\widehat{\boldsymbol{H}} - \boldsymbol{H})\boldsymbol{x}\| \leq\ & 2\epsilon_{\mathrm{d}} c_{\mathrm{d}} \|\boldsymbol{x}\| + 2C_{\mathrm{d}} \epsilon_{\mathrm{d}} \delta_{\mathrm{d}} \sqrt{n} \|\boldsymbol{x}\| + \mathcal{O}(\epsilon_{\mathrm{d}}^2) \|\boldsymbol{x}\| \\ & + 2\epsilon_{\mathrm{u}} c_{\mathrm{u}} \|\boldsymbol{x}\| + 2C_{\mathrm{u}} \epsilon_{\mathrm{u}} \delta_{\mathrm{u}} \sqrt{n} \|\boldsymbol{x}\| + \mathcal{O}(\epsilon_{\mathrm{u}}^2) \|\boldsymbol{x}\|. \end{aligned} \tag{65}$$

By defining $\Delta_{\mathrm{d}} = 2(1 + \delta_{\mathrm{d}} \sqrt{n})$ and $\Delta_{\mathrm{u}} = 2(1 + \delta_{\mathrm{u}} \sqrt{n})$, we can obtain that

$$\|\widehat{\boldsymbol{H}} - \boldsymbol{H}\| \leq c_{\mathrm{d}} \Delta_{\mathrm{d}} \epsilon_{\mathrm{d}} + c_{\mathrm{u}} \Delta_{\mathrm{u}} \epsilon_{\mathrm{u}} + \mathcal{O}(\epsilon_{\mathrm{d}}^2) + \mathcal{O}(\epsilon_{\mathrm{u}}^2). \tag{66}$$

Thus, we have $\|\boldsymbol{H}_k^l - \widehat{\boldsymbol{H}}_k^l\| \leq c_{k,\mathrm{d}} \Delta_{k,\mathrm{d}} \epsilon_{k,\mathrm{d}} + c_{k,\mathrm{u}} \Delta_{k,\mathrm{u}} \epsilon_{k,\mathrm{u}}$ with $\Delta_{k,\mathrm{d}} = 2(1 + \delta_{k,\mathrm{d}} \sqrt{n_k})$ and $\Delta_{k,\mathrm{u}} = 2(1 + \delta_{k,\mathrm{u}} \sqrt{n_k})$ where we ignore the second and higher order terms on $\epsilon_{k,\mathrm{d}}$ and $\epsilon_{k,\mathrm{u}}$. Likewise, we have $\|\boldsymbol{H}_{k,\mathrm{d}}^l - \widehat{\boldsymbol{H}}_{k,\mathrm{d}}^l\| \leq c_{k,\mathrm{d}} \Delta_{k,\mathrm{d}} \epsilon_{k,\mathrm{d}}$ for the lower SCF and $\|\boldsymbol{H}_{k,\mathrm{u}}^l - \widehat{\boldsymbol{H}}_{k,\mathrm{u}}^l\| \leq c_{k,\mathrm{u}} \Delta_{k,\mathrm{u}} \epsilon_{k,\mathrm{u}}$ for the upper SCF. □

F.2.2 Step II: Stability of SCCNNs

Proof. Given the initial input $\boldsymbol{x}_k^0$, the Euclidean distance between $\boldsymbol{x}_k^l$ and $\hat{\boldsymbol{x}}_k^l$ at layer l can be bounded by using triangle inequality and the c_σ-Lipschitz property of $\sigma(\cdot)$ [cf. Assumption 22] as

$$\|\hat{\boldsymbol{x}}_k^l - \boldsymbol{x}_k^l\|_2 \leq c_\sigma (\phi_{k,\mathrm{d}}^l + \phi_k^l + \phi_{k,\mathrm{u}}^l), \tag{67}$$

with

$$\begin{aligned} \phi_{k,\mathrm{d}}^l :=& \|\widehat{\boldsymbol{H}}_{k,\mathrm{d}}^l \widehat{\boldsymbol{R}}_{k,\mathrm{d}} \hat{\boldsymbol{x}}_{k-1}^{l-1} - \boldsymbol{H}_{k,\mathrm{d}}^l \boldsymbol{R}_{k,\mathrm{d}} \boldsymbol{x}_{k-1}^{l-1}\|, \\ \phi_k^l :=& \|\widehat{\boldsymbol{H}}_k^l \hat{\boldsymbol{x}}_k^{l-1} - \boldsymbol{H}_k^l \boldsymbol{x}_k^{l-1}\|, \\ \phi_{k,\mathrm{u}}^l :=& \|\widehat{\boldsymbol{H}}_{k,\mathrm{u}}^l \widehat{\boldsymbol{R}}_{k,\mathrm{u}} \hat{\boldsymbol{x}}_{k+1}^{l-1} - \boldsymbol{H}_{k,\mathrm{u}}^l \boldsymbol{R}_{k,\mathrm{u}} \boldsymbol{x}_{k+1}^{l-1}\|. \end{aligned} \tag{68}$$

We now focus on upper bounding each of the terms.

1. Term ϕ_k^l. By subtracting and adding $\widehat{\boldsymbol{H}}_k^l \boldsymbol{x}_k^{l-1}$ within the norm, and using the triangle inequality, we obtain

$$\begin{aligned} \phi_k^l &\leq \|\widehat{\boldsymbol{H}}_k^l (\hat{\boldsymbol{x}}_k^{l-1} - \boldsymbol{x}_k^{l-1})\| + \|(\widehat{\boldsymbol{H}}_k^l - \boldsymbol{H}_k^l) \boldsymbol{x}_k^{l-1}\| \leq \|\hat{\boldsymbol{x}}_k^{l-1} - \boldsymbol{x}_k^{l-1}\| + \|\widehat{\boldsymbol{H}}_k^l - \boldsymbol{H}_k^l\| \|\boldsymbol{x}_k^{l-1}\| \\ &\leq \|\hat{\boldsymbol{x}}_k^{l-1} - \boldsymbol{x}_k^{l-1}\| + (c_{k,\mathrm{d}} \Delta_{k,\mathrm{d}} \epsilon_{k,\mathrm{d}} + c_{k,\mathrm{u}} \Delta_{k,\mathrm{u}} \epsilon_{k,\mathrm{u}}) \|\boldsymbol{x}_k^{l-1}\| \end{aligned} \tag{69}$$

where we used the SCF stability in Eq. (66) and that all SCFs have a normalized bounded frequency response in Assumption 20. Note that $\widehat{\boldsymbol{H}}_k^l$ is also characterized by $\tilde{h}_{\mathrm{G}}(\lambda)$ with the same set of filter coefficients as $\boldsymbol{H}_k^l$.

2. Term $\phi_{k,\mathrm{d}}^l$ and $\phi_{k,\mathrm{u}}^l$. By subtracting and adding a term $\widehat{\boldsymbol{H}}_{k,\mathrm{d}}^l \widehat{\boldsymbol{R}}_{k,\mathrm{d}} \boldsymbol{x}_{k-1}^{l-1}$ within the norm, we have

$$\begin{aligned}\phi_{k,\mathrm{d}}^l &\leq \|\widehat{\boldsymbol{H}}_{k,\mathrm{d}}^l \widehat{\boldsymbol{R}}_{k,\mathrm{d}}(\hat{\boldsymbol{x}}_{k-1}^{l-1} - \boldsymbol{x}_{k-1}^{l-1})\| + \|(\widehat{\boldsymbol{H}}_{k,\mathrm{d}}^l \widehat{\boldsymbol{R}}_{k,\mathrm{d}} - \boldsymbol{H}_{k,\mathrm{d}}^l \boldsymbol{R}_{k,\mathrm{d}})\boldsymbol{x}_{k-1}^{l-1}\| \\ &\leq \|\widehat{\boldsymbol{R}}_{k,\mathrm{d}}\|\|\hat{\boldsymbol{x}}_{k-1}^{l-1} - \boldsymbol{x}_{k-1}^{l-1}\| + \|\widehat{\boldsymbol{H}}_{k,\mathrm{d}}^l \widehat{\boldsymbol{R}}_{k,\mathrm{d}} - \boldsymbol{H}_{k,\mathrm{d}}^l \boldsymbol{R}_{k,\mathrm{d}}\|\|\boldsymbol{x}_{k-1}^{l-1}\|,\end{aligned} \tag{70}$$

where we used again triangle inequality and $\|\widehat{\boldsymbol{H}}_{k,\mathrm{d}}^l\| \leq 1$ from Assumption 20. For the term $\|\widehat{\boldsymbol{R}}_{k,\mathrm{d}}^l\|$, we have $\|\widehat{\boldsymbol{R}}_{k,\mathrm{d}}^l\| \leq \|\boldsymbol{R}_{k,\mathrm{d}}^l\| + \|\boldsymbol{J}_{k,\mathrm{d}}\|\|\boldsymbol{R}_{k,\mathrm{d}}^l\| \leq r_{k,\mathrm{d}}(1+\boldsymbol{\epsilon}_{k,\mathrm{d}})$ where we used $\|\boldsymbol{R}_{k,\mathrm{d}}^l\| \leq r_{k,\mathrm{d}}$ in Assumption 21 and $\|\boldsymbol{J}_{k,\mathrm{d}}^l\| \leq \boldsymbol{\epsilon}_{k,\mathrm{d}}$. For the second term of RHS in Eq. (70), by adding and subtracting $\widehat{\boldsymbol{H}}_{k,\mathrm{d}}^l \boldsymbol{R}_{k,\mathrm{d}}^l$ we have

$$\begin{aligned}\|\widehat{\boldsymbol{H}}_{k,\mathrm{d}}^l \widehat{\boldsymbol{R}}_{k,\mathrm{d}} - \boldsymbol{H}_{k,\mathrm{d}}^l \boldsymbol{R}_{k,\mathrm{d}}\| &= \|\widehat{\boldsymbol{H}}_{k,\mathrm{d}}^l \widehat{\boldsymbol{R}}_{k,\mathrm{d}} - \widehat{\boldsymbol{H}}_{k,\mathrm{d}}^l \boldsymbol{R}_{k,\mathrm{d}}^l + \widehat{\boldsymbol{H}}_{k,\mathrm{d}}^l \boldsymbol{R}_{k,\mathrm{d}}^l - \boldsymbol{H}_{k,\mathrm{d}}^l \boldsymbol{R}_{k,\mathrm{d}}\| \\ &\leq \|\widehat{\boldsymbol{H}}_{k,\mathrm{d}}^l\|\|\widehat{\boldsymbol{R}}_{k,\mathrm{d}} - \boldsymbol{R}_{k,\mathrm{d}}\| + \|\widehat{\boldsymbol{H}}_{k,\mathrm{d}}^l - \boldsymbol{H}_{k,\mathrm{d}}^l\|\|\boldsymbol{R}_{k,\mathrm{d}}\| \\ &\leq r_{k,\mathrm{d}}\boldsymbol{\epsilon}_{k,\mathrm{d}} + C'_{k,\mathrm{d}}\Delta_{k,\mathrm{d}}\epsilon_{k,\mathrm{d}}r_{k,\mathrm{d}}\end{aligned} \tag{71}$$

where we use the stability result of the lower SCF $\boldsymbol{H}_{k,\mathrm{d}}^l$ in Eq. (66). By substituting Eq. (71) into Eq. (70), we have

$$\phi_{k,\mathrm{d}}^l \leq \hat{r}_{k,\mathrm{d}}\|\hat{\boldsymbol{x}}_{k-1}^{l-1} - \boldsymbol{x}_{k-1}^{l-1}\| + (r_{k,\mathrm{d}}\boldsymbol{\epsilon}_{k,\mathrm{d}} + C'_{k,\mathrm{d}}\Delta_{k,\mathrm{d}}\epsilon_{k,\mathrm{d}}r_{k,\mathrm{d}})\|\boldsymbol{x}_{k-1}^{l-1}\|. \tag{72}$$

By following the same procedure [cf. Eq. (70) and Eq. (71)], we obtain

$$\phi_{k,\mathrm{u}}^l \leq \hat{r}_{k,\mathrm{u}}\|\hat{\boldsymbol{x}}_{k+1}^{l-1} - \boldsymbol{x}_{k+1}^{l-1}\| + (r_{k,\mathrm{u}}\boldsymbol{\epsilon}_{k,\mathrm{u}} + C'_{k,\mathrm{u}}\Delta_{k,\mathrm{u}}\epsilon_{k,\mathrm{u}}r_{k,\mathrm{u}})\|\boldsymbol{x}_{k+1}^{l-1}\|. \tag{73}$$

3. Bound of $\|\hat{\boldsymbol{x}}_k^l - \boldsymbol{x}_k^l\|$. Using the notations $t_k, t_{k,\mathrm{d}}$ and $t_{k,\mathrm{u}}$ in Theorem 24, we then have a set of recursions, for $k = 0, 1, \ldots, K$

$$\begin{aligned}\|\hat{\boldsymbol{x}}_k^l - \boldsymbol{x}_k^l\| \leq& c_\sigma(\hat{r}_{k,\mathrm{d}}\|\hat{\boldsymbol{x}}_{k-1}^{l-1} - \boldsymbol{x}_{k-1}^{l-1}\| + t_{k,\mathrm{d}}\|\boldsymbol{x}_{k-1}^{l-1}\| + \|\hat{\boldsymbol{x}}_k^{l-1} - \boldsymbol{x}_k^{l-1}\| + t_k\|\boldsymbol{x}_k^{l-1}\| \\ &+ \hat{r}_{k,\mathrm{u}}\|\hat{\boldsymbol{x}}_{k+1}^{l-1} - \boldsymbol{x}_{k+1}^{l-1}\| + t_{k,\mathrm{u}}\|\boldsymbol{x}_{k+1}^{l-1}\|).\end{aligned} \tag{74}$$

Define vector $\boldsymbol{b}^l$ as $[\boldsymbol{b}^l]_k = \|\hat{\boldsymbol{x}}_k^l - \boldsymbol{x}_k^l\|$ with $\boldsymbol{b}^0 = \boldsymbol{0}$. Let $\boldsymbol{\beta}^l$ collect the energy of all outputs at layer l, with $[\boldsymbol{\beta}^l]_k := \|\boldsymbol{x}_k^{l-1}\|$. We can express the Euclidean distances of all k-simplicial signal outputs for $k = 0, 1, \ldots, K$, as

$$\boldsymbol{b}^l \preceq c_\sigma \widehat{\boldsymbol{Z}}\boldsymbol{b}^{l-1} + c_\sigma \boldsymbol{T}\boldsymbol{\beta}^{l-1} \tag{75}$$

where $\preceq$ indicates elementwise smaller than or equal, and we have

$$\boldsymbol{T} = \begin{bmatrix} t_0 & t_{0,\mathrm{u}} & & & & \\ t_{1,\mathrm{d}} & t_1 & t_{1,\mathrm{u}} & & & \\ & \ddots & \ddots & \ddots & & \\ & & t_{K-1,\mathrm{d}} & t_{K-1} & t_{K-1,\mathrm{u}} & \\ & & & t_{K,\mathrm{d}} & t_K \end{bmatrix} \text{ and } \widehat{\boldsymbol{Z}} = \begin{bmatrix} 1 & \hat{r}_{0,\mathrm{u}} & & & & \\ \hat{r}_{1,\mathrm{d}} & 1 & \hat{r}_{1,\mathrm{u}} & & & \\ & \ddots & \ddots & \ddots & & \\ & & \hat{r}_{K-1,\mathrm{d}} & 1 & \hat{r}_{K-1,\mathrm{u}} & \\ & & & \hat{r}_{K,\mathrm{d}} & 1 \end{bmatrix}. \tag{76}$$

We are now interested in building a recursion for Eq. (75) for all layers l. We start with term $\boldsymbol{x}_k^l$. Based on its expression in Eq. (35), we bound it as

$$\begin{aligned}\|\boldsymbol{x}_k^l\| &\leq c_\sigma(\|\boldsymbol{H}_{k,\mathrm{d}}^l\|\|\boldsymbol{R}_{k,\mathrm{d}}\|\|\boldsymbol{x}_{k-1}^{l-1}\| + \|\boldsymbol{H}_k^l\|\|\boldsymbol{x}_x^{l-1}\| + \|\boldsymbol{H}_{k,\mathrm{u}}^l\|\|\boldsymbol{R}_{k,\mathrm{u}}\|\|\boldsymbol{x}_{k+1}^{l-1}\|) \\ &\leq c_\sigma(r_{k,\mathrm{d}}\|\boldsymbol{x}_{k-1}^{l-1}\| + \|\boldsymbol{x}_x^{l-1}\| + r_{k,\mathrm{u}}\|\boldsymbol{x}_{k+1}^{l-1}\|),\end{aligned} \tag{77}$$

which holds for $k = 0, 1, \ldots, K$. Thus, it can be expressed in the vector form as $\boldsymbol{\beta}^l \preceq c_\sigma \boldsymbol{Z}\boldsymbol{\beta}^{l-1}$, with

$$\boldsymbol{Z} = \begin{bmatrix} 1 & r_{0,\mathrm{u}} & & & \\ r_{1,\mathrm{d}} & 1 & r_{1,\mathrm{u}} & & \\ & \ddots & \ddots & \ddots & \\ & & r_{K-1,\mathrm{d}} & 1 & r_{K-1,\mathrm{u}} \\ & & & r_{K,\mathrm{d}} & 1 \end{bmatrix}. \tag{78}$$

Similarly, we have $\boldsymbol{\beta}^{l-1} \preceq c_\sigma \boldsymbol{Z}\boldsymbol{\beta}^{l-2}$, leading to $\boldsymbol{\beta}^{l} \preceq c_\sigma^l \boldsymbol{Z}^l\boldsymbol{\beta}^{0}$ with $\boldsymbol{\beta}^0 = \boldsymbol{\beta}$ [cf. Assumption 23]. We can then express the bound Eq. (75) as

$$\boldsymbol{b}^l \preceq c_\sigma \widehat{\boldsymbol{Z}}\boldsymbol{b}^{l-1} + c_\sigma^l \boldsymbol{T}\boldsymbol{Z}^{l-1}\boldsymbol{\beta}. \tag{79}$$

Thus, we have

$$\boldsymbol{b}^0 = \boldsymbol{0},\ \boldsymbol{b}^1 \preceq c_\sigma \boldsymbol{T}\boldsymbol{\beta},\ \boldsymbol{b}^2 \preceq c_\sigma^2(\widehat{\boldsymbol{Z}}\boldsymbol{T}\boldsymbol{\beta} + \boldsymbol{T}\boldsymbol{Z}\boldsymbol{\beta}),\ \boldsymbol{b}^3 \preceq c_\sigma^3(\widehat{\boldsymbol{Z}}^2\boldsymbol{T}\boldsymbol{\beta} + \widehat{\boldsymbol{Z}}\boldsymbol{T}\boldsymbol{Z}\boldsymbol{\beta} + \boldsymbol{T}\boldsymbol{Z}^2\boldsymbol{\beta}),\ \boldsymbol{b}^4 \preceq \ldots, \tag{80}$$

which, inductively, leads to

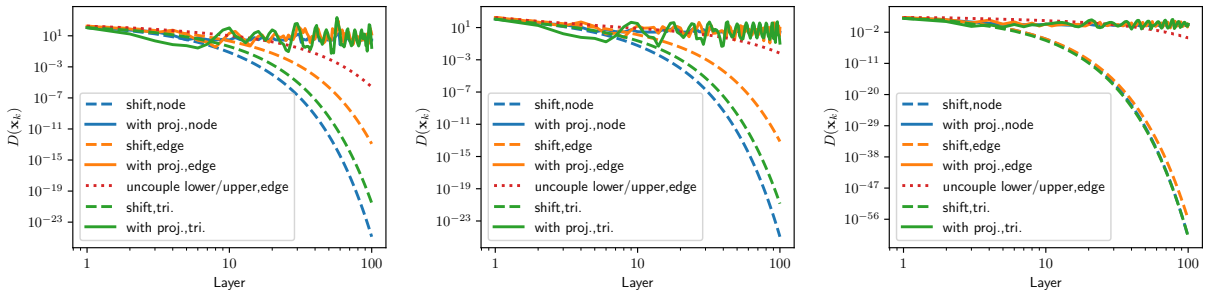
$$\boldsymbol{b}^l \preceq c_\sigma^l \sum_{i=1}^{l} \widehat{\boldsymbol{Z}}^{i-1}\boldsymbol{T}\boldsymbol{Z}^{l-i}\boldsymbol{\beta}. \tag{81}$$

Bt setting $l = L$, we obtain the bound $\boldsymbol{b}^L \preceq \boldsymbol{d} = c_\sigma^L \sum_{l=1}^{L} \widehat{\boldsymbol{Z}}^{l-1}\boldsymbol{T}\boldsymbol{Z}^{L-l}\boldsymbol{\beta}$ in Theorem 24. □

G Experiment details

G.1 Synthetic experiments on Dirichlet energy evolution

We created a synthetic SC with 100 nodes, 241 edges and 135 triangles with the GUDHI toolbox Rouvreau (2015), and we set the initial inputs on three levels of simplices to be random sampled from $\mathcal{U}([-5, 5])$. We then built a SCCNN composed of simplicial shifting layers with weight w_0 and nonlinearities including id, tanh and relu. When the weight follows the condition in Proposition 5, from Fig. 10 (the dashed lines labled as "shift"), we see that the Dirichlet energies of all three outputs exponentially decrease as the number of layers increases. We then uncoupled the lower and upper parts of the Laplacians in the edge space in the shifting layers by setting $\gamma \neq 1$. As shown in Fig. 10 (the dotted lines), the Dirichlet energies of the edge outputs decrease at a slower rate than before. Lastly, we added the inter-simplicial couplings, which overcome the oversmoothing problems, as shown by the solid lines.



(a) identity, $\gamma = 2$ (b) tanh, $\gamma = 2$ (c) relu, $\gamma = 5$

Figure 10: Oversmoothing effects of simplicial shifting and the mitigation effects of uncoupling lower and upper adjacencies and accounting for inter-simplicial couplings.

G.2 Additional details on Forex experiments

In the forex dataset, there are 25 currencies which can be exchanged pairwise at three timestamps. We first represented their exchange rates on the edges and took the logrithm, i.e., $[\boldsymbol{x}_1]_{[i,j]} = \log_{10} r^{i/j} = -[\boldsymbol{x}_1]_{[j,i]}$. Then, the total arbitrage can be computed as the total curl $\boldsymbol{B}_2^\top \boldsymbol{x}_1$.

We considered to recover the exchange rates under three types of settings: 1) random noise following normal distribution such that the signal-to-noise ration is -3dB, which is spread over the whole simplicial spectrum; 2) "curl noise" projected from triangle noise following normal distribution such that the signal-to-noise ration is -3dB, which is distributed only in the curl space; and 3) 50% of the total forex rates are recorded and the other half is not available, set as zero values.

Table 7: Forex results (nmse, arbitrage) and the corresponding hyperparameters.

Methods	Random noise	"Curl noise"	Interpolation
Input	0.119 ± 0.004, 25.19 ± 0.874	0.552 ± 0.027, 122.36 ± 5.90	0.717 ± 0.030, 106.40 ± 0.902
ℓ_2-norm	0.036 ± 0.005, 2.29 ± 0.079	0.050 ± 0.002, 11.12 ± 0.537	0.534 ± 0.043 , 9.67 ± 0.082
SNN	0.11 ± 0.005, 23.24 ± 1.03 $L = 5, F = 64, T = 4$, tanh	0.446 ± 0.017, 86.947 ± 2.197 $L = 6, F = 64, T = 3$, tanh	0.702 ± 0.033, 104.738 ± 1.042 $L = 2, F = 64, T = 1$, tanh
PSNN	0.008 ± 0.001, 0.984 ± 0.17 $L = 6, F = 64$, tanh	0.000 ± 0.000, 0.000 ± 0.000 $L = 5, F = 1$, id	0.009 ± 0.001, 1.128 ± 0.329 $L = 6, F = 64$, tanh
Bunch	0.981 ± 0.0 , 22.912 ± 1.228 —	0.981 ± 0.0, 22.912 ± 1.228 —	0.983 ± 0.005 , 19.887 ± 6.341 —
MPSN	0.039 ± 0.004, 7.748 ± 0.943 $L = 2, F = 64$, id, sum	0.076 ± 0.012, 14.922 ± 2.493 $L = 4, F = 64$, tanh, mean	0.117 ± 0.063, 23.147 ± 11.674 $L = 2, F = 64$, tanh, sum
SCCNN, id	0.027 ± 0.005, 0.000 ± 0.000 $L = 2, F = 16, T_{\mathrm{d}} = 0, T_{\mathrm{u}} = 3$	0.000 ± 0.000, 0.000 ± 0.000 $L = 5, F = 1, T_{\mathrm{d}} = 1, T_{\mathrm{u}} = 1$	0.265 ± 0.036 , 0.000 ± 0.000 $L = 2, F = 16, T_{\mathrm{d}} = 0, T_{\mathrm{u}} = 3$
SCCNN, tanh	0.002 ± 0.000, 0.325 ± 0.082 $L = 6, F = 64, T_{\mathrm{d}} = 5, T_{\mathrm{u}} = 2$	0.000 ± 0.000, 0.003 ± 0.003 $L = 1, F = 64, T_{\mathrm{d}} = 2, T_{\mathrm{u}} = 2$	0.003 ± 0.002, 0.279 ± 0.151 $L = 6, F = 64, T_{\mathrm{d}} = 5, T_{\mathrm{u}} = 1$

First, as a baseline method, we chose ℓ_2 norm of the curl $\boldsymbol{B}_2\boldsymbol{x}_1$ as a regularizer to reduce the total arbitrage, i.e., $\hat{\boldsymbol{x}}_1 = (\boldsymbol{I} + w\boldsymbol{L}_{1,\mathrm{u}})^{-1}\boldsymbol{x}_1$ with a regularization weight $w \in [0, 10]$. For the learning methods, we consider the following hyperparameter ranges: the number of layers to be $L \in \{1, 2, \ldots, 6\}$, the number of intermediate features to be $F \in \{1, 16, 32, 64\}$. For the convolutional methods including SNN Ebli et al. (2020), PSNN Roddenberry et al. (2021), Bunch Bunch et al. (2020) and SCCNN, we considered the intermediate layers with nonlinearities including id and tanh. The convolution orders of SNN and SCCNN are set to be $\{1, 2, \ldots, 5\}$. For the message-passing method, MPSN Bodnar et al. (2021b), we considered the setting from (Bodnar et al., 2021b, Eq. 35) where the sum and mean aggregations are used and each message update function is a two-layer MLP. With these noisy or masked rates as inputs and the clean arbitrage-free rates as outputs, we trained different learning methods at the first timestamp, and validated the hyperparameters at the second timestamp, and tested their performance at the thrid one. During the training of 1000 epochs, a normalized MSE loss function and adam optimizer with a fixed learning rate of 0.001 are used. We run the same experiments for 10 times. Table 7 reports the best results (nmse) and the total arbitrage, together with the hyperparameters.

G.3 Additional details on Simplex prediction

G.3.1 Method in Detail

The method for simplex prediction is generalized from link prediction based on GNNs by Zhang & Chen (2018): For k-simplex prediction, we use an SCCNN in an SC of order up to k to first learn the features of lower-order simplices up to order $k-1$. Then, we concatenate these embedded lower-order simplicial features and input them to a two-layer MLP which predicts if a k-simplex is positive (closed, shall be included in the SC) or negative (open, not included in the SC).

For example, in 2-simplex prediction, consider an SC of order two, which is built based on nodes, edges and (existing positive) triangles. Given the initial inputs on nodes $\boldsymbol{x}_0$ and on edges $\boldsymbol{x}_1$ and zero inputs on triangles $\boldsymbol{x}_2 = \boldsymbol{0}$ since we assume no prior knowledge on triangles, for an open triangle $t = [i, j, k]$, an SCCNN is used to learn features on nodes and edges (denoted by $\boldsymbol{y}$). Then, we input the concatenation of the features on three nodes or three edges to an MLP, i.e., $\mathrm{MLP}_{\mathrm{node}}([\boldsymbol{y}_0]_i \| [\boldsymbol{y}_0]_j \| [\boldsymbol{y}_0]_k)$ or $\mathrm{MLP}_{\mathrm{edge}}([\boldsymbol{y}]_{[i,j]} \| [\boldsymbol{y}]_{[j,k]} \| [\boldsymbol{y}]_{[i,k]})$, to predict if triangle t is positive or negative. A MLP taking both node and edge features is possible, but we keep it on one simplex level for complexity purposes. Similarly, we consider an SCCNN in an SC of order three for 3-simplex prediction, which is followed by an MLP operating on either nodes, edges or triangles.

G.3.2 Data Preprocessing

We consider the data from the Semantic Scholar Open Research Corpus Ammar et al. (2018) to construct a coauthorship complex where nodes are authors and collaborations between k-author are represented by $(k-1)$-simplices. Following the preprocessing in Ebli et al. (2020), we obtain 352 nodes, 1472 edges, 3285 triangles, 5019 tetrahedrons (3-simplices) and a number of other higher-order simplices. The node signal $\boldsymbol{x}_0$, edge flow $\boldsymbol{x}_1$ and triangle flow $\boldsymbol{x}_2$ are the numbers of citations of single author papers and the collaborations of two and three authors, respectively.

For the 2-simplex prediction, we use the collaboration impact (the number of citations) to split the total set of triangles into the positive set $\mathcal{T}_P = \{t|[\boldsymbol{x}_2]_t > 7\}$ containing 1482 closed triangles and the negative set $\mathcal{T}_N = \{t|[\boldsymbol{x}_2]_t \leq 7\}$ containing 1803 open triangles such that we have balanced positive and negative samples. We further split the 80% of the positive triangle set for training, 10% for validation and 10% for testing; likewise for the negative triangle set. Note that in the construction of the SC, i.e., the incidence matrix $\boldsymbol{B}_2$, Hodge Laplacians $\boldsymbol{L}_{1,\mathrm{u}}$ and $\boldsymbol{L}_{2,\mathrm{d}}$, we ought to remove negative triangles in the training set and all triangles in the test set. That is, for 2-simplex prediction, we only make use of the training set of the positive triangles since the negative ones are not in the SC.

Similarly, we prepare the dataset for 3-simplex (tetrahedron) prediction, amounting to the tetradic collaboration prediction. We obtain balanced positive and negative tetrahedron sets based on the citation signal $\boldsymbol{x}_3$. In the construction of $\boldsymbol{B}_3$, $\boldsymbol{L}_{2,\mathrm{u}}$ and $\boldsymbol{L}_{3,\mathrm{d}}$, we again only use the tetrahedrons in the positive training set.

G.3.3 Models

For comparison, we first use heuristic methods proposed in Benson et al. (2018) as baselines to determine if a triangle $t = [i, j, k]$ is closed, namely, 1) Harmonic mean: $s_t = 3/([\boldsymbol{x}_1]_{[i,j]}^{-1} + [\boldsymbol{x}_1]_{[j,k]}^{-1} + [\boldsymbol{x}_1]_{[i,k]}^{-1})$, 2) Geometric mean: $s_t = \lim_{p\to 0}[([\boldsymbol{x}_1]_{[i,j]}^{p} + [\boldsymbol{x}_1]_{[j,k]}^{p} + [\boldsymbol{x}_1]_{[i,k]}^{p})]^{1/p}$, and 3) Arithmetic mean: $s_t = ([\boldsymbol{x}_1]_{[i,j]} + [\boldsymbol{x}_1]_{[j,k]} + [\boldsymbol{x}_1]_{[i,k]})/3$, which compute the triangle weight based on its three faces. Similarly, we generalized these mean methods to compute the weight of a 3-simplex $[i, j, k, m]$ based on the four triangle faces in 3-simplex prediction.

We then consider different learning methods. Specifically, 1) "Bunch" by Bunch et al. (2020) (we also generalized this model to 3-dimension for 3-simplex prediction); 2) Message passing simplicial network ("MPSN") by Bodnar et al. (2021b) which provides a baseline of message passing scheme in comparison to the convolution scheme; 3) Principled SNN ("PSNN") by Roddenberry et al. (2021); 4) SNN by Ebli et al. (2020); 5) SCNN by Yang et al. (2021); 6) GNN by Defferrard et al. (2016); 7) MLP: providing as a baseline for the effect of using inductive models.

For MLP, Bunch, MPSN and our SCCNN, we consider the outputs in the node and edge spaces, respectively, for 2-simplex prediction, which are denoted by a suffix "-Node" or "-Edge". For 3-simplex prediction, the output in the triangle space can be used as well, denoted by a suffix "-Tri.", where we also build SCNNs in both edge and triangle spaces.

G.3.4 Experimental Setup and Hyperparameters

We consider the normalized Hodge Laplacians and incidence matrices, a particular version of the weighted ones Horak & Jost (2013); Grady & Polimeni (2010). Specifically, we use the symmetric version of the normalized random walk Hodge Laplacians in the edge space, proposed by Schaub et al. (2020), which were used in Bunch et al. (2020); Chen et al. (2022a) as well. We generalized the definitions for triangle predictions.

Hyperparameters 1) the number of layers: $L \in \{1, 2, 3, 4, 5\}$; 2) the number of intermediate and output features to be the same as $F \in \{16, 32\}$; 3) the convolution orders for SCCNNs are set to be the same, i.e., $T'_{\mathrm{d}} = T_{\mathrm{d}} = T_{\mathrm{u}} = T'_{\mathrm{u}} = T \in \{1, 2, 3, 4, 5\}$. We do so to avoid the exponential growth of the parameter search space. For GNNs (Defferrard et al., 2016) and SNNs (Ebli et al., 2020), we set the convolution orders to be $T \in \{1, 2, 3, 4, 5\}$ while for SCNNs (Yang et al., 2022a), we allow the lower and upper convolutions to have different orders with $T_{\mathrm{d}}, T_{\mathrm{u}} \in \{1, 2, 3, 4, 5\}$; 4) the nonlinearity in the feature learning phase: LeakyReLU with a negative slope 0.01; 5) MPSN is set as Bodnar et al. (2022); 6) the MLP in the prediction phase: two

layers with a sigmoid nonlinearity. For 2-simplex prediction, the number of the input features for the node features is $3F$, and for the edge features is $3F$. For 3-simplex prediction, the number of the input features for the node features is $4F$, for the edge features is $6F$ and for the triangle features is $4F$ since a 3-simplex has four nodes, six edges and four triangles. The number of the intermediate features is the same as the input features, and that of the output features is one; and, 7) the binary cross entropy loss and the adam optimizer with a learning rate of 0.001 are used; the number of the epochs is 1000 where an early stopping is used. We compute the AUC to compare the performance and run the same experiments for ten times with random data splitting.

G.3.5 Results

In Table 8, we report the best results of each method with the corresponding hyperparameters. Different hyperparameters can lead to similar results, but we report the ones with the *least* complexity. All experiments for simplex predictions were run on a single NVIDIA A40 GPU with 48 GB of memory using CUDA 11.5.

Table 8: 2- *(Left)* and 3-Simplex *(Right)* prediction AUC (%) results.

Methods	AUC	Parameters
Harm. Mean	62.8±2.7	—
Arith. Mean	60.8±3.2	—
Geom. Mean	61.7±3.1	—
MLP-Node	68.5±1.6	$L=1, F=32$
GNN	93.9±1.0	$L=5, F=32, T=2$
SNN-Edge	92.0±1.8	$L=5, F=32, T=5$
PSNN-Edge	95.6±1.3	$L=5, F=32$
SCNN-Edge	96.5±1.5	$L=5, F=32, T_{\mathrm{d}}=5, T_{\mathrm{u}}=2$
Bunch-Node	98.3±0.5	$K=1, L=4, F=32$
MPSN-Node	98.1±0.5	$K=1, L=3, F=32$
SCCNN-Node	98.7±0.5	$K=1, L=2, F=32, T=2$

Methods	AUC	Parameters
Harm. Mean	63.6±1.6	—
Arith. Mean	62.2±1.4	—
Geom. Mean	63.1±1.4	—
MLP-Tri.	69.0±2.2	$L=3, F=32$
GNN	96.6±0.5	$L=5, F=32, T=5$
SNN-Tri.	95.1±1.2	$L=5, F=32, T=5$
PSNN-Tri.	98.1±0.5	$L=5, F=32$
SCNN-Tri.	98.3±0.4	$L=5, F=32, T_{\mathrm{d}}=2, T_{\mathrm{u}}=1$
Bunch-Edge	98.5±0.5	$K=3, L=4, F=16$
MPSN-Edge	99.2±0.3	$K=3, L=3, F=32$
SCCNN-Node	99.4±0.3	$K=3, L=3, F=32, T=3$

G.3.6 Complexity

Table 9: *(Left)* Complexity of the best three methods for 2-simplex prediction. *(Right)* Running time of SCCNN with different layers and convolution orders.

Method	#params.	Running time (seconds per epoch)
SCCNN	24288	0.073
Bunch	21728	0.140
MPSN	84256	0.028

Hyperparams.	$T=2$	$T=5$
$L=2$	0.073	0.082
$L=3$	0.110	0.130
$L=5$	0.192	0.237

Here we report the number of parameters and the running time of SCCNN for 2-simplex prediction on one NVIDIA Quadro K2200 with 4GB memory, compared with the two best alternatives. MPSN, compared to convolutional methods, has three times more parameters, analogous to the comparison between message-passing and graph convolutional NNs. We also report the running time as the layers and convolution orders increase.

G.3.7 Ablation Study

We perform an ablation study to observe the roles of different components in SCCNNs.

SC Order K We investigate the influence of the SC order K. Table 10 reports the 2-simplex prediction results for $K=\{1,2\}$ and the 3-simplex prediction results for $K=\{1,2,3\}$. We observe that for k-simplex prediction, it does not necessarily guarantee a better prediction with a higher-order SC, which further

indicates that a positive simplex could be well encoded by both its faces and other lower-order subsets. For example, in 2-simplex prediction, SC of order one gives better results than SC of order two (similar for Bunch), showing that in this coauthorship complex, triadic collaborations are better encoded by features on nodes than pairwise collaborations. In 3-simplex prediction, SCs of different orders give similar results, showing that tetradic collaborations can be encoded by nodes, as well as by pairwise and triadic collaborations.

Table 10: Prediction results of SCCNNs with different SC order K.

Method	2-Simplex	Parameters
SCCNN-Node	98.7±0.5	$K=1, L=2, F=32, T=2$
SCCNN-Node	98.4±0.5	$K=2, L=2, F=32, T=2$
Bunch-Node	98.3±0.4	$K=1, L=4, F=32$
Bunch-Node	98.0±0.4	$K=2, L=4, F=32$
MPSN-Node	94.5±1.5	$K=1, L=3, F=32$
MPSN-Node	98.1±0.5	$K=2, L=3, F=32$
SCCNN-Edge	97.9±0.9	$K=1, L=3, F=32, T=5$
SCCNN-Edge	95.9±1.0	$K=2, L=5, F=32, T=3$
Bunch-Edge	97.3±1.1	$K=1, L=4, F=32$
Bunch-Edge	94.6±1.2	$K=2, L=4, F=32$
MPSN-Edge	94.1±2.4	$K=1, L=3, F=32$
MPSN-Edge	97.0±1.2	$K=2, L=2, F=16$

Method	3-Simplex	Parameters
SCCNN-Node	99.3±0.3	$K=1, L=2, F=32, T=1$
SCCNN-Node	99.3±0.2	$K=2, L=2, F=32, T=5$
SCCNN-Node	99.4±0.3	$K=3, L=3, F=32, T=3$
MPSN-Node	96.0±1.2	$K=1, L=3, F=32$
MPSN-Node	98.2±0.8	$K=2, L=2, F=32$
SCCNN-Edge	98.9±0.5	$K=1, L=3, F=32, T=5$
SCCNN-Edge	99.2±0.4	$K=2, L=5, F=32, T=5$
SCCNN-Edge	99.0±1.0	$K=3, L=5, F=32, T=5$
MPSN-Edge	96.3±1.1	$K=1, L=3, F=32$
MPSN-Edge	98.3±0.8	$K=2, L=3, F=32$
SCCNN-Tri.	97.9±0.7	$K=2, L=4, F=32, T=4$
SCCNN-Tri.	97.4±0.9	$K=3, L=4, F=32, T=4$
MPSN-Tri.	99.1±0.2	$K=2, L=3, F=32$

Missing Components in SCCNN With a focus on 2-simplex prediction with SCCNN-Node of order one, to avoid overcrowded settings, we study how each component of an SCCNN influences the prediction. We consider the following settings without: 1) "Edge-to-Node", where the projection $\boldsymbol{x}_{0,\mathrm{u}}$ from edge to node is not included, equivalent to GNN; 2) "Node-to-Node", where for node output, we have $\boldsymbol{x}_0^l = \sigma(\boldsymbol{H}_{0,\mathrm{u}}^l \boldsymbol{R}_{1,\mathrm{u}} \boldsymbol{x}_1^{l-1})$; 3) "Node-to-Edge", where the projection $\boldsymbol{x}_{1,\mathrm{d}}$ from node to edge is not included, i.e., we have $\boldsymbol{x}_1^l = \sigma(\boldsymbol{H}_1^l \boldsymbol{x}_1^{l-1})$; and 4) "Edge-to-Edge", where for edge output, we have $\boldsymbol{x}_1^l = \sigma(\boldsymbol{H}_{1,\mathrm{d}}^l \boldsymbol{R}_{1,\mathrm{d}} \boldsymbol{x}_0^{l-1})$.

Table 11: 2-Simplex prediction (SCCNN-Node without certain components or with limited inputs).

Missing Component	AUC	Parameters
—	98.7±0.5	$L=2, F=32, T=2$
Edge-to-Node	93.9±0.8	$L=5, F=32, T=2$
Node-to-Node	98.7±0.4	$L=4, F=32, T=2$
Edge-to-Edge	98.5±1.0	$L=3, F=32, T=3$
Node-to-Edge	98.8±0.3	$L=4, F=32, T=3$

Missing Input	AUC	Parameters
—	98.7±0.5	$L=2, F=32, T=2$
Node input	98.2±0.5	$L=2, F=32, T=4$
Edge input	98.1±0.4	$L=2, F=32, T=3$
Node, Edge inputs	50.0±0.0	—

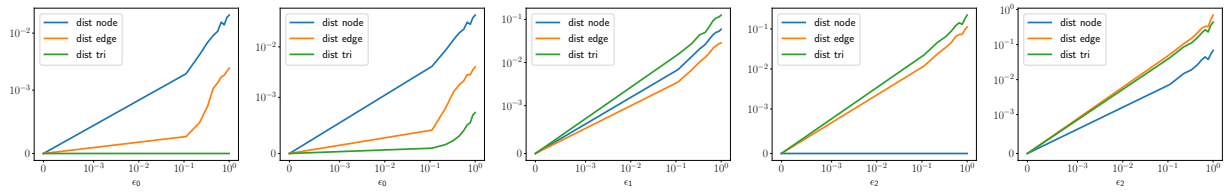
From the results in Table 11 (Left), we see that "No Edge-to-Node", i.e., GNN, gives much worse results as it leverages no information on edges with limited expressive power. For cases with other components missing, a similar performance can be achieved, however, at a cost of the model complexity, with either a higher convolution order or a larger number of layers L, while the latter in turn degrades the stability of the SCCNNs, as discussed in Section 5. SCCNNs with certain inter-simplicial couplings pruned/missing can be powerful as well (this is similarly shown by (Bodnar et al., 2021b, Thm. 6)), but if we did not consider certain component, it comes with a cost of complexity which might degrade the model stability if more number layers are required.

Limited Input We study the influence of limited input data for model SCCNN-Node of order two. Specifically, we consider the input on either nodes or edges is missing. From Table 11, we see that the prediction performance does not deteriorate at a cost of the model complexity (higher convolution orders) when a certain part of the input missing except with full zeros as input. This ability of learning from limited data shows the robustness of SCCNNs.

G.3.8 Stability Analysis

We then perform a stability analysis of SCCNNs. We artificially add perturbations to the normalization matrices when defining the Hogde Laplacians, which resemble the weights of simplices. We consider small perturbations $\boldsymbol{E}_0$ on node weights which is a diagonal matrix following that $\|\boldsymbol{E}_0\| \leq \epsilon_0/2$. We generate its diagonal entries from a uniform distribution $[-\epsilon_0/2, \epsilon_0/2)$ with $\epsilon_0 \in [0, 1]$, which represents one degree of deviation of the node weigths from the true ones. Similarly, perturbations on edge weights and triangle weights are applied to study the stability. In a SCCNN-Node for 2-simplex prediction of $K = 2$, we measure the distance between the simplicial outputs with and without perturbations on nodes, edges, and triangles, i.e., $\|\boldsymbol{x}_k^L - \hat{\boldsymbol{x}}_k^L\|/\|\boldsymbol{x}_k^L\|$, for $k = 0, 1, 2$.

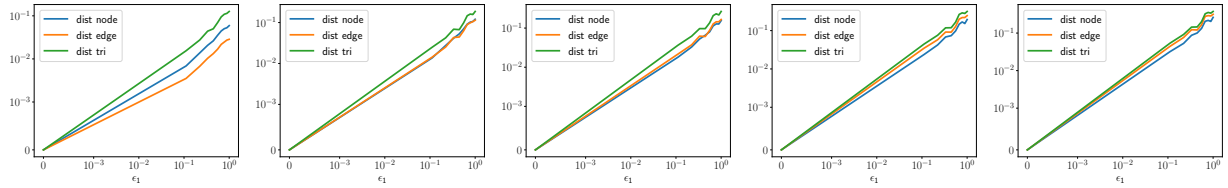
Stability dependence We first show the stability mutual dependence between different simplices in Fig. 11. We see that under perturbation on node weights, the triangle output is not influenced until the number of layers becomes two; likewise, the node output is not influenced by perturbations on triangle weights with a one-layer SCCNN. Also, a one-layer SCCNN under perturbations on edge weights will cause outputs on nodes, edges, triangles perturbed. Lastly, we observe that the same degree of perturbations added to different simplices causes different degrees of instability, owing to the number n_k of k-simplices in the stability bound. Since $N_0 < N_1 < N_2$, the perturbations on node weights cause less instability than those on edge and triangle weights.



(a) Node pert.$L = 1$ (b) Node pert.$L = 2$ (c) Edge pert.$L = 1$ (d) Tri. pert.$L = 1$ (e) Tri. pert.$L = 2$

Figure 11: The stabilities of different simplicial outputs are dependent on each other.

Number of Layers Fig. 12 shows that the stability of SCCNNs degrades as the number of layers increases as studied in Theorem 24. As the NN deepens, the stability deteriorates, which corresponds to our analysis of using shallow layers.



(a) $L = 1$ (b) $L = 2$ (c) $L = 3$ (d) $L = 4$ (e) $L = 5$

Figure 12: The relative difference of SCCNN outputs with and without perturbations in terms of different numbers of layers. We consider perturbations on edge weights.

G.4 Additional details on Trajectory prediction

G.4.1 Problem Formulation

A trajectory of length m can be modeled as a sequence of nodes $[v_0, v_1, \ldots, v_{m-1}]$ in an SC. The task is to predict the next node v_m from the neighbors of v_{m-1}, $\mathcal{N}_{v_{m-1}}$. The algorithm in Roddenberry et al. (2021) first represents the trajectory equivalently as a sequence of oriented edges $[[v_0, v_1], [v_1, v_2], \ldots, [v_{m-2}, v_{m-1}]]$. Then, an edge flow $\boldsymbol{x}_1$ is defined, whose value on an edge e is $[\boldsymbol{x}_1]_e = 1$ if edge e is traversed by the trajectory

in a forward direction, $[\boldsymbol{x}_1]_e = -1$ if edge e is traversed in a backward direction by the trajectory, and $[\boldsymbol{x}_1]_e = 0$, otherwise.

With the trajectory flow $\boldsymbol{x}_1$ as the input, together with zero inputs on the nodes and triangles, an SCCNN of order two is used to generate a representation $\boldsymbol{x}_1^L$ of the trajectory, which is the output on edges. This is followed by a projection step $\boldsymbol{x}_{0,\mathrm{u}}^L = \boldsymbol{B}_1 \boldsymbol{W} \boldsymbol{x}_1^L$, where the output is first passed through a linear transformation via $\boldsymbol{W}$, then projected into the node space via $\boldsymbol{B}_1$. Lastly, a distribution over the candidate nodes $\mathcal{N}_{v_{m-1}}$ is computed via a softmax operation, $\boldsymbol{n}_j = \mathrm{softmax}([\boldsymbol{x}_{0,\mathrm{u}}^L]_j), j \in \mathcal{N}_{v_{m-1}}$. The best candidate is selected as $v_m = \mathrm{argmax}_j \boldsymbol{n}_j$. We refer to Roddenberry & Segarra (2019, Alg. S-2) for more details.

Given that an SCCNN of order two generates outputs also on nodes, we can directly apply the node feature output $\boldsymbol{x}_0^L$ to compute a distribution over the candidate nodes $\mathcal{N}_{v_{m-1}}$ without the projection step. We refer to this as SCCNN-Node, and the method of using the edge features with the projection step as SCCNN-Edge.

G.4.2 Model

In this experiment, we consider the following methods: 1) PSNN by Roddenberry et al. (2021); 2) SNN by Ebli et al. (2020); 3) SCNN by Yang et al. (2022a) where we consider different lower and upper convolution orders $T_\mathrm{d}, T_\mathrm{u}$; and 4) Bunch by Bunch et al. (2020) where we consider both the node features and edge features, namely, Bunch-Node and Bunch-Edge.

Synthetic Data Following the procedure in Schaub et al. (2020), we generate 1000 trajectories as follows. First, we create an SC with two "holes" by uniformly drawing 400 random points in the unit square, and then a Delaunay triangulation is applied to obtain a mesh, followed by the removal of nodes and edges in two regions. To generate a trajectory, we consider a starting point at random in the lower-left corner, and then connect it via a shortest path to a random point in the upper left, center, or lower-right region, which is connected to another random point in the upper-right corner via a shortest path.

We consider the random walk Hodge Laplacians Schaub et al. (2020). For Bunch method, we set the shifting matrices as the simplicial adjacency matrices defined in Bunch et al. (2020). We consider different NNs with three intermediate layers where each layer contains $F = 16$ intermediate features. The tanh nonlinearity is used such that the orientation equivariance holds. The final projection $\boldsymbol{n}$ generates a node feature of dimension one. In the 1000-epoch training, we use the cross-entropy loss function between the output $\boldsymbol{d}$ and the true candidate and we consider an adam optimizer with a learning rate of 0.001 and a batch size 100. To avoid overfitting, we apply a weight decay of $5 \cdot 10^{-6}$ and an early stopping.

As done in Roddenberry et al. (2021), besides the standard trajectory prediction task, we also perform a reverse task where the training set remains the same but the direction of the trajectories in the test set is reversed and a generalization task where the training set contains trajectories running along the upper left region and the test set contains trajectories around the other region. We evaluate the correct prediction ratio by averaging the performance over 10 different data generations.

Real Data We also consider the Global Drifter Program dataset[3] localized around Madagascar. It consists of ocean drifters whose coordinates are logged every 12 hours. An SC can then be created as Schaub et al. (2020) by treating each mesh as a node, connecting adjacent meshes via an edge and filling the triangles, where the "hole" is yielded by the island. Following the process in Roddenberry et al. (2021), it results in 200 trajectories and we use 180 of them for training. In the training, a batch size of 10 is used and no weight decay is used. The rest experiment setup remains the same as the synthetic case.

G.4.3 Results

We report the prediction accuracy of different tasks for both datasets in Table 12. We first investigate the effects of applying higher-order SCFs in the simplicial convolution and accounting for the lower and upper contributions. From the standard accuracy for both datasets, we observe that increasing the convolution orders improves the prediction accuracy, e.g., SCNNs become better as the orders $T_\mathrm{d}, T_\mathrm{u}$ increase and perform

[3] http://www.aoml.noaa.gov/envids/gld/,

always better than PSNN, and SCCNNs better than Bunch. Also, differentiating the lower and upper convolutions does help improve the performance as SCNN of orders $T_{\mathrm{d}} = T_{\mathrm{u}} = 3$ performs better than SNN of $T = 3$.

However, accounting for the node and triangle contributions in SCCNNs does not help the prediction compared to the SCNNs, likewise for Bunch compared to PSNN. This is due to the zero node and triangle inputs because there are no available node and triangle features. Similarly, the prediction directly via the node output features is not accurate compared to projection from edge features.

Moreover, we also observe that the performance of SCCNNs that are trained with the same data does not deteriorate in the reverse task because the orientation equivariance ensures SCCNNs to be unaffected by the orientations of the simplicial data. Lastly, we see that, like other NNs on SCs, SCCNNs have good transferability to the unseen data.

Table 12: Trajectory Prediction Accuracy. *(Left)*: Synthetic trajectory in the standard, reverse and generalization tasks. *(Right)*: Ocean drifter trajectories. For SCCNNs, we set the lower and upper convolution orders $T_{\mathrm{d}}, T_{\mathrm{u}}$ to be the same as T.

Methods	Standard	Reverse	Generalization	Parameters	Standard	Parameters
PSNN	63.1±3.1	58.4±3.9	55.3±2.5	—	49.0±8.0	—
SCNN	65.6±3.4	56.6±6.0	56.1±3.6	$T_{\mathrm{d}} = T_{\mathrm{u}} = 2$	52.5±9.8	$T_{\mathrm{d}} = T_{\mathrm{u}} = 2$
SCNN	66.5±5.8	57.7±5.4	60.6±4.0	$T_{\mathrm{d}} = T_{\mathrm{u}} = 3$	52.5±7.2	$T_{\mathrm{d}} = T_{\mathrm{u}} = 3$
SCNN	67.3±2.3	56.9±4.8	59.4±4.2	$T_{\mathrm{d}} = T_{\mathrm{u}} = 4$	52.5±8.7	$T_{\mathrm{d}} = T_{\mathrm{u}} = 4$
SCNN	67.7±1.7	55.3±5.3	61.2±3.2	$T_{\mathrm{d}} = T_{\mathrm{u}} = 5$	53.0±7.8	$T_{\mathrm{d}} = T_{\mathrm{u}} = 5$
SNN	65.5±2.4	53.6±6.1	59.5±3.7	$T = 3$	52.5±6.0	$T = 3$
Bunch-Node	35.4±3.4	38.1±4.6	29.0±3.0	—	35.0±5.9	—
Bunch-Edge	62.3±4.0	59.6±6.1	53.9±3.1	—	46.0±6.2	—
SCCNN-Node	46.8±7.3	44.5±8.2	31.9±5.0	$T = 1$	40.5±4.7	$T = 1$
SCCNN-Edge	64.6±3.9	57.2±6.3	54.0±3.0	$T = 1$	52.5±7.2	$T = 1$
SCCNN-Node	43.5±9.6	44.4±7.6	32.8±2.6	$T = 2$	45.5±4.7	$T = 2$
SCCNN-Edge	65.2±4.1	58.9±4.1	56.8±2.4	$T = 2$	54.5±7.9	$T = 2$

G.4.4 Convolution Order and Integral Lipschitz Property

We investigate the effect of the integral Lipschitz property of the SCFs in an NN on SC. To do so, given an NN on SCs with an SCF $\boldsymbol{H}_k$ for k-simplicial signals, we add the following integral Lipschitz regularizer to the loss function during training so to promote the integral Lipschitz property

$$r_{\mathrm{IL}} = \|\lambda_{k,\mathrm{G}}\tilde{h}'_{k,\mathrm{G}}(\lambda_{k,\mathrm{G}})\| + \|\lambda_{k,\mathrm{C}}\tilde{h}'_{k,\mathrm{C}}(\lambda_{k,\mathrm{C}})\| = \left\|\sum_{t=0}^{T_{\mathrm{d}}} t w_{k,\mathrm{d},t}\lambda_{k,\mathrm{G}}^{t}\right\| + \left\|\sum_{t=0}^{T_{\mathrm{u}}} t w_{k,\mathrm{u},t}\lambda_{k,\mathrm{C}}^{t}\right\| \tag{82}$$

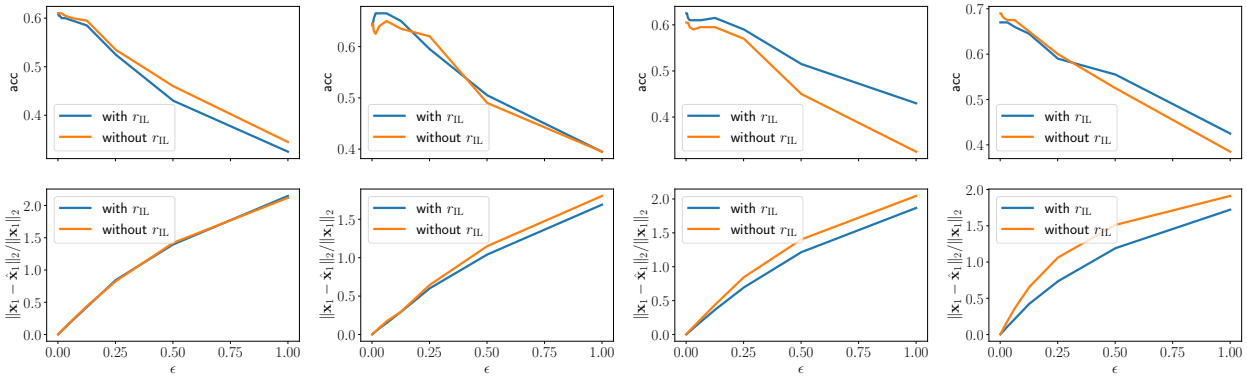
for $\lambda_{k,\mathrm{G}} \in \{\lambda_{k,\mathrm{G},i}\}_{i=1}^{n_{k,\mathrm{G}}}$ and $\lambda_{k,\mathrm{C}} \in \{\lambda_{k,\mathrm{C},i}\}_{i=1}^{n_{k,\mathrm{C}}}$, which are the gradient and curl frequencies. To avoid computing the eigendecomposition of the Hodge Laplacian, we can approximate the true frequencies by sampling certain number of points in the frequency band $(0, \lambda_{k,\mathrm{G,m}}]$ and $(0, \lambda_{k,\mathrm{C,m}}]$ where the maximal gradient and curl frequencies can be computed by efficient algorithms, e.g., power iteration (Watkins, 2007; Sleijpen & Van der Vorst, 2000).

Here, to illustrate that the integral Lipschitz property of the SCFs helps the stability of NNs on SCs, we consider the effect of regularizer r_{IL} against perturbations in PSNNs and SCNNs with different T_{d} and T_{u} for the standard synthetic trajectory prediction. The regularization weight on r_{IL} is set as $5 \cdot 10^{-4}$ and the number of samples to approximate the frequencies is set such that the sampling interval is 0.01.

Fig. 13 shows the prediction accuracy and the relative distance between the edge outputs of the NNs trained with and without the integral Lipschitz regularizer in terms of different levels of perturbations. We see that the integral Lipschitz regularizer helps the stability of the NNs, especially for large SCF orders, where the

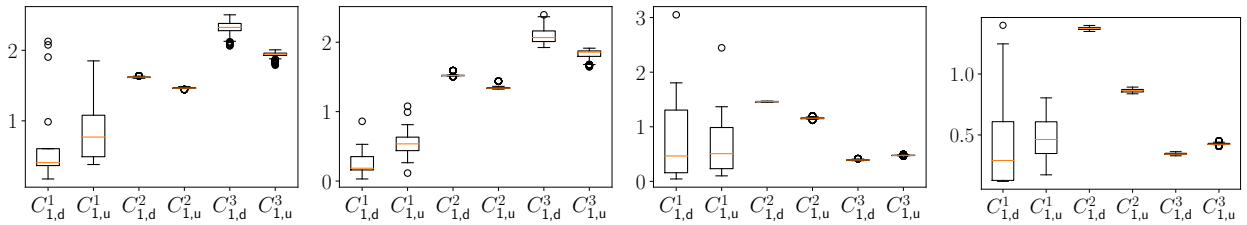
edge output is less influenced by the perturbations compared to without the regularizer. Meanwhile, SCNN with higher-order SCFs, e.g., $T_{\mathrm{d}} = T_{\mathrm{u}} = 5$, achieves better prediction than PSNN (with one-step simplicial shifting), while maintaining a good stability with its output not influenced by perturbations drastically.

We also measure the lower and upper integral Lipschitz constants of the trained NNs across different layers and features, given by $\max_{\lambda_{k,\mathrm{G}}} |\lambda_{k,\mathrm{G}} \tilde{h}_{k,\mathrm{G}}(\lambda_{k,\mathrm{G}})|$ and $\max_{\lambda_{k,\mathrm{C}}} |\lambda_{k,\mathrm{C}} \tilde{h}_{k,\mathrm{C}}(\lambda_{k,\mathrm{C}})|$, shown in Fig. 14. We see that the SCNN trained with r_{IL} indeed has smaller integral Lipschitz constants than the one trained without the regularizer, thus, a better stability, especially for NNs with higher-order SCFs.



(a) PSNN (b) SCNN13 (c) SCNN31 (d) SCNN55

Figure 13: Effect of the integral Lipschitz regularizer r_{IL} in the task of synthetic trajectory prediction against different levels ϵ of random perturbations on $\boldsymbol{L}_{1,\mathrm{d}}$ and $\boldsymbol{L}_{1,\mathrm{u}}$. We show the accuracy (Top row) and the relative distance between the edge output (Bottom row) for different NNs on SCs with and without r_{IL}. SCNN13 is the SCNN with $T_{\mathrm{d}} = 1$ and $T_{\mathrm{u}} = 3$.



(a) SCNN31, without r_{IL} (b) SCNN31, with r_{IL} (c) SCNN55, without r_{IL} (d) SCNN55, with r_{IL}

Figure 14: The integral Lipschitz constants of SCFs at each layer of the trained SCNNs with and without the integral Lipschitz regularizer r_{IL}. We use symbols $c^l_{k,\mathrm{d}}$ and $c^l_{k,\mathrm{u}}$ to denote the lower and upper integral Lipschitz constants at layer l. Regularizer r_{IL} promotes the integral Lipschitz property, thus, the stability, especially for NNs with large SCF orders.