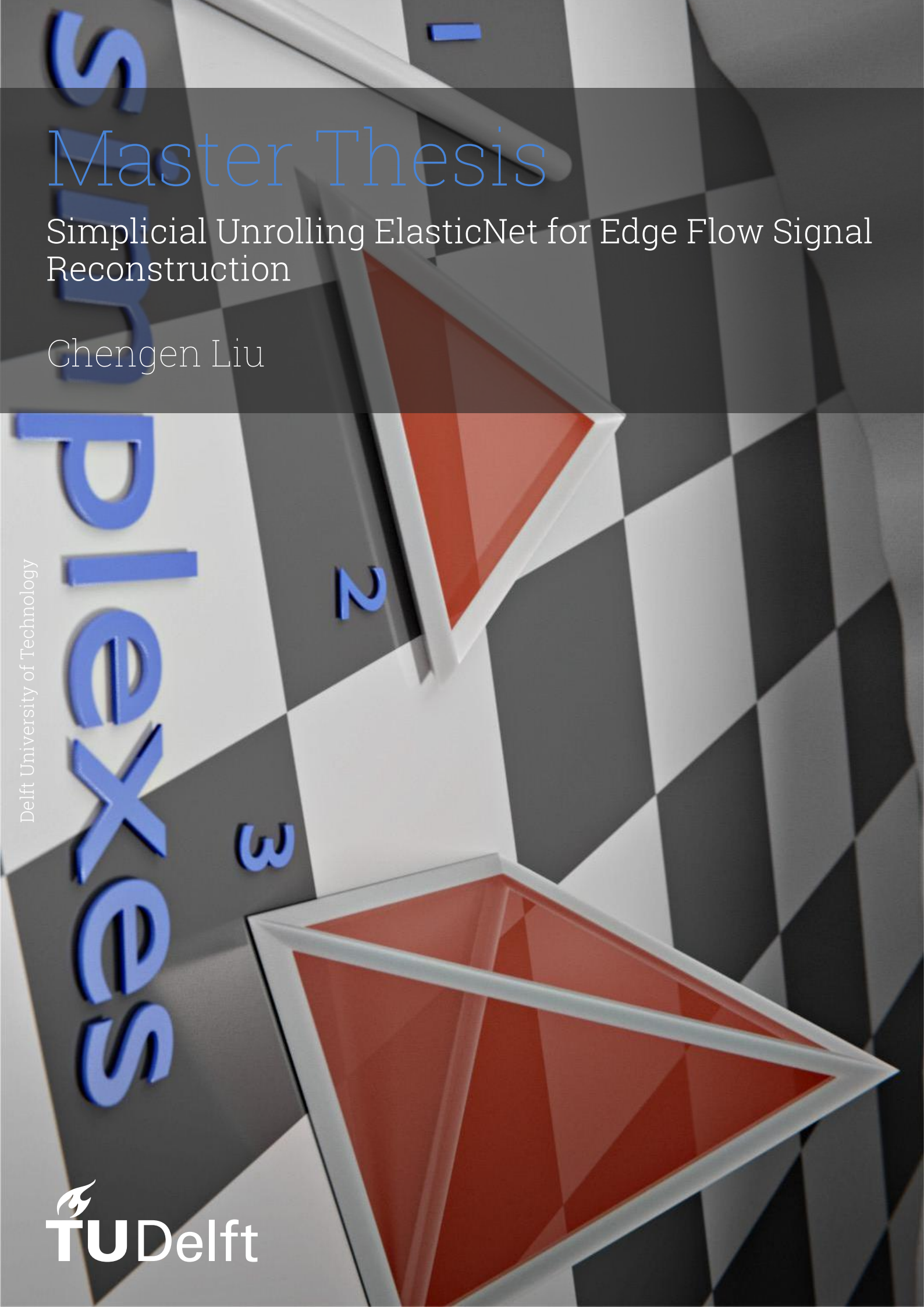


Master Thesis

Simplicial Unrolling ElasticNet for Edge Flow Signal Reconstruction

Chengen Liu

Delft University of Technology

TU Delft

Master Thesis

Simplicial Unrolling ElasticNet for Edge Flow Signal Reconstruction

by

Chengen Liu

Student Name	Student Number
Chengen Liu	5512646

Supervisor:	G. Leus
Daily supervisor:	E. Isufi
Committee member:	R. Taormina
Project Duration:	November, 2022 - July, 2023
Faculty:	Faculty of Electrical Engineering, Mathematics and Computer Science

TU Delft

Acknowledgments

The two years I spent studying for my master's degree at Delft University of Technology were a valuable experience for me.

I would like to thank my supervisors, Prof.Geert Leus and Dr.Elvin Isufi, who are always willing to help me with their guidance and patience.

I would also like to thank my friend Tianyi Li. He was there to support me during my most difficult time spiritually.

Last but not least, I would like to thank my parents. They have been my strongest support since I was a child. I hope they will always be healthy and happy.

Chengen Liu
Delft, July 2023

Abstract

The edge flow reconstruction task improves the integrity and accuracy of edge flow data by recovering corrupted or incomplete signals. This can be solved by a regularized optimization problem, and the corresponding regularizers are chosen based on prior knowledge. However, obtaining prior information is challenging in some fields. Thus, we consider exploiting the learning ability of neural networks to acquire prior knowledge. In this paper, we propose a new optimization problem for the simplicial edge flow reconstruction task, the simplicial ElasticNet, which is a regularized optimization problem that combines the advantages of the ℓ_1 and ℓ_2 norm. It is solved iteratively by the multi-block ADMM algorithm, and the convergence conditions are illustrated. By unrolling the simplicial ElasticNet's iterative steps, we propose a neural network with high interpretability and low requirement for the number of training data for the reconstruction task of simplicial edge flows. The unrolling network replaces the fixed parameters in the iterative algorithm with the learnable weights in the neural networks, thus exploiting the neural network's learning capability while preserving the iterative algorithm's interpretability. The core component of this unrolling network is simplicial convolutional filters with learnable weights to aggregate information from the edge flow neighbors, thus enhancing the learning and expressive ability of the network. We conduct extensive experiments on real-world and artificial datasets to validate the proposed approach. It is demonstrated that the simplicial unrolling network is significantly more advantageous than the traditional iterative algorithms and standard non-model-based neural networks in the case of limited training data.

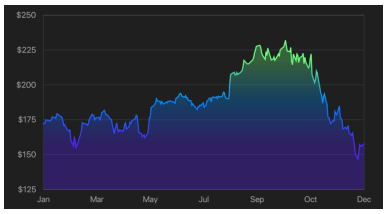
Contents

1

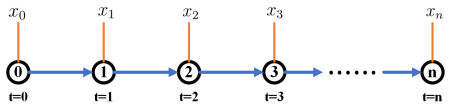
Introduction

1.1. Motivation

Data plays a vital role in our lives as various activities generate large amounts of data. The information contained in the data needs to be processed before it can be accessed and utilized by humans. Common data include time series, as shown in Figure 1.1(a), and images, as in Figure 1.1(c). For example, stock price changes are time-series data. Photos are image data. Time series and images are essentially regular domain data because they have a regular structure. Different time instants in the time series and pixels in the image can be modeled as nodes of a graph and adjacently connected to each other. Values of time series and pixels can be modeled as data on nodes, as shown in Figure 1.1(b) and Figure 1.1(d). Thus, regular data can be processed by standard machine learning techniques such as convolutional or recurrent neural networks [28] [32].

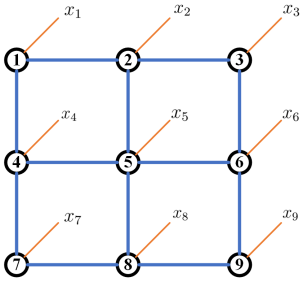


(a) Time-series.



(b) Structure of time-series.

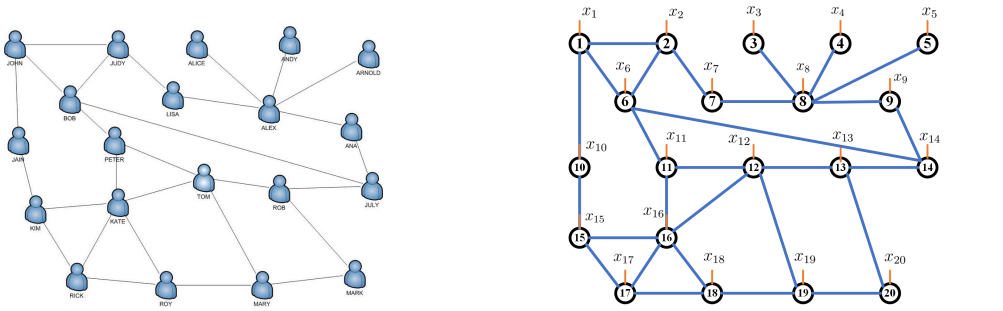
(c) Image.



(d) Structure of image.

Figure 1.1: Examples of regular data.

However, regular data form only a small part of the vast amount of data we collect, which is irregular and cannot be processed by the aforementioned techniques. Many real-world systems can be modeled as graphs generate graph-based data [38] [46] [58], such as human social relationship networks, transportation networks, protein molecular structures, and literature citation relationship networks. For example, people in social networks can be modeled as nodes in a graph, and their relationships can be modeled as edges between them, as shown in Figure 1.2(a). Their personal information can be modeled as data on nodes, such as height, weight, etc., as shown in Figure 1.2(b).



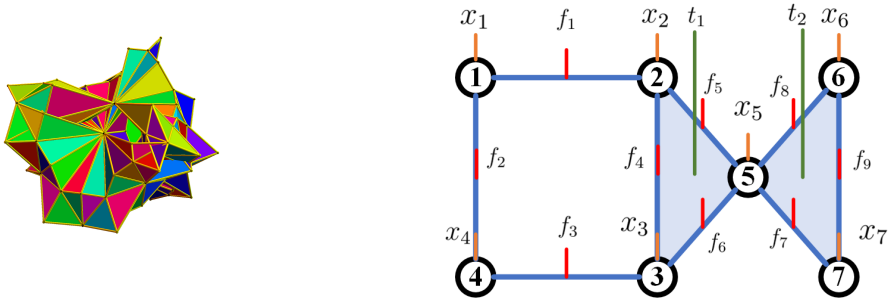
(a) Social networks. (b) Graph data for social networks.

Figure 1.2: Examples of graph data.

There are many methods that can be applied to process graph data, such as graph signal processing (GSP) or graph neural networks (GNNs). GSP contains tools that can process graph data on the nodes, such as graph filters [21]. GNNs are a class of neural network models that have achieved promising results in practical applications [20].

However, the expressive ability of graphs is limited since they can only capture pairwise relationships [6]. For example, a graph can model a molecular structure where each atom is modeled as a node in the graph, and the molecular bonds between atoms can be modeled as an edge. However, the relationship between atomic groups cannot be described by low-order networks such as graphs [2]. Therefore, higher-order networks, such as simplicial complexes, need to be applied to model more complex relationships, as shown in Figure 1.3(a). Data can be defined on nodes, edges, triangles, tetrahedrons, and so on, as shown in Figure 1.3(b), and are also known as simplicial signals. Currently, edge flows are one of the most widely studied areas with a practical interest in traffic flow prediction in a road network [48] and currency exchange rate analysis [22].

There exist various tools available to process simplicial signals. For instance, the topological signal processing (TSP) [2] extends the concepts in signal processing (SP) [40] and graph signal processing (GSP) [38] to simplicial signals, such as simplicial convolutional filters [38] and simplicial Fourier transforms [47]. In addition, machine learning methods have led to various applications of higher-order network data, such



(a) Simplicial complexes. (b) Simplicial signals.

Figure 1.3: Examples of higher-order network data

as recommendation systems [57], point clouds [50], finance [54], and biochemistry [4].

Therefore, this paper focuses on the data defined on simplicial complexes. The task is to reconstruct the edge flow signal of simplicial complexes.

1.2. Scope of the research

Reconstructing signals from noisy or partial measurements is a long-lasting challenge in signal processing. This task requires exploiting particular behavior of the signal w.r.t. the underlying medium. One of the most common approaches is based on the regularized optimization [45]. Regularizers introduce a bias between the solution and the actual value, and various regularizers possess distinct characteristics. For example, in graph signal processing (GSP) [38], where the data is defined on the nodes of graphs, the Tikhonov regularizer is often used to recover the graph signals based on the assumption that connected nodes have similar values. When the graph signal is piece-wise smooth, graph trend filtering [55] with ℓ_1 norm regularizer is more effective as it promotes sparsity in the signal difference of connected nodes.

Graphs can only model pair-wise relationships [1]. However, in many cases, we are interested in signals defined on edges or multi-way relationships. For simplicial edge flows, the reconstruction task can also be solved by regularized methods. The papers in [48] and [22] use the Tikhonov regularizer in the denoising and interpolation tasks of simplicial edge flow, which can reduce the curl and divergence of the edge flow. However, Tikhonov regularizers can only reduce the divergence and curl of the reconstructed signal globally but cannot reconstruct the divergence-free and curl-free edge flow exactly. The divergence- or curl-free properties are important because many edge flows in reality possess these properties. Therefore, this thesis proposes a simplicial ElasticNet that combines the Tikhonov regularizer with the ℓ_1 regularizer to solve the reconstruction task.

The regularizers and their coefficients are determined based on prior knowledge.

This prior knowledge can often be challenging to obtain [33]. Therefore, it is a viable scheme to use neural networks to learn the prior. This thesis combines the learning ability of neural networks with conventional iterative algorithms to develop an unrolling network on simplicial complexes for edge flow reconstruction. The unrolling network is a model-based neural network and has been successfully applied in several fields such as medical imaging [51], smart power grid [64], remote sensing [31]. Paper [8] proposes a graph unrolling network for the graph signal denoising task.

The core idea of the unrolling technique is to map each iteration of optimization algorithms into a layer of the unrolling networks. On the one hand, we consider the simplicial ElasticNet, which is a convex optimization problem and can be solved by various standard iterative algorithms such as ADMM [5]. On the other hand, we replace the fixed parameters in the iterations with trainable weights in the unrolling networks, map each iteration into a network layer, and stack multiple layers sequentially to form the complete unrolling network [15] [63]. There are many different parameterization strategies for trainable weights. This thesis replaces a portion of the fixed parameters with learnable simplicial convolutional filters which allows to aggregate information from the neighbors of the edge flow [62], thus improving the expressive ability of the unrolling network. Using the ADMM to build an unrolling network, we give a more tailored neural network solution for edge flow reconstruction. There have been many attempts to develop neural networks on simplicial complexes, such as simplicial neural networks (SNNs) [9], simplicial convolutional neural networks (SCNNs) [61], and simplicial complexes convolutional neural networks (SCCNNs) [60]. These models were developed to accomplish simplex prediction, trajectory prediction, and imputing citations on a coauthorship complex. However, these standard models are entirely data-driven and require a large amount of training data, limiting their ability because the cost of acquiring training data is high. In the edge flow reconstruction task, we often have access to limited data. Due to the special structure of the unrolling networks, its trainable parameters are a few. Therefore, learning tasks with fewer training samples can be accomplished. We validate the advantages of unrolling networks for simplicial edge flow reconstruction tasks with one-shot learning.

1.3. Research statement

In this thesis work, we focus on the following challenging situations:

- Existing edge flow reconstruction tasks are often formulated based on regularized optimization problems. The regularizers often affect the solution of the optimization problem. Finding out the appropriate regularizers for a particular problem is a challenge.
- Solving regularized optimization problems using traditional iterative algorithms requires the use of existing prior knowledge, e.g., the choice of regularizer coefficients. However, obtaining this prior knowledge often requires a combination of

relevant domain expertise, which is sometimes difficult.

Based on these two challenges, the following two research questions are proposed:

- How to choose the proper regularizer for the edge flow reconstruction task?
- How to reduce our reliance on a priori knowledge?

1.4. Contributions of this research

The main contributions of this thesis are:

- We propose an ElasticNet problem for the simplicial edge flow reconstruction task. This is based on both the ℓ_1 and ℓ_2 norm, which generalizes existing regularization approaches.
- We solve the simplicial ElasticNet and trend filtering problem by multi-block ADMM algorithm and build the corresponding simplicial unrolling network for them.
- We prove the convergence of this simplicial unrolling network under some weak assumptions and conduct corresponding experiments to corroborate the proof.
- We conduct numerical experiments on synthetic and real datasets to corroborate the proposed method and show its superior performance not only to simplicial-based regularizers but also to neural network solutions under the one-shot learning setting.

2

Background

Signals defined on higher-order networks have similarities with graph signals but have a stronger expressiveness. Among them, simplicial complexes are a classical higher-order network and the object of this paper. In this chapter, section 2.1 defines the concepts related to the topology of the simplicial complexes. Section 2.2 describes the simplicial signal. Section 2.3 introduces the ADMM algorithm. Section 2.4 introduces the concepts related to unrolling networks. Section 2.5 analyzes a graph unrolling network as an example. Section 2.6 summarizes the chapter.

2.1. Higher-order networks

2.1.1. Related works

A topological space is a set that defines discrete elements and neighborhood relationships. The graph data is a typical example of topological spaces. The graph encodes pairwise relations in the form of edges. However, many relationships in the real world cannot be modeled as pairwise relationships but should be modeled as multiway relationships. For example, many chemical reactions involve more than one reactant and product [25], which makes it impossible to express such relationships using graphs. Likewise, in natural language processing (NLP), the meaning of a sentence is often related to the combination of many vocabularies. Therefore, it should not be simply modeled as the relationship between pairs of words. Multilayer graphs were proposed to model more complex relationships [3]. However, multilayer graphs are still modeling pairwise relationships. To upgrade from pairwise relation to multiway relation, a framework utilizing topological tools to obtain valuable information from data called Topological Data Analysis (TDA) is proposed [6]. In TDA, simplicial complexes are a representative research direction. Its rich algebraic structure facilitates the modeling of higher-order interactions.

TDA studies the properties of simplicial complexes. To analyze signals defined on

topological spaces, topological signal processing (TSP) is proposed [43]. GSP can be counted as a particular case of TSP. Relationships in higher-order networks, such as simplicial complexes, can be represented by matrices or tensors. GSP focuses on signals defined on graphs, i.e., signals at nodes and edges. In contrast, TSP focuses on signals on simplicial complexes, i.e., nodes, edges, triangles, tetrahedra, etc. Signals defined on edges include blood flow data [18], communication flow data, and traffic network data [26]. Signals defined on a triangle include co-author networks [39], etc. Among them, considerable research is on the signal defined on edges, the 1-simplicial signal. Paper [11] studies the signal on edge based on the line graph. In this way, the edge signals can be converted into node signals, and then the signal can be processed by the method in GSP. However, modeling edge signals with line graphs is not the best option. Paper [48] points out that in the edge flow signal denoising task, the regularizer based on the line graph Laplacian matrix has worse denoising performance than the edge-space filter. Paper [35] extends the random walk algorithm to simplicial complexes. These papers complete the theory of TSP.

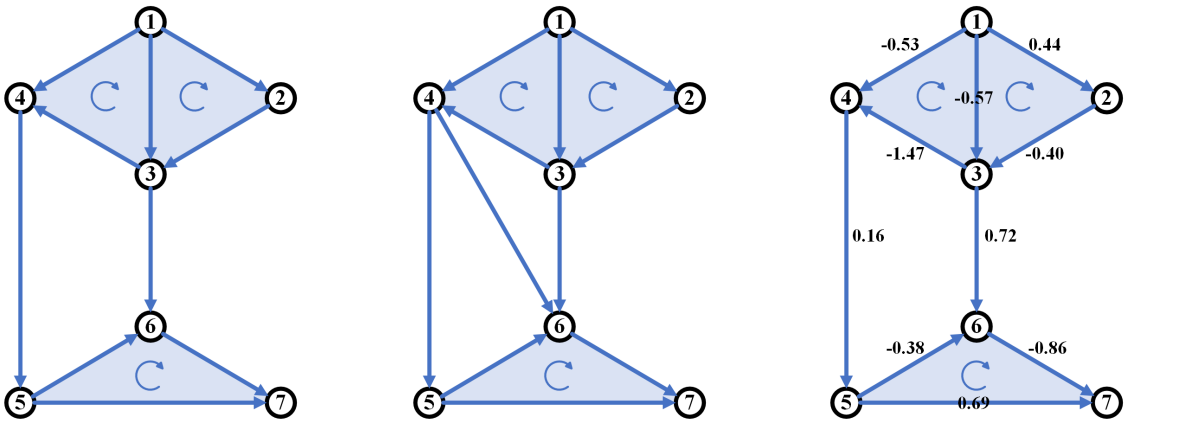
TSP also inherits some concepts from classical signal processing. In paper [2], the concept of the simplicial Fourier transform is defined. Similar to GSP, the concept of filtering is also defined in TSP. Some works use regularization to filter simplicial signals. For example, [48], [49] perform regularization optimization based on Hodge-Laplacian matrices to obtain signals with conservation of edge flow. The paper [62] proposes a simplicial convolutional filter. This filter mainly comprises the polynomial of the upper Hodge Laplacian matrix and the lower Hodge Laplacian matrix. The paper proves that simplicial convolutional filters have several important properties, including linearity, shift-invariance, and permutation and orientation equivariance. Similar to GNNs, neural networks for simplicial complexes have been proposed. The paper [9] proposes the Simplicial Neural Network (SNN). The convolutional layer of the SNN consists of a primary simplicial filter and a nonlinear activation function. Inspired by Convolutional Neural Networks (CNN), Simplicial Convolutional Neural Networks (SCNN) are proposed [61]. A SCNN is more flexible than a SNN in exploiting simplicial upper and lower neighborhoods. The SNN can be regarded as a particular form of the SCNN.

Simplicial complexes have applications in control engineering [34], statistical ranking [23], tumor progression data decomposition [44], harmonic clustering [10], brain network analysis [13], neuronal morphology [24], co-authorship networks [39] and collaboration networks [41]. Paper [34] analyzes the Laplacian flow on the simplicial complex and points out that the stability of some dynamical systems is related to the structural properties of the simplicial complexes. Paper [23] proposes a technique called *HodgeRank* to rank data. The authors model pairwise rankings as flows at the edges of the graph. The 1-simplicial signal can explain this, and the global ordering of the dataset by the Hodge decomposition makes sense. Paper [44] is the application of

simplicial complexes in bioinformatics. Many different factors influence the evolution of tumors. Mixed cell populations have different effects on tumors. Paper [44] applies simplicial complexes to model this mixed modeling problem. Inspired by graph spectral clustering, paper [10] proposes a simplicial complexes-based clustering algorithm. A simplicial complexes clustering algorithm is used to study the feature extraction of topological data. In past research, neural networks in the brain are often modeled using graphs. However, such modeling requires the assumption that the basic unit of the brain is a dyad that contains two neurons and their connections. The expressive power of this way of modeling with graphs is limited. To solve this problem, the paper [13] uses simplicial complexes to model the brain neural network. The simplicial complexes framework can potentially replace graphs for brain neural network analysis research.

2.1.2. Simplicial complexes

Given a finite set of vertices $\mathcal{V}$, a k-simplex $\mathcal{S}^k$ is a subset of $\mathcal{V}$ with cardinality $k+1$. A simplicial complex of order K, $\mathcal{X}^K$, is a finite collection of k-simplices $\mathcal{S}^k$ for $k = 0, 1, ..., K$ satisfying the inclusion property: for any $\mathcal{S}^k \in \mathcal{X}$, all of its subsets $\mathcal{S}^{k-1} \subset \mathcal{S}^k$ satisfy $\mathcal{S}^{k-1} \in \mathcal{X}$. The number of k-simplices in a simplicial complex is denoted by N_k. For example, a node is a 0-simplex, an edge is a 1-simplex, and a (filled) triangle is a 2-simplex, as shown in Fig.1. A graph is, therefore, a simplicial complex of order one where its nodes are 0-simplices, and its edges are 1-simplices. A *face* of simplex $\mathcal{S}^k$ is defined as a subset with cardinality k. A *coface* of $\mathcal{S}^k$ is defined as a (k+1)-simplex which contains $\mathcal{S}^k$. Two simplices are lower adjacent if they have a common face and upper adjacent if they have a common coface. For example, if two edges share a node, they are lower adjacent, and if two edges share a (filled) triangle, they are upper adjacent.



(a) Simplicial complex. (b) Simplicial complex. (c) Simplicial signals on edges.

Figure 2.1: Simplicial complexes and simplicial signals

Take Figure 2.1(a) as an example to illustrate these concepts. Figure 2.1(a) is a simplicial

complex of order two, which contains seven 0-simplices (nodes), ten 1-simplices (edges), and three 2-simplices (filled triangles). The faces of the edge $\{3,4\}$ are node $\{3\}$ and node $\{4\}$. The faces of the triangle $\{1,3,4\}$ are edge $\{1,4\}$, edge $\{1,3\}$ and edge $\{3,4\}$. Edges $\{3,4\}$ and $\{1,3\}$ share a common face, the node $\{3\}$ and a common coface, the triangle $\{1,3,4\}$. Thus, they are both lower neighbors and upper neighbors. Edge $\{3,4\}$ and $\{3,6\}$ share a common face, the node $\{3\}$ but they do not have a common coface. Thus, they are lower neighbors but not upper neighbors. It is worth noting that only filled triangles are 2-simplices. For example, in Figure 2.1(b), triangle $\{1,3,4\}$ is a 2-simplex which is filled but triangle $\{3,4,6\}$ is not a 2-simplex because it is empty.

2.1.3. Incidence matrix

The incidence matrix of a graph can be used to evaluate the connectivity between different nodes. A similar relationship exists between simplices of different orders. Incidence matrix $\mathbf{B}_k \in \mathbb{R}^{N_{k-1} \times N_k}$ can also be used to represent the relationship between (*k-1*)-simplices and *k*-simplices. If there is a simplex $\mathcal{S}_j^{k-1} \subset \mathcal{S}_i^k$, we can use $\mathcal{S}_j^{k-1} \sim \mathcal{S}_i^k$ to denote that their orientations are consistent. And if their orientations are not consistent, $\mathcal{S}_j^{k-1} \nsim \mathcal{S}_i^k$. The definition of the incidence matrix is as follows:

$$\mathbf{B}_k(i,j) = \begin{cases} 0, \text{ if } \mathcal{S}_i^{k-1} \not\subset \mathcal{S}_j^k \\ 1, \text{ if } \mathcal{S}_i^{k-1} \subset \mathcal{S}_j^k \text{ and } \mathcal{S}_i^{k-1} \sim \mathcal{S}_j^k \\ -1, \text{ if } \mathcal{S}_i^{k-1} \subset \mathcal{S}_j^k \text{ and } \mathcal{S}_i^{k-1} \nsim \mathcal{S}_j^k. \end{cases} \tag{2.1}$$

The topology of a simplicial complex can be entirely described by $\mathbf{B}_k$ for $k = 1, ..., K$. $\mathbf{B}_1$ is the node-to-edge incidence matrix which is used in GSP, and $\mathbf{B}_2$ is the edge-to-triangle incidence matrix. For example, the incidence matrices of the simplicial complexes presented in Figure 2.1(a) is shown as follows:

$$\mathbf{B}_1 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{cccccccccc} (1,2) & (1,3) & (1,4) & (2,3) & (3,4) & (3,6) & (4,5) & (5,6) & (5,7) & (6,7) \end{array} \\ \begin{pmatrix} -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \end{array} \quad \mathbf{B}_2 = \begin{array}{c} \\ (1,2) \\ (1,3) \\ (1,4) \\ (2,3) \\ (3,4) \\ (3,6) \\ (4,5) \\ (5,6) \\ (5,7) \\ (6,7) \end{array} \begin{array}{c} \begin{array}{ccc} (1,3,4) & (1,2,3) & (5,6,7) \end{array} \\ \begin{pmatrix} 0 & 1 & 0 \\ 1 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & -1 \\ 0 & 0 & 1 \end{pmatrix} \end{array} \tag{2.2}$$

Each row of $\mathbf{B}_2$ represents an edge (1-simplex) in the simplicial complex, and each column represents a filled triangle (2-simplex). The orientations of the 2-simplex $\{1,3,4\}$ and its faces are given in Figure 1(a). It has three faces $\{1,3\}$, $\{1,4\}$ and $\{3,4\}$. Therefore, the first column of $\mathbf{B}_2$ has three non-zero elements, which are in rows two, three, and five. The orientations of $\{1,3\}$ and $\{1,4\}$ are coherent with $\{1,3,4\}$. Therefore, the corresponding elements in $\mathbf{B}_2$ are 1. The orientation of $\{3,4\}$ is not coherent with $\{1,3,4\}$. Thus, the corresponding element is -1.

2.1.4. Hodge Laplacians

The whole simplicial complexes structure can be represented by the *Hodge Laplacian* matrices

$$\begin{aligned}\mathbf{L}_0 &= \mathbf{B}_1\mathbf{B}_1^\top \\ \mathbf{L}_k &= \mathbf{B}_k^\top\mathbf{B}_k + \mathbf{B}_{k+1}\mathbf{B}_{k+1}^\top, k = 1, \dots, K-1 \\ \mathbf{L}_K &= \mathbf{B}_K^\top\mathbf{B}_K\end{aligned} \tag{2.3}$$

Except for $\mathbf{L}_0$ and $\mathbf{L}_K$, all other Hodge-Laplacian matrices can be decomposed into the sum of two terms: the first term $\mathbf{L}_{k,\ell} = \mathbf{B}_k^\top\mathbf{B}_k$ is the *lower Laplacian* and the second term $\mathbf{L}_{k,u} = \mathbf{B}_{k+1}\mathbf{B}_{k+1}^T$ is the *upper Laplacian*. The lower Laplacian represents the lower adjacencies of k-simplices (e.g., how two edges are adjacent via a common node), while the upper Laplacian represents the upper adjacencies (e.g., how two edges are adjacent by being the faces of the same triangle). From the definition of the incidence matrix, an important property is:

$$\mathbf{B}_k\mathbf{B}_{k+1} = \mathbf{0}. \tag{2.4}$$

2.2. Simplicial signal processing

2.2.1. Simplicial signals

A $k-$simplicial signal, for short $k-$signal, is a mapping from the $k-$simplex to the set of real numbers. We collect the $k-$signal into the vector $\mathbf{s}^k = \begin{bmatrix} s_1^k, \dots, s_{N_k}^k \end{bmatrix}^\top \in \mathbb{R}^{N_k}$ where entry s_i^k corresponds to ith k-simplex. In this thesis, we will be mostly interested in processing edge flows; hence we will deal with simplicial complexes of order $K = 2$. In this setting, we denote the 0-signal as $\mathbf{v} := \mathbf{s}^0 = [v_1, \dots, v_{N_0}]^\top \in \mathbb{R}^{N_0}$, the 1-signal as $\mathbf{f} := \mathbf{s}^1 = [f_1, \dots, f_{N_1}]^\top \in \mathbb{R}^{N_1}$, and the 2-signal as $\mathbf{t} := \mathbf{s}^2 = [t_1, \dots, t_{N_2}]^\top \in \mathbb{R}^{N_2}$. For processing purposes, we define an arbitrary reference orientation of each simplex and follow the lexicographical ordering of the vertices. A simplex can only have two orientations. The orientations of the edges in the graph are set based on the labeling of vertices. If the value of the edge signal is positive, the set orientations are consistent with the real situation. If it is negative, the set orientation is opposite to the real one. This definition also holds for higher-order simplicial complexes. The 1-simplicial signal in Figure 2.1(c) is $[0.44, -0.57, -0.53, -0.40, -1.47, 0.72, 0.16, -0.38, 0.69, -0.86]$.

2.2.2. Spectral of simplicial complexes

In GSP, the graph Fourier transform performs the eigenvalue decomposition of the graph Laplacian matrix as:

$$\mathbf{L}_0 = \mathbf{U}_0\mathbf{\Lambda}_0\mathbf{U}_0^T \tag{2.5}$$

where $\mathbf{U}_0$ contains the eigenvectors of $\mathbf{L}_0$ and $\mathbf{\Lambda}_0$ the eigenvalues. Then the eigenvectors of $\mathbf{L}_0$ can be used to represent the graph signal as:

$$\mathbf{s^0} = \mathbf{U}_0\widehat{\boldsymbol{s}}^0 \tag{2.6}$$

where $\widehat{s}^0$ is the projection of the graph signal in the space that is spanned by the eigenvectors of $\mathbf{L}_0$ as:

$$\widehat{s}^0 = \mathbf{U}_0^T s^0 \tag{2.7}$$

These concepts can be extended to simplicial complexes. The simplicial signal can also be represented by the eigenvectors of $\mathbf{L}_k$. Given the eigendecomposition of $\mathbf{L}_k$,

$$\mathbf{L}_k = \mathbf{U}_k \mathbf{\Lambda}_k \mathbf{U}_k^T \tag{2.8}$$

the k-simplicial signal can be projected onto the space spanned by the eigenvectors of $\mathbf{L}_k$ as:

$$\widehat{s}^k = \mathbf{U}_k^T s^k. \tag{2.9}$$

The k-simplicial signal can also be represented by its GFT coefficients $\widehat{s}^k$ as:

$$s^k = \mathbf{U}_k \widehat{s}^k. \tag{2.10}$$

Four properties hold for Laplacian matrices $\mathbf{L}_k$: 1) the eigenvectors of $\mathbf{L}_{k,\ell}$ corresponding to non-zero eigenvalues are orthogonal to the eigenvectors of $\mathbf{L}_{k,\mathrm{u}}$ corresponding to non-zero eigenvalues; 2) if v is an eigenvector of $\mathbf{L}_{k-1,\mathrm{u}}$ corresponding to eigenvalue λ, $\mathbf{B}_k^T v$ is also an eigenvector of $\mathbf{L}_{k,\ell}$ with eigenvalue λ; 3) the eigenvectors of $\mathbf{L}_k$ with non-zero eigenvalue λ are either the eigenvectors of $\mathbf{L}_{k,\ell}$ or $\mathbf{L}_{k,\mathrm{u}}$; 4) the non-zero eigenvalues of $\mathbf{L}_k$ are either the eigenvalues of $\mathbf{L}_{k,\ell}$ or $\mathbf{L}_{k,\mathrm{u}}$.

2.2.3. Hodge decomposition

The space of a simplicial signal $\mathbb{R}^{N_k}$ can be decomposed into three orthogonal subspaces

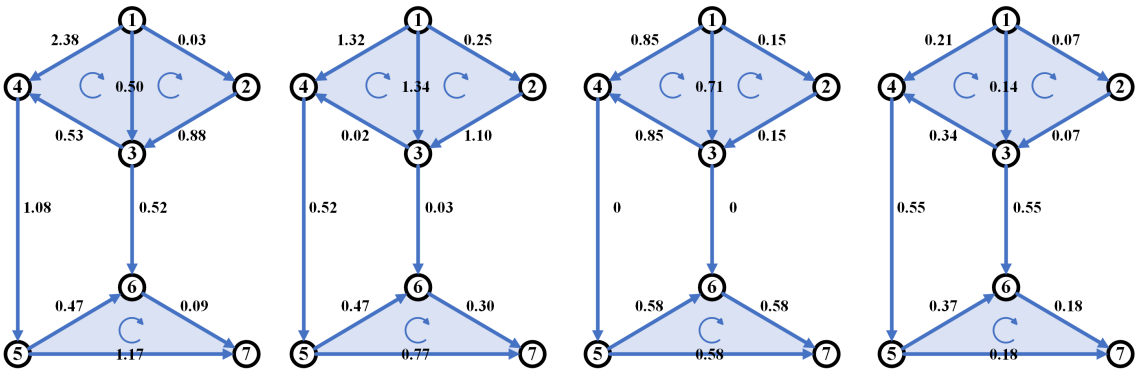
$$\mathbb{R}^{N_k} \equiv \mathrm{im}\left(\mathbf{B}_k^\top\right) \oplus \ker\left(\mathbf{L}_k\right) \oplus \mathrm{im}\left(\mathbf{B}_{k+1}\right) \tag{2.11}$$

where $\mathrm{im}(\cdot)$ and $\ker(\cdot)$ are the image and kernel spaces of a matrix and $\oplus$ is the direct sum. That is, for any simplicial signal s^k of order k, there exists three simplicial signals of orders $k-1$, k, and $k+1$ so that we can decompose s^k as

$$s^k = \mathbf{B}_k^\top s^{k-1} + s_{\mathrm{H}}^k + \mathbf{B}_{k+1} s^{k+1}. \tag{2.12}$$

This *Hodge decomposition* expresses the relationship between different orders of simplicial signals. For edge flow signals, the Hodge decomposition carries the following specific intuition. The edge flow $\mathbf{f} \in \mathbb{R}^{N_1}$ can be written as a sum of three edge flows as $\mathbf{f} = \mathbf{f}_{\mathrm{G}} + \mathbf{f}_{\mathrm{C}} + \mathbf{f}_{\mathrm{H}}$, where the $\mathbf{f}_{\mathrm{G}} \in \mathrm{im}\left(\mathbf{B}_1^\top\right)$ is the gradient component, $\mathbf{f}_{\mathrm{H}} \in \ker\left(\mathbf{L}_1\right)$ the harmonic component, and $\mathbf{f}_{\mathrm{C}} \in \mathrm{im}\left(\mathbf{B}_2\right)$ the curl component (see also Figure 2.2) with the following explanation:

- Gradient component: $\mathbf{f}_{\mathrm{G}} = \mathbf{B}_1^\top \mathbf{v}$ is an edge flow induced by taking the difference between the two node signals at the extremities of the edge. Operator $\mathbf{B}_1^\top$ is referred to as the gradient operator, and likewise, the space $\mathrm{im}\left(\mathbf{B}_1^\top\right)$ is referred to as the gradient space.



(a) Edge flow $\mathbf{f}$. (b) $\mathbf{f}_{\mathrm{G}}$. (c) $\mathbf{f}_{\mathrm{C}}$. (d) $\mathbf{f}_{\mathrm{H}}$.

Figure 2.2: Decomposition of the edge flow.

- Curl component: $\mathbf{f}_{\mathrm{C}} = \mathbf{B}_2\mathbf{t}$ is a curl flow locally circling along the edges of triangles induced by a triangle signal $\mathbf{t}$. Operator $\mathbf{B}_2$ is the curl adjoint. The space $\mathrm{im}\,(\mathbf{B}_2)$ is the curl space.
- Harmonic component: $\mathbf{f}_{\mathrm{H}}$ is the part of the edge flow that satisfies $\mathbf{L}_1\mathbf{f}_{\mathrm{H}} = \mathbf{0}$. The space $\ker\,(\mathbf{L}_1)$ is called the harmonic space.

In addition, we also define the following two operators:

- Curl operator: The curl operator is defined as $\mathrm{curl}(\mathbf{f}) = \mathbf{B}_2^{\top}\mathbf{f}$ which measures the curl of an edge flow. The ith element corresponds to the sum of the flow of each edge forming the ith triangle. If the curl of an edge flow is zero at each triangle, it is curl-free. The gradient component $\mathbf{f}_{\mathrm{G}}$ and the harmonic component $\mathbf{f}_{\mathrm{H}}$ are curl-free by definition [2].
- Divergence operator: The divergence operator is defined as $\mathrm{div}(\mathbf{f}) = \mathbf{B}_1\mathbf{f}$, which measures the divergence of an edge flow. The ith element corresponds to the flow passing through the ith node. If the divergence of an edge flow is zero at each node, it is divergence-free. The curl component $\mathbf{f}_{\mathrm{C}}$ and harmonic component $\mathbf{f}_{\mathrm{H}}$ are divergence-free by definition [2].

2.2.4. Simplicial Convolutional Filters

The *simplicial convolution* is defined as follows:

$$\mathbf{y}^k = \left(\sum_{l_1=0}^{L_1} \alpha_{l_1}\mathbf{L}_{k,\ell}^{l_1} + \sum_{l_2=0}^{L_2} \beta_{l_2}\mathbf{L}_{k,u}^{l_2}\right)\mathbf{s}^k \tag{2.13}$$

where the $\mathbf{s}^k$ is a k-simplicial signal, $\mathbf{L}_{k,\ell}$ is the lower Laplacian, $\mathbf{L}_{k,u}$ is the upper Laplacian. L_1 and L_2 are the convolutional orders. The simplicial convolutional filter is defined as follows:

$$\mathbf{H}_k := \left(\sum_{l_1=0}^{L_1} \alpha_{l_1}\mathbf{L}_{k,\ell}^{l_1} + \sum_{l_2=0}^{L_2} \beta_{l_2}\mathbf{L}_{k,u}^{l_2}\right) \tag{2.14}$$

It can be decomposed into two terms. The first term gathers the information from lower-adjacencies, and the second term gathers the information from upper-adjacencies. The simplicial convolutional filter has some important properties. The first one is that they are linear operators as follows:

$$\mathbf{H}_k\left(a\mathbf{s}_1^k + b\mathbf{s}_2^k\right) = a\mathbf{H}_k\mathbf{s}_1^k + b\mathbf{H}_k\mathbf{s}_2^k \tag{2.15}$$

The second property is shift invariance as follows:

$$\begin{aligned}\mathbf{L}_{k,\ell}\left(\mathbf{H}_k\mathbf{s}^k\right) &= \mathbf{H}_k\left(\mathbf{L}_{k,\ell}\mathbf{s}^k\right)\\ \mathbf{L}_{k,\mathrm{u}}\left(\mathbf{H}_k\mathbf{s}^k\right) &= \mathbf{H}_k\left(\mathbf{L}_{k,\mathrm{u}}\mathbf{s}^k\right)\end{aligned} \tag{2.16}$$

The third and fourth properties are permutation equivariance and orientation equivariance, respectively, which means that the output will not be influenced by labeling order and reference orientation [62].

2.3. ADMM algorithm

The Alternating Direction Multiplier Method (ADMM) is a convex optimization algorithm. It is a primary method for solving separable convex optimization problems. The ADMM decomposes a large global problem into multiple smaller and easier-to-solve local sub-problems through Decomposition-Coordination and obtains the solution of the large global problem by coordinating the solution of the sub-problems. ADMM was first proposed by Glowinski, Marrocco [14], and Gabay, Mercier [12] in 1975 and 1976, respectively, and was reviewed by Boyd et al. in 2011 [5] and proved that it is suitable for large-scale distributed optimization problems.

The classic ADMM can solve the optimization problem with equality constraints of the form:

$$\begin{aligned}\min \quad & f(\mathbf{x}) + g(\mathbf{z})\\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} = \mathbf{c}\end{aligned} \tag{2.17}$$

where $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{z} \in \mathbb{R}^m$, $\mathbf{A} \in \mathbb{R}^{d\times n}$, $\mathbf{B} \in \mathbb{R}^{d\times m}$, $\mathbf{c} \in \mathbb{R}^d$. To solve this optimization problem, the first step is to introduce the augmented Lagrangian function defined as:

$$L_c(\mathbf{x}, \mathbf{z}; \boldsymbol{\lambda}) = f(\mathbf{x}) + g(\mathbf{z}) + \boldsymbol{\lambda}^\top(\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}) + \frac{\rho}{2}\|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}\|_2^2 \tag{2.18}$$

where $\boldsymbol{\lambda} \in \mathbb{R}^d$ is the Lagrange multiplier and ρ is a penalty parameter. After adding the latter quadratic term $\frac{\rho}{2}\|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}\|_2^2$, we no longer need to assume that f is strictly convex or bounded. For example, even if f is linear, we can calculate the extreme value. And it makes our solving process more smooth, making the iterative process more robust and performing better on convergence. Adding this item will not change the primal problem because the constraints of the problem constrain it and finally get the

solution that satisfies $\mathbf{Ax} + \mathbf{Bz} = \mathbf{c}$. Therefore, it is equivalent to the primal problem. The iteration steps of ADMM can be described as:

$$\begin{cases} \mathbf{x}^{k+1} = \operatorname{argmin}_{\mathbf{x}} L_c\left(\mathbf{x}, \mathbf{z}^k, \boldsymbol{\lambda}^k\right) \\ \mathbf{z}^{k+1} = \operatorname{argmin}_{\mathbf{z}} L_c\left(\mathbf{x}^{k+1}, \mathbf{z}, \boldsymbol{\lambda}^k\right) \\ \boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \rho(\mathbf{Ax} + \mathbf{Bz} - \mathbf{c}). \end{cases} \tag{2.19}$$

Sometimes, the objective function should be divided into $N (N > 2)$ blocks, and the corresponding optimization problem is as follows:

$$\begin{array}{ll} \min & f_1(\mathbf{x}_1) + f_2(\mathbf{x}_2) + \cdots + f_N(\mathbf{x}_N) \\ \text{s.t.} & \mathbf{A}_1\mathbf{x}_1 + \mathbf{A}_2\mathbf{x}_2 + \cdots + \mathbf{A}_N\mathbf{x}_N = \mathbf{c} \\ & \mathbf{x}_i \in \mathcal{X}_i, i = 1, \ldots, N \end{array} \tag{2.20}$$

where $\mathbf{A}_i \in \mathbb{R}^{d \times n_i}$, $\mathbf{c} \in \mathbb{R}^d$, and $\mathcal{X}_i \subset \mathbb{R}^{n_i}$. The multi-block ADMM algorithm for solving this optimization problem is also based on the augmented Lagrangian function, which is defined as:

$$L_c(\mathbf{x}_1, \ldots, \mathbf{x}_N; \boldsymbol{\lambda}) := \sum_{j=1}^{N} f_j(\mathbf{x}_j) - \left\langle \boldsymbol{\lambda}, \sum_{j=1}^{N} \mathbf{A}_j\mathbf{x}_j - \mathbf{c} \right\rangle + \frac{\rho}{2} \left\| \sum_{j=1}^{N} \mathbf{A}_j\mathbf{x}_j - \mathbf{c} \right\|_2^2, \tag{2.21}$$

where $\langle , \rangle$ is the inner product between two vectors. The standard multi-block ADMM algorithm iteration steps for solving this problem can be described as follows:

$$\begin{cases} \mathbf{x}_1^{k+1} := \operatorname{argmin}_{\mathbf{x}_1 \in \mathcal{X}_1} L_c\left(\mathbf{x}_1, \mathbf{x}_2^k, \ldots, \mathbf{x}_N^k; \boldsymbol{\lambda}^k\right) \\ \mathbf{x}_2^{k+1} := \operatorname{argmin}_{\mathbf{x}_2 \in \mathcal{X}_2} L_c\left(\mathbf{x}_1^{k+1}, \mathbf{x}_2, \mathbf{x}_3^k, \ldots, \mathbf{x}_N^k; \boldsymbol{\lambda}^k\right) \\ \quad \vdots \\ \mathbf{x}_N^{k+1} := \operatorname{argmin}_{\mathbf{x}_N \in \mathcal{X}_N} L_c\left(\mathbf{x}_1^{k+1}, \mathbf{x}_2^{k+1}, \ldots, \mathbf{x}_{N-1}^{k+1}, \mathbf{x}_N; \boldsymbol{\lambda}^k\right) \\ \boldsymbol{\lambda}^{k+1} := \boldsymbol{\lambda}^k - \rho\left(\sum_{j=1}^{N} \mathbf{A}_j\mathbf{x}_j^{k+1} - \mathbf{c}\right) \end{cases} \tag{2.22}$$

Standard ADMM minimizes the augmented Lagrangian function with respect to $\mathbf{x}_1, \ldots, \mathbf{x}_N$ alternately. When $N = 2$, it degenerates into the 2-block ADMM and has been extensively studied. The convergence property of 2-block ADMM has been shown in [5]. However, the convergence of multi-block ADMM needs to be discussed case by case. For example, paper [16] shows that if the functions f_1, f_2,..., f_N are strongly convex, the multi-block ADMM algorithm converges.

2.4. Unrolling networks

Deep learning is a data-driven technology, which means that the performance of an algorithm is directly dependent on the quantity and quality of the training data.

The deep learning network has many layers, which can theoretically be mapped to any function. Therefore, it can solve many highly complex problems. In general, deep learning algorithms will perform better if the data is more significant and of better quality. These advantages make deep learning widely used in more and more fields. However, deep learning algorithms also have some flaws. First, they often lack interpretability. The black-box nature of deep networks makes it impossible for researchers to know how their internal structure relates to the problem they need to solve. This characteristic is not conducive to designing effective deep neural networks and makes most neural network designs based on experience and trial and error. Second, the algorithm's generalization needs to be improved. Deep learning algorithms are purely data-driven; without high-quality data, the algorithm's performance can be significantly reduced. For example, high-quality data is often lacking in medical imaging [19] and 3D reconstruction [37] tasks. Data sets for medical imaging tasks often involve patient privacy, making obtaining a large amount of training data challenging. This results in poor performance of deep learning algorithms in these fields, even inferior to traditional algorithms sometimes.

Compared with deep learning algorithms, conventional iterative algorithms tend to have higher interpretability because they often acquire prior knowledge of the problem. However, not all prior knowledge can be obtained easily. To get this prior knowledge, we often need to combine relevant knowledge in specific fields, which generally requires professional expertise. However, some prior knowledge is even unknown, which makes the conventional iterative algorithms challenging. The recent rise of unrolling networks combines traditional iterative algorithms with deep learning. The unrolling network combines the learning ability of deep networks and the high interpretability of iterative algorithms. Gregor and LeCun [15] first formally proposed the concept of unrolling networks and combined iterative algorithms for sparse coding with neural networks. Since then, more research on unrolling networks has been explored, covering areas such as speech processing [17], computational imaging [56], vision and recognition [30], medical imaging [51], smart power grid [64], and remote sensing [31].

Paper [17] designs an unrolling network based on a nonnegative matrix factorization algorithm. Experimental results on speech enhancement tasks show that unrolling networks with fewer parameters still produce results close to traditional neural networks. Paper [56] builds an unrolling network based on sparse coding for image super-resolution. The proposed cascade of sparse coding based network model has higher accuracy, faster computation speed, and a more compact model structure in solving super-resolution problems. In [30], the unrolling network is applied to solve the problem of semantic image segmentation. Liu et al. unroll the Markov Random Field algorithm into a deep network called Deep Parsing Network (DPN). DPN achieves the best performance on benchmark datasets, cityscape, CamVid, etc. Paper [51] proposes

an unrolling network that combines the robust principle component analysis algorithm with deep learning. This unrolling network is applied to preprocess the signal in a super-resolution ultrasound problem. The experimental results show that compared with the fast iterative shrinkage algorithm and the data-driven deep network, the unrolling network achieves speedier convergence and more accurate results. In paper [64], Zhang et al. apply the unrolling algorithm to power system state estimation. The proposed model in this paper, prox-linear nets, is based on double-loop prox–linear iterations. Its performance on the IEEE 118-bus system is better than nearly all its competitors, including the Gauss–Newton solver.

The core of the unrolling network is an iterative algorithm, and different iterative algorithms often produce various unrolling networks. Frequently used iterative algorithms include nonnegative matrix factorization (NMF), coupled sparse coding with the ISTA, ADMM, primal-dual hybrid gradient, proximal gradient descent, half-quadratic splitting, and projected gradient descent. Specifically, the unrolling network converts each iteration in the iterative algorithm into a one-layer unrolling network. The parameters appearing in the iterative algorithm are converted into learnable parameters. The unrolling network thus constructed can also be trained using backpropagation, thereby optimizing the parameters in the network. In addition to higher interpretability, another advantage of unrolling the network is the higher computational efficiency. They tend to have fewer parameters than traditional neural networks, which makes their training more efficient and requires relatively less training data.

2.5. Graph unrolling networks

2.5.1. Related works

Recently, unrolling algorithms have been applied to graph signal processing. The paper in [8] applies the unrolling network to the task of graph signal denoising. They solve the graph sparse coding and graph trend filtering problems. After giving the underlying iterative algorithm, the authors replace the fixed parameters in the iterative algorithm with learnable parameters. To train the constructed unrolling network, this paper adopts an unsupervised training method to update the parameters in the network. The unrolling network can achieve a better denoising effect than conventional iterative algorithms and standard graph neural networks. The paper in [7] designs a graph unrolling network for the inpainting task of time-varying graph signals. Each layer corresponds to one iteration of the original iterative algorithm. The regularization term is replaced with a trainable function. This paper moves away from the manual selection of graph signal priors, a classic example of adaptive learning of priors. Paper [53] models the image denoising problem from a graph perspective and constructs a graph unrolling network. It minimizes the graph's total variation to denoise the signal. They give an analytical solution to this optimization problem while

replacing the fixed parameters in the iterative algorithm with trainable parameters in the unrolling network. The experimental results show that this method is close to the state-of-the-art DnCNN. However, the required parameters are 80% less than DnCNN. This dramatically saves computing resources. In [36], the authors apply the ADMM algorithm and Plug-and-Play ADMM (PnP-ADMM) algorithm to restore the graph signal. Based on these two algorithms, they construct the corresponding graph unrolling network. The unrolling network is then trained in a supervised manner.

2.5.2. Graph unrolling network via trend filtering

Consider a graph $G = (V, E, \mathbf{A})$ with vertices $V = \{v_n\}_{n=1}^{N}$, and undirected edges $E = \{e_m\}_{m=1}^{M}$. Let $\mathbf{A} \in \mathbb{R}^{N \times N}$ be the adjacency matrix of this graph. A graph signal is a vector $\mathbf{x} = [x_1, x_2, ..., x_N]^\top$ and the nth element x_n is indexed by the vertex v_n. The univariate trend filtering problem provides an effective idea for solving the problem of graph signal smoothing. Consider a graph signal with noise

$$\mathbf{t} = \mathbf{x} + \mathbf{n} \tag{2.23}$$

where $\mathbf{x}$ is the ground truth graph signal, $\mathbf{n}$ is the noise and $\mathbf{t}$ is the measurement. Graph signal denoising aims to recover the $\mathbf{x}$ from $\mathbf{t}$. It should first be assumed that part of the prior information of the real graph signal is available. The penalty term makes the recovered signal as close to the prior knowledge as possible. This problem can be formulated as an optimization problem, and the kth order graph trend filtering estimate is defined as [55]:

$$\hat{\mathbf{x}} = \underset{\mathbf{x} \in \mathbb{R}^N}{\arg\min} \frac{1}{2} \|\mathbf{x} - \mathbf{t}\|_2^2 + \alpha \|\mathbf{B}^{(k+1)}\mathbf{x}\|_1 \tag{2.24}$$

where $\mathbf{B}$ is the oriented graph incidence matrix. If $e_l = (i, j)$, the lth row of $\mathbf{B}$ is defined as

$$\mathbf{B}_l = (0, ... \underset{\uparrow}{-1}_{i}, ... \underset{\uparrow}{1}_{j}, ...0) \tag{2.25}$$

When $k = 0$, the penalty term is the first-order graph difference operator, which is the optimization problem:

$$\begin{aligned} &\min_{\mathbf{x} \in \mathbb{R}^N} \tfrac{1}{2} \|\mathbf{t} - \mathbf{x}\|_2^2 + \alpha \|\mathbf{y}\|_1 \\ &\text{subject to } \mathbf{y} = \mathbf{B}\mathbf{x} \end{aligned} \tag{2.26}$$

The regularization term is the sum of the absolute difference between connected node pairs which encourages the recovered graph signal to be piecewise-linear [52]. When $k \geq 1$, the higher order graph difference operator can be defined in a recursive way

$$\mathbf{B}^{(k+1)} = \begin{cases} (\mathbf{B})^\top \mathbf{B}^{(k)} = \mathbf{L}^{\frac{k+1}{2}} & \text{for odd k} \\ \mathbf{B}\mathbf{B}^{(k)} = \mathbf{B}^{(1)}\mathbf{L}^{\frac{k}{2}} & \text{for even k} \end{cases} \tag{2.27}$$

where $\mathbf{L}$ is the unnormalized Laplacian matrix.The penalty function of the optimization problem (2.26) is

$$J = \frac{1}{2}\|\mathbf{t} - \mathbf{x}\|_2^2 + \alpha\|\mathbf{y}\|_1 + \frac{\mu}{2}\|\mathbf{y} - \mathbf{B}\mathbf{x}\|_2^2 \tag{2.28}$$

The update equation obtained by the method of alternating minimization based on the half-quadratic splitting algorithm is [27]

$$\begin{cases} \mathbf{x}^{(k+1)} = (I + \mu\mathbf{B}^\top\mathbf{B})^{-1}(\mathbf{t} + \mu\mathbf{B}^\top\mathbf{y}^{(k)}) \\ \mathbf{y}^{(k+1)} = \arg\min \frac{\mu}{2}\|\mathbf{y} - \mathbf{B}\mathbf{x}^{(k+1)}\|_2^2 + \alpha\|\mathbf{y}\|_1 \end{cases} \tag{2.29}$$

To build an unrolling network, the iterative parameters in (2.29) are replaced with the trainable graph convolution. Notice that the $\boldsymbol{y}$-update in (2.29) has an analytical solution. The lth layer of the corresponding unrolling network for (2.29) can be computed as

$$\begin{cases} \mathbf{X}^{(l+1)} = \mathbf{H}_1\mathbf{T} + \mathbf{H}_2(\mathbf{B}^\top\mathbf{Y}^{(l)}) \\ \mathbf{Y}^{(l+1)} = S_\delta(\mathbf{B}\mathbf{X}^{(l+1)}) \end{cases} \tag{2.30}$$

where $\mathbf{T} \in \mathbb{R}^{N\times K}$ is the measurements matrix and $\mathbf{H}_1$ and $\mathbf{H}_2$ are graph convolution operators with trainable coefficients. The learnable parameters of the graph convolution can be shared or be independent between layers. We set the parameters to be shared, which helps to reduce the computational complexity. $S_\delta(\cdot)$ is a soft-thresholding function which is defined as [5]

$$S_\delta(a) = \begin{cases} a - \delta & \text{if } a > -\delta \\ 0 & \text{if } -\delta \le a \le \delta \\ a + \delta & \text{if } a < -\delta \end{cases} \tag{2.31}$$

Consider each iteration as a layer in the unrolling network. The overall graph unrolling trend filtering algorithm is shown in Table 2.1.

Table 2.1: algorithm of graph unrolling trend filtering

Algorithm	Graph Unrolling Networks via Trend Filtering
Parameters	$\mathbf{T} \in \mathbb{R}^{N\times K}$: Original measurements
	$\mathbf{X}_0 \in \mathbb{R}^{N\times K}$: Clean signal
	$\mathbf{A} \in \mathbb{R}^{N\times N}$: Adjacency matrix
	L: Number of unrolling layers
	δ: Thresholding value
	$\mathbf{B} \in \mathbb{R}^{M\times N}$: Oriented incidence matrix
Estimation	$\mathbf{X} \in \mathbb{R}^{N\times K}$: The denoised graph signal
Function	**GUTF**($\mathbf{T}$,$\mathbf{A}$,L,$\mathbf{B}$)
	$\mathbf{X}^{(0)} \leftarrow \mathbf{0}$
	for $l = 1 : L$
	$\quad\mathbf{Y}^{(l)} \leftarrow S_\delta(\mathbf{B}\boldsymbol{X}^{(l-1)})$
	$\quad\mathbf{X}^{(l)} \leftarrow \mathbf{H}_1\mathbf{T} + \mathbf{H}_2(\mathbf{B}^\top\mathbf{Y}^{(l-1)})$
	end
	$\mathbf{X} \leftarrow \mathbf{X}^{(L)}$
	minimize $\|\mathbf{X} - \mathbf{X}_0\|_F^2$ and update all weights
	Return X

For the graph signal denoising problem, the smoothness assumption is normally considered to hold. This means that signals on connected nodes tend to have close values. When this assumption does not hold, graph trend filtering and its corresponding unrolling network will no longer fit this denoising problem.

2.6. Summary

This chapter provides a comprehensive introduction and definition of the background concepts needed for this topic. The object of study is the simplicial complexes, a higher-order network. Specifically, this thesis studies higher-order signals defined on simplicial complexes. The algorithms used in this paper include the ADMM algorithm and the unrolling algorithm.

3

Problem formulation

Consider the edge flow reconstruction task from noisy or partial measurements $\mathbf{y}$. The goal is to estimate an edge flow signal $\hat{\mathbf{f}}$ from measurements $\mathbf{y}$ by leveraging particular prior of the edge flows w.r.t. the underlying simplex such as curl-free or divergence-free. This can be framed as a regularized optimization problem

$$\begin{aligned} &\underset{\hat{\mathbf{f}}\in\mathbb{R}^{N_1}}{\operatorname{argmin}} \quad \|\hat{\mathbf{f}}-\mathbf{y}\|_2^2 + \textstyle\sum_{i=1}^{n} r_i(\hat{\mathbf{f}}, \mathcal{S}) \\ &\text{subject to} \qquad\quad \mathbf{P}\hat{\mathbf{f}} = \mathbf{P}\mathbf{y} \end{aligned} \tag{3.1}$$

where $\|\hat{\mathbf{f}}-\mathbf{y}\|_2^2$ is the data-fitting term that forces the recovered signal to be close to the measurements. The terms $r_i(\hat{\mathbf{f}}, \mathcal{S})$ are the regularizers, which are monotone non-decreasing functions penalizing particular behavior of the edge flows w.r.t. the 1-simplex. All regularizers are designed based on the prior knowledge of the edge flow, such as the properties of curl-free or divergence-free. $\mathbf{P}\hat{\mathbf{f}} = \mathbf{P}\mathbf{y}$ adds particular constraints on the observed signal and the reconstructed ones. We will detail problem (3.1) to two particular settings: i) signal denoising when all edge flows are observed, but the measurements are noisy; ii) signal reconstruction, when the edge flows are observed noiseless on a subset of edges.

3.1. Denoising simplicial edge flow

For the denoising task, we can assume a noisy model as $\mathbf{y} = \mathbf{f}_0 + \mathbf{n}$, where $\mathbf{f}_0$ is the real edge flow and $\mathbf{n}$ is the additive Gaussian noise. When $\mathbf{P} = \mathbf{0}$ and $\mathbf{y}$ is a noisy edge flow, this is an optimization problem for the denoising task. The goal is to recover the real $\mathbf{f}_0$ from the noisy measurements $\mathbf{y}$.

3.2. Interpolation for simplicial edge flow

When $\mathbf{P} \in \{0,1\}^{M \times N_1}$ is the sampling matrix, and $\mathbf{y}$ is the edge flow with M sampled non-zero values, and $N_1 - M$ missing values, (3.1) is an optimization problem for interpolation. It is worth noting that (3.1) is an accurate optimization problem for denoising, while it is an approximation for interpolation. The reason is that the interpolation does not require the data-fitting term. To make the interpolation and denoising formally uniform, we keep the data-fitting term in the interpolation and adapt it to the interpolation by adjusting the coefficients of regularizers. When the coefficients of the regularizers are much larger than the coefficient of the data-fitting term, the influence of the data-fitting term on this optimization problem can be ignored.

4

Methodology

The regularizer in an optimization problem can affect the quality of the solution. Different regularizers generate solutions with different properties. The innovation of this paper is to combine ℓ_1 and ℓ_2 based regularizers to design a more flexible optimization problem, which is an ElasticNet problem. This ElasticNet problem is solved by a multi-block ADMM iterative algorithm.

4.1. Simplicial ElasticNet Problem

To regularize problem (3.1) with simplicial prior, we consider an ElasticNet principle w.r.t. the three signal components in (2.12), which has both ℓ_1 and ℓ_2 norm regularizers as follows

$$\begin{aligned}\underset{\hat{\mathbf{f}}\in\mathbb{R}^{N_1}}{\operatorname{argmin}} &\|\hat{\mathbf{f}}-\mathbf{y}\|_2^2+\alpha_1\|\mathbf{B}_1\hat{\mathbf{f}}\|_1+\alpha_2\|\mathbf{B}_1\hat{\mathbf{f}}\|_2^2+\beta_1\|\mathbf{B}_2^\top\hat{\mathbf{f}}\|_1\\ &+\beta_2\|\mathbf{B}_2^\top\hat{\mathbf{f}}\|_2^2+\gamma_1\|\hat{\mathbf{f}}\|_1+\gamma_2\|\hat{\mathbf{f}}\|_2^2 \qquad (4.1)\\ \text{subject to}\quad & \mathbf{P}\hat{\mathbf{f}}=\mathbf{P}\mathbf{y}\end{aligned}$$

where $\alpha_1, \alpha_2, \beta_1, \beta_2, \gamma_1, \gamma_2$ are all positive constants. There are three tuples of ElasticNet regularizers in (4.1) that are reminiscent of the Hodge decomposition in (2.12):

- The first tuple $\alpha_1\|\mathbf{B}_1\hat{\mathbf{f}}\|_1+\alpha_2\|\mathbf{B}_1\hat{\mathbf{f}}\|_2^2$ regularizes the divergence component of the edge flows by promoting a sparse divergence via the ℓ_1 norm and a low-energy divergence via the ℓ_2 norm.
- The second tuple $\beta_1\|\mathbf{B}_2^\top\hat{\mathbf{f}}\|_1+\beta_2\|\mathbf{B}_2^\top\hat{\mathbf{f}}\|_2^2$ regularizes the curl component of the edge flows. The ℓ_1 norm promotes the sparsity of the curl on the triangles: that is, it forces the recovered signal to have more triangles with zero curls. The ℓ_2 norm reduces the total curl of recovered signal globally.

- The last tuple $\gamma_1\|\hat{\mathbf{f}}\|_1 + \gamma_2\|\hat{\mathbf{f}}\|_2^2$ are additional regularizers that guarantee the completeness of the optimization problem. In some tasks, these two terms are redundant. In this case, γ_1 and γ_2 can be set to zero.

Scalars $\alpha_1, \alpha_2, \beta_1, \beta_2, \gamma_1$, and γ_2 control the trade-off between the fidelity, divergence, and curl of the recovered signals. For example, when α_1 and α_2 are large, the denoised edge flow tends to be divergence-free, and when β_1 and β_2 are large, the denoised edge flow tends to be curl-free. Problem (4.1) generalizes two existing edge flow reconstruction problems.

4.1.1. Special case 1: Tikhonov regularizer

When the parameters $\alpha_1 = \beta_1 = \gamma_1 = \gamma_2 = 0$, a common optimization problem for the edge flow recovery becomes

$$\begin{aligned} &\underset{\hat{\mathbf{f}}\in\mathbb{R}^{N_1}}{\operatorname{argmin}} \|\hat{\mathbf{f}} - \mathbf{y}\|_2^2 + \alpha_2\|\mathbf{B}_1\hat{\mathbf{f}}\|_2^2 + \beta_2\|\mathbf{B}_2^\top\hat{\mathbf{f}}\|_2^2 \\ &\text{subject to} \qquad \mathbf{P}\hat{\mathbf{f}} = \mathbf{P}\mathbf{y} \end{aligned} \tag{4.2}$$

where the regularizers force the recovered signal to have a low divergence and curl. Regularizers $\|\mathbf{B}_1\hat{\mathbf{f}}\|_2^2$ represents the total squared divergence penalty of the nodes. The second regularization term $\|\mathbf{B}_2^\top\hat{\mathbf{f}}\|_2^2$ represents the total squared curl penalty of the triangles. Thus, the recovered signal obtained by this optimal solution tends to be divergence-free or curl-free. However, the properties of the ℓ_2 norm regularizers are suboptimal for some tasks as they can only reduce the divergence and curl of the reconstructed signal globally but cannot reconstruct the divergence-free and curl-free signal exactly.

4.1.2. Special case 2: Simplicial trend filtering

When we consider only the sparsity of divergence and curl of the edge flow, the ℓ_1 norm regularizer should be considered. Setting $\alpha_2 = \beta_2 = \gamma_1 = \gamma_2 = 0$, the simplicial ElasticNet reduces to the simplicial trend filtering problem

$$\begin{aligned} &\underset{\hat{\mathbf{f}}\in\mathbb{R}^{N_1}}{\operatorname{argmin}} \|\hat{\mathbf{f}} - \mathbf{y}\|_2^2 + \alpha_1\|\mathbf{B}_1\hat{\mathbf{f}}\|_1 + \beta_1\|\mathbf{B}_2^\top\hat{\mathbf{f}}\|_1 \\ &\text{subject to} \qquad \mathbf{P}\hat{\mathbf{f}} = \mathbf{P}\mathbf{y} \end{aligned} \tag{4.3}$$

When the prior knowledge curl-free or divergence-free is explicit, simplicial trend filtering is more advantageous compared to Tikhonov regularization. The properties of the ℓ_2 norm regularizers are suboptimal for some tasks because they can only reduce the divergence and curl of the reconstructed signal globally but cannot reconstruct the divergence-free and curl-free edge flow exactly.

The ElasticNet takes the benefit of both regularizers, and it is of interest when: (i) the data are approximately curl-free or divergence-free; (ii) the noise level is comparable to that of the clean signal; and (iii) excessive flows that are missing. The reason is that the ElasticNet makes a trade between the ℓ_1 norm and ℓ_2 norm of the regularizers.

4.2. ADMM Solution for Simplicial ElasticNet

Optimization problem (4.1) is a convex problem and there are several separable blocks in its objective function. This optimization problem can be solved by a multi-block ADMM algorithm. Consider the auxiliary variables $\mathbf{z}_1 = \mathbf{B}_1\hat{\mathbf{f}}$, $\mathbf{z}_2 = \mathbf{B}_2^\top\hat{\mathbf{f}}$ and $\mathbf{z}_3 = \hat{\mathbf{f}}$ for the regularizers so that to reformulate problem (4.1) as

$$\begin{aligned}\underset{\hat{\mathbf{f}}\in\mathbb{R}^{N_1}}{\operatorname{argmin}} &\|\hat{\mathbf{f}}-\mathbf{y}\|_2^2 + \alpha_1\|\mathbf{z}_1\|_1 + \alpha_2\|\mathbf{z}_1\|_2^2 + \beta_1\|\mathbf{z}_2\|_1 \\ &+ \beta_2\|\mathbf{z}_2\|_2^2 + \gamma_1\|\mathbf{z}_3\|_1 + \gamma_2\|\mathbf{z}_3\|_2^2 \\ \text{subject to} \quad &\mathbf{B}_1\hat{\mathbf{f}} = \mathbf{z}_1, \mathbf{B}_2^\top\hat{\mathbf{f}} = \mathbf{z}_2, \hat{\mathbf{f}} = \mathbf{z}_3, \mathbf{P}\hat{\mathbf{f}} = \mathbf{P}\mathbf{y}.\end{aligned} \tag{4.4}$$

The corresponding augmented Lagrangian function is defined as

$$\begin{aligned}L =&\|\hat{\mathbf{f}}-\mathbf{y}\|_2^2 + \alpha_1\|\mathbf{z}_1\|_1 + \alpha_2\|\mathbf{z}_1\|_2^2 + \beta_1\|\mathbf{z}_2\|_1 + \beta_2\|\mathbf{z}_2\|_2^2 \\ &+ \gamma_1\|\mathbf{z}_3\|_1 + \gamma_2\|\mathbf{z}_3\|_2^2 - \boldsymbol{\lambda}_1^\top(\mathbf{B}_1\hat{\mathbf{f}}-\mathbf{z}_1) - \boldsymbol{\lambda}_2^\top(\mathbf{B}_2^\top\hat{\mathbf{f}}-\mathbf{z}_2) \\ &- \boldsymbol{\lambda}_3^\top(\hat{\mathbf{f}}-\mathbf{z}_3) - \boldsymbol{\lambda}_4^\top(\mathbf{P}\hat{\mathbf{f}}-\mathbf{P}\mathbf{y}) + \frac{\rho}{2}\|\mathbf{B}_1\hat{\mathbf{f}}-\mathbf{z}_1\|_2^2 \\ &+ \frac{\rho}{2}\|\mathbf{B}_2^\top\hat{\mathbf{f}}-\mathbf{z}_2\|_2^2 + \frac{\rho}{2}\|\hat{\mathbf{f}}-\mathbf{z}_3\|_2^2 + \frac{\rho}{2}\|\mathbf{P}\hat{\mathbf{f}}-\mathbf{P}\mathbf{y}\|_2^2\end{aligned} \tag{4.5}$$

where $\boldsymbol{\lambda}_1 \in \mathbb{R}^{N_0}$, $\boldsymbol{\lambda}_2 \in \mathbb{R}^{N_2}$, $\boldsymbol{\lambda}_3 \in \mathbb{R}^{N_1}$ and $\boldsymbol{\lambda}_4 \in \mathbb{R}^{M}$ are the Lagrangian multipliers and ρ is the penalty parameter. There are four blocks in this problem: $\|\hat{\mathbf{f}}-\mathbf{y}\|_2^2$; $\alpha_1\|\mathbf{z}_1\|_1 + \alpha_2\|\mathbf{z}_1\|_2^2$; $\beta_1\|\mathbf{z}_2\|_1 + \beta_2\|\mathbf{z}_2\|_2^2$ and $\gamma_1\|\mathbf{z}_3\|_1 + \gamma_2\|\mathbf{z}_3\|_2^2$. The iterative steps of

four-block ADMM comprise

$$\begin{cases} \hat{\mathbf{f}}^{(k+1)} = & (2\mathbf{I} + \rho\mathbf{B}_1^\top\mathbf{B}_1 + \rho\mathbf{B}_2\mathbf{B}_2^\top + \rho\mathbf{I} + \rho\mathbf{P}^\top\mathbf{P})^{-1} \\ & \quad (2\mathbf{y} + \mathbf{B}_1^\top\boldsymbol{\lambda}_1^{(k)} + \mathbf{B}_2\boldsymbol{\lambda}_2^{(k)} + \boldsymbol{\lambda}_3^{(k)} + \mathbf{P}^\top\boldsymbol{\lambda}_4^{(k)} \\ & \quad + \rho\mathbf{B}_1^\top\mathbf{z}_1^{(k)} + \rho\mathbf{B}_2\mathbf{z}_2^{(k)} + \rho\mathbf{z}_3^{(k)} + \rho\mathbf{P}^\top\mathbf{P}\mathbf{y}) \\ \mathbf{z}_1^{(k+1)} = & S_{\frac{\alpha_1}{2\alpha_2+\rho}}\left(\frac{1}{2\alpha_2+\rho}(\rho\mathbf{B}_1\mathbf{f}^{(k+1)} - \boldsymbol{\lambda}_1^{(k)})\right) \\ \mathbf{z}_2^{(k+1)} = & S_{\frac{\beta_1}{2\beta_2+\rho}}\left(\frac{1}{2\beta_2+\rho}(\rho_2\mathbf{B}_2^\top\mathbf{f}^{(k+1)} - \boldsymbol{\lambda}_2^{(k)})\right) \\ \mathbf{z}_3^{(k+1)} = & S_{\frac{\gamma_1}{2\gamma_2+\rho}}\left(\frac{1}{2\gamma_2+\rho}(\rho_3\mathbf{f}^{(k+1)} - \boldsymbol{\lambda}_3^{(k)})\right) \\ \boldsymbol{\lambda}_1^{(k+1)} = & \boldsymbol{\lambda}_1^{(k)} - \rho(\mathbf{B}_1\mathbf{f}^{(k+1)} - \mathbf{z}_1^{(k+1)}) \\ \boldsymbol{\lambda}_2^{(k+1)} = & \boldsymbol{\lambda}_2^{(k)} - \rho(\mathbf{B}_2^\top\mathbf{f}^{(k+1)} - \mathbf{z}_2^{(k+1)}) \\ \boldsymbol{\lambda}_3^{(k+1)} = & \boldsymbol{\lambda}_3^{(k)} - \rho(\mathbf{f}^{(k+1)} - \mathbf{z}_3^{(k+1)}) \\ \boldsymbol{\lambda}_4^{(k+1)} = & \boldsymbol{\lambda}_4^{(k)} - \rho(\mathbf{P}\mathbf{f}^{(k+1)} - \mathbf{P}\mathbf{y}) \end{cases} \tag{4.6}$$

where $S_\delta(\cdot)$ is the element-wise soft-thresholding function with threshold δ. The following proposition provides a sufficient condition for the convergence of the four-block ADMM.

Proposition 1 (convergence): Given that each block in the cost function is a strongly convex function and that its constants are μ_i satisfies $\mu_1 = 2, \mu_2 = 2\alpha_2, \mu_3 = 2\beta_2, \mu_4 = 2\gamma_2$. Consider also the matrices $\mathbf{A}_i$ for equality constraints of (4.4) defined as

$$\mathbf{A}_1 = \left[\mathbf{B}_1, \mathbf{B}_2^\top, \mathbf{I}, \mathbf{P}\right] \in \mathbb{R}^{(N_0+N_1+N_2+M)\times N_1} \tag{4.7a}$$

$$\mathbf{A}_2 = [-\mathbf{I},\ \mathbf{0},\ \mathbf{0},\ \mathbf{0}] \in \mathbb{R}^{(N_0+N_1+N_2+M)\times N_0} \tag{4.7b}$$

$$\mathbf{A}_3 = [\mathbf{0},\ -\mathbf{I},\ \mathbf{0},\ \mathbf{0}] \in \mathbb{R}^{(N_0+N_1+N_2+M)\times N_2} \tag{4.7c}$$

$$\mathbf{A}_4 = [\mathbf{0},\ \mathbf{0},\ -\mathbf{I},\ \mathbf{0}] \in \mathbb{R}^{(N_0+N_1+N_2+M)\times N_1}. \tag{4.7d}$$

If the penalty parameter ρ satisfies

$$0 < \rho < \min_{1\le i\le 4}\left\{\frac{2\mu_i}{9\|\mathbf{A}_i\|_2^2}\right\}, \tag{4.8}$$

the four-block ADMM iterative steps converge to the optimal solution of the problem (4.1).

Proof. See Appendix A.

4.3. Simplicial unrolling networks

4.3.1. Simplicial Unrolling Network For ElasticNet

Choosing an appropriate regularization coefficient is critical to achieving a satisfactory performance by solving the problem (4.1). However, in practice, such prior knowledge

may be unavailable or unclear to be framed as an explicit regularizer. The core idea of building an unrolling network is to map each iteration of the iterative algorithm into one neural network layer and replace the fixed parameters with learnable ones [33]. Therefore, the unrolling network leads to a specific architecture that is tailored to the problem at hand; hence, it restricts the degrees of freedom compared with a conventional solution and, ultimately, demands less training data. We here propose two variants in this report.

Variant 1. The core idea of constructing the unrolling network is to replace some of the steps in the original iterative algorithm with trainable parameters. If we replace the $\hat{\mathbf{f}}$-update, $\mathbf{z}$-update and $\boldsymbol{\lambda}_i$-update steps in (4.6), we obtain:

$$\begin{cases} \hat{\mathbf{f}}^{(l+1)} = (2\mathbf{I}/\rho + \mathbf{B}_1^\top \mathbf{B}_1 + \mathbf{B}_2\mathbf{B}_2^\top + \mathbf{I} + \mathbf{P}^\top\mathbf{P})^{-1} \\ \qquad\qquad (2\mathbf{y}/\rho + \mathbf{B}_1^\top \mathbf{q}_1^{(l)} + \mathbf{B}_2\mathbf{q}_2^{(l)} + \mathbf{q}_3^{(l)} + \mathbf{P}^\top\mathbf{q}_4^{(l)} \\ \qquad\qquad + \mathbf{B}_1^\top \mathbf{z}_1^{(l)} + \mathbf{B}_2\mathbf{z}_2^{(l)} + \mathbf{z}_3^{(l)} + \mathbf{P}^\top\mathbf{P}\mathbf{y}) \\ \mathbf{z}_1^{(l+1)} = S_1(\mathbf{B}_1\hat{\mathbf{f}}^{(l+1)} - \mathbf{q}_1^{(l)}; \theta_1) \\ \mathbf{z}_2^{(l+1)} = S_2(\mathbf{B}_2^\top\hat{\mathbf{f}}^{(l+1)} - \mathbf{q}_2^{(l)}; \theta_2) \\ \mathbf{z}_3^{(l+1)} = S_3(\hat{\mathbf{f}}^{(l+1)} - \mathbf{q}_3^{(l)}; \theta_3) \\ \mathbf{q}_1^{(l+1)} = \mathbf{q}_1^{(l)} - (\mathbf{B}_1\mathbf{f}^{(l+1)} - \mathbf{z}_1^{(l+1)}) \\ \mathbf{q}_2^{(l+1)} = \mathbf{q}_2^{(l)} - (\mathbf{B}_2^\top\mathbf{f}^{(l+1)} - \mathbf{z}_2^{(l+1)}) \\ \mathbf{q}_3^{(l+1)} = \mathbf{q}_3^{(l)} - (\mathbf{f}^{(l+1)} - \mathbf{z}_3^{(l+1)}) \\ \mathbf{q}_4^{(l+1)} = \mathbf{q}_4^{(l)} - (\mathbf{P}\mathbf{f}^{(l+1)} - \mathbf{P}\mathbf{y}) \end{cases} \tag{4.9}$$

where S_i are the soft-thresholding functions with learnable parameters θ_i and $\mathbf{q}_i = \boldsymbol{\lambda}_i/\rho$ is an auxiliary variable formulates the iterative steps in a more organized way. Then we set the parameter ρ to be a function of layer l as $\rho(l)$. The overall learning framework of the four-block ADMM is given in Algorithm 1.

Variant 2. Algorithm 1 fixes the parameter $\rho(l)$ as a function of layer l. We can also set the ρ to be learnable weights. For instance, another straightforward way is to directly replace the fixed parameters α_1, α_2, β_1, β_2, γ_1, γ_2 and ρ with trainable parameters $a_1, a_2, b_1, b_2, c_1, c_2, r_1, r_2, r_3, r_4$ which can be updated through the backpropagation.

Unrolling networks formed by replacing fixed iterative parameters with scalar learnable weights preserve the interpretability of the iterative algorithm to the greatest extent but have limited expressive power and need to compute the inverse of matrices. A more expressive unrolling network can be obtained by replacing the fixed iteration parameters with simplicial convolutional filters to aggregate the information of adjacent edge flows. Therefore, this unrolling network is more flexible and expressive. The corresponding lth layer of the simplicial unrolling network for ElasticNet (SUEN)

Algorithm 1: Learning framework of the four-block ADMM for Simplicial ElasticNet

Input : $\mathbf{P}$, $\mathbf{y}$, $\mathbf{f}_0$, L
Output: $\mathbf{f}^{(L)}$

Function SUEN ($\mathbf{P}, \mathbf{y}, \mathbf{f}_0, L$);
Initialize $\mathbf{f}^{(0)}, \mathbf{z}_1^{(0)}, \mathbf{z}_2^{(0)}, \mathbf{z}_3^{(0)}, \mathbf{q}_1^{(0)}, \mathbf{q}_2^{(0)}, \mathbf{q}_3^{(0)}, \mathbf{q}_4^{(0)}$;
for $b = 1 : L$ **do**
    update $\hat{\mathbf{f}}^{(l+1)}$ by (4.9).;
    update $\mathbf{z}_i^{(l+1)}$ by (4.9) for $i = 1, 2, 3$;
    update $\mathbf{q}_i^{(l+1)}$ by (4.9) for $i = 1, 2, 3, 4$.;
end
Minimize $\|\mathbf{f}^{(L)} - \mathbf{f}_0\|_2^2$ and update all weights;
Return $\mathbf{f}^{(L)}$

consists of:

$$
\begin{cases}
\hat{\mathbf{f}}^{(l+1)} = \mathbf{H}_1\mathbf{y} + \mathbf{H}_2\mathbf{B}_1^\top\boldsymbol{\lambda}_1^{(l)} + \mathbf{H}_3\mathbf{B}_2\boldsymbol{\lambda}_2^{(l)} + \mathbf{H}_4\boldsymbol{\lambda}_3^{(l)} + \mathbf{H}_5\mathbf{P}^\top\boldsymbol{\lambda}_4^{(l)} \\
\qquad\qquad + \mathbf{H}_6\mathbf{B}_1^\top\mathbf{z}_1^{(l)} + \mathbf{H}_7\mathbf{B}_2\mathbf{z}_2^{(l)} + \mathbf{H}_8\mathbf{z}_3^{(l)} + \mathbf{H}_9\mathbf{P}^\top\mathbf{P}\mathbf{y} \\
\mathbf{z}_1^{(l+1)} = S_{\frac{a_1}{2a_2+r_1}}\left(\dfrac{1}{2a_2+r_1}(r_1\mathbf{B}_1\mathbf{f}^{(l+1)} - \boldsymbol{\lambda}_1^{(l)})\right) \\
\mathbf{z}_2^{(l+1)} = S_{\frac{b_1}{2b_2+r_2}}\left(\dfrac{1}{2b_2+r_2}(r_2\mathbf{B}_2^\top\mathbf{f}^{(l+1)} - \boldsymbol{\lambda}_2^{(l)})\right) \\
\mathbf{z}_3^{(l+1)} = S_{\frac{c_1}{2c_2+r_3}}\left(\dfrac{1}{2c_2+r_3}(r_3\mathbf{f}^{(l+1)} - \boldsymbol{\lambda}_3^{(l)})\right) \\
\boldsymbol{\lambda}_1^{(l+1)} = \boldsymbol{\lambda}_1^{(l)} - r_1(\mathbf{B}_1\mathbf{f}^{(l+1)} - \mathbf{z}_1^{(l+1)}) \\
\boldsymbol{\lambda}_2^{(l+1)} = \boldsymbol{\lambda}_2^{(l)} - r_2(\mathbf{B}_2^\top\mathbf{f}^{(l+1)} - \mathbf{z}_2^{(l+1)}) \\
\boldsymbol{\lambda}_3^{(l+1)} = \boldsymbol{\lambda}_3^{(l)} - r_3(\mathbf{f}^{(l+1)} - \mathbf{z}_3^{(l+1)}) \\
\boldsymbol{\lambda}_4^{(l+1)} = \boldsymbol{\lambda}_4^{(l)} - r_4(\mathbf{P}\mathbf{f}^{(l+1)} - \mathbf{P}\mathbf{y})
\end{cases}
\tag{4.10}
$$

where $\mathbf{H}_i$ are simplicial convolutional filters defined in (2.13), which contain some trainable parameters. In contrast to variant 1, this variant replaces the update step of $\hat{\mathbf{f}}$ with a polynomial containing simplicial filters and replaces ρ at different positions with four different trainable parameters r_1, r_2, r_3, and r_4. There are $1 + L_1 + L_2$ coefficients for each simplicial filter. Variant 2 has $13 + 3L_1 + 3L_2$ trainable parameters while variant 1 just has 6. Thus, variant 2 is more flexible and expressive than variant 1.

4.3.2. Convergence Analysis of Variant 1

Although the unrolling neural networks are designed based on optimization algorithms which often have a theoretical guarantee of convergence, the dynamic nature of their internal trainable parameters still complicates the convergence of their generated

sequences. We show that the sequences generated by variant 1 of the simplicial unrolling network for ElasticNet are globally convergent under the following conventional assumptions.

Assumption 1. For any given input $\mathbf{x}$, the soft-thresholding functions Φ_i with learnable parameters should satisfy $\|S_i(\mathbf{x}) - \mathbf{x}\|_2 \leq C_i/\rho$, where $C_i > 0$ are universal constants for $i = 1, 2, 3$.

This is an assumption on the bound. The common linear and nonlinear operations in deep neural networks are basically consistent with this assumption [29]. This also holds true for ADMM-based unrolling networks [59].

Assumption 2. The following four inequalities hold:

$$\|\mathbf{B}_1\hat{\mathbf{f}}^{(l+1)} - (\mathbf{z}_1^{(l)} + \mathbf{q}_1^{(l)})\|_2 \leq R_1/\rho \tag{4.11a}$$

$$\|\mathbf{B}_2^\top\hat{\mathbf{f}}^{(l+1)} - (\mathbf{z}_2^{(l)} + \mathbf{q}_2^{(l)})\|_2 \leq R_2/\rho \tag{4.11b}$$

$$\|\hat{\mathbf{f}}^{(l+1)} - (\mathbf{z}_3^{(l)} + \mathbf{q}_3^{(l)})\|_2 \leq R_3/\rho \tag{4.11c}$$

$$\|\mathbf{P}\hat{\mathbf{f}}^{(l+1)} - (\mathbf{P}\mathbf{y} + \mathbf{q}_4^{(l)})\|_2 \leq R_4/\rho \tag{4.11d}$$

where $0 < R_i < \infty$ for $i = 1, 2, 3, 4$.

This assumption assumes the difference between edge flow $\hat{\mathbf{f}}$ and generated sequences $\mathbf{z}_i$ and $\mathbf{q}_i$ is limited and bounded.

Assumption 3. The fixed parameter $\rho(l)$ should satisfy that $\rho(l) \to \infty$ when the depth of unrolling network $l \to \infty$.

When the penlty parameter ρ is set as $\rho = \rho_0 r_0^l$ where $r_0 > 1$, Assumption 3 can be satisfied.

Proposition 2. Variant 1 of simplicial unrolling network for ElasticNet (SUEN) in Algorithm 1 converges when Assumptions 1, 2, and 3 are satisfied. That is, $\{(\mathbf{f}^{(l)}, \mathbf{z}_1^{(l)}, \mathbf{z}_2^{(l)}, \mathbf{z}_3^{(l)}, \mathbf{q}_1^{(l)}, \mathbf{q}_2^{(l)}, \mathbf{q}_3^{(l)}, \mathbf{q}_4^{(l)})\}$ generated by SUEN is globally converges to a fixed-point.

Proof. See Appendix B.

This proposition indicates that if we check the NMSE or Pearson coefficient of the output at each layer of the trained unrolling network, the curves converge.

4.4. Summary

This chapter proposes a new optimization problem, ElasticNet, for the denoising and interpolation task of simplicial signals. The corresponding iterative solution is given based on the ADMM algorithm, and the corresponding simplicial unrolling network is constructed. The convergence of the proposed unrolling network SUEN is explored to prove the convergence of the network under some assumptions. This provides us with ideas for settings unrolling network parameters.

5

Experimental results

This section evaluates the proposed simplicial unrolling networks on the simplicial edge flows recovery tasks. In this chapter, section 5.1 introduces the datasets we used; section 5.2 explains the experimental setup; section 5.3 is the ablation study.

5.1. Datasets

5.1.1. Foreign Currency Exchange (Forex) dataset

We consider pairwise currency exchanges between 25 different currencies. This can be modeled as a network where the edge flow is the logarithm of the exchange rate. An essential rule in the exchange rate market is that the exchange rate value must guarantee the no-arbitrage condition. I.e., the rate difference and income cannot be obtained through repeated exchange between currency pairs. For currencies A and B, the exchange rate consists of represented by $r^{\mathrm{A/B}}$ and the no-arbitrage condition can be $r^{\mathrm{A/B}} r^{\mathrm{B/C}} = r^{\mathrm{A/C}}$. When this condition is met, the two paths of trading from currency A by using currency B as intermediate, trading to currency C, and directly trading from currency A to currency C are equivalent. If we use the logarithm to describe the edge flow, we obtain $\mathbf{f}_{[A,B]} = \log\left(r^{\mathrm{A/B}}\right)$ and the no-arbitrage condition means that the edge flow is curl-free as shown in Figure 5.1. Therefore, an arbitrage free exchange setting satisfies $\left\|\mathbf{B}_2^\top \mathbf{f}\right\|_1 = 0$ or $\left\|\mathbf{B}_2^\top \mathbf{f}\right\|_2^2 = 0$. We model the dataset as a simplicial complex with 25 nodes, 300 edges, and 2300 triangles. Our goal task here is to recover exchange rates under the arbitrary free condition, which is relevant in noisy fluctuation settings or when anomalies may be present.

Figure 5.1: Exchange rate modeled by simplicial complexes. The edge flow, composed directly of the exchange rate, is not curl-free (left). Moreover, the edge flow after taking the logarithm of the exchange rate can be considered approximately curl-free (right).

5.1.2. Lastfm dataset

The Lastfm dataset records the process of users switching artists while playing music. Each distinct artist can be modeled as a node, and an edge models the switch between two adjacent artists. When the user switches from artist A to artist B, we add a unit on the edge flow from A to B. Edge flows modeled in this way should be approximately divergence-free. Since only the nodes where the user starts and ends have nonzero divergence, whereas the rest of the nodes are divergence-free. Therefore, we can constrain the edge flow to satisfy $\left\|\mathbf{B}_2^\top \mathbf{f}\right\|_1 = 0$ or $\left\|\mathbf{B}_2^\top \mathbf{f}\right\|_2^2 = 0$. The Lastfam dataset can be modeled as a simplicial complex with 657 nodes, 1997 edges, and 1276 triangles. We use the topology of the Lastfm dataset is used and generate synthetic divergence-free edge flows. The task consists of recovering edge flows from noisy or partial measurements, which are typical when the information of users is inaccurate.

5.1.3. Chicago road network

This is the network of the city of Chicago and contains 546 nodes, 1088 edges, and 112 triangles [62]. Junctions on traffic roads are modeled as nodes, roads as edges, and the area enclosed by three roads as triangles. Based on this topology, we artificially generate divergence-free edge flows. Specifically, we perform random walks on the topology and record the number of walks on each edge to simulate the flow on the traffic road. The edge flow constructed in this way is roughly divergence-free. Gaussian noise or sampling is then added to corrupt the original edge-flow signal and to complete the corresponding signal reconstruction task. The task here consists of estimating the edge flows from noisy and partial measurements, which could be of relevance for traffic monitoring from a few sensors.

5.2. Experimental Setup

5.2.1. Models

We compare different iterative models, including:

- ADMM-SEN: ADMM for simplicial ElasticNet in (4.6).
- ADMM-STF: ADMM for simplicial trend filtering in (C.1).

Table 5.1: Parameters of ADMM-SEN

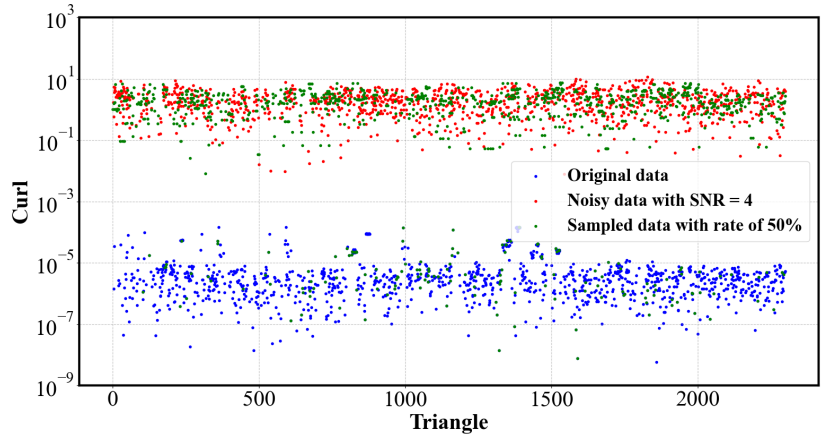
Dataset	α_1	α_2	β_1	β_2	γ_1	γ_2
Forex	0.01	0.01	1	1	0.01	0.01
Lastfm	1	1	0.01	0.01	0.01	0.01
Chicago	1	1	0.01	0.01	0.01	0.01

- SUEN: Simplicial unrolling network for ElasticNet in (4.10).
- SUTF: Simplicial unrolling network for trend filtering in (C.2).
- MLP [42]: Multilayer perceptrons are a fully connected neural network that has black-box nature.
- SNN [9]: Simplicial neural networks are developed for processing simplicial signals, and it is a non-model-based neural network.
- GUTF [8]: Graph unrolling network for trend filtering as a baseline. We build a line graph [48] where edges become nodes and vice versa and consider the edge flows as node signals. Then we deploy the approach of [8] to show the reconstruction performance.

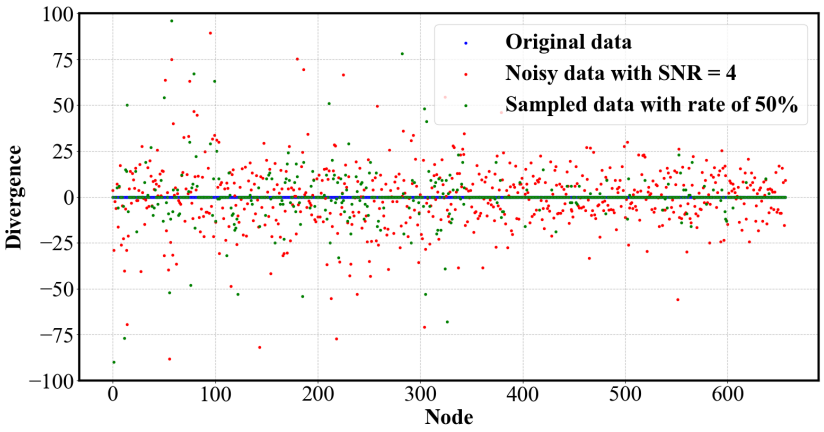
The coefficients of the regularizers in the ADMM-SEN are set in Table 5.1. For the divergence-free dataset, we set the regularizer coefficients associated with divergence much larger than that associated with curl. The opposite is true for the curl-free dataset. The coefficients of the regularizers in the ADMM-STF, the parameters are set as (i)$\alpha = 0.01, \beta = 1$ (ii)$\alpha = 1, \beta = 0.01$ (iii)$\alpha = 1, \beta = 0.01$ for Forex, Lastfm and Chicago dataset, respectively. Hyperparameters such as the number of layers are shown in Tables 5.2 and 5.3. We ran it till we got the best performance that we reported. The number of layers in MLP is 5, and the number of neurons in each layer is 16, 128, 128, 16, 1. The number of layers of SNN is 3, and the number of features output from each layer is 2, 2, and 1, respectively. Adam optimizer is used in all experiments comparing neural networks. All neural networks are trained using one-shot learning.

5.2.2. Noise and sampling models

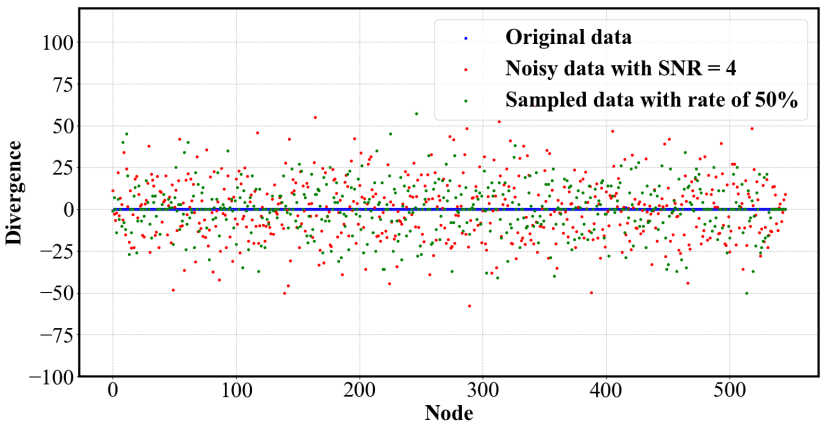
We add zero mean Gaussian noise with SNRs ranging from 0dB to 10 dB. For the interpolation task, we sample the edge flows randomly with a sampling rate from 20% to 90%. We evaluate the denoising performance via the normalized mean squared error (NMSE) and the interpolation performance via the Pearson correlation coefficient between the recovered and clean true signal as in [22]. The curl-free and divergence-free property of the original signal is destroyed after adding noise, as shown in Figure 5.2.



(a) Curl of Forex after adding noise or sampling



(b) Divergence of Lastfm after adding noise or sampling



(c) Divergence of Chicago after adding noise or sampling

Figure 5.2: Property of datasets. The top, middle, and bottom figures show the curl on Forex, divergence on the Lastfm, and divergence on Chicago datasets, respectively. For the denoising task, the curl of all triangles and the divergence of all nodes become larger after adding Gaussian noise. For the interpolation task, some of the triangles' curl and some of the nodes' divergence become larger after sampling, and some of the triangles' and nodes' properties remain the same.

5.2.3. Evaluation metrics

For the denoising task, the normalized mean square error

$$NMSE = \|\hat{\mathbf{f}} - \mathbf{f}_0\|_2^2 / \|\mathbf{f}_0\|_2^2 \tag{5.1}$$

is used to evaluate the performance. Here $\hat{\mathbf{f}}$ is the denoised edge flow signal, and $\mathbf{f}_0$ is the actual edge flow signal. The smaller the value of NMSE, the closer the denoised signal is to the actual signal; that is, the better the denoising effect. For the interpolation task, the Pearson correlation coefficient between $\hat{\mathbf{f}}$ and $\mathbf{f}_0$ is as follows

$$r_{\hat{\mathbf{f}}\mathbf{f}_0} = \frac{\sum_{i=1}^{n} \left(\hat{\mathbf{f}}_i - \bar{\hat{\mathbf{f}}}\right)\left(\mathbf{f}_{0i} - \bar{\mathbf{f}}_0\right)}{\sqrt{\sum_{i=1}^{n} \left(\hat{\mathbf{f}}_i - \bar{\hat{\mathbf{f}}}\right)^2} \sqrt{\sum_{i=1}^{n} \left(\mathbf{f}_{0i} - \bar{\mathbf{f}}_0\right)^2}} \tag{5.2}$$

are used to evaluate performance. In the interpolation task, the Pearson correlation coefficient between the recovered and original signals represents the interpolation performance. The closer it is to 1, the better the interpolation performance, i.e., the closer it is to the original edge-flow signal. The closer it is to 0, the more significant the difference between the recovered edge-flow signal and the original signal.

5.2.4. Denoising and Interpolation Performance

Table 5.2 shows the NMSE of the edge flow denoising task. On the Forex dataset, the unrolling network achieves comparable performance in NMSE to the iterative algorithm ADMM-STF. In contrast, the non-model-based neural networks MLPs and SNNs produce substandard results. On the Lastfm and Chicago datasets, the unrolling network SUEN with SUTF outperforms all other models in NMSE. This is because the unrolling network learns more accurate prior knowledge in these two datasets. The advantage of the unrolling network is more apparent when the SNR is close to zero. This is because the unrolling network can capture the patterns in the signal more accurately when the noise is significant.

Table 5.3 shows the Pearson correlation coefficients for the edge flow interpolation task. The performance of the SUEN and SUTF is close to that of ADMM on the Forex dataset. In contrast, on the Lastfm and Chicago datasets, SUEN and SUTF interpolate significantly better than all other models.

A significant difference in the performance of the iterative algorithm for the signal recovery task on these three datasets can be noticed, with the iterative algorithm performing better on the Forex dataset. The reason is that the curl-free property of the Forex dataset provides more practical information than the divergence-free property of the Lastfm and Chicago dataset. Curl-free is defined for triangles in simplicial complexes; thus, when there are more triangles in simplicial complexes, the curl-free property provides more information. Furthermore, the topology of the Forex dataset

Table 5.2: NMSE of denoising task

FOREX	Parameter	0dB	2dB	4dB	6dB	8dB	10dB
Original		0.99	0.63	0.39	0.24	0.15	0.09
MLP[42]	$L = 5$	0.70	0.48	0.34	0.21	0.15	0.10
SNN[9]	$L = 3, k = 1$	0.50	0.41	0.31	0.20	0.14	0.09
GUTF[8]	$L = 2$	0.47	0.36	0.28	0.19	0.14	0.09
ADMM-STF	$K = 500$	0.10	0.06	0.04	0.02	0.01	0.01
ADMM-SEN	$K = 500$	0.11	0.09	0.07	0.06	0.05	0.05
SUEN	$L = 4$	0.09	0.06	0.04	0.02	0.01	0.01
SUTF	$L = 6$	0.09	0.06	0.04	0.02	0.01	0.01
LASTFM	Parameter	0dB	2dB	4dB	6dB	8dB	10dB
Original		0.99	0.62	0.39	0.24	0.15	0.09
MLP[42]	$L = 5$	0.39	0.30	0.22	0.16	0.11	0.07
SNN[9]	$L = 3, k = 1$	0.28	0.21	0.15	0.11	0.09	0.07
GUTF[8]	$L = 2$	0.87	0.87	0.87	0.87	0.87	0.86
ADMM-STF	$K = 200$	0.94	0.59	0.36	0.22	0.14	0.08
ADMM-SEN	$K = 200$	0.70	0.44	0.28	0.18	0.12	0.07
SUEN	$L = 2$	0.09	0.08	0.07	0.06	0.04	0.04
SUTF	$L = 6$	0.09	0.08	0.08	0.05	0.04	0.03
CHICAGO	Parameter	0dB	2dB	4dB	6dB	8dB	10dB
Original		1.01	0.64	0.40	0.25	0.16	0.10
MLP[42]	$L = 5$	0.56	0.43	0.32	0.22	0.15	0.09
SNN[9]	$L = 3, k = 1$	0.47	0.35	0.27	0.19	0.14	0.10
GUTF[8]	$L = 2$	0.79	0.73	0.68	0.63	0.59	0.57
ADMM-STF	$K = 500$	0.96	0.59	0.37	0.22	0.14	0.08
ADMM-SEN	$K = 500$	0.53	0.34	0.21	0.13	0.08	0.05
SUEN	$L = 6$	0.37	0.26	0.18	0.12	0.08	0.05
SUTF	$L = 8$	0.37	0.26	0.18	0.12	0.08	0.05

is a completed simplicial complex, meaning a filled triangle exists among any three nodes. Therefore, the curl-free is a solid prior for the Forex dataset.

Overall, the traditional iterative algorithm ADMM and the neural networks can perform the signal recovery task. The difference between them is that the traditional iterative algorithm needs to have excellent prior knowledge to perform better. The regularizers and coefficients in the optimization problem, as well as the coefficients in the ADMM algorithm, need to be chosen based on the dataset's priori. For instance, in our experiments, we know that Forex, Lastfm, and Chicago dataset are curl- and divergence-free, respectively. Thus, six different regularizers are purposefully designed, and all coefficients are given, as shown in Table 5.1. However, this specific prior knowledge is often challenging to obtain in practice.

Therefore, considering neural networks to learn this prior knowledge is a potential strategy. However, obtaining a large amount of training data for neural networks is formidable because of the cost of labeled data. This leads to the fact that standard neural networks such as SNNs and MLPs often need more training data to achieve satisfactory results. However, the unrolling networks designed on the mathematical models can achieve considerable performance despite the one training sample. The

Table 5.3: Pearson correlation coefficient of interpolation task

FOREX	Parameter	20%	30%	40%	50%	60%	70%	80%	90%
Original		0.47	0.56	0.62	0.71	0.75	0.82	0.86	0.91
MLP[42]	$L = 5$	0.48	0.56	0.63	0.71	0.75	0.82	0.87	0.91
SNN[9]	$L = 3, k = 1$	0.48	0.56	0.62	0.71	0.75	0.82	0.86	0.91
GUTF[8]	$L = 2$	0.53	0.61	0.66	0.74	0.77	0.82	0.87	0.91
ADMM-STF	$K = 2500$	0.93	0.96	0.99	0.99	0.99	0.99	0.99	0.99
ADMM-SEN	$K = 1500$	0.96	0.98	0.99	0.99	0.99	0.99	0.99	0.99
SUEN	$L = 2$	0.96	0.97	0.99	0.99	0.99	0.99	0.98	0.99
SUTF	$L = 8$	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
LASTFM	Parameters	20%	30%	40%	50%	60%	70%	80%	90%
Original		0.39	0.53	0.62	0.68	0.74	0.82	0.86	0.93
MLP[42]	$L = 5$	0.39	0.54	0.63	0.69	0.75	0.82	0.86	0.93
SNN[9]	$L = 3, k = 1$	0.45	0.58	0.68	0.73	0.76	0.83	0.87	0.93
GUTF[8]	$L = 2$	0.16	0.20	0.23	0.23	0.24	0.25	0.25	0.26
ADMM-STF	$K = 2000$	0.39	0.54	0.63	0.69	0.75	0.82	0.87	0.93
ADMM-SEN	$K = 500$	0.41	0.56	0.65	0.72	0.78	0.85	0.90	0.96
SUEN	$L = 2$	0.93	0.94	0.95	0.96	0.97	0.97	0.98	0.99
SUTF	$L = 6$	0.94	0.95	0.96	0.96	0.97	0.97	0.98	0.99
CHICAGO	Parameters	20%	30%	40%	50%	60%	70%	80%	90%
Original		0.45	0.55	0.64	0.73	0.79	0.86	0.90	0.96
MLP[42]	$L = 5$	0.45	0.55	0.64	0.73	0.78	0.86	0.90	0.96
SNN[9]	$L = 3, k = 1$	0.53	0.63	0.73	0.82	0.84	0.92	0.92	0.96
GUTF[8]	$L = 2$	0.28	0.34	0.42	0.49	0.52	0.58	0.63	0.66
ADMM-STF	$K = 250$	0.47	0.56	0.65	0.74	0.80	0.87	0.91	0.96
ADMM-SEN	$K = 1000$	0.60	0.71	0.80	0.88	0.93	0.96	0.98	0.99
SUEN	$L = 12$	0.63	0.73	0.82	0.91	0.95	0.98	0.99	0.99
SUTF	$L = 8$	0.61	0.72	0.81	0.89	0.93	0.96	0.98	0.99

reason is that unrolling networks make full use of limited prior knowledge, and their underlying structure is based on iterative steps. Thus the number of trainable weights is notably less than that of standard non-model-based neural networks. This indicates that the unrolling networks will search a much smaller function space than other neural networks for an optimal solution. Non-model-based neural networks, such as SNNs and MLPs, are a more general type of neural network, and their structure can be similar for distinct tasks. This is advantageous when the available training data is sufficient because it implies a more powerful expressive ability. However, when the training data is limited, this generality can increase the training burden. In contrast, the structure of unrolling networks needs to be designed for specific tasks. This results in a lower generality, which has a higher interpretability and training efficiency.

GUTF experiments in table 5.2 and 5.3 transform the edge flow signals in the simplicial complexes into node signals in the corresponding line graph and uses the graph unrolling network of trend filtering (GUTF) to complete the edge flow reconstruction task. The experimental results show that the graph-based unrolling network cannot achieve satisfactory results. This is because the structure of GUTF is designed based on the graph signal reconstruction task, and the underlying optimization problem is the graph trend filtering problem without utilizing the topology of simplicial complexes.

Graph trend filtering forces the differences of the recovered graph signals between connected nodes to be sparse, which implies that the reconstructed neighboring edge flows are close, which is not realistic. Real-world edge flows tend to be curl-free or divergence-free. Therefore, although GUTF has a similar number of trainable parameters as SUEN and SUTF, it still cannot achieve good results in one-shot learning. This indicates that the unrolling network is a model-based neural network, and the degree of matching between the underlying model and the solved task will directly affect the training effect of the unrolling network.

5.2.5. Convergence Results

We test the convergence of the multi-block ADMM algorithm with the penalty parameter ρ varying from 0.1 to 0.4. The experimental results are shown in Figure 5.3, where the multi-block ADMM algorithm is always guaranteed to converge as ρ varies. The larger the ρ value, the faster the convergence of the algorithm. As for variant 1 of SUEN, we check the output of each layer of the unrolling network, and the number of layers is gradually increased from 1 layer to up to 14 layers. To satisfy Assumption 3, ρ in SUEN is set to $\rho = 1.01^l \rho_0$, which is a non-learnable parameter that varies with the number of layers. The results of the convergence experiments of SUEN are shown in Figure 5.3. The NMSE and the Pearson correlation coefficients of the recovered edge flow generated by the trained SUEN network converge gradually as the number of layers is gradually increased.

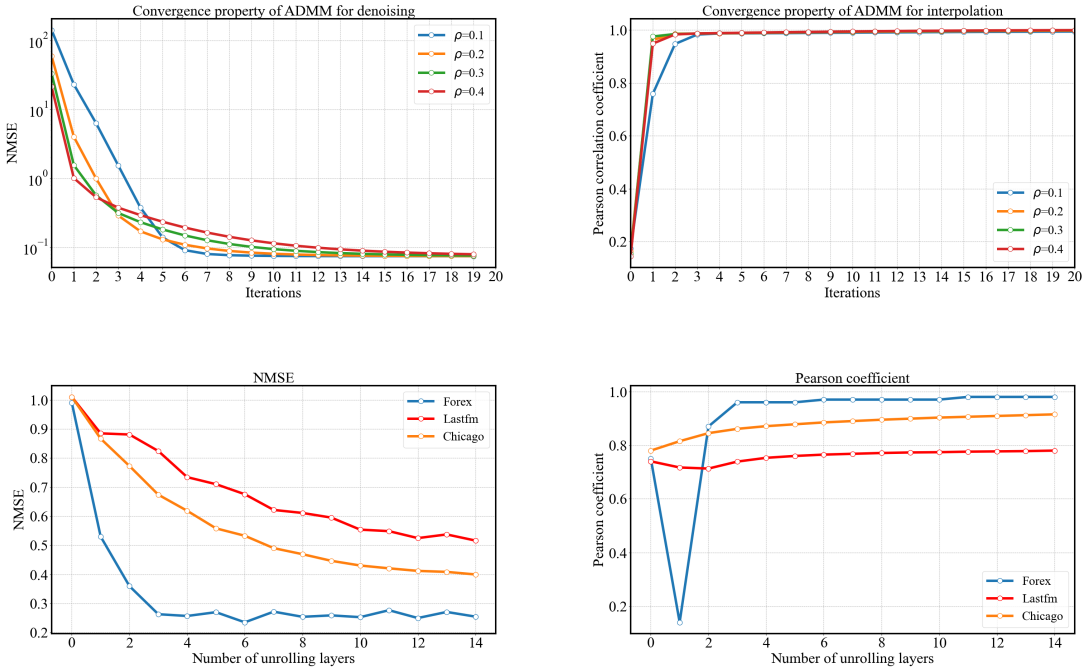


Figure 5.3: Convergence property of multi-block ADMM and unrolling network. The figures on the top are the convergence performance of the 4-block ADMM. As ρ changes from small to large, the ADMM algorithm is converging. The figures on the bottom are the convergence performance of the simplicial unrolling network. We can observe that the network converges as the depth becomes deeper.

5.2.6. Effect of simplicial convolutional filters

We verify the contribution of adding trainable simplicial convolutional filters by ablation study. Figure 5.4 show the effect of trainable filters on the NMSE and Pearson correlation coefficients in the denoising and interpolation tasks, respectively. The introduction of the simplicial convolutional filters in the unrolling network structure improves the learning ability of the network, resulting in better performance in the edge flow reconstruction task. This is because the network with the simplicial convolutional filters aggregates the information of the edge flow and its neighbors at each layer, thus giving the network a stronger learning and representation capability. Furthermore, the interpretability of the network is highest when the fixed parameters in the iterative algorithm are directly replaced with trainable scalar parameters. In this case, each trainable parameter in the network directly corresponds to a component in the iterative algorithm, which allows us to interpret the network parameters after training based on the iterative algorithm. However, this replacement strategy does not make full use of the information from the edge flow neighbors and overall reduces the learning ability of the unrolling network.

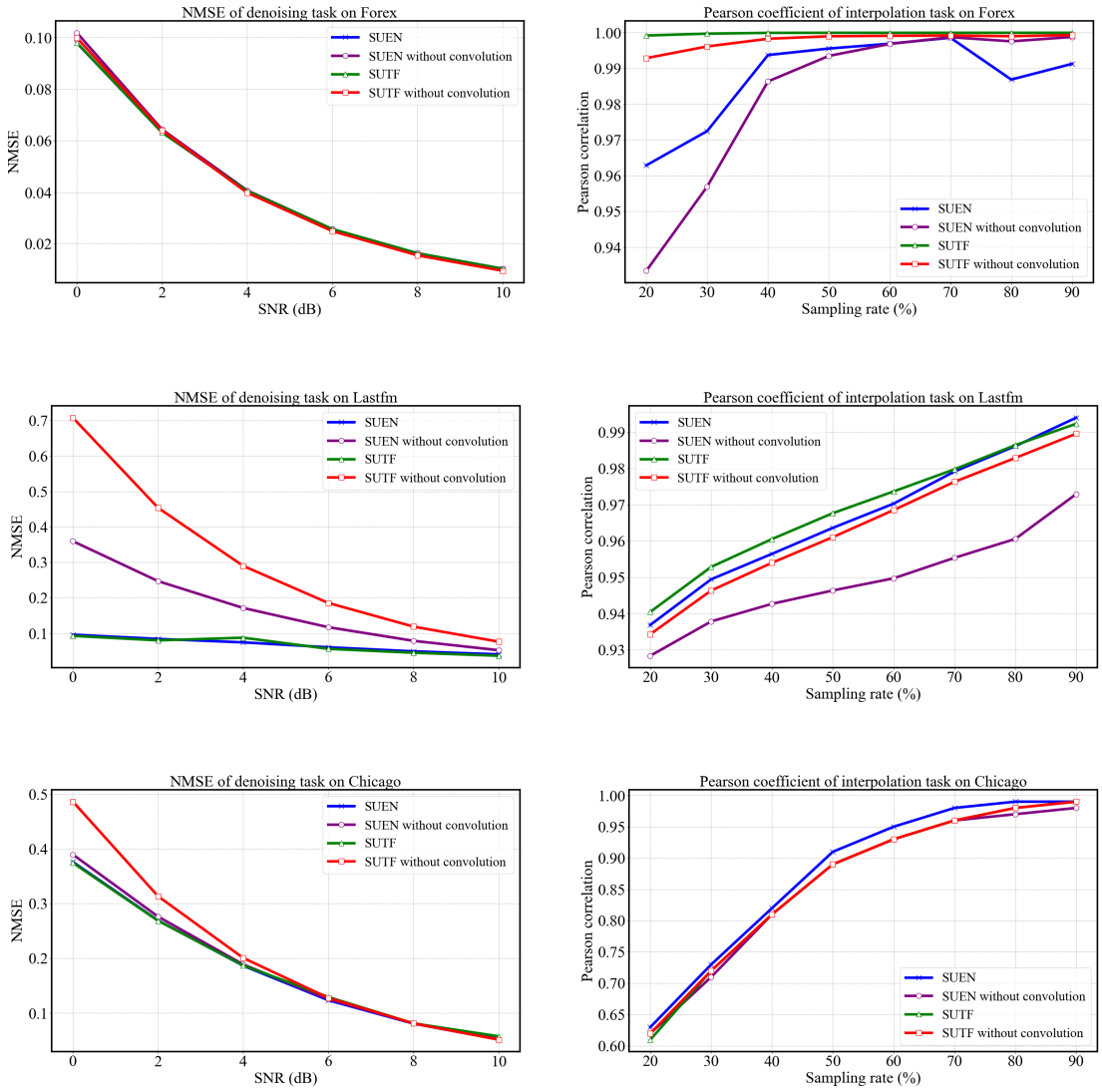


Figure 5.4: Reconstruction performance of different parametric strategies. SNR for denoising tasks ranges from 0 dB to 10 dB. The sampling rate for the interpolation task is ranging from 20% to 90%. 'SUEN' and 'SUTF' are unrolling networks containing simplicial convolutional filters, while 'SUEN without convolution' and 'SUTF without convolution' employs scalar weights. The left figures correspond to the result of the denoising task and the right to the interpolation task. The top, middle, and bottom figures show the performance on the Forex, Lastfm, and Chicago datasets, respectively.

5.3. Summary

This chapter conducts experiments on real-world as well as artificial datasets. The experiments verify that the simplicial unrolling network has advantages over traditional iterative algorithms as well as non-model-based neural networks in simplicial signal recovery tasks when the training sample is limited. We verify the importance of simplicial filters in unrolling networks. It aggregates the information of the edge flow and its neighbors at each layer, thus giving the network a stronger learning and representation capability.

6

Conclusion

6.1. Summary

We propose a highly interpretable and easily trainable simplicial unrolling network for the reconstruction task of simplicial edge flows. The core of this unrolling network is trainable simplicial convolutional filters that collect information from the neighbors of the edge flows and adjusts the filter parameters according to the input data, thus improving the learning and expression capabilities of the network. Then, we design the corresponding unrolling networks SUEN and SUTF for the simplicial ElasticNet and the simplicial trend filtering, respectively. Under some mild conditions, we prove the convergence of this simplicial unrolling network. Through experiments on real-world and semi-artificial datasets, we verify that the simplicial unrolling network can achieve better reconstruction results than non-model-based neural networks and traditional iterative algorithms with limited training data. Here, we recall the questions proposed in the introduction:

- How to choose the proper regularizer for the edge flow reconstruction task?
- How to reduce our reliance on a priori knowledge?

The answer is:

- When dealing with curl- or divergence-free edge flow, the simplicial ElasticNet optimization problem is proposed. Its regularizers contain two different types of regularizers, which are ℓ_1 and ℓ_2 norm regularizers. The ℓ_1 norm regularizers promote the sparsity of the divergence or curl of the recovered edge flow, and the ℓ_2 norm regularizers keep the value of divergence or curl to be as low as possible.
- Based on the traditional iterative algorithm, the corresponding unrolling network is constructed. We can learn the potential prior knowledge in the data in a data-driven manner. This avoids the errors caused by empirically determined prior.

6.2. Future work

In future research, this project can be improved from the following perspectives. First, this project only considers the simplicial signal at a certain moment, but not the time-varying signal. In future research, the dimension of time can be added. This means that we need to solve new regularized optimization problems. Secondly, this project only considers ADMM as an iterative optimization algorithm. In future research, other algorithms can be used to construct the corresponding unrolling networks, which may give better performance. Finally, this project only considers the interaction of nodes, edges, and signals on triangles in simplexes but does not design for higher order signals. In future research, the inclusion of tetrahedral signals or even higher dimensional signals can be considered.

A

Proof of the Proposition 1

Before the proof, we recall this useful lemma from [16].
Lemma 1. If all the blocks in the multi-block ADMM algorithm are strongly convex functions, the convergence of multi-block ADMM is guaranteed if the penalty parameter ρ in augmented Lagrangian function satisfies the condition

$$0 < \rho < \min_{1 \le i \le m} \left\{ \frac{2\mu_i}{3(m-1)\|\mathbf{A}_i\|_2^2} \right\} \tag{A.1}$$

where μ_i is the strongly convex constant of each block, m is the number blocks and $\mathbf{A}_i$ is the coefficient matrix for equality constraints.

The optimization problem (4.4) can be written as follow

$$\begin{aligned} \underset{\hat{\mathbf{f}} \in \mathbb{R}^{N_1}}{\operatorname{argmin}} & \|\hat{\mathbf{f}} - \mathbf{y}\|_2^2 + \alpha_1 \|\mathbf{z}_1\|_1 + \alpha_2 \|\mathbf{z}_1\|_2^2 + \beta_1 \|\mathbf{z}_2\|_1 \\ & + \beta_2 \|\mathbf{z}_2\|_2^2 + \gamma_1 \|\mathbf{z}_3\|_1 + \gamma_2 \|\mathbf{z}_3\|_2^2 \end{aligned} \tag{A.2}$$

$$\text{subject to} \quad \mathbf{A}_1 \hat{\mathbf{f}} + \mathbf{A}_2 \mathbf{z}_1 + \mathbf{A}_3 \mathbf{z}_2 + \mathbf{A}_4 \mathbf{z}_3 = \mathbf{c}$$

where $\mathbf{A}_i$ are defined in (4.7a) - (4.7d) and matrix $\mathbf{c}$ is

$$\mathbf{c} = [\mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{P}\mathbf{y}] \in \mathbb{R}^{(N_0+N_1+N_2+M)\times 1} \tag{A.3}$$

There are four different blocks for multi-block ADMM: $\|\hat{\mathbf{f}} - \mathbf{y}\|_2^2$; $\alpha_1\|\mathbf{z}_1\|_1 + \alpha_2\|\mathbf{z}_1\|_2^2$; $\beta_1\|\mathbf{z}_2\|_1 + \beta_2\|\mathbf{z}_2\|_2^2$ and $\gamma_1\|\mathbf{z}_3\|_1 + \gamma_2\|\mathbf{z}_3\|_2^2$. These four components are all strongly convex functions, and their constants are $\mu_1 = 2, \mu_2 = 2\alpha_2, \mu_3 = 2\beta_2, \mu_4 = 2\gamma_2$. This brings us to the setting of Lemma 1, which under the conditions in (A.1), completes the proof.

B

Proof of the Proposition 2

Define $\widetilde{\mathbf{z}}_1^{(l)} = \mathbf{B}_1\hat{\mathbf{f}}^{(l+1)} - \boldsymbol{\lambda}_1^{(l)}/\rho$, $\widetilde{\mathbf{z}}_2^{(l)} = \mathbf{B}_2^\top\hat{\mathbf{f}}^{(l+1)} - \boldsymbol{\lambda}_2^{(l)}/\rho$, $\widetilde{\mathbf{z}}_3^{(l)} = \hat{\mathbf{f}}^{(l+1)} - \boldsymbol{\lambda}_3^{(l)}/\rho$ and according to (4.9) then we have

$$\|\mathbf{z}_1^{(l+1)} - \widetilde{\mathbf{z}}_1^{(l)}\|_2 = \|S_1(\widetilde{\mathbf{z}}_1^{(l)}) - \widetilde{\mathbf{z}}_1^{(l)}\|_2 \leq C_1/\rho(l) \tag{B.1}$$

According to (B.1) and (4.11a), we can show

$$\begin{aligned}\|\mathbf{z}_1^{(l+1)} - \mathbf{z}_1^{(l)}\|_2 &\leq \|\mathbf{z}_1^{(l+1)} - \widetilde{\mathbf{z}}_1^{(l)}\|_2 + \|\widetilde{\mathbf{z}}_1^{(l)} - \mathbf{z}_1^{(l)}\|_2 \\ &\leq C_1/\rho(l) + \|\mathbf{B}_1\hat{\mathbf{f}}^{(l+1)} - (\mathbf{z}_1^{(l)} + \mathbf{q}_1^{(l)})\|_2 \\ &\leq C_1/\rho(l) + R_1/\rho(l)\end{aligned} \tag{B.2}$$

As for $\mathbf{z}_2$, we have

$$\|\mathbf{z}_2^{(l+1)} - \widetilde{\mathbf{z}}_2^{(l)}\|_2 = \|S_2(\widetilde{\mathbf{z}}_2^{(l)}) - \widetilde{\mathbf{z}}_2^{(l)}\|_2 \leq C_2/\rho(l) \tag{B.3}$$

According to (B.3) and (4.11b), we can show

$$\begin{aligned}\|\mathbf{z}_2^{(l+1)} - \mathbf{z}_2^{(l)}\|_2 &\leq \|\mathbf{z}_2^{(l+1)} - \widetilde{\mathbf{z}}_2^{(l)}\|_2 + \|\widetilde{\mathbf{z}}_2^{(l)} - \mathbf{z}_2^{(l)}\|_2 \\ &\leq C_2/\rho(l) + \|\mathbf{B}_2^\top\hat{\mathbf{f}}^{(l+1)} - (\mathbf{z}_2^{(l)} + \mathbf{q}_2^{(l)})\|_2 \\ &\leq C_2/\rho(l) + R_2/\rho(l)\end{aligned} \tag{B.4}$$

As for $\mathbf{z}_3$, we have

$$\|\mathbf{z}_3^{(l+1)} - \widetilde{\mathbf{z}}_3^{(l)}\|_2 = \|S_3(\widetilde{\mathbf{z}}_3^{(l)}) - \widetilde{\mathbf{z}}_3^{(l)}\|_2 \leq C_3/\rho(l) \tag{B.5}$$

According to (B.5) and (4.11c), we can show

$$\begin{aligned}\|\mathbf{z}_3^{(l+1)} - \mathbf{z}_3^{(l)}\|_2 &\leq \|\mathbf{z}_3^{(l+1)} - \widetilde{\mathbf{z}}_3^{(l)}\|_2 + \|\widetilde{\mathbf{z}}_3^{(l)} - \mathbf{z}_3^{(l)}\|_2 \\ &\leq C_3/\rho(l) + \|\hat{\mathbf{f}}^{(l+1)} - (\mathbf{z}_3^{(l)} + \mathbf{q}_3^{(l)})\|_2 \\ &\leq C_3/\rho(l) + R_3/\rho(l)\end{aligned} \tag{B.6}$$

As for $\mathbf{q}_1^{(l)}$, according to (4.9) and Assumption 1, we have

$$\begin{aligned} \|\mathbf{q}_1^{(l+1)}\|_2 &= \|\mathbf{q}_1^{(l)} - (\mathbf{B}_1\mathbf{f}^{(l+1)} - \mathbf{z}_1^{(l+1)})\|_2 \\ &= \|\mathbf{z}_1^{(l+1)} - (\mathbf{B}_1\mathbf{f}^{(l+1)} - \mathbf{q}_1^{(l)})\|_2 \\ &= \|S_1(\widetilde{\mathbf{z}}_1^{(l)}) - \widetilde{\mathbf{z}}_1^{(l)}\|_2 \leq C_1/\rho(l) \end{aligned} \tag{B.7}$$

then, we can show

$$\begin{aligned} \|\mathbf{q}_1^{(l+1)} - \mathbf{q}_1^{(l)}\|_2 &\leq \|\mathbf{q}_1^{(l+1)}\|_2 + \|\mathbf{q}_1^{(l)}\|_2 \\ &\leq C_1/\rho(l) + C_1/\rho(l-1) \end{aligned} \tag{B.8}$$

As for $\mathbf{q}_2^{(l)}$, according to (4.9) and Assumption 1, we have

$$\begin{aligned} \|\mathbf{q}_2^{(l+1)}\|_2 &= \|\mathbf{q}_2^{(l)} - (\mathbf{B}_2^\top\mathbf{f}^{(l+1)} - \mathbf{z}_2^{(l+1)})\|_2 \\ &= \|\mathbf{z}_2^{(l+1)} - (\mathbf{B}_2^\top\mathbf{f}^{(l+1)} - \mathbf{q}_2^{(l)})\|_2 \\ &= \|S_2(\widetilde{\mathbf{z}}_2^{(l)}) - \widetilde{\mathbf{z}}_2^{(l)}\|_2 \leq C_2/\rho(l) \end{aligned} \tag{B.9}$$

then, we can show

$$\begin{aligned} \|\mathbf{q}_2^{(l+1)} - \mathbf{q}_2^{(l)}\|_2 &\leq \|\mathbf{q}_2^{(l+1)}\|_2 + \|\mathbf{q}_2^{(l)}\|_2 \\ &\leq C_2/\rho(l) + C_2/\rho(l-1) \end{aligned} \tag{B.10}$$

As for $\mathbf{q}_3^{(l)}$, according to (4.9) and Assumption 1, we have

$$\begin{aligned} \|\mathbf{q}_3^{(l+1)}\|_2 &= \|\mathbf{q}_3^{(l)} - (\mathbf{f}^{(l+1)} - \mathbf{z}_3^{(l+1)})\|_2 \\ &= \|\mathbf{z}_3^{(l+1)} - (\mathbf{f}^{(l+1)} - \mathbf{q}_3^{(l)})\|_2 \\ &= \|S_3(\widetilde{\mathbf{z}}_3^{(l)}) - \widetilde{\mathbf{z}}_3^{(l)}\|_2 \leq C_3/\rho(l) \end{aligned} \tag{B.11}$$

then, we can show

$$\begin{aligned} \|\mathbf{q}_3^{(l+1)} - \mathbf{q}_3^{(l)}\|_2 &\leq \|\mathbf{q}_3^{(l+1)}\|_2 + \|\mathbf{q}_3^{(l)}\|_2 \\ &\leq C_3/\rho(l) + C_3/\rho(l-1) \end{aligned} \tag{B.12}$$

As for $\mathbf{q}_4^{(l)}$, according to (4.9) and Assumption 2, we have

$$\begin{aligned} \|\mathbf{q}_4^{(l+1)}\|_2 &= \|\mathbf{q}_4^{(l)} - (\mathbf{P}\mathbf{f}^{(l+1)} - \mathbf{P}\mathbf{y})\|_2 \\ &= \|\mathbf{P}\mathbf{f}^{(l+1)} - (\mathbf{P}\mathbf{y} + \mathbf{q}_4^{(l)})\|_2 \\ &\leq R_4/\rho(l) \end{aligned} \tag{B.13}$$

then, we can show

$$\begin{aligned} \|\mathbf{q}_4^{(l+1)} - \mathbf{q}_4^{(l)}\|_2 &\leq \|\mathbf{q}_4^{(l+1)}\|_2 + \|\mathbf{q}_4^{(l)}\|_2 \\ &\leq R_4/\rho(l) + R_4/\rho(l-1) \end{aligned} \tag{B.14}$$

As for $\hat{\mathbf{f}}$, according to $\mathbf{q}_3$-update in (4.9), we have

$$\begin{aligned} & \|\mathbf{f}^{(l+1)} - \mathbf{f}^{(l)}\|_2 \\ = & \|(\mathbf{q}_3^{(l)} - \mathbf{q}_3^{(l+1)} + \mathbf{z}_3^{(l+1)}) - (\mathbf{q}_3^{(l-1)} - \mathbf{q}_3^{(l)} + \mathbf{z}_3^{(l)})\|_2 \\ \leq & \|\mathbf{q}_3^{(l+1)} - \mathbf{q}_3^{(l)}\|_2 + \mathbf{q}_3^{(l)} - \mathbf{q}_3^{(l-1)}\|_2 + \|\mathbf{z}_3^{(l+1)} - \mathbf{z}_3^{(l)}\|_2 \\ \leq & (C_3 + C_1 + R_1)/\rho(l) + 2C_3/\rho(l-1) + C_3/\rho(l-2) \end{aligned} \tag{B.15}$$

Therefore, according to (B.2), (B.4), (B.6), (B.8), (B.10), (B.12), (B.14), (B.15) and Assumption 3, $\{(\mathbf{f}^{(l)}, \mathbf{z}_1^{(l)}, \mathbf{z}_2^{(l)}, \mathbf{z}_3^{(l)}, \mathbf{q}_1^{(l)}, \mathbf{q}_2^{(l)}, \mathbf{q}_3^{(l)}, \mathbf{q}_4^{(l)})\}$ generated by SUEN is a Cauchy sequence and there exists a fixed-point $\{(\mathbf{f}^*, \mathbf{z}_1^*, \mathbf{z}_2^*, \mathbf{z}_3^*, \mathbf{q}_1^*, \mathbf{q}_2^*, \mathbf{q}_3^*, \mathbf{q}_4^*)\}$ such that $\{(\mathbf{f}^{(l)}, \mathbf{z}_1^{(l)}, \mathbf{z}_2^{(l)}, \mathbf{z}_3^{(l)}, \mathbf{q}_1^{(l)}, \mathbf{q}_2^{(l)}, \mathbf{q}_3^{(l)}, \mathbf{q}_4^{(l)})\} \rightarrow \{(\mathbf{f}^*, \mathbf{z}_1^*, \mathbf{z}_2^*, \mathbf{z}_3^*, \mathbf{q}_1^*, \mathbf{q}_2^*, \mathbf{q}_3^*, \mathbf{q}_4^*)\}$ if the depth of unrolling network $l \rightarrow \infty$, which completes the proof.

C

ADMM and unrolling network for trend filtering

The iteration steps of three-block ADMM for the reconstruction task can be described as follows

$$\begin{cases} \hat{\mathbf{f}}^{(k+1)} :=& (2\mathbf{I} + \rho\mathbf{B}_1^\top\mathbf{B}_1 + \rho\mathbf{B}_2\mathbf{B}_2^\top + \rho\mathbf{P}^\top\mathbf{P})^{-1} \\ & (2\mathbf{y} + \mathbf{B}_1^\top\boldsymbol{\lambda}_1^{(k)} + \mathbf{B}_2\boldsymbol{\lambda}_2^{(k)} + \mathbf{P}^\top\boldsymbol{\lambda}_3^{(k)} + \rho\mathbf{B}_1^\top\mathbf{z}_1^{(k)} \\ & + \rho\mathbf{B}_2\mathbf{z}_2^{(k)} + \rho\mathbf{P}^\top\mathbf{P}\mathbf{y}) \\ \mathbf{z}_1^{(k+1)} :=& S_{\frac{\alpha_1}{\rho}}(\frac{1}{\rho}(\rho\mathbf{B}_1\mathbf{f}^{(k+1)} - \boldsymbol{\lambda}_1^{(k)})) \\ \mathbf{z}_2^{(k+1)} :=& S_{\frac{\beta_1}{\rho}}(\frac{1}{\rho}(\rho\mathbf{B}_2^\top\mathbf{f}^{(k+1)} - \boldsymbol{\lambda}_2^{(k)})) \\ \boldsymbol{\lambda}_1^{(k+1)} :=& \boldsymbol{\lambda}_1^{(k)} - \rho(\mathbf{B}_1\mathbf{f}^{(k+1)} - \mathbf{z}_1^{(k+1)}) \\ \boldsymbol{\lambda}_2^{(k+1)} :=& \boldsymbol{\lambda}_2^{(k)} - \rho(\mathbf{B}_2^\top\mathbf{f}^{(k+1)} - \mathbf{z}_2^{(k+1)}) \\ \boldsymbol{\lambda}_3^{(k+1)} :=& \boldsymbol{\lambda}_3^{(k)} - \rho(\mathbf{P}\mathbf{f}^{(k+1)} - \mathbf{P}\mathbf{y}) \end{cases} \tag{C.1}$$

To construct a simplicial unrolling network for trend filtering (SUTF), certain iterative parameters should also be substituted with trainable parameters. The first method is replacing the fixed parameters α_1, β_1 and ρ in trend filtering and ADMM directly by learnable parameters a_1, b_1, r_1 and r_2. By substituting the fixed parameters

with simplicial convolutional filters, the lth layer of SUTF can be represented as follows

$$\begin{cases} \hat{\mathbf{f}}^{(l+1)} :=& \mathbf{H}_1\mathbf{y} + \mathbf{H}_2\mathbf{B}_1^\top\boldsymbol{\lambda}_1^{(l)} + \mathbf{H}_3\mathbf{B}_2\boldsymbol{\lambda}_2^{(l)} + \mathbf{H}_4\mathbf{P}^\top\boldsymbol{\lambda}_3^{(l)} \\ & + \mathbf{H}_5\mathbf{B}_1^\top\mathbf{z}_1^{(l)} + \mathbf{H}_6\mathbf{B}_2\mathbf{z}_2^{(l)} + \mathbf{H}_7\mathbf{P}^\top\mathbf{P}\mathbf{y} \\ \mathbf{z}_1^{(l+1)} :=& S_{\frac{a_1}{r_1}}(\frac{1}{r_1}(r_1\mathbf{B}_1\mathbf{f}^{(l)} - \boldsymbol{\lambda}_1^{(l)})) \\ \mathbf{z}_2^{(l+1)} :=& S_{\frac{b_1}{r_2}}(\frac{1}{r_2}(r_2\mathbf{B}_2^\top\mathbf{f}^{(l)} - \boldsymbol{\lambda}_2^{(l)})) \\ \boldsymbol{\lambda}_1^{(l+1)} :=& \boldsymbol{\lambda}_1^{(l)} - r_1(\mathbf{B}_1\mathbf{f}^{(l+1)} - \mathbf{z}_1^{(l+1)}) \\ \boldsymbol{\lambda}_2^{(l+1)} :=& \boldsymbol{\lambda}_2^{(l)} - r_2(\mathbf{B}_2^\top\mathbf{f}^{(l+1)} - \mathbf{z}_2^{(l+1)}) \\ \boldsymbol{\lambda}_3^{(l+1)} :=& \boldsymbol{\lambda}_3^{(l)} - r_3(\mathbf{P}\mathbf{f}^{(l+1)} - \mathbf{P}\mathbf{y}) \end{cases} \tag{C.2}$$

Where $\mathbf{H}_i$ are simplicial convolutional filters that contain some trainable parameters. Unlike SUEN, SUTF has fewer learnable parameters and only constrains the curl and divergence, but not the regularizers that constrain the signal's sparsity.

References

[1] Sergio Barbarossa and Stefania Sardellitti. "Topological signal processing: Making sense of data building on multiway relations". In: *IEEE Signal Processing Magazine* 37.6 (2020), pp. 174–183.

[2] Sergio Barbarossa and Stefania Sardellitti. "Topological signal processing over simplicial complexes". In: *IEEE Transactions on Signal Processing* 68 (2020), pp. 2992–3007.

[3] Ginestra Bianconi. *Multilayer networks: structure and function*. Oxford university press, 2018.

[4] Pasquale Bove et al. "Prediction of Dynamical Properties of Biochemical Pathways with Graph Neural Networks." In: *Bioinformatics*. 2020, pp. 32–43.

[5] Stephen Boyd et al. "Distributed optimization and statistical learning via the alternating direction method of multipliers". In: *Foundations and Trends® in Machine learning* 3.1 (2011), pp. 1–122.

[6] Gunnar Carlsson. "Topology and data". In: *Bulletin of the American Mathematical Society* 46.2 (2009), pp. 255–308.

[7] Siheng Chen and Yonina C Eldar. "Time-Varying Graph Signal Inpainting Via Unrolling Networks". In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2021, pp. 8092–8097.

[8] Siheng Chen, Yonina C Eldar, and Lingxiao Zhao. "Graph unrolling networks: Interpretable neural networks for graph signal denoising". In: *IEEE Transactions on Signal Processing* 69 (2021), pp. 3699–3713.

[9] Stefania Ebli, Michaël Defferrard, and Gard Spreemann. "Simplicial neural networks". In: *arXiv preprint arXiv:2010.03633* (2020).

[10] Stefania Ebli and Gard Spreemann. "A notion of harmonic clustering in simplicial complexes". In: *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*. IEEE. 2019, pp. 1083–1090.

[11] Tim S Evans and Renaud Lambiotte. "Line graphs, link partitions, and overlapping communities". In: *Physical Review E* 80.1 (2009), p. 016105.

[12] Daniel Gabay and Bertrand Mercier. "A dual algorithm for the solution of nonlinear variational problems via finite element approximation". In: *Computers & mathematics with applications* 2.1 (1976), pp. 17–40.

[13] Chad Giusti, Robert Ghrist, and Danielle S Bassett. "Two's company, three (or more) is a simplex". In: *Journal of computational neuroscience* 41.1 (2016), pp. 1–14.

[14] R Glowinski and A Marrocco. *Approximation par éléments finis d'ordre un et résolution par pénalisation-dualité d'une classe de problèmes non linéaires. RAIRO. Rech. Opér*. 1975.

[15] Karol Gregor and Yann LeCun. "Learning fast approximations of sparse coding". In: *Proceedings of the 27th international conference on international conference on machine learning*. 2010, pp. 399–406.

[16] Deren Han and Xiaoming Yuan. "A note on the alternating direction method of multipliers". In: *Journal of Optimization Theory and Applications* 155.1 (2012), pp. 227–238.

[17] John R Hershey, Jonathan Le Roux, and Felix Weninger. "Deep unfolding: Model-based inspiration of novel deep architectures". In: *arXiv preprint arXiv:1409.2574* (2014).

[18] Weiyu Huang et al. "A graph signal processing perspective on functional brain imaging". In: *Proceedings of the IEEE* 106.5 (2018), pp. 868–885.

[19] Nabil Ibtehaz and M Sohel Rahman. "MultiResUNet: Rethinking the U-Net architecture for multimodal biomedical image segmentation". In: *Neural networks* 121 (2020), pp. 74–87.

[20] Elvin Isufi, Fernando Gama, and Alejandro Ribeiro. "EdgeNets: Edge varying graph neural networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.11 (2021), pp. 7457–7473.

[21] Elvin Isufi, Geert Leus, and Paolo Banelli. "2-dimensional finite impulse response graph-temporal filters". In: *2016 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. IEEE. 2016, pp. 405–409.

[22] Junteng Jia et al. "Graph-based semi-supervised & active learning for edge flows". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2019, pp. 761–771.

[23] Xiaoye Jiang et al. "Statistical ranking and combinatorial Hodge theory". In: *Mathematical Programming* 127.1 (2011), pp. 203–244.

[24] Lida Kanari et al. "A topological representation of branching neuronal morphologies". In: *Neuroinformatics* 16.1 (2018), pp. 3–13.

[25] Steffen Klamt, Utz-Uwe Haus, and Fabian Theis. "Hypergraphs and cellular networks". In: *PLoS computational biology* 5.5 (2009), e1000385.

[26] Kin K Leung, William A Massey, and Ward Whitt. "Traffic models for wireless communication networks". In: *IEEE Journal on selected areas in Communications* 12.8 (1994), pp. 1353–1364.

[27] Yuelong Li et al. "An algorithm unrolling approach to deep image deblurring". In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2019, pp. 7675–7679.

[28] Zewen Li et al. "A survey of convolutional neural networks: analysis, applications, and prospects". In: *IEEE transactions on neural networks and learning systems* (2021).

[29] Risheng Liu et al. "Knowledge-driven deep unrolling for robust image layer separation". In: *IEEE transactions on neural networks and learning systems* 31.5 (2019), pp. 1653–1666.

[30] Ziwei Liu et al. "Deep learning markov random field for semantic segmentation". In: *IEEE transactions on pattern analysis and machine intelligence* 40.8 (2017), pp. 1814–1828.

[31] Suhas Lohit et al. "Unrolled projected gradient descent for multi-spectral image fusion". In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2019, pp. 7725–7729.

[32] Larry R Medsker and LC Jain. "Recurrent neural networks". In: *Design and Applications* 5 (2001), pp. 64–67.

[33] Vishal Monga, Yuelong Li, and Yonina C Eldar. "Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing". In: *IEEE Signal Processing Magazine* 38.2 (2021), pp. 18–44.

[34] Abubakr Muhammad and Magnus Egerstedt. "Control using higher order Laplacians in network topologies". In: *Proc. of 17th International Symposium on Mathematical Theory of Networks and Systems*. Citeseer. 2006, pp. 1024–1038.

[35] Sayan Mukherjee and John Steenbergen. "Random walks on simplicial complexes and harmonics". In: *Random structures & algorithms* 49.2 (2016), pp. 379–405.

[36] Masatoshi Nagahama et al. "Graph signal restoration using nested deep algorithm unrolling". In: *IEEE Transactions on Signal Processing* 70 (2022), pp. 3296–3311.

[37] Gen Nishida, Adrien Bousseau, and Daniel G Aliaga. "Procedural modeling of a building from a single image". In: *Computer graphics forum*. Vol. 37. 2. Wiley Online Library. 2018, pp. 415–429.

[38] Antonio Ortega et al. "Graph signal processing: Overview, challenges, and applications". In: *Proceedings of the IEEE* 106.5 (2018), pp. 808–828.

[39] Alice Patania, Giovanni Petri, and Francesco Vaccarino. "The shape of collaborations". In: *EPJ Data Science* 6 (2017), pp. 1–16.

[40] Lawrence R Rabiner and Bernard Gold. "Theory and application of digital signal processing". In: *Englewood Cliffs: Prentice-Hall* (1975).

[41] R Ramanathan et al. "Beyond graphs: Capturing groups in networks". In: *2011 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE. 2011, pp. 870–875.

[42] Herbert Robbins and Sutton Monro. "A stochastic approximation method". In: *The annals of mathematical statistics* (1951), pp. 400–407.

[43] Michael Robinson. *Topological signal processing*. Vol. 81. Springer, 2014.

[44] Theodore Roman et al. "A simplicial complex-based approach to unmixing tumor progression data". In: *BMC bioinformatics* 16.1 (2015), pp. 1–17.

[45] Leonid I Rudin, Stanley Osher, and Emad Fatemi. "Nonlinear total variation based noise removal algorithms". In: *Physica D: nonlinear phenomena* 60.1-4 (1992), pp. 259–268.

[46] Aliaksei Sandryhaila and José MF Moura. "Discrete signal processing on graphs". In: *IEEE transactions on signal processing* 61.7 (2013), pp. 1644–1656.

[47] Aliaksei Sandryhaila and José MF Moura. "Discrete signal processing on graphs: Graph Fourier transform". In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE. 2013, pp. 6167–6170.

[48] Michael T Schaub and Santiago Segarra. "Flow smoothing and denoising: Graph signal processing in the edge-space". In: *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. IEEE. 2018, pp. 735–739.

[49] Michael T Schaub et al. "Signal processing on higher-order networks: Livin'on the edge... and beyond". In: *Signal Processing* 187 (2021), p. 108149.

[50] Weijing Shi and Raj Rajkumar. "Point-gnn: Graph neural network for 3d object detection in a point cloud". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 1711–1719.

[51] Oren Solomon et al. "Deep unfolded robust PCA with application to clutter suppression in ultrasound". In: *IEEE transactions on medical imaging* 39.4 (2019), pp. 1051–1063.

[52] Rohan Varma et al. "Vector-valued graph trend filtering with non-convex penalties". In: *IEEE transactions on signal and information processing over networks* 6 (2019), pp. 48–62.

[53] Huy Vu, Gene Cheung, and Yonina C Eldar. "Unrolling of deep graph total variation for image denoising". In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2021, pp. 2050–2054.

[54] Jianian Wang et al. "A review on graph neural network methods in financial applications". In: *arXiv preprint arXiv:2111.15367* (2021).

[55] Yu-Xiang Wang et al. "Trend filtering on graphs". In: *Artificial Intelligence and Statistics*. PMLR. 2015, pp. 1042–1050.

[56] Zhaowen Wang et al. "Deep networks for image super-resolution with sparse prior". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 370–378.

[57] Shiwen Wu et al. "Graph neural networks in recommender systems: a survey". In: *ACM Computing Surveys (CSUR)* (2020).

[58] Zonghan Wu et al. "A comprehensive survey on graph neural networks". In: *IEEE transactions on neural networks and learning systems* 32.1 (2020), pp. 4–24.

[59] Liu Yang, Haifeng Wang, and Hua Qian. "An ADMM-ResNet for data recovery in wireless sensor networks with guaranteed convergence". In: *Digital Signal Processing* 111 (2021), p. 102956.

[60] Maosheng Yang and Elvin Isufi. "Convolutional Learning on Simplicial Complexes". In: *arXiv preprint arXiv:2301.11163* (2023).

[61] Maosheng Yang, Elvin Isufi, and Geert Leus. "Simplicial convolutional neural networks". In: *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2022, pp. 8847–8851.

[62] Maosheng Yang et al. "Simplicial Convolutional Filters". In: *arXiv preprint arXiv:2201.11720* (2022).

[63] Yan Yang et al. "ADMM-CSNet: A deep learning approach for image compressive sensing". In: *IEEE transactions on pattern analysis and machine intelligence* 42.3 (2018), pp. 521–538.

[64] Liang Zhang, Gang Wang, and Georgios B Giannakis. "Real-time power system state estimation and forecasting via deep unrolled neural networks". In: *IEEE Transactions on Signal Processing* 67.15 (2019), pp. 4069–4077.