TMLCN Logo

; revised .

Unrolling of Simplicial ElasticNet for Edge Flow Signal Reconstruction

CHENGEN LIU, GEERT LEUS AND ELVIN ISUFI

[1]Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, Delft, The Netherlands

Corresponding author: CHENGEN LIU (email: C.Liu-35@student.tudelft.nl)

ABSTRACT The edge flow reconstruction task consists of retreiving edge flow signals from corrupted or incomplete measurements. This is typically solved by a regularized optimization problem on higher-order networks such as simplicial complexes and the corresponding regularizers are chosen based on prior knowledge. Tailoring this prior to the setting of interest can be challenging or it may not even be possible. Thus, we consider to learn this prior knowledge via a model-based deep learning approach. We propose a new regularized optimization problem for the simplicial edge flow reconstruction task, the simplicial ElasticNet, which combines the advantages of the ℓ_1 and ℓ_2 norms. We solve the simplicial ElasticNet problem via the multi-block alternating direction method of multipliers (ADMM) algorithm and provide conditions on its convergence. By unrolling the ADMM iterative steps, we develop a model-based neural network with a low requirement on the number of training data. This unrolling network replaces the fixed parameters in the iterative algorithm by learnable weights, thus exploiting the neural network's learning capability while preserving the iterative algorithm's interpretability. We enhance this unrolling network via simplicial convolutional filters to aggregate information from the edge flow neighbors, ultimately, improving the network learning expressivity. Extensive experiments on real-world and synthetic datasets validate the proposed approaches and show considerable improvements over both baselines and traditional non-model-based neural networks.

INDEX TERMS Signal processing over higher-order networks, simplicial convolutional filters, topological signal processing.

I. INTRODUCTION

Reconstructing signals from noisy or partial measurements is a long-lasting challenge in signal processing. This task requires exploiting a particular signal behavior w.r.t. the underlying medium, which typically resorts to framing a structure-based regularized problem [1] [2] [3]. Regularizers introduce a bias between the solution and the actual value, and they possess distinct characteristics. For example, in graph signal processing (GSP) [4], where the data is defined on the nodes of a graph, the Tikhonov regularizer is used to recover the graph signal based on the assumption that connected nodes have similar values [5] [6]. When the signal is piece-wise smooth, graph trend filtering with an ℓ_1 norm regularizer is more effective as it promotes sparsity in the signal difference of connected nodes [7] [8].

In many applications, we are interested in signals defined on the edges of a network such as in transportation, water, or power networks. Therefore, data defined on higher-order networks have been studied recently [9] [10]. One effective way to represent the structure of these flow data is via simplicial complexes [10] which is an algebraic tool to capture multi-way relationships where edge flows can be seen as data over one-simplices [11]. Such oberved edge flows are in practice noisy or have missing values, thus we need to reconstruct them by means of simplicial-based regularized problems. Prior information or physical constraints about the flow behaviour could be used to frame regularized problems so as to estimate the true signal. The papers [12] and [13] respectively solve the edge flow denoising and interpolation task in simplicial complexes based on Tikhonov principles.

However, they only penalize the flow divergence component but not their curl component. This makes them suboptimal as edge flows are often not only divergence-free, but also curl-free which means the net flows circulating along all the triangles are zero, such as currency exchange flows [13]. Secondly, Tikhonov regularizers can only smoothen the divergence and curl components but are unsuitable when the signals present sudden jumps, which commonly occurs in the case of local (i.e., only in a few close-by edges) divergence-free and curl-free flows [8]. Therefore, this paper proposes a simplicial ElasticNet that combines the Tikhonov regularizer with the ℓ_1 regularizer to solve the reconstruction task. The simplicial ElasticNet has a broader range of applications. It is a convex problem and can be solved by various standard iterative algorithms, such as the alternating direction method of multipliers (ADMM) [14] [15].

The regularizers are determined based on the prior knowledge which can often be challenging to obtain [16]. Therefore, it is a viable scheme to learn the prior, which is typically done via model-based neural networks. One such method is the unrolling technique which maps each iteration of the optimization algorithm into a neural network layer [16]. It has been successfully applied in medical imaging [17], power grids [18], remote sensing [19] and graph signal denoising [20]. Compared with unrolling networks, standard neural network models have a black-box nature and are entirely data-driven. Since they are not tailored to the reconstruction task, they need more training data as their function spaces are larger. When the training data is limited, the unrolling network is advantageous because it restricts effectively the function search space by leveraging the priors in the optimization problem and the iterative solutions.

We build unrolling networks for the simplicial ElasticNet via its ADMM block structure. We replace the fixed parameters with learnable simplicial convolutional filters to aggregate information from the neighbors of the edge flow [21]. Consequently, we improve the expressive ability of the unrolling network. Therefore, we make the following contributions:

- We propose an ElasticNet problem for the simplicial edge flow reconstruction task. The regularizer is based on both the ℓ_1 and ℓ_2 norm. It generalizes existing regularization approaches and reconstructs both global and local divergence-free and/or curl-free edge flows. We solve this convex problem by the multi-block ADMM algorithm and provide conditions on its convergence.
- We build the corresponding unrolling network for simplicial ElasticNet (USEN). It is based on the ADMM equations and on simplicial convolutional filters which aggregate information from the neighbors of the edge flow. This also provides a new perspective on simplicial edge flow reconstruction tasks.

We conduct numerical experiments on synthetic and real datasets to corroborate the proposed methods and show their superior performance compared to state-of-the-art simplicial-based regularizers and neural network solutions.

This paper is organized as follows. Section II illustrates some preliminary concepts. In Section III, we propose our simplicial ElasticNet and the corresponding ADMM solution with the convergence analysis. In Section IV, we propose the simplicial unrolling network based on the ADMM steps of the simplicial ElasticNet. Section V shows some experimental results. Finally, Section VI concludes the paper. All proofs are collected in the appendix.

II. PRELIMINARY

In this section, we introduce some concepts related to simplicial complexes and signals.

A. 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 $\mathcal{S}^k \in \mathcal{X}^K$, all of its subsets $\mathcal{S}^{k-1} \subset \mathcal{S}^k$ satisfy $\mathcal{S}^{k-1} \in \mathcal{X}^K$. 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 Figure 1. A graph is therefore a simplicial complex of order $K = 1$.

We can represent the adjacencies between different simplices via the incidence matrices $\mathbf{B}_k \in \mathbb{R}^{N_{k-1} \times N_k}$ which capture the relationship between (*k-1*)-simplices and *k*-simplices [10]. Matrix $\mathbf{B}_1$ is the node-to-edge incidence matrix, $\mathbf{B}_2$ is the edge-to-triangle incidence matrix, and so on. The whole simplicial complex structure can then be represented by the *Hodge Laplacian* matrices

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

Except for $\mathbf{L}_0$ and $\mathbf{L}_K$, all other matrices can be decomposed into the sum of two terms: the *lower Laplacian* $\mathbf{L}_{k,\ell} = \mathbf{B}_k^\top\mathbf{B}_k$ and the *upper Laplacian* $\mathbf{L}_{k,u} = \mathbf{B}_{k+1}\mathbf{B}_{k+1}^\top$. 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).

B. 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 $\mathbf{s}^k = \left[s_1^k, \dots, s_{N_k}^k\right]^\top \in \mathbb{R}^{N_k}$ where entry s_i^k corresponds to the *i*th *k*-simplex [10]. In this paper, we are interested in processing edge flows; hence, we will deal with simplicial complexes of order $K = 2$. Thus, 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 $\mathbf{t} := \mathbf{s}^2 = [t_1, \dots, t_{N_2}]^\top \in \mathbb{R}^{N_2}$. A simplex can 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 orientations are opposite. For processing purposes, we define an arbitrary reference orientation of each simplex and follow for simplicity the lexicographical ordering of the vertices.

The space of the simplicial edge flow signal $\mathbb{R}^{N_1}$ can be decomposed into three orthogonal subspaces

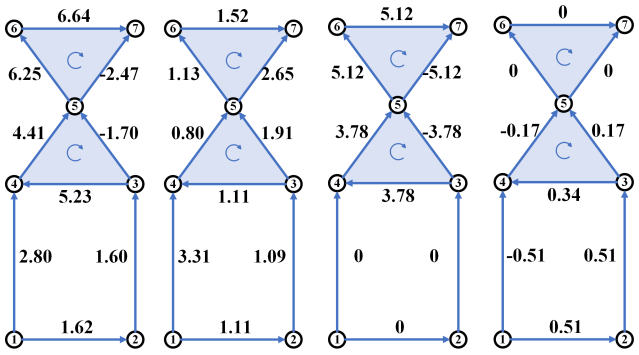
$$\mathbb{R}^{N_1} \equiv \operatorname{im}\left(\mathbf{B}_1^\top\right) \oplus \ker\left(\mathbf{L}_1\right) \oplus \operatorname{im}\left(\mathbf{B}_2\right) \tag{2}$$

where $\operatorname{im}(\cdot)$ and $\ker(\cdot)$ are the image and kernel spaces of a matrix and $\oplus$ is the direct sum. That is, for any edge flow signal $\mathbf{f}$, there exist three simplicial signals of orders 0, 1, and 2 so that we can decompose the edge flow as

$$\mathbf{f} = \mathbf{B}_1^\top\mathbf{v} + \mathbf{f}_{\mathrm{H}} + \mathbf{B}_2\mathbf{t}. \tag{3}$$

This *Hodge decomposition* expresses the relationship between different orders of simplicial signals [10]. It implies that the edge flow can be written as a sum of three flows $\mathbf{f} = \mathbf{f}_{\mathrm{G}} + \mathbf{f}_{\mathrm{C}} + \mathbf{f}_{\mathrm{H}}$ (see also Figure 1) with the explanation:

This article has been accepted for publication in IEEE Open Journal of Signal Processing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/OJSP.2023.3339376



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

FIGURE 1. Decomposition of the edge flow. The simplicial complex contains seven nodes, nine edges, and two (filled) triangles. The edge signals can be decomposed into three different components $\mathbf{f}_G$, $\mathbf{f}_C$ and $\mathbf{f}_H$: $\mathbf{f}_G$ is the gradient component; $\mathbf{f}_C$ is the curl component; and $\mathbf{f}_H$ is the harmonic component.

- *Gradient component*: $\mathbf{f}_{\mathrm{G}} = \mathbf{B}_1^\top \mathbf{v} \in \mathrm{im}(\mathbf{B}_1^\top)$ 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 the gradient operator and the space $\mathrm{im}\,(\mathbf{B}_1^\top)$ is the gradient space.
- *Curl component*: $\mathbf{f}_C = \mathbf{B}_2\mathbf{t} \in \mathrm{im}(\mathbf{B}_2)$ is a curl flow locally circulating along the edges of triangles induced by a triangle signal $\mathbf{t}$. Operator $\mathbf{B}_2$ is the curl adjoint and the space $\mathrm{im}\,(\mathbf{B}_2)$ is the curl space.
- *Harmonic component*: $\mathbf{f}_{\mathrm{H}} \in \ker(\mathbf{L}_1)$ 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.

For future reference, we define the following two operators:

- *Curl operator*: $\mathrm{curl}(\mathbf{f}) = \mathbf{B}_2^\top \mathbf{f}$ which yields a triangle signal that measures the curl of an edge flow. The *i*th element corresponds to the sum of the flow of each edge forming the *i*th 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 [10].
- *Divergence operator*: $\mathrm{div}(\mathbf{f}) = \mathbf{B}_1\mathbf{f}$ which yields a node signal and measures the divergence of an edge flow. The *i*th element corresponds to the flow passing through the *i*th node. If the divergence of an edge flow is zero at each node, it is divergence-free. The curl component $\mathbf{f}_C$ and the harmonic component $\mathbf{f}_{\mathrm{H}}$ are divergence-free by definition [10].

Real edge flows tend to be divergence-free or curl-free. For example, the traffic flow entering into a junction in a road network is equal to the traffic flow out going that junction, which means it is divergence-free. Another example is the principle of non-arbitrage in the foreign exchange market, which implies that the conversion between exchange rates should be curl-free [13].

III. SIMPLICIAL ELASTICNET

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

$$\underset{\hat{\mathbf{f}}\in\mathbb{R}^{N_1}}{\operatorname{argmin}} \quad \|\mathbf{P}(\hat{\mathbf{f}}-\mathbf{y})\|_2^2 + \textstyle\sum_{i=1}^{n} r_i(\hat{\mathbf{f}}, \mathcal{S}^1) \tag{4}$$

where $\|\mathbf{P}(\hat{\mathbf{f}}-\mathbf{y})\|_2^2$ is the data-fitting term. The terms $r_i(\hat{\mathbf{f}}, \mathcal{S}^1)$ represent the regularizers, which are monotone non-decreasing functions that penalize a particular behavior of the edge flows w.r.t. the 1-simplex $\mathcal{S}^1$. We consider problem (4) as a unified formulation for 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 on a subset of the edges. When $\mathbf{P} = \mathbf{I}$ and $\mathbf{y} = \mathbf{f}_0 + \mathbf{n}$ is a noisy edge flow, the optimization problem corresponds to the denoising task. Here, $\mathbf{f}_0$ is the real edge flow and $\mathbf{n}$ is the additive Gaussian noise. When $\mathbf{P} \in \{0,1\}^{M\times N_1}$ is a sampling matrix with $M \leq N_1$ sampled values, the optimization problem corresponds to the interpolation task.

A. SIMPLICIAL ELASTICNET PROBLEM

To regularize problem (4) with a simplicial prior, we consider the ElasticNet [22] principle w.r.t. the three signal components in (3) which contains both ℓ_1 and ℓ_2 norm regularizers

$$\begin{aligned}\underset{\hat{\mathbf{f}}\in\mathbb{R}^{N_1}}{\operatorname{argmin}} \; &\|\mathbf{P}(\hat{\mathbf{f}}-\mathbf{y})\|_2^2 + \alpha_1\|\mathbf{B}_1\hat{\mathbf{f}}\|_1 + \alpha_2\|\mathbf{B}_1\hat{\mathbf{f}}\|_2^2 + \beta_1\|\mathbf{B}_2^\top\hat{\mathbf{f}}\|_1 \\ &+ \beta_2\|\mathbf{B}_2^\top\hat{\mathbf{f}}\|_2^2 + \gamma_1\|\hat{\mathbf{f}}\|_1 + \gamma_2\|\hat{\mathbf{f}}\|_2^2\end{aligned} \tag{5}$$

where α_1, α_2, β_1, β_2, γ_1, γ_2 are all positive constants. There are three regularization pairs in (5) that are reminiscent of the Hodge decomposition in (3):

- The first pair $\alpha_1\|\mathbf{B}_1\hat{\mathbf{f}}\|_1 + \alpha_2\|\mathbf{B}_1\hat{\mathbf{f}}\|_2^2$ regularizes the divergence component of the edge flows by promoting the divergence to be sparse via the ℓ_1 norm and low-energy via the ℓ_2 norm.
- The second pair $\beta_1\|\mathbf{B}_2^\top\hat{\mathbf{f}}\|_1 + \beta_2\|\mathbf{B}_2^\top\hat{\mathbf{f}}\|_2^2$ regularizes the curl component of the edge flows. The ℓ_1 norm promotes the sparsity of the curl on the triangles while the ℓ_2 norm reduces the total curl of the recovered signal globally.
- The last pair $\gamma_1\|\hat{\mathbf{f}}\|_1 + \gamma_2\|\hat{\mathbf{f}}\|_2^2$ contains the additional regularizers that guarantee the completes of the optimization problem; that is, having an overall signal that is either sparse or of low energy. In most tasks, these two terms are redundant, hence, γ_1 and γ_2 can be set to zero.

Scalars α_1, α_2, β_1, β_2, γ_1, and γ_2 control the trade-off between the fidelity, the divergence, and the curl of the recovered signal. Problem (5) generalizes two existing edge flow reconstruction problems.

Tikhonov regularizer [12] [13]. When the parameters $\alpha_1 = \beta_1 = \gamma_1 = \gamma_2 = 0$, a common optimization problem for the edge flow recovery becomes

$$\underset{\hat{\mathbf{f}}\in\mathbb{R}^{N_1}}{\operatorname{argmin}} \; \|\mathbf{P}(\hat{\mathbf{f}}-\mathbf{y})\|_2^2 + \alpha_2\|\mathbf{B}_1\hat{\mathbf{f}}\|_2^2 + \beta_2\|\mathbf{B}_2^\top\hat{\mathbf{f}}\|_2^2 \tag{6}$$

where the regularizers force the recovered signal to have low divergence and curl.

Simplicial trend filtering [23][1] When we consider only the sparsity of the 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

$$\underset{\hat{\mathbf{f}}\in\mathbb{R}^{N_1}}{\operatorname{argmin}} \|\mathbf{P}(\hat{\mathbf{f}}-\mathbf{y})\|_2^2 + \alpha_1\|\mathbf{B}_1\hat{\mathbf{f}}\|_1 + \beta_1\|\mathbf{B}_2^\top\hat{\mathbf{f}}\|_1 \tag{7}$$

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 benefits from 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 are missing. The reason is that the ElasticNet makes a trade-off between the ℓ_1 norm and ℓ_2 norm.

B. ADMM SOLUTION FOR SIMPLICIAL ELASTICNET

The optimization problem in (5) is a convex problem and there are several blocks in its objective function. It can be solved by a multi-block ADMM algorithm [24]. 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 problem (5) reformulates as

$$\begin{aligned}\underset{\hat{\mathbf{f}}\in\mathbb{R}^{N_1}}{\operatorname{argmin}} &\|\mathbf{P}(\hat{\mathbf{f}}-\mathbf{y})\|_2^2 + \alpha_1\|\mathbf{z}_1\|_1 + \alpha_2\|\mathbf{z}_1\|_2^2 + \beta_1\|\mathbf{z}_2\|_1 \\ &+ \beta_2\|\mathbf{z}_2\|_2^2 + \gamma_1\|\mathbf{z}_3\|_1 + \gamma_2\|\mathbf{z}_3\|_2^2 \\ \text{subject to}\quad & \mathbf{B}_1\hat{\mathbf{f}} = \mathbf{z}_1, \mathbf{B}_2^\top\hat{\mathbf{f}} = \mathbf{z}_2, \hat{\mathbf{f}} = \mathbf{z}_3.\end{aligned} \tag{8}$$

The corresponding augmented Lagrangian is

$$\begin{aligned}L =&\|\mathbf{P}(\hat{\mathbf{f}}-\mathbf{y})\|_2^2 + \alpha_1\|\mathbf{z}_1\|_1 + \alpha_2\|\mathbf{z}_1\|_2^2 + \beta_1\|\mathbf{z}_2\|_1 \\ &+ \beta_2\|\mathbf{z}_2\|_2^2 + \gamma_1\|\mathbf{z}_3\|_1 + \gamma_2\|\mathbf{z}_3\|_2^2 - \boldsymbol{\lambda}_1^\top(\mathbf{B}_1\hat{\mathbf{f}} - \mathbf{z}_1) \\ &- \boldsymbol{\lambda}_2^\top(\mathbf{B}_2^\top\hat{\mathbf{f}} - \mathbf{z}_2) - \boldsymbol{\lambda}_3^\top(\hat{\mathbf{f}} - \mathbf{z}_3) + \frac{\rho}{2}\|\mathbf{B}_1\hat{\mathbf{f}} - \mathbf{z}_1\|_2^2 \\ &+ \frac{\rho}{2}\|\mathbf{B}_2^\top\hat{\mathbf{f}} - \mathbf{z}_2\|_2^2 + \frac{\rho}{2}\|\hat{\mathbf{f}} - \mathbf{z}_3\|_2^2\end{aligned} \tag{9}$$

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

$$\begin{cases}\hat{\mathbf{f}}^{(k+1)} =(2\mathbf{P}^\top\mathbf{P} + \rho\mathbf{B}_1^\top\mathbf{B}_1 + \rho\mathbf{B}_2\mathbf{B}_2^\top + \rho\mathbf{I})^{-1} \\ \qquad\qquad (2\mathbf{P}^\top\mathbf{P}\mathbf{y} + \mathbf{B}_1^\top\boldsymbol{\lambda}_1^{(k)} + \mathbf{B}_2\boldsymbol{\lambda}_2^{(k)} + \boldsymbol{\lambda}_3^{(k)} \\ \qquad\qquad + \rho\mathbf{B}_1^\top\mathbf{z}_1^{(k)} + \rho\mathbf{B}_2\mathbf{z}_2^{(k)} + \rho\mathbf{z}_3^{(k)}) \\ \mathbf{z}_1^{(k+1)} =S_{\frac{\alpha_1}{2\alpha_2+\rho}}\left(\frac{1}{2\alpha_2+\rho}(\rho\mathbf{B}_1\hat{\mathbf{f}}^{(k+1)} - \boldsymbol{\lambda}_1^{(k)})\right) \\ \mathbf{z}_2^{(k+1)} =S_{\frac{\beta_1}{2\beta_2+\rho}}\left(\frac{1}{2\beta_2+\rho}(\rho\mathbf{B}_2^\top\hat{\mathbf{f}}^{(k+1)} - \boldsymbol{\lambda}_2^{(k)})\right) \\ \mathbf{z}_3^{(k+1)} =S_{\frac{\gamma_1}{2\gamma_2+\rho}}\left(\frac{1}{2\gamma_2+\rho}(\rho\hat{\mathbf{f}}^{(k+1)} - \boldsymbol{\lambda}_3^{(k)})\right) \\ \boldsymbol{\lambda}_1^{(k+1)} =\boldsymbol{\lambda}_1^{(k)} - \rho(\mathbf{B}_1\hat{\mathbf{f}}^{(k+1)} - \mathbf{z}_1^{(k+1)}) \\ \boldsymbol{\lambda}_2^{(k+1)} =\boldsymbol{\lambda}_2^{(k)} - \rho(\mathbf{B}_2^\top\hat{\mathbf{f}}^{(k+1)} - \mathbf{z}_2^{(k+1)}) \\ \boldsymbol{\lambda}_3^{(k+1)} =\boldsymbol{\lambda}_3^{(k)} - \rho(\hat{\mathbf{f}}^{(k+1)} - \mathbf{z}_3^{(k+1)})\end{cases} \tag{10}$$

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 ADMM.

Proposition 1 (convergence): Assume that $\mathbf{P}^\top\mathbf{P}$ is a positive definite matrix and there exists a constant $\mu_1 > 0$ satisfying $\mathbf{P}^\top\mathbf{P} \succeq \mu_1\mathbf{I}$. Each regularization block in the cost function is a strongly convex function with modulus μ_i [24] satisfying $\mu_2 = 2\alpha_2, \mu_3 = 2\beta_2, \mu_4 = 2\gamma_2$. Consider also the matrices $\mathbf{A}_i$ related to the equality constraints of (8) defined as

$$\mathbf{A}_1 = [\mathbf{B}_1, \mathbf{B}_2^\top, \mathbf{I}] \in \mathbb{R}^{(N_0+N_1+N_2)\times N_1} \tag{11a}$$

$$\mathbf{A}_2 = [-\mathbf{I},\ \mathbf{0},\ \mathbf{0}] \in \mathbb{R}^{(N_0+N_1+N_2)\times N_0} \tag{11b}$$

$$\mathbf{A}_3 = [\mathbf{0},\ -\mathbf{I},\ \mathbf{0}] \in \mathbb{R}^{(N_0+N_1+N_2)\times N_2} \tag{11c}$$

$$\mathbf{A}_4 = [\mathbf{0},\ \mathbf{0},\ -\mathbf{I}] \in \mathbb{R}^{(N_0+N_1+N_2)\times N_1}. \tag{11d}$$

If the penalty parameter ρ satisfies

$$0 < \rho < \min_{1\le i\le 4}\left\{\frac{2\mu_i}{9\|\mathbf{A}_i\|_2^2}\right\}, \tag{12}$$

the four-block ADMM iterative steps converge to the optimal solution of the problem (5).

Proof. See Appendix A.

When we specify (5) as the reconstruction task, this proposition only works for the denoising task as $\mathbf{P}^\top\mathbf{P}$ is positive semidefinite in the interpolation task.

IV. SIMPLICIAL UNROLLING NETWORKS

Choosing appropriate regularization coefficients is critical to achieve a satisfactory performance by solving problem (5). However, such a prior knowledge may be unavailable or unclear to be framed as an explicit regularizer. In these cases, the regularization coefficients need to be selected empirically which often leads to a gap between the obtained and optimal solutions. This problem can be avoided by exploiting the learning ability of an unrolling network. The unrolling network maps each iteration of the iterative algorithm into one neural network layer and replaces the fixed parameters with learnable ones [16]. Therefore, it leads to an architecture that is tailored to the problem at hand. The unrolling network restricts the degrees of freedom by using such an iteration as inductive bias, ultimately, demanding less training data.

[1]This can be considered as a preliminary version of this paper presented at Asilomar. The previous work contains only ℓ_1 terms. Therefore, this approach is an extension of this previous work, which contains both the ℓ_1 and ℓ_2 terms to promote the sparsity and keep the low energy, respectively. The work in [23] does not focus on the unrolling technique as we do here.

Algorithm 1: Learning framework of the USEN

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

Function USEN($\mathbf{P}, \mathbf{y}, \mathbf{f}_0, L$);
for $l = 1 : L$ **do**
  update $\hat{\mathbf{f}}^{(l+1)}$, $\mathbf{z}_i^{(l+1)}$ and $\lambda_i^{(l+1)}$ by (13);
end
Minimize $\|\hat{\mathbf{f}}^{(L)} - \mathbf{f}_0\|_2^2$ and update all weights;

One of the most straightforward ways to construct an unrolling network is to use the iterative step in (10) and learn the scalars in it. However, this involves an inverse operation, which increases the computational complexity. Instead, we want to avoid it and use simplicial convolutional filters to learn the propagation rule in the $\hat{\mathbf{f}}$-update. The corresponding lth layer of the unrolling network for the simplicial ElasticNet (USEN) can then be computed as

$$\begin{cases} \hat{\mathbf{f}}^{(l+1)} = \mathbf{H}_1\mathbf{P}^\top\mathbf{P}\mathbf{y} + \mathbf{H}_2\mathbf{B}_1^\top\boldsymbol{\lambda}_1^{(l)} + \mathbf{H}_3\mathbf{B}_2\boldsymbol{\lambda}_2^{(l)} + \mathbf{H}_4\boldsymbol{\lambda}_3^{(l)} \\ \qquad + \mathbf{H}_5\mathbf{B}_1^\top\mathbf{z}_1^{(l)} + \mathbf{H}_6\mathbf{B}_2\mathbf{z}_2^{(l)} + \mathbf{H}_7\mathbf{z}_3^{(k)} \\ \mathbf{z}_1^{(l+1)} = S_{\frac{a_1}{2a_2+r_1}}\left(\frac{1}{2a_2+r_1}(r_1\mathbf{B}_1\hat{\mathbf{f}}^{(l+1)} - \boldsymbol{\lambda}_1^{(l)})\right) \\ \mathbf{z}_2^{(l+1)} = S_{\frac{b_1}{2b_2+r_2}}\left(\frac{1}{2b_2+r_2}(r_2\mathbf{B}_2^\top\hat{\mathbf{f}}^{(l+1)} - \boldsymbol{\lambda}_2^{(l)})\right) \\ \mathbf{z}_3^{(l+1)} = S_{\frac{c_1}{2c_2+r_3}}\left(\frac{1}{2c_2+r_3}(r_3\hat{\mathbf{f}}^{(l+1)} - \boldsymbol{\lambda}_3^{(l)})\right) \\ \boldsymbol{\lambda}_1^{(l+1)} = \boldsymbol{\lambda}_1^{(l)} - r_1(\mathbf{B}_1\hat{\mathbf{f}}^{(l+1)} - \mathbf{z}_1^{(l+1)}) \\ \boldsymbol{\lambda}_2^{(l+1)} = \boldsymbol{\lambda}_2^{(l)} - r_2(\mathbf{B}_2^\top\hat{\mathbf{f}}^{(l+1)} - \mathbf{z}_2^{(l+1)}) \\ \boldsymbol{\lambda}_3^{(l+1)} = \boldsymbol{\lambda}_3^{(l)} - r_3(\hat{\mathbf{f}}^{(l+1)} - \mathbf{z}_3^{(l+1)}) \end{cases} \tag{13}$$

where the $\mathbf{H}_i := \mathbf{H}(\mathbf{L}_{1,l}, \mathbf{L}_{1,u})$ are simplicial convolutional filters defined as follows. Given a simplicial edge flow $\mathbf{f}$ and Laplacians $\mathbf{L}_{1,\ell}$ and $\mathbf{L}_{1,u}$, the *simplicial convolutional filtering operation* is defined as

$$\mathbf{y} = \left(\sum_{l_1=0}^{L_1}\alpha_{l_1}\mathbf{L}_{1,\ell}^{l_1} + \sum_{l_2=0}^{L_2}\beta_{l_2}\mathbf{L}_{1,u}^{l_2}\right)\mathbf{f} \tag{14}$$

where $\mathbf{y}$ is the filter output, L_1 and L_2 are the convolution orders, and α_{l_1}, β_{l_2} are the coefficients. A simplicial convolutional filter is thus defined as

$$\mathbf{H} := \left(\sum_{l_1=0}^{L_1}\alpha_{l_1}\mathbf{L}_{1,\ell}^{l_1} + \sum_{l_2=0}^{L_2}\beta_{l_2}\mathbf{L}_{1,u}^{l_2}\right) \tag{15}$$

which allows writing (15) as $\mathbf{y} = \mathbf{H}\mathbf{f}$. This convolution operator propagates the edge flow signal $\mathbf{f}$ via upper and lower neighbors, weighs differently each shift, and sums all the shifted signals. Specifically, operations $\sum_{l_1=0}^{L_1}\alpha_{l_1}\mathbf{L}_{1,\ell}^{l_1}$ and $\sum_{l_2=0}^{L_2}\beta_{l_2}\mathbf{L}_{1,u}^{l_2}$ gather the information from the lower- and the upper-adjacencies up to L_1 and L_2 hops away, respectively [21]. The learning framework of the USEN is shown in Algorithm 1 where $\mathbf{P}$ is the sampling matrix, $\mathbf{y}$ collects the measurement, $\mathbf{f}_0$ is the clean signal, and L is the number of layers.

This USEN replaces the update step of $\hat{\mathbf{f}}$ with trainable simplicial convolutional filters. This allows learning the influence of the multi-hop edge flow neightbors into reconstructing the signal by acting on the respective Hodge decomposition spectrum. The USEN also replaces ρ at different positions with three different trainable parameters r_1, r_2 and r_3. The fixed parameters α_i, β_i and γ_i in ADMM are replaced by the trainable weights a_i, b_i and c_i. Each convolutional filter has $1 + L_1 + L_2$ parameters. Thus, each layer has $19 + 7L_1 + 7L_2$ parameters in total. These parameters ensure that USEN is flexible and expressive. The major complexity comes from the computation of convolutional filters, which is a weighted linear combination of different shifts of edge flow signals. Therefore, the complexity of one layer of the USEN is $\mathcal{O}\left((L_1 + L_2)\, D\right)$ where D is the dimension of the input data. This complexity grows L times with the depth. The unrolling network based on trend filtering is shown in Appendix B.

TABLE 1. Properties of the datasets.

Datasets	Nodes	Edges	Triangles	Property
Forex	25	300	2300	curl-free
Lastfm	657	1997	1276	divergence-free
Chicago	546	1088	122	divergence-free

V. EXPERIMENTAL RESULTS

This section evaluates the proposed approaches on the edge flow recovery task.

A. DATASETS

In this section we give an overview of the datasets that are considered. The properties are summarized in Table 1.

Foreign Currency Exchange (Forex) [13]. 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. This exchange rate value must guarantee the no-arbitrage condition, i.e., an income cannot be obtained through repeated exchange between currency pairs. For currencies A and B, the exchange rate is $r^{\mathrm{A/B}}$ and the no-arbitrage condition implies $r^{\mathrm{A/B}}r^{\mathrm{B/C}} = r^{\mathrm{A/C}}$. 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. 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 task is to recover the exchange rates under the arbitrary free condition which is relevant in noisy fluctuation settings or when anomalies may be present.

Lastfm [13]. 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 B, we add a unit on the edge flow from A to

B. Edge flows modeled in this way should be approximately divergence-free. 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 $\|\mathbf{B}_1\mathbf{f}\|_1 = 0$ or $\|\mathbf{B}_1\mathbf{f}\|_2^2 = 0$. The Lastfam dataset can be modeled as a simplicial complex with 657 nodes, 1997 edges, and 1276 triangles. We generate a synthetic (divergence-free) curl component based on this topology by $\mathbf{B}_2\mathbf{t}$ where $\mathbf{t}$ is a random triangle signal. The task consists of recovering edge flows from noisy or partial measurements which are typical when the information of users is inaccurate.

Chicago road network [21]. This is the road network of the city of Chicago and contains 546 nodes, 1088 edges and 112 triangles. Junctions are modeled as nodes, roads as edges and the area enclosed by three roads as triangles. We 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 considered to corrupt the original edge-flow signal. The task here consists of estimating the edge flows from noisy and partial measurements which is relevant for traffic monitoring from a few sensors.

B. EXPERIMENTAL SETUP

We compare different iterative models including:

- ADMM-SEN: ADMM for simplicial ElasticNet in (12).
- ADMM-STF [23]: ADMM for simplicial trend filtering in (18). This acts as a baseline for the proposed ADMM-SEN and has shown a superior performance than the Tikhonov counterpart (6).
- USEN: Simplicial unrolling network for ElasticNet (15).
- USTF: Simplicial unrolling network for trend filtering in (19). This model unrolls the ADMM-STF and has fewer trainable weights than USEN.
- MLP [25]: Multilayer perceptrons are fully connected neural networks which act as a baseline that ignores any signal structure.
- SNN [26]: Simplicial neural networks developed for processing simplicial signals which are non-model-based neural networks.
- SCNN [27]: Simplicial convolutional neural networks. Differently from [26], it puts different weights on the lower and upper edge flow propagations. Together with the SNN it acts as a baseline to highlight the importance of a model-based approach over simplices.
- GUTF [20]: Graph unrolling network for trend filtering. We build a line graph [12] where edges become nodes and viceversa and consider the edge flows as node signals. Then we deploy the approach of [20] to show the reconstruction performance.

The ratio of training, validation, and testing samples is 1:1:10. Hyperparameters such as the number of layers and learning rate l_r are shown in Tables 2 and 3. The number of layers is searched from 1 to 6, and the learning rate l_r ranges from 0.001 to 0.1. The batch size is 1 for all experiments. The number of layers in the MLP is 5, and the number of neurons in each layer is 16, 128, 128, 16, 1. The number of layers of SNN and SCNN is 3, and the number of features output from each layer is 2, 2, 1, respectively. The order of the filters in SNN is $k = 1$ and in SCNN are $L_1 = L_2 = 1$. K is the number of iterations in ADMM. The Adam optimizer is used in all experiments to update the learnable weights. All neural networks are trained using one-shot learning (i.e., with a single training point). We ran all methods until convergence and the convergence criterion is the maximum number of iterations K.

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 [13]. In denoising tasks, the NMSE measures the difference between the denoised signal and the real signal. The Pearson correlation coefficient measures the degree of correlation between the reconstructed signal and the real signal. We test the performance of the models on ten different noisy realizations and report the average values.

C. PERFORMANCE COMPARISON

Table 2 shows the NMSE of the edge flow denoising task. Table 3 shows the Pearson correlation coefficients for the edge flow interpolation. Overall, ADMM-SEN achieves close or slightly better results than ADMM-STF. This is because ADMM-SEN adds regularizers based on the ℓ_2 norm, which makes the recovered signal have a lower divergence or curl. It is worth noting that ADMM-SEN requires fewer iterations to converge for the Forex dataset. The proposed unrolling networks USEN and USTF achieve a significantly better performance than the non-model-based neural networks. The need for substantial amounts of labeled data makes it challenging for standard neural networks to achieve satisfactory results. The unrolling networks are designed based on the mathematical models that are tailored to specific inverse problems on simplices. This indicates that unrolling networks will search a much smaller function space than other neural networks. Compared with the iterative algorithms, the unrolling network works better when the prior is inaccurate. The advantage of the unrolling network is more apparent when the SNR is close to zero or sampling rate is close to 20% because it can capture the patterns in the signal more accurately even with a significant noise or low sampling rate.

We note that the two ADMM iterative algorithms perform significantly better on the Forex dataset than the others. 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; thus, when there are more triangles

TABLE 2. NMSE of denoising task. The SNR ranges from 0dB to 10dB. The unrolling networks achieve the best denoising effect.

FOREX	Parameter	0dB	2dB	4dB	6dB	8dB	10dB
MLP[25]	$L=5, l_r=0.01$	0.71	0.52	0.39	0.28	0.24	0.12
SNN[26]	$L=3, l_r=0.01$	0.49	0.39	0.31	0.20	0.13	0.09
SCNN[27]	$L=3, l_r=0.01$	0.11	0.08	0.07	0.04	0.03	0.01
GUTF[20]	$L=2, l_r=0.001$	0.46	0.36	0.28	0.19	0.14	0.09
ADMM-STF[23]	$K=100$	0.07	0.04	0.03	0.02	0.01	0.01
ADMM-SEN(ours)	$K=50$	0.07	0.04	0.03	0.02	0.01	0.01
USEN(ours)	$L=4, l_r=0.01$	0.07	0.04	0.03	0.02	0.01	0.01
USTF(ours)	$L=6, l_r=0.001$	**0.07**	**0.04**	**0.03**	**0.02**	**0.01**	**0.01**
LASTFM	Parameter	0dB	2dB	4dB	6dB	8dB	10dB
MLP[25]	$L=5, l_r=0.01$	0.39	0.30	0.22	0.16	0.11	0.07
SNN[26]	$L=3, l_r=0.001$	0.27	0.21	0.16	0.12	0.09	0.07
SCNN[27]	$L=3, l_r=0.01$	0.21	0.19	0.12	0.11	0.07	0.05
GUTF[20]	$L=2, l_r=0.01$	0.87	0.87	0.87	0.87	0.87	0.86
ADMM-STF[23]	$K=100$	0.66	0.42	0.26	0.16	0.10	0.06
ADMM-SEN(ours)	$K=100$	0.66	0.42	0.26	0.16	0.10	0.06
USEN(ours)	$L=2, l_r=0.03$	**0.09**	**0.08**	**0.06**	**0.05**	**0.04**	**0.03**
USTF(ours)	$L=2, l_r=0.001$	0.09	0.08	0.07	0.06	0.05	0.04
CHICAGO	Parameter	0dB	2dB	4dB	6dB	8dB	10dB
MLP[25]	$L=5, l_r=0.01$	0.53	0.40	0.30	0.21	0.14	0.09
SNN[26]	$L=3, l_r=0.01$	0.44	0.33	0.25	0.18	0.13	0.10
SCNN[27]	$L=3, l_r=0.01$	0.37	0.26	0.20	0.15	0.08	0.05
GUTF[20]	$L=2, l_r=0.01$	0.76	0.70	0.65	0.61	0.58	0.55
ADMM-STF[23]	$K=100$	0.49	0.31	0.19	0.12	0.07	0.04
ADMM-SEN(ours)	$K=100$	0.49	0.31	0.19	0.12	0.07	0.04
USEN(ours)	$L=6, l_r=0.05$	**0.34**	**0.24**	**0.17**	**0.11**	**0.07**	**0.05**
USTF(ours)	$L=4, l_r=0.01$	0.35	0.25	0.17	0.11	0.08	0.05

TABLE 3. Pearson correlation coefficient of interpolation task. The sampling rate ranges from 20% to 90%. The ADMM performs best on the Forex and Chicago datasets while the USEN performs best on the Lastfm dataset.

FOREX	Parameter	20%	30%	40%	50%	60%	70%	80%	90%
MLP[25]	$L=5, l_r=0.01$	0.44	0.54	0.63	0.69	0.76	0.83	0.89	0.94
SNN[26]	$L=3, l_r=0.001$	0.44	0.53	0.62	0.68	0.76	0.82	0.88	0.94
SCNN[27]	$L=3, l_r=0.001$	0.79	0.90	0.93	0.97	0.97	0.97	0.96	0.98
GUTF[20]	$L=2, l_r=0.001$	0.48	0.57	0.65	0.70	0.78	0.83	0.89	0.94
ADMM-STF[23]	$K=500$	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**
ADMM-SEN(ours)	$K=300$	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**
USEN(ours)	$L=6, l_r=0.001$	0.85	0.89	0.95	0.97	0.98	0.98	0.99	0.99
USTF(ours)	$L=4, l_r=0.001$	0.86	0.90	0.96	0.97	0.98	0.98	0.99	0.99
LASTFM	Parameters	20%	30%	40%	50%	60%	70%	80%	90%
MLP[25]	$L=5, l_r=0.01$	0.43	0.53	0.61	0.69	0.77	0.83	0.90	0.95
SNN[26]	$L=3, l_r=0.01$	0.44	0.55	0.62	0.71	0.78	0.84	0.90	0.95
SCNN[27]	$L=3, l_r=0.05$	0.30	0.60	0.68	0.70	0.77	0.87	0.88	0.94
GUTF[20]	$L=2, l_r=0.1$	0.15	0.18	0.21	0.23	0.24	0.25	0.26	0.27
ADMM-STF[23]	$K=50$	0.26	0.36	0.44	0.54	0.63	0.76	0.85	**0.98**
ADMM-SEN(ours)	$K=50$	0.26	0.36	0.44	0.54	0.63	0.76	0.85	**0.98**
USEN(ours)	$L=2, l_r=0.01$	**0.92**	**0.92**	**0.92**	**0.93**	**0.93**	**0.94**	**0.95**	0.97
USTF(ours)	$L=2, l_r=0.001$	0.92	0.92	0.92	0.93	0.93	0.94	0.95	0.97
CHICAGO	Parameters	20%	30%	40%	50%	60%	70%	80%	90%
MLP[25]	$L=5, l_r=0.01$	0.45	0.55	0.64	0.71	0.78	0.84	0.90	0.95
SNN[26]	$L=3, l_r=0.1$	0.53	0.63	0.73	0.81	0.83	0.91	0.91	0.96
SCNN[27]	$L=3, l_r=0.1$	0.51	0.67	0.74	0.80	0.85	0.92	0.94	0.97
GUTF[20]	$L=2, l_r=0.01$	0.30	0.36	0.42	0.47	0.52	0.57	0.62	0.66
ADMM-STF[23]	$K=100$	**0.60**	**0.74**	**0.83**	**0.92**	**0.97**	**0.99**	**0.99**	**0.99**
ADMM-SEN(ours)	$K=100$	**0.60**	**0.74**	**0.83**	**0.92**	**0.97**	**0.99**	**0.99**	**0.99**
USEN(ours)	$L=3, l_r=0.01$	0.56	0.66	0.75	0.81	0.86	0.90	0.94	0.97
USTF(ours)	$L=3, l_r=0.01$	0.56	0.66	0.74	0.80	0.85	0.90	0.94	0.97

in simplicial complexes, the curl-free property provides more information. Furthermore, the topology of the Forex dataset is a complete simplicial complex, meaning a filled triangle exists among any three nodes. Therefore, the curl-free property is a solid prior for it.

GUTF transforms the edge flow signals into node signals in the corresponding line graph and is a graph unrolling network based on the trend filtering to complete the edge flow reconstruction task. It achieves unsatisfactory results because the underlying optimization problem does not use 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- or divergence-free; hence, advocating simplicial-based alternatives.

D. EFFECT OF SIMPLICIAL CONVOLUTIONAL FILTERS

We verify the contribution of adding trainable simplicial convolutional filters in the unrolling network. Figure 2 shows the effect of trainable filters in the denoising and interpolation tasks, respectively. We take the results on the Lastfm dataset as an example because the difference between its curves is the most obvious and more noticeable compared to the Forex and Chicago datasets. When the filters are removed, the reconstruction becomes worse because the simplicial convolutional filters improve the learning ability as they aggregate the information of the edge flow and its neighbors at each layer. The gap is larger for the USTF, especially for low SNR or low sampling rates which indicates that USEN can capture the patterns in the signal more accurately than USTF even in a more challenging setting.

E. CONVERGENCE

We check 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 3 (top), where the multi-block ADMM algorithm is always guaranteed to converge as ρ varies. The larger ρ, the faster the convergence. As for USEN, we conduct experiments on all the datasets. We check the output of each layer and the number of layers is gradually increased. The convergence results of USEN are shown in Figure 3 (bottom). The NMSE and the Pearson correlation coefficients of the recovered edge flow generated by USEN converge gradually as the number of layers is gradually increased. The convergence property of USEN has some similarities with the ADMM because it is constructed based on its iterative steps.

VI. CONCLUSION

We propose the simplicial ElasticNet for the edge flow reconstruction task. It contains both the ℓ_1 and ℓ_2 norm regularizers which promote sparsity and keep low energy, respectively. We solve the simplicial ElasticNet and trend filtering problem by multi-block ADMM iteratively. Then, we design the corresponding unrolling networks USEN and USTF based on their ADMM steps. The core idea is mapping each iteration into a layer of the neural network. Simplicial convolutional filters are considered in the unrolling networks to collect information from the neighbors of the edge flows and we learn the filter parameters to improve the learning capabilities of the network. Numerical experiments show that the simplicial unrolling network can achieve better reconstruction results than non-model-based neural networks and other unrolling algorithms with limited training data. In practice, when the prior knowledge is unclear, the learning ability of an unrolling network can be beneficial. Traditional iterative algorithms still have advantages when there is no data to learn from. In future research, an unrolling network can be developed for different iterative algorithms.

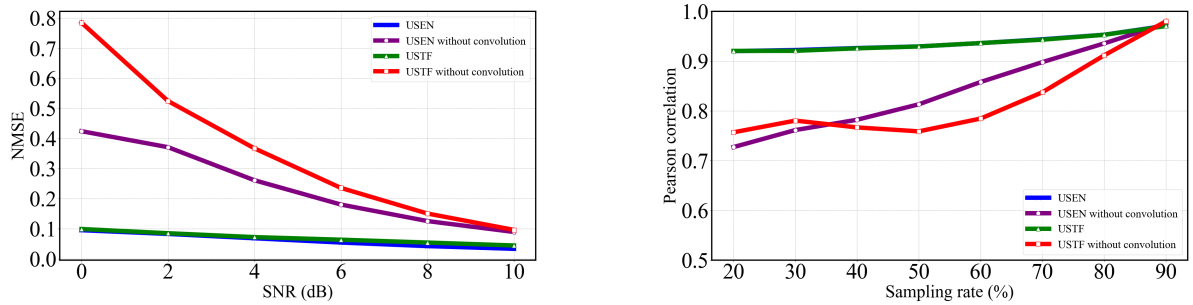


FIGURE 2. **Reconstruction performance of different parametric strategies on the Lastfm dataset. SNR for denoising task is ranging from 0 dB to 10 dB. Sampling rate for interpolation task is ranging from 20% to 90%. USEN and USTF are unrolling networks containing simplicial convolutional filters, while USEN without convolution and USTF without convolution are unrolling networks employing scalar weights. The left figures corresponds to the result of the denoising task and the right to the interpolation task.**

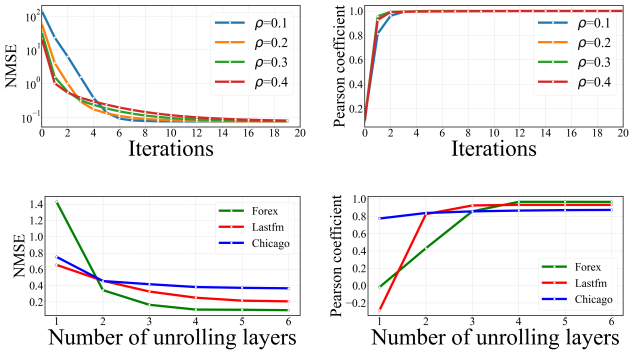


FIGURE 3. **Convergence property of multi-block ADMM and unrolling network. The figures on the top show the convergence performance of 4-block ADMM. As ρ changes from small to large, the ADMM algorithm is converging. The figures on the bottom are convergence performance of the simplicial unrolling network. We can observe that the network converges as it becomes deeper.**

APPENDIX

A. PROOF OF PROPOSITION 1

Before the proof, we recall this useful lemma from [24].

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 the augmented Lagrangian function satisfies the condition

$$0 < \rho < \min_{1\le i\le m} \left\{ \frac{2\mu_i}{3(m-1)\|\mathbf{A}_i\|_2^2} \right\} \tag{16}$$

where μ_i is the strongly convex modulus of each block [24], m is the number blocks and $\mathbf{A}_i$ is the coefficient matrix related to the ith equality constraint. Optimization problem (8) can be written as

$$\begin{aligned} \underset{\hat{\mathbf{f}}\in\mathbb{R}^{N_1}}{\operatorname{argmin}} & \|\mathbf{P}(\hat{\mathbf{f}}-\mathbf{y})\|_2^2 + \alpha_1\|\mathbf{z}_1\|_1 + \alpha_2\|\mathbf{z}_1\|_2^2 + \beta_1\|\mathbf{z}_2\|_1 \\ & + \beta_2\|\mathbf{z}_2\|_2^2 + \gamma_1\|\mathbf{z}_3\|_1 + \gamma_2\|\mathbf{z}_3\|_2^2 \\ \text{subject to} \quad & \mathbf{A}_1\hat{\mathbf{f}} + \mathbf{A}_2\mathbf{z}_1 + \mathbf{A}_3\mathbf{z}_2 + \mathbf{A}_4\mathbf{z}_3 = \mathbf{0} \end{aligned} \tag{17}$$

where the $\mathbf{A}_i$ are defined in (11a) - (11d). There are four different blocks for multi-block ADMM: $\|\mathbf{P}(\hat{\mathbf{f}}-\mathbf{y})\|_2^2$; $\alpha_1\|\mathbf{z}_1\|_1 + \alpha_2\|\mathbf{z}_1\|_2^2$; $\beta_1\|\mathbf{z}_2\|_1 + \beta_2\|\mathbf{z}_2\|_2^2$ and $\gamma_1\|\mathbf{z}_3\|_1 + \gamma_2\|\mathbf{z}_3\|_2^2$. Considering the condition $\mathbf{P}^\top\mathbf{P} \succeq \mu_1\mathbf{I}$, we have $\nabla^2\|\mathbf{P}(\hat{\mathbf{f}}-\mathbf{y})\|_2^2 = 2\mathbf{P}^\top\mathbf{P} \succeq 2\mu_1\mathbf{I}$. Therefore, the first block is a strongly convex function with modulu μ_1. The other three components are all strongly convex functions and their moduli are $\mu_2 = 2\alpha_2, \mu_3 = 2\beta_2, \mu_4 = 2\gamma_2$. This brings us under the setting of Lemma 1, which under the conditions in (16) completes the proof.

B. UNROLLING NETWORK FOR TREND FILTERING

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

$$\begin{cases} \hat{\mathbf{f}}^{(k+1)} = (2\mathbf{P}^\top\mathbf{P} + \rho\mathbf{B}_1^\top\mathbf{B}_1 + \rho\mathbf{B}_2\mathbf{B}_2^\top)^{-1} \\ \qquad (2\mathbf{P}^\top\mathbf{P}\mathbf{y} + \mathbf{B}_1^\top\boldsymbol{\lambda}_1^{(k)} + \mathbf{B}_2\boldsymbol{\lambda}_2^{(k)} \\ \qquad + \rho\mathbf{B}_1^\top\mathbf{z}_1^{(k)} + \rho\mathbf{B}_2\mathbf{z}_2^{(k)}) \\ \mathbf{z}_1^{(k+1)} = S_{\frac{\alpha_1}{\alpha\rho}}(\frac{1}{\rho}(\rho\mathbf{B}_1\hat{\mathbf{f}}^{(k+1)} - \boldsymbol{\lambda}_1^{(k)})) \\ \mathbf{z}_2^{(k+1)} = S_{\frac{\beta_1}{\alpha\rho}}(\frac{1}{\rho}(\rho\mathbf{B}_2^\top\hat{\mathbf{f}}^{(k+1)} - \boldsymbol{\lambda}_2^{(k)})) \\ \boldsymbol{\lambda}_1^{(k+1)} = \boldsymbol{\lambda}_1^{(k)} - \rho(\mathbf{B}_1\hat{\mathbf{f}}^{(k+1)} - \mathbf{z}_1^{(k+1)}) \\ \boldsymbol{\lambda}_2^{(k+1)} = \boldsymbol{\lambda}_2^{(k)} - \rho(\mathbf{B}_2^\top\hat{\mathbf{f}}^{(k+1)} - \mathbf{z}_2^{(k+1)}) \end{cases} \tag{18}$$

To construct a simplicial unrolling network for trend filtering (USTF), certain iterative parameters should also be substituted with trainable parameters. By substituting the fixed parameters with simplicial convolutional filters, the lth layer of USTF can be represented as follows

$$\begin{cases} \hat{\mathbf{f}}^{(l+1)} = \mathbf{H}_1\mathbf{P}^\top\mathbf{P}\mathbf{y} + \mathbf{H}_2\mathbf{B}_1^\top\boldsymbol{\lambda}_1^{(l)} + \mathbf{H}_3\mathbf{B}_2\boldsymbol{\lambda}_2^{(l)} \\ \qquad + \mathbf{H}_4\mathbf{B}_1^\top\mathbf{z}_1^{(l)} + \mathbf{H}_5\mathbf{B}_2\mathbf{z}_2^{(l)} \\ \mathbf{z}_1^{(l+1)} = S_{\frac{a_1}{r_1}}(\frac{1}{r_1}(r_1\mathbf{B}_1\hat{\mathbf{f}}^{(l)} - \boldsymbol{\lambda}_1^{(l)})) \\ \mathbf{z}_2^{(l+1)} = S_{\frac{b_1}{r_2}}(\frac{1}{r_2}(r_2\mathbf{B}_2^\top\hat{\mathbf{f}}^{(l)} - \boldsymbol{\lambda}_2^{(l)})) \\ \boldsymbol{\lambda}_1^{(l+1)} = \boldsymbol{\lambda}_1^{(l)} - r_1(\mathbf{B}_1\hat{\mathbf{f}}^{(l+1)} - \mathbf{z}_1^{(l+1)}) \\ \boldsymbol{\lambda}_2^{(l+1)} = \boldsymbol{\lambda}_2^{(l)} - r_2(\mathbf{B}_2^\top\hat{\mathbf{f}}^{(l+1)} - \mathbf{z}_2^{(l+1)}) \end{cases} \tag{19}$$

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

REFERENCES

[1] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D: nonlinear phenomena*, vol. 60, no. 1-4, pp. 259–268, 1992.

[2] R. G. Gavaskar and K. N. Chaudhury, "Regularization using denoising: Exact and robust signal recovery," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2022, pp. 5533–5537.

[3] S. Chen, A. Sandryhaila, J. M. Moura, and J. Kovačević, "Signal recovery on graphs: Variation minimization," *IEEE Transactions on Signal Processing*, vol. 63, no. 17, pp. 4609–4624, 2015.

[4] A. Ortega, P. Frossard, J. Kovačević, J. M. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 808–828, 2018.

[5] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs," *IEEE Signal Proc. Magazine*, 2013.

[6] M. Yang, M. Coutino, G. Leus, and E. Isufi, "Node-adaptive regularization for graph signal reconstruction," *IEEE Open Journal of Signal Processing*, vol. 2, pp. 85–98, 2021.

[7] R. Varma, H. Lee, J. Kovačević, and Y. Chi, "Vector-valued graph trend filtering with non-convex penalties," *IEEE transactions on signal and information processing over networks*, vol. 6, pp. 48–62, 2019.

[8] Y.-X. Wang, J. Sharpnack, A. Smola, and R. Tibshirani, "Trend filtering on graphs," in *Artificial Intelligence and Statistics*, PMLR, 2015, pp. 1042–1050.

[9] M. T. Schaub, Y. Zhu, J.-B. Seby, T. M. Roddenberry, and S. Segarra, "Signal processing on higher-order networks: Livin'on the edge... and beyond," *Signal Processing*, vol. 187, p. 108 149, 2021.

[10] S. Barbarossa and S. Sardellitti, "Topological signal processing over simplicial complexes," *IEEE Transactions on Signal Processing*, vol. 68, pp. 2992–3007, 2020.

[11] R. Money, J. Krishnan, B. Beferull-Lozano, and E. Isufi, "Online edge flow imputation on networks," *IEEE Signal Processing Letters*, vol. 30, pp. 115–119, 2022.

[12] M. T. Schaub and S. 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.

[13] J. Jia, M. T. Schaub, S. Segarra, and A. R. Benson, "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.

[14] D.-R. Han, "A survey on some recent developments of alternating direction method of multipliers," *Journal of the Operations Research Society of China*, pp. 1–52, 2022.

[15] J. Eckstein and W. Yao, "Understanding the convergence of the alternating direction method of multipliers: Theoretical and computational perspectives," *Pac. J. Optim.*, vol. 11, no. 4, pp. 619–644, 2015.

[16] V. Monga, Y. Li, and Y. C. Eldar, "Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing," *IEEE Signal Processing Magazine*, vol. 38, no. 2, pp. 18–44, 2021.

[17] O. Solomon, R. Cohen, Y. Zhang, *et al.*, "Deep unfolded robust PCA with application to clutter suppression in ultrasound," *IEEE transactions on medical imaging*, vol. 39, no. 4, pp. 1051–1063, 2019.

[18] L. Zhang, G. Wang, and G. B. Giannakis, "Real-time power system state estimation and forecasting via deep unrolled neural networks," *IEEE Transactions on Signal Processing*, vol. 67, no. 15, pp. 4069–4077, 2019.

[19] S. Lohit, D. Liu, H. Mansour, and P. T. Boufounos, "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.

[20] S. Chen, Y. C. Eldar, and L. Zhao, "Graph unrolling networks: Interpretable neural networks for graph signal denoising," *IEEE Transactions on Signal Processing*, vol. 69, pp. 3699–3713, 2021.

[21] M. Yang, E. Isufi, M. T. Schaub, and G. Leus, "Simplicial convolutional filters," *IEEE Transactions on Signal Processing*, vol. 70, pp. 4633–4648, 2022.

[22] C. De Mol, E. De Vito, and L. Rosasco, "Elastic-net regularization in learning theory," *Journal of Complexity*, vol. 25, no. 2, pp. 201–230, 2009.

[23] M. Yang and E. Isufi, "Simplicial trend filtering," in *2022 56th Asilomar Conference on Signals, Systems, and Computers*, IEEE, 2022, pp. 930–934.

[24] D. Han and X. Yuan, "A note on the alternating direction method of multipliers," *Journal of Optimization Theory and Applications*, vol. 155, no. 1, pp. 227–238, 2012.

[25] M. Riedmiller and A Lernen, "Multi layer perceptron," *Machine Learning Lab Special Lecture, University of Freiburg*, pp. 7–24, 2014.

[26] S. Ebli, M. Defferrard, and G. Spreemann, "Simplicial neural networks," *arXiv preprint arXiv:2010.03633*, 2020.

[27] M. Yang, E. Isufi, and G. Leus, "Simplicial convolutional neural networks," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2022, pp. 8847–8851.