Estimating Dynamic Graph Flows with Kernel Models and Hadamard-Structured Riemannian Constraints

Duc Thien Nguyen*, Konstantinos Slavakis*, Dimitris Pados†
* Institute of Science Tokyo, Japan
E-mails: nguyen.t.au@m.titech.ac.jp, slavakis@ict.eng.isct.ac.jp
† Florida Atlantic University, USA
E-mail: dpados@fau.edu

***Abstract*—This paper proposes a novel framework for missing-data imputation, coined kernel regression and Hadamard over-parameterization via Riemannian optimization (KROHMO), and applies it to the problem of estimating dynamic graph flows. The method captures latent feature geometries within a reproducing kernel Hilbert space (RKHS) and ensures parameter efficiency through matrix factorization and sparse coding, where the factor matrices are constrained to lie on Riemannian manifolds of fixed-rank matrices. Sparsity is induced through Hadamard overparameterization (HO), which enhances approximation capability while yielding a smooth objective function suitable for Riemannian optimization. To incorporate the underlying graph topology in time-varying edge flow estimation, the framework further exploits simplicial-complex theory and Hodge Laplacians. Numerical experiments on real-world dynamic network datasets show that KROHMO consistently outperforms several state-of-the-art methods in imputing missing edge flows.**

I. Introduction

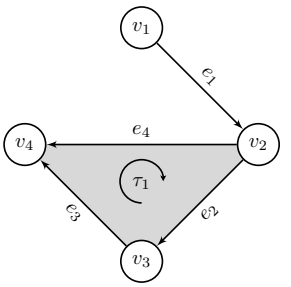
Graph signal processing [1], [2], [3] plays a central role in data analysis, as signals are frequently linked to entities that exhibit physical or abstract connections, typically represented using graphs. However, due to factors such as privacy concerns, sensor malfunctions, or resource conservation, signal observations are often incomplete (i.e., missing data), which can introduce bias and degrade performance in downstream learning tasks [4]. A considerable body of work has tackled the missing data problem when signals are observed at graph nodes; see e.g., [3], [5], [6]. Yet, in many real-world scenarios, signals are measured over the *edges* of a graph (i.e., edge flows), and research in this area has only recently begun to gain traction [7], [8], [9], [10].

Traditional node-signal imputation methods are not directly applicable to the edge-flow case. A common workaround is to construct the line graph of the original graph, where each node represents an edge from the original graph. This reformulates the task as a node-based imputation problem, typically tackled by enforcing spatial smoothness using a graph Laplacian. However, this strategy often yields suboptimal results, as the smoothness assumption does not hold well for edge flows [11], [12]. A more appropriate assumption is that flows are almost *divergence-free*, i.e., the inflow and outflow at each node are approximately balanced, or *curl-free*, where the net flow around any "triangle" is close to zero [8], [9], [11], [12], [13]. These properties are realized through the theories of simplicial complexes and the Hodge Laplacian [14]. These theories also introduce simplicial convolutional filters, defined as matrix polynomials of Hodge Laplacians, which model the multi-hop shift of signals across simplicial complexes [15], [16].

Although theories effectively model spatial structure through graph topology, time-varying edge flows are commonly analyzed using vector autoregression (VAR) [17]. Standard VAR models, however, disregard the underlying graph structure. To address this and simplify model complexity, recent studies have introduced simplicial convolutional filters into VAR, termed simplicial VAR (S-VAR), which leverage topological information while reducing the number of learnable parameters [18], [19]. Nonetheless, (S-)VAR depends on historical data for predictions, making it prone to missing data. Additionally and similarly to linear models, VAR-based methods may struggle to capture complex data patterns and dependencies. To tackle these issues, researchers have begun integrating simplicial complexes into neural networks [20], [21], [22]. However, these models often require large training datasets and involve multiple nonlinear layers. Furthermore, existing work on simplicial-complex neural networks seems to have overlooked the specific problem of imputing time-varying edge flows.

Time-varying edge flow imputation has also been addressed by matrix completion methods. Based on the fact that temporal dependency usually manifests itself in low-rank structures, MultiL-KRIM [10] proposes a kernel-based multilinear matrix factorization method, where the kernel matrix encodes the correlation among a set of observed data points. Moreover, MultiL-KRIM follows a collaborative-filtering approach via affine approximation and sparse coding. Such attributes help MultiL-KRIM outperform the multilayer matrix factorization (MMF) method [23], which decomposes a matrix into multiple factors, but does not use kernel methods and collaborative-filtering arguments. However, MultiL-KRIM minimizes a non-smooth loss function under affine constraints, thus has to

979-8-3315-7206-8/25/$31.00



$$\mathcal{X}^2 = \{\mathcal{S}^0, \mathcal{S}^1, \mathcal{S}^2\}$$

$$\mathcal{S}^0 = \{v_i\}_{i=1}^4, \mathcal{S}^1 = \{e_j\}_{j=1}^4, \mathcal{S}^2 = \{\tau_1\}$$

Incidence matrices:

$$\mathbf{B}_1 = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 1 & -1 & 0 & -1 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}, \mathbf{B}_2 = \begin{bmatrix} 0 \\ 1 \\ 1 \\ -1 \end{bmatrix}$$

Hodge Laplacians:

$$\mathbf{L}_0 = \mathbf{B}_1\mathbf{B}_1^\intercal, \mathbf{L}_1 = \mathbf{B}_1^\intercal\mathbf{B}_1 + \mathbf{B}_2\mathbf{B}_2^\intercal \quad (1)$$

Fig. 1: An example of an order-2 SC.

compute each individual matrix factor in an iterative routine.

This paper proposes a framework that employs kernel regression and Hadamard overparameterization via Riemannian optimization (KROHMO), and applies it to the problem of dynamic graph-edge flow imputation. KROHMO assumes that missing entries can be estimated by a set of landmark points, extracted from measurements and embedded in a reproducing kernel Hilbert space (RKHS). As such, functional approximation is enabled by the RKHS, effecting thus nonlinear data modeling, which is a lacking attribute in VAR models. Unlike neural networks, KROHMO needs no training data and offers a more explainable approach through kernel regression and sparse coding. While S-VAR applies simplicial convolutional filters for dimensionality reduction, KROHMO achieves this by matrix factorization, where the matrix factors belong to smooth manifolds of fixed-rank matrices. Additionally, while the number of hops in the simplicial convolutional filter captures "locality," KROHMO does so via the sparsity of its matrix factors. To promote a smooth optimization problem, sparsity is imposed via HO [24], [25], [26], [27]. Compared to its precursor MultiL-KRIM [10], KROHMO assigns explicit geometric structures for its parameter matrices, yielding a smooth optimization task that KROHMO solves via standard Riemannian-gradient-descent techniques, avoiding the iterative sub-tasks of MultiL-KRIM. Numerical tests on real networks show that KROHMO outperforms state-of-the-art methods, including its precursor MultiL-KRIM.

II. Preliminaries

A. Signals over simplicial complexes

Given a finite set of nodes $\mathcal{V}$, a k-simplex $\mathcal{S}^k \subset \mathcal{V}$ comprises $k+1$ distinct elements of $\mathcal{V}$. A simplicial complex (SC) $\mathcal{X}$ is a finite collection of simplices with the inclusion property: for any simplex $\mathcal{S}^k \in \mathcal{X}$, if $\mathcal{S}^{k-1} \subset \mathcal{S}^k$, then $\mathcal{S}^{k-1} \in \mathcal{X}$. An SC of order K, denoted as $\mathcal{X}^K$, contains at least one K-simplex [8], [14]. A graph $G = (\mathcal{V}, \mathcal{E})$ is an SC of order 1, where nodes are 0-simplices and edges in $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ are 1-simplices. If $\mathcal{T}$ is the set 2-simplices of $\mathcal{V}$ such that $(\mathcal{V}, \mathcal{E}, \mathcal{T})$ is an SC of order 2, then elements of $\mathcal{T}$ are called "triangles" of G. Let $N_0 = |\mathcal{V}|, N_1 = |\mathcal{E}|$, and $N_2 = |\mathcal{T}|$.

Hodge Laplacians [8], [14] describe adjacency in an SC. As the discussion focuses on $\mathcal{X}^2$, the incidence matrix $\mathbf{B}_1 \in \mathbb{R}^{N_0 \times N_1}$ captures the node-to-edge adjacency, and $\mathbf{B}_2 \in \mathbb{R}^{N_1 \times N_2}$ captures the edge-to-triangle adjacency. Incidence matrices depend on an arbitrary choice of simplices *orientation*. An edge e has an orientation as an ordered pair of nodes (i, j), and a triangle τ is oriented as (m, n, p). Then, the entries of the node-to-edge incidence matrix $\mathbf{B}_1$ are $[\mathbf{B}_1]_{i,e} = -[\mathbf{B}_1]_{j,e} = -1$ and $[\mathbf{B}_1]_{k,e} = 0$ for all $k \neq i, j$. The (e, τ)th entry of the edge-to-triangle incidence matrix $\mathbf{B}_2$ is 0 unless e is an edge of the triangle τ. Furthermore, $[\mathbf{B}_2]_{e,\tau} = 1$ if e has the same orientation as τ, otherwise $[\mathbf{B}_2]_{e,\tau} = -1$. An example of an order-2 SC is in Figure 1. The Hodge Laplacians $\mathbf{L}_0$ and $\mathbf{L}_1$ are defined in Eq. (1), where $\mathbf{L}_0$ is the well-known graph Laplacian matrix describing node-adjacency via shared edges, and $\mathbf{L}_1$ describes edge-adjacency via shared nodes and triangles.

"Simplicial signals" are abstracted as functions which map a k-simplex to a real number. In case of time-varying signals, an "edge-flow signal" at time $t \in \{1, 2, \ldots, T\}$ is denoted by $\mathbf{y}_t = [y_{1t}, y_{2t}, \ldots, y_{N_1 t}]^\intercal \in \mathbb{R}^{N_1}$. These vectors comprise the data matrix $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_T] \in \mathbb{R}^{N_1 \times T}$. To account for missing entries, let the index set of observed entries $\Omega := \{(i,t) \in \{1, \ldots, N_1\} \times \{1, \ldots, T\} \,|\, y_{it} \neq +\infty\}$, where $+\infty$ denotes a missing entry, and define the linear *sampling mapping* $\mathscr{S}_\Omega \colon (\mathbb{R} \cup \{+\infty\})^{N_1 \times T} \to \mathbb{R}^{N_1 \times T} \colon \mathbf{Y} \mapsto \mathscr{S}_\Omega(\mathbf{Y})$, which operates entry-wisely as follows: $[\mathscr{S}_\Omega(\mathbf{Y})]_{it} := [\mathbf{Y}]_{it}$, if $(i,t) \in \Omega$, while $[\mathscr{S}_\Omega(\mathbf{Y})]_{it} := 0$, if $(i,t) \notin \Omega$.

B. Prior art

An edge-flow imputation framework solves the following inverse problem

$$\min_{\mathbf{X} \in \mathbb{R}^{N_1 \times T}} \mathcal{L}(\mathbf{X}) + \mathcal{R}(\mathbf{X}) \quad \text{s.to } \mathscr{S}_\Omega(\mathbf{Y}) = \mathscr{S}_\Omega(\mathbf{X}) \text{ and other constraints}, \quad (2)$$

where $\mathcal{L}(\cdot)$ is the data-fit loss and $\mathcal{R}(\cdot)$ is the regularizer for structural priors, while constraint $\mathscr{S}_\Omega(\mathbf{Y}) = \mathscr{S}_\Omega(\mathbf{X})$ preserves the consistency of the observed entries of $\mathbf{Y}$. For example, FlowSSL [12] sets $\mathcal{L}(\mathbf{X}) := 0$ and $\mathcal{R}(\mathbf{X}) := (\lambda_l/2)\|\mathbf{B}_1\mathbf{X}\|_{\mathrm{F}}^2 + (\lambda_u/2)\|\mathbf{B}_2^\intercal\mathbf{X}\|_{\mathrm{F}}^2$, which imposes the conservation of the flows at the nodes and cyclic flows along edges of triangles. To work with time-varying signals, [28], [29] consider the product of two complexes representing time and space, where time is seen as a linear/path graph, thus can incorporate also temporal smoothness. Recently, S-VAR [18], [19] combines the Hodge Laplacians and VAR to approximate each snapshot $\mathbf{y}_t \approx \sum_{p=1}^P \mathbf{H}_p(\mathbf{L}_1)\mathbf{y}_{t-p}$, where $\{\mathbf{H}_p(\mathbf{L}_1)\}_{p=1}^P$ are simplicial convolutional filters [16], whose role is to capture the multi-hop dependencies between edges. As a result, S-VAR sets $\mathcal{L}(\mathbf{X}) := \sum_{t=1}^T \|\mathbf{x}_t - \mathbf{F}_t\boldsymbol{\beta}_t\|_2^2$, where $\{\boldsymbol{\beta}_t\}_{t=1}^T$ are learnable parameters, and $\{\mathbf{F}_t\}_{t=1}^T$ are "shifted signals" of $\{\mathbf{y}_t\}_{t=1}^T$.

C. Multilinear kernel regression and imputation

MultiL-KRIM, a generic data-imputation framework built on matrix factorization, kernel methods, and manifold learning, has been utilized for node-signal imputation and dynamic imaging reconstruction [3], as well as edge-flow imputation [10].

A subset of the observed data $\mathscr{S}_\Omega(\mathbf{Y})$, called *navigator/pilot* data and denoted as $\check{\mathbf{Y}}_{\rm f} \coloneqq [\check{\mathbf{y}}^{\rm f}_1, \ldots, \check{\mathbf{y}}^{\rm f}_{N_{\rm nav}}] \in \mathbb{R}^{\nu \times N_{\rm nav}}$, are selected by the user; for example, if all of $\mathscr{S}_\Omega(\mathbf{Y})$ is selected, i.e., $\check{\mathbf{Y}}_{\rm f} \coloneqq \mathscr{S}_\Omega(\mathbf{Y})$, then $\nu \coloneqq N_1$ and $N_{\rm nav} \coloneqq T$. As the cardinality of the point-cloud $\{\check{\mathbf{y}}^{\rm f}_t\}_{t=1}^{N_{\rm nav}}$ grows for large datasets, a subset $\{\boldsymbol{l}_k\}_{k=1}^{N_l}$, coined *landmark/representative* points, with $N_l \leq N_{\rm nav}$, is selected from $\{\check{\mathbf{y}}^{\rm f}_t\}_{t=1}^{N_{\rm nav}}$. Let $\check{\mathbf{L}} \coloneqq [\boldsymbol{l}_1, \boldsymbol{l}_2, \ldots, \boldsymbol{l}_{N_l}] \in \mathbb{R}^{\nu \times N_l}$.

To enable nonlinear approximation, a feature map $\varphi(\cdot)$ transforms the vector $\boldsymbol{l}_k$ into $\varphi(\boldsymbol{l}_k)$ within an RKHS $\mathscr{H}$, which is equipped with a reproducing kernel $\kappa(\cdot,\cdot) : \mathbb{R}^\nu \times \mathbb{R}^\nu \to \mathbb{R}$, known for its rich properties in approximation theory [30]. Accordingly, $\varphi(\boldsymbol{l}) \coloneqq \kappa(\boldsymbol{l}, \cdot) \in \mathscr{H}$ for all $\boldsymbol{l} \in \mathbb{R}^\nu$. For simplicity, define $\boldsymbol{\Phi}(\check{\mathbf{L}}) \coloneqq [\varphi(\boldsymbol{l}_1), \ldots, \varphi(\boldsymbol{l}_{N_l})]$. Thus, the kernel matrix $\mathbf{K} \coloneqq \boldsymbol{\Phi}(\check{\mathbf{L}})^\intercal \boldsymbol{\Phi}(\check{\mathbf{L}})$ is an $N_l \times N_l$ matrix, with its (k,k')th entry given by $\langle \varphi(\boldsymbol{l}_k) \mid \varphi(\boldsymbol{l}_{k'}) \rangle_{\mathscr{H}} = \kappa(\boldsymbol{l}_k, \boldsymbol{l}_{k'})$, where $\intercal$ denotes vector/matrix transposition and $\langle \cdot \mid \cdot \rangle_{\mathscr{H}}$ represents the inner product in the RKHS $\mathscr{H}$.

The flow x_{it} through the edge i at time t is estimated as

$$[\mathbf{X}]_{it} = x_{it} \approx f_i(\check{\boldsymbol{\mu}}_t) = \langle f_i \mid \varphi(\check{\boldsymbol{\mu}}_t) \rangle_{\mathscr{H}}\,, \tag{3}$$

where $f_i(\cdot) : \mathbb{R}^\nu \to \mathbb{R}$ is an unknown non-linear function in the RKHS functional space $\mathscr{H}$, $\check{\boldsymbol{\mu}}_t$ is an unknown vector in $\mathbb{R}^\nu$, and the last equality of (3) is the reproducing property of $\mathscr{H}$ [30]. Motivated by the representer theorem [31], f_i is assumed to belong to the linear span of $\{\varphi(\boldsymbol{l}_k)\}_{k=1}^{N_l}$, i.e., there exists $\mathbf{u}_i \coloneqq [u_{i1}, \ldots, u_{iN_l}]^\intercal \in \mathbb{R}^{N_l}$ s.t. $f_i = \sum_{k=1}^{N_l} u_{ik} \varphi(\boldsymbol{l}_k) = \boldsymbol{\Phi}(\check{\mathbf{L}})\mathbf{u}_i$. Similarly, it is assumed that there exists $\mathbf{v}_t \in \mathbb{R}^{N_l}$ s.t. $\varphi(\check{\boldsymbol{\mu}}_t) = \boldsymbol{\Phi}(\check{\mathbf{L}})\mathbf{v}_t$.

So far, the entry-wise estimation $[\mathbf{X}]_{it} \approx f_i(\check{\boldsymbol{\mu}}_t) = \langle f_i \mid \varphi(\check{\boldsymbol{\mu}}_t) \rangle_{\mathscr{H}} = \langle \boldsymbol{\Phi}(\check{\mathbf{L}})\mathbf{u}_i \mid \boldsymbol{\Phi}(\check{\mathbf{L}})\mathbf{v}_t \rangle_{\mathscr{H}} = \mathbf{u}_i^\intercal \mathbf{K} \mathbf{v}_t$. To offer compact notations, if $\mathbf{U} \coloneqq [\mathbf{u}_1, \ldots, \mathbf{u}_{N_1}]^\intercal \in \mathbb{R}^{N_1 \times N_l}$ and $\mathbf{V} \coloneqq [\mathbf{v}_1, \ldots, \mathbf{v}_T] \in \mathbb{R}^{N_l \times T}$, then data are modeled as

$$\mathbf{X} \approx \mathbf{U}\mathbf{K}\mathbf{V}\,. \tag{4}$$

For dimensionality reduction, MultiL-KRIM [10] uses low-rank matrix factorization, i.e., $\mathbf{U} = \mathbf{U}_1\mathbf{U}_2$ and $\mathbf{V} = \mathbf{V}_1\mathbf{V}_2$. Inspired by the concept of tangent spaces to smooth manifolds, each column $\mathbf{v}_t$ of $\mathbf{V}$ is a sparse vector satisfying the sum-to-one constraint. Ultimately, the inverse problem (2) of MultiL-KRIM is non-convex, minimizing a composite loss function due to ℓ_1-norm loss for sparsity, subjected to affine constraints. The inverse problem is solved by a successive-convex-approximation algorithm, wherein some sub-tasks need to be solved iteratively.

III. Proposed method

Different from existing approaches, KROHMO makes the following assumptions.

Modeling Assumption 1. *Matrices* $\mathbf{U}$ *and* $\mathbf{V}$ *in Eq.* (4) *are sparse. Moreover,* $\mathbf{U} = \mathbf{U}_1 \odot \mathbf{U}_2$ *and* $\mathbf{V} = \mathbf{V}_1 \odot \mathbf{V}_2$*, where* $\odot$ *stands for the Hadamard product.*

KROHMO imposes sparsity on $\mathbf{U}$ and $\mathbf{V}$, which will retain a small number of effective parameters, thus increasing efficiency and robustness. Via Modeling Assumption 1, the sparsity of $\mathbf{U}$ and $\mathbf{V}$ can be activated by penalizing the Frobenius norm of $\mathbf{U}_1, \mathbf{U}_2, \mathbf{V}_1$, and $\mathbf{V}_2$, which are smooth terms (C^∞) so that solving KROHMO's optimization problem is handy. Specifically, the regularizer $2\|\mathbf{U}\|_1$ is induced by the surrogate regularizer $\|\mathbf{U}_1\|_{\rm F}^2 + \|\mathbf{U}_2\|_{\rm F}^2$ (similarly holds for $\mathbf{V}$). In general, for a positive integer k and a HO $\mathbf{Z} = \odot_{i=1}^k \mathbf{Z}_i$, penalizing $\sum_{i=1}^k \|\mathbf{Z}_i\|_{\rm F}^2$ induces the $\ell_{2/k}$-norm regularizer $k\|\mathbf{Z}\|_{2/k}^{2/k}$, which is nonconvex when $k > 2$, but permits some desirable theoretical results and yields sparser solutions than the convex ℓ_1 norm [27]. Theorems in [27] suggest that the Frobenius-norm surrogate affects the loss surface to induce sparsity. Results in [24], [25], [26], [27] also support the idea that HO increases the number of parameters, potentially improving the approximation quality. Indeed, in Section IV-C, numerical experiments show that a considerable number of parameters can be truncated without impairing much the imputation performance.

Modeling Assumption 2. $\mathbf{U}_1, \mathbf{U}_2, \mathbf{V}_1$*, and* $\mathbf{V}_2$ *belong to sets of fixed-rank matrices, which are Riemannian manifolds [32].*

Instead of using low-rank matrix factorization like MultiL-KRIM [10], KROHMO assumes that its matrix factors belong to smooth manifolds of fixed-rank matrices [32]. While MultiL-KRIM borrows manifold arguments to approximate data points, KROHMO assigns explicit geometric structures for its parameter matrices. Since the objective function is smooth, KROHMO can leverage Riemannian gradient descent techniques to solve its optimization problem, avoiding iterative sub-tasks as in MultiL-KRIM.

A. Manifolds of fixed-rank matrices

Consider the set of all $m \times n$ matrices of rank k, i.e.,

$$\mathcal{M}_{m,n,k} \coloneqq \{\mathbf{X} \in \mathbb{R}^{m \times n} \,|\, \text{rank}(\mathbf{X}) = k\}\,.$$

It is well known that $\mathcal{M}_{m,n,k}$ is a Riemannian manifold of dimension $(m+n-k)k$ embedded in $\mathbb{R}^{m \times n}$ [32]. Singular value decomposition (SVD) allows an equivalent characterization

$$\mathcal{M}_{m,n,k} = \{\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\intercal \,|\, \mathbf{U} \in \text{St}_k^m, \mathbf{V} \in \text{St}_k^n, \boldsymbol{\Sigma} = \text{diag}(\sigma_i)\}\,,$$

where St_k^m is the set of $m \times k$ real, orthonormal matrices, and $\text{diag}(\sigma_i)$ denotes a diagonal matrix with positive singular values on the diagonal.

Denote as $T_{\mathbf{X}}\mathcal{M}_{m,n,k}$ the tangent space of $\mathcal{M}_{m,n,k}$ at $\mathbf{X}$, the Riemannian metric is given as $\langle \boldsymbol{\xi} \mid \boldsymbol{\eta} \rangle_{\mathbf{X}} = \text{tr}(\boldsymbol{\xi}^\intercal \boldsymbol{\eta})$, with $\mathbf{X} \in \mathcal{M}_{m,n,k}$ and the tangent vectors $\boldsymbol{\xi}, \boldsymbol{\eta} \in T_{\mathbf{X}}\mathcal{M}_{m,n,k}$ are seen as matrices in $\mathbb{R}^{m \times n}$. Since $\mathcal{M}_{m,n,k}$ is embedded in $\mathbb{R}^{m \times n}$, the Riemannian gradient of a smooth objective function $f : \mathcal{M}_{m,n,k} \to \mathbb{R}$ is given as the orthogonal projection of the (Euclidean) gradient onto the tangent space [33], i.e.,

$$\text{grad}\, f(\mathbf{X}) \coloneqq \text{P}_{T_{\mathbf{X}}\mathcal{M}_{m,n,k}}(\nabla_{\mathbf{X}} f(\mathbf{X}))\,.$$

In contrast, a retraction maps a vector $\boldsymbol{\xi}$ in the tangent space at $\mathbf{X}$ back to the manifold. If defined based on metric projection, the retraction $R_{\mathbf{X}}(\boldsymbol{\xi})$ in this case can be computed by the SVD. Details such as the parameterization of the tangent space, the projection onto the tangent space, and the calculation of the retraction can be found in [32].

B. Inverse problem

Based on Modeling Assumptions 1 and 2 as well as the priors for edge flows, the following inverse problem is obtained.

$$\begin{aligned}\min_{\boldsymbol{\mathcal{O}}\in\mathscr{M}}\ & f(\boldsymbol{\mathcal{O}}) \coloneqq \tfrac{1}{2}\|\mathscr{S}_\Omega(\mathbf{X}-\mathbf{Y})\|_{\mathrm{F}}^2 \\ & + \tfrac{\lambda_l}{2}\|\mathbf{B}_1\mathbf{X}\|_{\mathrm{F}}^2 + \tfrac{\lambda_u}{2}\|\mathbf{B}_2^\intercal\mathbf{X}\|_{\mathrm{F}}^2 \\ & + \tfrac{\lambda_1}{2}\sum_{q=1}^{2}\|\mathbf{U}_q\|_{\mathrm{F}}^2 + \tfrac{\lambda_2}{2}\sum_{p=1}^{2}\|\mathbf{V}_p\|_{\mathrm{F}}^2 \qquad (5a)\\ \text{where } & \mathbf{X} = (\mathbf{U}_1\odot\mathbf{U}_2)\mathbf{K}(\mathbf{V}_1\odot\mathbf{V}_2)\,, \qquad (5b)\\ & \boldsymbol{\mathcal{O}} = (\mathbf{U}_1,\mathbf{U}_2,\mathbf{V}_1,\mathbf{V}_2)\,,\end{aligned}$$

and $\mathscr{M}$ is the Riemannian manifold $\mathcal{M}_{N,N_l,r_1}\times\mathcal{M}_{N,N_l,r_1}\times\mathcal{M}_{N_l,T,r_2}\times\mathcal{M}_{N_l,T,r_2}$ [33]. Note that the hard data-consistency constraint $\mathscr{S}_\Omega(\mathbf{X}) = \mathscr{S}_\Omega(\mathbf{Y})$ is relaxed and takes the form of a quadratic loss term. The term $\frac{\lambda_l}{2}\|\mathbf{B}_1\mathbf{X}\|_{\mathrm{F}}^2 + \frac{\lambda_u}{2}\|\mathbf{B}_2^\intercal\mathbf{X}\|_{\mathrm{F}}^2$ realizes the common priors that edge flows are approximately *divergence-free* and *curl-free* (cf., Section II-B). The loss function (5a) is smooth, which facilitates Riemannian optimization [33]. Algorithm 1 shows the pseudocode of the proposed framework. The Riemannian gradient $\operatorname{grad} f(\hat{\boldsymbol{\mathcal{O}}}^{(n)})$ in step 4 consists of partial Riemannian gradients $\big(\operatorname{grad}_{\mathbf{U}_1} f, \operatorname{grad}_{\mathbf{U}_2} f, \operatorname{grad}_{\mathbf{V}_1} f, \operatorname{grad}_{\mathbf{V}_2} f\big)(\hat{\boldsymbol{\mathcal{O}}}^{(n)})$, which can be computed following [34]. The retraction map in step 7 can be found in [32].

Algorithm 1 Solve (5) using Riemannian gradient descent

Output: Limit point $\hat{\boldsymbol{\mathcal{O}}}^{(*)}$ of the sequence $(\hat{\boldsymbol{\mathcal{O}}}^{(n)})_{n\in\mathbb{N}}$.

1: Initialize $\hat{\boldsymbol{\mathcal{O}}}^{(0)}\in\mathcal{M}$.
2: **while** $n\geq 0$ **do**
3: Available are $\hat{\boldsymbol{\mathcal{O}}}^{(n)}$.
4: Get the Riemannian gradient $\boldsymbol{\xi}_n \coloneqq \operatorname{grad} f(\hat{\boldsymbol{\mathcal{O}}}^{(n)})$.
5: Set the descent direction as $\boldsymbol{\eta}_n = -\operatorname{grad} f(\hat{\boldsymbol{\mathcal{O}}}^{(n)})$.
6: Find the smallest integer m s.t.

$$f(\hat{\boldsymbol{\mathcal{O}}}^{(n)}) - f(R_{\hat{\boldsymbol{\mathcal{O}}}^{(n)}}(\alpha\beta^m\boldsymbol{\eta}_n)) \geq -\gamma\langle\boldsymbol{\xi}_n \mid \alpha\beta^m\boldsymbol{\eta}_n\rangle\,.$$

7: Update by retraction $\hat{\boldsymbol{\mathcal{O}}}^{(n+1)} = R_{\hat{\boldsymbol{\mathcal{O}}}^{(n)}}(\alpha\beta^m\boldsymbol{\eta}_n)$.
8: Set $n\leftarrow n+1$ and go to step 2.
9: **end while**

IV. Numerical Tests

KROHMO is tested on traffic flows in the Sioux Falls transportation network [35] and the Eastern Massachusetts (EMA) network [36], and on water flows in the Cherry Hills water network [37]. KROHMO is compared against the state-of-the-art edge-flow imputation methods FlowSSL [12], PS [28], and S-VAR [18], [19]. Note that, besides PS [28], the other imputation method in product space [29] requires inversion of square matrices of size $(N_1T)\times(N_1T)$, which could not be implemented by the CPU setting (memory) in this study. KROHMO is also compared against matrix factorization methods such as MMF [23] and MultiL-KRIM [10].

The evaluation metric is the mean absolute error (MAE) (lower is better), defined as $\text{MAE} \coloneqq \|\hat{\mathbf{X}}-\mathbf{Y}\|_1/(N_1\cdot T)$, where $\mathbf{Y}$ is the fully-sampled data, $\hat{\mathbf{X}}$ gathers all reconstructed flows, and $\|\cdot\|_1$ is the ℓ_1-norm. All methods are finely tuned to achieve their lowest MAE. Reported metric values are mean values of 30 independent runs. Source codes of FlowSSL [12] and S-VAR [18], [19] were made publicly available by the authors. Source code for KROHMO, MultiL-KRIM and MMF was written in Python. All tests were run on an 8-core Intel(R) i7-11700 2.50GHz CPU with 32GB RAM.

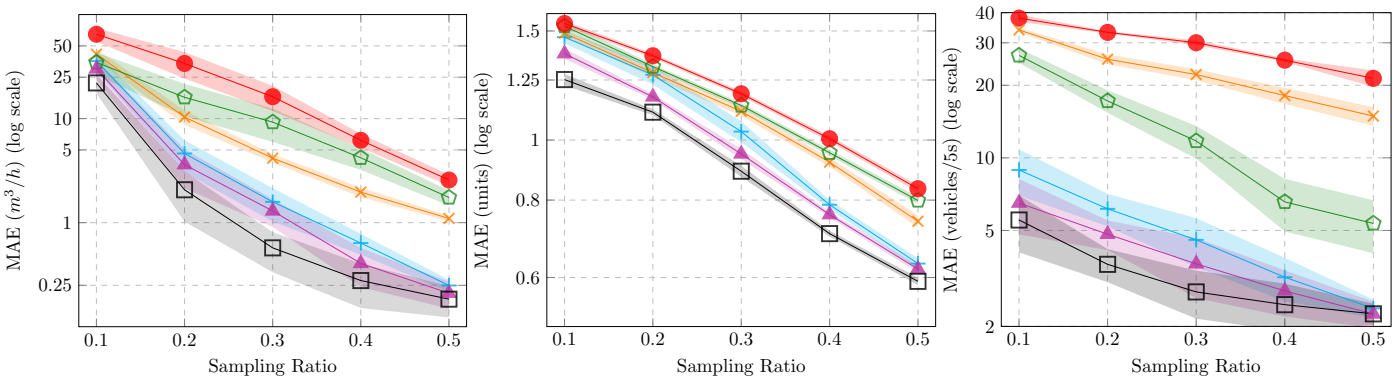
With sampling ratio $s\in\{0.1,0.2,0.3,0.4,0.5\}$, signals of $\lceil N_1\cdot s\rceil$ edges are sampled per time instant t, where $\lceil\cdot\rceil$ is the ceiling function. This sampling pattern suggests that the number of observations is consistent over time. Navigator data $\check{\mathbf{Y}}_{\text{nav}}$ are formed by snapshots, that is, columns of $\mathscr{S}_\Omega(\mathbf{Y})$. Landmark points are selected by the greedy max-min-distance strategy [38], based on Euclidean distances among navigator data. Several kernel functions κ are tested, including: **(i)** the Gaussian kernel $\kappa_{\mathrm{G};\sigma}(\boldsymbol{l},\boldsymbol{l}') \coloneqq \exp[-\sigma\|\boldsymbol{l}-\boldsymbol{l}'\|_{\mathrm{F}}^2]$, where $\sigma\in\mathbb{R}_{>0}$; **(ii)** the polynomial kernel $\kappa_{\mathrm{P};(c,r)}(\boldsymbol{l},\boldsymbol{l}') \coloneqq (\boldsymbol{l}^\intercal\boldsymbol{l}'+c)^r$, where $r\in\mathbb{N}_*$, $c\in\mathbb{R}$; and, **(iii)** the Matern kernel [39] with the smoothness parameter $\nu\in\{0.5,1.5\}$. Meanwhile, regularization hyperparameters $\lambda_1,\lambda_2,\lambda_l$, and λ_u in (5a), the number of landmark points N_l, and the ranks (r_1,r_2) are identified by grid-search. The constants for steps 6 and 7 in Algorithm 1 are $\alpha>0,\beta\in(0,1),\gamma\in(0,1)$.

A. Cherry Hills water network

Cherry Hills water network consists of 36 nodes (0-simplices), 40 pipes (1-simplices), and 2 triangles (2-simplices) [37]. Time-varying water flow over the network is generated using the EPANET software [40]. Flows are measured hourly in cubic meter per hour (m^3/h) in a span of 300 hours. The size of data $\mathbf{Y}$ is 40×300, and the number of landmark points is $N_l = 100$. Figure 2a shows MAE values of competing methods across different sampling ratios. KROHMO outperforms the competing methods at every sampling ratio. It significantly outperforms S-VAR [18], [19], especially at higher sampling ratios. FlowSSL [12], which solely depends on the Hodge Laplacians-based regularizers, scores much higher MAE values than the other methods. KROHMO outperforms also the blind matrix-factorization MMF [23] as well as the kernel- and manifold-utilizing MultiL-KRIM [10].

B. Sioux Falls transportation network

The Sioux Falls transportation network has 24 nodes (0-simplices), 38 edges (1-simplices), and 2 triangles (2-simplices) [35]. The synthetic time-varying flow is generated as in [18], [19], so that it contains both divergence flows and cyclic flows. The measurement unit is neglected, simply denoted as "units." The size of data $\mathbf{Y}$ is 38×300, while $N_l = 50$. Figure 2b plots MAE curves against sampling ratios. KROHMO continues to outperform the competing methods, with notable gaps when compared with S-VAR [18], [19] and FlowSSL [12]. Meanwhile, the gaps between KROHMO, MultiL-KRIM [10], and MMF [23] accentuate at lower sampling ratios. FlowSSL [12] scores again the highest MAE values.



(a) Cherry Hills water network (b) Sioux Falls traffic network (c) Eastern Massachusetts traffic network

Fig. 2: Mean MAE value curves (the lower the better) w.r.t. the groud truth signals vs. sampling ratios. The shaded areas indicate a range of one standard deviation above and below the mean. Curve markers: MMF [23]: +, S-VAR [18], [19]: ×, PS [28]: ⬠, FlowSSL [12]: ●, MultiL-KRIM [10]: ▲, KROHMO: □.

C. Eastern Massachusetts transportation network

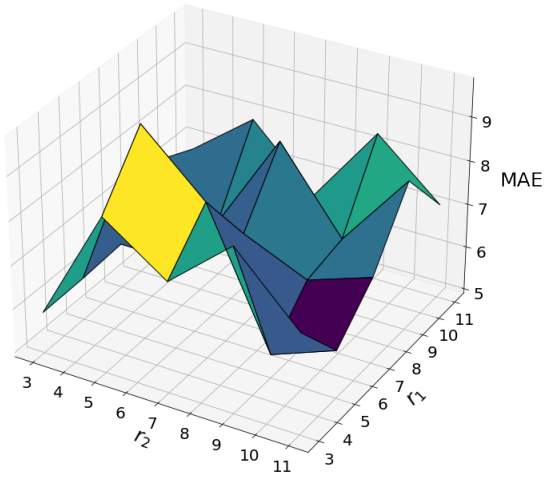


Fig. 3: Sensitivity to variation of parameters r_1 and r_2 for EMA network data at 10% sampling ratio.

TABLE I: Sensitivity to different kernels in the context: EMA network, 10% sampling ratio, $(N_l, r_1, r_2) = (50, 5, 9)$.

Index	Kernel function	Parameters	MAE
1	Gaussian	$\sigma = 1$	6.4578
2	Gaussian	$\sigma = 2$	6.9269
3	Gaussian	$\sigma = 3$	6.2031
4	Gaussian	$\sigma = 4$	**5.5470**
5	Polynomial	$r = 1$	7.8489
6	Polynomial	$r = 2$	8.1236
7	Polynomial	$r = 3$	7.9350
8	Polynomial	$r = 4$	7.1160
9	Matern	$\nu = 0.5$	6.0837
10	Matern	$\nu = 1.5$	8.2185

The Eastern Massachusetts (EMA) network consists of 74 nodes, 258 edges, and 33 triangles [36]. Time-varying edge signals are generated by a specialized traffic flow simulator [41]. The simulation was run over 300 time points with 5-second resolution. Two edges with no recorded flow are removed, thus the size of data $\mathbf{Y}$ is 256×300. The number of landmark points is $N_l = 50$. In this larger and more complex network, Figure 2c shows that KROHMO continues to outperform the state-of-the-art methods.

Table I shows the MAE values when using different kernel functions. On average, Gaussian kernels give the lowest MAE value, achieved at $\sigma = 4$. Meanwhile, Figure 3 draws the sensitivity at different settings of the ranks (r_1, r_2). The lowest MAE is achieved when $(r_1, r_2) = (5, 9)$.

TABLE II: Effect of sparsity when testing on the EMA network.

s	Sparsity of $\hat{\mathbf{U}}^{(*)}$	Sparsity of $\hat{\mathbf{V}}^{(*)}$	%MAE change
0.1	70.47%	30.15%	+0.7
0.2	59.76%	29.47%	+1.1
0.3	15.65%	16.78%	+1.6
0.4	13.01%	13.99%	+1.7
0.5	16.02%	16.74%	+1.5

The output matrices $\hat{\mathbf{U}}^{(*)} := \hat{\mathbf{U}}_1^{(*)} \odot \hat{\mathbf{U}}_2^{(*)}$ and $\hat{\mathbf{V}}^{(*)} := \hat{\mathbf{V}}_1^{(*)} \odot \hat{\mathbf{V}}_2^{(*)}$ are approximately sparse. Because $\mathbf{K}$ is fixed, it can be rescaled so that the outputs $\hat{\mathbf{U}}^{(*)}$ and $\hat{\mathbf{V}}^{(*)}$ have entries in the range $[-1, 1]$. Table II shows the parameter efficiency by using sparsity. The sparsity measure of each matrix is the portion of entries whose absolute value is less than 10^{-3}, while the last column of Table II shows the percentage of change of MAE after zeroing out those entries. It can be seen that the errors increase by only less than 2% even when removing many small coefficients. Particularly, when the sampling ratio is low, i.e., $s \in \{0.1, 0.2\}$, roughly 30% of $\hat{\mathbf{U}}^{(*)}$ and 60% to 70% of $\hat{\mathbf{V}}^{(*)}$ can be saved while losing only around 1% of performance.

V. Conclusion

This paper proposed a kernel-based imputation-by-regression framework via Riemannian optimization (KROHMO) and applied it to the problem of imputing time-varying edge flows. KROHMO approximated the data using kernels and sparse

coding, with matrix factors constrained to lie on fixed-rank matrix manifolds. To enable smooth and tractable optimization, HO was employed for sparsity regularization, thereby making Riemannian optimization applicable. To incorporate common structural priors of edge flows, the inverse problem formulation of KROHMO also included the graph's Hodge Laplacians. Numerical tests on real-world water and traffic networks demonstrated that KROHMO outperformed several state-of-the-art approaches in edge-flow imputation.

ACKNOWLEDGEMENT

The work of D. T. Nguyen was supported by JST SPRING, Japan Grant Number JPMJSP2180. The work of D. A. Pados was supported by the US Air Force Office of Scientific Research under Grant W911NF-20-1-028.

REFERENCES

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

[2] G. Leus, A. G. Marques, J. M. Moura, et al., "Graph signal processing: History, development, impact, and outlook," *IEEE Signal Processing Magazine*, vol. 40, no. 4, pp. 49–60, 2023.

[3] D. T. Nguyen and K. Slavakis, "Multilinear kernel regression and imputation via manifold learning," *IEEE Open Journal of Signal Processing*, pp. 1–15, 2024.

[4] S. Chen, A. Sandryhaila, J. M. Moura, et al., "Signal recovery on graphs: Variation minimization," *IEEE Trans. Signal Processing*, vol. 63, no. 17, pp. 4609–4624, 2015.

[5] K. Qiu, X. Mao, X. Shen, et al., "Time-varying graph signal reconstruction," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 6, pp. 870–883, 2017.

[6] D. Romero, V. N. Ioannidis, and G. B. Giannakis, "Kernel-based reconstruction of space-time functions on dynamic graphs," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 6, pp. 856–869, 2017.

[7] F. Battiston, G. Cencetti, I. Iacopini, et al., "Networks beyond pairwise interactions: Structure and dynamics," *Physics Reports*, vol. 874, pp. 1–92, 2020.

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

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

[10] D. T. Nguyen, K. Slavakis, and D. Pados, "Imputation of time-varying edge flows in graphs by multilinear kernel regression and manifold learning," *Signal Processing*, p. 110 077, 2025.

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

[12] J. Jia, M. T. Schaub, S. Segarra, et al., "Graph-based semi-supervised & active learning for edge flows," in *SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 761–771.

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

[14] L.-H. Lim, "Hodge Laplacians on graphs," *SIAM Review*, vol. 62, no. 3, pp. 685–715, 2020.

[15] E. Isufi and M. Yang, "Convolutional filtering in simplicial complexes," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 5578–5582.

[16] M. Yang, E. Isufi, M. T. Schaub, et al., "Simplicial convolutional filters," *IEEE Trans. Signal Processing*, vol. 70, pp. 4633–4648, 2022.

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

[18] J. Krishnan, R. Money, B. Beferull-Lozano, et al., "Simplicial vector autoregressive models," *IEEE Transactions on Signal Processing*, vol. 72, pp. 5454–5469, 2024.

[19] R. Money, J. Krishnan, B. Beferull-Lozano, et al., "Evolution backcasting of edge flows from partial observations using simplicial vector autoregressive models," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2024, pp. 9516–9520.

[20] T. M. Roddenberry and S. Segarra, "HodgeNet: Graph neural networks for edge data," in *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, 2019, pp. 220–224.

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

[22] H. Wu, A. Yip, J. Long, et al., "Simplicial complex neural networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 1, pp. 561–575, 2024.

[23] A. Cichocki and R. Zdunek, "Multilayer nonnegative matrix factorization using projected gradient approaches," *International Journal of Neural Systems*, vol. 17, no. 06, pp. 431–446, 2007.

[24] P. D. Hoff, "Lasso, fractional norm and structured sparse estimation using a Hadamard product parametrization," *Computational Statistics & Data Analysis*, vol. 115, pp. 186–198, 2017.

[25] G. Li, S. Li, D. Li, et al., "The tail-Hadamard product parametrization algorithm for compressed sensing," *Signal Processing*, vol. 205, p. 108 853, 2023.

[26] L. Ziyin and Z. Wang, "Spred: Solving L_1 penalty with SGD," in *International Conference on Machine Learning*, PMLR, 2023, pp. 43 407–43 422.

[27] C. Kolb, C. L. Müller, B. Bischl, et al., "Smoothing the edges: Smooth optimization for sparse regularization using Hadamard overparametrization," *arXiv preprint arXiv:2307.03571*, 2023.

[28] T. M. Roddenberry, V. P. Grande, F. Frantzen, et al., "Signal processing on product spaces," in *ICASSP 2023*, IEEE, 2023, pp. 1–5.

[29] T. S. Reddy and S. P. Chepuri, "Sampling and recovery of signals over product cell structures," in *ICASSP 2024*, IEEE, 2024, pp. 13 191–13 195.

[30] N. Aronszajn, "Theory of reproducing kernels," *Trans. American Mathematical Society*, vol. 68, no. 3, pp. 337–404, 1950.

[31] G. Kimeldorf and G. Wahba, "Some results on Tchebycheffian spline functions," *Journal of mathematical analysis and applications*, vol. 33, no. 1, pp. 82–95, 1971.

[32] B. Vandereycken, "Low-rank matrix completion by Riemannian optimization," *SIAM Journal on Optimization*, vol. 23, no. 2, pp. 1214–1236, 2013.

[33] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2008, ISBN: 9780691132983.

[34] J. Townsend, "Differentiating the singular value decomposition," Technical Report 2016, Tech. Rep., 2016.

[35] L. A. Rossman, R. M. Clark, and W. M. Grayman, "Modeling chlorine residuals in drinking-water distribution systems," *Journal of Environmental Engineering*, vol. 120, no. 4, pp. 803–820, 1994.

[36] T. N. for Research Core Team, *Transportation networks for research*, https : / / github . com / bstabler / TransportationNetworks, Accessed: November, 23rd, 2024.

[37] L. J. Leblanc, "An algorithm for the discrete network design problem," *Transportation Science*, vol. 9, no. 3, pp. 183–199, 1975.

[38] V. De Silva and J. B. Tenenbaum, "Sparse multidimensional scaling using landmark points," Stanford University, Tech. Rep., 2004.

[39] C. K. Williams and C. E. Rasmussen, *Gaussian Processes for Machine Learning*. MIT press Cambridge, MA, 2006, vol. 2.

[40] L. A. Rossman, "An overview of epanet version 3.0," *Water Distribution Systems Analysis*, pp. 14–18, 2010.

[41] T. Seo, "Uxsim: Lightweight mesoscopic traffic flow simulator in pure python," *Journal of Open Source Software*, vol. 10, no. 106, p. 7617, 2025.