Graph Kalman Filters

Daniele Zambon[*1], **Cesare Alippi**[*1 2]
[1] The Swiss AI Lab IDSIA USI-SUPSI, Università della Svizzera italiana
[2] Politecnico di Milano
{daniele.zambon, cesare.alippi}@usi.ch

Abstract

The well-known Kalman filters model dynamical systems by relying on state-space representations with the next state updated, and its uncertainty controlled, by fresh information associated with newly observed system outputs. This paper generalizes, for the first time in the literature, Kalman and extended Kalman filters to discrete-time settings where inputs, states, and outputs are represented as attributed graphs whose topology and attributes can change with time. The setup allows us to adapt the framework to cases where the output is a vector or a scalar too (node/graph level tasks). Within the proposed theoretical framework, the unknown state transition and readout are learned end-to-end along with the downstream prediction task.

1 Introduction

The Kalman Filter (KF) [17] is a state-space representation architecture for modeling dynamical systems. Since its introduction more than 60 years ago, the KF has been standing out for its performance in tracking and controlling applications as well as for its relative simplicity. The KF operates on linear dynamical systems of the form

$$\begin{cases} \mathbf{h}_t = \mathbf{F}\,\mathbf{h}_{t-1} + \mathbf{G}\,\mathbf{x}_{t-1} + \boldsymbol{\eta}_{t-1}, \\ \mathbf{y}_t = \mathbf{H}\,\mathbf{h}_t + \boldsymbol{\nu}_t, \end{cases} \tag{1}$$

by estimating the hidden system state vector $\mathbf{h}_t \in \mathbb{R}^{d_h}$ at time t given input vector $\mathbf{x}_{t-1} \in \mathbb{R}^{d_x}$ and output vector $\mathbf{y}_t \in \mathbb{R}^{d_y}$; $\{\boldsymbol{\eta}_t\}_t$ and $\{\boldsymbol{\nu}_t\}_t$ are Gaussian white-noise stochastic processes. At each time step t, the system state and its uncertainty update by iteratively leveraging on previous estimates and incorporating newly observed system outputs and inputs while accounting for uncertainties. Notably, KF estimators are proven to be optimal, being unbiased and of minimum variance. Generalizations and variants of the KF cover continuous-time reformulations of Equation 1, non-Gaussian noise distributions, time-variant state-space setups, and nonlinear formulations that can be cast in the form

$$\begin{cases} \mathbf{h}_t = f_{\text{ST}}(\mathbf{h}_{t-1}, \mathbf{x}_{t-1}) + \boldsymbol{\eta}_{t-1}, \\ \mathbf{y}_t = f_{\text{RO}}(\mathbf{h}_t) + \boldsymbol{\nu}_t; \end{cases} \tag{2}$$

we refer the reader to [26] and Section 3 for a review. Nowadays graph-based models integrating relational information among the sensors/components of multivariate systems have been demonstrated to be extremely powerful and effective spatio-temporal predictors [18, 28, 12]. These include graph neural networks [3, 4] and their extensions incorporating temporal information, usually referred to as spatio-temporal graph neural networks (STGNNs) [24, 23, 10]. The literature on KF with graph-structured data is however less mature, as discussed in Section 3.

[*] Equal contribution.

Temporal Graph Learning Workshop @ NeurIPS 2023, New Orleans.

Contribution In this paper, for the first time in the literature, we provide Graph KF, a graph-based formulation of the KF where inputs, outputs, and states are attributed graphs whose topology is allowed to change over time. A deep neural implementation derives from a graph state space (GSS) [29] modeling a discrete-time, time-invariant, stochastic data-generating process $\mathcal{P}$

$$\begin{cases} \mathbf{h}_t = f_{\text{ST}}(\mathbf{h}_{t-1}, \mathbf{x}_{t-1}, \boldsymbol{\eta}_{t-1}), \\ \mathbf{y}_t = f_{\text{RO}}(\mathbf{h}_t, \boldsymbol{\nu}_t), \end{cases} \tag{3}$$

where inputs $\mathbf{x}_{t-1}$, states $\mathbf{h}_t$, and outputs $\mathbf{y}_t$ are attributed graphs belonging to graph spaces $\mathcal{X}$, $\mathcal{H}$, and $\mathcal{Y}$, respectively. Stochastic processes $\{\boldsymbol{\eta}_t\}_t$, $\{\boldsymbol{\nu}_t\}_t$ are white noise impacting on the node signals and/or the topology of the graphs.

Challenges The GSS formulation in Equation 3 poses three main challenges that are addressed in this paper: (a) State-transition and the readout functions f_{ST} and f_{RO} are assumed to be unknown and, differently from Equation 2, are nonlinear also with respect to the noise components. (b) Unknown states are attributed graphs of unknown, and possibly time-varying, topology, hence it is requested us to estimate both node features and graph topology. Note that the topology is a discrete entity; as such, it is not amenable to standard gradient-based optimization. (c) The data dimensionality – say the nodes and edges of the involved graphs – is not fixed and can be very large, yielding ill-posed estimation, in general. We address the above challenges, by devising a spatio-temporal graph neural network (STGNN) that approximates the state transition and the readout functions.

After reviewing the standard KF (Section 2), we introduce the considered GSS models (Section 4), we derive the proposed Graph KF architecture (Section 5), and we empirically validate it (Section 6).

2 Kalman filters

Consider the discrete time-invariant system model (2) with random initial state $\mathbf{h}_0 \in \mathbb{R}^{d_h}$ drawn from a known finite-variance distribution, state transition f_{SC} and readout f_{RO} are differentiable with respect to the states and affected by white-noise stochastic processes with covariance matrices $\text{Cov}[\boldsymbol{\eta}_t] = \mathbf{Q}_t$ and $\text{Cov}[\boldsymbol{\nu}_t] = \mathbf{R}_t$, for all t. Let $\mathbf{h}_0, \boldsymbol{\eta}_t$ and $\boldsymbol{\nu}_t$ be mutually independent, for all t.

Assume to have observed $\mathbf{x}_{i-1}, \mathbf{y}_i$ for all $i < t$ and generated an estimate $\mathbf{h}_{t-1}^+$ of $\mathbb{E}[\mathbf{h}_{t-1}]$ with error covariance matrix $P_{t-1}^+ \doteq \text{Cov}[\mathbf{h}_{t-1} - \mathbf{h}_{t-1}^+]$. A single iteration of the KF algorithm aimed at modeling Equation 2 is two-step: (i) Once input $\mathbf{x}_{t-1}$ is available, an *a priori* estimate $\mathbf{h}_t^-$ of $\mathbb{E}[\mathbf{h}_t]$ is produced along with error covariance matrix $\mathbf{P}_t^- \doteq \text{Cov}[\mathbf{h}_t - \mathbf{h}_t^-]$ and followed by a prediction $\mathbf{y}_t^- = f_{\text{ST}}(\mathbf{h}_t^-)$ of the system output $\mathbf{y}_t$; estimates denoted with superscript "$-$" are named "a priori" as they are obtained before observing the system output $\mathbf{y}_t$. (ii) Once $\mathbf{y}_t$ is observed, an *a posteriori* estimate $\mathbf{h}_t^+$ refines $\mathbf{h}_t^-$ and the updated matrix $\mathbf{P}_t^+ \doteq \text{Cov}[\mathbf{h}_t - \mathbf{h}_t^+]$ is derived from $\mathbf{P}_t^-$.

2.1 KF for linear systems

This section summarizes the KF procedure for discrete-time, time-variant, linear systems

$$\begin{cases} \mathbf{h}_t = \mathbf{F}_{t-1}\mathbf{h}_{t-1} + \mathbf{G}_{t-1}\mathbf{x}_{t-1} + \boldsymbol{\eta}_{t-1}, \\ \mathbf{y}_t = \mathbf{H}_t\mathbf{h}_t + \boldsymbol{\nu}_t, \end{cases} \tag{4}$$

a generalization of the time-invariant system (1) where matrices $\mathbf{F}_{t-1}$, $\mathbf{G}_{t-1}$ and $\mathbf{H}_t$ depends on t. For a detailed derivation, we refer to [26], and point out that the literature shows different equivalent rewritings to meet specific implementational requirements. Although we aim at developing a KF for time-invariant GSS models like that in Equation 3, in order to deal with nonlinear state transition and readout, it is suitable to rely on the following time-variant derivation.

A priori estimate Note that $\mathbb{E}[\mathbf{h}_t] = \mathbf{F}_{t-1}\mathbb{E}[\mathbf{h}_{t-1}] + \mathbf{G}_{t-1}\mathbf{x}_{t-1} + 0$, so, if $\mathbb{E}[\mathbf{h}_{t-1}^+] = \mathbb{E}[\mathbf{h}_{t-1}]$, then the following is an unbiased estimator of $\mathbb{E}[\mathbf{h}_t]$:

$$\mathbf{h}_t^- \doteq \mathbf{F}_{t-1}\mathbf{h}_{t-1}^+ + \mathbf{G}_{t-1}\mathbf{x}_{t-1}; \tag{5}$$

Thanks to the independence between $\boldsymbol{\eta}_{t-1}$ and both $\mathbf{h}_{t-1}$ and $\mathbf{h}_{t-1}^+$, the covariance matrix of the a priori estimation error can be expressed as a function of the $\mathbf{P}_{t-1}^+$, from the previous time step:

$$\mathbf{P}_t^- = \mathbf{F}_{t-1}\mathbf{P}_{t-1}^+\mathbf{F}_{t-1}^\top + \mathbf{Q}_{t-1}. \tag{6}$$

Unbiased predictions can be made as $\mathbf{y}_t^- = \mathbf{H}_t \mathbf{h}_t^-$.

A posteriori estimate The a posteriori estimate has the form

$$\mathbf{h}_t^+ \doteq \mathbf{h}_t^- + \mathbf{K}_t(\mathbf{y}_t - \mathbf{y}_t^-) \tag{7}$$

where matrix $\mathbf{K}_t$ is known as *gain* while residual $\mathbf{y}_t - \mathbf{y}_t^-$ is called *innovation*. As $\mathbf{h}_t^-$ is unbiased, then $\mathbb{E}[\mathbf{y}_t - \mathbf{y}_t^-] = 0$, and we see that $\mathbf{h}_t^+$ is unbiased as well, regardless of the choice of the gain. Therefore, we can select $\mathbf{K}_t$ to minimize the total variance

$$\text{Tr}(\mathbf{P}_t^+) = \mathbb{E}[(\mathbf{h}_t - \mathbf{h}_t^+)^\top (\mathbf{h}_t - \mathbf{h}_t^+)], \tag{8}$$

i.e., the trace of the matrix $\mathbf{P}_t^+$. By exploiting the independence between $\mathbf{h}_t^-$ and $\boldsymbol{\nu}_t$, the gain minimizing $\text{Tr}(\mathbf{P}_t^+)$ is

$$\mathbf{K}_t = \mathbf{P}_t^- \mathbf{H}_t^\top (\mathbf{H}_t \mathbf{P}_t^- \mathbf{H}_t^\top + \mathbf{R}_t)^{-1}. \tag{9}$$

2.2 Extended KF for nonlinear systems

The Extended KF (EKF) [27] adapts the KF of Section 2.1 to the nonlinear case of Equation 2. EKF operates by linearizing the state-transition and readout functions around the last available state estimate. EKF requires the first-order Taylor approximation of function $f_{\text{ST}}(\,\cdot\,, \mathbf{x}_{t-1})$ around $\mathbf{h}_{t-1}^+$

$$f_{\text{ST}}(\mathbf{h}, \mathbf{x}_{t-1}) \approx f_{\text{ST}}\left(\mathbf{h}_{t-1}^+, \mathbf{x}_{t-1}\right) + \underbrace{\nabla_{\mathbf{h}} f_{\text{ST}}(\mathbf{h}_{t-1}^+, \mathbf{x}_{t-1})}_{\mathbf{F}_{t-1}}(\mathbf{h} - \mathbf{h}_{t-1}^+) \tag{10}$$

Similarly, we expand with Taylor function f_{RO} around estimate $\mathbf{h}_t^-$

$$f_{\text{RO}}(\mathbf{h}) \approx f_{\text{RO}}\left(\mathbf{h}_t^-\right) + \big(\underbrace{\nabla_{\mathbf{h}} f_{\text{RO}}(\mathbf{h}_t^-)}_{\mathbf{H}_t}\big)(\mathbf{h} - \mathbf{h}_t^-). \tag{11}$$

By linearizing the time-invariant system (2) we obtain a linear time-variant system like (4), where $\mathbf{F}_{t-1}$, $\mathbf{G}_{t-1}$, and $\mathbf{H}_t$ depend on the given input and current state estimates. The EKF is applicable to more general system models, where the interaction with the noise processes is nonlinear. We expand the discussion in Section 5, when deriving the KF for graphs.

3 Related work

The EKF [27] has been further generalized to account for orders beyond the first [2]. The unscented KF employs particle filtering to address some of the drawbacks of EKF [16, 19]. The theory of reproducing kernel Hilbert space is another viable solution to operate with nonlinear systems and non-Gaussian noise [21, 5, 7]. Regarding graph data, the research focused on linear systems with known topology [25, 14, 15]. With [22] introducing the analysis over a known, dynamic topology.

4 Graph state-space models

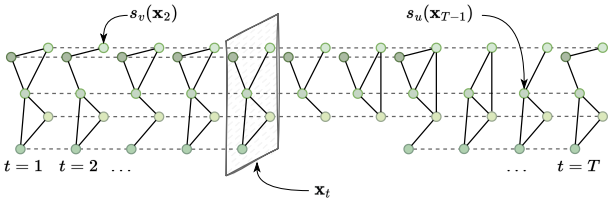

Figure 1: An example of a spatio-temporal data over a set $\mathbb{V}_{\mathbf{x}}$ of 5 nodes. Graph $\mathbf{x}_t$ is given at each step t. $\mathbf{x}_t$ is defined over a node set $V(\mathbf{x}_t) \subseteq \mathbb{V}_{\mathbf{x}}$ and has node signals $s_v(\mathbf{x}_t) \in \mathbb{R}^{d_x}$ associated with each given node.

The input graph $\mathbf{x}_t \in \mathcal{X}$ at time t is defined over node set $V(\mathbf{x}_t)$, e.g., associated with the sensors of a sensor network, and edge set represented as adjacency matrix $\mathbf{A}(\mathbf{x}_t)$ that encodes the relations existing among the nodes, such as physical proximity, signal correlations, or causal dependencies. The node sets and the topologies observed at different time steps t, t' are generally different but, typically, $V(\mathbf{x}_t) \cap V(\mathbf{x}_{t'}) \neq \emptyset$, implying the existence of a partial node correspondence over time. We denote with $\mathbb{V}_{\mathbf{x}} = \bigcup_t V(\mathbf{x}_t)$ the union set of all nodes, whose cardinality is assumed to be finite. Input graphs are attributed with node features attached to them, like sensor readings, collected in graph signal $s(\mathbf{x}_t) \in \mathbb{R}^{|\mathbb{V}_{\mathbf{x}}| \times d_x}$. A

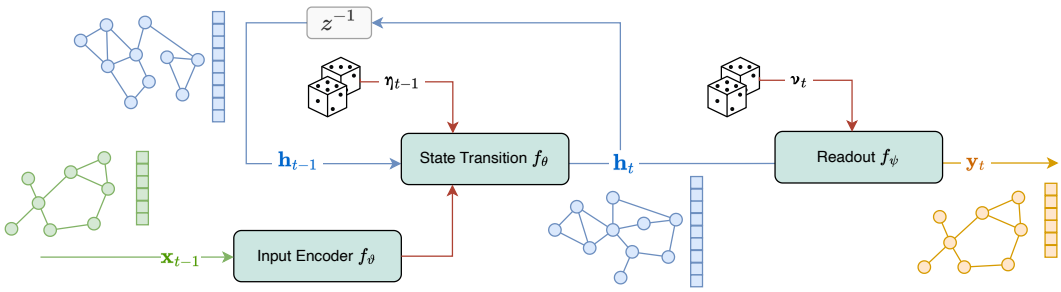

Figure 2: Block diagram of the GSS stochastic model in Equation 13.

visual representation of the resulting graph-based spatio-temporal data sequence $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t, \ldots$ is provided in Figure 1.

Given sequence $\{\mathbf{x}_t\}_t$ of graphs defined over node set $\mathbb{V}_\mathbf{x}$, we aim at predicting output graphs $\mathbf{y}_t \in \mathcal{Y}$ at each time step t. We model the data-generating process $\mathcal{P}$ as formulated in Equation 3, with system model

$$\begin{cases} \mathbf{h}_t = f_{\text{ST}}(\mathbf{h}_{t-1}, \mathbf{x}_{t-1}, \boldsymbol{\eta}_{t-1}), \\ \mathbf{y}_t = f_{\text{RO}}(\mathbf{h}_t, \boldsymbol{\nu}_t), \end{cases} \tag{12}$$

involving representations of the system states as graphs $\mathbf{h}_t \in \mathcal{H}$; with a consistent notation, $V(\mathbf{h}_t) \in \mathbb{V}_\mathbf{h}$, $\mathbf{A}(\mathbf{h}_t)$, and $s(\mathbf{h}_t)$ ($V(\mathbf{y}_t) \in \mathbb{V}_\mathbf{y}$, $\mathbf{A}(\mathbf{y}_t)$, and $s(\mathbf{y}_t)$) denote the node set, the adjacency matrix and the signal of state graph $\mathbf{h}_t$ (output graph $\mathbf{y}_t$). Stochastic processes $\{\boldsymbol{\eta}_t\}_t$ and $\{\boldsymbol{\nu}_t\}_t$ are white noise impacting the edges and the node signals[2] of the states and outputs, respectively. Functions f_{ST} and f_{RO}, as well as the noise distributions, are assumed unknown with finite second moments. Finally, exogenous variables, like those referring to extra sensor information or functional relations, can be included as well in the framework and encoded in $\mathbf{x}_t$ to avoid overwhelming notation. The general formulation of the GSS model in Equation 12 does not require any identification between the nodes in sets $\mathbb{V}_\mathbf{x}$, $\mathbb{V}_\mathbf{h}$, and $\mathbb{V}_\mathbf{y}$, although this might be the case in some scenarios (see [29]).

We introduce a GSS family of stochastic predictive models [29]

$$\begin{cases} \mathbf{h}_t = f_\theta(\mathbf{h}_{t-1}, f_\vartheta(\mathbf{x}_{t-1}), \boldsymbol{\eta}_{t-1}) \\ \mathbf{y}_t = f_\psi(s(\mathbf{h}_t), \boldsymbol{\nu}_t) \end{cases} \tag{13}$$

where $\mathbf{h}_t, \mathbf{y}_t, \boldsymbol{\eta}_{t-1}$ and $\boldsymbol{\nu}_t$ are random variables. Parameters θ, ϑ, and ψ are learned from data, i.e., from a realization of process $\mathcal{P}$. Initial condition $\mathbf{h}_0$ is drawn from a given prior distribution $P_{\mathbf{h}_0}$. Function f_ϑ is the input encoder, mapping $\mathbf{x}_{t-1}$ to the nodes of graph space $\mathcal{H}$, function f_θ models the state transition inferring both the graph topology of $\mathbf{h}_t$ and the associated node signal, whereas f_ψ is the readout. Graph functions f_ϑ, f_θ, and f_ψ are parametrized in real vectors ϑ, θ, and ψ, respectively, and learned directly from data; we assume them differentiable with respect to the associated parameter vectors. A schematic view of GSS model (13) is given in Figure 2.

Importantly, and differently from what is most often done with state-space models, state representations are first predicted and then refined once the system output is observed, in line with traditional vector KF. While training the model parameters is carried out with standard deep learning techniques, the state estimate refinement follows the KF proposed here and is derived in the following section.

5 Graph Kalman filter

The proposed Graph KF follows the linearization of the EKF, taking care of differentiating with respect to the noise components, too. Accordingly, we assume that both f_θ and f_ψ are differentiable with respect to the state and the noise terms, as better formalized in following Section 5.1. To facilitate readability, we assume the following.

Assumption 1. *The node sets of input, state, and output graphs coincide ($\mathbb{V}_\mathbf{y} = \mathbb{V}_\mathbf{h} = \mathbb{V}_\mathbf{x}$).*

[2]For instance, $\boldsymbol{\eta}_{t-1}$ can be a vector such that $s(\mathbf{h}_t) = \mathbf{z}_{t-1} + \boldsymbol{\eta}_{t-1}$, with $\mathbf{z}_{t-1} \doteq s(f(\mathbf{h}_{t-1}, \mathbf{x}_{t-1}))$, or $s_v(\mathbf{h}_t) = g(\mathbf{z}_{t-1,v}, \boldsymbol{\eta}_{t-1,v})$ for generic function g applied to each node $v \in V(\mathbf{h}_t)$. Differently, an example of noise impacting the state's topology is when $\boldsymbol{\eta}_{t-1}$ is a matrix of entries in $\{-1, 0, 1\}$ perturbing a constant (binary) matrix $\mathbf{A}_0$ so that adjacency $\mathbf{A}(\mathbf{h}_t)$ is given as $\mathbf{A}_0 + \boldsymbol{\eta}_{t-1}$.

Assumption 2. *The topology of the output graph is either that of the state ($\mathbf{A}(\mathbf{y}_t) = \mathbf{A}(\mathbf{h}_t)$) or the input ($\mathbf{A}(\mathbf{y}_t) = \mathbf{A}(\mathbf{x}_{t-1})$).*

We stress that the above assumptions are either made to render the derivation more amenable or ease the readability of the outcomes by providing a simplified notation. For instance, Assumption 1 ensures an immediate correspondence between nodes of the inputs, states, and outputs (without the need for graph pooling and upscaling operators as in [29]), whereas predicting the output adjacency matrix $\mathbf{A}(\mathbf{y}_t)$ – thus relaxing Assumption 2 – would only request additional components of f_ψ to be linearized. It follows that weaker assumptions can be considered at the cost of more complex mathematics that, however, will not change the spirit of what is derived.

5.1 State-transition and readout functions linearization

In the following, we derive the Graph KF assuming that $\boldsymbol{\nu}_t$ is a $|\mathbb{V}_\mathbf{h}|$-dimensional vector whose components perturb the output signal $s(\mathbf{y}_t)$ and $\boldsymbol{\eta}_{t-1}$ is a $|\mathbb{V}_\mathbf{h}| \times |\mathbb{V}_\mathbf{h}|$ matrix impacting the state topology $\mathbf{A}(\mathbf{h}_t)$, thus showing both node-level and edge-level noise scenarios at once. For brevity, we also denote signal $s(\mathbf{h}_t)$ as $\mathbf{s}_t$ and encoded input $f_\vartheta(\mathbf{x}_{t-1})$ as $\tilde{\mathbf{x}}_{t-1}$.

Following the EKF procedure, we expand the nonlinear system (13) with Taylor approximating $f_\theta(\mathbf{s}^+_{t-1}, \tilde{\mathbf{x}}_{t-1}, \boldsymbol{\eta}_{t-1})$ and $f_\psi(\mathbf{s}^-_t, \boldsymbol{\nu}_t)$. Accordingly, we requests f_θ to be differentiable with respect to $\mathbf{s}_{t-1}$ and $\boldsymbol{\eta}_{t-1}$; similarly, f_ψ to be differentiable with respect to $\mathbf{s}_t$ and $\boldsymbol{\nu}_t$. We get

$$\begin{aligned} f_\theta(\mathbf{s}, \tilde{\mathbf{x}}_{t-1}, \boldsymbol{\eta}) \approx f_\theta\left(\mathbf{s}^+_{t-1}, \tilde{\mathbf{x}}_{t-1}, \mathbf{0}\right) + \left(\nabla_\mathbf{s} f_\theta(\mathbf{s}^+_{t-1}, \tilde{\mathbf{x}}_{t-1}, \mathbf{0})\right)(\mathbf{s} - \mathbf{s}^+_{t-1}) \\ + \left(\nabla_{\boldsymbol{\eta}} f_\theta(\mathbf{s}^+_{t-1}, \tilde{\mathbf{x}}_{t-1}, \mathbf{0})\right) \bullet \boldsymbol{\eta} \qquad (14) \\ = f_\theta\left(\mathbf{s}^+_{t-1}, \tilde{\mathbf{x}}_{t-1}, \mathbf{0}\right) + \mathbf{F}_{t-1}(\mathbf{s} - \mathbf{s}^+_{t-1}) + \mathbf{L}_{t-1} \bullet \boldsymbol{\eta} \qquad (15) \\ = \mathbf{F}_{t-1}\mathbf{s} + f_\theta\left(\mathbf{s}^+_{t-1}, \tilde{\mathbf{x}}_{t-1}, \mathbf{0}\right) - \mathbf{F}_{t-1}\mathbf{s}^+_{t-1} + \mathbf{L}_{t-1} \bullet \boldsymbol{\eta} \qquad (16) \end{aligned}$$

where $\mathbf{B} \bullet \mathbf{C} \in \mathbb{R}^{|\mathbb{V}_\mathbf{h}|}$ denotes the product $[\mathbf{B} \bullet \mathbf{C}]_v = \sum_{i,j=1}^{|\mathbb{V}_\mathbf{h}|} \mathbf{B}_{v,i,j}\mathbf{C}_{i,j}$ for all $\mathbf{B} \in \mathbb{R}^{|\mathbb{V}_\mathbf{h}| \times |\mathbb{V}_\mathbf{h}| \times |\mathbb{V}_\mathbf{h}|}$ and $\mathbf{C} \in \mathbb{R}^{|\mathbb{V}_\mathbf{h}| \times |\mathbb{V}_\mathbf{h}|}$. Similarly, we linearize the readout function:

$$\begin{aligned} f_\psi(\mathbf{s}, \boldsymbol{\nu}) \approx f_\psi(\mathbf{s}^-_t, \mathbf{0}) + \underbrace{(\nabla_\mathbf{s} f_\psi(\mathbf{s}^-_t, \mathbf{0}))}_{\mathbf{H}_t}(\mathbf{s} - \mathbf{s}^-_t) + \underbrace{(\nabla_{\boldsymbol{\nu}} f_\psi(\mathbf{s}^-_t, \mathbf{0}))}_{\mathbf{M}_t}\boldsymbol{\nu} \qquad (17) \\ = \mathbf{H}_t\mathbf{s} + f_\psi(\mathbf{s}^-_t, \mathbf{0}) - \mathbf{H}_t\mathbf{s}^-_t + \mathbf{M}_t\boldsymbol{\nu}_t. \qquad (18) \end{aligned}$$

We stress that, even though the role of the graph topology might not be immediately evident from the notation above, graph-based processing is still carried out while computing, e.g., $f_\theta\left(\mathbf{s}^+_{t-1}, \tilde{\mathbf{x}}_{t-1}, \mathbf{0}\right)$. Moreover, whenever the state transition and readout act component-wise on their inputs and share parameters among nodes – as in many STGNNs – then part of the computation can be parallelized and reused. Finally, note that with modern libraries, such as PyTorch [20] and TensorFlow [1] along with their ecosystems [9, 6, 11], the linearizations can be computed automatically in a closed form, thus allowing us to apply the proposed Graph KF to basically any deep learning architecture.

5.2 Graph KF iterations

Assume to have learned parameter vectors θ, ϑ, and ψ of GSS model (13) (see Section 5.3), then the following iterations define the proposed Graph KF. Initialize

$$\mathbf{s}^+_0 = \mathbb{E}_{\mathbf{s} \sim P_{\mathbf{s}_0}}[\mathbf{s}] \in \mathbb{R}^{|\mathbb{V}_\mathbf{h}|}, \qquad \mathbf{P}^+_0 = \mathrm{Cov}_{\mathbf{s} \sim P_{\mathbf{s}_0}}[\mathbf{s}] \in \mathbb{R}^{|\mathbb{V}_\mathbf{h}| \times |\mathbb{V}_\mathbf{h}|}, \qquad (19)$$

from a prior distribution $P_{\mathbf{s}_0}$. Then, for $t = 1, 2, 3, \ldots$

(i) Encode input graph: $\tilde{\mathbf{x}}_{t-1} = f_\vartheta(\mathbf{x}_{t-1})$;

(ii) Update the a priori state estimate: $\mathbf{s}^-_t = f_\theta(\mathbf{s}^+_{t-1}, \tilde{\mathbf{x}}_{t-1}, \mathbf{0})$;

(iii) Make the prediction $\mathbf{y}^-_t = f_\psi(\mathbf{s}^-_t, \mathbf{0})$.

Steps (i)–(iii) are standard practice in state-space modeling, where $\mathbf{s}^-_{t-1}$ is considered as $\mathbf{s}^+_{t-1}$. The refinement $\mathbf{s}^+_t \leftarrow \mathbf{s}^-_t$ of the a priori state estimate $\mathbf{s}^-_t$ is carried out as follows

(iv) Compute the Jacobian associated with the state transition:

$$\mathbf{F}_{t-1} = \nabla_{\mathbf{s}} f_\theta(\mathbf{s}^+_{t-1}, \tilde{\mathbf{x}}_{t-1}, \mathbf{0}) \in \mathbb{R}^{|\mathbb{V}_\mathbf{h}| \times |\mathbb{V}_\mathbf{h}|}, \tag{20}$$

$$\mathbf{L}_{t-1} = \nabla_{\boldsymbol{\eta}} f_\theta(\mathbf{s}^+_{t-1}, \tilde{\mathbf{x}}_{t-1}, \mathbf{0}) \in \mathbb{R}^{|\mathbb{V}_\mathbf{h}| \times |\mathbb{V}_\mathbf{h}|^2}; \tag{21}$$

(v) Compute the Jacobian of the readout:

$$\mathbf{H}_t = \nabla_{\mathbf{s}} f_\psi(\mathbf{s}^-_t, \mathbf{0})) \in \mathbb{R}^{|\mathbb{V}_\mathbf{h}| \times |\mathbb{V}_\mathbf{h}|}, \qquad \mathbf{M}_t = \nabla_{\boldsymbol{\nu}} f_\psi(\mathbf{s}^-_t, \mathbf{0})) \in \mathbb{R}^{|\mathbb{V}_\mathbf{h}| \times |\mathbb{V}_\mathbf{h}|}; \tag{22}$$

(vi) Update the a priori error covariance: $\mathbf{P}^-_t = \mathbf{F}_{t-1}\mathbf{P}^+_{t-1}\mathbf{F}^\top_{t-1} + \mathbf{L}_{t-1}\mathbf{Q}_{t-1}\mathbf{L}^\top_{t-1}$;

(vii) Compute the gain matrix: $\mathbf{K}_t = \mathbf{P}^-_t\mathbf{H}^\top_t(\mathbf{H}_t\mathbf{P}^-_t\mathbf{H}^\top_t + \mathbf{M}_t\mathbf{R}_t\mathbf{M}^\top_t)^{-1}$;

(viii) Update the a posteriori state estimate:

$$\mathbf{s}^+_t = \mathbf{s}^-_t + \mathbf{K}_t(\mathbf{y}_t - \mathbf{y}^-_t), \tag{23}$$

$$\mathbf{P}^+_t = (\mathbb{I} - \mathbf{K}_t\mathbf{H}_t)\mathbf{P}^-_t(\mathbb{I} - \mathbf{K}_t\mathbf{H}_t)^\top + \mathbf{K}_t\mathbf{M}_t\mathbf{R}_t\mathbf{M}^\top_t\mathbf{K}^\top_t. \tag{24}$$

5.3 Model training and Kalman filtering

We train model parameters θ, ϑ, and ψ by gradient-based optimization minimizing the mean squared error (MSE) between $\mathbf{y}_t$ and $\mathbf{y}^-_t$, while the Kalman gain $\mathbf{K}_t$ and error covariance matrices $\mathbf{P}^-_t$ and $\mathbf{P}^+_t$ are estimated during inference. Note that the computations in steps (ii) and (iv) do not involve the noise term $\boldsymbol{\eta}_{t-1}$; similarly, steps (iii) and (v) do not account for the noise term $\boldsymbol{\nu}_t$. However, if the noise distributions have to be learned along with model parameters, the above iterations can be modified to consider samples of $\boldsymbol{\eta}_{t-1}$ and $\boldsymbol{\nu}_t$ drawn from the probabilistic model learned so far, and their probability distributions optimized, e.g., by relying on the reparameterization trick and maximum likelihood criteria.

6 Empirical validation

As our contribution is theoretical and methodological, here we validate the correctness of the procedures, methods, and outcomes, and showcase the potential of the proposed Graph KF. In particular, we design a set of controlled experiments to analyze the main interplaying elements: the training of a GSS model as in Equation 13, the predictions based on the learned model (steps (i)–(iii)), and the Kalman-filter refinement (KFR) in steps (iv)–(viii). We leave real applications to future research.

6.1 Datasets

We generate two GSS system models like (12), LinGSS and NonLinGSS, both characterized by the data-generating process detailed in next paragraphs. The specific parameters of the two GSS models are reported in Table 1, further details are provided in the supplementary material; in particular, Figure 3 shows the input, state, and output signals of three sample nodes.

Table 1: Parameter configuration of the GSS system models. "id" is the identity function.

		LinGSS		**NonLinGSS**	
λ_0	λ_1	20	5	20	5
θ_{TM}	θ_{SP}	0.6	0.3	0.6	-0.3
ψ_0	ψ_1	-0.5	2.0	-2.0	5.0
σ_η	σ_ν	0.25	0.12	0.25	0.12
ρ_{ST}	ρ_{RO}	id	id	tanh	tanh

Inputs For each node $v \in \mathbb{V}_\mathbf{x}$, input signal $s(\mathbf{x}_t)$ is binary and random in time duration, generated alternating between runs of 1's and 0's so that the associated run lengths are drawn from a Poisson(λ_1) and a Poisson(λ_0), respectively. Inputs have no relational information ($E(\mathbf{x}_t) = \emptyset$).

States Recalling the notation $\mathbf{s}_t = s(\mathbf{h}_t)$ adopted in previous sections, states are updated according to the state-transition function

$$\mathbf{s}_t = f_{\text{ST}}(\mathbf{h}_{t-1}, \mathbf{x}_{t-1}, \boldsymbol{\eta}_{t-1}) \doteq \rho_{\text{ST}}\left(\left(\theta_{\text{TM}}\mathbb{I} + \theta_{\text{SP}}\bar{\mathbf{A}}_\theta\right)(\mathbf{s}_{t-1} + s(\mathbf{x}_{t-1}))\right) + \boldsymbol{\eta}_{t-1} \tag{25}$$

where $\boldsymbol{\eta}_{t-1}$ is i.i.d. from a zero-mean Gaussian distribution with standard deviation σ_η, matrix $\bar{\mathbf{A}}_\theta = \mathbf{D}^{-1/2}(\mathbb{I} + \mathbf{A}_\theta)\mathbf{D}^{-1/2}$ is the adjacency $\mathbf{A}_\theta$ normalized by diagonal matrix $\mathbf{D}$ of node degrees (self-loops included), $\mathbb{V}_\mathbf{h} = \mathbb{V}_\mathbf{x}$, and ρ_{ST} is a nonlinearity applied component-wise. Initial state $\mathbf{s}_0 = \boldsymbol{\eta}_0$ is white noise.

Table 2: Performance of Replica and STGNN models with and without the KFR. The prediction error is averaged over 10 runs whereas the RPI is estimated over the test mini-batches and the 10 runs. Results are reported in the format "mean$_{\pm \text{std}}$" The prediction error in the first three rows is a single value as the model parameters are predefined.

	LinGSS			NonLinGSS		
	Pred. Err. (MSE)		**RPI**	**Pred. Err.** (MSE)		**RPI**
Model	w/o KFR	w/ KFR	(MSE%)	w/o KFR	w/ KFR	(MSE%)
Replica	0.384	0.271	-98.6$_{\pm 1.2}$	0.483	0.359	-88.7$_{\pm 1.6}$
Replica Exp	0.267	0.267	-98.8$_{\pm 0.6}$	0.349	0.349	-85.6$_{\pm 2.2}$
Replica GT	0.014	0.014	-98.9$_{\pm 0.0}$	0.014	0.014	-62.3$_{\pm 4.5}$
Replica Retrained	0.384$_{\pm 0.000}$	0.271$_{\pm 0.000}$	-98.6$_{\pm 1.1}$	0.429$_{\pm 0.000}$	0.327$_{\pm 0.000}$	-94.0$_{\pm 0.8}$
STGNN	0.389$_{\pm 0.001}$	0.336$_{\pm 0.019}$	-34.2$_{\pm 15.9}$	0.434$_{\pm 0.001}$	0.407$_{\pm 0.010}$	-13.5$_{\pm 7.7}$

Outputs System output $\mathbf{y}_t$ is obtained from the same readout applied to each node-level state $\mathbf{s}_{t,v}$:

$$s(\mathbf{y}_{t,v}) = f_{\text{RO}}(\mathbf{h}_{t,v}, \boldsymbol{\nu}_{t,v}) \doteq \rho_{\text{RO}}(\psi_0 + \psi_1 \mathbf{s}_{t,v}) + \boldsymbol{\nu}_{t,v}. \tag{26}$$

Noise terms $\{\boldsymbol{\nu}_{t,v}\}$ are i.i.d. Gaussian distributed with zero as mean and σ_ν as standard deviation. ρ_{RO} is a nonlinearity.

6.2 Approximating family of models

In the experiments below, we consider two types of approximating families of models (13).

Replica The first family of models is designed to contain the state transition (25) and readout (26) of the data-generating process. In particular, the Replica model parameters are exactly the four parameters $\theta_{\text{TM}}, \theta_{\text{SP}}, \psi_0$, and ψ_1.

STGNN The second family of models is a generic and relatively simple STGNN that does not contain the state-transition and readout functions of the data-generating process. This family is defined by the following architecture. Input encoder f_ϑ and the readout f_ψ are 2-layer dense networks both applied node-wise. State-transition function f_θ is composed of a message-passing layer

$$\mathbf{s}_t = f_\theta(\mathbf{s}_{t-1}, f_\vartheta(\mathbf{x}_{t-1}), \mathbf{0}) \doteq \mathbf{s}_{t-1} + f_\vartheta(\mathbf{x}_{t-1}) + \tanh\left(\mathbf{z}\mathbf{W}'_\theta + \tilde{\mathbf{A}}_\theta \mathbf{z}\mathbf{W}''_\theta\right) \tag{27}$$

performed on $\mathbf{z} = \gamma_\theta(\mathbf{s}_{t-1} + f_\vartheta(\mathbf{x}_{t-1}))$; γ_θ is a 2-layer dense network, and matrix $\tilde{\mathbf{A}}_\theta$ is a normalized version of the adjacency matrix $\mathbf{A}_\theta$ where each row adds up to 1. All modules have 7 hidden neurons per layer and the rectified linear unit as activation function.

The models are trained to predict the expected value $\mathbb{E}[\mathbf{y}_t]$; accordingly, the MSE is considered as loss function. Parameters σ_η, σ_ν and the topology of the states are considered known here. Note that considering a probabilistic or dynamic topology would not change the application of the KFR. Therefore, we removed such elements from the empirical validation to focus on the KFR part of the proposed Graph KF. Further experimental details are given in the supplementary material.

6.3 Positive effect of the Graph KF refinement

The first experiment studies the improvement brought by the KFR assuming to know the system model, i.e., the Replica model with parameters identical to those in Table 1. We compare the prediction error $\|\mathbf{y}_t^- - \mathbf{y}_t\|_2^2$ when $\mathbf{y}_t^- = f_{\text{RO}}(\hat{\mathbf{h}}_t)$ is computed from the following different states $\hat{\mathbf{h}}_t$:

(w/o KFR) $\hat{\mathbf{h}}_t = \mathbf{h}_t^-$ is the state estimate produced from the past states $\{\mathbf{h}_i : i < t\}$ and inputs $\{\mathbf{x}_i : i < t\}$ by performing steps (i)–(iii) with f_{ST} as state transition.

(w/ KFR) $\hat{\mathbf{h}}_t = \mathbf{h}_t^+$ is the state estimate produced from the past states $\{\mathbf{h}_i : i < t\}$ and inputs $\{\mathbf{x}_i : i < t\}$ by performing steps (i)–(viii) with f_{ST} as state transition, but applying also the KFR.

(Exp) $\hat{\mathbf{h}}_t = \mathbb{E}[\mathbf{h}_t]$ is the expected value of the state computed from Equation 25.

(GT) $\hat{\mathbf{h}}_t = \mathbf{h}_t$ is the true system state in Equation 25 affected by noise, thus serving as ground truth.

Results are reported in the top part of Table 2.

Results w/o KFR On LinGSS dataset, Replica GT relies on the true state and its performance matches the readout noise $\sigma_\nu^2 = (0.12)^2 = 0.0144$, whereas the Replica Exp performance is close to the target value $\sigma_\nu^2 + (\sigma_\eta \psi_1)^2 = 0.2644$. Note that although the state transition and readout are exactly those that generated the data, the noise on the state transition has a negative impact on the prediction performance. Therefore, and as expected, Replica model without the KFR performs worse than both baselines Replica Exp and Replica GT.

Results w/ KFR Conversely, KFR allows Replica model to approach the baseline performance of Replica Exp, while the performance of the reference model Replica GT cannot be achieved without information about the random realization of $\boldsymbol{\eta}_t$ noise. Table 2 displays also the following relative prediction improvement (RPI)

$$\frac{\|\mathbf{y}_t^+ - \mathbf{y}_t\|_2^2 - \|\mathbf{y}_t^- - \mathbf{y}_t\|_2^2}{\|\mathbf{y}_t^- - \mathbf{y}_t\|_2^2} \tag{28}$$

to further compare a priori and a posteriori estimates. The RPI values in Table 2 show that the prediction error is reduced to almost zero (equivalent to RPI of -100%), reassuring of the correct functioning of the KFR procedure. We should note that $\mathbf{y}_t^+$ here is receiving $\mathbf{y}_t$ as feedback to update the state estimate, however, we stress that for computing the prediction error in the first column of the table, the target $\mathbf{y}_t$ is *not* seen by the predictive model. For the same reason, observe also that despite the RPI values, applying the KFR to Replica GT and Replica Exp does not bring further down the MSE, as they are already optimal in their respective sense.

Results on NonLinGSS Similar results are observed for NonLinGSS dataset, too, where Replica model with KFR improves over Replica model without KFR, and whose performance approaches that of Replica Exp; we comment that in this nonlinear setting, the performance of Replica Ext is not expressed as $\sigma_\nu^2 + (\sigma_\eta \psi_1)^2$, as it depends on the nonlinearity ρ_{RO}, as well.

6.4 Graph KF refinement on trained models

The second set of experiments concerns performance improvements when using approximating models. In this problem setup, we expect that the trained models fit well the data-generating process. However, we do not expect them to identify (match exactly) the state transition and readout of the system model. Bottom part of Table 2 shows that all considered models benefit from the KFR. In particular, we observe that the Replica model trained on LinGSS matches the performance of Table 2, where the model parameters were set equal to those that generated the data (Table 1). Moreover, inspecting the learned parameters, we see they get close to the ground truth reported in Table 1. Interestingly, the parameters learned by Replica on NonLinGSS differ from the ground truth ($\sim$10% off) and enable better predictions than those of Table 2. Finally, the STGNN models (whose family does not contain the system model) do not performed as good as Replica ones, especially when comparing the refined versions, as confirmed also by the RPI. In spite of that, we stress that STGNN benefits from the KFR and, in fact, displays an RPI smaller than zero and achieves a better MSE than all the other models without KFR.

7 Conclusions

This paper extends for the first time the Kalman filter to scenarios where inputs, states, and outputs are attributed graphs of variable topology. The theoretical contribution of the paper addresses three main challenges related to dealing with nonlinear systems, learning graph topologies, and operating with a possibly large number of nodes. The correctness of the proposed Graph KF is empirically validated and the potentially achievable increase in the prediction performance is demonstrated. While hypotheses are made to facilitate the reading and derivation thanks to a lighter notational setup, their relaxation is the subject of future research that, however, does not change the spirit of what is here proposed.

Acknowledgments and Disclosure of Funding

This work was supported by the Swiss National Science Foundation project FNS 204061: *High-Order Relations and Dynamics in Graph Neural Networks*.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: a system for large-scale machine learning. In *Osdi*, volume 16, pages 265–283. Savannah, GA, USA, 2016.

[2] M. Athans, R. Wishner, and A. Bertolini. Suboptimal state estimation for continuous-time nonlinear systems from discrete noisy measurements. *IEEE Transactions on Automatic Control*, 13(5):504–514, October 1968. ISSN 1558-2523. doi: 10.1109/TAC.1968.1098986.

[3] Davide Bacciu, Federico Errica, Alessio Micheli, and Marco Podda. A gentle introduction to deep learning for graphs. *Neural Networks*, 129:203–221, 2020.

[4] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond Euclidean data. *IEEE Signal Processing Magazine*, 34 (4):18–42, 2017.

[5] Badong Chen, Xi Liu, Haiquan Zhao, and Jose C Principe. Maximum correntropy kalman filter. *Automatica*, 76:70–77, 2017.

[6] Andrea Cini and Ivan Marisca. Torch Spatiotemporal, 3 2022. URL `https://github.com/TorchSpatiotemporal/tsl`.

[7] Lujuan Dang, Badong Chen, Shiyuan Wang, Yuantao Gu, and José C. Príncipe. Kernel Kalman Filtering With Conditional Embedding and Maximum Correntropy Criterion. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 66(11):4265–4277, November 2019. ISSN 1558-0806. doi: 10.1109/TCSI.2019.2920773.

[8] William Falcon and The PyTorch Lightning team. PyTorch Lightning, 3 2019. URL `https://github.com/PyTorchLightning/pytorch-lightning`.

[9] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

[10] Jianfei Gao and Bruno Ribeiro. On the equivalence between temporal and static equivariant graph representations. In *International Conference on Machine Learning*, pages 7052–7076. PMLR, 2022.

[11] Daniele Grattarola and Cesare Alippi. Graph neural networks in tensorflow and keras with spektral [application notes]. *IEEE Computational Intelligence Magazine*, 16(1):99–106, 2021.

[12] Jonas Berg Hansen, Stian Normann Anfinsen, and Filippo Maria Bianchi. Power flow balancing with decentralized graph neural networks. *IEEE Transactions on Power Systems*, 2022.

[13] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2. URL `https://doi.org/10.1038/s41586-020-2649-2`.

[14] Elvin Isufi, Andreas Loukas, Nathanael Perraudin, and Geert Leus. Forecasting time series with varma recursions on graphs. *IEEE Transactions on Signal Processing*, 67(18):4870–4885, 2019.

[15] Elvin Isufi, Paolo Banelli, Paolo Di Lorenzo, and Geert Leus. Observing and tracking bandlimited graph processes from sampled measurements. *Signal Processing*, 177:107749, December 2020. ISSN 0165-1684. doi: 10.1016/j.sigpro.2020.107749.

[16] S.J. Julier and J.K. Uhlmann. Unscented Filtering and Nonlinear Estimation. *Proceedings of the IEEE*, 92(3):401–422, March 2004. ISSN 0018-9219. doi: 10.1109/JPROC.2003.823141.

[17] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, March 1960. ISSN 0021-9223. doi: 10.1115/1.3662552.

[18] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=SJiHXGWAZ.

[19] Henrique M. T. Menegaz, João Y. Ishihara, Geovany A. Borges, and Alessandro N. Vargas. A Systematization of the Unscented Kalman Filter Theory. *IEEE Transactions on Automatic Control*, 60(10):2583–2598, October 2015. ISSN 1558-2523. doi: 10.1109/TAC.2015.2404511.

[20] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[21] Liva Ralaivola and Florence d'Alché-Buc. Time series filtering, smoothing and learning using the kernel Kalman filter. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 3, pages 1449–1454. IEEE, 2005.

[22] Daniel Romero, Vassilis N. Ioannidis, and Georgios B. Giannakis. Kernel-Based Reconstruction of Space-Time Functions on Dynamic Graphs. *IEEE Journal of Selected Topics in Signal Processing*, 11(6):856–869, September 2017. ISSN 1941-0484. doi: 10.1109/JSTSP.2017.2726976.

[23] Luana Ruiz, Fernando Gama, and Alejandro Ribeiro. Gated graph recurrent neural networks. *IEEE Transactions on Signal Processing*, 68:6303–6318, 2020.

[24] Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. Structured sequence modeling with graph convolutional recurrent networks. In *International conference on neural information processing*, pages 362–373. Springer, 2018.

[25] Ling Shi. Kalman filtering over graphs: Theory and applications. *IEEE transactions on automatic control*, 54(9):2230–2234, 2009.

[26] Dan Simon. *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*. John Wiley & Sons, June 2006. ISBN 978-0-470-04533-6.

[27] Gerald L. Smith, Stanley F. Schmidt, Leonard A. McGee, and United States National Aeronautics and Space Administration. *Application of Statistical Filter Theory to the Optimal Estimation of Position and Velocity on Board a Circumlunar Vehicle*. National Aeronautics and Space Administration, 1962.

[28] Z Wu, S Pan, G Long, J Jiang, and C Zhang. Graph wavenet for deep spatial-temporal graph modeling. In *The 28th International Joint Conference on Artificial Intelligence (IJCAI)*. International Joint Conferences on Artificial Intelligence Organization, 2019.

[29] Daniele Zambon, Andrea Cini, Lorenzo Livi, and Cesare Alippi. Graph state-space models, 2023. URL https://arxiv.org/abs/2301.01741.

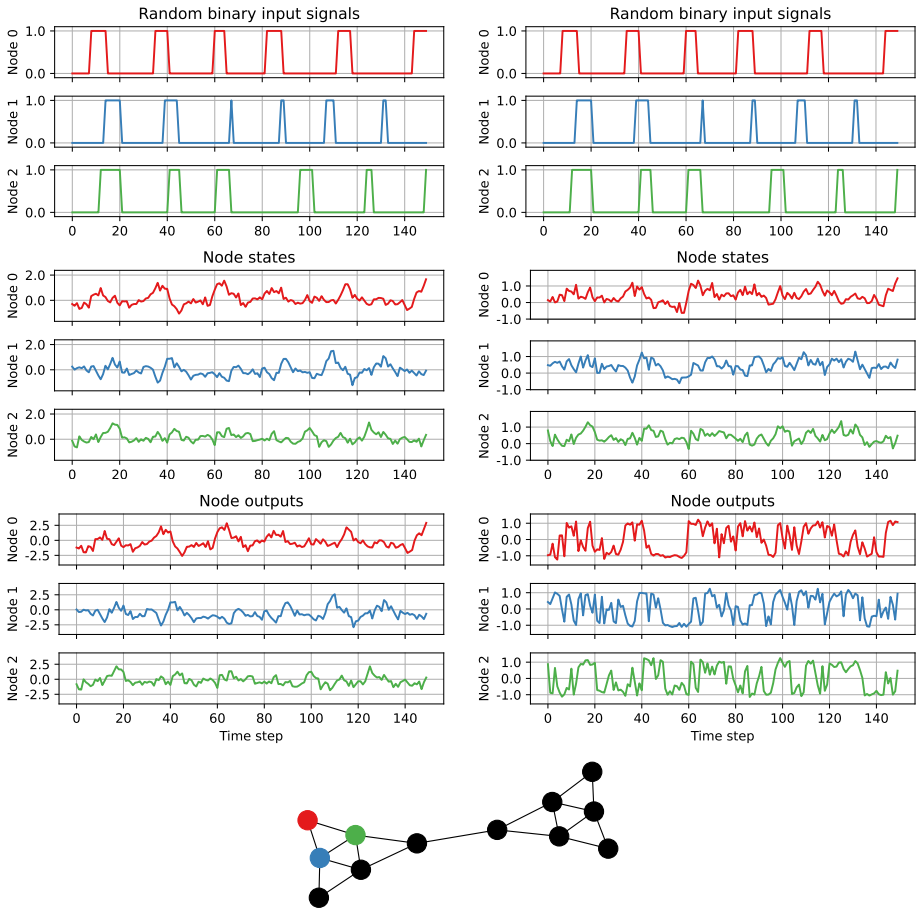

Figure 3: Inputs, states, and outputs of three nodes from LinGSS dataset (left) and NonLinGSS (right). The graph considered for generating the data is at the bottom, with node 0, 1 and 2 highlighted.

A Hardware and software

The code for the empirical evaluation of the proposed method has been developed in Python relying on the following open-source libraries: PyTorch [20], PyTorch Geometric [9], Torch Spatiotemporal [6], PyTorch Lightning [8] and NumPy [13]. The experiments were run on machines equipped with AMD EPYC 7513 processors and NVIDIA RTX A5000 GPUs.

B Model training

The models are trained to minimize the mean squared error (MSE) with Adam optimizer for 100 epochs and learning rate of 0.01. 70% of the data is used for training, 10% for validation, and 20% for testing. The batch size is set to 32 and the predictions are made from a window size of 12 time steps. Early stopping on the best validation MSE is applied with a 10-epoch patience. Training a model typically takes less than 10 minutes for Replica models and about 20 minutes for the STGNNs.

C Dataset visualization

Figure 3 shows examples of node-level inputs, states, and outputs of three nodes in LinGSS and NonLinGSS. The considered state graphs have 12 nodes connected as shown at the bottom of the figure.