State and Dynamics Estimation with the Kalman–Langevin filter

Martín Sevilla[†⋆], Nicolas Zilberstein[†⋆], Benjamin Cox[§], Sara Pérez-Vieites[§], Víctor Elvira[§], Santiago Segarra[†]
[†]Rice University, USA [§]University of Edinburgh, United Kingdom

***Abstract*—We present a novel sampling algorithm designed for the joint estimation of the states and a graph which partially models the state dynamics within a state-space representation. In particular, the dynamics are composed by the output of a graph filter whose input is a partially known adjacency matrix. Our algorithm learns the unknown part of the adjacency matrix. Our proposed method combines the classical Kalman filter for state estimation with annealed Langevin diffusion. This allows us to approximate the maximum a posteriori of the states and dynamics given observations by sampling from the posterior distribution. We exploit the prior information embedded in a dataset of graphs of varying sizes through the utilization of graph neural networks. We empirically demonstrate that integrating a learned prior distribution into the estimation process significantly improves estimation performance.**

***Index Terms*—Network topology inference, system identification, graph signal processing, diffusion models**

I. Introduction

State-space models (SSMs) find widespread applications in science and engineering by providing a statistical framework to describe dynamical systems. These models represent hidden states that change over time, with only partially observed information available through noisy observations. In their basic form, SSMs exhibit a Markovian dependency in the hidden states, and each observation depends on the current state, with conditional independence from the previous ones. Within this framework, the main objective is estimating the hidden state based on the available measurements.

When this estimation is approached from a probabilistic standpoint, it is referred to as Bayesian filtering. In the case of a linear-Gaussian SSM, the optimal estimator is the Kalman filter [1], computing exactly the posterior distribution of the state. For non-linear SSMs, alternative methods include the unscented Kalman filter (UKF) [2] and the particle filter (PF) [3], that approximate this distribution using a set of samples. However, it is essential to note that all these algorithms require knowledge of the model dynamics.

In this paper, we focus on network processes. Within this framework, the matrix that describes the linear-Gaussian SSMs under study corresponds to a graph filter [4], defined as a polynomial function of the network's adjacency matrix. This approach aligns with graphical modeling for time series [5], where the multidimensional state is composed of several unidimensional time series, with each one corresponding to a specific node in the underlying graph. The main goal is to estimate each of the nodal time series as well as the graph. The family of GraphEM algorithms [6]–[9] proposes a graphical interpretation of state-space models, connects them with Granger causality, solves the estimation problem when the dynamics are given by the adjacency matrix (i.e., the graph filter polynomial corresponds to the identity function), and enforces sparsity in the graph estimation.

In this study, we introduce an algorithm designed to incorporate additional prior information beyond graph sparsity and to utilize any graph filter in the dynamics. We leverage two sources of prior information. First, we make use of a dataset containing adjacency matrices of potentially varying sizes to estimate the prior distribution of the graph. This is particularly advantageous in real-world scenarios where datasets of graphs are more commonly available than closed-form prior distributions. Furthermore, we incorporate edge-to-edge constraints, allowing us to fix values in the adjacency matrix based on edges that are *known* to either exist or be absent. This feature is valuable when certain pairs of variables are known to be conditionally independent, but the goal is to estimate the remaining structure. Examples of such scenarios include gene expression data [10] and social networks [11].

Our algorithm combines the classical Kalman filter with Langevin dynamics, a Markov chain Monte Carlo (MCMC) sampler [12], [13]. By defining a stochastic dynamic process with a carefully designed stationary distribution, we can directly sample from the posterior distribution of states and graphs given the measurements.

Contributions. Our three main contributions are:

1) We extend the Kalman-Langevin filter (KLF), an algorithm to estimate both the states and the dynamics of a network process based on *sampling* from a posterior distribution.

2) We consider a graph transformer as backbone architecture instead of the permutation equivariant EDP-GNN in [14]employ annealed Langevin dynamics to implement this estimator, enabling us to incorporate an arbitrary prior distribution learned from data.

3) Through numerical experiments conducted on real-world graphs, we demonstrate that the incorporation of arbitrary prior distributions outperforms estimators that use no prior information or solely consider sparsity.

[⋆]These authors contributed equally to this work.

Research was sponsored by the Army Research Office and was accomplished under Grant Numbers W911NF-17-S-0002 and W911NF-22-1-0235. In addition, we acknowledge support from the 2022 University of Edinburgh - Rice University Strategic Collaboration Award.

Emails of corresponding authors: {msevilla, nzilberstein}@rice.edu.

II. Problem formulation

Let $\mathcal{G}$ be a graph of N nodes represented by its adjacency matrix $\mathbf{A}$. We work with unweighted and undirected graphs with no self-loops. Mathematically, these properties mean that $A_{ij} \in \{0,1\}$, $A_{ij} = A_{ji}$, and $A_{ii} = 0$, respectively. We assume to know some subset of the edges of $\mathcal{G}$, so that some entries A_{ij} are *known* to be either 0 or 1. One of our goals in this work is to estimate the missing edges. To distinguish between the known entries of $\mathbf{A}$ and the unknown ones, we define two sets of indices

$$\mathcal{O} = \{(i,j) : A_{ij} \text{ is observed} \wedge i < j\}, \tag{1}$$

$$\mathcal{U} = \{(i,j) : A_{ij} \text{ is unknown} \wedge i < j\}. \tag{2}$$

Throughout this work, we refer to the known and unknown fractions of the adjacency matrix as $\mathbf{A}^{\mathcal{O}}$ and $\mathbf{A}^{\mathcal{U}}$, respectively.

Let us consider the following linear-Gaussian SSM

$$\begin{cases} \mathbf{x}_{k+1} = h(\mathbf{A})\mathbf{x}_k + \mathbf{w}_k \\ \mathbf{y}_k = \mathbf{C}\mathbf{x}_k + \mathbf{v}_k \end{cases}, \tag{3}$$

where $k = 1, \dots, K$ denotes discrete time, $\mathbf{x}_k \in \mathbb{R}^N$ is the hidden state at time k, $\mathbf{y}_k \in \mathbb{R}^D$ is the corresponding observation, and the state and observation noises are $\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{W})$ and $\mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{V})$, respectively. The dynamics of (3) are given by $h_{\boldsymbol{\theta}}(\mathbf{A}) \in \mathbb{R}^{N \times N}$, which stands for a known graph filter [4] defined on the underlying graph $\mathcal{G}$, such that $h_{\boldsymbol{\theta}} : \{0,1\}^{N \times N} \to \mathbb{R}^{N \times N}$, where $\boldsymbol{\theta} \in \mathbb{R}^{N_\theta}$ is a vector of known parameters (see Section V for examples). The latent process is initialized with $\mathbf{x}_0 \sim \mathcal{N}(\bar{\mathbf{x}}_0, \mathbf{P}_0)$, with both $\bar{\mathbf{x}}_0$ and $\mathbf{P}_0$ known. The hidden states $\mathbf{x}_k$ can be considered, under our perspective, a *graph signal* – they describe the state of each node in $\mathcal{G}$. We do not have access to the actual states, but rather to the noisy measurements $\mathbf{y}_k$, with $\mathbf{C}$ assumed to be known.

Our goal is to jointly estimate $\mathbf{A}$ and $\mathbf{x}_{1:K} = \{\mathbf{x}_k\}_{k=1}^K$ while incorporating prior information into our estimator. Usually, sparsity is the only prior knowledge that is used when estimating the edges of a graph [6], [15]. Our objective is, however, to be able to leverage *any* available prior information.

Consider that we have access to a set of adjacency matrices denoted as $\mathcal{A}$, where each matrix corresponds to a distinct graph and, importantly, the prior distribution $p(\mathbf{A})$ of each one of them matches that of the graph we intend to estimate. In this setting, our problem is defined as follows:

Problem 1: *Given the measurements $\{\mathbf{y}_k\}_{k=1}^K$, a partially known adjacency matrix $\mathbf{A}^{\mathcal{O}}$, and prior knowledge given by a set of adjacency matrices $\mathcal{A}$, find an estimate of $\mathbf{A}^{\mathcal{U}}$ and $\{\mathbf{x}_k\}_{k=1}^K$ knowing that the dynamics are given by* (3)*.*

A natural way of solving Problem 1 is to compute the maximum *a posteriori* (MAP):

$$\begin{aligned} \widehat{\mathbf{A}}_{\text{MAP}}, \hat{\mathbf{x}}_{1:K_{\text{MAP}}} &= \underset{\mathbf{A}, \mathbf{x}_{1:K}}{\operatorname{argmax}} \quad p(\mathbf{A}, \mathbf{x}_{1:K} \,|\, \mathbf{y}_{1:K}) \\ &= \underset{\mathbf{A}, \mathbf{x}_{1:K}}{\operatorname{argmax}} \quad p(\mathbf{x}_{1:K}, \mathbf{y}_{1:K} \mid \mathbf{A}) p(\mathbf{A}) \\ &\quad \text{subject to } A_{ij} = A_{ij}^{\mathcal{O}}, \ \forall (i,j) \in \mathcal{O}. \end{aligned} \tag{4}$$

Such an approach is not feasible, since a closed-form expression for $p(\mathbf{A})$ is not available. Moreover, as we aim to solve Problem 1 for a *generic* graph filter $h_{\boldsymbol{\theta}}$, the expression $p(\mathbf{x}_{1:K}, \mathbf{y}_{1:K} \mid \mathbf{A})$ may not even be tractable from an optimization standpoint, since $h_{\boldsymbol{\theta}}$ could have any non-convex functional form.

A main challenge in solving Problem 1 is how to extract the prior information in $\mathcal{A}$ and use it in our estimation. We propose to replace the optimization formulation of (4) with a sampling approach. Namely, our method is based on *drawing a sample* from the posterior distribution. It is worth emphasizing that our approach does not depend on having a closed-form expression for $p(\mathbf{A})$. While the sampled $\mathbf{A}^{\mathcal{U}}$ and $\mathbf{x}_{1:K}$ may not equal the MAP, we draw samples that attain large values for the objective in (4) with high probability.

III. Background

A. Langevin diffusion

The Langevin diffusion is a particular type of continuous-time diffusion process [16]. For practical applications, we need a discrete-time version. Applying an Euler-Maruyama discretization to the original dynamics leads to the unadjusted Langevin algorithm (ULA), which is an MCMC algorithm [13] described by

$$\mathbf{z}_{t+1} = \mathbf{z}_t + \epsilon \nabla_{\mathbf{z}_t} \log p(\mathbf{z}_t) + \sqrt{2\epsilon\tau}\, \mathbf{b}_t, \tag{5}$$

where τ is known as the *temperature* parameter, t is an iteration index, ϵ is the discretization step size, and $\mathbf{b}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$.

Under some regularity conditions, the distribution of $\mathbf{z}_t$ converges to $\pi(\mathbf{z}) \propto p(\mathbf{z})^{1/\tau}$ when $\epsilon \to 0$ and $t \to \infty$. In essence, ULA generates samples from a target distribution $p(\mathbf{z})$ by iteratively moving in the direction of the gradient of the logarithm of the target density (i.e., the *score function*), and introducing noise to avoid local maxima. Although this result is asymptotic, non-asymptotic convergence results have been obtained under some conditions on the target distribution [17].

B. Annealed Langevin diffusion

Our primary objective is to extract samples from a *discrete* random variable $\mathbf{z} \in \{0,1\}^d$ (recall that $A_{ij} \in \{0,1\}$). Since the score function $\nabla_{\mathbf{z}} \log p(\mathbf{z})$ is not defined for a discrete random vector, we cannot sample using (5). To address this issue, we introduce the notion of *annealed* Langevin dynamics [18]–[22].

Let $\{\sigma_l\}_{l=1}^L$ be a sequence of *noise levels* such that $\sigma_1 > \sigma_2 > \cdots > \sigma_L > 0$. Then, for a given noise level, we define a perturbed version of the original random variable $\mathbf{z}$ in (5),

$$\tilde{\mathbf{z}}_t = \mathbf{z} + \mathbf{n}_t, \tag{6}$$

where $\mathbf{n}_t \sim \mathcal{N}\left(\mathbf{0}, \sigma_{l(t)}^2 \mathbf{I}_d\right)$. Then, the iterative process that describes the annealed Langevin dynamics is given by

$$\tilde{\mathbf{z}}_{t+1} = \tilde{\mathbf{z}}_t + \alpha_t \nabla_{\tilde{\mathbf{z}}_t} \log p(\tilde{\mathbf{z}}_t) + \sqrt{2\alpha_t\tau}\, \mathbf{z}_t, \tag{7}$$

where $\alpha_t = \epsilon \cdot \sigma_{l(t)}^2 / \sigma_L^2$. The final sample can be obtained by projecting $\tilde{\mathbf{z}}_t$ onto the domain of $\mathbf{z}$ at the end of the process.

Algorithm 1 Kalman filter

Require: $\bar{\mathbf{x}}_0, \mathbf{P}_{0|0}, h_{\boldsymbol{\theta}}(\mathbf{A}), \mathbf{C}, \mathbf{V}, \mathbf{W}, \mathbf{y}_{1:K}$
  Sample $\hat{\mathbf{x}}_{0|0} \sim \mathcal{N}(\bar{\mathbf{x}}_0, \mathbf{P}_{0|0})$
  for $k = 1$ to K **do**
    $\hat{\mathbf{x}}_{k|k-1} = h_{\boldsymbol{\theta}}(\mathbf{A})\hat{\mathbf{x}}_{k-1|k-1}$
    $\hat{\mathbf{P}}_{k|k-1} = h_{\boldsymbol{\theta}}(\mathbf{A})\hat{\mathbf{P}}_{k-1|k-1}h_{\boldsymbol{\theta}}(\mathbf{A})^\top + \mathbf{W}$
    $\tilde{\mathbf{r}}_k = \mathbf{y}_k - \mathbf{C}\hat{\mathbf{x}}_{k|k-1}$
    $\mathbf{F}_k = \mathbf{C}\hat{\mathbf{P}}_{k|k-1}\mathbf{C}^\top + \mathbf{V}$
    $\mathbf{K}_k = \hat{\mathbf{P}}_{k|k-1}\mathbf{C}^\top\mathbf{F}_k^{-1}$
    $\hat{\mathbf{P}}_{k|k} = (\mathbf{I}_N - \mathbf{K}_k\mathbf{C})\hat{\mathbf{P}}_{k|k-1}$
    $\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k\tilde{\mathbf{r}}_k$
  end for
  return $\hat{\mathbf{x}}_{1:K}$

Not only does the annealed dynamics improve the algorithm's overall performance [18], but it also renders the target distribution differentiable, since $\tilde{\mathbf{z}}_t$ is differentiable as defined in (6). By adequately choosing the sequence $\{\sigma_l\}_{l=1}^L$ and the step size ϵ, after a sufficient number of iterations, the continuous sample $\tilde{\mathbf{z}}_t$ gradually approaches an authentic sample from the discrete distribution $\pi(\mathbf{z}) \propto p(\mathbf{z})^{1/\tau}$.

Each noise level σ_l is a predefined hyperparameter. The initial levels must be sufficiently high to facilitate thorough exploration of the entire space, while the final levels should be much smaller so that the target distribution converges to the discrete one. Furthermore, setting $\tau < 1$ allows to accentuate the peaks in $p(\mathbf{z})$ within the landscape of the annealed continuous distribution.

C. Kalman filter

Given a dynamical system represented by (3), the objective is to estimate the hidden variables $\mathbf{x}_{1:K}$ given a set of observations $\mathbf{y}_{1:K}$. We consider the conditional expectation as our estimator

$$\hat{\mathbf{x}}_k = \mathbb{E}[\mathbf{x}_k|\mathbf{y}_{1:k}] = \int \mathbf{x}_k \; p(\mathbf{x}_k|\mathbf{y}_{1:k})\mathrm{d}\mathbf{x}_k. \tag{8}$$

In the context of Bayesian filtering, the posterior $p(\mathbf{x}_k|\mathbf{y}_{1:k})$ is computed recursively by alternating between two steps: a *prediction step* that propagates the state distribution of the previous update step following the dynamic model, and an *update step*, that combines the prediction and the likelihood associated to the new observation to compute the new state estimation [1]. Formally, given an initial prior distribution $p(\mathbf{x}_0)$, the prediction and update densities are computed as

$$p(\mathbf{x}_k|\mathbf{y}_{1:k-1}) = \int p(\mathbf{x}_k|\mathbf{x}_{k-1})p(\mathbf{x}_{k-1}|\mathbf{y}_{1:k-1})\mathrm{d}\mathbf{x}_{k-1}, \tag{9}$$

$$p(\mathbf{x}_k|\mathbf{y}_{1:k}) \propto p(\mathbf{y}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{y}_{1:k-1}), \tag{10}$$

where $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ is the dynamic model and $p(\mathbf{y}_k|\mathbf{x}_k)$ is the measurement model that describes the system in (3). When considering a linear-Gaussian SSM, and when $h_{\boldsymbol{\theta}}(\mathbf{A})$, $\mathbf{C}$, $\mathbf{W}$, and $\mathbf{V}$ are known, we get a closed-form solution to (8) given by the *Kalman filter* [23]. Specifically, we have that $p(\mathbf{x}_k|\mathbf{y}_{1:k-1}) = \mathcal{N}(\mathbf{x}_k; \hat{\mathbf{x}}_{k|k-1}, \hat{\mathbf{P}}_{k|k-1})$ and $p(\mathbf{x}_k|\mathbf{y}_{1:k}) = \mathcal{N}(\mathbf{x}_k; \hat{\mathbf{x}}_{k|k}, \hat{\mathbf{P}}_{k|k})$, where $\hat{\mathbf{x}}_{k|k}$ is the state estimation given by (8) at time k, and $\hat{\mathbf{P}}_{k|k}$ the associated covariance matrix. The equations of the Kalman filter for our SSM in (3) are presented in Algorithm 1, since this is an integral part of our proposed method.

IV. Proposed method

Recall our aim is to solve Problem 1 by sampling from the joint posterior $p(\mathbf{A}, \mathbf{x}_{1:K} \,|\, \mathbf{y}_{1:K})$. In this endeavor, we consider the factorization in (4). Notice that in the case of a SSM like (3), the density $p(\mathbf{x}_{1:K}, \mathbf{y}_{1:K}|\mathbf{A})$ is given by a product of Gaussian distributions. Hence, akin to the particle marginal Metropolis–Hastings (PMMH) sampler [24], we propose to leverage this closed-form solution to jointly generate samples $\{\mathbf{A}, \mathbf{x}_{1:K}\}$ by alternating between the Langevin diffusion to sample candidates $\mathbf{A}$, and a Kalman filter to compute a sample candidate of the states $\mathbf{x}_{1:K}$. This also connects with the probabilistic version of the GraphEM family of algorithms [25], where reversible jump MCMC is used instead. In Section IV-A, we first present how to leverage the annealed Langevin diffusion to sample graphs from the joint posterior. Then, in Section IV-B, we explain how to exploit the prior information given by $\mathcal{A}$. Last, in Section IV-C, we describe our proposed algorithm.

A. Factorized posterior

Our goal is to draw samples from

$$p(\mathbf{A}, \mathbf{x}_{1:K} \mid \mathbf{y}_{1:K}) = \frac{p(\mathbf{A})p(\mathbf{x}_0)}{p(\mathbf{y}_{1:K})} \prod_{k=1}^{K} p(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \mathbf{A}) \times \prod_{k=1}^{K} p(\mathbf{y}_k \mid \mathbf{x}_k). \tag{11}$$

The *only* requirement to sample from (11) by using (7) is to know the score function of the target distribution. Since the variable $\mathbf{A}$ is discrete, we cannot compute the score function of (11) with respect to $\mathbf{A}$. This is where the annealing technique described in Section III-B comes into play.

In a slight abuse of notation, we use the half-vectorization $\mathbf{a} = \mathrm{vech}(\mathbf{A})$ and $\mathbf{A}$ interchangeably. Analogously, we use $\tilde{\mathbf{a}}$ instead of $\widetilde{\mathbf{A}}$. Taking the logarithm of the annealed counterpart of (11), we get

$$\log p(\tilde{\mathbf{a}}, \mathbf{x}_{1:K} \mid \mathbf{y}_{1:K}) = M + \log p(\tilde{\mathbf{a}}) + \sum_{k=1}^{K} \log p(\mathbf{y}_k \mid \mathbf{x}_k) + \sum_{k=1}^{K} \log p(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \tilde{\mathbf{a}}), \tag{12}$$

where M groups the constant terms. Before using (7) to draw samples, we need to compute the score function of (12) with respect to $\tilde{\mathbf{a}}$:

$$\nabla_{\tilde{\mathbf{a}}} \log p(\tilde{\mathbf{a}}, \mathbf{x}_{1:K} \mid \mathbf{y}_{1:K}) = \nabla_{\tilde{\mathbf{a}}} \log p(\tilde{\mathbf{a}}) + \sum_{k=1}^{K} \nabla_{\tilde{\mathbf{a}}} \log p(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \tilde{\mathbf{a}}). \tag{13}$$

The conditional transition probabilities are given by

$$\log p(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \tilde{\mathbf{a}}) = -\frac{1}{2}\left(\mathbf{x}_k - h_{\boldsymbol{\theta}}(\widetilde{\mathbf{A}})\mathbf{x}_{k-1}\right)^{\top} \mathbf{Q}^{-1} \left(\mathbf{x}_k - h_{\boldsymbol{\theta}}(\widetilde{\mathbf{A}})\mathbf{x}_{k-1}\right) + C, \tag{14}$$

where C does not depend on $\mathbf{x}_k$. Regardless of the functional form of $h_{\boldsymbol{\theta}}(\cdot)$, as long as it is differentiable, we are always able to compute $\nabla_{\tilde{\mathbf{a}}} \log p(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \tilde{\mathbf{a}})$ via automatic differentiation [26]. On the contrary, calculating the first term in (13) poses a notably more intricate challenge. The density $p(\tilde{\mathbf{a}})$ corresponds to a discretely distributed vector that is perturbed by white Gaussian noise. Moreover, in our specific scenario, we even lack an explicit expression for $p(\mathbf{a})$ and solely rely on the collection of graphs $\mathcal{A}$. Fortunately, we can tackle this issue by developing an estimator to *approximate* $\nabla_{\tilde{\mathbf{a}}} \log p(\tilde{\mathbf{a}})$. We achieve this through the use of neural networks [18], [21].

B. Learned graph priors

It is important to emphasize that our primary objective is to estimate $\nabla \log p(\tilde{\mathbf{a}})$ in order to enable sampling from (11) by employing the diffusion process defined in (7). We define the output of the graph neural network (GNN) we intend to train as $\mathbf{s}_{\boldsymbol{\xi}}(\tilde{\mathbf{a}}, \sigma)$, where $\boldsymbol{\xi}$ represents its trainable parameters. Ideally, the network's output for a given $\tilde{\mathbf{a}}$ (corresponding to the current noise level σ_l for the ongoing iteration) should closely resemble the true score function $\nabla \log p(\tilde{\mathbf{a}})$.

For a single noise level σ_l, the GNN output should approximate the desired score function, such that

$$\mathcal{D}\left(\tilde{\mathbf{a}}|\boldsymbol{\xi}, \sigma_l\right) = \left\|\mathbf{s}_{\boldsymbol{\xi}}(\tilde{\mathbf{a}}, \sigma_l) - \nabla \log p(\tilde{\mathbf{a}})\right\|_2^2 \tag{15}$$

is minimized. The issue is that the loss function cannot be dependent on $\nabla \log p(\tilde{\mathbf{a}})$, as this is the function we are aiming to learn. An alternative conditional loss can be defined as

$$\mathcal{D}_c\left(\tilde{\mathbf{a}}|\boldsymbol{\xi}, \sigma_l\right) = \left\|\mathbf{s}_{\boldsymbol{\xi}}(\tilde{\mathbf{a}}, \sigma_l) - \nabla \log p(\tilde{\mathbf{a}}|\mathbf{a})\right\|_2^2, \tag{16}$$

since $\nabla \log p(\tilde{\mathbf{a}}|\mathbf{a}) = (\mathbf{a} - \tilde{\mathbf{a}})/\sigma_l^2$ is known during training: $\mathbf{a}$ is one element of $\mathcal{A}$, and both $\tilde{\mathbf{a}}$ and σ_l are the GNN inputs. Namely, $\tilde{\mathbf{a}}$ is easily generated using $\mathbf{a}$ and noise of variance σ_l^2. The values for $\boldsymbol{\xi}$ that optimize (16) also optimize (15) since they are equivalent from an optimization standpoint, as shown in [27]. In order to jointly minimize the MSE across all noise levels, we train the GNN by minimizing

$$\mathcal{J}\left(\boldsymbol{\xi}|\{\sigma_l\}_{l=1}^L\right) = \frac{1}{2L}\sum_{l=1}^{L} \sigma_l^2 \mathbb{E}\left[\mathcal{D}_c\left(\tilde{\mathbf{a}}|\boldsymbol{\xi}, \sigma_l\right)\right], \tag{17}$$

where the weights given by σ_l^2 are used so that every term in the summation presents the same order of magnitude [18]. In this study, we use the EDP-GNN architecture [14] to minimize (17).

C. The KLF algorithm

We propose to jointly estimate $\mathbf{A}$ and $\mathbf{x}_{1:K}$ by running one step of the Langevin dynamics for the former together with a Kalman filter for the latter, as described in Algorithm 2. Notice that the score estimator $\mathbf{s}(\cdot)$ is an input, meaning that an already trained GNN is required to run the KLF. The output of this GNN should correctly approximate the prior score, so that $\mathbf{s}\left(\tilde{\mathbf{a}}, \sigma_l\right) \simeq \nabla_{\tilde{\mathbf{a}}} \log p\left(\tilde{\mathbf{a}}\right)$.

Algorithm 2 Kalman–Langevin filter

Require: $\mathbf{y}_{1:K}$, $\mathbf{A}^{\mathcal{O}}$, $h_{\boldsymbol{\theta}}(\cdot)$, $p(\mathbf{x}_0)$, $\mathbf{s}(\cdot)$, $\{\sigma_l\}_{l=1}^L$, T, ϵ, τ

1: Initialize $\widetilde{\mathbf{A}}$ at random
2: $\widetilde{\mathbf{A}}^{\mathcal{O}} \leftarrow \mathbf{A}^{\mathcal{O}}$ ▷ Fix the known values
3: **for** $l \leftarrow 1$ to L **do**
4: $\quad \alpha_l \leftarrow \epsilon \cdot \sigma_l^2/\sigma_L^2$
5: $\quad$ Compute $\hat{\mathbf{x}}_{1:K}$ with Algorithm 1 using $h_{\boldsymbol{\theta}}(\widetilde{\mathbf{A}})$
6: $\quad$ **for** $t \leftarrow 1$ to T **do**
7: $\quad\quad$ Draw $\mathbf{n} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{I}_{|\mathcal{U}|}\right)$
8: $\quad\quad$ Compute $\sum_{k=1}^{K} \nabla_{\tilde{\mathbf{a}}} \log p(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \tilde{\mathbf{a}})$ using (14)
9: $\quad\quad$ Compute $\mathbf{s}(\tilde{\mathbf{a}}, \sigma_l)$ using the trained GNN
10: $\quad\quad$ $\Delta \leftarrow \sum_{k=1}^{K} \nabla_{\tilde{\mathbf{a}}} \log p(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \tilde{\mathbf{a}}) + \mathbf{s}(\tilde{\mathbf{a}}, \sigma_l)$
11: $\quad\quad$ $\tilde{\mathbf{a}}^{\mathcal{U}} \leftarrow \tilde{\mathbf{a}}^{\mathcal{U}} + \alpha_l \Delta^{\mathcal{U}} + \sqrt{2\alpha_l\tau}\mathbf{n}$
12: $\quad$ **end for**
13: **end for**
14: $\widehat{\mathbf{A}} \leftarrow \mathbb{I}\left\{\widetilde{\mathbf{A}} \geq 0.5\right\}$ ▷ Threshold each value in $\widetilde{\mathbf{A}}$
15: **return** $\widehat{\mathbf{A}}$, $\hat{\mathbf{x}}_{1:K}$

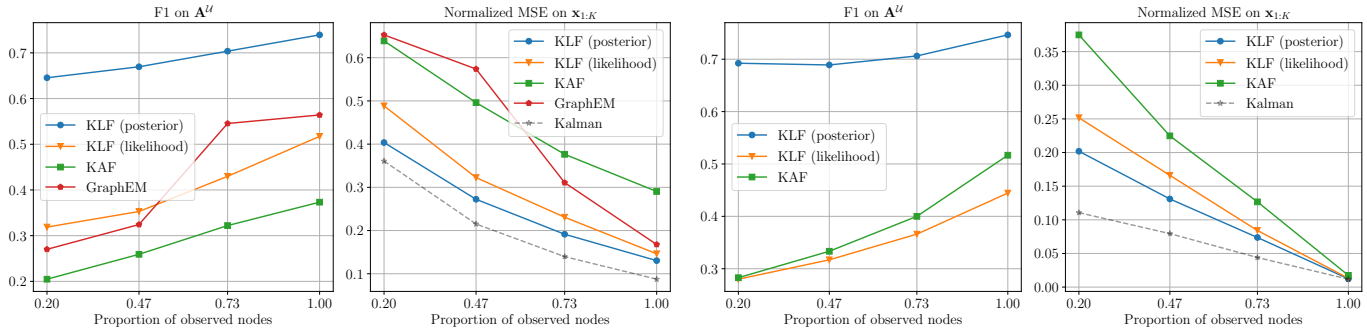
For each of the L noise levels, we update $\hat{\mathbf{x}}_{1:K}$ once at the beginning of the Langevin estimation for that level (see line 5). The choice of not running a Kalman filter for each Langevin iteration t is due to computational reasons, since doing so could become too computationally expensive for large values of K. After updating the state estimates, we run Langevin on a whole noise level for T iterations. After LT Langevin steps (and, therefore, L Kalman updates), the algorithm still yields a continuous matrix $\widetilde{\mathbf{A}}$, which can be thresholded to obtain a binary-valued matrix.

V. Numerical results

We run experiments using ego-nets that correspond to real networks from the music streaming service Deezer [28]. We use the graphs from the original dataset such that $N \leq 25$, selecting $|\mathcal{A}| = 2926$ to train the EDP-GNN and the remaining 100 for testing. We assume 50% of the values in each $\mathbf{a}$ to be unknown. Regarding the KLF hyperparameters, we use $L = 10$ noise levels, evenly spaced between $\sigma_1 = 0.5$ and $\sigma_L = 0.03$, and $T = 300$ steps per level. We set the step size at $\epsilon = 10^{-6}$ and the temperature at $\tau = 0.5$.

Normalizing filter. The first graph filter used to generate the signals is $h_\theta(\mathbf{A}) = \frac{1}{\theta}\mathbf{A}$, where θ is a known constant that we fix at the maximum eigenvalue of the true (unknown) $\mathbf{A}$. We use $K = 10$, $\mathbf{W} = 0.1\mathbf{I}_N$ and $\mathbf{V} = 10\mathbf{I}_D$. We vary the measurement matrix $\mathbf{C}$ so that, in each experiment, different numbers of nodes are observed, but the matrix is such that it has a 1 in each row and 0 everywhere else. The results of the simulations are shown in Fig. 1a.

Heat diffusion filter. The second graph filter we used is $h_\theta(\mathbf{A}) = \exp(-\theta\mathbf{L})$, where $\mathbf{L} = \mathrm{diag}(\mathbf{A1}) - \mathbf{A}$ is the graph Laplacian. For every simulation, the coefficient θ is generated independently as $\theta \sim \mathrm{Unif}[0.01, 0.05]$. We set $K = 100$, and the matrices $\mathbf{W}$, $\mathbf{V}$, and $\mathbf{C}$ are defined as in the previous experiment. The results of the simulations are shown in Fig. 1b.



(a) Normalizing filter

(b) Heat diffusion filter

Fig. 1: Estimation performance of our method (blue curves) compared to other benchmarks. The orange curves correspond to the KLF using only the likelihood (i.e., Algorithm 2 but setting $\mathbf{s}(\tilde{\mathbf{a}}, \sigma_l) = 0$ in line 10). The Kalman-Adam filter (KAF) tries to find the maximum likelihood estimator by optimizing (14) (i.e., it also ignores prior information, but uses optimization instead of sampling). In the case of Fig. 1a, the dynamics of the SSM are given by the graph itself, and thus, we can compare our results to GraphEM [6]. The dashed curves correspond to the Kalman estimate of the states if the dynamics were completely known and are just plotted to provide a lower bound on the error. The normalized MSE for the trajectories is computed as $\|\mathbf{x}_{1:K} - \hat{\mathbf{x}}_{1:K}\|_2^2 \,/\, \|\mathbf{x}_{1:K}\|_2^2$. The reported metrics correspond to the median over 100 simulations.

In both test cases, the estimation performance of all methods improves as the number of observed nodes increases. We observe that incorporating the prior information encoded in the dataset $\mathcal{A}$ through our proposed Kalman-Langevin filter improves the estimation performance of both $\mathbf{A}^{\mathcal{U}}$ and $\mathbf{x}_{1:K}$.

VI. Conclusions

We have introduced an algorithm that combines the classical Kalman filter with annealed Langevin dynamics for state and dynamics inference, incorporating prior knowledge about graph distributions. This knowledge is exploited through training GNNs on diverse graphs, irrespective of their size or prior distribution. Our approach offers a sample from the posterior distribution as an estimator, which is considered to be an approximate solution to the MAP estimation problem. Experiments conducted on real-world networks have demonstrated that the integration of observed noisy graph signals with prior knowledge yields superior results compared to using only observed data.

References

[1] S. Särkkä and L. Svensson, *Bayesian Filtering and Smoothing*, vol. 17, Cambridge university press, 2023.

[2] S. J. Julier and J. K. Uhlmann, "Unscented filtering and nonlinear estimation," *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401–422, 2004.

[3] P. M. Djuric, M. F. Bugallo, and J. Míguez, "Density assisted particle filters for state and parameter estimation," in *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE, 2004, vol. 2, pp. ii–701.

[4] E. Isufi, F. Gama, D. I. Shuman, and S. Segarra, "Graph filters for signal processing and machine learning on graphs," *arXiv preprint arXiv:2211.08854*, 2022.

[5] M. Eichler, "Graphical modelling of multivariate time series," *Probability Theory and Related Fields*, vol. 153, pp. 233–268, 2012.

[6] E. Chouzenoux and V. Elvira, "GraphEM: EM algorithm for blind kalman filtering under graphical sparsity constraints," in *IEEE Intl. Conf. Acoust., Speech and Signal Process. (ICASSP)*. IEEE, 2020, pp. 5840–5844.

[7] V. Elvira and É. Chouzenoux, "Graphical inference in linear-gaussian state-space models," *IEEE Transactions on Signal Processing*, vol. 70, pp. 4757–4771, 2022.

[8] E. Chouzenoux and V. Elvira, "Sparse graphical linear dynamical systems," *arXiv preprint arXiv:2307.03210*, 2023.

[9] M. Greiff, S. Di Cairano, H. Mansour, and K. Berntorp, "A generalized GraphEM for sparse time-varying dynamical systems," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 5957–5962, 2023.

[10] Y. Li and S. Jackson, "Gene network reconstruction by integration of biological prior knowledge," *G3-Genes Genomes Genetics*, vol. 5, pp. 1075–1079, 2015.

[11] Q. Wu, Z. Zhang, J. Waltz, T. Ma, D. Milton, and S. Chen, "Predicting latent links from incomplete network data using exponential random graph model with outcome misclassification," *bioRxiv*, 2019.

[12] C. Robert and G. Casella, *Monte Carlo Statistical Method*, Springer, 1999.

[13] G. O. Roberts and R. L. Tweedie, "Exponential convergence of Langevin distributions and their discrete approximations," *Bernoulli*, vol. 2, pp. 341–363, 1996.

[14] C. Niu, Y. Song, J. Song, S. Zhao, A. Grover, and S. Ermon, "Permutation invariant graph generation via score-based generative modeling," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 4474–4484.

[15] G. Mateos, S. Segarra, A. G. Marques, and A. Ribeiro, "Connecting the dots: Identifying network structure via graph signal processing," *IEEE Signal Process. Mag.*, vol. 36, no. 3, pp. 16–43, 2019.

[16] G. A. Pavliotis, *Stochastic Processes and Applications: Diffusion Processes, the Fokker-Planck and Langevin Equations*, Springer, 2014.

[17] A. S. Dalalyan and A. Karagulyan, "User-friendly guarantees for the Langevin Monte Carlo with inaccurate gradient," *Stoch. Process. Their Appl.*, vol. 129, no. 12, pp. 5278–5311, 2019.

[18] Y. Song and S. Ermon, "Generative modeling by estimating gradients of the data distribution," *arXiv preprint arXiv:1907.05600*, 2020.

[19] B. Kawar, G. Vaksman, and M. Elad, "SNIPS: Solving noisy inverse problems stochastically," *arXiv preprint arXiv:2105.14951*, 2021.

[20] N. Zilberstein, C. Dick, R. Doost-Mohammady, A. Sabharwal, and S. Segarra, "Annealed Langevin dynamics for massive MIMO detection," *IEEE Trans. Wireless Commun.*, vol. 22, no. 6, 2023 (Online Nov 2022).

[21] M. Sevilla and S. Segarra, "Bayesian topology inference on partially known networks from input-output pairs," *arXiv preprint arXiv:2309.06532*, 2023.

[22] N. Zilberstein, A. Sabharwal, and S. Segarra, "Solving linear inverse problems using higher-order annealed Langevin diffusion," *arXiv preprint arXiv:2305.05014*, 2023.

[23] B. D. O. Anderson and J. B. Moore, *Optimal Filtering*, Prentice Hall, 1979.

[24] C. Andrieu, A. Doucet, and R. Holenstein, "Particle Markov chain Monte Carlo methods (with discussion)," *Journal of the Royal Statistical Society, Series B*, vol. 72, pp. 1–33, 01 2010.

[25] B. Cox and V. Elvira, "Sparse bayesian estimation of parameters in linear-gaussian state-space models," *IEEE Transactions on Signal Processing*, 2023.

[26] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind,

"Automatic differentiation in machine learning: a survey," *Journal of Marchine Learning Research*, vol. 18, pp. 1–43, 2018.

[27] P. Vincent, "A connection between score matching and denoising autoencoders," *Neural Comput.*, vol. 23, no. 7, pp. 1661–1674, 2011.

[28] B. Rozemberczki, O. Kiss, and R. Sarkar, "Karate Club: an API oriented open-source Python framework for unsupervised learning on graphs," in *ACM International Conference on Information and Knowledge Management*. ACM, 2020, p. 3125–3132.