Received XX September 2021; revised ; accepted XX October 2021; Date of publication XX November 2021; date of current version XX November 2021.

Digital Object Identifier 10.1109/OJCSYS.2021.Doi Number

State and Sparse Input Estimation in Linear Dynamical Systems using Low-Dimensional Measurements

RUPAM KALYAN CHAKRABORTY[1], GEETHU JOSEPH[1] (Senior Member, IEEE), CHANDRA R. MURTHY[2] (Fellow, IEEE)

[1]Faculty of Electrical Engineering, Mathematics, and Computer Science, Delft University of Technology, Delft 2628 CD, The Netherlands; emails: {R.K.Chakraborty,G.Joseph}@tudelft.nl

[2]Department of Electrical Communication Engineering, Indian Institute of Science (IISc), Bangalore 560012, India; email: cmurthy@iisc.ac.in

R.K. Chakraborty's work was initiated while at IISc, Bangalore. This work was presented in part in [1].

ABSTRACT Sparsity constraints on the control inputs of a linear dynamical system naturally arise in several practical applications such as networked control, computer vision, seismic signal processing, and cyber-physical systems. In this work, we consider the problem of jointly estimating the states and sparse inputs of such systems from low-dimensional (compressive) measurements. Due to the low-dimensional measurements, conventional Kalman filtering and smoothing algorithms fail to accurately estimate the states and inputs. We present a Bayesian approach that exploits the input sparsity to significantly improve estimation accuracy. Sparsity in the input estimates is promoted by using different prior distributions on the input. We investigate two main approaches: regularizer-based maximum a posteriori estimation and Bayesian learning-based estimation. We also extend the approaches to handle control inputs with common support and analyze the time and memory complexities of the presented algorithms. Finally, using numerical simulations, we show that our algorithms outperform the state-of-the-art methods in terms of accuracy and time/memory complexities, especially in the low-dimensional measurement regime.

INDEX TERMS Kalman filtering and smoothing, sparsity-promoting regularizer, joint sparsity, ADMM, ℓ_1 minimization, reweighted ℓ_2 minimization, sparse Bayesian learning, variational Bayesian methods.

I. Introduction

Sparse actuator control of linear dynamical systems (LDSs) has recently gained considerable interest in the literature [2]–[5]. This new research area deals with LDSs with sparsity constraints on the control inputs, i.e., each input vector contains only a small number of nonzero entries. An LDS with sparse control inputs models several practical applications such as networked control systems [2], [3], opinion dynamics manipulation [6], [7], computer vision [8], seismic signal processing [9], [10], and cyber-physical systems [11], [12]. In such applications, an important goal is to jointly estimate the states and sparse inputs of the LDS from its measurements or output. For example, malicious attacks on cyber-physical systems can be modeled as sparse inputs [11], [12]. Recovery of these sparse attacks is crucial to detecting and mitigating the attacks, as demonstrated in applications such as aircraft engine [13]–[15] and power system network [16].

Motivated by the above applications, this paper focuses on developing *joint state and input recovery algorithms* for observable LDSs with sparse control inputs. Specifically, we consider a discrete-time LDS, with state transition matrix $\boldsymbol{A}_k \in \mathbb{R}^{n\times n}$, input matrix $\boldsymbol{B}_k \in \mathbb{R}^{n\times m}$, measurement matrices $\boldsymbol{C}_k \in \mathbb{R}^{p\times n}$ and $\boldsymbol{D}_k \in \mathbb{R}^{p\times m}$ at time instant k, whose dynamics are governed by

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}_k\boldsymbol{x}_k + \boldsymbol{B}_k\boldsymbol{u}_k + \boldsymbol{w}_k \tag{1}$$

$$\boldsymbol{y}_k = \boldsymbol{C}_k\boldsymbol{x}_k + \boldsymbol{D}_k\boldsymbol{u}_k + \boldsymbol{v}_k. \tag{2}$$

Here, $\boldsymbol{u}_k \in \mathbb{R}^m$ is the input, $\boldsymbol{x}_k \in \mathbb{R}^n$ is the state, and $\boldsymbol{y}_k \in \mathbb{R}^p$ is the measurement at time k. Also, $\boldsymbol{w}_k$ and $\boldsymbol{v}_k$ are the process noise and measurement noise, respectively. The noise $\boldsymbol{w}_k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q}_k)$ and $\boldsymbol{v}_k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{R}_k)$ are independent, where $\boldsymbol{Q}_k \in \mathbb{R}^{n\times n}$ and $\boldsymbol{R}_k \in \mathbb{R}^{p\times p}$ are positive definite matrices. Here, $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes the Gaussian distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$.

We aim to simultaneously estimate the states and sparse inputs $\{\boldsymbol{x}_k, \boldsymbol{u}_k : \|\boldsymbol{u}_k\|_0 \ll m\}_{k=1}^K$ from the low dimensional measurements $\{\boldsymbol{y}_k\}_{k=1}^K$ with $p < m$, for a given $K > 0$. Here, $\|\cdot\|_0$ denotes the ℓ_0 norm. We emphasize that our focus is not on the design of sparse control inputs, but on recovering the states and sparse inputs of the system.

A. Related Work

Joint recovery of states and input without assuming any specific structure on the inputs or states has been studied extensively [17]–[21], but these works ignore any underlying sparsity structure that may exist in the system. Exploiting sparsity can facilitate the recovery of states and inputs with far fewer measurements than conventional approaches. The existing studies on the recovery of states and inputs in sparsity-constrained LDS consider one of three problems:

1) Estimation of sparse initial state $\boldsymbol{x}_1$

Estimating the sparse initial state of an LDS is equivalent to the standard sparse recovery problem, and can be solved using algorithms like LASSO, orthogonal matching pursuit, or sparse Bayesian learning (SBL) [22]–[24]. Theoretical results on the recoverability of a sparse initial state from low dimensional linear measurements of the states when the sparsity is exploited have been derived in [23]–[27]. These works focus on estimating the sparse initial state, assuming complete knowledge of the inputs applied to the system.

2) Estimation of sparse states $\{\boldsymbol{x}_k\}_{k=1}^N$

The sparse state estimation of an LDS without the knowledge of inputs has been studied in diverse applications. Sparsity-aware Kalman filtering was proposed in [28] to track abrupt changes in the sequence of sparse states of an LDS. Further, sparse state estimation was discussed in [29], where an SBL-based algorithm was used under the assumption of jointly sparse states. Another approach for jointly sparse state recovery algorithms imposed an ℓ_1-regularizer in the Kalman smoothing cost, followed by the alternating direction method of multipliers (ADMM) method [30]. Furthermore, the recovery of sparse states without the joint sparsity assumption was studied via ℓ_1-regularization-based dynamic filtering [31] and via a variational form of SBL in [32]. Additionally, [33] considered a general non-linear state space model with linear measurements of sparse states and developed an SBL-based dynamic filtering algorithm. However, these works assume the system matrices and inputs are restricted to ensure that $\boldsymbol{x}_{k+1}$ in (1) is sparse.

3) Joint estimation of states and sparse inputs $\{\boldsymbol{x}_k, \boldsymbol{u}_k\}_{k=1}^N$

In [34], the problem of jointly recovering the state and sparse input sequences as an ℓ_1-minimization using convex optimization methods was considered, i.e., minimizing $\sum_{k=1}^K \|\boldsymbol{u}_k\|_1$. Necessary and sufficient conditions for observability of sparse control inputs and the initial state for a noiseless LDS have been investigated in [35], [36]. These methods involve solving for a large-dimensional unknown sparse vector obtained by stacking the input vectors and do not exploit the temporal correlation in the state evolution. Consequently, the resulting algorithms have high computational complexity and memory requirements. To address these literature gaps, we presents new sparsity-driven estimators with better performance and lower complexities.

B. Contributions

We address the joint estimation of states and sparse inputs from (1) and (2) using p observations per time step over K time steps. With $K(n+m)$ unknowns and only Kp observations with $p \ll n+m$, the system is highly underdetermined. We solve the problem by enforcing sparsity via fictitious priors from the exponential family and integrating the resulting Bayesian framework with Kalman smoothing to exploit temporal correlations, yielding different estimation methods. Our contributions are as follows:

1) Regularizer-based approach: We integrate the sparsity-promoting priors in the maximum a posteriori (MAP) estimator of the system's state and inputs leading to ℓ_1-regularized and reweighted ℓ_2-regularized algorithms. (See Section III.)

2) Bayesian learning-based approach: We develop two techniques that use hierarchical Gaussian priors to induce sparsity. First, we rely on type-II maximum likelihood (ML) estimation combined with Kalman smoothing to determine the states and sparse inputs. Second, we develop a variational Bayesian (VB) algorithm that groups the prior parameters and unknown states and inputs as unobserved variables, and infers their posterior distributions. (See Section IV.)

3) Comparison and extension: We analyze and derive the time and memory complexities of both approaches in Section IV.C. Our empirical studies in Section V show that Bayesian learning-based algorithms outperform the regularizer-based approach and have lower complexity. We also extend the two approaches to the case of jointly sparse control inputs and present a similar analysis.

The key innovation in this work is seamlessly integrating advanced sparse recovery techniques with the Kalman smoothing framework. Unlike most control algorithms that exploit sparsity using basic ℓ_1-norm regularization, often neglecting the temporal evolution of the state process, our work bridges the gap between sparse signal processing and control theory by introducing sophisticated Bayesian sparse signal recovery methods within the Kalman smoothing framework.

This work significantly extends the conference paper [1] in several key aspects: we present a family of regularizer-based approaches, convergence analysis of the algorithms, and more comprehensive simulation results along with comparisons to extended versions of state-of-the-art algorithms.

Notation: The ith entry of vector $\boldsymbol{a}$ is $\boldsymbol{a}(i)$ and the (i,j)th entry of matrix $\boldsymbol{A}$ is $\boldsymbol{A}(i,j)$. We denote a sequence of vectors $\{\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_N\}$ by $\boldsymbol{A}_1^N$. The ℓ_1-norm is denoted by $\|\cdot\|_1$, and $\|\cdot\|$ denotes the induced ℓ_2-norm for matrices and Euclidean norm for vectors. Also, $\boldsymbol{A}_{\mathcal{S}}$ denotes the submatrix of $\boldsymbol{A}$ with the columns indexed by set $\mathcal{S}$, and $\boldsymbol{x}_{\mathcal{S}}$ denotes the subvector of $\boldsymbol{x}$ with entries indexed by $\mathcal{S}$. If $\boldsymbol{P}$ is a nonsingular matrix, we define $\|\boldsymbol{x}\|_{\boldsymbol{P}} = \boldsymbol{x}^\top \boldsymbol{P}^{-1} \boldsymbol{x}$.

This article has been accepted for publication in IEEE Open Journal of Control Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/OJCSYS.2025.3624615

II. MAP Estimation of States and Sparse Inputs

In this section, we introduce the generalized framework for Bayesian estimation of initial state and sparse inputs via sparsity-promoting priors. Specifically, we consider the estimation of the states and inputs $\{\boldsymbol{x}_k, \boldsymbol{u}_k\}_{k=1}^K$ using measurements $\boldsymbol{Y}_1^K \triangleq \{\boldsymbol{y}_k\}_{k=1}^K$ in (2) using a Bayesian framework, exploiting the sparsity of the inputs and the temporal correlation across the states dictated by (1).

The MAP estimates $\hat{\boldsymbol{x}}_{k|K}, \hat{\boldsymbol{u}}_{k|K}$ of the states and inputs are computed using the joint distribution of all the states and inputs $\{\boldsymbol{x}_k, \boldsymbol{u}_k\}_{k=1}^K$ given all observations $\boldsymbol{Y}_1^K$, subject to the system dynamics in (1) and (2), i.e.,

$$\begin{aligned}\left\{\hat{\boldsymbol{x}}_{k|K}, \hat{\boldsymbol{u}}_{k|K}\right\}_{k=1}^K &= \underset{\substack{\boldsymbol{x}_k, \boldsymbol{u}_k \\ k=1,\ldots,K}}{\arg\max}\, p\left(\{\boldsymbol{x}_k, \boldsymbol{u}_k\}_{k=1}^K \mid \boldsymbol{Y}_1^K\right) \\ &= \underset{\substack{\boldsymbol{x}_k, \boldsymbol{u}_k \\ k=1,\ldots,K}}{\arg\max} \prod_{k=1}^K p(\boldsymbol{y}_k|\boldsymbol{x}_k, \boldsymbol{u}_k) p(\boldsymbol{u}_k) \\ &\quad \times \prod_{k=2}^K p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}) \times p(\boldsymbol{x}_1), \quad (3)\end{aligned}$$

where we use the Markov structure implied by (1), and assume the input $\boldsymbol{u}_k$ is independent of the inputs. We note that the control inputs applied to the LDS need not be independent of the initial state. However, we make no assumptions about any prior knowledge of such relationships. Instead, we treat the inputs as unknown parameters and impose a fictitious prior to enforce sparsity (which we discuss later) to develop our algorithms.

Further, we do not make any assumptions about the initial state $\boldsymbol{x}_1$ and assume $p(\boldsymbol{x}_1)$ to be a (an improper) uniform prior. So, from (1) and (2), and using Gaussianity of the process and measurement noise, the optimization problem in (3) reduces to

$$\begin{aligned}\left\{\hat{\boldsymbol{x}}_{k|K}, \hat{\boldsymbol{u}}_{k|K}\right\}_{k=1}^K &= \underset{\substack{\boldsymbol{x}_k, \boldsymbol{u}_k \\ k=1,\ldots,K}}{\arg\min}\, \frac{1}{2}\sum_{k=1}^K \|\boldsymbol{y}_k - \boldsymbol{C}_k\boldsymbol{x}_k - \boldsymbol{D}_k\boldsymbol{u}_k\|_{\boldsymbol{R}_k}^2 \\ &+ \frac{1}{2}\sum_{k=1}^{K-1} \|\boldsymbol{x}_{k+1} - \boldsymbol{A}_k\boldsymbol{x}_k - \boldsymbol{B}_k\boldsymbol{u}_k\|_{\boldsymbol{Q}_k}^2 - \sum_{k=1}^K \ln(p(\boldsymbol{u}_k)). \quad (4)\end{aligned}$$

Since the input is known to be sparse, we encode this information into the estimation model via suitable priors on the inputs. Based on different sparsity-promoting priors, we develop two approaches: (a) regularized robust Kalman smoothing (RKS) and (b) Bayesian RKS, presented next.

III. Regularized Robust Kalman Smoothing

Inspired by the convex optimization-based sparse signal recovery algorithms [22], [37], we use the following prior to induce sparsity:

$$p(\boldsymbol{u}_k) = \prod_{i=1}^m \chi \exp\left[-\frac{\tau_k}{2}|\boldsymbol{u}_k(i)|^l\right],$$

where χ is the normalizing constant and $\tau_k, l > 0$ are known distribution parameters. Here, τ_k controls the sparsity of the control inputs, i.e., a large value of τ_k leads to sparser solutions. This parameter is often chosen by cross-validation. The choice of l determines the properties of the optimization problem in (4), leading to different estimators as given below.

A. ℓ_1-regularized Robust Kalman Smoothing

Motivated by the ℓ_1 norm-based Laplacian prior, the most popular choice is $l = 1$. With $l = 1$, the optimization problem in (4) becomes

$$\begin{aligned}\left\{\hat{\boldsymbol{x}}_{k|K}, \hat{\boldsymbol{u}}_{k|K}\right\}_{k=1}^K &= \underset{\substack{\boldsymbol{x}_k, \boldsymbol{u}_k \\ k=1,\ldots,K}}{\arg\min} \sum_{k=1}^K \|\boldsymbol{y}_k - \boldsymbol{C}_k\boldsymbol{x}_k - \boldsymbol{D}_k\boldsymbol{u}_k\|_{\boldsymbol{R}_k}^2 \\ &+ \sum_{k=1}^{K-1} \|\boldsymbol{x}_{k+1} - \boldsymbol{A}_k\boldsymbol{x}_k - \boldsymbol{B}_k\boldsymbol{u}_k\|_{\boldsymbol{Q}_k}^2 + \sum_{k=1}^K \tau_k \|\boldsymbol{u}_k\|_1. \quad (5)\end{aligned}$$

The convex optimization problem in (5) has no closed-form solution. So, we use the ADMM algorithm [38] to solve it. ADMM decomposes the convex optimization problem in (5) into simpler optimization problems. We reformulate (5) to another equivalent optimization problem using auxiliary variables $\{\boldsymbol{t}_k \in \mathbb{R}^m\}_{k=1}^K$. We define the augmented Lagrangian function as

$$\begin{aligned}\mathcal{L}\left(\{\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{t}_k, \boldsymbol{\lambda}_k\}_{k=1}^K\right) &= \sum_{k=1}^K \|\boldsymbol{y}_k - \boldsymbol{C}_k\boldsymbol{x}_k - \boldsymbol{D}_k\boldsymbol{u}_k\|_{\boldsymbol{R}_k}^2 \\ &+ \sum_{k=1}^{K-1} \|\boldsymbol{x}_{k+1} - \boldsymbol{A}_k\boldsymbol{x}_k - \boldsymbol{B}_k\boldsymbol{u}_k\|_{\boldsymbol{Q}_k}^2 \\ &+ \sum_{k=1}^K \left[\tau_k \|\boldsymbol{t}_k\|_1 + \boldsymbol{\lambda}_k^\mathsf{T}(\boldsymbol{t}_k - \boldsymbol{u}_k) + c\|\boldsymbol{t}_k - \boldsymbol{u}_k\|^2\right], \quad (6)\end{aligned}$$

where $\{\boldsymbol{\lambda}_k \in \mathbb{R}^m\}_{k=1}^K$ are the Lagrangian multipliers that arise from the K constraints $\boldsymbol{t}_k = \boldsymbol{u}_k$. Also, $c > 0$ is a scalar. ADMM is an iterative algorithm that alternately solves subproblems of (6), each focusing on a specific block of variables. The rth iteration updates are

$$\left\{\boldsymbol{x}_k^{(r)}, \boldsymbol{u}_k^{(r)}\right\}_{k=1}^K = \underset{\{\boldsymbol{x}_k, \boldsymbol{u}_k\}_{k=1}^K}{\arg\min}\, \mathcal{L}\left(\left\{\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{t}_k^{(r-1)}\boldsymbol{\lambda}_k^{(r-1)}\right\}_{k=1}^K\right) \quad (7)$$

$$\boldsymbol{t}_k^{(r)} = \arg\min_{\boldsymbol{t}_k} \mathcal{L}\left(\left\{\boldsymbol{x}_k^{(r)}, \boldsymbol{u}_k^{(r)}, \boldsymbol{t}_k, \boldsymbol{\lambda}_k^{(r-1)}\right\}_{k=1}^K\right) \quad (8)$$

$$\boldsymbol{\lambda}_k^{(r)} = \boldsymbol{\lambda}_k^{(r-1)} + 2c\left(\boldsymbol{u}_k^{(r)} - \boldsymbol{t}_k^{(r)}\right), \quad (9)$$

for $k = 1, 2, \ldots, K$. Further, (7) can be simplified as

$$\begin{aligned}\left\{\boldsymbol{x}_k^{(r)}, \boldsymbol{u}_k^{(r)}\right\}_{k=1}^K &= \underset{\substack{\boldsymbol{x}_k, \boldsymbol{u}_k \\ k=1,\ldots,K}}{\arg\min} \sum_{k=1}^K \left\|\boldsymbol{y}_k^{\ell_1} - \tilde{\boldsymbol{C}}_k\boldsymbol{x}_k - \tilde{\boldsymbol{D}}_k\boldsymbol{u}_k\right\|_{\boldsymbol{R}_k^{\ell_1}}^2 \\ &+ \sum_{k=1}^{K-1} \|\boldsymbol{x}_{k+1} - \boldsymbol{A}_k\boldsymbol{x}_k - \boldsymbol{B}_k\boldsymbol{u}_k\|_{\boldsymbol{Q}_k}^2. \quad (10)\end{aligned}$$

Here, we define the new matrices

$$\boldsymbol{y}_k^{\ell_1} = \begin{bmatrix} \boldsymbol{y}_k \\ \boldsymbol{t}_k^{(r-1)} - c^{-1}\boldsymbol{\lambda}_k^{(r-1)} \end{bmatrix}, \quad \boldsymbol{R}_k^{\ell_1} = \begin{bmatrix} \boldsymbol{R}_k & \boldsymbol{0} \\ \boldsymbol{0} & c^{-1}\boldsymbol{I} \end{bmatrix} \tag{11}$$

$$\tilde{\boldsymbol{C}}_k = \begin{bmatrix} \boldsymbol{C}_k^\mathsf{T} & \boldsymbol{0} \end{bmatrix}^\mathsf{T}, \qquad \tilde{\boldsymbol{D}}_k = \begin{bmatrix} \boldsymbol{D}_k^\mathsf{T} & \boldsymbol{I} \end{bmatrix}^\mathsf{T}. \tag{12}$$

To solve (10), we next extend Kalman filtering and smoothing, laying the foundation for our algorithms in this paper.

Theorem 1:
Consider the MAP estimates of the states and inputs of the linear system in (1) and (2) with the assumption that $\boldsymbol{D}_k$'s have full column rank and without any prior information on the inputs, given by

$$\underset{\substack{\boldsymbol{x}_k, \boldsymbol{u}_k \\ k=1,\dots,K}}{\arg\min} \sum_{k=1}^{K} \|\boldsymbol{y}_k - \boldsymbol{C}_k\boldsymbol{x}_k - \boldsymbol{D}_k\boldsymbol{u}_k\|_{\boldsymbol{R}_k}^2 + \sum_{k=1}^{K-1} \|\boldsymbol{x}_{k+1} - \boldsymbol{A}_k\boldsymbol{x}_k - \boldsymbol{B}_k\boldsymbol{u}_k\|_{\boldsymbol{Q}_k}^2. \tag{13}$$

This problem can be solved recursively using the prediction, filtering, and smoothing steps, using the robust Kalman smoothing (RKS) algorithm summarized in Algorithm 1.

Proof: See Appendix A.

We note that the statement of Theorem 1 requires $p \geq m$, which implies that, in the absence of additional structural information about the inputs $\boldsymbol{u}_k$, a necessary condition for obtaining MAP estimates is that the number of measurements must be at least as large as the input dimension.

In Algorithm 1, we determine the statistics of the joint Gaussian posterior distribution of $\boldsymbol{x}_k$ and $\boldsymbol{u}_k$ given $\boldsymbol{Y}_1^k$. We know for any t and k the posterior distribution of the state $\boldsymbol{x}_t$ and input $\boldsymbol{u}_t$ given measurements $\boldsymbol{Y}_1^K$ is given by

$$p\left(\boldsymbol{x}_t, \boldsymbol{u}_t | \boldsymbol{Y}_1^k\right) = \mathcal{N}\left(\begin{bmatrix} \hat{\boldsymbol{x}}_{t|k} \\ \hat{\boldsymbol{u}}_{t|k} \end{bmatrix}, \begin{bmatrix} \boldsymbol{P}_{t|k}^{\boldsymbol{x}} & \boldsymbol{P}_{t|k}^{\boldsymbol{xu}} \\ \left(\boldsymbol{P}_{t|k}^{\boldsymbol{xu}}\right)^\mathsf{T} & \boldsymbol{P}_{t|k}^{\boldsymbol{u}} \end{bmatrix}\right). \tag{14}$$

Here, $\hat{\boldsymbol{x}}_{t|k} \in \mathbb{R}^n$ and $\hat{\boldsymbol{u}}_{t|k} \in \mathbb{R}^m$ are the estimates of $\boldsymbol{x}_t$ and $\boldsymbol{u}_t$ given $\boldsymbol{Y}_1^k$ with associated covariances $\boldsymbol{P}_{t|k}^{\boldsymbol{x}} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{P}_{t|k}^{\boldsymbol{u}} \in \mathbb{R}^{m\times m}$, respectively, and cross-covariance $\boldsymbol{P}_{t|k}^{\boldsymbol{xu}} \in \mathbb{R}^{n\times m}$. Then, defining $\boldsymbol{\xi}_t^\mathsf{T} = \begin{bmatrix} \boldsymbol{x}_t^\mathsf{T} & \boldsymbol{u}_t^\mathsf{T} \end{bmatrix}$, we get $p\left(\boldsymbol{\xi}_t | \boldsymbol{Y}_1^k\right) \sim \mathcal{N}\left(\hat{\boldsymbol{\xi}}_{t|k}, \boldsymbol{P}_{t|k}^{\boldsymbol{\xi}}\right)$, where $\hat{\boldsymbol{\xi}}_{t|k}$ and $\boldsymbol{P}_{t|k}^{\boldsymbol{\xi}}$ are the mean and covariance of the distribution in (14).

Since (10) has the same form as (13), its solution follows the RKS algorithm with the measurement model in (11) and (12). Further, from (8), the auxiliary variable update is

$$\begin{aligned} \boldsymbol{t}_k^{(r)} &= \underset{\boldsymbol{t}_k}{\arg\min}\ \tau_k \|\boldsymbol{t}_k\|_1 + c\left\|\boldsymbol{t}_k - c^{-1}\boldsymbol{\lambda}_k^{(r-1)} - \boldsymbol{u}_k^{(r)}\right\|^2 \\ &= S_{c^{-1}\tau_k}\left(\boldsymbol{u}_k^{(r)} + c^{-1}\boldsymbol{\lambda}_k^{(r-1)}\right), \end{aligned} \tag{15}$$

where the last step follows from the LASSO solution. Also, $S(\cdot)$ is the entry-wise soft thresholding function, $S_b(a) = \mathrm{Sgn}(a)\max\{|a| - b, 0\}$. Combing (9), (10) with Theorem 1, and (15), ℓ_1-regularized RKS is summarized in Algorithm 2.

Algorithm 1 Robust Kalman Smoothing

Inputs: $\{\boldsymbol{y}_k, \boldsymbol{A}_k, \boldsymbol{B}_k, \boldsymbol{C}_k, \boldsymbol{D}_k, \boldsymbol{Q}_k, \boldsymbol{R}_k\}_{k=1}^K$
Initialization: $\hat{\boldsymbol{\xi}}_{0|0} = \boldsymbol{0} \in \mathbb{R}^{n+m}$, $\boldsymbol{P}_{0|0}^{\boldsymbol{\xi}} \in \mathbb{R}^{(n+m)\times(n+m)}$ and $\boldsymbol{Q}_0 = \boldsymbol{0} \in \mathbb{R}^{n\times n}$
$\boldsymbol{\xi}_k = \begin{bmatrix} \boldsymbol{x}_k^\mathsf{T} & \boldsymbol{u}_k^\mathsf{T} \end{bmatrix}^\mathsf{T}$, $\tilde{\boldsymbol{A}}_k = \begin{bmatrix} \boldsymbol{A}_k & \boldsymbol{B}_k \end{bmatrix}$ and $\boldsymbol{T} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \end{bmatrix}^\mathsf{T} \in \mathbb{R}^{(n+m)\times n}$
$\boldsymbol{J}_k = \left(\boldsymbol{D}_k^\mathsf{T}\boldsymbol{R}_k^{-1}\boldsymbol{D}_k\right)^{-1}\boldsymbol{D}_k^\mathsf{T}\boldsymbol{R}_k^{-1}$ for $k = 1, \dots K$
for $k = 1, 2, \dots, K$ **do**
#Prediction:
$\hat{\boldsymbol{x}}_{k|k-1} = \tilde{\boldsymbol{A}}_{k-1}\hat{\boldsymbol{\xi}}_{k-1|k-1}$
$\boldsymbol{P}_{k|k-1}^{\boldsymbol{x}} = \tilde{\boldsymbol{A}}_{k-1}\boldsymbol{P}_{k-1|k-1}^{\boldsymbol{\xi}}\tilde{\boldsymbol{A}}_{k-1}^\mathsf{T} + \boldsymbol{Q}_{k-1}$
#Filtering:
$\boldsymbol{L}_k = \boldsymbol{P}_{k|k-1}^{\boldsymbol{x}}\boldsymbol{C}_k^\mathsf{T}\left(\boldsymbol{R}_k + \boldsymbol{C}_k\boldsymbol{P}_{k|k-1}^{\boldsymbol{x}}\boldsymbol{C}_k^\mathsf{T}\right)^{-1}$
$\boldsymbol{\mathcal{G}}_k = \begin{bmatrix} (\boldsymbol{I} - \boldsymbol{L}_k\boldsymbol{D}_k\boldsymbol{J}_k\boldsymbol{C}_k)^{-1}\boldsymbol{L}_k(\boldsymbol{I} - \boldsymbol{D}_k\boldsymbol{J}_k) \\ (\boldsymbol{I} - \boldsymbol{J}_k\boldsymbol{C}_k\boldsymbol{L}_k\boldsymbol{D}_k)^{-1}\boldsymbol{J}_k(\boldsymbol{I} - \boldsymbol{C}_k\boldsymbol{L}_k) \end{bmatrix}$
$\hat{\boldsymbol{\xi}}_{k|k} = \boldsymbol{T}\hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{\mathcal{G}}_k\left(\boldsymbol{y}_k - \boldsymbol{C}_k\hat{\boldsymbol{x}}_{k|k-1}\right)$
$\boldsymbol{P}_{k|k}^{\boldsymbol{\xi}} = (\boldsymbol{T} - \boldsymbol{\mathcal{G}}_k\boldsymbol{C}_k)\boldsymbol{P}_{k|k-1}^{\boldsymbol{x}}(\boldsymbol{T} - \boldsymbol{\mathcal{G}}_k\boldsymbol{C}_k)^\mathsf{T} + \boldsymbol{\mathcal{G}}_k\boldsymbol{R}_k\boldsymbol{\mathcal{G}}_k^\mathsf{T}$
Compute $\hat{\boldsymbol{x}}_{k|k}$ and $\boldsymbol{P}_{k|k}^{\boldsymbol{x}}$ from $\hat{\boldsymbol{\xi}}_{k|k}$ and $\boldsymbol{P}_{k|k}^{\boldsymbol{\xi}}$
end for
#Smoothing:
for $k = K-1, K-2, \dots, 1$ **do**
$\boldsymbol{K}_k = \boldsymbol{P}_{k|k}^{\boldsymbol{\xi}}\tilde{\boldsymbol{A}}_k^\mathsf{T}\left(\boldsymbol{P}_{k+1|k}^{\boldsymbol{x}}\right)^{-1}$
$\boldsymbol{P}_{k|K}^{\boldsymbol{\xi}} = \boldsymbol{P}_{k|k}^{\boldsymbol{\xi}} + \boldsymbol{K}_k\left(\boldsymbol{P}_{k+1|K}^{\boldsymbol{x}} - \boldsymbol{P}_{k+1|k}^{\boldsymbol{x}}\right)\boldsymbol{K}_k^\mathsf{T}$
$\hat{\boldsymbol{\xi}}_{k|K} = \hat{\boldsymbol{\xi}}_{k|k} + \boldsymbol{K}_k\left(\hat{\boldsymbol{x}}_{k+1|K} - \tilde{\boldsymbol{A}}_k\hat{\boldsymbol{\xi}}_{k|k}\right)$
Compute $\hat{\boldsymbol{x}}_{k|K}$ and $\boldsymbol{P}_{k|K}^{\boldsymbol{x}}$ from $\hat{\boldsymbol{\xi}}_{k|K}$ and $\boldsymbol{P}_{k|K}^{\boldsymbol{\xi}}$
end for
Outputs: $\{\hat{\boldsymbol{x}}_{k|K}\}_{k=1}^K$ and $\{\hat{\boldsymbol{u}}_{k|K}\}_{k=1}^K$

In Algorithm 2, the full column rank requirement of Theorem 1 holds for $\tilde{\boldsymbol{D}}_k = \begin{bmatrix} \boldsymbol{D}_k^\mathsf{T} & \boldsymbol{I} \end{bmatrix}^\mathsf{T} \in \mathbb{R}^{(p+m)\times m}$.

Since (5) is convex, ADMM is guaranteed to converge to the global solution. A general limitation of ℓ_1-based regularization, however, is that it may yield biased estimates for large coefficients, thereby necessitating careful tuning of the penalty parameters [39].

B. Reweighted ℓ_2-regularized Robust Kalman Smoothing

When $0 < l < 2$, an alternative algorithm, similar in spirit to the iterative reweighting-based sparse recovery [37], can be derived. Then, the optimization problem (13) changes to

$$\{\hat{\boldsymbol{x}}_{k|K}, \hat{\boldsymbol{u}}_{k|K}\}_{k=1}^K = \underset{\substack{\boldsymbol{x}_k, \boldsymbol{u}_k \\ k=1,\dots,K}}{\arg\min} \sum_{k=1}^{K} \|\boldsymbol{y}_k - \boldsymbol{C}_k\boldsymbol{x}_k - \boldsymbol{D}_k\boldsymbol{u}_k\|_{\boldsymbol{R}_k}^2 + \sum_{k=1}^{K-1} \|\boldsymbol{x}_{k+1} - \boldsymbol{A}_k\boldsymbol{x}_k - \boldsymbol{B}_k\boldsymbol{u}_k\|_{\boldsymbol{Q}_k}^2 + \sum_{k=1}^{K}\tau_k\sum_{i=1}^{m}|\boldsymbol{u}_k(i)|^l. \tag{16}$$

Algorithm 2 ℓ_1-regularized Robust Kalman Smoothing

Inputs: $\{\boldsymbol{y}_k, \boldsymbol{A}_k, \boldsymbol{B}_k, \boldsymbol{C}_k, \boldsymbol{D}_k, \boldsymbol{Q}_k, \boldsymbol{R}_k\}_{k=1}^K$

Parameters: c, $r_{\max}$, and τ_k for $k = 1, 2, \ldots, K$, $r = 1$, $\epsilon = 2\epsilon_{\text{thres}}$

Initialization: $\boldsymbol{t}_k^{(0)} = \boldsymbol{\lambda}_k^{(0)} = \boldsymbol{0}$, for $k = 1, 2, \ldots, K$

Compute $\boldsymbol{R}_k^{\ell_1}$, $\tilde{\boldsymbol{C}}_k$, $\tilde{\boldsymbol{D}}_k$ using (11) and (12)

while $(r < r_{max})$ or $(\epsilon > \epsilon_{\text{thres}})$ **do**

  Compute $\boldsymbol{y}_k^{\ell_1}$ using (11)

  Compute $\boldsymbol{x}_k^{(r)}$ and $\boldsymbol{u}_k^{(r)}$ via Algorithm 1 replacing $\boldsymbol{y}_k$, $\boldsymbol{C}_k$, $\boldsymbol{D}_k$, and $\boldsymbol{R}_k$ with $\boldsymbol{y}_k^{\ell_1}$, $\tilde{\boldsymbol{C}}_k$, $\tilde{\boldsymbol{D}}_k$, and $\boldsymbol{R}_k^{\ell_1}$, respectively

  for $k = 1, 2, \ldots, K$ **do**

    Using (15), $\boldsymbol{t}_k^{(r)} = S_{c^{-1}\tau_k}\left(\boldsymbol{u}_k^{(r)} + c^{-1}\boldsymbol{\lambda}_k^{(r-1)}\right)$

    $\boldsymbol{\lambda}_k^{(r)} = \boldsymbol{\lambda}_k^{(r-1)} + 2c\left(\boldsymbol{u}_k^{(r)} - \boldsymbol{t}_k^{(r)}\right)$

  end for

  $\epsilon = \sum_{k=1}^K \left\|\boldsymbol{\xi}_k^{(r)} - \boldsymbol{\xi}_k^{(r-1)}\right\|_2^2$ where $\boldsymbol{\xi}_k^{(r)} = \begin{bmatrix}\boldsymbol{x}_k^{(r)} \\ \boldsymbol{u}_k^{(r)}\end{bmatrix}$

  $r = r + 1$

end while

Outputs: $\left\{\boldsymbol{x}_k^{(r)}\right\}_{k=1}^K$ and $\left\{\boldsymbol{u}_k^{(r)}\right\}_{k=1}^K$

We use the majorization minimization (MM) technique to solve (16). In MM, the cost function is upper-bounded around the current iterate by a tractable function that is easy to optimize, and we minimize that upper bound to get the new iterate. Here, we bound the regularizer term by noting that the function $|u|^l = (|u|^2)^{l/2}$ is a concave function of $|u|^2$. Therefore, the function is bounded above by its first-order Taylor approximation, and for any $u \in \mathbb{R}$,

$$\left(|\boldsymbol{u}_k(i)|^2\right)^{l/2} \leq \left(|u|^2\right)^{l/2} + \frac{l}{2}\left(|u|^2\right)^{l/2-1}\left(|\boldsymbol{u}_k(i)|^2 - |u|^2\right).$$

With $u = \boldsymbol{u}_k^{(r-1)}(i)$, we get

$$\left(|\boldsymbol{u}_k(i)|^2\right)^{l/2} \leq \frac{l}{2}\left|\boldsymbol{u}_k^{(r-1)}(i)\right|^{-(2-l)}|\boldsymbol{u}_k(i)|^2 + \left(1 - \frac{l}{2}\right)\left|\boldsymbol{u}_k^{(r-1)}(i)\right|^l. \quad (17)$$

We now iteratively solve (16) by optimizing the upper bound in each iteration. So, the rth iteration computes

$$\left\{\boldsymbol{x}_k^{(r)}, \boldsymbol{u}_k^{(r)}\right\}_{k=1}^K = \underset{\substack{\boldsymbol{x}_k, \boldsymbol{u}_k \\ k=1,\ldots,K}}{\arg\min} \sum_{k=1}^K \|\boldsymbol{y}_k - \boldsymbol{C}_k\boldsymbol{x}_k - \boldsymbol{D}_k\boldsymbol{u}_k\|_{\boldsymbol{R}_k}^2 + \sum_{k=1}^{K-1}\|\boldsymbol{x}_{k+1} - \boldsymbol{A}_k\boldsymbol{x}_k - \boldsymbol{B}_k\boldsymbol{u}_k\|_{\boldsymbol{Q}_k}^2 + \frac{l}{2}\sum_{k=1}^K \tau_k \|\boldsymbol{u}_k\|_{\boldsymbol{W}_k^{(r)}}^2. \quad (18)$$

Here, the diagonal weight matrix $\boldsymbol{W}_k^{(r)} \in \mathbb{R}^{m\times m}$ is

$$\boldsymbol{W}_k^{(r)} = \operatorname{diag}\left\{\left|\boldsymbol{u}_k^{(r-1)}\right|\right\}^{(2-l)},$$

with $|\cdot|$ representing the element-wise modulus. When an entry of $\boldsymbol{u}_k^{(r-1)}$ become close to zeros, $\boldsymbol{W}_k^{(r)}$ can potentially become non-invertible. To avoid numerical instabilities, we prune the entries of $\boldsymbol{u}_k^{(r-1)}$ falls below some small threshold κ (e.g., $\kappa = 10^{-6}$) in magnitude. Now, (18) modifies to

$$\left\{\boldsymbol{x}_k^{(r)}, \tilde{\boldsymbol{u}}_k^{(r)}\right\}_{k=1}^K = \underset{\substack{\boldsymbol{x}_k, \tilde{\boldsymbol{u}}_k \\ k=1,\ldots,K}}{\arg\min} \sum_{k=1}^K \left\|\boldsymbol{y}_k^{\ell_2} - \boldsymbol{C}_k^{\ell_2}\boldsymbol{x}_k - \boldsymbol{D}_k^{\ell_2}\tilde{\boldsymbol{u}}_k\right\|_{\boldsymbol{R}_k^{\ell_2}}^2 + \sum_{k=1}^{K-1}\left\|\boldsymbol{x}_{k+1} - \boldsymbol{A}_k\boldsymbol{x}_k - \boldsymbol{B}_k^{\ell_2}\tilde{\boldsymbol{u}}_k\right\|_{\boldsymbol{Q}_k}^2, \quad (19)$$

where we define

$$\boldsymbol{y}_k^{\ell_2} = \begin{bmatrix}\boldsymbol{y}_k^\mathsf{T} & \boldsymbol{0}_{|\mathcal{I}_k|}\end{bmatrix}^\mathsf{T}, \qquad \boldsymbol{B}_k^{\ell_2} = (\boldsymbol{B}_k)_{\mathcal{I}_k} \quad (20)$$

$$\boldsymbol{C}_k^{\ell_2} = \begin{bmatrix}\boldsymbol{C}_k^\mathsf{T} & \boldsymbol{0}_{n\times|\mathcal{I}_k|}\end{bmatrix}^\mathsf{T}, \quad \boldsymbol{D}_k^{\ell_2} = \begin{bmatrix}(\boldsymbol{D}_k)_{\mathcal{I}_k}^\mathsf{T} & \boldsymbol{I}_{|\mathcal{I}_k|}\end{bmatrix}^\mathsf{T}, \quad (21)$$

where $\tilde{\boldsymbol{u}}_k^{(r-1)} \in \mathbb{R}^{|\mathcal{I}_k|}$ is the vector obtained after removing the entries of $\boldsymbol{u}_k^{(r-1)}$ whose absolute values are less than κ, and $\mathcal{I}_k$ is the set of indices corresponding to the entries of $\boldsymbol{u}_k^{(r-1)}$ which have absolute value at least κ. Also, we define

$$\boldsymbol{R}_k^{\ell_2} = \begin{bmatrix}\boldsymbol{R}_k & \boldsymbol{0} \\ \boldsymbol{0} & \frac{2}{\tau_k l}\operatorname{diag}\left\{\left|\tilde{\boldsymbol{u}}_k^{(r-1)}\right|\right\}^{(2-l)}\end{bmatrix}. \quad (22)$$

Here, (19) takes the same form as (5) and can be solved via RKS in Algorithm 1. Finally, we reconstruct $\boldsymbol{u}_k^{(r)}$ as

$$\left(\boldsymbol{u}_k^{(r)}\right)_{\mathcal{I}_k} = \tilde{\boldsymbol{u}}_k^{(r)} \quad \text{and} \quad \left(\boldsymbol{u}_k^{(r)}\right)_{\mathcal{I}_k^\complement} = \boldsymbol{0}. \quad (23)$$

The pseudocode for the overall reweighted ℓ_2-regularized RKS is summarized in Algorithm 3.

Algorithm 3 Reweighted ℓ_2-regularized Robust Kalman Smoothing

Inputs: $\{\boldsymbol{y}_k, \boldsymbol{A}_k, \boldsymbol{B}_k, \boldsymbol{C}_k, \boldsymbol{D}_k, \boldsymbol{Q}_k, \boldsymbol{R}_k\}_{k=1}^K$

Parameters: l, $r_{\max}$, κ, and τ_k for $k = 1, 2, \ldots, K$, $\epsilon = 2\epsilon_{\text{thres}}$

Initialization: $\boldsymbol{u}_k^{(0)} = \boldsymbol{1}$, for $k = 1, 2, \ldots, K$

while $(r < r_{max})$ or $(\epsilon > \epsilon_{\text{thres}})$ **do**

  Determine $\mathcal{I}_k$ for each $\boldsymbol{u}_k^{(r-1)}$ and compute $\tilde{\boldsymbol{u}}_k^{(r-1)}$

  Compute $\boldsymbol{y}_k^{\ell_2}$, $\boldsymbol{B}_k^{\ell_2}$, $\boldsymbol{C}_k^{\ell_2}$, $\boldsymbol{D}_k^{\ell_2}$, $\boldsymbol{R}_k^{\ell_2}$ using (20), (21) and (22)

  Compute $\boldsymbol{x}_k^{(r)}$ and $\tilde{\boldsymbol{u}}_k^{(r)}$ via Algorithm 1 replacing $\boldsymbol{y}_k$, $\boldsymbol{B}_k$, $\boldsymbol{C}_k$, $\boldsymbol{D}_k$, and $\boldsymbol{R}_k$ with $\boldsymbol{y}_k^{\ell_2}$, $\boldsymbol{B}_k^{\ell_2}$, $\boldsymbol{C}_k^{\ell_2}$, $\boldsymbol{D}_k^{\ell_2}$, and $\boldsymbol{R}_k^{\ell_2}$ respectively

  Construct $\boldsymbol{u}_k^{(r)}$ using (23)

  $\epsilon = \sum_{k=1}^K \left\|\boldsymbol{\xi}_k^{(r)} - \boldsymbol{\xi}_k^{(r-1)}\right\|_2^2$ where $\boldsymbol{\xi}_k^{(r)} = \begin{bmatrix}\boldsymbol{x}_k^{(r)} \\ \boldsymbol{u}_k^{(r)}\end{bmatrix}$

end while

Outputs: $\left\{\boldsymbol{x}_k^{(r)}\right\}_{k=1}^K$ and $\left\{\boldsymbol{u}_k^{(r)}\right\}_{k=1}^K$

Next, we show that iteratively optimizing the surrogate function in (18) via Algorithm 3 either decreases the value of the cost function in (16) or leaves it unchanged, and that the sequence of cost function values over iterations converges.

Proposition 1:

For a given set of inputs, let $\left\{\boldsymbol{x}_k^{(r)}, \boldsymbol{u}_k^{(r)}\right\}_{k=1}^{K}$ be the sequence generated by Algorithm 3. Then, the corresponding sequence of cost function values in (5) is monotonically non-increasing and converges to a limit in $\mathbb{R}$.

Proof:

For brevity, we define $\boldsymbol{x} = \boldsymbol{X}_1^K$ and $\boldsymbol{u} = \boldsymbol{U}_1^K$. Let the cost function in (16) be denoted by $f(\boldsymbol{x}, \boldsymbol{u}) = f_1(\boldsymbol{x}, \boldsymbol{u}) + f_2(\boldsymbol{u})$, where $f_1(\boldsymbol{x}, \boldsymbol{u})$ denotes the first two quadratic terms and $f_2(\boldsymbol{u}) = \sum_{k=1}^{K} \tau_k \sum_{i=1}^{m} |\boldsymbol{u}_k(i)|^l$ is the regularizer term. Also, let the upper bound in (17) be $g(\boldsymbol{u}|\boldsymbol{u}^{(r-1)}) \geq f_2(\boldsymbol{u})$ and equality holds when $\boldsymbol{u} = \boldsymbol{u}^{(r-1)}$. Hence, we derive

$$\begin{aligned} f(\boldsymbol{x}^{(r-1)}, \boldsymbol{u}^{(r-1)}) &= f_1(\boldsymbol{x}^{(r-1)}, \boldsymbol{u}^{(r-1)}) + g(\boldsymbol{u}^{(r-1)}|\boldsymbol{u}^{(r-1)}) \\ &\geq f_1(\boldsymbol{x}^{(r)}, \boldsymbol{u}^{(r)}) + g(\boldsymbol{u}^{(r)}|\boldsymbol{u}^{(r)}) \\ &\geq f_1(\boldsymbol{x}^{(r)}, \boldsymbol{u}^{(r)}) + f_2(\boldsymbol{u}^{(r)}) \\ &= f(\boldsymbol{x}^{(r)}, \boldsymbol{u}^{(r)}). \end{aligned}$$

The first inequality holds because, in the rth iteration, $(\boldsymbol{x}^{(r)}, \boldsymbol{u}^{(r)})$ the RKS algorithm minimizes the cost function in (18), which is equivalent to minimizing $f_1(\boldsymbol{x}, \boldsymbol{u}) + g(\boldsymbol{u}|\boldsymbol{u}^{(r-1)})$. The second inequality holds because $g(\boldsymbol{u}|\boldsymbol{u}^{(r)}) \geq f_2(\boldsymbol{u})$ for any $\boldsymbol{u}$. Hence, the sequence of the cost function values is monotonically non-increasing. Finally, $f(\boldsymbol{x}, \boldsymbol{u})$ is lower bounded by 0. Thus, by the monotone convergence theorem, the cost function sequence converges. ■

The above result guarantees that cost function derived from the iterates converges to a limit, but the iterates may oscillate, cycle, or converge to a local minimum or a saddle point.

Iterative reweighted ℓ_2 regularization provides stronger sparsity promotion than ℓ_1-based regularizer, since $0 < l < 1$ offers a closer approximation to the ℓ_0 norm than ℓ_1. However, for $0 < l < 1$ the problem becomes non-convex, and a key drawback is that the algorithm may get trapped in local minima, as indicated by Proposition 1.

C. Extension to Jointly Sparse Inputs

We can extend ℓ_1-regularized RKS to estimate jointly sparse inputs by modifying (5) as follows:

$$\begin{aligned} \left\{\hat{\boldsymbol{x}}_{k|K}, \hat{\boldsymbol{u}}_{k|K}\right\}_{k=1}^{K} = \underset{\substack{\boldsymbol{x}_k, \boldsymbol{u}_k \\ k=1,\ldots,K}}{\arg\min} & \sum_{k=1}^{K} \|\boldsymbol{y}_k - \boldsymbol{C}_k\boldsymbol{x}_k - \boldsymbol{D}_k\boldsymbol{u}_k\|_{\boldsymbol{R}_k}^2 \\ + \sum_{k=1}^{K-1} \|\boldsymbol{x}_{k+1} - \boldsymbol{A}_k\boldsymbol{x}_k - \boldsymbol{B}_k\boldsymbol{u}_k\|_{\boldsymbol{Q}_k}^2 & + \tau \sum_{i=1}^{m} \sqrt{\sum_{k=1}^{K} \boldsymbol{u}_k^2(i)}, \end{aligned} \quad (24)$$

where regularizer $\tau \sum_{i=1}^{m} \sqrt{\sum_{k=1}^{K} \boldsymbol{u}_k^2(i)}$ is inspired by the LASSO type regularization [40]. As $\tau > 0$ increases, the common support output by the algorithm shrinks, making the inputs sparser. Now, using auxiliary variables $\boldsymbol{t}_k$'s, we reformulate (24) and solve the following using ADMM:

$$\begin{aligned} \left\{\hat{\boldsymbol{x}}_{k|K}, \hat{\boldsymbol{u}}_{k|K}\right\}_{k=1}^{K} = \underset{\substack{\boldsymbol{x}_k, \boldsymbol{u}_k \\ k=1,\ldots,K}}{\arg\min} & \sum_{k=1}^{K} \|\boldsymbol{y}_k - \boldsymbol{C}_k\boldsymbol{x}_k - \boldsymbol{D}_k\boldsymbol{u}_k\|_{\boldsymbol{R}_k}^2 \\ + \sum_{k=1}^{K-1} \|\boldsymbol{x}_{k+1} - \boldsymbol{A}_k\boldsymbol{x}_k - \boldsymbol{B}_k\boldsymbol{u}_k\|_{\boldsymbol{Q}_k}^2 & + \tau \sum_{i=1}^{m} \sqrt{\sum_{k=1}^{K} t_k^2(i)} \\ + \sum_{k=1}^{K} \boldsymbol{\lambda}_k^{\mathsf{T}} (\boldsymbol{t}_k - \boldsymbol{u}_k) & + c \sum_{k=1}^{K} \|\boldsymbol{t}_k - \boldsymbol{u}_k\|^2, \end{aligned} \quad (25)$$

where $\{\boldsymbol{\lambda}_k \in \mathbb{R}^m\}_{k=1}^{K}$ are the Lagrangian multipliers that arise from the equality constraints $\boldsymbol{t}_k = \boldsymbol{u}_k$ and $c > 0$ is a positive scalar. Since (25) is identical to (6) except for the terms in $\boldsymbol{t}_k$, our modified ℓ_1-regularized RKS is identical to Algorithm 2 except for Step 6, which changes to [40],

$$\begin{aligned} \boldsymbol{t}^{(r)}(i,:) = & \frac{\boldsymbol{u}^{(r)}(i,:) + c^{-1}\boldsymbol{\lambda}^{(r-1)}(i,:)}{\left\|\boldsymbol{u}^{(r)}(i,:) + c^{-1}\boldsymbol{\lambda}^{(r-1)}(i,:)\right\|} \\ & \times S_{c^{-1}\tau}\left(\left\|\boldsymbol{u}^{(r)}(i,:) + c^{-1}\boldsymbol{\lambda}^{(r-1)}(i,:)\right\|\right), \end{aligned}$$

where function S is defined in (15), and $\boldsymbol{t}^{(r)}(i,:)$, $\boldsymbol{u}^{(r)}(i,:)$, $\boldsymbol{\lambda}^{(r-1)}(i,:)$ follow the definition,

$$\boldsymbol{a}(i,:) = \begin{bmatrix} \boldsymbol{a}_1(i) & \boldsymbol{a}_2(i) & \ldots & \boldsymbol{a}_K(i) \end{bmatrix}^{\mathsf{T}} \in \mathbb{R}^K, \quad (26)$$

for any set $\{\boldsymbol{a}_k \in \mathbb{R}^m\}_{k=1}^{K}$ and $i = 1, 2, \ldots, K$. This modified version is referred to as *group ℓ_1-regularized RKS*.

IV. Bayesian Robust Kalman Smoothing

In this section, we present an alternative approach, called Bayesian RKS, to estimate the states and sparse inputs $\{\boldsymbol{x}_k, \boldsymbol{u}_k\}_{k=1}^{K}$ using measurements $\{\boldsymbol{y}_k\}_{k=1}^{K}$ in (2). Unlike the regularized RKS that assumes the knowledge of the parameters of the sparse input prior, the Bayesian approach uses hierarchical priors which account for the uncertainty in the prior distribution. Specifically, we use a hierarchical Gaussian prior on the inputs to promote sparsity:

$$p(\boldsymbol{u}_k; \boldsymbol{\gamma}_k) = \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\gamma_k(i)}} \exp\left(-\frac{\boldsymbol{u}_k(i)^2}{2\gamma_k(i)}\right), \quad (27)$$

where $\boldsymbol{\gamma}_k \in \mathbb{R}^m$ is the unknown hyperparameter of the distribution. The Bayesian RKS learns the hyperparameters from the measurements, unlike the regularized RKS. We present two Bayesian RKS variants: SBL-RKS and VB-RKS.

A. Sparse Bayesian Learning-based RKS

In the SBL framework, we first compute the ML estimate $\hat{\boldsymbol{\gamma}}_k^{\mathrm{ML}}$ of the hyperparameter as

$$\{\hat{\boldsymbol{\gamma}}_k^{\mathrm{ML}}\}_{k=1}^{K} = \underset{\substack{\boldsymbol{\gamma}_k \in \mathbb{R}_+^{m\times 1} \\ k=1,\ldots,K}}{\arg\max}\, p\left(\boldsymbol{Y}_1^K; \{\boldsymbol{\gamma}_k\}_{k=1}^{K}\right). \quad (28)$$

This article has been accepted for publication in IEEE Open Journal of Control Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/OJCSYS.2025.3624615

Using the estimate $\hat{\boldsymbol{\gamma}}_k^{\mathrm{ML}}$, we can estimate the states and inputs using the Kalman filtering and smoothing algorithm. For this, we note that (1) and (2) are equivalent to

$$\boldsymbol{\xi}_{k+1} = \begin{bmatrix} \tilde{\boldsymbol{A}}_k \\ \boldsymbol{0} \end{bmatrix} \boldsymbol{\xi}_k + \begin{bmatrix} \boldsymbol{w}_k \\ \boldsymbol{z}_k \end{bmatrix} \text{ and } \boldsymbol{y}_k = \begin{bmatrix} \boldsymbol{C}_k & \boldsymbol{D}_k \end{bmatrix} \boldsymbol{\xi}_k + \boldsymbol{v}_k, \tag{29}$$

where $\boldsymbol{\xi}_k$ is defined as

$$\boldsymbol{\xi}_k = \begin{bmatrix} \boldsymbol{x}_k^{\mathsf{T}} & \boldsymbol{u}_k^{\mathsf{T}} \end{bmatrix}^{\mathsf{T}}, \tag{30}$$

and $\boldsymbol{z}_k = \boldsymbol{u}_{k+1}$ is an auxiliary variable. From (27), we get

$$\begin{bmatrix} \boldsymbol{w}_k \\ \boldsymbol{z}_k \end{bmatrix} \sim \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix} \boldsymbol{Q}_k & \boldsymbol{0} \\ \boldsymbol{0} & \mathrm{Diag}\left\{\boldsymbol{\gamma}_{k+1}\right\} \end{bmatrix}\right).$$

Given $\{\hat{\boldsymbol{\gamma}}_k^{\mathrm{ML}}\}_{k=1}^K$, estimating the states and inputs reduces to estimating $\{\boldsymbol{\xi}_k\}_{k=1}^K$ using $\boldsymbol{Y}_1^K$ via Kalman filtering and smoothing, due to the Gaussian assumptions.

Since optimization problem in (28) lack a closed-form solution, we employ expectation-maximization (EM), an iterative method with the expectation (E) and maximization (M) steps. In the rth iteration, the E-step computes the expected log-likelihood function $\mathcal{Q}^{(r)}$ of $\{\boldsymbol{\gamma}_k\}_{k=1}^K$ with respect to $\{\boldsymbol{\xi}_k\}_{k=1}^K$ given data $\boldsymbol{Y}_1^K$ and the current estimate $\boldsymbol{\gamma}_k^{(r-1)}$ of the hyperparameter $\boldsymbol{\gamma}_k$ obtained in the previous iteration. The M-step maximizes the expected log-likelihood to obtain the new estimate of $\boldsymbol{\gamma}_k$. Further, from the state space model in (29), the distribution of the data is given by

$$p\left(\boldsymbol{Y}_1^K, \{\boldsymbol{\xi}_k\}_{k=1}^K; \{\boldsymbol{\gamma}_k\}_{k=1}^K\right) = \prod_{k=1}^K p(\boldsymbol{y}_k|\boldsymbol{\xi}_k) p\left(\boldsymbol{\xi}_k|\boldsymbol{\xi}_{k-1}; \boldsymbol{\gamma}_k\right),$$

where $\boldsymbol{\xi}_0 = \boldsymbol{0}$. Thus, the E-step is given by

$$\begin{aligned} \mathcal{Q}^{(r)} &\left(\{\boldsymbol{\gamma}_k\}_{k=1}^K \mid \left\{\boldsymbol{\gamma}_k^{(r-1)}\right\}_{k=1}^K\right) \\ &= \sum_{k=1}^K \mathbb{E}_{\boldsymbol{\xi}_k, \boldsymbol{\xi}_{k-1}|\boldsymbol{Y}_1^K; \boldsymbol{\gamma}_k^{(r-1)}} \Big\{ \log p(\boldsymbol{y}_k|\boldsymbol{\xi}_k) \\ &\qquad \times p\left(\boldsymbol{x}_k \mid \boldsymbol{\xi}_{k-1}\right) p\left(\boldsymbol{u}_k; \boldsymbol{\gamma}_k\right) \Big\}. \end{aligned}$$

From the above relation, the M-step that maximizes $\mathcal{Q}^{(r)}$ with respect to $\{\boldsymbol{\gamma}_k\}_{k=1}^K$ is separable, and ignoring the terms independent of $\boldsymbol{\gamma}_k$, the M-step reduces to

$$\boldsymbol{\gamma}_k^{(r+1)} = \arg\max_{\boldsymbol{\gamma}_k} \mathbb{E}_{\boldsymbol{u}_k|\boldsymbol{Y}_1^K; \boldsymbol{\gamma}_k^{(r-1)}} \left\{p\left(\boldsymbol{u}_k; \boldsymbol{\gamma}_k\right)\right\}.$$

Using (27), we derive the M-step as

$$\begin{aligned} \boldsymbol{\gamma}_k^{(r+1)} &= \underset{\boldsymbol{\Gamma}=\mathrm{Diag}\{\boldsymbol{\gamma}_k\}}{\arg\min} \log|\boldsymbol{\Gamma}| + \mathrm{Tr}\left\{\boldsymbol{\Gamma}^{-1}(\hat{\boldsymbol{u}}_{k|K}\hat{\boldsymbol{u}}_{k|K}^{\mathsf{T}} + \boldsymbol{P}_{k|K}^{\boldsymbol{u}})\right\} \\ &= \mathrm{Diag}\left\{\hat{\boldsymbol{u}}_{k|K}\hat{\boldsymbol{u}}_{k|K}^{\mathsf{T}} + \boldsymbol{P}_{k|K}^{\boldsymbol{u}}\right\}. \end{aligned}$$

Further, $\hat{\boldsymbol{u}}_{k|K}$ and $\boldsymbol{P}_{k|K}^{\boldsymbol{u}}$ can be computed by applying Kalman filtering and smoothing on the modified state space model in (29). The overall SBL-RKS algorithm is summarized in Algorithm 4.

Next, we establish a result similar to Proposition 1 for SBL-RKS. We show that the EM updates in Algorithm 4 are guaranteed to monotonically increase (or maintain) the data log-likelihood in each iteration. Hence, the sequence of objective function values over iterations converges.

Algorithm 4 RKS with Sparse Bayesian Learning

Inputs: $\{\boldsymbol{y}_k, \boldsymbol{A}_k, \boldsymbol{B}_k, \boldsymbol{C}_k, \boldsymbol{D}_k, \boldsymbol{Q}_k, \boldsymbol{R}_k\}_{k=1}^K$
Parameters: $r_{\max}$, $\epsilon_{\mathrm{thres}}$
Initialization: $\boldsymbol{\gamma}_k^{(0)} = \boldsymbol{1}$ for $k = 1, 2, \ldots, K$, $r = 1$, $\epsilon = 2\epsilon_{\mathrm{thres}}$
$\bar{\boldsymbol{A}}_k = \begin{bmatrix} \boldsymbol{A}_k & \boldsymbol{B}_k \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \in \mathbb{R}^{(n+m)\times(n+m)}$ and $\bar{\boldsymbol{C}}_k = \begin{bmatrix} \boldsymbol{C}_k & \boldsymbol{D}_k \end{bmatrix}$
while $(r < r_{max})$ or $(\epsilon > \epsilon_{\mathrm{thres}})$ **do**
#E-Step:
  $\hat{\boldsymbol{\xi}}_{0|0} = \boldsymbol{0}$, $\boldsymbol{P}_{0|0}^{\boldsymbol{\xi}} = \boldsymbol{I}$
  for $k = 1, 2, \ldots, K$ **do**
    $\bar{\boldsymbol{Q}}_{k-1} = \begin{bmatrix} \boldsymbol{Q}_{k-1} & \boldsymbol{0} \\ \boldsymbol{0} & \mathrm{Diag}\left\{\boldsymbol{\gamma}_k\right\} \end{bmatrix}$
#Prediction:
    $\hat{\boldsymbol{\xi}}_{k|k-1} = \bar{\boldsymbol{A}}_{k-1}\hat{\boldsymbol{\xi}}_{k-1|k-1}$
    $\boldsymbol{P}_{k|k-1}^{\boldsymbol{\xi}} = \bar{\boldsymbol{A}}_{k-1}\boldsymbol{P}_{k-1|k-1}^{\boldsymbol{\xi}}\bar{\boldsymbol{A}}_{k-1}^{\mathsf{T}} + \bar{\boldsymbol{Q}}_{k-1}$
#Filtering:
    $\boldsymbol{G}_k = \boldsymbol{P}_{k|k-1}^{\boldsymbol{\xi}}\bar{\boldsymbol{C}}_k^{\mathsf{T}}\left(\boldsymbol{R}_k + \bar{\boldsymbol{C}}_k\boldsymbol{P}_{k|k-1}^{\boldsymbol{\xi}}\bar{\boldsymbol{C}}_k^{\mathsf{T}}\right)^{-1}$
    $\hat{\boldsymbol{\xi}}_{k|k} = \hat{\boldsymbol{\xi}}_{k|k-1} + \boldsymbol{G}_k\left(\boldsymbol{y}_k - \bar{\boldsymbol{C}}_k\hat{\boldsymbol{\xi}}_{k|k-1}\right)$
    $\boldsymbol{P}_{k|k}^{\boldsymbol{\xi}} = \left(\boldsymbol{I} - \boldsymbol{G}_k\bar{\boldsymbol{C}}_k\right)\boldsymbol{P}_{k|k-1}^{\boldsymbol{\xi}}$
  end for
#Smoothing:
  for $k = K-1, K-2, \ldots, 1$ **do**
    $\boldsymbol{K}_k = \boldsymbol{P}_{k|k}^{\boldsymbol{\xi}}\bar{\boldsymbol{A}}_k^{\mathsf{T}}\left(\boldsymbol{P}_{k+1|k}^{\boldsymbol{\xi}}\right)^{-1}$
    $\boldsymbol{P}_{k|K}^{\boldsymbol{\xi}} = \boldsymbol{P}_{k|k}^{\boldsymbol{\xi}} + \boldsymbol{K}_k\left(\boldsymbol{P}_{k+1|K}^{\boldsymbol{\xi}} - \boldsymbol{P}_{k+1|k}^{\boldsymbol{\xi}}\right)\boldsymbol{K}_k^{\mathsf{T}}$
    $\hat{\boldsymbol{\xi}}_{k|K} = \hat{\boldsymbol{\xi}}_{k|k} + \boldsymbol{K}_k\left(\hat{\boldsymbol{\xi}}_{k+1|K} - \bar{\boldsymbol{A}}_k\hat{\boldsymbol{\xi}}_{k|k}\right)$
    $\boldsymbol{P}_{k+1,k|K}^{\boldsymbol{\xi}} = \begin{bmatrix} \boldsymbol{P}_{k+1|K}^{\boldsymbol{x}} & \boldsymbol{P}_{k+1|K}^{\boldsymbol{xu}} \end{bmatrix}^{\mathsf{T}}\boldsymbol{K}_k^{\mathsf{T}}$
    Compute $\hat{\boldsymbol{u}}_{k|K}$ and $\boldsymbol{P}_{k|K}^{\boldsymbol{u}}$ using (30) from $\hat{\boldsymbol{\xi}}_{k|K}$ and $\boldsymbol{P}_{k|K}^{\boldsymbol{\xi}}$
  end for
#M-step:
  $\boldsymbol{\gamma}_k^{(r)} = \mathrm{Diag}\left\{\hat{\boldsymbol{u}}_{k|K}\hat{\boldsymbol{u}}_{k|K}^{\mathsf{T}} + \boldsymbol{P}_{k|K}^{\boldsymbol{u}}\right\}$, for $k = 1, 2, \ldots, K$
  $\epsilon = \sum_{k=1}^K \left\|\boldsymbol{\gamma}_k^{(r)} - \boldsymbol{\gamma}_k^{(r-1)}\right\|_2^2$
  $r = r + 1$
end while
Compute $\{\hat{\boldsymbol{x}}_{k|K}, \hat{\boldsymbol{u}}_{k|K}\}_{k=1}^K$ using (30) from $\hat{\boldsymbol{\xi}}_{k|K}$
Outputs: $\{\hat{\boldsymbol{x}}_{k|K}\}_{k=1}^K$ and $\{\hat{\boldsymbol{u}}_{k|K}\}_{k=1}^K$

Proposition 2:
For a given set of inputs, let $\{\boldsymbol{\gamma}_k^{(r)} : k = 1, \ldots, K\}_{r=1}^\infty$ be the sequence generated by Algorithm 4 such that $\left\|\boldsymbol{\gamma}_k^{(r)}\right\|_2 < \infty$. Then, the sequence $\{\log p(\boldsymbol{Y}_1^K \mid \{\boldsymbol{\gamma}_k\}_{k=1}^K\}_{r=1}^\infty$ is monotonically nondecreasing and converges to a finite limit in $\mathbb{R}$.

Proof:
For notational brevity, we denote the observations by $\boldsymbol{y} = \boldsymbol{Y}_1^K$, the hidden variables by $\boldsymbol{z} = \{\boldsymbol{X}_1^K, \boldsymbol{U}_1^K\}$, and the hyperparameters by $\boldsymbol{\gamma} = \{\boldsymbol{\gamma}_k\}_1^K$. In the rth iteration of the SBL algorithm, we derive

$$\begin{aligned}\boldsymbol{\gamma}^{(r+1)} &= \arg\max_{\boldsymbol{\gamma}} \mathcal{Q}(\boldsymbol{\gamma} \mid \boldsymbol{\gamma}^{(r)}) \\ &= \arg\max_{\boldsymbol{\gamma}} \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z}\mid\boldsymbol{y},\boldsymbol{\gamma}^{(r)})}[\log p(\boldsymbol{y},\boldsymbol{z} \mid \boldsymbol{\gamma})].\end{aligned}$$

Then, by the properties of the EM algorithm [41, Section 3.2], the ML objective function in (28) is monotonically increasing,

$$\log p(\boldsymbol{y} \mid \boldsymbol{\gamma}^{(r)}) \le \log p(\boldsymbol{y} \mid \boldsymbol{\gamma}^{(r+1)}).$$

Further, (1) and (2), we derive

$$\tilde{\boldsymbol{y}}_K = \tilde{\boldsymbol{f}}(\boldsymbol{z}, \boldsymbol{W}_1^{K-1}) + \tilde{\boldsymbol{v}}_K,$$

where $\tilde{\boldsymbol{y}}_K = \begin{bmatrix}\boldsymbol{y}_1^\mathsf{T} & \boldsymbol{y}_2^\mathsf{T} & \dots & \boldsymbol{y}_K^\mathsf{T}\end{bmatrix}^T \in \mathbb{R}^{Kp}$ denotes the concatenated measurement vector. Likewise, $\tilde{\boldsymbol{v}}_K \in \mathbb{R}^{Kp}$ stacks the measurement noise. It is Gaussian with zero mean and covariance $\boldsymbol{R} = \mathrm{Blkdiag}(\boldsymbol{R}_1, \dots, \boldsymbol{R}_K)$, where the operator $\mathrm{Blkdiag}(\cdot)$ denotes a block diagonal matrix with its arguments as its block diagonal entries. The term $\tilde{\boldsymbol{f}}(\boldsymbol{z}, \boldsymbol{W}_1^{K-1})$ is a linear function of the initial state, inputs, and process noise. It is also Gaussian distributed with zero mean and covariance denoted by $\boldsymbol{\Delta}(\boldsymbol{\gamma}) \in \mathbb{R}^{Kp\times Kp}$, i.e., it is a function of $\boldsymbol{\gamma}$. Then, we derive

$$\begin{aligned}\log p(\boldsymbol{y} \mid \boldsymbol{\gamma}^{(r)}) &= -\frac{Kp}{2}\log(2\pi) - \frac{1}{2}\log|\boldsymbol{\Delta}(\boldsymbol{\gamma}) + \boldsymbol{R}| \\ &\quad - \frac{1}{2}\tilde{\boldsymbol{y}}_K^\mathsf{T}(\boldsymbol{\Delta}(\boldsymbol{\gamma}) + \boldsymbol{R})^{-1}\tilde{\boldsymbol{y}}_K \\ &\le -\frac{Kp}{2}\log(2\pi) - \frac{1}{2}\log|\boldsymbol{R}|,\end{aligned}$$

since $\boldsymbol{\Delta}(\boldsymbol{\gamma})$ is positive semi-definite and bounded. Moreover, since $\boldsymbol{R}$ is positive definite, the monotonically nondecreasing sequence $\{\log p(\boldsymbol{Y}_1^K \mid \{\boldsymbol{\gamma}_k\}_{k=1}^K\}_{r=1}^\infty$ is bounded from above and the sequence converges to a finite limit point. ■

Remark: When all the inputs are jointly sparse, we can use a common prior $\boldsymbol{u}_k \sim \mathcal{N}(\boldsymbol{0}, \mathrm{Diag}\{\boldsymbol{\gamma}\})$, i.e., $\boldsymbol{\gamma}_k = \boldsymbol{\gamma}$ for $k = 1, 2, \dots, K$. Then, the SBL-RKS for joint sparse input recovery is identical to Algorithm 4 except for Steps 5 and 19. The noise covariance in Step 5 changes to

$$\bar{\boldsymbol{Q}}_k = \begin{bmatrix}\boldsymbol{Q}_k & \boldsymbol{0} \\ \boldsymbol{0} & \mathrm{Diag}\{\boldsymbol{\gamma}\}\end{bmatrix}, k = 1, 2, \dots, K.$$

Similarly, the M-step in Step 19 is modified as

$$\boldsymbol{\gamma}^{(r+1)} = \frac{1}{K}\sum_{k=1}^K \mathrm{Diag}\left\{\hat{\boldsymbol{u}}_{k|K}\hat{\boldsymbol{u}}_{k|K}^\mathsf{T} + \boldsymbol{P}_{k|K}^{\boldsymbol{u}}\right\}.$$

The modified SBL-RKS for jointly sparse inputs is referred to as *multiple measurement vector SBL-RKS (MSBL-RKS)*.

The advantage of SBL is that the hierarchical Gaussian priors together with EM updates automatically learn the penalty weights associated with the relevant nonzero entries of the sparse vector. In contrast, regularization-based approaches require manual tuning of penalty parameters [42].

B. Variational Bayesian Robust Kalman Smoothing

In the variational Bayesian inference (VBI) approach, we employ a two-stage hierarchical prior. Specifically, we assume $\boldsymbol{\beta}_k(i) \sim \mathrm{Gamma}(a, b)$, where the precision hyperparameter $\boldsymbol{\beta}_k(i) = 1/\boldsymbol{\gamma}_k(i)$ in (27), and $\mathrm{Gamma}(a, b)$ is the Gamma distribution with shape parameter $a > 0$ and rate parameter $b > 0$, i.e.,

$$p(\boldsymbol{\beta}_k) = \prod_{i=1}^m \Gamma^{-1}(a)\, b^a(\boldsymbol{\beta}_k(i))^{a-1}\exp(-b\boldsymbol{\beta}_k(i)). \quad (31)$$

The VB-RKS algorithm estimates the set of unknown parameters $\mathcal{Z} = \left\{\boldsymbol{X}_1^K, \boldsymbol{U}_1^K, \{\boldsymbol{\beta}_k\}_{k=1}^K\right\}$ as the mean of their posterior distribution. However, the posterior distribution computation is intractable, and we approximate it using a family of factorized distributions:

$$p(\mathcal{Z}|\boldsymbol{Y}_1^K) \approx q(\mathcal{Z}) = \prod_{k=1}^K q_k^{\boldsymbol{x}}(\boldsymbol{x}_k) q_k^{\boldsymbol{u}}(\boldsymbol{u}_k) q_k^{\boldsymbol{\beta}}(\boldsymbol{\beta}_k),$$

where $q_k^{\boldsymbol{x}}(\cdot), q_k^{\boldsymbol{u}}(\cdot)$, and $q_k^{\boldsymbol{\beta}}(\cdot)$ are the marginal distributions of the latent variables $\boldsymbol{x}_k, \boldsymbol{u}_k$, and $\boldsymbol{\beta}_k$, respectively.

We seek the optimal distribution $q(\mathcal{Z})$ that minimizes the Kullback-Leibler (KL) divergence

$$\mathcal{F}(q(\mathcal{Z})) = \mathrm{KL}(q(\mathcal{Z})\|p(\mathcal{Z} \mid \boldsymbol{y})).$$

Now, the optimal marginal distribution is given by [43]

$$\begin{aligned}\ln q_k^{\boldsymbol{x}}(\boldsymbol{x}_k) &\propto \mathbb{E}_{q(\mathcal{Z}\setminus\boldsymbol{x}_k)}\left\{\ln p\left(\mathcal{Z}, \boldsymbol{Y}_1^K\right)\right\} \\ \ln q_k^{\boldsymbol{u}}(\boldsymbol{u}_k) &\propto \mathbb{E}_{q(\mathcal{Z}\setminus\boldsymbol{u}_k)}\left\{\ln p\left(\mathcal{Z}, \boldsymbol{Y}_1^K\right)\right\},\end{aligned}$$

where $\propto$ denotes the equality up to an additive constant and $\mathbb{E}_{q(\mathcal{Z}\setminus\boldsymbol{x}_k)}\{\cdot\}$ and $\mathbb{E}_{q(\mathcal{Z}\setminus\boldsymbol{u}_k)}\{\cdot\}$ are the expectations with respect to all the latent variables except $\boldsymbol{x}_k$ and $\boldsymbol{u}_k$, respectively. Further, we recall that

$$\begin{aligned}p(\mathcal{Z}, \boldsymbol{Y}_1^K) = \prod_{k=1}^K &p(\boldsymbol{y}_k|\boldsymbol{x}_k, \boldsymbol{u}_k)p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}) \\ &\times p(\boldsymbol{u}_k|\boldsymbol{\beta}_k)p(\boldsymbol{\beta}_k),\end{aligned}$$

where $p(\boldsymbol{u}_k|\boldsymbol{\beta}_k)$ and $p(\boldsymbol{\beta}_k)$ are given by (27) with $\boldsymbol{\beta}_k(i) = 1/\boldsymbol{\gamma}_k(i)$, and (31), respectively. Consequently, we arrive at

$$\begin{aligned}\ln q_k^{\boldsymbol{x}}(\boldsymbol{x}_k) \propto &\|\boldsymbol{y}_k - \boldsymbol{C}_k\boldsymbol{x}_k - \boldsymbol{D}_k\langle\boldsymbol{u}_k\rangle\|_{\boldsymbol{R}_k}^2 \\ &+ \|\langle\boldsymbol{x}_{k+1}\rangle - \boldsymbol{A}_k\boldsymbol{x}_k - \boldsymbol{B}_k\langle\boldsymbol{u}_k\rangle\|_{\boldsymbol{Q}_k}^2 \\ &+ \|\boldsymbol{x}_k - \boldsymbol{A}_{k-1}\langle\boldsymbol{x}_{k-1}\rangle - \boldsymbol{B}_{k-1}\langle\boldsymbol{u}_{k-1}\rangle\|_{\boldsymbol{Q}_{k-1}}^2,\end{aligned}$$

where $\langle\cdot\rangle$ denotes the mean of a random variable following the marginal distribution $q(\cdot)$. Hence, the marginal distribution $q_k^{\boldsymbol{x}}(\boldsymbol{x}_k)$ is Gaussian. Its mean can be computed by setting the gradient with respect to $\boldsymbol{x}_k$ to 0, leading to

$$\begin{aligned}\langle\boldsymbol{x}_k\rangle = \boldsymbol{P}_k^{\boldsymbol{x}}\Big[&\boldsymbol{C}_k^\mathsf{T}\boldsymbol{R}_k^{-1}\boldsymbol{y}_k + \boldsymbol{Q}_{k-1}^{-1}\boldsymbol{B}_{k-1}\langle\boldsymbol{u}_{k-1}\rangle \\ &- \left(\boldsymbol{C}_k^\mathsf{T}\boldsymbol{R}_k^{-1}\boldsymbol{D}_k + \boldsymbol{A}_k^\mathsf{T}\boldsymbol{Q}_k^{-1}\boldsymbol{B}_k\right)\langle\boldsymbol{u}_k\rangle \\ &+ \boldsymbol{Q}_{k-1}^{-1}\boldsymbol{A}_{k-1}\langle\boldsymbol{x}_{k-1}\rangle + \boldsymbol{A}_k^\mathsf{T}\boldsymbol{Q}_k^{-1}\langle\boldsymbol{x}_{k+1}\rangle\Big], \quad (32)\end{aligned}$$

This article has been accepted for publication in IEEE Open Journal of Control Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/OJCSYS.2025.3624615

where we define

$$\boldsymbol{P}_k^{\boldsymbol{x}} = \left(\boldsymbol{C}_k^{\mathsf{T}}\boldsymbol{R}_k^{-1}\boldsymbol{C}_k + \boldsymbol{Q}_{k-1}^{-1} + \boldsymbol{A}_k^{\mathsf{T}}\boldsymbol{Q}_k^{-1}\boldsymbol{A}_k\right)^{-1}. \quad (33)$$

Similarly, the marginal distribution of $\boldsymbol{u}_k$ is computed as

$$\begin{aligned}\ln q_k^{\boldsymbol{u}}(\boldsymbol{u}_k) &\propto \mathbb{E}_{q(\mathcal{Z}\setminus\boldsymbol{u}_k)}\left\{\ln p\left(\mathcal{Z}, \boldsymbol{Y}_1^K\right)\right\}\\ &\propto \|\boldsymbol{y}_k - \boldsymbol{C}_k\langle\boldsymbol{x}_k\rangle - \boldsymbol{D}_k\boldsymbol{u}_k\|_{\boldsymbol{R}_k}^2\\ &\quad + \|\boldsymbol{x}_{k+1} - \boldsymbol{A}_k\langle\boldsymbol{x}_k\rangle - \boldsymbol{B}_k\boldsymbol{u}_k\|_{\boldsymbol{Q}_k}^2 + \sum_{i=1}^{m}\boldsymbol{u}_k(i)^2\langle\boldsymbol{\beta}_k(i)\rangle.\end{aligned}$$

The mean of the Gaussian distribution $q_k^{\boldsymbol{u}}(\boldsymbol{x}_k)$ is

$$\begin{aligned}\langle\boldsymbol{u}_k\rangle = \boldsymbol{P}_k^{\boldsymbol{u}}\Big[\boldsymbol{D}_k^{\mathsf{T}}\boldsymbol{R}_k^{-1}\boldsymbol{y}_k - \left(\boldsymbol{D}_k^{\mathsf{T}}\boldsymbol{R}_k^{-1}\boldsymbol{C}_k + \boldsymbol{B}_k^{\mathsf{T}}\boldsymbol{Q}_k^{-1}\boldsymbol{A}_k\right)\\ \times\langle\boldsymbol{x}_k\rangle + \boldsymbol{B}_k^{\mathsf{T}}\boldsymbol{Q}_k^{-1}\langle\boldsymbol{x}_{k+1}\rangle\Big], \quad (34)\end{aligned}$$

where the matrix $\boldsymbol{P}_k^{\boldsymbol{u}}$ is

$$\boldsymbol{P}_k^{\boldsymbol{u}} = \left(\boldsymbol{D}_k^{\mathsf{T}}\boldsymbol{R}_k^{-1}\boldsymbol{D}_k + \boldsymbol{B}_k^{\mathsf{T}}\boldsymbol{Q}_k^{-1}\boldsymbol{B}_k + \langle\mathrm{diag}\left\{\boldsymbol{\beta}_k\right\}\rangle\right)^{-1}.$$

We use $\boldsymbol{x}_{K+1} = \boldsymbol{0}$ for $k = K$ and $\boldsymbol{x}_0 = \boldsymbol{0}$ for $k = 1$ in (32) and (34). Finally, $q(\boldsymbol{\beta}_k) = \prod_{i=1}^m q(\boldsymbol{\beta}_k(i))$ is a Gamma distribution with mean

$$\langle\boldsymbol{\beta}_k(i)\rangle = \frac{a+0.5}{b+0.5\langle\boldsymbol{u}_k^2(i)\rangle} = \frac{a+0.5}{b+0.5\left[\langle\boldsymbol{u}_k(i)\rangle^2 + \boldsymbol{P}_k^{\boldsymbol{u}}(i,i)\right]}. \quad (35)$$

Using (32), (34), and (35), the marginal distribution parameters are iteratively updated until convergence to obtain the approximate posterior distribution. The pseudocode is summarized in Algorithm 5.

Algorithm 5 Variational Bayesian RKS

Inputs: $\{\boldsymbol{y}_k, \boldsymbol{A}_k, \boldsymbol{B}_k, \boldsymbol{C}_k, \boldsymbol{D}_k, \boldsymbol{Q}_k, \boldsymbol{R}_k\}_{k=1}^K$

Parameters: $r_{\max}$ and $\tilde{r}_{\max}$

Initialization: $\langle\boldsymbol{x}_k\rangle = \boldsymbol{0}$, $\langle\boldsymbol{u}_k\rangle = \boldsymbol{0}$, $\langle\boldsymbol{\beta}_k\rangle = \boldsymbol{1}$ for $k = 1, 2, \ldots, K$

1: **for** $r = 1, 2, \ldots, r_{\max}$ **do**
2: **for** $\tilde{r} = 1, 2, \ldots, \tilde{r}_{\max}$ **do**
3: Compute $\boldsymbol{x}_k^{(r,\tilde{r})} = \langle\boldsymbol{x}_k\rangle$ using (32) for $k = 1, \ldots, K$
4: Compute $\boldsymbol{u}_k^{(r,\tilde{r})} = \langle\boldsymbol{u}_k\rangle$ using (34) for $k = 1, \ldots, K$
5: **end for**
6: Compute $\boldsymbol{\beta}_k^{(r)} = \langle\boldsymbol{\beta}_k\rangle$ using (35) for $k = 1, \ldots, K$
7: **end for**

Outputs: $\left\{\boldsymbol{x}_k^{(r,\tilde{r})}\right\}_{k=1}^K$ and $\left\{\boldsymbol{u}_k^{(r,\tilde{r})}\right\}_{k=1}^K$

The following result shows that Algorithm 5 monotonically decreases the KL divergence $\mathcal{F}(\mathcal{Z})$ at each iteration.

Proposition 3:
For a given set of inputs, the sequence $\{\mathcal{F}^{(r)}\}_{r=1}^{\infty}$ of KL divergence generated by Algorithm 5 is monotonically non-increasing and converges to a limit in $\mathbb{R}$.

Proof:

From Algorithm 5, let the rth iterate be $\mathcal{Z}^{(r)} = (\boldsymbol{x}^{(r,\tilde{r}_{\max})}, \boldsymbol{u}^{(r,\tilde{r}_{\max})}, \boldsymbol{\beta}^{(r)})$. Algorithm 5 proceeds by updating one set of parameters at a time, implying

$$\begin{aligned}\mathcal{F}^{(r)} &= \mathcal{F}(\mathcal{Z}^{(r)}) \geq \mathcal{F}(\boldsymbol{x}^{(r+1,1)}, \boldsymbol{u}^{(r,\tilde{r}_{\max})}, \boldsymbol{\beta}^{(r)})\\ &\geq \mathcal{F}(\boldsymbol{x}^{(r+1,1)}, \boldsymbol{u}^{(r+1,1)}, \boldsymbol{\beta}^{(r)})\\ &\geq \mathcal{F}(\boldsymbol{x}^{(r+1,\tilde{r}_{\max})}, \boldsymbol{u}^{(r+1,\tilde{r}_{\max})}, \boldsymbol{\beta}^{(r)})\\ &\geq \mathcal{F}(\boldsymbol{x}^{(r+1,\tilde{r}_{\max})}, \boldsymbol{u}^{(r+1,\tilde{r}_{\max})}, \boldsymbol{\beta}^{(r+1)}) = \mathcal{F}^{(r+1)}.\end{aligned}$$

Thus, the sequence $\{\mathcal{F}^{(r)}\}_{r=1}^{\infty}$ decreases monotonically, and since KL divergence is bounded below by 0, it converges. ■

We note that our result does not establish that the KL divergence converges to zero, but rather that it converges to a stable value. This indicates that the estimated state and input distributions have stabilized at the closest achievable approximation to the true distribution, though not necessarily identical to it.

Also, when all the inputs are jointly sparse, similar to SBL-RKS, we use a common prior $\boldsymbol{u}_k \sim \mathcal{N}(\boldsymbol{0}, \mathrm{Diag}\{\boldsymbol{\beta}\})$, i.e., $\boldsymbol{\beta}_k = \boldsymbol{\beta}$ for $k = 1, 2, \ldots, K$. In that case, the VB-RKS for joint sparse input recovery is identical to Algorithm 5 except that (35) in Step 6 changes as follows.

$$\langle\boldsymbol{\beta}(i)\rangle = \frac{a+0.5}{b+\frac{0.5}{K}\sum_{k=1}^K\langle\boldsymbol{u}_k^2(i)\rangle}.$$

We refer to this algorithm as multiple measurement vector VB-RKS (MVB-RKS).

Remark: A special case of our problem is the conventional KF problem when $\boldsymbol{u}_k = \boldsymbol{0}\ \forall k$. The update steps consist of only two equations (32) and (33) with $\boldsymbol{B}_k = \boldsymbol{0}$, $\boldsymbol{D}_k = \boldsymbol{0}$. The state update equation is given by,

$$\begin{aligned}\langle\boldsymbol{x}_k\rangle = \boldsymbol{P}_k^{\boldsymbol{x}}\Big[\boldsymbol{C}_k^{\mathsf{T}}\boldsymbol{R}_k^{-1}\boldsymbol{y}_k + \boldsymbol{Q}_{k-1}^{-1}\boldsymbol{A}_{k-1}\langle\boldsymbol{x}_{k-1}\rangle\\ + \boldsymbol{A}_k^{\mathsf{T}}\boldsymbol{Q}_k^{-1}\langle\boldsymbol{x}_{k+1}\rangle\Big].\end{aligned}$$

VB-RKS allows for simple updates of both states and inputs, whereas the other algorithms typically require running a Kalman filter or an RKS subroutine. Consequently, each iteration of VB-RKS (Step 3, 4, and 6) is computationally faster than those of the other iterative algorithms. However, in simulations (see Section V) we observe that VB-RKS requires significantly more iterations to converge, making it overall slower.

C. Complexity Comparisons

All four algorithms, ℓ_1-regularized RKS, reweighted ℓ_2-regularized RKS, SBL-RKS, and VB-RKS are iterative and uses the RKS algorithm in every iteration. Also, the RKS step is the most computationally complex step in these algorithms and dominates the overall complexity, leading to per-iteration time complexity of all the algorithms as $\mathcal{O}(K(n^3 + m^3 + p^3))$ for the versions with and without the joint sparsity assumption. Since the sparsity-driven algorithms consider low-dimensional measurements where $m \geq p$, the time complexity further reduces to $\mathcal{O}(K(n^3 + m^3))$.

Further, we have observed from our simulation results that the other algorithms require a larger number of iterations and a longer time to converge than SBL-RKS (see Table 1). In particular, VB-RKS takes a longer run time owing to the large number of iterations required for convergence. For comparison, we consider the state-of-the-art ℓ_1 minimization-based algorithm, referred to as basis pursuit (BP)-RKS (group BP-RKS for the joint support case). BP-RKS is a non-iterative algorithm whose complexity scales as $\mathcal{O}(K^{\frac{7}{2}}m^{\frac{3}{2}}p^2 + K(n^3+p^3))$ due to the convex programming optimization using the interior point method. So, our algorithms have low complexity order when the number of iterations is small. The auxiliary space (memory) complexity of all the algorithms is $\mathcal{O}(p^2+K(n^2+m^2))$. The total time complexities of all the algorithms are summarized in Table 2 for comparison.

V. Simulation Results

In this section, we present empirical results to demonstrate the superior performance of the algorithms that exploit sparsity. We choose the state dimension $n = 30$, the input dimension $m = 100$, the output dimension $p = 20$, and the number of time steps $K = 30$. The sparsity level of the input is $s = 5$, and the locations of s nonzero entries are chosen uniformly at random from the set $\{1, 2, \ldots, m\}$. Further, the nonzero entries are drawn independently from a normal distribution $\mathcal{N}(0, \sigma_u^2)$ with $\sigma_u = 5$. For the time-varying support case, we independently choose different supports for each time instant k, and for the jointly sparse case, we use the same support for all values of k. The entries of the time invariant system matrices $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$, and $\boldsymbol{D}$ and the initial state $\boldsymbol{x}_1$ are independently drawn from the standard normal distribution. Also, the process noise covariance $\boldsymbol{Q}$ and the measurement noise covariance $\boldsymbol{R}$ are the identity matrix and $\sigma_v^2\boldsymbol{I}$, respectively. Finally, σ_v is computed from the measurement SNR via the relation $\mathrm{SNR} = s\sigma_u^2/\sigma_v^2$.

For the above setting, we compare the performance of our algorithms: ℓ_1-regularized RKS, reweighted ℓ_2-regularized RKS, SBL-RKS, and VB-RKS for the time-varying support and jointly sparse cases. We also consider two benchmark approaches: basis pursuit (BP)-RKS and group BP-RKS, which are adapted from the algorithm in [34] (see Appendix B for details), and the RKS algorithm (Algorithm 1). The following metrics are used for comparison: normalized mean squared error (NMSE) in the state and input estimation, false support recovery rate (FSRR) for input estimation, and run time. The FSRR is the sum of the false alarm and missed detection rates of the support estimation. The results in Figs. 1 to 3 and Table 1 compare the algorithms' performance as a function of the measurement dimension p.

1) Comparison with RKS and an oracle baseline

Fig. 1 shows different algorithms' NMSE and FSRR performance for the joint sparsity case. We omit the time-varying support case due to space limitations; it can be found in [44, Fig. 1]. From Fig. 1, we infer that the conventional RKS algorithm has poor NMSE performance compared to the sparsity-driven approaches. This underscores the importance of exploiting sparsity to achieve low NMSE. RKS requires $p > m$ for the inverse to exist in Step 2 of Algorithm 1. Naturally, the algorithm fails in the low-dimensional measurement regime. Also, the NMSE of RKS is comparable to that of the sparsity-driven algorithms only when $p > m$, but the latter outperform RKS even in that regime.

We also compare performance against an oracle RKS baseline that assumes the knowledge of input supports. With the support set $\mathcal{S}_k$ of $\boldsymbol{u}_k$ known, the system can be represented with reduced matrices $\boldsymbol{A}_k, \boldsymbol{B}_k, (\boldsymbol{C}_k)_{\mathcal{S}_k}$, and $(\boldsymbol{D}_k)_{\mathcal{S}_k}$, driven by non-sparse inputs $(\boldsymbol{u}_k)_{\mathcal{S}_k}$. Then, oracle RKS estimates $(\boldsymbol{u}_k)_{\mathcal{S}_k}$. As expected, oracle RKS outperforms all other algorithms in both state and input NMSE performance, as shown in Fig. 1a and Fig. 1b, respectively. However, SBL-RKS and VB-RKS approach oracle-level performance in state estimation in the high-p regime.

TABLE 1: Run time comparison of algorithms when $n = 30$, $p = 20$, $m = 100$, $K = 30$, $s = 5$ and SNR is 20 dB

Support	Algorithm	Runtime
Time Varying	RKS	1.2 s
	BP-RKS	73.94 s
	ℓ_1-regularized RKS	33.76 s
	reweighted ℓ_2-regularized RKS	44.5 s
	SBL-RKS	14.5 s
	VB-RKS	∼5 min
Jointly sparse	Group BP-RKS	52 s
	Group ℓ_1-regularized RKS	34.6 s
	MSBL-RKS	13.5 s
	MVB-RKS	∼5 min

TABLE 2: Complexity comparison of algorithms

Type	Algorithm	Complexity
Non Iterative	RKS	$\mathcal{O}(K(n^3+m^3+p^3))$
	BP-RKS	$\mathcal{O}(K^{\frac{7}{2}}m^{\frac{3}{2}}p^2 + K(n^3+p^3))$
Iterative	ℓ_1-RKS	$\mathcal{O}(r_{\max}K(n^3+m^3+p^3))$
	ℓ_2-RKS	$\mathcal{O}(r_{\max}K(n^3+m^3+p^3))$
	SBL-RKS	$\mathcal{O}(r_{\max}K(n^3+m^3+p^3))$
	VB-RKS	$\mathcal{O}(r_{\max}\tilde{r}_{\max}K(n^3+m^3+p^3))$

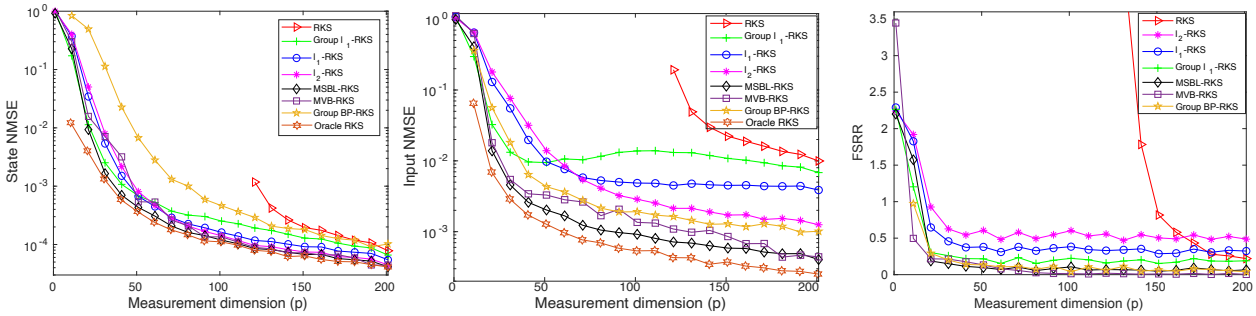
2) Comparison of Sparsity-driven Algorithms

Fig. 1 and Table 1 show that the SBL-RKS and VB-RKS algorithms outperform BP-RKS and regularized RKS in terms of NMSE in both states and input estimation, FSRR, and run time, in line with our arguments in Section IV.C. VB-RKS performs similar to SBL-RKS, except for runtime, which is much higher for the former algorithm.

3) Time-varying Support and Joint Sparsity

Fig. 2 and Table 1 show different algorithms' performances for the time-varying support and joint sparsity cases. Fig. 2

This article has been accepted for publication in IEEE Open Journal of Control Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/OJCSYS.2025.3624615



(a) NMSE in state estimation (b) NMSE in input estimation (c) Percentage false support recovery rate

FIGURE 1: Performance comparison of our sparse recovery algorithms and RKS as a function of measurement dimension p when the control inputs are jointly sparse with $n = 30$, $m = 100$, $K = 30$, $s = 5$, and SNR $= 20$ dB.

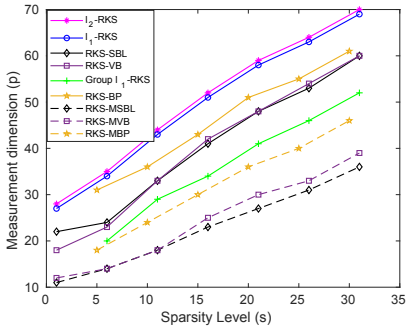
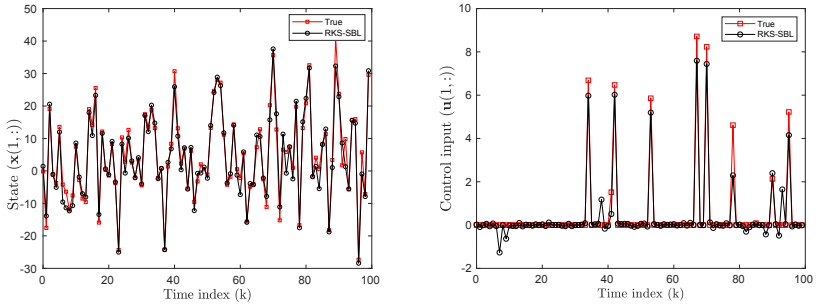


FIGURE 2: Phase transition diagram for our sparse recovery algorithms with $n = 30$, $m = 100$, $K = 30$, and SNR $= 20$ dB.



(a) Time tracking of state (b) Time tracking of input

FIGURE 3: Time domain tracking performance of SBL-RKS with $n = 30$, $m = 100$, $p = 20$, $K = 100$, $s = 5$, and SNR $= 20$ dB.

plots the minimum value of the number of measurements p required for 90% recovery accuracy (i.e., successful recovery of the sparse signals in 90% of the random experiments). Here, a sparse vector is said to be successfully recovered if the normalized mean square error between the original signal and the recovered signal is below 0.05. For the same number of measurements p, the NMSE in input estimation is better for the joint sparsity-aware algorithms: group BP-RKS, group ℓ_1-regularized RKS, MSBL-RKS, and MVB-RKS. This behavior is because of the additional joint sparse structure exploited by these algorithms. Clearly, the Bayesian RKS algorithms require the least number of measurements, followed by group BP-RKS and group ℓ_1-regularized RKS. The joint sparsity-aware algorithms are followed by Bayesian-RKS, BP-RKS, and regularized RKS. The regularized RKS algorithms have similar phase transition curves, but SBL-RKS and VB-RKS require fewer measurements than the regularized RKS.

Finally, Table 1 indicates that joint sparsity-aware algorithms have a shorter run time than their counterparts for the time-varying support case because they have fewer parameters to estimate.

4) Time domain tracking of states and inputs

The time domain state and input tracking performance of SBL-RKS is shown in Fig. 3 for visual comparison. Here, the estimate of one entry in the state and input vectors across time obtained using SBL-RKS are depicted in Fig. 3a and in Fig. 3b, respectively. It is clear from Fig. 3b that SBL-RKS is able to recover the spikes in the entries of input vectors accurately. Also, Fig. 3a shows that SBL-RKS tracks the state tightly. We observe similar results for all the other sparsity-aware algorithms, and hence, we omit plotting them to avoid clutter.

VI. Conclusion

In this paper, we studied the joint estimation of states and sparse inputs as an observer design problem in an LDS. We developed novel algorithms using fictitious sparsity-promoting priors, integrating sparse signal recovery techniques within the Kalman smoothing framework to create sparsity-aware Kalman smoothers. Our methods include a regularization-based MAP estimation approach and a hierarchical Bayesian learning framework built on Gaussian priors. Empirical results show that exploiting sparsity improves estimation and enables recovery with fewer measurements

than conventional methods. Among the proposed methods, SBL-RKS is preferred for its low complexity and superior accuracy. We also extended our approaches for the jointly sparse input case and demonstrated the efficacy of exploiting additional structures with sparsity. We also guarantee convergence of the cost functions of our algorithms to a local minimum or saddle point. However, convergence of the iterates is not ensured due to the combined Kalman smoothing and ADMM, EM, or VBI framework, which needs future study. Investigating the fundamental limits of sparse recovery in LDS is an interesting direction for further research. Future work can also extend our algorithms to handle system identification and estimation of the system matrices.

Appendix A
Proof of Theorem 1: The Kalman Filtering and Smoothing Steps

A. Prediction and Filtering steps

For detailed derivation of step 2-11 see [44]. The filtered estimates of states and inputs are mathematically equivalent to the estimator derived in [20]; hence, we omit it for brevity, although the approach in [44] differs from that in [20].

B. Smoothing steps

We develop the smoothing updates for the case of unknown inputs along the lines of the Kalman smoothing in [45]. The smoothed posterior distribution $p\left(\boldsymbol{x}_k, \boldsymbol{u}_k \mid \boldsymbol{Y}_1^K\right)$ is Gaussian with mean $\hat{\boldsymbol{x}}_{k|K}, \hat{\boldsymbol{u}}_{k|K}$ and covariance $\boldsymbol{P}_{k|K}^{\boldsymbol{\xi}}$ as derived below. The joint posterior distribution is

$$\begin{aligned} & p\left(\boldsymbol{\xi}_{k+1}, \boldsymbol{\xi}_k \mid \boldsymbol{Y}_1^K\right) \\ & = p\left(\boldsymbol{\xi}_k \mid \boldsymbol{\xi}_{k+1}, \boldsymbol{Y}_1^k\right) p\left(\boldsymbol{\xi}_{k+1} \mid \boldsymbol{Y}_1^K\right) \qquad (36) \\ & = \frac{p\left(\boldsymbol{\xi}_{k+1} \mid \boldsymbol{\xi}_k, \boldsymbol{Y}_1^k\right) p\left(\boldsymbol{\xi}_k, \boldsymbol{Y}_1^k\right)}{p\left(\boldsymbol{\xi}_{k+1}, \boldsymbol{Y}_1^k\right)} p\left(\boldsymbol{\xi}_{k+1} \mid \boldsymbol{Y}_1^K\right) \\ & = \frac{p\left(\boldsymbol{\xi}_{k+1} \mid \boldsymbol{\xi}_k\right) p\left(\boldsymbol{\xi}_k \mid \boldsymbol{Y}_1^k\right)}{p\left(\boldsymbol{\xi}_{k+1} \mid \boldsymbol{Y}_1^k\right)} p\left(\boldsymbol{\xi}_{k+1} \mid \boldsymbol{Y}_1^K\right), \qquad (37) \end{aligned}$$

where (36) follows since $\{\boldsymbol{y}_{k+1}, \ldots, \boldsymbol{y}_K\}$ are linear combinations of $\boldsymbol{\xi}_{k+1}$ and other random variables $\left(\boldsymbol{U}_{k+1}^K, \boldsymbol{W}_{k+1}^K, \boldsymbol{V}_{k+1}^K\right)$ which are independent of $\boldsymbol{\xi}_{k+1}$. Also, (37) follows due to the Markovian nature of the dynamics in (29).

$$\begin{aligned} & p\left(\boldsymbol{\xi}_{k+1}, \boldsymbol{\xi}_k \mid \boldsymbol{Y}_1^K\right) \\ & = \frac{p\left(\boldsymbol{x}_{k+1} \mid \boldsymbol{x}_k, \boldsymbol{u}_k\right) p\left(\boldsymbol{u}_{k+1}\right) p\left(\boldsymbol{\xi}_k \mid \boldsymbol{Y}_1^k\right)}{p\left(\boldsymbol{x}_{k+1} \mid \boldsymbol{Y}_1^k\right) p\left(\boldsymbol{u}_{k+1}\right)} p\left(\boldsymbol{\xi}_{k+1} \mid \boldsymbol{Y}_1^K\right) \qquad (38) \\ & = \frac{p\left(\boldsymbol{x}_{k+1} \mid \boldsymbol{x}_k, \boldsymbol{u}_k\right) p\left(\boldsymbol{\xi}_k \mid \boldsymbol{Y}_1^k\right)}{p\left(\boldsymbol{x}_{k+1} \mid \boldsymbol{Y}_1^k\right)} p\left(\boldsymbol{\xi}_{k+1} \mid \boldsymbol{Y}_1^K\right). \qquad (39) \end{aligned}$$

Here, (38) follows since $\boldsymbol{u}_{k+1}$ is assumed to be independent of $\boldsymbol{x}_k$, $\boldsymbol{x}_{k+1}$, $\boldsymbol{u}_k$ and all past observations $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k\}$.

Taking the logarithm of each term in (39), we have

$$\begin{aligned} \log p\left(\boldsymbol{x}_{k+1} \mid \boldsymbol{x}_k, \boldsymbol{u}_k\right) &= -\frac{1}{2}\|\boldsymbol{x}_{k+1} - \boldsymbol{A}_k \boldsymbol{x}_k - \boldsymbol{B}_k \boldsymbol{u}_k\|_{\boldsymbol{Q}_k}^2 \\ &= -\frac{1}{2}\|\boldsymbol{T}\boldsymbol{\xi}_{k+1} - \tilde{\boldsymbol{A}}_k \boldsymbol{\xi}_k\|_{\boldsymbol{Q}_k}^2, \end{aligned}$$

where we used $\boldsymbol{x}_{k+1} = \boldsymbol{T}\boldsymbol{\xi}_{k+1}$ with $\boldsymbol{T}$ and $\tilde{\boldsymbol{A}}_k$ defined in Algorithm 1. Similarly, we can show that

$$\begin{aligned} \log p\left(\boldsymbol{x}_{k+1} \mid \boldsymbol{Y}_1^k\right) &= -\frac{1}{2}\|\boldsymbol{x}_{k+1} - \hat{\boldsymbol{x}}_{k+1|k}\|_{\boldsymbol{P}_{k+1|k}^{\boldsymbol{x}}}^2 \\ &= -\frac{1}{2}\|\boldsymbol{T}\left(\boldsymbol{\xi}_{k+1} - \hat{\boldsymbol{\xi}}_{k+1|k}\right)\|_{\boldsymbol{P}_{k+1|k}^{\boldsymbol{x}}}^2 \\ \log p\left(\boldsymbol{\xi}_k \mid \boldsymbol{Y}_1^k\right) &= -\frac{1}{2}\|\boldsymbol{\xi}_k - \hat{\boldsymbol{\xi}}_{k|k}\|_{\boldsymbol{P}_{k|k}^{\boldsymbol{\xi}}}^2 \\ \log p\left(\boldsymbol{\xi}_{k+1} \mid \boldsymbol{Y}_1^K\right) &= -\frac{1}{2}\|\boldsymbol{\xi}_{k+1} - \hat{\boldsymbol{\xi}}_{k+1|K}\|_{\boldsymbol{P}_{k+1|K}^{\boldsymbol{\xi}}}^2. \end{aligned}$$

Using the above four relations in (39), we derive

$$\begin{aligned} \log p(\boldsymbol{\xi}_{k+1}, \boldsymbol{\xi}_k \mid \boldsymbol{Y}_1^K) = & -\frac{1}{2}\boldsymbol{\xi}_{k+1}^{\mathsf{T}}\Big[\boldsymbol{T}^{\mathsf{T}}\boldsymbol{Q}_k^{-1}\boldsymbol{T} \\ & - \boldsymbol{T}^{\mathsf{T}}\left(\boldsymbol{P}_{k+1|k}^{\boldsymbol{x}}\right)^{-1}\boldsymbol{T} + \left(\boldsymbol{P}_{k+1|K}^{\boldsymbol{\xi}}\right)^{-1}\Big]\boldsymbol{\xi}_{k+1} \\ & + \frac{1}{2}\boldsymbol{\xi}_{k+1}^{\mathsf{T}}\boldsymbol{T}^{\mathsf{T}}\boldsymbol{Q}_k^{-1}\tilde{\boldsymbol{A}}_k\boldsymbol{\xi}_k + \frac{1}{2}\boldsymbol{\xi}_k^{\mathsf{T}}\tilde{\boldsymbol{A}}_k^{\mathsf{T}}\boldsymbol{Q}_k^{-1}\boldsymbol{T}\boldsymbol{\xi}_{k+1} \\ & - \frac{1}{2}\boldsymbol{\xi}_k^{\mathsf{T}}\left[\tilde{\boldsymbol{A}}_k^{\mathsf{T}}\boldsymbol{Q}_k^{-1}\tilde{\boldsymbol{A}}_k + \left(\boldsymbol{P}_{k|k}^{\boldsymbol{\xi}}\right)^{-1}\right]\boldsymbol{\xi}_k + \boldsymbol{\xi}_k^{\mathsf{T}}\left(\boldsymbol{P}_{k|k}^{\boldsymbol{\xi}}\right)^{-1}\hat{\boldsymbol{\xi}}_{k|k} \\ & + \text{linear and constant terms.} \end{aligned}$$

Next, we follow the proof technique for deriving the smoothing updates in [45], and we use same notation $\boldsymbol{S}_{11}$, $\boldsymbol{S}_{12}$, $\boldsymbol{S}_{21}$, $\boldsymbol{S}_{22}$ and $\boldsymbol{F}_{11}$ as given in the reference which indicate same variables in our context. In our case, $\boldsymbol{S}_{22}^{-1}$ is

$$\begin{aligned} \boldsymbol{S}_{22}^{-1} &= \left(\tilde{\boldsymbol{A}}_k^{\mathsf{T}}\boldsymbol{Q}_k^{-1}\tilde{\boldsymbol{A}}_k + \left(\boldsymbol{P}_{k|k}^{\boldsymbol{\xi}}\right)^{-1}\right)^{-1} \\ &= \boldsymbol{P}_{k|k}^{\boldsymbol{\xi}} - \boldsymbol{P}_{k|k}^{\boldsymbol{\xi}}\tilde{\boldsymbol{A}}_k^{\mathsf{T}}\left(\boldsymbol{Q}_k + \tilde{\boldsymbol{A}}_k\boldsymbol{P}_{k|k}^{\boldsymbol{\xi}}\tilde{\boldsymbol{A}}_k^{\mathsf{T}}\right)^{-1}\tilde{\boldsymbol{A}}_k\boldsymbol{P}_{k|k}^{\boldsymbol{\xi}} \\ &= \boldsymbol{P}_{k|k}^{\boldsymbol{\xi}} - \boldsymbol{P}_{k|k}^{\boldsymbol{\xi}}\tilde{\boldsymbol{A}}_k^{\mathsf{T}}\left(\boldsymbol{P}_{k+1|k}^{\boldsymbol{x}}\right)^{-1}\tilde{\boldsymbol{A}}_k\boldsymbol{P}_{k|k}^{\boldsymbol{\xi}} \\ &= \boldsymbol{P}_{k|k}^{\boldsymbol{\xi}} - \boldsymbol{K}_k\boldsymbol{P}_{k+1|k}^{\boldsymbol{x}}\boldsymbol{K}_k^{\mathsf{T}}, \end{aligned}$$

where we define

$$\boldsymbol{K}_k = \boldsymbol{P}_{k|k}^{\boldsymbol{\xi}}\tilde{\boldsymbol{A}}_k^{\mathsf{T}}\left(\boldsymbol{P}_{k+1|k}^{\boldsymbol{x}}\right)^{-1}.$$

Similarly, $\boldsymbol{S}_{21} = -\tilde{\boldsymbol{A}}_k^{\mathsf{T}}\boldsymbol{Q}_k^{-1}\boldsymbol{T}$, and $\boldsymbol{S}_{22}^{-1}\boldsymbol{S}_{21}$ simplifies to

$$\begin{aligned} \boldsymbol{S}_{22}^{-1}\boldsymbol{S}_{21} &= -\Big(\boldsymbol{P}_{k|k}^{\boldsymbol{\xi}} - \boldsymbol{P}_{k|k}^{\boldsymbol{\xi}}\tilde{\boldsymbol{A}}_k^{\mathsf{T}}\left(\boldsymbol{Q}_k + \tilde{\boldsymbol{A}}_k\boldsymbol{P}_{k|k}^{\boldsymbol{\xi}}\tilde{\boldsymbol{A}}_k^{\mathsf{T}}\right)^{-1} \\ &\qquad \tilde{\boldsymbol{A}}_k\boldsymbol{P}_{k|k}^{\boldsymbol{\xi}}\Big)\tilde{\boldsymbol{A}}_k^{\mathsf{T}}\boldsymbol{Q}_k^{-1}\boldsymbol{T} \\ &= -\boldsymbol{P}_{k|k}^{\boldsymbol{\xi}}\tilde{\boldsymbol{A}}_k^{\mathsf{T}}\Big(\boldsymbol{I}_n - \left(\boldsymbol{Q}_k + \tilde{\boldsymbol{A}}_k\boldsymbol{P}_{k|k}^{\boldsymbol{\xi}}\tilde{\boldsymbol{A}}_k^{\mathsf{T}}\right)^{-1} \\ &\qquad \tilde{\boldsymbol{A}}_k\boldsymbol{P}_{k|k}^{\boldsymbol{\xi}}\tilde{\boldsymbol{A}}_k^{\mathsf{T}}\Big)\boldsymbol{Q}_k^{-1}\boldsymbol{T} \\ &= -\boldsymbol{P}_{k|k}^{\boldsymbol{\xi}}\tilde{\boldsymbol{A}}_k^{\mathsf{T}}\left(\boldsymbol{Q}_k + \tilde{\boldsymbol{A}}_k\boldsymbol{P}_{k|k}^{\boldsymbol{\xi}}\tilde{\boldsymbol{A}}_k^{\mathsf{T}}\right)^{-1}\boldsymbol{T} \\ &= -\boldsymbol{P}_{k|k}^{\boldsymbol{\xi}}\tilde{\boldsymbol{A}}_k^{\mathsf{T}}\left(\boldsymbol{P}_{k+1|k}^{\boldsymbol{x}}\right)^{-1}\boldsymbol{T} = -\boldsymbol{K}_k\boldsymbol{T}. \end{aligned}$$

This article has been accepted for publication in IEEE Open Journal of Control Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/OJCSYS.2025.3624615

Hence, the covariance update is given by (see (19) of [45]),

$$\begin{aligned}\boldsymbol{P}^{\boldsymbol{\xi}}_{k|K} &= \boldsymbol{S}_{22}^{-1} + \boldsymbol{S}_{22}^{-1}\boldsymbol{S}_{21}\boldsymbol{F}_{11}^{-1}\boldsymbol{S}_{12}\boldsymbol{S}_{22}^{-1} \\ &= \left(\boldsymbol{P}^{\boldsymbol{\xi}}_{k|k} - \boldsymbol{K}_k\boldsymbol{P}^{\boldsymbol{x}}_{k+1|k}\boldsymbol{K}_k^{\mathsf{T}}\right) + (-\boldsymbol{K}_k\boldsymbol{T})\,\boldsymbol{P}^{\boldsymbol{\xi}}_{k+1|K}\,(-\boldsymbol{K}_k\boldsymbol{T})^{\mathsf{T}} \\ &= \boldsymbol{P}^{\boldsymbol{\xi}}_{k|k} + \boldsymbol{K}_k\left(\boldsymbol{P}^{\boldsymbol{x}}_{k+1|K} - \boldsymbol{P}^{\boldsymbol{x}}_{k+1|k}\right)\boldsymbol{K}_k^{\mathsf{T}},\end{aligned}$$

where we used $\boldsymbol{T}\boldsymbol{P}^{\boldsymbol{\xi}}_{k+1|K}\boldsymbol{T}^{\mathsf{T}} = \boldsymbol{P}^{\boldsymbol{x}}_{k+1|K}$ in the last step. Hence, we prove Step 14 of Algorithm 1.

Finally, we can compute $\boldsymbol{P}^{\boldsymbol{\xi}}_{k+1,k|K}$ as (see (20) in [45])

$$\begin{aligned}\boldsymbol{P}^{\boldsymbol{\xi}}_{k+1,k|K} &= -\boldsymbol{F}_{11}^{-1}\boldsymbol{S}_{12}\boldsymbol{S}_{22}^{-1} = -\boldsymbol{P}^{\boldsymbol{\xi}}_{k+1|K}\,(-\boldsymbol{K}_k\boldsymbol{T})^{\mathsf{T}} \\ &= \begin{bmatrix}\boldsymbol{P}^{\boldsymbol{x}}_{k+1|K}\boldsymbol{K}_k^{\mathsf{T}} \\ \left(\boldsymbol{P}^{\boldsymbol{xu}}_{k+1|K}\right)^{\mathsf{T}}\boldsymbol{K}_k^{\mathsf{T}}\end{bmatrix}.\end{aligned}$$

We can find the smoothed posterior estimate using [45, Equation (23)], which leads to $\hat{\boldsymbol{\xi}}_{k|K} = -\boldsymbol{S}_{22}^{-1}\boldsymbol{S}_{21}\hat{\boldsymbol{\xi}}_{k+1|K} + \boldsymbol{S}_{22}^{-1}\left(\boldsymbol{P}^{\boldsymbol{\xi}}_{k|k}\right)^{-1}\hat{\boldsymbol{\xi}}_{k|k}$ and simplifies to

$$\hat{\boldsymbol{\xi}}_{k|K} = \boldsymbol{K}_k\hat{\boldsymbol{x}}_{k+1|K} + \left(\boldsymbol{I}_{n+m} - \boldsymbol{K}_k\tilde{\boldsymbol{A}}_k\right)\hat{\boldsymbol{\xi}}_{k|k}.$$

Thus, we prove Step 15 of Algorithm 1, and it completes proof of all the steps in the algorithm.

Appendix B
Derivation of BP-RKS and Group BP-RKS

In this section, we derive two benchmark algorithms inspired by [34], namely BP-RKS for the time-varying support case, and its extension to the joint sparsity case. To derive BP-RKS, we first consider the problem of estimating the initial state $\boldsymbol{x}_1$ and inputs $\boldsymbol{U}_1^K$ for the linear system defined in (1) and (2), which can be written as

$$\tilde{\boldsymbol{y}}_K = \boldsymbol{O}_K\boldsymbol{x}_1 + \boldsymbol{\Gamma}_K\tilde{\boldsymbol{u}}_K + \boldsymbol{M}_K\tilde{\boldsymbol{w}}_{K-1} + \tilde{\boldsymbol{v}}_K, \tag{40}$$

where $\tilde{\boldsymbol{y}}_K = \begin{bmatrix}\boldsymbol{y}_1^{\mathsf{T}} & \boldsymbol{y}_2^{\mathsf{T}} & \dots & \boldsymbol{y}_K^{\mathsf{T}}\end{bmatrix}^T \in \mathbb{R}^{Kp}$ denotes the concatenated measurement vector. Likewise, $\tilde{\boldsymbol{u}}_K \in \mathbb{R}^{Km}, \tilde{\boldsymbol{w}}_{K-1} \in \mathbb{R}^{(K-1)n}$, and $\tilde{\boldsymbol{v}}_K \in \mathbb{R}^{Kp}$ are obtained by concatenating the inputs, process noise terms and measurement noise terms, respectively. The system matrices, $\boldsymbol{O}_K \in \mathbb{R}^{Kp\times n}$, $\boldsymbol{\Gamma}_K \in \mathbb{R}^{Kp\times Km}$ and $\boldsymbol{M}_K \in \mathbb{R}^{Kp\times(K-1)n}$ in (40) are given by

$$\boldsymbol{O}_K = \begin{bmatrix}\boldsymbol{C} \\ \boldsymbol{CA} \\ \vdots \\ \boldsymbol{CA}^{K-1}\end{bmatrix},\ \boldsymbol{M}_K = \begin{bmatrix}\boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{C} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{CA}^{K-2} & \boldsymbol{CA}^{K-3} & \cdots & \boldsymbol{C}\end{bmatrix} \tag{41}$$

$$\boldsymbol{\Gamma}_K = \begin{bmatrix}\boldsymbol{D} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{CB} & \boldsymbol{D} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{CAB} & \boldsymbol{CB} & \boldsymbol{D} & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & & \\ \boldsymbol{CA}^{K-2}\boldsymbol{B} & \boldsymbol{CA}^{K-3}\boldsymbol{B} & \boldsymbol{CA}^{K-4}\boldsymbol{B} & \cdots & \boldsymbol{D}\end{bmatrix}. \tag{42}$$

For notational simplicity, we present (41) and (42) for the constant system matrices case, i.e., $\boldsymbol{A}_k = \boldsymbol{A}$, $\boldsymbol{B}_k = \boldsymbol{B}$, $\boldsymbol{D}_k = \boldsymbol{C}$, and $\boldsymbol{D}_k = \boldsymbol{D}$, $k = 1, 2, \dots, K$. However, the extension to time-varying matrices is straightforward.

Similar to the approach in [34], we first eliminate the initial state term in (40) by multiplying it with $\boldsymbol{\Pi}$, the projection matrix onto the orthogonal complement of the column space of the observability matrix $\boldsymbol{O}_K$, to get

$$\boldsymbol{\Pi}\tilde{\boldsymbol{y}}_K = \boldsymbol{\Pi}\boldsymbol{\Gamma}_K\tilde{\boldsymbol{u}}_K + \boldsymbol{\Pi}\tilde{\boldsymbol{n}}_K, \tag{43}$$

where the noise term $\tilde{\boldsymbol{n}}_K = \boldsymbol{M}_K\tilde{\boldsymbol{w}}_{K-1} + \tilde{\boldsymbol{v}}_K$ and the projection matrix is

$$\boldsymbol{\Pi} = \boldsymbol{I} - \boldsymbol{O}_K(\boldsymbol{O}_K^T\boldsymbol{O}_K)^{-1}\boldsymbol{O}_K^T, \tag{44}$$

where $\boldsymbol{O}_K$ is assumed to have full rank n.

Now, we estimate the states and inputs in two steps, using a Laplacian prior on inputs to encourage sparsity. We first solve for inputs $\tilde{\boldsymbol{u}}_K$ using (43). Substituting the estimate $\tilde{\boldsymbol{u}}_K^*$ into (40), we then compute the weighted least square estimate $\boldsymbol{x}_1^*$ as

$$\boldsymbol{x}_1^* = (\boldsymbol{O}_K^{\mathsf{T}}\boldsymbol{Q}_{\tilde{n}}^{-1}\boldsymbol{O}_K)^{-1}\boldsymbol{O}_K^{\mathsf{T}}\boldsymbol{Q}_{\tilde{n}}^{-1}(\tilde{\boldsymbol{y}}_K - \boldsymbol{\Gamma}_K\tilde{\boldsymbol{u}}_K^*). \tag{45}$$

Here, $\tilde{\boldsymbol{Q}}_K$ is the covariance of the noise $\tilde{\boldsymbol{n}}$ in (40), given by

$$\begin{aligned}\tilde{\boldsymbol{Q}}_K = \boldsymbol{M}_K\,&\mathrm{Blkdiag}(\boldsymbol{Q}_1, \dots, \boldsymbol{Q}_K)\boldsymbol{M}_K^{\mathsf{T}} \\ &+ \mathrm{Blkdiag}(\boldsymbol{R}_1, \dots, \boldsymbol{R}_K).\end{aligned} \tag{46}$$

Having estimated the initial state and inputs, the states $\boldsymbol{X}_2^K$ can be reconstructed using the Kalman smoothing algorithm.

To estimate the sparse inputs $\tilde{\boldsymbol{u}}_K$ from (43), we note that the projection step makes $\boldsymbol{\Pi}\boldsymbol{\Gamma}_K$ rank deficient. This is because from (44), we get $\mathrm{rank}(\boldsymbol{\Pi}) = Kp - n$ as $\mathrm{rank}(\boldsymbol{O}_K) = n$. We further reduce the system in (43) to get linearly independent measurements. Denote the singular value decomposition of the matrix $\boldsymbol{\Pi}\boldsymbol{\Gamma}_K$ by

$$\boldsymbol{\Pi}\boldsymbol{\Gamma}_K = \begin{bmatrix}\boldsymbol{\Psi}_1 \\ \boldsymbol{\Psi}_2\end{bmatrix}^{\mathsf{H}}\begin{bmatrix}\boldsymbol{\Lambda} \in \mathbb{R}^{R\times R} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0}\end{bmatrix}\begin{bmatrix}\boldsymbol{\Phi}_1 \\ \boldsymbol{\Phi}_2\end{bmatrix} = \boldsymbol{\Psi}_1^{\mathsf{H}}\boldsymbol{\Lambda}\boldsymbol{\Phi}_1, \tag{47}$$

where $\boldsymbol{\Psi} = \begin{bmatrix}\boldsymbol{\Psi}_1^{\mathsf{H}} & \boldsymbol{\Psi}_2^{\mathsf{H}}\end{bmatrix}^{\mathsf{H}}$ is an orthonormal matrix and R is the rank of $\boldsymbol{\Pi}\boldsymbol{\Gamma}_K$. When the system matrices have full rank, we have $R = \min\{Kp - n, Km\}$. From (43), we derive

$$\boldsymbol{\Psi}\boldsymbol{\Pi}\tilde{\boldsymbol{y}}_K = \begin{bmatrix}\boldsymbol{\Psi}_1\boldsymbol{\Pi}\tilde{\boldsymbol{y}}_K \\ \boldsymbol{\Psi}_2\boldsymbol{\Pi}\tilde{\boldsymbol{y}}_K\end{bmatrix} = \begin{bmatrix}\boldsymbol{\Lambda}\boldsymbol{\Phi}_1 \\ \boldsymbol{0}\end{bmatrix}\tilde{\boldsymbol{u}}_K + \begin{bmatrix}\boldsymbol{\Psi}_1\boldsymbol{\Pi}\tilde{\boldsymbol{n}}_K \\ \boldsymbol{\Psi}_2\boldsymbol{\Pi}\tilde{\boldsymbol{n}}_K\end{bmatrix}. \tag{48}$$

Hence, the reduced system of equations is

$$\boldsymbol{\Psi}_1\boldsymbol{\Pi}\tilde{\boldsymbol{y}}_K = \boldsymbol{\Psi}_1\boldsymbol{\Pi}\boldsymbol{\Gamma}_K\tilde{\boldsymbol{u}}_K + \boldsymbol{\Psi}_1\boldsymbol{\Pi}\tilde{\boldsymbol{n}}_K, \tag{49}$$

where the covariance of the noise component $\boldsymbol{\Psi}_1\boldsymbol{\Pi}\tilde{\boldsymbol{n}}_K$ is

$$\bar{\boldsymbol{Q}}_K = \boldsymbol{\Psi}_1\boldsymbol{\Pi}\tilde{\boldsymbol{Q}}_K\boldsymbol{\Pi}^{\mathsf{T}}\boldsymbol{\Psi}_1^{\mathsf{H}}. \tag{50}$$

Finally, we multiply the reduced measurements in (49) with the prewhitening matrix $\bar{\boldsymbol{Q}}_K^{-\frac{1}{2}}$ to make the noise uncorrelated. Hence, the new system of equations is

$$\bar{\boldsymbol{y}}_K = \bar{\boldsymbol{Q}}_K^{-\frac{1}{2}}\boldsymbol{\Psi}_1\boldsymbol{\Pi}\tilde{\boldsymbol{y}}_K = \bar{\boldsymbol{\Gamma}}_K\tilde{\boldsymbol{u}}_K + \bar{\boldsymbol{n}}_K, \tag{51}$$

Algorithm 6 Basis Pursuit Robust Kalman Filtering

Inputs: $\{\boldsymbol{y}_k, \boldsymbol{A}_k, \boldsymbol{B}_k, \boldsymbol{C}_k, \boldsymbol{D}_k, \boldsymbol{Q}_k, \boldsymbol{R}_k\}_{k=1}^K$

Compute $\tilde{\boldsymbol{y}}_K, \boldsymbol{O}_K, \boldsymbol{M}_K, \boldsymbol{\Gamma}_K$ from (40), (41), (42) and $\boldsymbol{\Pi}$ using (44)

Determine $\boldsymbol{\Phi}_1$ using the singular value decomposition of $\boldsymbol{\Pi}\boldsymbol{\Gamma}_K$ as given in (47)

Compute $\bar{\boldsymbol{Q}}_K$ using (46) and (50)

Compute $\bar{\boldsymbol{y}}_K$ and $\bar{\boldsymbol{\Gamma}}_K$ from (51)

Set parameter $\epsilon = \sqrt{R\left(1+2\sqrt{\frac{2}{R}}\right)}$

Solve for inputs $\boldsymbol{U}_1^K$ using the convex optimization problem (52)

Calculate the initial state estimate $\boldsymbol{x}_1$ using (45)

Set $\boldsymbol{x}_{1|K} = \boldsymbol{x}_{1|1} = \boldsymbol{x}_1$

Kalman Smoother

for $k = 2, \ldots, K$ **do**

#Prediction:

$\hat{\boldsymbol{x}}_{k|k-1} = \boldsymbol{A}_{k-1}\hat{\boldsymbol{x}}_{k-1|k-1} + \boldsymbol{B}_k\boldsymbol{u}_{k-1}$

$\boldsymbol{P}^{\boldsymbol{x}}_{k|k-1} = \boldsymbol{A}_{k-1}\boldsymbol{P}^{\boldsymbol{x}}_{k-1|k-1}\boldsymbol{A}_{k-1}^{\mathsf{T}} + \boldsymbol{Q}_{k-1}$

#Filtering:

$\boldsymbol{G}_k = \boldsymbol{P}^{\boldsymbol{x}}_{k|k-1}\boldsymbol{C}_k^{\mathsf{T}}\left(\boldsymbol{R}_k + \boldsymbol{C}_k\boldsymbol{P}^{\boldsymbol{x}}_{k|k-1}\boldsymbol{C}_k^{\mathsf{T}}\right)^{-1}$

$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{G}_k\left(\boldsymbol{y}_k - \boldsymbol{D}_k\boldsymbol{u}_k - \boldsymbol{C}_k\hat{\boldsymbol{x}}_{k|k-1}\right)$

$\boldsymbol{P}^{\boldsymbol{x}}_{k|k} = \left(\boldsymbol{I} - \boldsymbol{G}_k\boldsymbol{C}_k\right)\boldsymbol{P}^{\boldsymbol{x}}_{k|k-1}$

end for

#Smoothing:

for $k = K-1, K-2, \ldots, 2$ **do**

$\boldsymbol{K}_k = \boldsymbol{P}^{\boldsymbol{x}}_{k|k}\boldsymbol{A}_k^{\mathsf{T}}\left(\boldsymbol{P}^{\boldsymbol{x}}_{k+1|k}\right)^{-1}$

$\boldsymbol{P}^{\boldsymbol{x}}_{k|K} = \boldsymbol{P}^{\boldsymbol{x}}_{k|k} + \boldsymbol{K}_k\left(\boldsymbol{P}^{\boldsymbol{x}}_{k+1|K} - \boldsymbol{P}^{\boldsymbol{x}}_{k+1|k}\right)\boldsymbol{K}_k^{\mathsf{T}}$

$\hat{\boldsymbol{x}}_{k|K} = \hat{\boldsymbol{x}}_{k|k} + \boldsymbol{K}_k\left(\hat{\boldsymbol{x}}_{k+1|K} - \boldsymbol{A}_k\hat{\boldsymbol{x}}_{k|k}\right) - \boldsymbol{P}^{\boldsymbol{x}}_{k|k}(\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{A}_k)\boldsymbol{A}_k^{\mathsf{T}}\boldsymbol{Q}_k^{-1}\boldsymbol{B}_k\boldsymbol{u}_k$

end for

Outputs: $\left\{\hat{\boldsymbol{x}}_{k|K}\right\}_{k=1}^K$ and $\{\boldsymbol{u}_k\}_{k=1}^K$

where matrix $\bar{\boldsymbol{\Gamma}}_K = \bar{\boldsymbol{Q}}_K^{-\frac{1}{2}}\boldsymbol{\Phi}_1\boldsymbol{\Pi}\boldsymbol{\Gamma}_K$ and the noise term $\bar{\boldsymbol{n}}_K = \bar{\boldsymbol{Q}}_K^{-\frac{1}{2}}\boldsymbol{\Phi}_1\boldsymbol{\Pi}\tilde{\boldsymbol{n}}_K$ follows the standard Gaussian distribution $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. Under a Laplacian prior on $\tilde{\boldsymbol{u}}_K$ to promote sparsity, the corresponding MAP estimate is obtained by solving a LASSO problem [46, Section 3.4.3] given by

$$\tilde{\boldsymbol{u}}_K^* = \arg\min_{\tilde{\boldsymbol{u}}_K} \|\tilde{\boldsymbol{u}}_K\|_1 \text{ s.t. } \|\bar{\boldsymbol{y}}_K - \bar{\boldsymbol{\Gamma}}_K\tilde{\boldsymbol{u}}_K\|_2 \leq \epsilon. \quad (52)$$

Here, $\epsilon > 0$ is typically chosen as $\sqrt{\text{var}(\bar{\boldsymbol{n}}_K)R}\sqrt{1+2\sqrt{\frac{2}{R}}}$ which is slightly larger than $\sqrt{\text{var}(\bar{\boldsymbol{n}}_K)R}$ (in our case $\text{var}(\bar{\boldsymbol{n}}_K) = 1$) [47].

In the second step, we compute the state estimates using the optimal solution $\tilde{\boldsymbol{u}}_K^*$ obtained by solving (52). Treating these inputs as the true ones, $\boldsymbol{u}_k = \boldsymbol{u}_k^*$, we compute the MAP estimate of the states $\hat{\boldsymbol{x}}_k$ using the Kalman smoothing algorithm applied to the system (1) and (2). The resulting BP-RKS algorithm is summarized in Algorithm 6.

When control inputs share a common support, we can rearrange $\tilde{\boldsymbol{u}}_K$ into a block-sparse vector $\hat{\boldsymbol{u}}_K = \left[\boldsymbol{u}(1,:)^{\mathsf{T}} \quad \boldsymbol{u}(2,:)^{\mathsf{T}} \ldots \boldsymbol{u}(m,:)^{\mathsf{T}}\right]^{\mathsf{T}} \in \mathbb{R}^{Km}$ with block length K, where $u(i,:)$ as defined in (26). The corresponding columns of $\bar{\boldsymbol{\Gamma}}_K$ are also rearranged, which we denote as $\hat{\boldsymbol{\Gamma}}_K$, leading to the system of equations $\bar{\boldsymbol{y}}_K = \hat{\boldsymbol{\Gamma}}_K\hat{\boldsymbol{u}}_K + \bar{\boldsymbol{n}}_K$. We can now exploit the block sparsity structure by imposing an ℓ_1/ℓ_2 type penalty [48] on $\hat{\boldsymbol{u}}_K$ to arrive at the following optimization problem:

$$\hat{\boldsymbol{u}}_K^* = \arg\min_{\hat{\boldsymbol{u}}_K} \sum_{i=1}^m \|\boldsymbol{u}(i,:)\|_2 \text{ s.t. } \|\bar{\boldsymbol{y}}_K - \hat{\boldsymbol{\Gamma}}_K\hat{\boldsymbol{u}}_K\|^2 \leq \epsilon. \quad (53)$$

The problem (53) can be solved by rewriting it as a second-order cone program using an auxiliary variable $\boldsymbol{t}$ as

$$\min_{\substack{\boldsymbol{t}\in\mathbb{R}^m \\ \hat{\boldsymbol{u}}_K\in\mathbb{R}^{Km}}} \sum_{i=1}^m \boldsymbol{t}_i \text{ s.t. } \boldsymbol{t}_i \geq \|\boldsymbol{u}(i,:)\|_2 \text{ for } i = 1, \ldots, m \text{ and } \|\bar{\boldsymbol{y}}_K - \hat{\boldsymbol{\Gamma}}_K\hat{\boldsymbol{u}}_K\|_2^2 \leq \epsilon. \quad (54)$$

The resulting group BP-RKS algorithm is identical to Algorithm 6 except that in Step 6 we solve (54) instead of (52).

REFERENCES

[1] R. K. Chakraborty, G. Joseph, and C. R. Murthy, "Bayesian learning-based kalman smoothing for linear dynamical systems with unknown sparse inputs," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2024, pp. 13 431–13 435.

[2] A. Olshevsky, "Minimal controllability problems," *IEEE Trans. Control Network Syst.*, vol. 1, no. 3, pp. 249–258, Sep. 2014.

[3] M. Siami, A. Olshevsky, and A. Jadbabaie, "Deterministic and randomized actuator scheduling with guaranteed performance bounds," *IEEE Trans. Autom. Control*, vol. 66, no. 4, pp. 1686–1701, Jun. 2020.

[4] G. Joseph and C. R. Murthy, "Controllability of linear dynamical systems under input sparsity constraints," *IEEE Trans. Autom. Control*, vol. 66, no. 2, pp. 924–931, Apr. 2020.

[5] G. Joseph, "Controllability of a linear system with nonnegative sparse controls," *IEEE Trans. Autom. Control*, vol. 67, no. 1, pp. 468–473, May 2021.

[6] N. Wendt, C. Dhal, and S. Roy, "Control of network opinion dynamics by a selfish agent with limited visibility," *IFAC-PapersOnLine*, vol. 52, no. 3, pp. 37–42, 2019.

[7] G. Joseph, B. Nettasinghe, V. Krishnamurthy, and P. K. Varshney, "Controllability of network opinion in Erdös-Rényi graphs using sparse control inputs," *SIAM J. Control Optim.*, vol. 59, no. 3, pp. 2321–2345, 2021.

[8] M. Raptis, K. Wnuk, and S. Soatto, "Spike train driven dynamical models for human actions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 2077–2084.

[9] H. L. Taylor, S. C. Banks, and J. F. McCoy, "Deconvolution with the ℓ_1 norm," *Geophys.*, vol. 44, no. 1, pp. 39–52, Jan. 1979.

[10] M. S. O'Brien, A. N. Sinclair, and S. M. Kramer, "Recovery of a sparse spike time series by L_1 deconvolution," *IEEE Trans. Signal Process.*, vol. 42, no. 12, pp. 3353–3365, 1994.

[11] A. A. Cárdenas, S. Amin, and S. Sastry, "Research challenges for the security of control systems," *HotSec*, vol. 5, p. 15, Jul. 2008.

[12] J. Slay and M. Miller, "Lessons learned from the Maroochy water breach," in *Proc. Int. Conf. Crit. Infrastruct. Prot.*, Mar. 2007, pp. 73–82.

[13] R. Ma, P. Shi, and L. Wu, "Sparse false injection attacks reconstruction via descriptor sliding mode observers," *IEEE Trans. Autom. Control*, vol. 66, no. 11, pp. 5369–5376, 2021.

[14] H. Yang and R. Ma, "Sparse attack reconstruction for cyber–physical systems via descriptor reduced-order observer," *Nonlinear Anal.: Hybrid Syst.*, vol. 50, p. 101380, 2023.
[15] Z. Zhao, Y. Xu, Y. Li, Y. Zhao, B. Wang, and G. Wen, "Sparse actuator attack detection and identification: A data-driven approach," *IEEE Trans. Cybern.*, vol. 53, no. 6, pp. 4054–4064, 2023.
[16] L. An and G.-H. Yang, "Secure state estimation against sparse sensor attacks with adaptive switching mechanism," *IEEE Trans. Autom. Control*, vol. 63, no. 8, pp. 2596–2603, 2018.
[17] B. Friedland, "Treatment of bias in recursive filtering," *IEEE Trans. Autom. Control*, vol. 14, no. 4, pp. 359–367, Aug. 1969.
[18] P. K. Kitanidis, "Unbiased minimum-variance linear state estimation," *Automatica*, vol. 23, no. 6, pp. 775–778, Nov. 1987.
[19] S. Gillijns and B. De Moor, "Unbiased minimum-variance input and state estimation for linear discrete-time systems," *Automatica*, vol. 43, no. 1, pp. 111–116, Jan. 2007.
[20] ——, "Unbiased minimum-variance input and state estimation for linear discrete-time systems with direct feedthrough," *Automatica*, vol. 43, no. 5, pp. 934–937, May 2007.
[21] G. Gakis and M. C. Smith, "A limit Kalman filter and smoother for systems with unknown inputs," *Int. J. Control*, vol. 97, no. 3, pp. 532–542, 2024.
[22] S. Foucart and H. Rauhut, *A Mathematical Introduction to Compressive Sensing*. Birkhäuser, 2013.
[23] G. Joseph and C. R. Murthy, "On the observability of a linear system with a sparse initial state," *IEEE Signal Process. Lett.*, vol. 25, no. 7, pp. 994–998, May 2018.
[24] ——, "Measurement bounds for observability of linear dynamical systems under sparsity constraints," *IEEE Trans. Signal Process.*, vol. 67, no. 8, pp. 1992–2006, 2019.
[25] W. Dai and S. Yüksel, "Observability of a linear system under sparsity constraints," *IEEE Trans. on Autom. Control*, vol. 58, no. 9, pp. 2372–2376, Mar. 2013.
[26] M. B. Wakin, B. M. Sanandaji, and T. L. Vincent, "On the observability of linear systems from random, compressive measurements," in *Proc. IEEE Conf. Decis. Control*, Dec. 2010, pp. 4447–4454.
[27] B. M. Sanandaji, M. B. Wakin, and T. L. Vincent, "Observability with random observations," *IEEE Trans. Autom. Control*, vol. 59, no. 11, pp. 3002–3007, Aug. 2014.
[28] N. Vaswani, "Kalman filtered compressed sensing," in *Proc. IEEE Int. Conf. Image Process.*, Oct. 2008, pp. 893–896.
[29] R. Prasad, C. R. Murthy, and B. D. Rao, "Joint approximately sparse channel estimation and data detection in OFDM systems using sparse Bayesian learning," *IEEE Trans. Signal Process.*, vol. 62, no. 14, pp. 3591–3603, Jun. 2014.
[30] D. Angelosante, G. B. Giannakis, and E. Grossi, "Compressed sensing of time-varying signals," in *Proc. IEEE Int. Conf. Digit. Signal Process.*, 2009, pp. 1–8.
[31] A. S. Charles, A. Balavoine, and C. J. Rozell, "Dynamic filtering of time-varying sparse signals via ℓ_1 minimization," *IEEE Trans. Signal Process.*, vol. 64, no. 21, pp. 5644–5656, Jul. 2016.
[32] C. Kurisummoottil Thomas and D. Slock, "Gaussian variational Bayes Kalman filtering for dynamic sparse Bayesian learning," in *Proc. IEEE Intl. Conf. Time Ser. Forecast.*, Mar. 2018.
[33] M. R. O'Shaughnessy, M. A. Davenport, and C. J. Rozell, "Sparse Bayesian learning with dynamic filtering for inference of time-varying sparse signals," *IEEE Trans. Signal Process.*, vol. 68, pp. 388–403, 2020.
[34] S. Sefati, N. J. Cowan, and R. Vidal, "Linear systems with sparse inputs: Observability and input recovery," in *Proc. IEEE Am. Control Conf.*, 2015, pp. 5251–5257.
[35] K. Poe, E. Mallada, and R. Vidal, "Necessary and sufficient conditions for simultaneous state and input recovery of linear systems with sparse inputs by ℓ_1-minimization," in *Proc. IEEE Conf. Decis. Control*, 2023, pp. 6499–6506.
[36] ——, "Invertibility of discrete-time linear systems with sparse inputs," in *Proc. IEEE Conf. Decis. Control*, 2024, pp. 6095–6101.
[37] D. Wipf and S. Nagarajan, "Iterative reweighted ℓ_1 and ℓ_2 methods for finding sparse solutions," *IEEE J. Sel. Top. Signal Process.*, vol. 4, no. 2, pp. 317–329, Feb. 2010.
[38] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2010.
[39] H. Zou, "The adaptive lasso and its oracle properties," *J. Am. Stat. Assoc.*, vol. 101, no. 476, pp. 1418–1429, 2006.
[40] D. Angelosante, S. I. Roumeliotis, and G. B. Giannakis, "Lasso-Kalman smoother for tracking sparse signals," in *Proc. Asilomar*, Nov. 2009, pp. 181–185.
[41] G. McLachlan and T. Krishnan, *The EM algorithm and extensions*, 2nd ed., ser. Wiley series in probability and statistics. Hoboken, NJ: Wiley, 2008.
[42] D. Wipf and B. Rao, "Sparse Bayesian learning for basis selection," *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2153–2164, 2004.
[43] D. Nguyen, "An in-depth introduction to variational Bayes note," *SSRN*, 2023.
[44] R. K. Chakraborty, G. Joseph, and C. R. Murthy, "Joint state and input estimation for linear dynamical systems with sparse control," 2023. [Online]. Available: https://arxiv.org/abs/2312.02082
[45] M. Y. Byron, K. V. Shenoy, and M. Sahani, "Derivation of Kalman filtering and smoothing equations," in *Technical report*. Stanford University, 2004.
[46] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed. New York, NY: Springer, 2009.
[47] E. J. Candès and J. Romberg, "ℓ_1-MAGIC: Recovery of sparse signals via convex programming," 2005, MATLAB routines package.
[48] Y. C. Eldar and M. Mishali, "Robust recovery of signals from a structured union of subspaces," *IEEE Trans. Inf. Theory*, vol. 55, no. 11, pp. 5302–5316, 2009.

Rupam Kalyan Chakraborty received a B.Tech. in Electronics and Communication Engineering from West Bengal University of Technology in 2016, and M.Tech. degrees in Microwave Engineering from the University of Calcutta and Signal Processing from the Indian Institute of Science, Bangalore in 2018 and 2021, respectively. From 2021–2023, he worked as an Algorithm Design Engineer at Signalchip Innovations on GNSS tracking. Currently, he is a Ph.D. student in the signal processing systems group at the Delft University of Technology, Netherlands. His research interests include statistical signal processing and compressive sensing.

Geethu Joseph received the Ph.D. degree in electrical communication engineering from the Indian Institute of Science, Bangalore, in 2019. She is currently an assistant professor in the signal processing systems group at the Delft University of Technology, Delft, Netherlands. Her research interests include statistical signal processing, network control, and machine learning.

Chandra R. Murthy (S'03–M'06–SM'11–F'23) received Ph.D. degree in Electrical and Computer Engineering from UC San Diego in 2006. He worked at Qualcomm (2000–2002) on WCDMA and 802.11b receivers, and at Beceem Communications (2006–2007) on 802.16e WiMAX. Since 2007, he has been with the Department of Electrical Communication Engineering at IISc Bangalore, where he is now a Professor. His research interests are in the areas of energy harvesting communications, 5G/6G technologies and compressed sensing. He is a recipient of the MeitY Young Faculty Fellowship from the Govt. of India and the Prof. Satish Dhawan state award for engineering from the Karnataka State Government. He is a fellow of the IEEE.