2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

An Efficient Implementation for Bayesian Manifold Regularization Method

Junpeng Zhang, Yue Ju, Bo Wahlberg, Biqiang Mu and Tianshi Chen

Abstract— **When applying the Bayesian manifold regularization method to function estimation problem with manifold constraints, the direct implementation has computational complexity $\mathcal{O}(N^3)$, where N is the number of input-output data measurements. This becomes particularly costly when N is large. In this paper, we propose a more efficient implementation based on the Kalman filter and smoother using a state-space model realization of the underlying Gaussian process. Moreover, we explore the sequentially semi-separable structure of the Laplacian matrix and the posterior covariance matrix. Our proposed implementation has computational complexity $\mathcal{O}(N)$ and thus can be applied to large data problems. We exemplify the effectiveness of our proposed implementation through numerical simulations.**

Index Terms— **Bayesian manifold regularization, Kalman filter and smoother, Sequentially semi-separable matrix**

I. INTRODUCTION

In the past decade, kernel-based regularization (KRM) has attracted increasing attention and become a complement to the prediction error/maximum likelihood (PE/ML) method (see, e.g., [1]–[3]). There are two key issues in KRM: one is the kernel design, which determines the model structure and embeds the prior knowledge of the function to be estimated; the other one is the estimation of the parameter in the kernel, often known as the hyper-parameter estimation, which determines the model complexity. Many interesting results have been obtained for KRM, e.g., the regularization design [4]–[10] and the efficient implementation [11]–[13].

This work was supported by NSFC under contract No. 62273287, by the Shenzhen Science and Technology Innovation Council under contract No. JCYJ20220530143418040 and JCY20170411102101881, the Thousand Youth Talents Plan funded by the central government of China, and the Strategic Priority Research Program of Chinese Academy of Sciences under Grant No. XDA27000000, the Wallenberg AI, Autonomous Systems and Software Program (WASP), and the Swedish Research Council Research Environment NewLEADS under contract 2016-06079.

Junpeng Zhang and Tianshi Chen are with the School of Data Science and Shenzhen Research Institute of Big Data, The Chinese University of Hong Kong, Shenzhen 518172, China junpengzhang@link.cuhk.edu.cn, tschen@cuhk.edu.cn

Yue Ju was with the School of Data Science and Shenzhen Research Institute of Big Data, The Chinese University of Hong Kong, Shenzhen 518172, China. She is now with the Division of Decision and Control Systems, School of Electrical Engineering and Computer Science, KTH Royal Instituite of Technology, Brinellvägen 8, SE-10044, Stockholm, Sweden yuej@kth.se

Bo Wahlberg is with the Division of Decision and Control Systems, School of Electrical Engineering and Computer Science, KTH Royal Instituite of Technology, Brinellvägen 8, SE-10044, Stockholm, Sweden bo@kth.se

Biqiang Mu is with Key Laboratory of Systems and Control, Institute of Systems Science, Academy of Mathematics and System Science, Chinese Academy of Sciences, Beijing 100190, China bqmu@amss.ac.cn

One of these results is the Bayesian manifold regularization method proposed in [10]. It provides a Bayesian interpretation of the manifold regularization method [4], [7] and incorporates the prior knowledge of, among others, local smoothness of the function to be estimated, i.e., the more closely the inputs lie on the manifold, the closer the corresponding outputs. Unfortunately, since the direct implementation of hyper-parameter estimation and the corresponding function estimation in [10] has computational complexity $\mathcal{O}(N^3)$, where N is the number of input-output data points, such implementation is expensive to be applied to data with large N.

In this paper, we will address this problem and focus on developing an efficient implementation for the Bayesian manifold regularization method, when applied to a function estimation problem with manifold constraint. We will first propose to use a simulation-induced (SI) kernel such that its corresponding zero-mean Gaussian process has a state-space model realization, and then transform the function estimation problem into a Kalman filtering and smoothing problem. Moreover, we exploit the sequentially semi-separable (SSS) structures of the Laplacian matrix and the posterior covariance matrix to further reduce the computational complexity of hyper-parameter and function estimation. The primary contribution of this paper is the introduction of a proposed implementation with a computational complexity of $\mathcal{O}(N)$ and its efficiency is illustrated by numerical simulations.

The remaining parts of this paper are organized as follows. In Section II, we introduce some preliminaries and then the problem statement. In Section III, we propose an efficient implementation with computational complexity $\mathcal{O}(N)$. In Section IV, we present numerical simulations to demonstrate the efficiency of our proposed implementation. In Section V, we give the conclusion of this paper.

II. PRELIMINARIES AND PROBLEM STATEMENT

In this section, we first present some background materials and then the problem statement of this paper.

A. Function Estimation Problem

We consider the function estimation problem as follows,

$$y_j = f(t_j) + v_j,\ t_j = jT_s,\ j = 1, \cdots, N, \tag{1}$$

where $t_j \in \mathbb{R}_+ = \{x | x \geq 0, x \in \mathbb{R}\}$ is the jth time instant, $T_s > 0$ is the sampling interval, $f(t_j)$, v_j, $y_j \in \mathbb{R}$ are the unknown function, the measurement noise and the measurement output at the jth time instant, respectively,

v_j is assumed to be independent and identically Gaussian distributed (i.i.d.) with zero mean and variance $\sigma^2 > 0$, i.e.,

$$v_j \sim \mathcal{N}(0, \sigma^2), \tag{2}$$

and N is the number of measurements. We assume that the function $f : \mathbb{R}_+ \to \mathbb{R}$ is locally smooth, i.e., f satisfies the following widely used assumption (e.g., [4], [10]).

Assumption 1: For $i, j = 1, \cdots, N$, if t_i and t_j are close over the manifold where they lie on, then so are y_i and y_j.

To encode the local smoothness of f mentioned in Assumption 1, one possible way is to define the manifold constraint of f by using its weighted gradient operator (see, e.g., [10]). More specifically, we first let $\omega_{j,j+1} \geq 0$ measure the smoothness between $f(t_j)$ and $f(t_{j+1})$ for $j = 1, \cdots, N-1$, where the larger $\omega_{j,j+1}$, the closer $f(t_j)$ and $f(t_{j+1})$, and $\omega_{j,j+1} = 0$ denotes that t_j and t_{j+1} are lie on different manifolds. For convenience, we define a new set $\{j|\omega_{j,j+1} > 0\}$ with size N_Z, which can be reindexed as $\{\mathrm{a}_1 < \cdots < \mathrm{a}_{\mathrm{N_Z}}\}$. Then for $i = 1, \cdots, N_Z$ and $j = 1, \cdots, N$, we define the (i, j)th element of the weight matrix $L \in \mathbb{R}^{N_Z \times N}$ as

$$[L]_{i,j} = \begin{cases} \sqrt{\omega_{\mathrm{a_i},\mathrm{a_i}+1}}, & \text{if } j = \mathrm{a_i}, \\ -\sqrt{\omega_{\mathrm{a_i},\mathrm{a_i}+1}}, & \text{if } j = \mathrm{a_i} + 1, \\ 0, & \text{otherwise,} \end{cases} \tag{3}$$

where $\omega_{\mathrm{a_i},\mathrm{a_i}+1}$ is often parameterized by $\sigma_\omega \in \mathbb{R}^{d_\omega}$. Then $L(\omega_{\mathrm{a_i},\mathrm{a_i}+1}(\sigma_\omega))$ can be regarded as a weighted gradient operator on f to describe its smoothness (one example is given in [10, Example 1]) and we often write $L(\omega_{\mathrm{a_i},\mathrm{a_i}+1}(\sigma_\omega))$ as L for simplicity. Similar to [10], [14], the smoothness constraint of f takes the following form,

$$Lf_E = 0 \in \mathbb{R}^{N_Z} \tag{4}$$

with $f_E = [f(t_1), \cdots, f(t_N)]^T$, which is also known as the manifold constraint.

By rewriting (1) in compact form and adding pseudo measurement noise $D \in \mathbb{R}^{N_Z}$ to (4), we have

$$Y = f_E + V, \tag{5a}$$

$$Z = Lf_E + D, \tag{5b}$$

where $Y = [y_1, \cdots, y_N]^T$, $V = [v_1, \cdots, v_N]^T$, $Z \in \mathbb{R}^{N_Z}$ is often known as the pseudo observation, and D is assumed to be

$$D \sim \mathcal{N}(0, \eta^2 I_{N_Z}) \text{ with } \eta^2 > 0, \tag{6}$$

and independent of V, and I_{N_Z} denotes an N_Z-dimensional identity matrix.

For the function estimation problem, our objective is to estimate the function f with soft manifold constraint [10] $Z = 0$ based on the input-output data $\{t_j, y_j\}_{j=1}^N$ as well as possible.

B. Bayesian Manifold Regularization Method

To estimate the function f with $Z = 0$, we apply the Bayesian manifold regularization method proposed in [10]. Here, the function $f(t_j)$ is modeled as a Gaussian process:

$$f(t_j) \sim \mathcal{GP}\left(0, k(t_j, t_{j'}; \alpha)\right), \tag{7}$$

$$k(t_j, t_{j'}; \alpha) = \mathbb{E}\left[f(t_j)f(t_{j'})^T\right], \tag{8}$$

where $j, j' = 1, \cdots, N$, $\mathcal{GP}\left(0, k(t_j, t_{j'}; \alpha)\right)$ represents a Gaussian process with zero mean and covariance function $k(t_j, t_{j'}; \alpha)$, $\mathbb{E}(\cdot)$ denotes the mathematical expectation, $k(t_j, t_{j'}; \alpha) : \mathbb{R}_+ \times \mathbb{R}_+ \to \mathbb{R}$ the covariance function (also called the kernel), $\alpha \in \Omega \subset \mathbb{R}^d$ with $d \in \mathbb{N}$ the hyper-parameters of $k(t_j, t_{j'}; \alpha)$. Moreover, it is assumed that for any $j = 1, \cdots, N$, $f(t_j)$ is independent of V and D in (5).

For simplicity, we define $X = \{t_j\}_{j=1}^N$. Based on the soft manifold constraint $Z = 0$, it follows from [15, p. 16] and [10, Proposition 1, Theorem 2] that

$$f_E|Y, X \sim \mathcal{N}\left(\hat{f}_E, \Sigma_{f_E}\right), \tag{9a}$$

$$Z|Y, X \sim \mathcal{N}(\hat{f}_Z, \Sigma_Z), \tag{9b}$$

$$f_E|Y, Z = 0, X \sim \mathcal{N}(\hat{f}_M, \Sigma_{f_M}), \tag{9c}$$

where

$$\hat{f}_E = K(K + \sigma^2 I_N)^{-1} Y, \tag{10a}$$

$$\Sigma_{f_E} = K - K(K + \sigma^2 I_N)^{-1} K, \tag{10b}$$

$$\hat{f}_Z = L\hat{f}_E, \tag{10c}$$

$$\Sigma_Z = L\Sigma_{f_E} L^T + \eta^2 I_{N_Z}, \tag{10d}$$

$$\hat{f}_M = K(K + \sigma^2 I_N + (\sigma^2/\eta^2) MK)^{-1} Y, \tag{10e}$$

and moreover, $K \in \mathbb{R}^{N \times N}$ and for $j, j' = 1, \cdots, N$, its (j, j')th element is

$$[K]_{j,j'} = k(t_j, t_{j'}; \alpha), \tag{11}$$

and $M = L^T L$ is the Laplacian matrix. Note that Σ_{f_M} is not required in this paper and its expression is omitted here.

C. Kernel Design and Hyper-parameter Estimation

There are two main concerns in the Bayesian manifold regularization method: kernel design and hyper-parameter estimation.

The design of kernel $k(t_j, t_{j'}; \alpha)$ in (8) determines the model structure and should take into consideration at least two issues: one is to embed the prior knowledge of the function to be estimated, the other one is to ease the computational complexity of (10e) and the hyper-parameter estimation to be introduced shortly. For instance, *for KRM without manifold constraint*, the simulation-induced (SI) kernel is introduced in [6] such that its corresponding zero-mean Gaussian process has a state-space model realization and then it is possible to reduce the computational complexity of (10e) and the hyper-parameter estimation by using Kalman filter and smoother.

The estimation of hyper-parameter α determines the model complexity and can be tackled using many methods, e.g., the marginal likelihood (ML) method. In this paper, we first treat σ^2 in (2), σ_ω mentioned after (3) and η^2 in (6) as additional hyper-parameters and assume the hyper-prior distribution,

$$p(\eta^2) = \frac{\beta_\Gamma^{\alpha_\Gamma}}{\Gamma(\alpha_\Gamma)} \cdot \frac{1}{\eta^{2(\alpha_\Gamma+1)}} \cdot \exp\left(-\frac{\beta_\Gamma}{\eta^2}\right), \tag{12a}$$

$$p([\alpha^T, \sigma^2, \sigma_\omega]^T|\eta^2) \propto \textit{const}, \tag{12b}$$

where $\alpha_\Gamma, \beta_\Gamma \in \mathbb{R}_+$, and *const* denotes a constant. Then we estimate the hyper-parameter $\alpha^{MR} = [\alpha^T, \sigma^2, \sigma_\omega, \eta^2]^T$,

$$\hat{\alpha}^{MR} = \underset{\alpha^{MR}\in\Omega^{MR}}{\arg\min} \left\{J_{ML} + \varrho J_{PML} + J_{HP}\right\}, \tag{13a}$$

where $\Omega^{MR} = \{\alpha^{MR} | \alpha \in \Omega, \sigma^2 > 0, \sigma_\omega \in \mathbb{R}^{d_\omega}, \eta^2 > 0\}$,

$$\begin{aligned} J_{ML} &= -\log p(Y|X;\alpha,\sigma^2) \qquad\qquad (13b)\\ &= \mathit{const} + \frac{1}{2}\log|K+\sigma^2 I_N| + \frac{1}{2}Y^T(K+\sigma^2 I_N)^{-1}Y,\\ J_{PML} &= -\log p(Z=0|Y,X;\sigma_\omega,\eta^2)\\ &= \mathit{const} + \frac{1}{2}\log|\Sigma_Z| + \frac{1}{2}\hat{f}_Z^T\Sigma_Z^{-1}\hat{f}_Z, \qquad (13c)\\ J_{HP} &= -\log p(\eta^2) = \mathit{const} + (\alpha_\Gamma+1)\log(\eta^2) + \frac{\beta_\Gamma}{\eta^2}, \qquad (13d) \end{aligned}$$

and moreover, $|\cdot|$ denotes the determinant of a square matrix, and $\varrho \in [0,1]$ controls the trade-off between J_{ML} and J_{PML}. When $\varrho = 1$, (13) is the same as the hyper-parameter estimation method proposed in [10, Section 3.2 (15)]; when $\varrho = 0$, (13) is reduced to the ML method together with the hyper-prior assumption (12).

D. Problem Statement

To state the problem, we first recall that the direct implementation of (10e) and (13) in [10] has computational complexity $\mathcal{O}(N^3)$ and will be expensive for large N. In this paper, our objective is to develop more efficient implementation of (10e) and (13) than [10] by using Kalman filtering and smoothing, and exploring the rank structure of Σ_{f_E} in (10b) and Σ_Z in (10d); our secondary focus is to obtain better estimates of f_E than [10] by tuning ϱ in (13a).

III. An Efficient Implementation

In this section, we propose an efficient implementation algorithm for the function estimation problem with computational complexity $\mathcal{O}(N)$. We will first design $k(t_j, t_{j'};\alpha)$ (8) as a stationary SI kernel and derive the state-space model realization of model (1). Then we rephrase the function estimation problem into a Kalman filtering and smoothing problem as stated in [13], [16]. Lastly, we explore the rank structure of Σ_{f_E} in (10b) and Σ_Z in (10d) to compute (13c) and (10e) with the computational complexity $\mathcal{O}(N)$.

A. State-space Model Realization of (1)

If $k(t_j, t_{j'};\alpha)$ (8) satisfies the following assumption, then (7) has a state-space model realization.

Assumption 2: $k(t_j, t_{j'};\alpha)$ (8) is a stationary SI kernel.

Recall that $k(t_j, t_{j'};\alpha)$ is the covariance function of $f(t_j)$ in (7). Then using the realization theory of linear systems [17], the discrete-time state-space model realization of (7) is given by

$$\begin{aligned} x_j &= Fx_{j-1} + Gw_{j-1}, x_0 \sim \mathcal{N}(0,\Sigma_0),\\ f(t_j) &= Hx_j, j = 1,\cdots, \end{aligned} \tag{14}$$

where $F \in \mathbb{R}^{r\times r}, G \in \mathbb{R}^r$ and $H \in \mathbb{R}^{1\times r}$ are the system matrix, the input matrix and the output matrix, respectively, $x_j \in \mathbb{R}^r$ is the state vector at the jth time instant, $w_j \in \mathbb{R}$ is i.i.d. Gaussian noise with zero mean and unit variance, w_j and $v_{j'}$ are independent for $j, j' = 1,\cdots,N$, and $\Sigma_0 \in \mathbb{R}^{r\times r}$ is the solution of the discrete-time Lyapunov equation $\Sigma_0 = F\Sigma_0F^T + GG^T$. Then, the state-space model realization of (1) can be accordingly rewritten as follows

$$x_j = Fx_{j-1} + Gw_{j-1}, x_0 \sim \mathcal{N}(0,\Sigma_0), \tag{15a}$$
$$y_j = Hx_j + v_j, j = 1,\cdots. \tag{15b}$$

Now, we can convert the function estimation problem into a Kalman filtering and smoothing problem based on (15).

B. Kalman filter and smoother

The direct implementation of $\hat{f}_E$ in (10a) and Σ_{f_E} in (10b) is $\mathcal{O}(N^3)$. To compute them more efficiently, we apply the Kalman filter and smoother.

For $j = 1, 2, \cdots, m = 0, 1, \cdots, N$, we let $\hat{x}_{j|m} \in \mathbb{R}$ and $\Sigma_{j|m} \in \mathbb{R}^{r\times r}$ denote

$$\hat{x}_{j|m} = \mathbb{E}[x_j|y_{0:m}], \tag{16a}$$
$$\Sigma_{j|m} = \mathbb{E}[(x_j - \hat{x}_{j|m})(x_j - \hat{x}_{j|m})^T|y_{0:m}], \tag{16b}$$

where $y_{0:m} = \{y_0, \cdots, y_m\}$ and y_0 is a null value.

Then we apply the Kalman filter as follows,

$$e_j = y_j - H\hat{x}_{j|j-1}, \tag{17a}$$
$$E_j = H\Sigma_{j|j-1}H^T + \sigma^2 I_M, \tag{17b}$$
$$\hat{x}_{j|j} = F\hat{x}_{j-1|j-1} + \Sigma_{j|j-1}H^TE_j^{-1}e_j, \tag{17c}$$
$$\Sigma_{j|j} = \Sigma_{j|j-1} - \Sigma_{j|j-1}H^TE_j^{-1}H\Sigma_{j|j-1}, \tag{17d}$$
$$\Sigma_{j+1|j} = F\Sigma_{j|j}F^T + GG^T,\ j = 1,\cdots,N, \tag{17e}$$

and the Kalman smoother as follows,

$$J_j = \Sigma_{j|j}F^T\Sigma_{j+1|j}^{-1}, \tag{18a}$$
$$\hat{x}_{j|N} = \hat{x}_{j|j} + J_j(\hat{x}_{j+1|N} - F\hat{x}_{j|j}), \tag{18b}$$
$$\Sigma_{j|N} = \Sigma_{j|j} + J_j(\Sigma_{j+1|N} - \Sigma_{j+1|j})J_j^T, \tag{18c}$$
$$\hat{f}_{j|N} = H\hat{x}_{j|N},\ j = N-1,\cdots,1. \tag{18d}$$

Note that for $j, k = 1,\cdots,N$, the jth element of $\hat{f}_E$ and the (j,k)th element of Σ_{f_E} can be obtained by

$$[\hat{f}_E]_j = \hat{f}_{j|N}, \tag{19a}$$

$$[\Sigma_{f_E}]_{j,k} = \begin{cases} H\Sigma_{j|N}H^T, & \text{if } j = k,\\ H\prod_{i=k}^{j-1}J_i\Sigma_{j|N}H^T, & \text{if } k < j,\\ H\Sigma_{k|N}\prod_{i=k-1}^{j}J_jH^T, & \text{if } j < k, \end{cases} \tag{19b}$$

which shows that the computations of $\hat{f}_E$ and $[\Sigma_{f_E}]_{j,k}$ become linear of N.

C. Recursive Implementation of Hyper-parameter Estimation

Since the computation of J_{HP} in (13d) only involves scalar operations, we mainly focus on the efficient implementation of J_{ML} in (13b) and J_{PML} in (13c). It is worth to note that the recursive computation of J_{ML} in (13b) has been studied in [13, Proposition 2] and its complexity is linear in N. Note that the implementation in [13, Proposition

2] is designed for the spatial-temporal case and can hence be reduced to the temporal case in this paper.

Given $\hat{f}_E$ and Σ_{f_E} in the form of (19), the direct implementation of J_{PML} has computational complexity $\mathcal{O}(N_Z N^2)$. To further reduce the computational complexity of J_{PML}, we explore the rank structure of Σ_{f_E} in (10b) and Σ_Z in (10d), and will show that they are both sequentially semi-separable (SSS) in what follows. The definition of the SSS matrix is given in the following.

Definition 1 ((1.1) in [18]): A block matrix $A \in \mathbb{R}^{m\times n}$ is said to be SSS if for $i, j = 1, \cdots, N$, it can be factorized as

$$A_{i,j} = \begin{cases} p_i q_j, & \text{if } i = j+1, \\ p_i a_{i-1}\cdots a_{j+1} q_j, & \text{if } j < i+1, \\ d_i, & \text{if } i = j, \\ g_i h_j, & \text{if } j = i+1, \\ g_i b_{i-1}\cdots b_{j+1} h_j, & \text{if } j > i+1, \end{cases} \tag{20}$$

where $A_{i,j} \in \mathbb{R}^{m_i\times n_j}$, $\sum_{i=1}^N m_i = m$, $\sum_{j=1}^N n_j = n$, $p_{i+1} \in \mathbb{R}^{m_{i+1}\times r'_i}$, $a_i \in \mathbb{R}^{r'_i\times r'_{i-1}}$ and $q_i \in \mathbb{R}^{r'_i\times n_i}$ are said to be the lower generators of A with order r'_i, $g_i \in \mathbb{R}^{m_i\times r''_i}$, $b_i \in \mathbb{R}^{r''_{i-1}\times r''_i}$ and $h_{i+1} \in \mathbb{R}^{r''_i\times n_{i+1}}$ are said to be the upper generators of A with order r''_i, and $d_i \in \mathbb{R}^{m_i\times n_i}$ are said to be the diagonal entry of A.

We can show that Σ_{f_E} in (10b) is SSS.

Proposition 1: Under Assumption 2, Σ_{f_E} in (10b) is SSS and the orders of its generators are r, where r is the order of the state-space model in (15). In particular, Σ_{f_E} can be put into the form of (20) with

$$d_i(\Sigma_{f_E}) = H\Sigma_{i|N}H^T \in \mathbb{R}, i = 1, \cdots, N, \tag{21a}$$
$$p_i(\Sigma_{f_E}) = H\Sigma_{i|N}J_{i-1}^T \in \mathbb{R}^{1\times r}, i = 2, \cdots, N, \tag{21b}$$
$$a_i(\Sigma_{f_E}) = J_{i-1}^T \in \mathbb{R}^{r\times r}, i = 2, \cdots, N-1, \tag{21c}$$
$$q_i(\Sigma_{f_E}) = H^T \in \mathbb{R}^r, i = 1, \cdots, N-1, \tag{21d}$$
$$g_i(\Sigma_{f_E}) = H \in \mathbb{R}^{1\times r}, i = 1, \cdots, N-1, \tag{21e}$$
$$b_i(\Sigma_{f_E}) = J_{i-1} \in \mathbb{R}^{r\times r}, i = 2, \cdots, N-1, \tag{21f}$$
$$h_i(\Sigma_{f_E}) = J_{i-1}\Sigma_{i|N}H^T \in \mathbb{R}^r, i = 2, \cdots, N, \tag{21g}$$

where $d_i(\cdot), p_i(\cdot), a_i(\cdot), q_i(\cdot), g_i(\cdot), h_i(\cdot), b_i(\cdot)$ are the ith diagonal entry and upper (lower) generators as defined in (20).

By using (21), we can prove that Σ_Z is also SSS.

Proposition 2: Under Assumption 2, Σ_Z in (10d) is symmetric and SSS and the orders of its generators are $4r$. The expressions of generators of Σ_Z are omitted due to the limitation of space.

Therefore, it is possible to exploit the SSS structure of Σ_Z in Proposition 2 to compute the cost function (13c) recursively in the following way.

Proposition 3: The cost function J_{PML} in (13c) can be computed by using

$$\Sigma_Z = V_Z U_Z S_Z, \tag{22a}$$
$$\log|\Sigma_Z| = \sum_{j=1}^{N_Z} \log|d_j(S_Z)|, \tag{22b}$$
$$\hat{f}_Z^T \Sigma_Z^{-1} \hat{f}_Z = \hat{f}_Z^T S_Z^{-1} U_Z^T V_Z^T \hat{f}_Z, \tag{22c}$$

where $V_Z \in \mathbb{R}^{N_Z\times N_Z}$ and $U_Z \in \mathbb{R}^{N_Z\times N_Z}$ are triangular orthogonal SSS matrices and $S_Z \in \mathbb{R}^{N_Z\times N_Z}$ is a upper triangular SSS matrix with diagonal elements $d_j(S_Z)$ for $j = 1, \cdots, N_Z$.

D. Recursive Implementation of function estimation

The direct implementation of $\hat{f}_M$ in (10e) is $\mathcal{O}(N^3)$. By exploring its SSS structure, we can also compute $\hat{f}_M$ more efficiently.

Proposition 4: The matrix $K + \sigma^2 I_N + (\sigma^2/\eta^2)MK$ contained in $\hat{f}_M$ (10e) is SSS.

Based on the SSS structure in Proposition 4, $\hat{f}_M$ can be computed efficiently in the following way.

Proposition 5: The function estimate $\hat{f}_M$ in (10e) can be computed by using

$$K + \sigma^2 I_N + (\sigma^2/\eta^2)MK = V_M U_M S_M, \tag{23a}$$
$$Y' = S_M^{-1} U_M^T V_M^T Y, \tag{23b}$$
$$K = L_1 + L_2, \tag{23c}$$
$$\hat{f}_M = L_1 Y' + L_2 Y', \tag{23d}$$

where $V_M \in \mathbb{R}^{N\times N}$ and $U_M \in \mathbb{R}^{N\times N}$ are triangular orthogonal SSS matrices, $S_M \in \mathbb{R}^{N\times N}$ is an upper triangular SSS matrix, and $L_1 \in \mathbb{R}^{N\times N}$ and $L_2 \in \mathbb{R}^{N\times N}$ are triangular semi-separable matrices.

E. Summary of the Efficient Implementation and its Computational Complexity Analysis

The proposed implementation in Sections III-C and III-D is summarized in Algorithm 1. To analyze its computational complexity, note that r, the orders of the generators of Σ_{f_E} defined in (10b), only depends on the structure of chosen kernel and is often much smaller than N, and thus we ignore the term r. Moreover, since $N_Z \leq N$, we let $N_Z = N$ for brevity in the analysis of computational complexity.

Theorem 1: The proposed implementation shown in Algorithm 1 has computational complexity $\mathcal{O}(N)$. In particular,

- the Kalman filter (17) and Kalman smoother (18) have computational complexity $\mathcal{O}(N)$;
- the evaluation of the cost functions in (13) has computational complexity $\mathcal{O}(N)$;
- the function estimation (23) has computational complexity $\mathcal{O}(N)$.

IV. NUMERICAL SIMULATIONS

In this section, we present numerical simulations to demonstrate the efficacy and efficiency of our proposed implementation.

A. Test Data Sets

We consider two test data sets: D1 and D2.

1) D1: We generate the noise-free output $f(t_j)$ in (1) as the piece-wise constant data mentioned in [20], which is defined as

$$f(t_j) = \begin{cases} 0, & t_j \in [0, \frac{1}{6}) \cup [\frac{2}{6}, \frac{3}{6}) \cup [\frac{4}{6}, \frac{5}{6}), \\ 1, & t_j \in [\frac{1}{6}, \frac{2}{6}), \\ 0.6, & t_j \in [\frac{3}{6}, \frac{4}{6}), \\ 0.4, & t_j \in [\frac{5}{6}, 1], \end{cases} \tag{24}$$

Algorithm 1 The Proposed Implementation

```
Input: data $\{t_j, y_j\}_{j=1}^N$, kernel $k(t_j, t_{j'}; \alpha)$ and its corresponding state-space model (15).
Output: $\hat{f}_M$.
  Step 1: Hyper-parameter Estimation
    • Kalman filter
        Calculate (17);
      Calculate $J_{ML}$ (13b) using [13, Proposition 2];
    • Kalman smoother
        Calculate (18);
      Calculate $J_{PML}$ (13c) by (21),
        (22a) using [18, Theorems 6.1 and 6.3],
        (22b), and (22c) using [18, Algorithm 6.4];
      Calculate $J_{HP}$ (13d);
  Step 2: Function estimation
      Calculate (23a) using [18, Theorems 6.1 and 6.3],
        (23b) using [18, Algorithm 6.4],
        (23c) using $L_1 = \texttt{tril}(K)$, $L_2 = \texttt{triu}(K, 1)$,
        and (23d) using [19, Algorithms C.1-C.2].
```

with $N = 500$ and $T_s = 1/(N-1) = 1/499$. Then $f(t_j)$ is corrupted with an additive measurement noise $v(t_j)$, which follows Gaussian distribution with zero mean and variance $\sigma^2 = 0.004, 0.005, 0.008, 0.010$, leading to 4 collections of measurement outputs. For each value of σ^2, we collect a data record with 500 pairs of input and measurement output, and then the above procedure is repeated 25 times. As a result, we obtain 4 data collections, each with 25 data records.

2) D2: We generate data sets using the same way as that for D1 but with $\sigma^2 = 0.008$ and $N = 500, 1000, 4000, 16000, 32000$.

B. Kernel Selection

For D1 and D2, we use the exponential kernel as introduced in [21], which is a SI kernel (see [6, Proposition 4.2]),

$$k(t_j, t_{j'}; \alpha) = \delta_t \rho^{|t_j - t_{j'}|}, \tag{25}$$

and its state-space model in the form of (15) with $F = \rho^{T_s}$, $G = 1$ and $H = \sqrt{\delta_t(1-\rho^{2T_s})}$, where $r = 1$, $\alpha = [\delta_t, \rho]^T$, $\delta_t > 0$ and $0 < \rho < 1$.

C. Hyper-parameter Estimation

For D1 and D2, when N is fixed, T_s is also fixed and then $\{t_j\}_{j=1}^N$ are evenly distributed. Thus we set $\omega_{j,j+1}(\sigma_\omega) = 1$ to simplify the computation and the hyper-parameter in (13a) is reduced to $\alpha^{MR} = [\alpha^T, \sigma^2, \eta^2]^T$. To estimate α^{MR}, we apply the hyper-parameter estimation method recommended in [10, Section 4] and our proposed one (13a) with different values of ϱ. In particular, for D1, we set $\varrho = 0.02, 0.05, 0.1, 0.5, 0.8, 1$; for D2, we set $\varrho = 0.02$. For both two types of methods, we set $\alpha_\Gamma = \beta_\Gamma = 5 \times 10^{-8}$. To solve the minimization problems in above hyper-parameter estimation methods, we use the `fmincon` function in Matlab. It is worth to note that for our proposed implementation, we use the same initial points as those for the recommended method in [10, Section 4], which makes their performance comparison fair.

D. Function Estimation

With the estimated hyper-parameter, we then compute the function estimation $\hat{f}_M$ in (10e). To evaluate the performance of the estimation $\hat{f}_M$, we use the measure of fit, e.g., [22],

$$\text{Fit} = 100 \times \left(1 - \frac{||\hat{f}_M - f_E||_2}{||f_E - \overline{f}||_2}\right), \overline{f} = \frac{1}{N}\sum_{j=1}^N f(t_j). \tag{26}$$

where $f_E = [f(t_1), \cdots, f(t_N)]^T$ is defined after (4). The maximum value of Fit is 100, indicating the perfect estimation of f_E.

E. Simulation Setup

For D1 and D2, we first apply the `kmeans` function in Matlab and divide $\{t_j, y_j\}_{j=1}^N$ into 6 clusters such that $\{t_j\}$ in each cluster are close over the manifold. As a result, we have $N_Z = N-6$. Then we apply the exponential kernel (25) and two types of hyper-parameter estimation methods: the recommended method in [10, Section 4], and our proposed one (13a), to estimate f_E.

Note that for D1, our main focus is to compare estimation performances of two types of hyper-parameter estimation methods; while for D2, our main focus is to compare the computing time of the cost function J_{PML} in (13c) and the function estimate $\hat{f}_M$ in (10e) by using the proposed implementation in Algorithm 1 and the direct implementation in [10].

F. Illustration of Estimation Performance

For D1, the average Fits (26) of $\hat{f}_M$ with hyper-parameter estimated by the recommended method in [10, Section 4] and our proposed one (13a) with different ϱ are shown in Table I. We can observe that for $\sigma^2 = 0.004, 0.005, 0.008, 0.01$,

- the average Fit obtained by our proposed hyper-parameter estimation method (13a) with $\varrho = 0.02$ is the largest, indicating that (13a) with $\varrho = 0.02$ performs best among all methods;
- the average Fits obtained by (13a) with $\varrho = 0.02, 0.05, 0.1$ are all larger than that by the method in [10, Section 4], while the average Fits obtained by (13a) with $\varrho = 0.5, 0.8, 1$ are all smaller than zero. It implies that the trade-off between J_{ML} in (13b) and J_{PML} in (13c) plays an important role in the estimation of hyper-parameter and also function. As long as we select a suitable ϱ, (13a) can outperform the method in [10, Section 4].

Moreover, one example of $\hat{f}_M$ with hyper-parameter estimated by our proposed method (13a) with $\varrho = 0.02$ and the recommended one in [10, Section 4] is given in Fig. 1, where $\hat{f}_M$ obtained by our proposed method can give a better estimate of f_E than the method in [10, Section 4].

G. Illustration of Computational Efficiency

For D2, the average computing time of the cost function (13c) and the function estimation (10e) by using our proposed implementation in Algorithm 1 and the direct implementation are shown in Table II. We can notice that as N increases, the average computing time of (13c) and (10e) using the direct implementation increases far more rapidly than those using our proposed implementation. In particular, when $N = 32000$, the average computing time of (13c) and (10e) using our proposed implementation is around 160 and 250 times faster, respectively, than those using the direct implementation.

TABLE I: The average Fits (26) for D1 by using the exponential kernel (25) and the hyper-parameter is estimated by the recommended method in [10, Section 4] and our proposed one (13a) with different ϱ. Note that we let the value of ϱ denote the method (13a) with ϱ for simplicity.

Method	$\sigma^2 = 0.004$	$\sigma^2 = 0.005$	$\sigma^2 = 0.008$	$\sigma^2 = 0.010$
[10, Sec. 4]	93.8944	92.9711	90.9012	88.7785
$\varrho = 0.02$	96.5807	96.0852	94.6166	92.4518
$\varrho = 0.05$	96.5379	96.0196	94.4620	92.2541
$\varrho = 0.1$	96.4959	95.9463	94.2553	90.4969
$\varrho = 0.5$	-0.0023	-0.0039	-0.0053	-0.0070
$\varrho = 0.8$	-0.0023	-0.0039	-0.0053	-0.0070
$\varrho = 1$	-0.0023	-0.0040	-0.0053	-0.0070

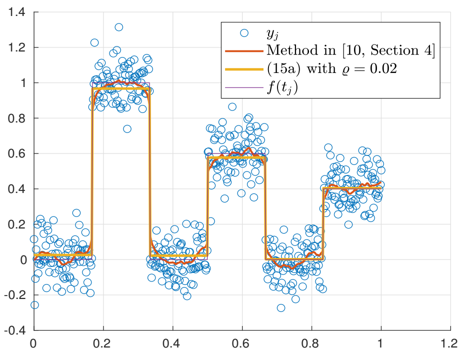

Fig. 1: One data record of D1 with $\sigma^2 = 0.010$ and the corresponding $\hat{f}_M$ with hyper-parameter estimated by using our proposed method (13a) with $\varrho = 0.02$ and the recommended one in [10, Section 4].

TABLE II: The average computing time (in seconds) for D2 by using the proposed implementation in Algorithm 1 and the direct one.

	Cost Function (13c)		Function Estimation (10e)	
N	Proposed	Direct	Proposed	Direct
500	0.0967	0.1684	0.0631	0.1732
1000	0.1866	0.6378	0.1269	0.7554
4000	0.7531	11.1638	0.5194	11.3680
16000	3.0165	201.8097	2.0921	213.3657
32000	5.9993	968.7789	4.2279	1.0647e+03

V. Conclusion

In this paper, we proposed an efficient implementation with computational complexity $\mathcal{O}(N)$ to tackle the function estimation problem by using the Bayesian manifold regularization method, where N is the number of input-output data points. We first converted the function estimation problem into the Kalman filtering and smoothing problem and then explored the sequentially semi-separable structures of the Laplacian matrix and the posterior covariance matrix. Our proposed implementation has been tested in numerical simulations and shown to be more efficient than the direct implementation. Note that our proposed implementation can be extended and applied to the the spatial-temporal function estimation and prediction problem, and will be studied in the journal version of this paper.

References

[1] G. Pillonetto, F. Dinuzzo, T. Chen, G. De Nicolao, and L. Ljung, "Kernel methods in system identification, machine learning and function estimation: A survey," *Automatica*, vol. 50, no. 3, pp. 657–682, 2014.

[2] A. Chiuso, "Regularization and Bayesian learning in dynamical systems: Past, present and future," *Annual Reviews in Control*, vol. 41, pp. 24 – 38, 2016.

[3] G. Pillonetto, T. Chen, A. Chiuso, G. De Nicolao, and L. Ljung, "Regularized system identification: Learning dynamic models from data," 2022.

[4] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples." *Journal of machine learning research*, vol. 7, no. 11, 2006.

[5] A. Marconato, M. Schoukens, and J. Schoukens, "Filter-based regularisation for impulse response modelling," *IET Control Theory & Applications*, vol. 11, no. 2, pp. 194–204, 2017.

[6] T. Chen, "On kernel design for regularized LTI system identification," *Automatica*, vol. 90, pp. 109–122, 2018.

[7] S. Formentin, M. Mazzoleni, M. Scandella, and F. Previdi, "Nonlinear system identification via data augmentation," *Systems & Control Letters*, vol. 128, pp. 56–63, 2019.

[8] Y. Xu, B. Mu, and T. Chen, "On kernel design for regularized volterra series model with application to wiener system identification," in *2022 41st Chinese Control Conference (CCC)*, 2022, pp. 1503–1508.

[9] X. Fang and T. Chen, "On kernel design for non-causal systems with application to feedforward control," in *2022 41st Chinese Control Conference (CCC)*, 2022, pp. 1523–1528.

[10] M. Mazzoleni, A. Chiuso, M. Scandella, S. Formentin, and F. Previdi, "Kernel-based system identification with manifold regularization: A bayesian perspective," *Automatica*, vol. 142, p. 110419, 2022.

[11] T. Chen and L. Ljung, "Implementation of algorithms for tuning parameters in regularized least squares problems in system identification," *Automatica*, vol. 49, no. 7, pp. 2213 – 2220, 2013.

[12] T. Chen and M. S. Andersen, "On semiseparable kernels and efficient implementation for regularized system identification and function estimation," *Automatica*, vol. 132, p. 109682, 2021.

[13] J. Zhang, Y. Ju, B. Mu, R. Zhong, and T. Chen, "An efficient implementation for spatial–temporal gaussian process regression and its applications," *Automatica*, vol. 147, p. 110679, 2023.

[14] M. Tahk and J. L. Speyer, "Target tracking problems subject to kinematic constraints," *IEEE transactions on automatic control*, vol. 35, no. 3, pp. 324–326, 1990.

[15] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006, vol. 2, no. 3.

[16] M. Todescato, A. Carron, R. Carli, G. Pillonetto, and L. Schenato, "Efficient spatio-temporal gaussian regression via kalman filtering," *Automatica*, vol. 118, p. 109032, 2020.

[17] C. Chen, *Linear system theory and design*, 3rd ed. New York: Oxford University Press, 1999.

[18] Y. Eidelman and I. Gohberg, "A modification of the dewilde–van der veen method for inversion of finite structured matrices," *Linear Algebra and its Applications*, vol. 343, pp. 419–450, 2002.

[19] M. S. Andersen and T. Chen, "Smoothing splines and rank structured matrices: Revisiting the spline kernel," *SIAM Journal on Matrix Analysis and Applications*, vol. 41, no. 2, pp. 389–412, 2020.

[20] Z. Zhao, M. Emzir, and S. Särkkä, "Deep state-space gaussian processes," *Statistics and Computing*, vol. 31, pp. 1–26, 2021.

[21] G. E. Uhlenbeck and L. S. Ornstein, "On the theory of the brownian motion," *Physical review*, vol. 36, no. 5, p. 823, 1930.

[22] L. Ljung, *System Identification Toolbox for use with* MATLAB*. Version 5.*, 5th ed. Natick, MA: The MathWorks, Inc, 2000.